การออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

นางสาวมฑุปายาส ทองมาก

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตร์มหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2545
ISBN 974-17-1051-8

DESIGN OF RULES FOR TRANSFORMING UML SEQUENCE DIAGRAMS INTO JAVA CODE

Miss Mathupayas Thongmak

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

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2002

ISBN 974-17-1051-8

หัวข้อวิทยานิพนธ์	การออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา
โดย	นางสาวมฑุปายาส ทองมาก
สาขา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร. วันชัย ริ้วไพบูลย์)

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

..... กรรมการ
(อาจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์)

..... กรรมการ
(อาจารย์ ชัยศิริ ปัณฑิตานนท์)

..... กรรมการ
(อาจารย์ นครทิพย์ พร้อมพูล)

มฑุปายาส ทองมาก : การออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา. (DESIGN OF RULES FOR TRANSFORMING UML SEQUENCE DIAGRAMS INTO JAVA CODE) อ. ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี; จำนวนหน้า 130 หน้า. ISBN 974-17-1051-8.

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา เพื่อสามารถนำกฎไปประยุกต์ใช้ในการสร้างเครื่องมือสำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาต่อไป การออกแบบกฎจะเริ่มจากการออกแบบยูเอ็มแอลเมต้าโมเดลเพื่อใช้ในการแปลงซีเควนซ์ไดอะแกรม แล้วจึงออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา 8 กฎ คือเมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย, เมต้ารูลสำหรับการแบ่งซีเควนซ์, เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง, เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร, เมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้, เมต้ารูลสำหรับการสร้างวัตถุใหม่, เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว และเมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเอง

หลังจากพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา ได้ทดลองแปลงซีเควนซ์ไดอะแกรมของเมทธอด 3 แผนภาพคือ ซีเควนซ์ไดอะแกรมของเมทธอดจองของระบบห้องพัก ซีเควนซ์ไดอะแกรมของเมทธอดคืนหนังสือของระบบห้องสมุด และซีเควนซ์ไดอะแกรมของเมทธอดแสดงของระบบกองไพ่ แล้วจึงคำนวณหาอัตราส่วนชุดคำสั่งที่สร้างได้ต่อชุดคำสั่งจริง โดยคิดเป็นร้อยละจากบรรทัดคำสั่งที่สร้างได้จากการประยุกต์ใช้กฎต่อบรรทัดคำสั่งจากชุดคำสั่งจริง ผลการคำนวณพบว่าเมทธอดจองของระบบห้องพักสามารถสร้างชุดคำสั่งได้ร้อยละ 81.25 เมทธอดคืนหนังสือของระบบห้องสมุดสามารถสร้างชุดคำสั่งได้ร้อยละ 71.43 และเมทธอดแสดงของระบบกองไพ่สามารถสร้างชุดคำสั่งได้ร้อยละ 93.3

ภาควิชาวิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา 2545

4370448421 : MAJOR COMPUTER SCIENCE

KEY WORD: TRANSFORMATION / UML / SEQUENCE DIAGRAMS / JAVA

MATHUPAYAS THONGMAK : DESIGN OF RULES FOR TRANSFORMING UML SEQUENCE DIAGRAMS INTO JAVA CODE. THESIS ADVISOR : ASSISTANT PROFESSOR PORNSIRI MUENCHAISRI, Ph.D. 130 pp. ISBN 974-17-1051-8.

This thesis objective was to design rules for transforming UML sequence diagrams into Java code. Using these rules, an automated tool to generate Java code from UML sequence diagrams was built. The design started by designing UML meta model for sequence diagrams, then designing rules for transforming UML sequence diagrams into Java code. Eight rules consist of meta rules for class diagram of a method that the sequence diagram depicts, meta rules for splitting of SEQUENCE, meta rules for conditional method invocation and branching, meta rules for assigning a value to a variable, meta rules for assigning object to pointer, meta rules for creating new object, meta rules for invoking a method of existing object, and meta rules for invoking a method of object itself.

With these rules, an automated tool was built to transform UML sequence diagrams into Java code. Three sequence diagrams were used as input. They are sequence diagrams that represent a method for making room reservation, a method for returning a book and a method for displaying a card pile. The percentage of generated source code per complete source code from method making room reservation is 81.25, from method returning a book is 71.43 and from method displaying a card pile is 93.3.

Department Computer Engineering Student's signature.....

Field of study Computer Science Advisor's signature.....

Academic year 2002

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือจาก ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี ขอขอบพระคุณที่ท่านได้ให้คำแนะนำ และข้อเสนอแนะต่างๆ ตลอดระยะเวลาของการจัดทำวิทยานิพนธ์ ขอบคุณเจ้าของชุดคำสั่งที่นำมาใช้ในการประเมินผล สุดท้ายนี้ขอกราบขอบพระคุณบิดา มารดา ขอบคุณพี่ชาย และเพื่อนๆ ทุกคนที่เป็นกำลังใจ และให้ความสนับสนุนมาโดยตลอด

มฑุปายาส ทองมาก

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

บทที่ หน้า

บทที่ หน้า

บทที่ หน้า

บทที่ หน้า

บทที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมา และความสำคัญของปัญหา

ปัจจุบันการพัฒนาซอฟต์แวร์ (Software) โดยใช้วิธีการเชิงวัตถุได้รับความนิยมเพิ่มมากขึ้น เนื่องจากการเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming) มีลักษณะคล้ายการจำลองเหตุการณ์ที่เกิดขึ้นจริงในโลกคือ มองทุกอย่างเป็นวัตถุที่มีคุณสมบัติ และพฤติกรรม (Behavior) ที่แตกต่างกันไป การเขียนโปรแกรมเชิงวัตถุอย่างมีประสิทธิภาพ ต้องอาศัยการออกแบบโปรแกรมที่ดี โดยใช้วิธีการออกแบบเชิงวัตถุ (Object-oriented design) ยูเอ็มแอล (UML – Unified Modeling Language) เป็นภาษาที่ใช้ในการสร้างแบบจำลองเชิงวัตถุ ซึ่งประกอบด้วยไดอะแกรมหลายชนิดที่สามารถจำลองการออกแบบโปรแกรมได้ ทั้งในเชิงโครงสร้าง (Structure) และเชิงพฤติกรรม (Behavior) ซีเควนซ์ไดอะแกรม (Sequence diagram) เป็นไดอะแกรมหนึ่งที่จำลองการออกแบบโปรแกรมในเชิงพฤติกรรม โดยจะแสดงการกระทำระหว่างกัน (Interaction) ระหว่างวัตถุ (Objects) ณ เวลาต่างๆ ด้วยการส่งสาร (Message) ไปมาระหว่างวัตถุ

เนื่องจากปัจจุบันความต้องการเครื่องมืออัตโนมัติ (Automated tools) ที่สามารถทำให้เกิด (Generate) โปรแกรมจากแบบจำลองมีมากขึ้น จึงมีงานวิจัยหลายงานที่ทำการวิจัยเกี่ยวกับการแปลงแบบจำลองยูเอ็มแอลเป็นโปรแกรม เช่น งานวิจัย XML Rule Based Source Code Generator for UML Case Tool [9] ที่นำเสนอเค้าโครง (Framework) สำหรับการสร้างเครื่องมือที่สามารถทำให้เกิดโปรแกรมจากยูเอ็มแอลคลาสไดอะแกรม งานวิจัย UML Collaboration Diagrams and Their Transformation to Java [3] ซึ่งนำเสนอกฎการแปลงยูเอ็มแอลคอแลบบอเรชันไดอะแกรมเป็นชุดคำสั่งภาษาจาวา และงานวิจัย Interaction Schemata: Compiling Interactions to Code [8] ซึ่งนำเสนอเครื่องมืออัตโนมัติที่ช่วยเขียนโปรแกรมจากยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยเครื่องมือนี้จะทำการแปลงซีเควนซ์ไดอะแกรมเป็นสกีมาตากลางที่แสดงถึงการกระทำระหว่างกัน (Interaction schemata) แล้วจึงแปลงสกีมาตากลางที่แสดงถึงการกระทำระหว่างกันที่ได้เป็นโปรแกรม

จากการศึกษาพบว่า ยังไม่มีงานวิจัยใดที่นำเสนอกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา งานวิจัยนี้จึงทำการออกแบบกฎการแปลงสำหรับยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยนำแนวคิดการออกแบบกฎการแปลงคอแลบบอเรชันไดอะแกรมเป็นโปรแกรมภาษาจาวา (Java) จากงานวิจัย [3] มาปรับใช้ เพื่อประโยชน์ในด้านความสามารถใน

การสร้างเครื่องมืออัตโนมัติสำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นโปรแกรมจากการประยุกต์ใช้กฎดังกล่าว

1.2 วัตถุประสงค์ของการวิจัย

เพื่อออกแบบกฎการแปลงสำหรับยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

1.3 ขอบเขตของการวิจัย

1) ข้อมูลนำเข้าที่นำกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวามาปรับใช้ต้องเป็นคลาสไดอะแกรม และซีเควนซ์ไดอะแกรมที่เขียนขึ้นอย่างถูกต้องตามวากยสัมพันธ์ ความหมายในทางสถิต (Static-semantically) และมีโครงสร้างที่ดี (Well-formed structure) ตรงตามมาตรฐานยูเอ็มแอล (UML standards)

2) ผลลัพธ์ของการปรับใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาจะได้เพียงบางส่วน (Fragments) ของโปรแกรมภาษาจาวาเท่านั้น จึงไม่สามารถนำโปรแกรมที่ได้ไปแปลโปรแกรม (Compile) และดำเนินงาน (Run) ได้

3) ในการทดสอบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาจะใช้ซีเควนซ์ไดอะแกรมซึ่งมีลักษณะตรงตามมาตรฐานยูเอ็มแอลอย่างน้อย 2 ไดอะแกรม ซึ่งจะต้องครอบคลุมถึงกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาทุกกฎที่ออกแบบเป็นข้อมูลนำเข้า

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้กฎใหม่สำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา ซึ่งกฎนี้จะเป็นเพียงอีกทางเลือกหนึ่งที่สามารถนำไปใช้สำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นโปรแกรม

2) สามารถนำกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาที่ได้ไปใช้สร้างเครื่องมือ สำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาต่อไป

1.5 ขั้นตอนการวิจัย

1) ศึกษาแนวคิดการออกแบบกฎการแปลงยูเอ็มแอลคอแลบบอเรชันไดอะแกรม

เป็นชุดคำสั่งภาษาจาวา

2) ออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

3) ทดสอบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

4) สรุปผลการวิจัย และข้อเสนอแนะ

5) จัดทำรายงานวิทยานิพนธ์

บทที่ 2
ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิด และทฤษฎี

แนวคิด และทฤษฎีที่เกี่ยวข้องกับการออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวามีดังนี้

2.1.1 การเขียนโปรแกรมเชิงวัตถุ [1]

โดยทั่วไปการมองระบบที่ซับซ้อนต่างๆ เราอาจมองเป็นระบบย่อยที่เกี่ยวข้องกัน หรือมองเป็นวัตถุที่มีความสัมพันธ์กัน การสร้างโปรแกรมที่ให้ผลลัพธ์ตรงตามระบบที่ออกแบบมาโดยตรงจะทำได้ยากกว่าการสร้างโปรแกรมที่จำลองการทำงานของแต่ละวัตถุในระบบนั้น แนวคิดในการสร้างโปรแกรมที่ประกอบขึ้นจากวัตถุเป็นหลักนี้เรียกว่า การเขียนโปรแกรมเชิงวัตถุ โดยภาษาที่มีกลไกสนับสนุนการสร้างวัตถุนี้เรียกว่า ภาษาเชิงวัตถุ (Object-oriented language) ตัวอย่างภาษาเชิงวัตถุที่เป็นที่รู้จักกันทั่วไป อาทิ เช่น ภาษาจาวา, ภาษาสมอลทอล์ค (Smalltalk) เป็นต้น

วัตถุในภาษาเชิงวัตถุจะหมายถึง สิ่งที่มีข้อมูลเป็นของตัวเอง มีความสามารถในการทำกิจกรรมบางอย่าง สามารถทำการเปลี่ยนแปลงข้อมูลของตัวเอง ติดต่อ หรือโต้ตอบกับวัตถุอื่นได้ ภาษาเชิงวัตถุจะมีคลาส (Class) เป็นโครงสร้างของโปรแกรมที่อาจมีสมาชิกเป็นข้อมูล หรือฟังก์ชัน (Function) (อาจเรียกฟังก์ชันในภาษาเชิงวัตถุว่าเมทธอด (Method)) ประกอบอยู่ โดยเมทธอดจะเป็นเสมือนกลไกในการกำหนดพฤติกรรม และรูปแบบการทำงานของวัตถุนั้น การติดต่อกันระหว่างวัตถุใดๆ จะอยู่ในรูปการเรียกใช้เมทธอดของวัตถุอื่น โดยปกติการเรียกใช้เมทธอดของวัตถุอื่นจำเป็นต้องสร้างกรณีตัวอย่าง (Instance) หรือตัวแทนของวัตถุที่จะถูกเรียกใช้เสียก่อน การเรียกใช้เมทธอดของวัตถุอื่นนี้ทำได้โดยการส่งสาร ซึ่งในหนึ่งสารจะประกอบด้วยข้อมูล 3 ส่วนคือ

1) ผู้ส่งสาร (Sender)
2) ผู้รับสาร (Receiver)
3) เมทธอดที่ถูกเรียกใช้ (Message selector)

จากตัวอย่างส่วนของโปรแกรมต่อไปนี้

```
class Greeting{
    void greet (){
```

```
            System.out.println("Hello! World.")
        }
    }
    class Hello{
        public static void main (String args[]){
            Greeting s = new Greeting();
            s.greet(); // message
        }
    }
```

จะพบว่าในคลาส Hello มีการสร้างออบเจ็กต์ของคลาส Greeting ชื่อ s และในบรรทัดถัดไปมีการส่งสารเพื่อเรียกใช้เมทธอดชื่อ greet ที่ประกาศไว้ในคลาส Greeting สารที่ส่งมีผู้ส่งสารคือ คลาส Hello ผู้รับสารคือ s เมทธอดที่ถูกเรียกใช้คือ greet

2.1.2 **ซีเควนซ์ไดอะแกรม** [2]

ซีเควนซ์ไดอะแกรมเป็นไดอะแกรมซึ่งใช้อธิบายพฤติกรรมเชิงพลวัต (Dynamic) ที่แสดงการติดต่อระหว่างกันของวัตถุ ณ เวลาต่างๆ โดยจะแสดงถึงลำดับของการส่งสาร และแสดงว่าสารมีการส่ง และรับกันระหว่างวัตถุอย่างไร ดังรูปที่ 2.4

ซีเควนซ์ไดอะแกรมมีแกนอยู่ 2 แกน

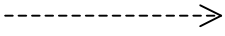
1) แกนตั้งแสดงเวลา
2) แกนนอนแสดงเซต (Set) ของวัตถุ

สำหรับสารที่ส่งระหว่างวัตถุนั้น มีอยู่หลายแบบดังนี้

1) สารแบบธรรมดาที่ส่งกันทั่วไป ใช้ลูกศรที่มีลักษณะดังรูปที่ 2.1

รูปที่ 2.1 แสดงลูกศรที่ใช้ในการส่งสารแบบธรรมดา

2) สารแสดงการย้อนกลับ (Return) ใช้ลูกศรที่มีลักษณะดังรูปที่ 2.2



รูปที่ 2.2 แสดงลูกศรที่ใช้ในการส่งสารแสดงการย้อนกลับ

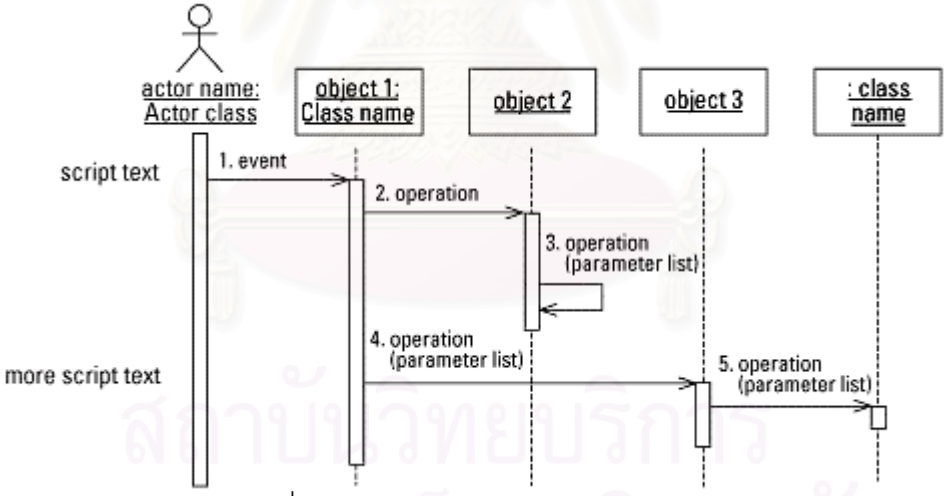
3) สารอสมวาร (Asynchronous message) การส่งสารแบบนี้ จะไม่หยุดการทำงานของผู้ส่ง ดังนั้นผู้ที่ส่งจะยังสามารถทำงานต่อได้ ใช้ลูกศรที่มีลักษณะดังรูปที่ 2.3

รูปที่ 2.3 แสดงลูกศรที่ใช้ในการส่งสารอสมวาร

การส่งสารแบบอสมวารมีประโยชน์คือ สามารถสร้างสายโยงใย (Thread) ใหม่ขึ้นมาทำงาน สร้างวัตถุใหม่ หรือสำหรับสื่อสารกับสายโยงใยที่กำลังทำงานอยู่

เส้นตามแนวตั้งเรียกว่า เส้นชีวิตของวัตถุ (Object's lifeline) ซึ่งแสดงชีวิตของวัตถุระหว่างการติดต่อกัน

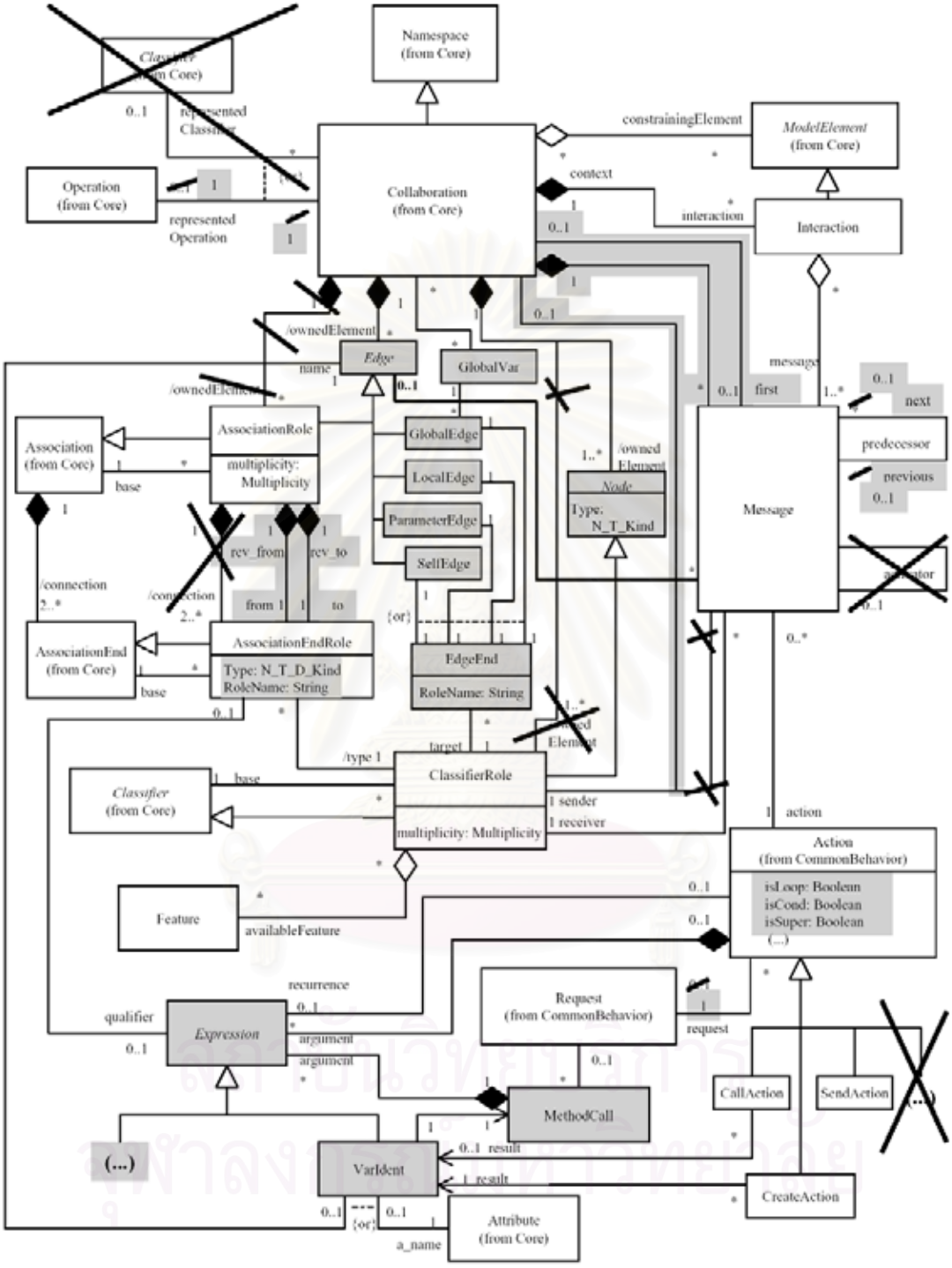
แต่ละสารจะถูกแสดงโดยลูกศรระหว่างเส้นชีวิตของวัตถุสองวัตถุ โดยข้อมูลบนลูกศรจะประกอบไปด้วยสารรวมไปถึงอาร์กิวเมนต์ (Argument) และข้อมูลควบคุมบางอย่าง สำหรับสารที่ส่งมาที่ตัวเองจะแสดงโดยให้ลูกศรวนกลับมาที่เส้นชีวิตเส้นเดิม



รูปที่ 2.4 แสดงซีเควนซ์ไดอะแกรม [6]

2.1.3 **ยูเอ็มแอลเมต้าโมเดล** (UML Meta model) [7]

เมต้าโมเดลเป็นภาษาสำหรับระบุ (Specify) แบบจำลองเชิงวัตถุ (Object model) หรือในอีกนัยหนึ่ง เมต้าโมเดลเป็นการจำลององค์ประกอบที่ใช้ในการสร้างแบบจำลอง ดังแสดงตัวอย่างในรูปที่ 2.5 วัตถุประสงค์ของยูเอ็มแอลเมต้าโมเดล คือ เพื่อให้คำจำกัดความ (Definition) ของวากยสัมพันธ์ (Syntax) และความหมาย (Semantic) สำหรับองค์ประกอบในแบบจำลองยูเอ็มแอล โดยเมต้าโมเดลจะช่วยให้ผู้พัฒนาซอฟต์แวร์



รูปที่ 2.5 แสดงตัวอย่างยูเอ็มแอลเมต้าโมเดลสำหรับคอแลบบอเรชันไดอะแกรม [3]

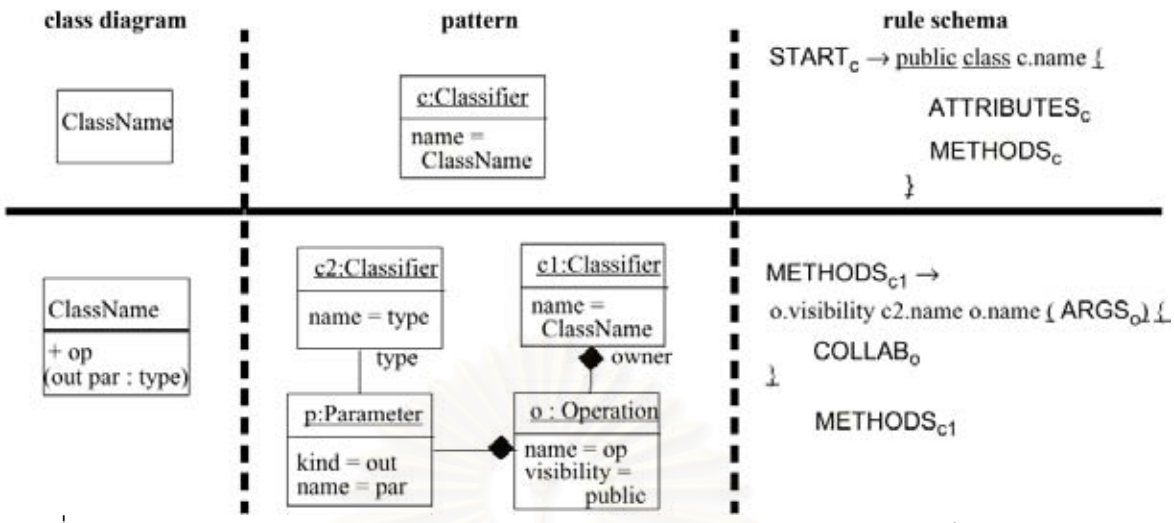
(Developers) สามารถเข้าใจความหมายของแบบจำลองได้ถูกต้องตรงกัน และนอกจากนี้เมต้าโมเดลยังช่วยให้ทีมสามารถสร้างแบบจำลองต่างๆ ตามมาตรฐานของยูเอ็มแอลได้ง่ายขึ้น เนื่องจากยูเอ็มแอลเมต้าโมเดลแสดงให้เห็นถึงความสอดคล้องกันระหว่างองค์ประกอบต่างๆ ของแบบจำลองยูเอ็มแอล

2.1.4 ขั้นตอนวิธีการแปลงที่ขึ้นกับรูปแบบ (Pattern-based transformation algorithm) [3]

แนวความคิดพื้นฐานของการแปลงทั้งจากคลาสไดอะแกรม (Class Diagrams) และคอแลบบอเรชันไดอะแกรมเป็นโปรแกรมบางส่วนของภาษาจาวาในงานวิจัย [3] คือ การระบุรูปแบบมาตรฐาน (Standard patterns) ของไดอะแกรมดังกล่าว แล้วจึงแปลงรูปแบบเหล่านั้นให้เป็นโปรแกรม แนวความคิดนี้เรียกว่า ขั้นตอนวิธีการแปลงที่ขึ้นอยู่กับรูปแบบ (Pattern-based transformation algorithm) ดังแสดงตัวอย่างในรูปที่ 2.6 โดยขั้นตอนวิธีการแปลงที่ขึ้นอยู่กับรูปแบบนี้ จะใช้เมต้ารูล (Meta rules) ซึ่งประกอบไปด้วยเค้าร่างของกฎ (Rule schema) และรูปแบบที่เพิ่มเติม (Additional pattern) ในการแปลงจากไดอะแกรมที่ต้องการไปเป็นโปรแกรมภาษาจาวา

1) รูปแบบ – รูปแบบเป็นส่วนของไดอะแกรมตัวอย่าง (Instance diagram) ของเมต้าโมเดลซึ่งใช้ในการแสดงส่วนของไดอะแกรมที่จะถูกแปลงไปเป็นโปรแกรมภาษาจาวาจริง

2) เค้าร่างของกฎ - เค้าร่างของกฎจะอธิบายการก่อกำเนิด (Generation) ของวากยสัมพันธ์ของภาษาจาวาที่ถูกต้อง โดยจะแสดงอยู่ในรูปของนิพจน์ที่ไม่ขึ้นกับบริบท (Context-free expression) โดยจะประกอบไปด้วยสัญลักษณ์ที่ไม่ถึงจุดจบ (Non-terminal symbols) 2 ชนิด ชนิดแรกเป็นสัญลักษณ์ที่ไม่ถึงจุดจบทั่วไป ซึ่งถูกแทนที่โดยการประยุกต์ใช้กฎ ด้วยลำดับของสัญลักษณ์ที่ไม่ถึงจุดจบ (Sequence of non-terminal) และจุดจบ (Terminals) ชนิดที่สองเป็นพารามิเตอร์ของเค้าร่างของกฎ ซึ่งพารามิเตอร์เหล่านี้ถูกกำหนดโดยใช้คำศัพท์ของเมต้าโมเดล แนวคิดนี้เค้าร่างของกฎนี้เกิดขึ้นมาจากการสร้างตัวแปลภาษา ซึ่งรู้จักกันในชื่อวิธีการไวยากรณ์สองระดับ (Two-level grammar approach)



รูปที่ 2.6 แสดงตัวอย่างรูปแบบ และเค้าร่างของกฎ (เมต้ารูลสำหรับการแปลงคลาสไดอะแกรม) [3]

2.1.5 **ไวยากรณ์สองระดับ** (Two-level grammar) [5]

ไวยากรณ์สองระดับ หรือไวยากรณ์ดับบลิว (W-grammars) เป็นวิธีการใหม่ที่นำเสนอ “กฎเกี่ยวกับกฎ” (Rules about rules) ซึ่งนอกจากจะมีสัญลักษณ์ที่ถึงจุดจบ (Terminal symbols) และกฎการสร้างที่มีจำนวนจำกัด (Production rules) เหมือนเช่นที่ไวยากรณ์หนึ่งระดับ หรือไวยากรณ์บีเอ็นเอฟ (BNF grammar) มีแล้ว ยังเพิ่มไวยากรณ์ในระดับที่สองซึ่งสามารถใช้สร้างกฎการสร้างที่มีจำนวนไม่จำกัดอีกด้วย

แนวคิดของไวยากรณ์สองระดับมีดังนี้

1) โพรโตโนชัน (Protonotions) เป็นส่วนของไวยากรณ์สองระดับซึ่งมีจุดที่ไม่ถึงจุดจบ (Non-terminals) และจุดจบเหมือนที่ไวยากรณ์บีเอ็นเอฟมี โดยจะใช้ลำดับ (Sequence) ของตัวอักษรที่เป็นตัวพิมพ์เล็ก และตัวอักษรที่เป็นตัวหนาในการแทนโพรโตโนชัน แสดงถึงจุดจบโดยใช้คำว่า **symbol** ในโพรโตโนชันสามารถมีช่องว่าง (Spaces) ได้ เนื่องจากช่องว่างในโพรโตโนชันไม่มีความหมาย ดังแสดงตัวอย่างในรูปที่ 2.7

2) ไฮเปอร์รูล (Hyper-rules) เป็นกฎการสร้างที่ให้นิยามโพรโตโนชันหนึ่งๆ หรือกลุ่มของโพรโตโนชัน โดยไฮเปอร์รูลอาจประกอบด้วยโพรโตโนชัน และเมต้าโนชัน (Metanotions) ได้ ดังแสดงตัวอย่างในรูปที่ 7 เครื่องหมายที่ใช้ในไฮเปอร์รูลมีดังนี้

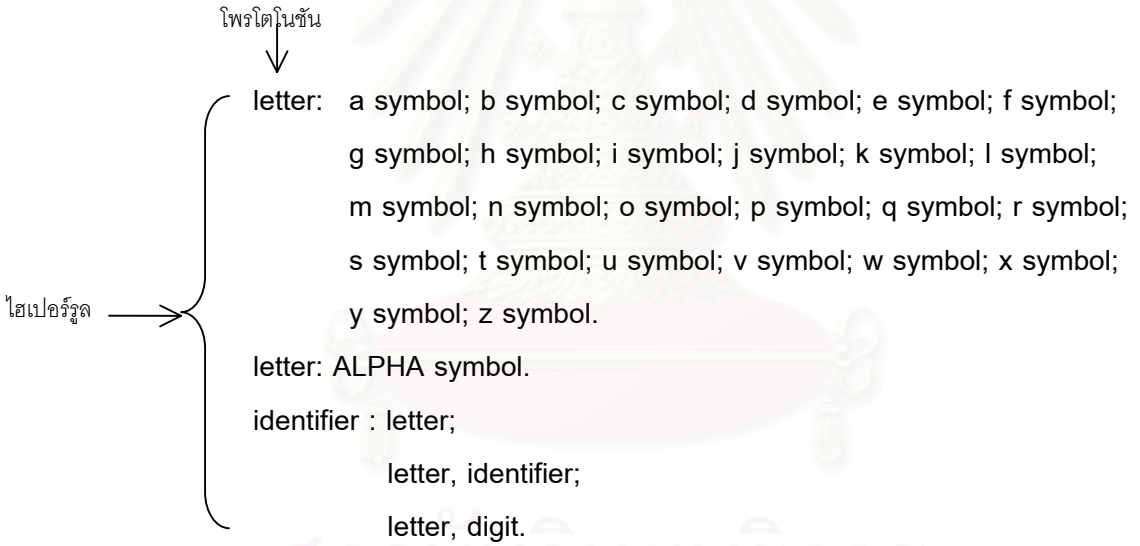
- เครื่องหมาย : (Colon) แบ่งด้านซ้าย และขวาของกฎ
- เครื่องหมาย , (Comma) แบ่งแต่ละโพรโตโนชัน
- เครื่องหมาย ; (Semicolon) แบ่งแต่ละทางเลือกที่อยู่ด้านขวาของกฎ

- เครื่องหมาย . (Period) แสดงถึงจุดจบของกฎ

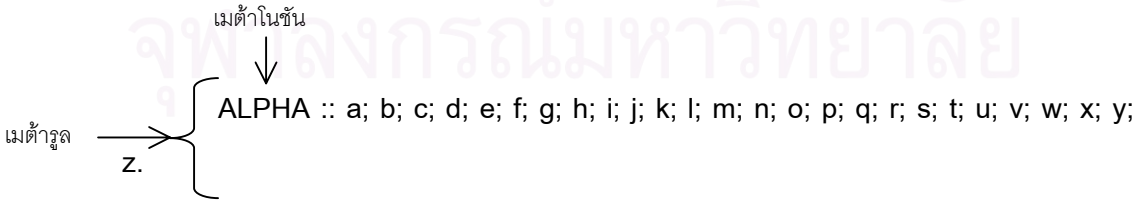
3) เมต้าโนชันเป็นระดับที่สองของไวยากรณ์สองระดับ โดยเมต้าโนชันสามารถแทนที่โพรโตโนชันกี่ตัวก็ได้ เมต้าโนชันจะใช้ตัวอักษรที่เป็นตัวพิมพ์ใหญ่ และตัวอักษรที่เป็นตัวหนา โดยไม่อนุญาตให้มีช่องว่าง ดังแสดงตัวอย่างในรูปที่ 2.8

4) เมต้ารูลเป็นกฎการสร้างที่ให้นิยามเมต้าโนชันหนึ่งที่อยู่ด้านซ้ายด้วยโพรโตโนชัน หรือเมต้าโนชันที่อยู่ด้านขวา ดังแสดงตัวอย่างในรูปที่ 2.8 เครื่องหมายที่ใช้ในเมต้ารูลมีดังนี้

- เครื่องหมาย :: (Double colon) แบ่งด้านซ้าย และขวาของกฎ
- ช่องว่างแบ่งแต่ละโพรโตโนชัน หรือเมต้าโนชัน
- เครื่องหมาย ; (Semicolon) แบ่งแต่ละทางเลือกที่อยู่ด้านขวาของกฎ
- เครื่องหมาย . (Period) แสดงถึงจุดจบของกฎ



รูปที่ 2.7 แสดงตัวอย่างโพรโตโนชัน และไฮเปอร์รูล



รูปที่ 2.8 แสดงตัวอย่างเมต้าโนชัน และเมต้ารูล

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องกับการออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวามีดังนี้

2.2.1 งานวิจัย "UML Collaboration Diagrams and Their Transformation to Java" [3]

งานวิจัยนี้นำเสนอกฎการแปลงยูเอ็มแอลคอแลบบอเรชันไดอะแกรมเป็นชุดคำสั่งภาษาจาวา ซึ่งกฎการแปลงจะอยู่บนพื้นฐานของการแปลงยูเอ็มแอลคลาสไดอะแกรมและใช้ขั้นตอนวิธีการแปลงที่ขึ้นกับรูปแบบที่กล่าวมาแล้วข้างต้นในการแปลง งานวิจัยนี้มีวัตถุประสงค์เพื่อลดการสูญหายของข้อมูลระหว่างขั้นตอนการแปลงโมเดลการออกแบบเป็นโปรแกรมจริง

กฎการแปลงคอแลบบอเรชันไดอะแกรมเป็นบางส่วนของโปรแกรมภาษาจาวาจากงานวิจัยดังกล่าวมีดังนี้ คือ (รายละเอียดดังแสดงในภาคผนวก ก)

1) เมต้ารูลสำหรับตัวแปรครอบคลุม (Meta rules for global variables)

2) เมต้ารูลสำหรับการแบ่งคอแลบบอเรชันไดอะแกรม (Meta rules for splitting of COLLAB)

3) เมต้ารูลสำหรับการประกาศตัวแปรท้องถิ่น (Meta rules for local variable declaration)

4) เมต้ารูลสำหรับการเรียกเมทธอดอย่างสมบูรณ์ (Meta rules for completing method invocation)

5) เมต้ารูลสำหรับการเรียกเมทธอดบนวัตถุพารามิเตอร์ (Meta rules for method invocation on parameter object)

6) เมต้ารูลสำหรับการเรียกเมทธอดผ่านการเชื่อมโยงที่มีความสัมพันธ์กัน (Meta rules for method invocation via association link)

7) เมต้ารูลสำหรับการเรียกเมทธอดผ่านการเชื่อมโยงใหม่ที่มีความสัมพันธ์กัน (Meta rules for method invocation via a {new} association link)

8) เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไขผ่านการเชื่อมโยงที่มีความสัมพันธ์กัน (Meta rules for conditional method invocation via association link)

9) เมต้ารูลสำหรับการผ่านลำดับของสารโดยอยู่บนพื้นฐานของการเรียกเมทธอดบนการเชื่อมโยงที่มีความสัมพันธ์กันที่มีอยู่แล้ว (Meta rules for traversing message sequences based on method invocation on existing association link)

10) เมต้ารูลสำหรับการเรียกตัวสร้าง (Meta rules for constructor invocation)

2.2.2 **งานวิจัย** "Rule-based Specification of Behavioral Consistency based on the UML Meta-Model" [4]

งานวิจัยนี้อภิปรายถึงประเด็นความสอดคล้อง (Consistency) ของแบบจำลองเชิงพฤติกรรม (Behavioral model) ในแบบจำลองยูเอ็มแอล และนำเสนอเทคนิคสำหรับระบุ และวิเคราะห์ความสอดคล้อง โดยใช้กฎยูเอ็มแอลเมต้าโมเดลแปลงองค์ประกอบ (Elements) ของแบบจำลองยูเอ็มแอล (UML model) ให้อยู่ในเขตความหมาย (Semantic domain) เพื่อให้สามารถระบุ และตรวจสอบความสมเหตุสมผล (Validate) โดยภาษา และเครื่องมือของเขตความหมายได้

2.2.3 **งานวิจัย** "Interaction Schemata: Compiling Interactions to Code" [8]

งานวิจัยนี้นำเสนอเครื่องมืออัตโนมัติที่ช่วยเขียนโปรแกรมจากยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยเครื่องมือนี้จะทำให้เกิดโปรแกรมจากการแปลงซีเควนซ์ไดอะแกรมเป็นสกีมาตากลางที่แสดงถึงการกระทำระหว่างกัน ซึ่งเป็นข้อความที่บรรยายถึงการติดต่อระหว่างกันของวัตถุ โดยสกีมาตาที่แสดงถึงการกระทำระหว่างกันนี้จะประกอบไปด้วยกลุ่มของการกระทำ ซึ่งในส่วนของการกระทำนี้ ผู้ใช้จะต้องทำการกำหนดการกระทำ และเพิ่มเติมรายละเอียดบางอย่างที่จำเป็นสำหรับการทำให้เกิดโปรแกรมที่สมบูรณ์ผ่านส่วนต่อประสาน (Interface) ของโปรแกรม เช่น ชนิดของพารามิเตอร์ เป็นต้น (แต่ละสารที่ส่งในยูเอ็มแอลซีเควนซ์ไดอะแกรมสามารถแปลงได้เป็นหนึ่ง หรือหลายการกระทำ) โดยการกระทำหลักๆ ที่งานวิจัยนี้สนใจคือ การเพิ่ม การลบ การวนซ้ำ (Iterate) และการค้นหา ที่ใช้ในการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นโปรแกรม

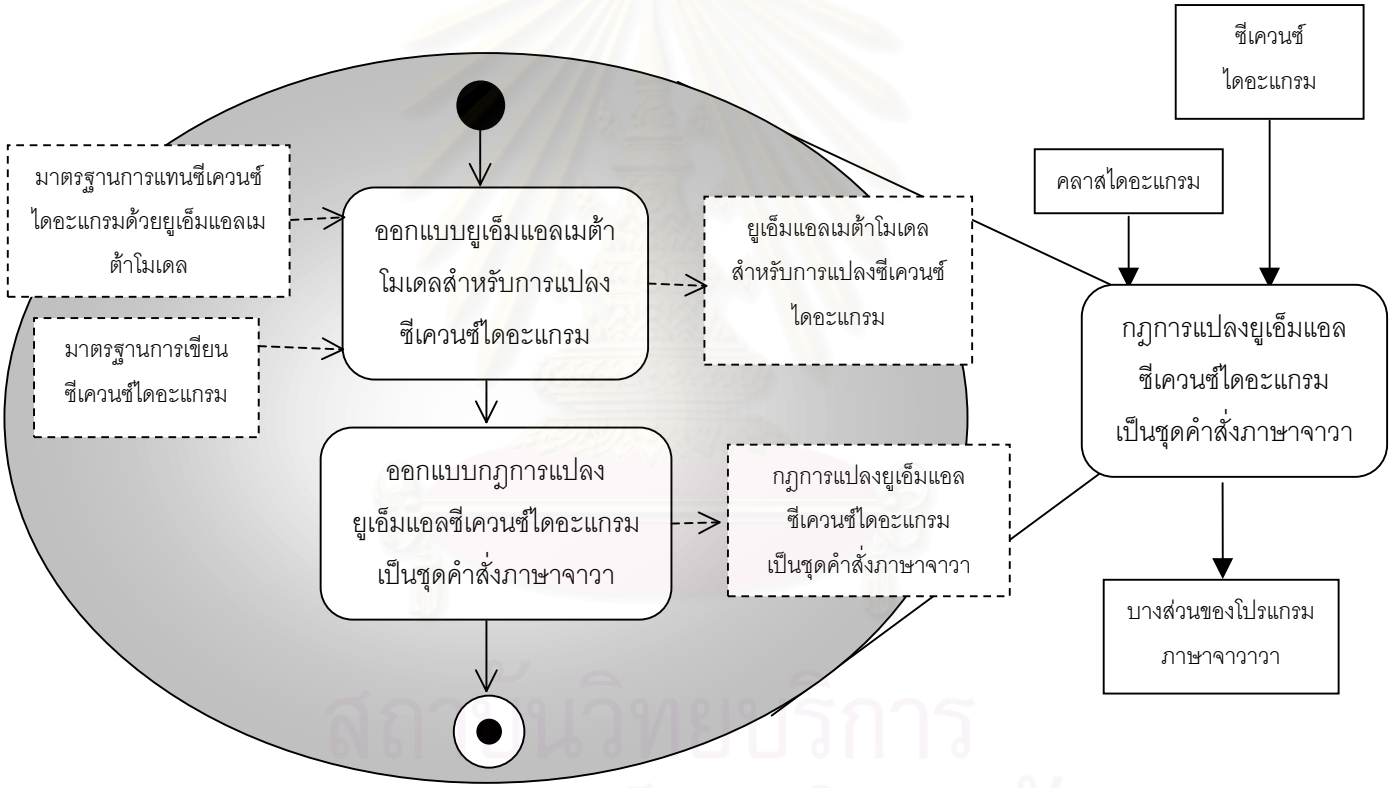
2.2.4 **งานวิจัย** "XML Rule Based Source Code Generator for UML Case Tool" [9]

งานวิจัยนี้มีวัตถุประสงค์เพื่อนำเสนอเค้าโครง (Framework) สำหรับการสร้างเครื่องมือที่สามารถทำให้เกิดโปรแกรมจากยูเอ็มแอลคลาสไดอะแกรม ที่สามารถเพิ่ม

ขยายได้ (เพิ่มขยายได้ในที่นี้หมายถึง ความสามารถในการดึงข้อมูลคลาสไดอะแกรมจากที่เก็บข้อมูลที่ต่างกัน และสามารถทำให้เกิดโปรแกรมได้หลายภาษา) โดยงานวิจัยนี้จะออกแบบส่วนต่อประสานสำหรับโปรแกรมประยุกต์ (Application Program Interface) ที่ใช้ในการสกัด (Extract) ข้อมูลของคลาสไดอะแกรมจากที่เก็บข้อมูลไว้ให้ ผู้ใช้ที่ต้องการสกัดข้อมูลจากที่เก็บข้อมูล (Repository) ของแต่ละเครื่องมือจะต้องทำการเพิ่มเติมตรรกะ (Logic) สำหรับการสกัดข้อมูลจากที่เก็บข้อมูลที่มีรูปแบบเฉพาะนั้นๆ เอง และนำเสนอเค้าโครงสำหรับการสร้างเครื่องมือที่สามารถทำให้เกิดโปรแกรมในภาษาต่างๆ ได้ โดยการประยุกต์ใช้กฎการแปลงที่แตกต่างกัน ทั้งนี้ขึ้นอยู่กับภาษาที่ต้องการแปลง

บทที่ 3
การออกแบบยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม

ในบทนี้จะเริ่มต้นจากการนำเสนอแนวคิดการออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา หลังจากนั้นจะอธิบายถึงการออกแบบยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม และแสดงยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรมที่ได้ออกแบบไว้ สำหรับกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมจะอธิบายในบทถัดไป



รูปที่ 3.1 แสดงแนวคิดการออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

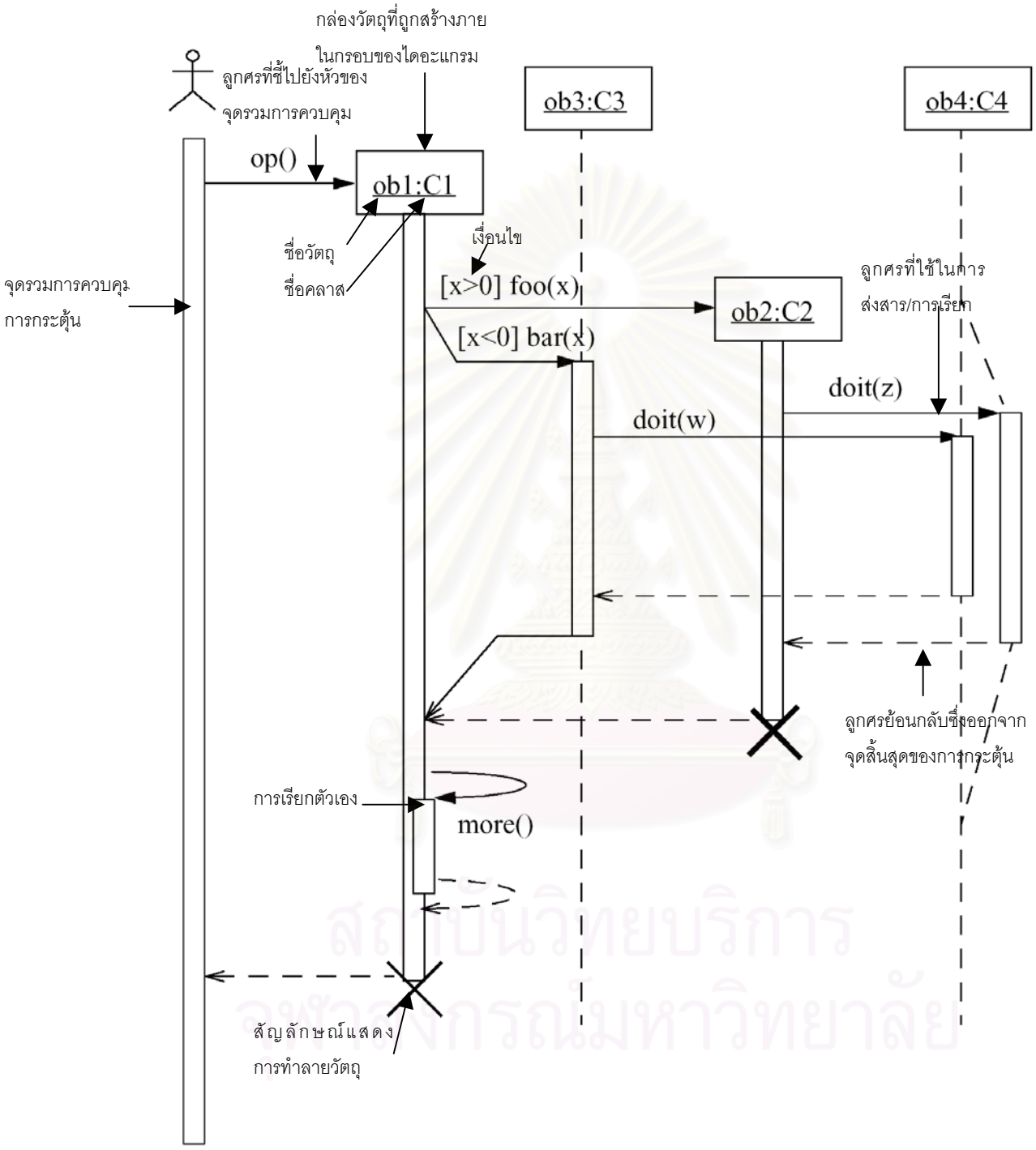
รูปที่ 3.1 แสดงแนวคิดการออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาซึ่งเริ่มต้นจากการออกแบบยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรมโดยใช้มาตรฐานการเขียนซีเควนซ์ไดอะแกรม และมาตรฐานการแทนซีเควนซ์ไดอะแกรมด้วยยูเอ็มแอลเมต้าโมเดลเป็นข้อมูลนำเข้า ผลลัพธ์ที่ได้จากการออกแบบจะได้ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรมดังรูปที่ 3.3 หลังจากนั้นจึงทำการออกแบบกฎการ

แปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมโดยใช้ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรมที่ได้ออกแบบไว้เป็นข้อมูลนำเข้า ผลลัพธ์ที่ได้คือ กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา สำหรับการประยุกต์ใช้กฎ จะใช้ซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมเป็นข้อมูลนำเข้า หลังจากผ่านการประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมที่ได้ออกแบบไว้ จะได้ผลลัพธ์เป็นบางส่วนของชุดคำสั่งภาษาจาวา สำหรับในส่วนถัดไปจะอธิบายถึงการออกแบบยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม

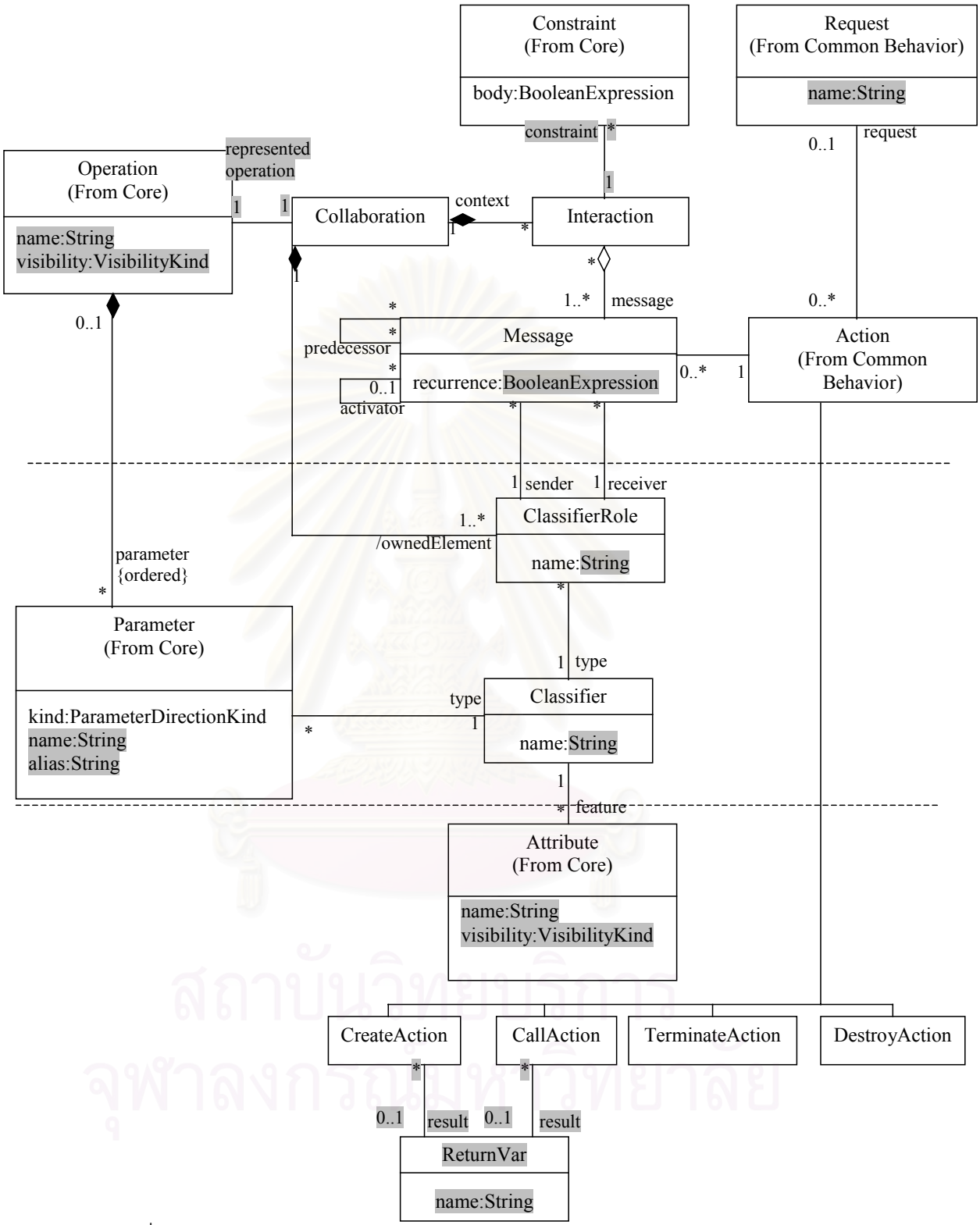
ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรมถูกแสดงในรูปของคลาสไดอะแกรม เพื่อใช้ในการแทนความหมายของซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมที่เป็นข้อมูลนำเข้า ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรมนี้มาจากการนำบางส่วนของยูเอ็มแอลเมต้าโมเดลที่มีอยู่ตามมาตรฐานความหมายของยูเอ็มแอลเวอร์ชัน 1.1 (UML Semantics. Version 1.1) [7] เฉพาะที่จำเป็นต้องใช้ในการเก็บข้อมูลซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมเพื่อสร้างชุดคำสั่งภาษาจาวา และทำการเปลี่ยนแปลง หรือเพิ่มเติมบางส่วนของยูเอ็มแอลเมต้าโมเดลที่นำมานั้น เพื่อให้เพียงพอกับการเก็บข้อมูลซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมที่เป็นข้อมูลนำเข้า

การออกแบบยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรมจะเริ่มจากพิจารณาเลือกคลาส คุณลักษณะประจำคลาส และความสัมพันธ์ระหว่างคลาสจากยูเอ็มแอลเมต้าโมเดลที่มีอยู่ตามมาตรฐานเท่าที่จำเป็น การพิจารณาความจำเป็นดูจากความสามารถในการแทนซีเควนซ์ไดอะแกรมโดยคลาส คุณลักษณะประจำคลาส และความสัมพันธ์ระหว่างคลาสของยูเอ็มแอลเมต้าโมเดลที่มีอยู่ตามมาตรฐาน โดยความสามารถในการแทนจะอธิบายโดยแบ่งออกเป็นสองส่วนได้แก่ ส่วนของการแทนซีเควนซ์ไดอะแกรมด้วยยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม ซึ่งในส่วนนี้จะเป็นการแทนซีเควนซ์ไดอะแกรมด้วยยูเอ็มแอลเมต้าโมเดลตามมาตรฐานการแทนซีเควนซ์ไดอะแกรมในหนังสือแนะนำยูเอ็มแอลเวอร์ชัน 1.1 (UML Notation Guide. Version 1.1) [10] และส่วนการเพิ่มเติมการแทนซีเควนซ์ไดอะแกรมด้วยยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม ซึ่งเป็นการกำหนดการแทนซีเควนซ์ไดอะแกรมเพิ่มเติม เนื่องจากมาตรฐานการแทนซีเควนซ์ไดอะแกรมในหนังสือแนะนำยูเอ็มแอลเวอร์ชัน 1.1 ไม่ได้กำหนดไว้ โดยการแทนในส่วนนี้อาจมีการเพิ่มเติมคลาส คุณลักษณะประจำคลาส และความสัมพันธ์ระหว่างคลาสขึ้นเอง ในกรณีที่คลาส คุณลักษณะประจำคลาส และความสัมพันธ์ระหว่างคลาสไม่มีในยูเอ็มแอลเมต้าโมเดลที่มีอยู่ตามมาตรฐาน สำหรับการเพิ่มเติมนี้จะแสดงด้วยแรเงาสีเทา

หลังจากการศึกษาซีเควนซ์ไดอะแกรมดังรูปที่ 3.2 และทำการแทนซีเควนซ์ไดอะ



รูปที่ 3.2 แสดงซีเควนซ์ไดอะแกรมที่มีจุดรวมการควบคุม (Focus of control), เงื่อนไข,การเรียกตัวเอง, การสร้าง และการทำลาย



รูปที่ 3.3 แสดงยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม

แกรมด้วยยูเอ็มแอลเมต้าโมเดลตามการแทนซีเควนซ์ไดอะแกรมด้วยยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม และการเพิ่มเติมการแทนซีเควนซ์ไดอะแกรมด้วยยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม จะได้ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรมดังรูปที่ 3.3

3.1 ส่วนการแทนซีเควนซ์ไดอะแกรมด้วยยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม

ในส่วนนี้จะทำการอธิบายถึงการแทนส่วนต่างๆ ของซีเควนซ์ไดอะแกรมลงในยูเอ็มแอลเมต้าโมเดลที่แสดงด้วยคลาสไดอะแกรมตามมาตรฐานการแทนซีเควนซ์ไดอะแกรมในหนังสือแนะนำยูเอ็มแอลเวอร์ชัน 1.1

ส่วนบนสุดของรูปที่ 3.3 (ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม) มีการแทนจากรูปที่ 3.2 (ซีเควนซ์ไดอะแกรม) ดังนี้

1) ซีเควนซ์ไดอะแกรมแทนด้วยคลาส Collaboration และคลาส Interaction ต่างๆ ที่อยู่ภายใต้การร่วมมือกันระหว่างวัตถุ (Collaboration)

2) ลูกศรที่ใช้ในการส่งสาร (Message arrow) แทนด้วยคลาส Message ระหว่างคลาส ClassifierRole

3) ข้อบังคับ (Constraint) ซึ่งเขียนในไดอะแกรมแทนด้วยคลาส Constraint ซึ่งเป็นข้อบังคับของการกระทำระหว่างกันทั้งหมด

4) สำหรับซีเควนซ์ไดอะแกรมที่มีจุดรวมการควบคุม และการเรียก (Calls) ลูกศรที่ตามมาภายหลังบนเส้นชีวิต (Lifeline) เส้นเดียวกันแทนด้วยคลาส Message และความสัมพันธ์ชื่อ predecessor

5) ลูกศรที่ชี้ไปยังหัวของจุดรวมการควบคุมแทนด้วยคลาส Message ที่สมวาร (Synchronous) และมีการกระตุ้น(Activation) ระหว่างคลาส ClassifierRole

6) เงื่อนไขที่คุ้มครอง (Guard condition) หรือเงื่อนไขที่วนซ้ำ (Iteration condition) ที่ผูกติดกับลูกศรที่ใช้ในการส่งสารแทนด้วยคุณลักษณะ recurrence ของคลาส Message

ส่วนกลางของรูปที่ 3.3 (ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม) มีการแทนจากรูปที่ 3.2 (ซีเควนซ์ไดอะแกรม) คือ แต่ละกล่องวัตถุ (Object box) ซึ่งแทนด้วยสัญลักษณ์สี่เหลี่ยม และเส้นชีวิตซึ่งแทนด้วยเส้นประ จะถูกแทนด้วยคลาส ClassifierRole ชื่อ

วัตถุแทนด้วยคุณลักษณะ name ของคลาส ClassifierRole และประเภทของวัถตุจะแทนด้วยความสัมพันธ์ชื่อ type จากคลาส ClassifierRole ถึงคลาส Classifier

ส่วนล่างสุดของรูปที่ 3.3 (ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม) มีการแทนจากรูปที่ 3.2 (ซีเควนซ์ไดอะแกรม) ดังนี้

1) กล่องวัตถุซึ่งถูกสร้างภายในกรอบของไดอะแกรมแทนด้วยคลาส Create Action ที่ผูกติดกับคลาส Message ของลูกศรที่ชี้เข้ากล่องวัตถุนั้น

2) ถ้ามีสัญลักษณ์แสดงการทำลายของวัตถุ (Object termination symbol: X) แล้วสัญลักษณ์แสดงการทำลายเป็นเป้าของลูกศร สัญลักษณ์แสดงการทำลายของวัตถุจะถูกแทนด้วยคลาส DestroyAction ซึ่งผูกติดกับคลาส Message มิฉะนั้นแล้วสัญลักษณ์แสดงการทำลายของวัตถุจะแทนด้วย TerminateAction

3) ลูกศรย้อนกลับซึ่งออกจากจุดสิ้นสุดของการกระตุ้นแทนด้วยคลาส Message ที่สมวาร และตอบกลับ(Reply) ที่มีความสัมพันธ์ชื่อ activation ไปยังคลาส Message ของลูกศรที่หัวของการกระตุ้น และความสัมพันธ์ชื่อ predecessor ไปยังข้อความอันก่อนที่อยู่ภายในการกระตุ้นเดียวกัน

4) ลูกศรที่ชี้ไปยังหัวของจุดรวมการควบคุมแทนด้วย CallAction

3.2 ส่วนเพิ่มเติมการแทนซีเควนซ์ไดอะแกรมด้วยยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม

ในส่วนนี้จะทำการอธิบายถึงการเพิ่มเติมการแทนส่วนต่างๆ ของซีเควนซ์ไดอะแกรมลงในยูเอ็มแอลเมต้าโมเดลที่แสดงด้วยคลาสไดอะแกรมซึ่งเป็นการเพิ่มเติม หรือแก้ไขยูเอ็มแอลเมต้าโมเดล เพื่อความเหมาะสมในการเก็บข้อมูลสำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม โดยจะแบ่งการอธิบายออกเป็นสองส่วนย่อย ได้แก่ ส่วนเพิ่มเติมการแทนที่ใช้ยูเอ็มแอลเมต้าโมเดลตามมาตรฐาน เนื่องจากมาตรฐานความหมายของยูเอ็มแอลเวอร์ชัน 1.1 มีการกำหนดไว้แล้ว และส่วนเพิ่มเติมการแทนที่กำหนดขึ้นเอง เนื่องจากมาตรฐานความหมายของยูเอ็มแอลเวอร์ชัน 1.1 ไม่ได้มีการกำหนดไว้

3.2.1 ส่วนเพิ่มเติมการแทนที่ใช้ยูเอ็มแอลเมต้าโมเดลตามมาตรฐาน

ส่วนบนสุดของรูปที่ 3.3 (ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม) มีการแทนจากรูปที่ 3.2 (ซีเควนซ์ไดอะแกรม) ดังนี้

1) เมทธอดที่ถูกเรียกใช้โดยเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายแทนด้วยคลาส Request

2) เมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายแทนด้วยคลาส Operation

ส่วนกลางของรูปที่ 3.3 (ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม) มีการแทนจากรูปที่ 3.2 (ซีเควนซ์ไดอะแกรม) คือ พารามิเตอร์ที่ส่ง และรับในเมทธอดแทนด้วยคลาส Parameter ชนิดของพารามิเตอร์แทนด้วยคุณลักษณะ kind ของคลาส Parameter ซึ่งมีประเภทเป็น ParameterDirectionKind ประเภทของ ParameterDirectionKind ได้แก่ in out inout และ return

3.2.2 ส่วนเพิ่มเติมการแทนที่กำหนดขึ้นเอง

ส่วนบนสุดของรูปที่ 3.3 (ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม) มีการแทนจากรูปที่ 3.2 (ซีเควนซ์ไดอะแกรม) ดังนี้

1) ชื่อเมทธอดที่ถูกเรียกใช้โดยเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายแทนด้วยคุณลักษณะ name ของคลาส Request ซึ่งมีประเภทเป็น String

2) ชื่อเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายแทนด้วยคุณลักษณะ name ของคลาส Operation ซึ่งมีประเภทเป็น String และระดับของการมองเห็นเมทธอดโดยวัตถุอื่นแทนด้วยคุณลักษณะ visibility ของคลาส Operation ซึ่งมีประเภทเป็น VisibilityKind ประเภทของ VisibilityKind ได้แก่ public private และ protected โดยเมทธอดหนึ่งจะถูกอธิบายโดยการร่วมมือกันระหว่างวัตถุต่างๆ ซึ่งแทนด้วยความสัมพันธ์ชื่อ represented operation จากคลาส Operation ไปยังคลาส Collaboration

3) ประเภทของข้อความของเงื่อนไขที่คุ้มครอง หรือเงื่อนไขที่วนซ้ำแทนด้วยประเภท BooleanExpression ของคุณลักษณะ recurrence ของคลาส Message

ส่วนกลางของรูปที่ 3.3 (ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม) มีการแทนจากรูปที่ 3.2 (ซีเควนซ์ไดอะแกรม) ดังนี้

1) ชื่อพารามิเตอร์ที่ส่ง และรับในเมทธอดแทนด้วยคุณลักษณะ name ของคลาส Parameter ซึ่งมีประเภทเป็น String ชื่อพารามิเตอร์ที่ถูกเรียกโดยเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายแทนด้วยคุณลักษณะ alias ของคลาส Parameter ซึ่งมีประเภทเป็น String

2) ประเภทของชื่อของวัตถุซึ่งแทนด้วยคุณลักษณะ name ของคลาส ClassifierRole มีประเภทเป็น String

3) ประเภทของวัตถุแทนด้วยคลาส Classifier ชื่อประเภทของวัตถุแทนด้วยคุณลักษณะ name ของคลาส Classifier ซึ่งมีประเภทเป็น String

ส่วนล่างสุดของรูปที่ 3.3 (ยูเอ็มแอลเมต้าโมเดลสำหรับการแปลงซีเควนซ์ไดอะแกรม) มีการแทนจากรูปที่ 3.2 (ซีเควนซ์ไดอะแกรม) ดังนี้

1) คุณลักษณะของคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายแทนด้วยคลาส Attribute ชื่อคุณลักษณะแทนด้วยคุณลักษณะ name ของคลาส Attribute ซึ่งมีประเภทเป็น String และระดับของการมองเห็นคุณลักษณะโดยวัตถุอื่นแทนด้วยคุณลักษณะ visibility ของคลาส Attribute ซึ่งมีประเภทเป็น VisibilityKind

2) ตัวแปรที่ใช้รับค่าที่ส่งกลับมาจากการเรียกใช้เมทธอดแทนด้วยคลาส ReturnVar ที่มีความสัมพันธ์ชื่อ result ไปยังคลาส CreateAction และคลาส CallAction ชื่อตัวแปรแทนด้วยคุณลักษณะ name ของคลาส ReturnVar ซึ่งมีประเภทเป็น String

บทที่ 4
กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาประกอบไปด้วยกฎ 9 กฎซึ่งแปลงทั้งซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมเป็นชุดคำสั่งภาษาจาวา โดยในการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา จะสนใจเฉพาะการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมที่มีลักษณะทั่วไปเท่านั้น แต่ไม่ครอบคลุมถึงซีเควนซ์ไดอะแกรมที่แสดงข้อความที่ไม่ได้เกิดพร้อมกัน (Asynchronous message) การเกิดภาวะพร้อมกัน (Concurrency) และเงื่อนไขที่มีการแข่งขัน (Race condition) นอกจากนี้ เนื่องจากภาษาจาวาไม่มีการทำลายตัวของวัตถุเอง จึงไม่มีการออกแบบเมต้ารูลสำหรับการทำลายตัววัตถุเองในกรณีที่มีสัญลักษณ์แสดงการทำลายของวัตถุเกิดขึ้นในซีเควนซ์ไดอะแกรม และเนื่องจากข้อมูลเงื่อนไขที่วนซ้ำไม่เพียงพอสำหรับการสร้างชุดคำสั่งภาษาจาวา จึงไม่ได้มีการออกแบบกฎที่ครอบคลุมถึงเงื่อนไขที่มีการวนซ้ำ

กฎ 8 กฎ สำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวามีดังนี้ (รายละเอียดภาพรวมของกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา ดังแสดงในภาคผนวก จ)

1) เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย (Meta rules for class diagram of a method that the sequence diagram depicts)

2) เมต้ารูลสำหรับการแบ่งซีเควนซ์ (Meta rules for splitting of SEQUENCE)

3) เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง (Meta rules for conditional method invocation and branching)

4) เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร (Meta rules for assigning a value to a variable)

5) เมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้ (Meta rules for assigning object to pointer)

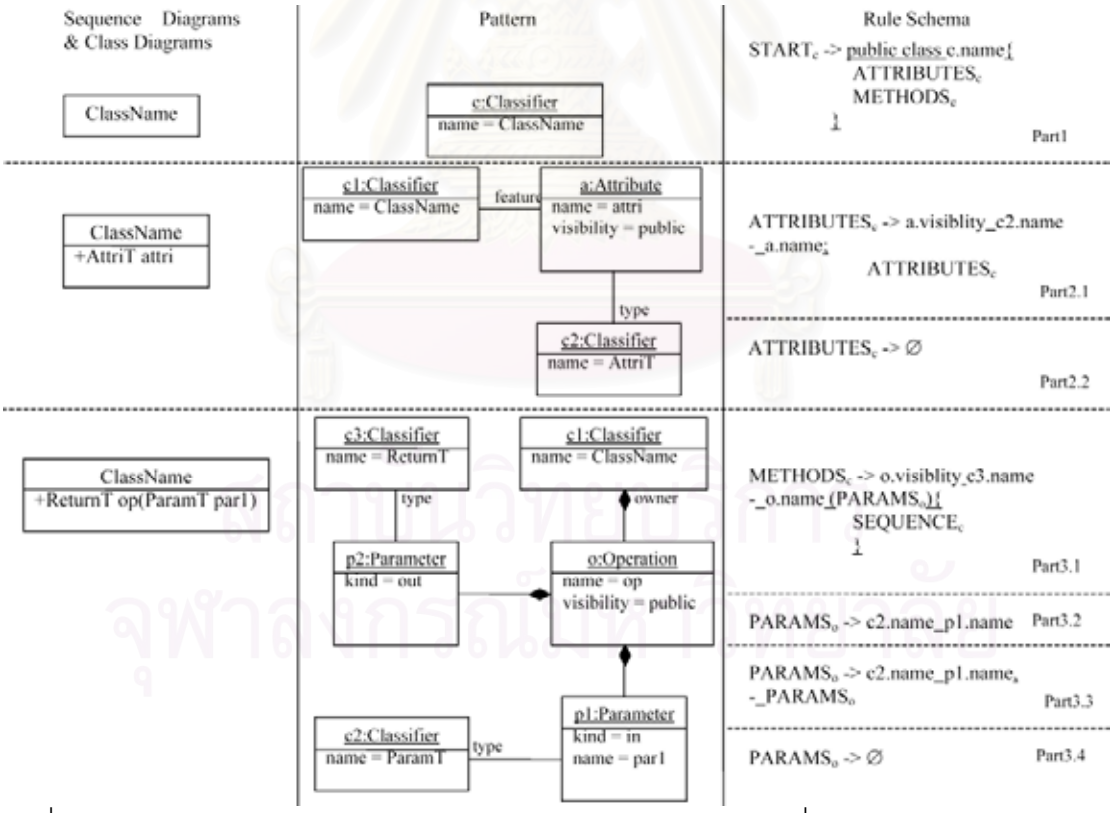
6) เมต้ารูลสำหรับการสร้างวัตถุใหม่ (Meta rules for creating new object)

7) เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว (Meta rules for invoking a method of existing object)

8) เมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเอง (Meta rules for invoking a method of object itself)

ในการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา คลาสไดอะแกรมต้องถูกนำมาใช้ด้วยเพราะซีเควนซ์ไดอะแกรมขาดข้อมูลประเภทของตัวแปร โดยกฎทั้งหมดสำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมจะแบ่งเป็นส่วนตามแถว (Column) ต่างๆ ดังนี้ แถวที่ 1 ดังเช่นตัวอย่างในรูปที่ 4.1 แสดงส่วนของซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมที่ต้องการแปลงเป็นชุดคำสั่งภาษาจาวา แถวที่ 2 แสดงรูปแบบสำหรับเก็บข้อมูลของคลาส และซีเควนซ์ไดอะแกรม แถวที่ 3 แสดงเค้าร่างของกฎสำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมเป็นชุดคำสั่งภาษาจาวา ซึ่งเค้าร่างของกฎจะแสดงอยู่ในรูปของนิพจน์ที่ไม่ขึ้นกับบริบทที่อยู่บนพื้นฐานของไวยากรณ์สองระดับ ในส่วนเค้าร่างของกฎ คำที่ขีดเส้นใต้จะถือเป็นจุดจบ และเนื่องจากการกำหนดกฎในเค้าร่างของกฎอาจไม่สามารถทำได้ภายในบรรทัดเดียว กฎที่เกินจะถูกกำหนดในบรรทัดต่อมาโดยมีสัญลักษณ์ขีด (Dashed symbol) นำหน้า เมื่อนำกฎไปประยุกต์ใช้ให้ปฏิบัติเหมือนบรรทัดก่อนหน้าและบรรทัดที่มีสัญลักษณ์ขีด เป็นบรรทัดเดียวกัน

4.1 เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย



รูปที่ 4.1 แสดงเมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย

คำอธิบายเมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายแสดงดังรูปที่ 4.1 มีดังนี้

1) ส่วนที่ 1 (Part1) - ส่วนนี้เป็นจุดเริ่มต้นของการแปลงซีเควนซ์ไดอะแกรม โดยเริ่มต้นจากการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายก่อน ในแถวที่ 2 ข้อมูลชื่อคลาสของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ชื่อคลาสของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกนำไปแทนที่ c.name ในแถวที่ 3

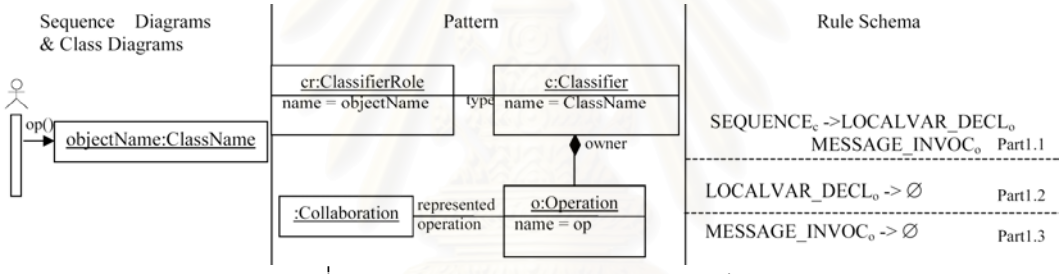
2) ส่วนที่ 2.1 และ 2.2 – ส่วนนี้แสดงการประกาศคุณลักษณะประจำคลาสของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย ในแถวที่ 2 ข้อมูลชื่อคลาสของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c1 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ชื่อคลาสของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกนำไปแทนที่ c1.name ข้อมูลชื่อคุณลักษณะประจำจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ a ซึ่งเป็นกรณีตัวอย่างของคลาส Attribute ข้อมูลระดับการมองเห็นจะถูกเก็บไว้ในคุณลักษณะประจำ visibility ของวัตถุ a ซึ่งเป็นกรณีตัวอย่างของคลาส Attribute เมื่อประยุกต์ใช้กฎ ชื่อ และระดับการมองเห็นของคุณลักษณะประจำจะถูกนำไปแทนที่ a.name และ a.visibility ตามลำดับ ข้อมูลประเภทของคุณลักษณะประจำจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c2 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ประเภทของคุณลักษณะประจำจะถูกนำไปแทนที่ c2.name $ATTRIBUTES_c$ ตัวที่ 2 ในส่วนที่ 2.1 ใช้สำหรับการวนซ้ำประกาศคุณลักษณะประจำ หากไม่มีคุณลักษณะประจำที่ต้องการประกาศแล้ว $ATTRIBUTES_c$ จะถูกแทนที่ด้วยเซตว่างจากการประยุกต์ใช้กฎในส่วนที่ 2.2

3) ส่วนที่ 3.1 – ส่วนนี้แสดงการประกาศเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย ข้อมูลระดับการมองเห็นของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกเก็บไว้ในคุณลักษณะประจำ visibility ของวัตถุ o ซึ่งเป็นกรณีตัวอย่างของคลาส Operation เมื่อประยุกต์ใช้กฎ ระดับการมองเห็นของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกนำไปแทนที่ o.visibility ข้อมูลประเภทของค่าที่คืนจากการเรียกเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c3 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ประเภทของค่าที่คืนจากการเรียกเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกนำไปแทนที่ c3.name ข้อมูลชื่อเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ o ซึ่งเป็นกรณีตัวอย่างของคลาส Operation เมื่อประยุกต์ใช้กฎ ชื่อเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายจะถูกนำไปแทนที่ o.name $SEQUENCE_c$ จะถูกแทนค่าด้วยชุดคำสั่งสำหรับซีเควนซ์ไดอะแกรม

4) ส่วนที่ 3.2 ถึง 3.4 – ส่วนนี้แสดงการประกาศพารามิเตอร์ ส่วนที่ 3.2 เป็นการประกาศพารามิเตอร์ที่รับ กรณีที่เมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายมีพารามิเตอร์ที่รับตัวเดียว

ส่วนที่ 3.3 เป็นการประกาศพารามิเตอร์ที่รับ กรณีที่เมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายมีพารามิเตอร์ที่รับมากกว่า 1 ตัว $PARAMS_o$ ตัวที่ 2 ในส่วนที่ 3.3 ใช้สำหรับการวนซ้ำประกาศพารามิเตอร์ที่รับ หากไม่มีพารามิเตอร์ที่รับที่ต้องการประกาศแล้ว $PARAMS_o$ จะถูกแทนที่ด้วยเซตว่าง จากการประยุกต์ใช้กฎในส่วนที่ 3.4 ในแถวที่ 2 ชนิดของพารามิเตอร์จะถูกเก็บไว้ในคุณลักษณะประจำ kind ของวัตถุ p1 และ p2 ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter ตามลำดับ ในส่วนที่ 3.2 และ 3.3 ข้อมูลประเภทของพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c2 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ประเภทของพารามิเตอร์ที่รับจะถูกนำไปแทนที่ c2.name ข้อมูลชื่อพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ p1 ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter เมื่อประยุกต์ใช้กฎ ชื่อพารามิเตอร์ที่รับจะถูกนำไปแทนที่ p1.name

4.2 เมต้ารูลสำหรับการแบ่งซีเควนซ์



รูปที่ 4.2 แสดงเมต้ารูลสำหรับการแบ่งซีเควนซ์

คำอธิบายเมต้ารูลสำหรับการแบ่งซีเควนซ์แสดงดังรูปที่ 4.2 มีดังนี้

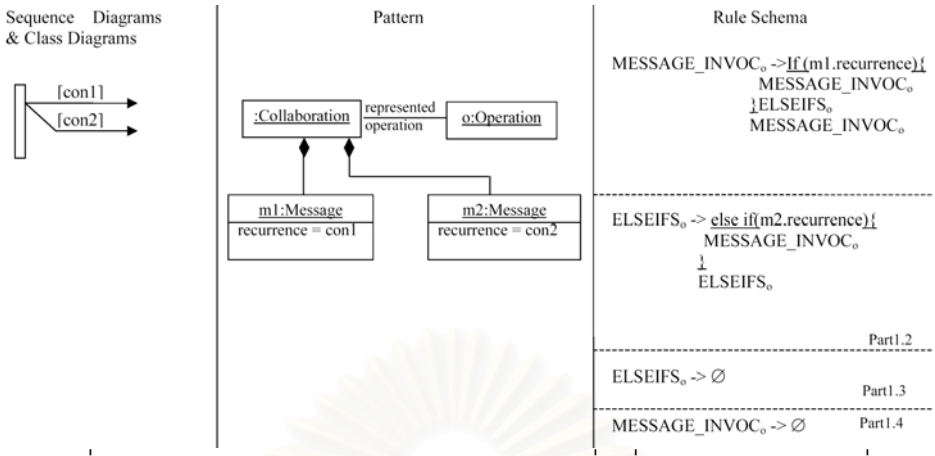
1) ส่วนที่ 1.1 - จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ $SEQUENCE_c$ ด้วยลำดับของสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_o$ และ $MESSAGE\_INVOC_o$ เพื่อกำหนดโครงสร้างของชุดคำสั่งในเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย

2) ส่วนที่ 1.2 ถึง 1.3 จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_o$ และ $MESSAGE\_INVOC_o$ ด้วยเซตว่าง เพื่อจบการสร้างชุดคำสั่งภาษาจาวา

4.3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง

คำอธิบายเมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่งแสดงดังรูปที่ 4.3 มีดังนี้

1) ส่วนที่ 1.1 - ส่วนนี้เป็นการประกาศเงื่อนไขแรก ข้อมูลเงื่อนไขจะถูกเก็บไว้ใน



รูปที่ 4.3 แสดงเมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง

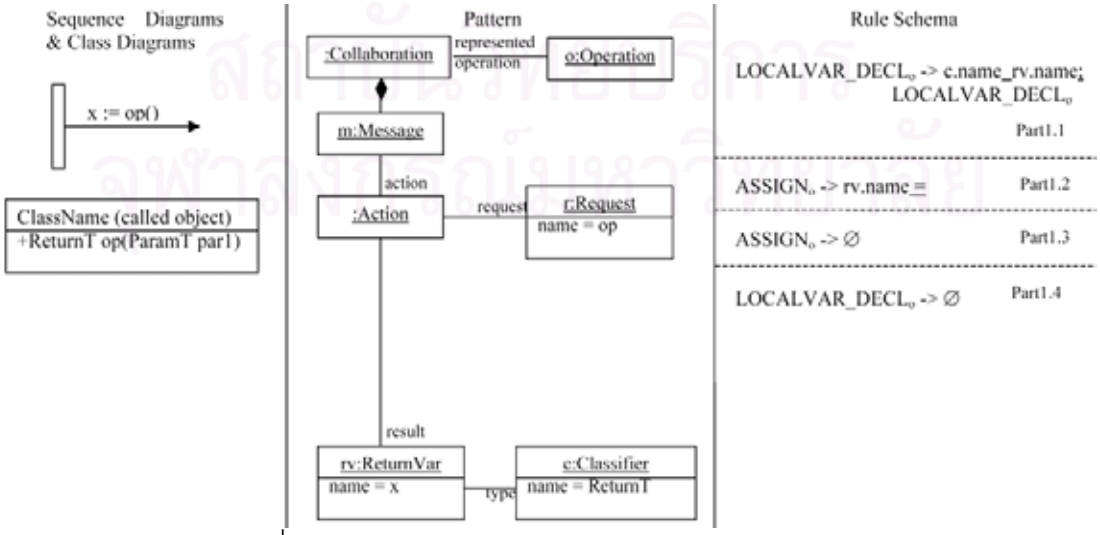
คุณลักษณะประจำ recurrence ของวัตถุ m1 ซึ่งเป็นกรณีตัวอย่างของคลาส Message เมื่อประยุกต์ใช้กฎ ประโยคเงื่อนไขจะถูกนำไปแทนที่ m1.recurrence ในแถวที่ 3 MESSAGE_INVOC$_o$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศการเรียกใช้เมทธอด

2) ส่วนที่ 1.2 – ส่วนนี้เป็นการประกาศเงื่อนไขต่อๆ มา ข้อมูลเงื่อนไขจะถูกเก็บไว้ในคุณลักษณะประจำ recurrence ของวัตถุ m2 ซึ่งเป็นกรณีตัวอย่างของคลาส Message เมื่อประยุกต์ใช้กฎ ประโยคเงื่อนไขจะถูกนำไปแทนที่ m2.recurrence ในแถวที่ 3 ELSEIFS$_o$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศเงื่อนไข

3) ส่วนที่ 1.3 – ส่วนนี้เป็นการทำให้การประกาศเงื่อนไขเสร็จสิ้นคือ หากไม่มีเงื่อนไขที่ต้องการประกาศแล้ว ELSEIFS$_o$ จะถูกแทนที่ด้วยเซตว่าง

4) ส่วนที่ 1.4 จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ MESSAGE_INVOC$_o$ ด้วยเซตว่าง เพื่อจบการสร้างชุดคำสั่งภาษาจาวา

4.4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร



รูปที่ 4.4 แสดงเมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร

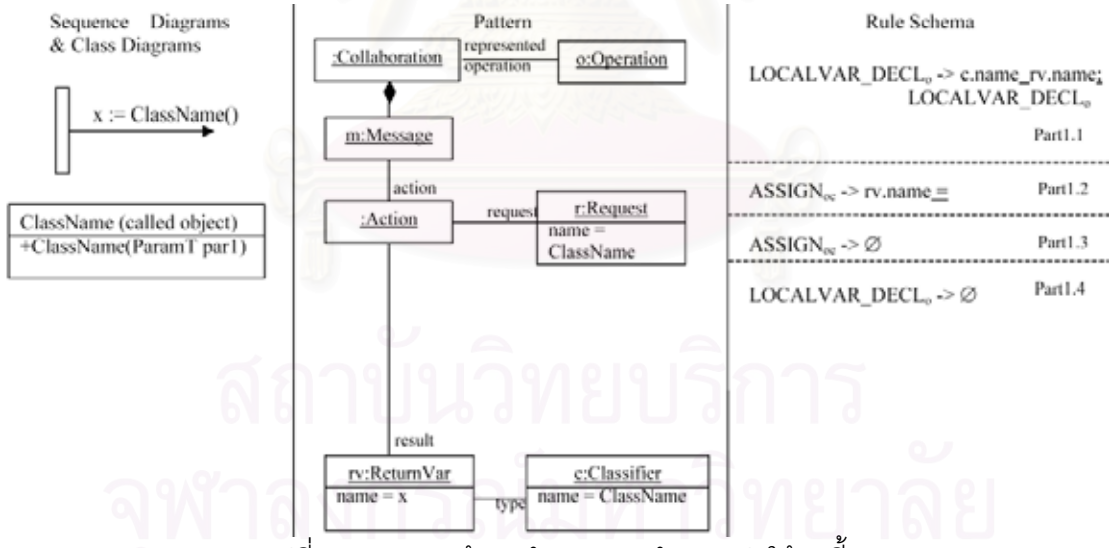
คำอธิบายเมต้ารูลสำหรับการกำหนดค่าให้ตัวแปรแสดงดังรูปที่ 4.4 มีดังนี้

1) ส่วนที่ 1.1 และ 1.2 – ส่วนที่ 1.1 เป็นการประกาศตัวแปรที่ใช้รับค่าที่ส่งกลับมาจากการเรียกใช้เมทธอด และส่วนที่ 1.2 เป็นการรับค่ามาเก็บไว้ในตัวแปรที่ประกาศไว้ ชื่อเมทธอดที่เรียกใช้จะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ r ซึ่งเป็นกรณีตัวอย่างของคลาส Request ชื่อตัวแปรจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ rv ซึ่งเป็นกรณีตัวอย่างของคลาส ReturnVar เมื่อประยุกต์ใช้กฎ ชื่อตัวแปรจะถูกนำไปแทนที่ rv.name ในแถวที่ 3 ประเภทของตัวแปรจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ประเภทของตัวแปรจะถูกนำไปแทนที่ c.name ในแถวที่ 3 $LOCALVAR\_DECL_{0}$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศตัวแปร

2) ส่วนที่ 1.3 – กรณีที่ไม่มีตัวแปรที่ใช้รับค่าที่ส่งกลับมาจากการเรียกใช้เมทธอดที่ต้องการประกาศ $ASSIGN_{0}$ จะถูกแทนที่ด้วยเซตว่าง

3) ส่วนที่ 1.4 จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_{0}$ ด้วยเซตว่าง เพื่อจบการสร้างชุดคำสั่งภาษาจาวา

4.5 เมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้



รูปที่ 4.5 แสดงเมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้

คำอธิบายเมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้แสดงดังรูปที่ 4.5 มีดังนี้

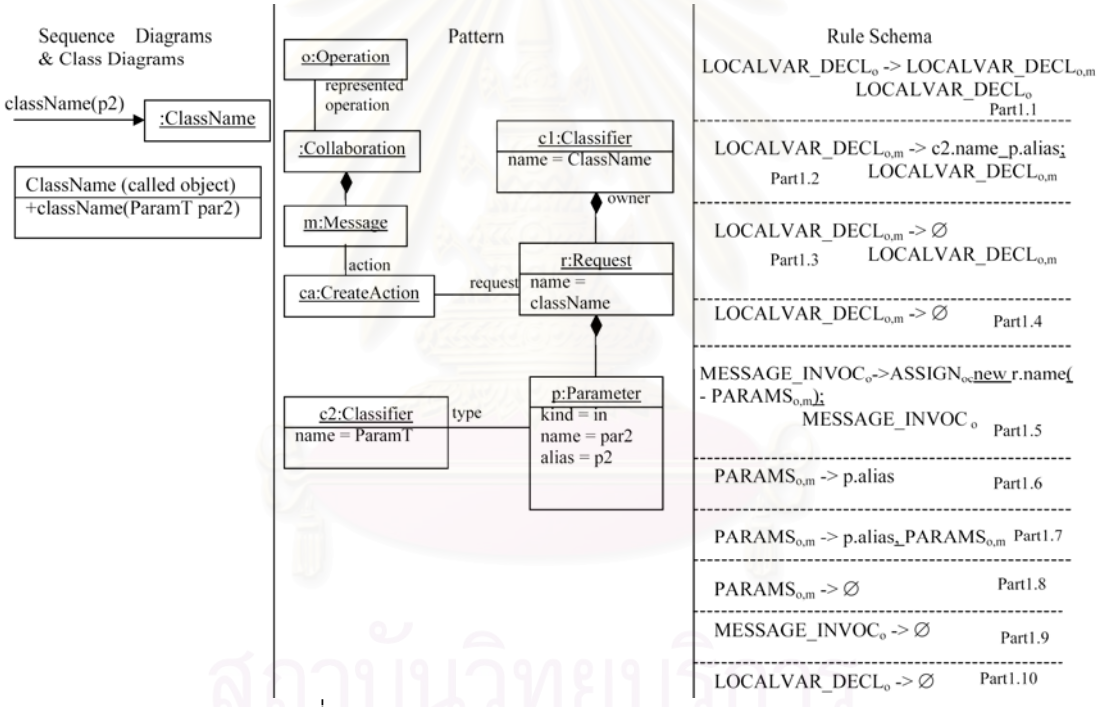
1) ส่วนที่ 1.1 และ 1.2 – ส่วนที่ 1.1 เป็นการประกาศตัวแปรที่ใช้รับค่าที่ส่งกลับมาจากการเรียกใช้ตัวสร้าง และส่วนที่ 1.2 เป็นการรับค่ามาเก็บไว้ในตัวแปรที่ประกาศไว้ ชื่อตัวสร้างที่เรียกใช้จะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ r ซึ่งเป็นกรณีตัวอย่างของคลาส Request ชื่อตัวแปรจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ rv ซึ่งเป็นกรณีตัวอย่างของ

คลาส ReturnVar เมื่อประยุกต์ใช้กฎ ชื่อตัวแปรจะถูกนำไปแทนที่ rv.name ในแถวที่ 3 ประเภทของตัวแปรจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ประเภทของตัวแปรจะถูกนำไปแทนที่ c.name ในแถวที่ 3 $LOCALVAR\_DECL_{o}$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศตัวแปร

2) ส่วนที่ 1.3 – กรณีที่ไม่มีตัวแปรที่ใช้รับค่าที่ส่งกลับมาจากการเรียกใช้ตัวสร้างที่ต้องการประกาศ $ASSIGN_{oc}$ จะถูกแทนที่ด้วยเซตว่าง

3) ส่วนที่ 1.4 จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_{o}$ ด้วยเซตว่าง เพื่อจบการสร้างชุดคำสั่งภาษาจาวา

4.6 เมต้ารูลสำหรับการสร้างวัตถุใหม่



รูปที่ 4.6 แสดงเมต้ารูลสำหรับการสร้างวัตถุใหม่

คำอธิบายเมต้ารูลสำหรับการสร้างวัตถุใหม่แสดงดังรูปที่ 4.6 มีดังนี้

1) ส่วนที่ 1.1 - ส่วนนี้เป็นการแบ่งการประกาศตัวแปร จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_{o}$ ด้วยลำดับของสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_{o,m}$ และ $LOCALVAR\_DECL_{o}$

2) ส่วนที่ 1.2 ถึง 1.4 –ส่วนที่ 1.2 เป็นการประกาศตัวแปรพารามิเตอร์ที่รับ ในแถวที่ 3 $LOCALVAR\_DECL_{o}$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศตัวแปร หากไม่มีตัวแปรพารามิเตอร์ที่รับที่ต้องการประกาศแล้ว $LOCALVAR\_DECL_{o}$ จะถูกแทนที่ด้วยเซตว่างจากการประยุกต์

ใช้กฎในส่วนที่ 1.4 ในส่วนที่ 1.2 แถวที่ 2 ชื่อพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter ชื่อพารามิเตอร์ที่รับจริงจะถูกเก็บไว้ในคุณลักษณะประจำ alias ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter เมื่อประยุกต์ใช้กฎ ชื่อพารามิเตอร์ที่รับจริงจะถูกนำไปแทนที่ p.alias ข้อมูลชนิดของพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c2 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ชนิดของพารามิเตอร์ที่รับจะถูกนำไปแทนที่ c2.name ในกรณีที่พารามิเตอร์นี้เคยถูกประกาศไว้แล้ว โดยอาจถูกประกาศพารามิเตอร์ชนิดเดียวกัน หรือชนิดอื่น จะประยุกต์ใช้กฎในส่วนที่ 1.3

3) ส่วนที่ 1.5 – ส่วนนี้เป็นการประกาศการเรียกใช้ตัวสร้าง ชื่อตัวสร้างจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ r ซึ่งเป็นกรณีตัวอย่างของคลาส Request เมื่อประยุกต์ใช้กฎ ชื่อตัวสร้างจะถูกนำไปแทนที่ r.name ชื่อคลาสของตัวสร้างจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c1 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier ในแถวที่ 3 $MESSAGE\_INVOC_{o}$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศการเรียกใช้เมทธอด

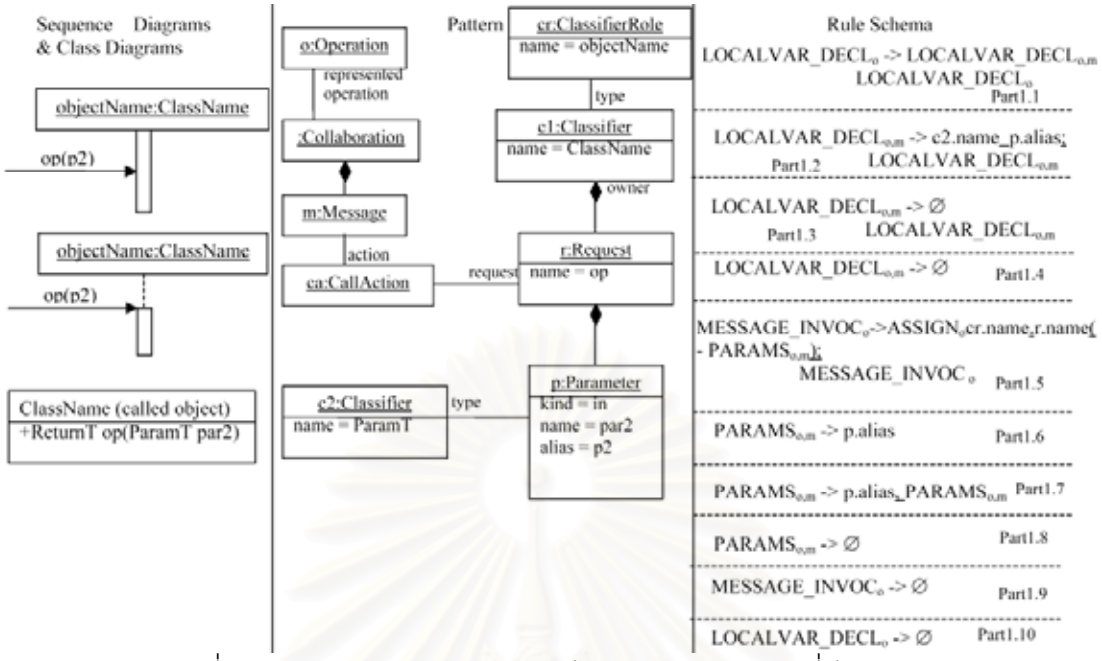
4) ส่วนที่ 1.6 ถึง 1.8 - ส่วนนี้แสดงการประกาศพารามิเตอร์ ส่วนที่ 1.6 เป็นการประกาศพารามิเตอร์ที่รับ กรณีที่ตัวสร้างมีพารามิเตอร์ที่รับตัวเดียว ส่วนที่ 1.7 เป็นการประกาศพารามิเตอร์ที่รับ กรณีที่ตัวสร้างมีพารามิเตอร์ที่รับมากกว่า 1 ตัว $PARAMS_{o}$ ตัวที่ 2 ในส่วนที่ 1.7 ใช้สำหรับการวนซ้ำประกาศพารามิเตอร์ที่รับ หากไม่มีพารามิเตอร์ที่รับที่ต้องการประกาศแล้ว $PARAMS_{o}$ จะถูกแทนที่ด้วยเซตว่างจากการประยุกต์ใช้กฎในส่วนที่ 1.8 ในแถวที่ 2 ชนิดของพารามิเตอร์จะถูกเก็บไว้ในคุณลักษณะประจำ kind ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter ข้อมูลประเภทของพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c2 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ชื่อพารามิเตอร์ที่รับจะถูกนำไปแทนที่ p.alias

5) ส่วนที่ 1.9 ถึง 1.10 จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_{o}$ และ $MESSAGE\_INVOC_{o}$ ด้วยเซตว่าง เพื่อจบการสร้างชุดคำสั่งภาษาจาวา

4.7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว

คำอธิบายเมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วแสดงดังรูปที่ 4.7 มีดังนี้

1) ส่วนที่ 1.1 - ส่วนนี้เป็นการแบ่งการประกาศตัวแปร จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_{o}$ ด้วยลำดับของสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_{o,m}$



รูปที่ 4.7 แสดงเมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว

และ LOCALVAR_DECL$_o$

2) ส่วนที่ 1.2 ถึง 1.4 –ส่วนที่ 1.2 เป็นการประกาศตัวแปรพารามิเตอร์ที่รับ ในแถวที่ 3 LOCALVAR_DECL$_o$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศตัวแปร หากไม่มีตัวแปรพารามิเตอร์ที่รับที่ต้องการประกาศแล้ว LOCALVAR_DECL$_o$ จะถูกแทนที่ด้วยเซตว่างจากการประยุกต์ใช้กฎในส่วนที่ 1.4 ในส่วนที่ 1.2 แถวที่ 2 ชื่อพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter ชื่อพารามิเตอร์ที่รับจริงจะถูกเก็บไว้ในคุณลักษณะประจำ alias ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter เมื่อประยุกต์ใช้กฎ ชื่อพารามิเตอร์ที่รับจริงจะถูกนำไปแทนที่ p.alias ข้อมูลชนิดของพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c2 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ชนิดของพารามิเตอร์ที่รับจะถูกนำไปแทนที่ c2.name ในกรณีที่พารามิเตอร์นี้เคยถูกประกาศไว้แล้ว โดยอาจถูกประกาศพารามิเตอร์ชนิดเดียวกัน หรือชนิดอื่น จะประยุกต์ใช้กฎในส่วนที่ 1.3

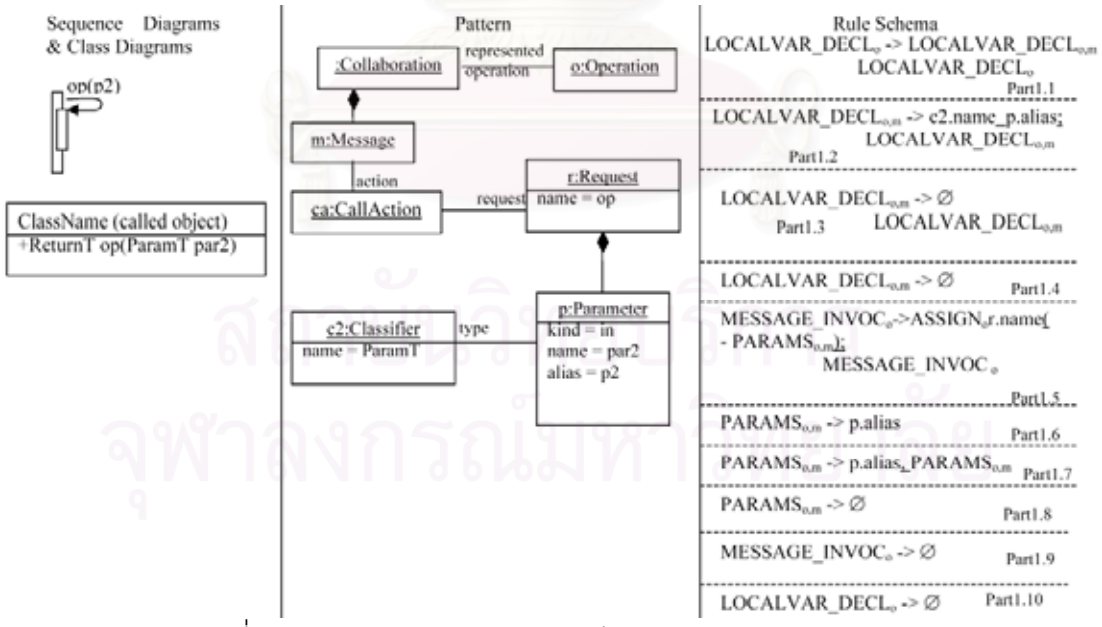
3) ส่วนที่ 1.5 – ส่วนนี้เป็นการประกาศการเรียกใช้เมทธอดของวัตถุที่มีอยู่แล้ว ชื่อเมทธอดจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ r ซึ่งเป็นกรณีตัวอย่างของคลาส Request เมื่อประยุกต์ใช้กฎ ชื่อเมทธอดจะถูกนำไปแทนที่ r.name ชื่อคลาสของเมทธอดจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c1 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier ชื่อวัตถุของเมทธอดจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ cr ซึ่งเป็นกรณีตัวอย่างของคลาส

ClassifierRole ในแถวที่ 3 $MESSAGE\_INVOC_o$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศการเรียกใช้เมทธอด

4) ส่วนที่ 1.6 ถึง 1.8 - ส่วนนี้แสดงการประกาศพารามิเตอร์ ส่วนที่ 1.6 เป็นการประกาศพารามิเตอร์ที่รับ กรณีที่ตัวสร้างมีพารามิเตอร์ที่รับตัวเดียว ส่วนที่ 1.7 เป็นการประกาศพารามิเตอร์ที่รับ กรณีที่ตัวสร้างมีพารามิเตอร์ที่รับมากกว่า 1 ตัว $PARAMS_o$ ตัวที่ 2 ในส่วนที่ 1.7 ใช้สำหรับการวนซ้ำประกาศพารามิเตอร์ที่รับ หากไม่มีพารามิเตอร์ที่รับที่ต้องการประกาศแล้ว $PARAMS_o$ จะถูกแทนที่ด้วยเซตว่างจากการประยุกต์ใช้กฎในส่วนที่ 1.8 ในแถวที่ 2 ชนิดของพารามิเตอร์จะถูกเก็บไว้ในคุณลักษณะประจำ kind ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter ข้อมูลประเภทของพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c2 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ชื่อพารามิเตอร์ที่รับจะถูกนำไปแทนที่ p.alias

5) ส่วนที่ 1.9 ถึง 1.10 จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ $LOCALVAR\_DECL_o$ และ $MESSAGE\_INVOC_o$ ด้วยเซตว่าง เพื่อจบการสร้างชุดคำสั่งภาษาจาวา

4.8 เมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเอง



รูปที่ 4.8 แสดงเมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเอง

คำอธิบายเมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วแสดงดังรูปที่ 4.8 มีดังนี้

1) ส่วนที่ 1.1 - ส่วนนี้เป็นการแบ่งการประกาศตัวแปร จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ LOCALVAR_DECL$_{o}$ ด้วยลำดับของสัญลักษณ์ที่ไม่ถึงจุดจบ LOCALVAR_DECL$_{o,m}$ และ LOCALVAR_DECL$_{o}$

2) ส่วนที่ 1.2 ถึง 1.4 –ส่วนที่ 1.2 เป็นการประกาศตัวแปรพารามิเตอร์ที่รับ ในแถวที่ 3 LOCALVAR_DECL$_{o}$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศตัวแปร หากไม่มีตัวแปรพารามิเตอร์ที่รับที่ต้องการประกาศแล้ว LOCALVAR_DECL$_{o}$ จะถูกแทนที่ด้วยเซตว่างจากการประยุกต์ใช้กฎในส่วนที่ 1.4 ในส่วนที่ 1.2 แถวที่ 2 ชื่อพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter ชื่อพารามิเตอร์ที่รับจริงจะถูกเก็บไว้ในคุณลักษณะประจำ alias ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter เมื่อประยุกต์ใช้กฎ ชื่อพารามิเตอร์ที่รับจริงจะถูกนำไปแทนที่ p.alias ข้อมูลชนิดของพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c2 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ชนิดของพารามิเตอร์ที่รับจะถูกนำไปแทนที่ c2.name ในกรณีที่พารามิเตอร์นี้เคยถูกประกาศไว้แล้ว โดยอาจถูกประกาศพารามิเตอร์ชนิดเดียวกัน หรือชนิดอื่น จะประยุกต์ใช้กฎในส่วนที่ 1.3

3) ส่วนที่ 1.5 – ส่วนนี้เป็นการประกาศการเรียกใช้เมทธอดของตัววัตถุเอง ชื่อเมทธอดจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ r ซึ่งเป็นกรณีตัวอย่างของคลาส Request เมื่อประยุกต์ใช้กฎ ชื่อเมทธอดจะถูกนำไปแทนที่ r.name ในแถวที่ 3 MESSAGE_INVOC$_{o}$ ตัวที่ 2 ใช้สำหรับการวนซ้ำประกาศการเรียกใช้เมทธอด

4) ส่วนที่ 1.6 ถึง 1.8 - ส่วนนี้แสดงการประกาศพารามิเตอร์ ส่วนที่ 1.6 เป็นการประกาศพารามิเตอร์ที่รับ กรณีที่ตัวสร้างมีพารามิเตอร์ที่รับตัวเดียว ส่วนที่ 1.7 เป็นการประกาศพารามิเตอร์ที่รับ กรณีที่ตัวสร้างมีพารามิเตอร์ที่รับมากกว่า 1 ตัว PARAMS$_{o}$ ตัวที่ 2 ในส่วนที่ 1.7 ใช้สำหรับการวนซ้ำประกาศพารามิเตอร์ที่รับ หากไม่มีพารามิเตอร์ที่รับที่ต้องการประกาศแล้ว PARAMS$_{o}$ จะถูกแทนที่ด้วยเซตว่างจากการประยุกต์ใช้กฎในส่วนที่ 1.8 ในแถวที่ 2 ชนิดของพารามิเตอร์จะถูกเก็บไว้ในคุณลักษณะประจำ kind ของวัตถุ p ซึ่งเป็นกรณีตัวอย่างของคลาส Parameter ข้อมูลประเภทของพารามิเตอร์ที่รับจะถูกเก็บไว้ในคุณลักษณะประจำ name ของวัตถุ c2 ซึ่งเป็นกรณีตัวอย่างของคลาส Classifier เมื่อประยุกต์ใช้กฎ ชื่อพารามิเตอร์ที่รับจะถูกนำไปแทนที่ p.alias

5) ส่วนที่ 1.9 ถึง 1.10 จะทำการแทนสัญลักษณ์ที่ไม่ถึงจุดจบ LOCALVAR_DECL$_{o}$ และ MESSAGE_INVOC$_{o}$ ด้วยเซตว่าง เพื่อจบการสร้างชุดคำสั่งภาษาจาวา

บทที่ 5
การใช้งานเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม

เนื้อหาในบทนี้เป็นการอธิบายวิธีการใช้งานเครื่องมือที่ประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา (รายละเอียดของการพัฒนาเครื่องมือดังแสดงในภาคผนวก ข) ซึ่งเริ่มต้นด้วยการเรียกใช้เครื่องมือ และคำสั่งพื้นฐานต่างๆ บนรายการหลัก (Main menu) หลังจากนั้นจะแสดงถึงการนำเข้าข้อมูลยูเอ็มแอลซีเควนซ์ไดอะแกรม และคลาสไดอะแกรม และวิธีการสร้างชุดคำสั่งภาษาจาวา

5.1 การเรียกใช้ และคำสั่งพื้นฐาน

การเรียกใช้เครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม สามารถทำได้ดังนี้

1) เรียกใช้โดยตรงจากการดับเบิ้ลคลิก (Double-click) ที่ไอคอน (Icon) SequenceDia2Java.exe แสดงดังรูปที่ 5.1

รูปที่ 5.1 แสดงไอคอนของเครื่องมือที่ประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรม

2) เรียกใช้โดยคลิกที่ปุ่ม Start -> Run แล้วพิมพ์ชื่อไดเรคทอรี (Directory) ที่โปรแกรม SequenceDia2Java อยู่ ตามด้วยคำว่า \SequenceDia2Java

คำสั่งพื้นฐานบนรายการหลักมีดังนี้

1) File – เมนูไฟล์ เมนูไฟล์สามารถเข้าถึงได้ด้วยการคลิกที่เมนูบาร์ (Menu bar) หรือกด Alt+f เมนูไฟล์ประกอบไปด้วยคำสั่งย่อยคือ

1.1) New – คำสั่งใหม่ คำสั่งนี้ใช้เพื่อเริ่มต้นการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาอีกครั้ง คำสั่งนี้สามารถเข้าถึงได้ด้วยการเลือกเมนูไฟล์ แล้วเลือกคำสั่งใหม่ หรือเลือกที่เมนูไฟล์แล้วกดตัวอักษร n

1.2) Save – คำสั่งบันทึก คำสั่งนี้ใช้เพื่อบันทึกชุดคำสั่งภาษาจาวาที่ได้จากการใช้เครื่องมือลงในไฟล์ ในกรณีที่ทำการบันทึกครั้งแรก คำสั่งบันทึก

จะทำงานเหมือนคำสั่งบันทึกเป็น คำสั่งนี้สามารถเข้าถึงได้ด้วยการเลือกเมนูไฟล์ แล้วเลือกคำสั่งบันทึก หรือเลือกที่เมนูไฟล์แล้วกดตัวอักษร s

1.3) Save As – คำสั่งบันทึกเป็น คำสั่งนี้ใช้เพื่อบันทึกชุดคำสั่งภาษาจาวาที่ได้จากการใช้เครื่องมือลงในไฟล์อีกไฟล์หนึ่ง คำสั่งนี้สามารถเข้าถึงได้ด้วยการเลือกเมนูไฟล์ แล้วเลือกคำสั่งบันทึกเป็น หรือเลือกที่เมนูไฟล์แล้วกดตัวอักษร a

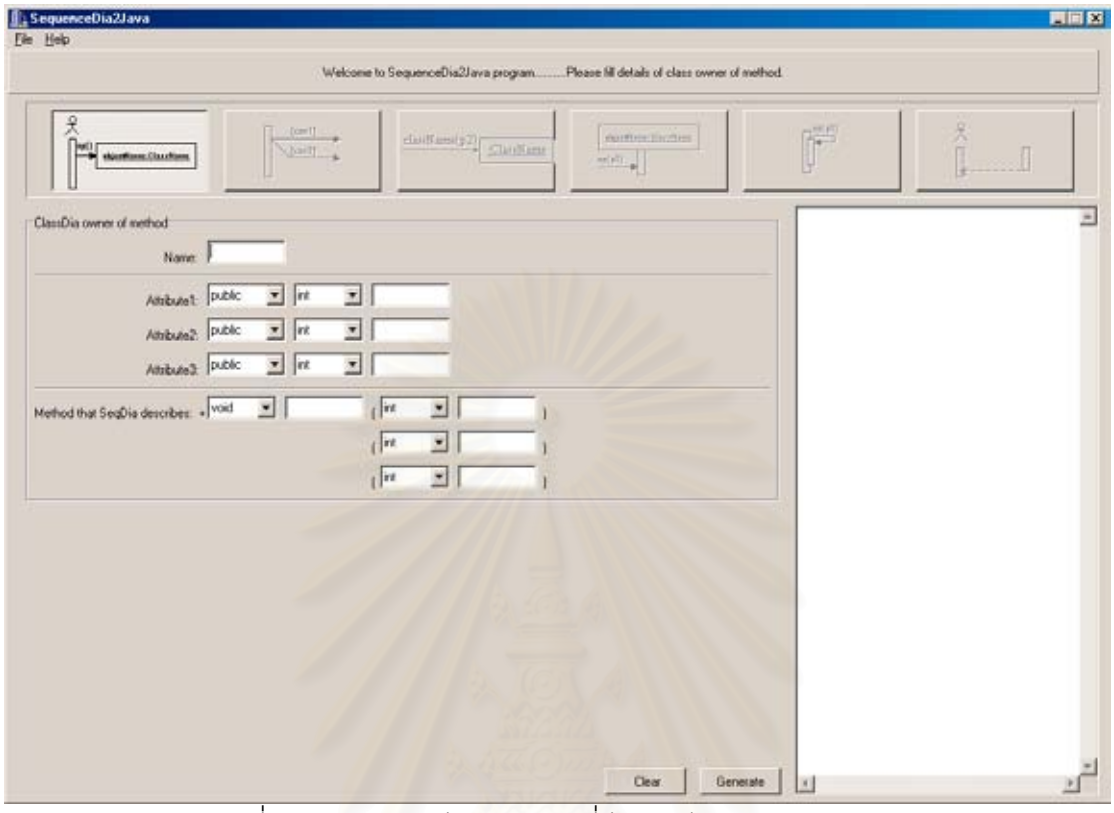
1.4) Exit – คำสั่งออก คำสั่งนี้ใช้เพื่อออกจากเครื่องมือ คำสั่งนี้สามารถเข้าถึงได้ด้วยการเลือกเมนูไฟล์ แล้วเลือกคำสั่งออก หรือเลือกที่เมนูไฟล์แล้วกดตัวอักษร x

2) Help – เมนูช่วยเหลือ เมนูช่วยเหลือสามารถเข้าถึงได้ด้วยการคลิกที่เมนูบาร์ (Menu bar) หรือกด Alt+h เมนูช่วยเหลือประกอบไปด้วยคำสั่งย่อยคือ About – คำสั่งเกี่ยวกับ คำสั่งนี้ใช้เพื่อแสดงข้อมูลเกี่ยวกับเครื่องมือที่ประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรม คำสั่งนี้สามารถเข้าถึงได้ด้วยการเลือกเมนูช่วยเหลือ แล้วเลือกคำสั่งเกี่ยวกับ หรือเลือกที่เมนูช่วยเหลือแล้วกดที่ตัวอักษร a

5.2 การนำเข้าข้อมูลยูเอ็มแอลซีเควนซ์ไดอะแกรม และคลาสไดอะแกรม และการสร้างชุดคำสั่งภาษาจาวา

ตลอดการใช้เครื่องมือจะมีคำแนะนำการใช้แสดงที่ส่วนบนสุดถัดจากรายการหลัก การนำเข้าข้อมูลยูเอ็มแอลซีเควนซ์ไดอะแกรม และคลาสไดอะแกรม ผู้ใช้สามารถทำได้โดยการอ่านข้อมูลซีเควนซ์ไดอะแกรมทีละส่วนจากซ้ายมาขวา และจากบนลงล่าง จากนั้นเลือกปุ่มที่มีรูปภาพที่ใกล้เคียงกับส่วนของซีเควนซ์ไดอะแกรมนั้นที่สุด แล้วจึงกรอกข้อมูลซีเควนซ์ไดอะแกรมส่วนนั้น และคลาสไดอะแกรมลงในกล่องข้อความ (Text box) ที่เว้นว่างไว้ให้ เมื่อนำเมาส์ไปวางเหนือทุกปุ่มกด และกล่องข้อความจะมีคำอธิบายสำหรับแต่ละปุ่มกด หรือกล่องข้อความนั้นๆ ปรากฏขึ้น ในกรณีที่ทำการกรอกผิด ต้องการล้างข้อมูลในกล่องข้อความทั้งหมด ให้กดปุ่มล้าง (Clear) ซึ่งเครื่องมือที่ประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรมประกอบด้วยหน้าต่างๆ ที่ต้องทำการกรอกข้อมูลดังนี้

1) การเรียกเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายดังรูปที่ 5.2 – เมื่อกดปุ่มที่มีคำอธิบาย Calling method that sequence diagram describes หน้านี้เป็นหน้าแรกสำหรับการเริ่มต้นใช้เครื่องมือ ซึ่งผู้ใช้ต้องกรอกข้อมูลรายละเอียดของคลาสของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายก่อน



รูปที่ 5.2 แสดงการเรียกเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย

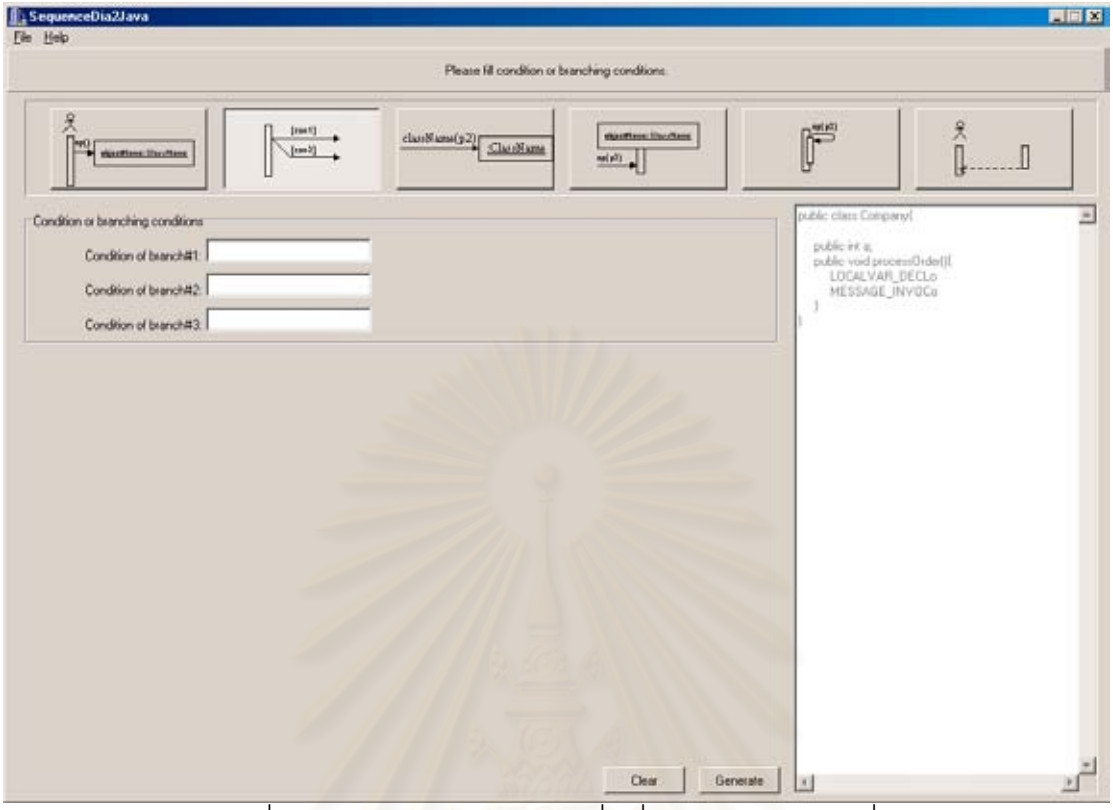
ข้อมูลที่ผู้ใช้ต้องกรอกประกอบด้วยส่วนหลักๆ คือข้อมูลคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย ซึ่งมีข้อมูลย่อยคือ

1.1) ข้อมูลชื่อคลาส (Name)

1.2) ข้อมูลคุณลักษณะประจำ (Attribute) – ระดับการมองเห็น, ชนิดของคุณลักษณะประจำ และชื่อคุณลักษณะประจำ

1.3) ข้อมูลเมทธอด (Method that SeqDia describes) – ประเภทของพารามิเตอร์ที่ส่ง ชื่อเมทธอด ประเภทของพารามิเตอร์ที่รับ และชื่อพารามิเตอร์ที่รับ

2) การเรียกเมทธอดที่มีเงื่อนไข หรือการแตกกิ่งดังรูปที่ 5.3 – เมื่อกดปุ่มที่มีคำอธิบาย Conditional method or branching



รูปที่ 5.3 แสดงการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง

ข้อมูลที่ผู้ใช้ต้องกรอกคือ ประโยคเงื่อนไขสำหรับเมทธอดที่มีเงื่อนไข หรือการแตกกิ่ง (Condition of branch)

3) การสร้างวัตถุใหม่ดังรูปที่ 5.4 – เมื่อกดปุ่มที่มีคำอธิบาย New object

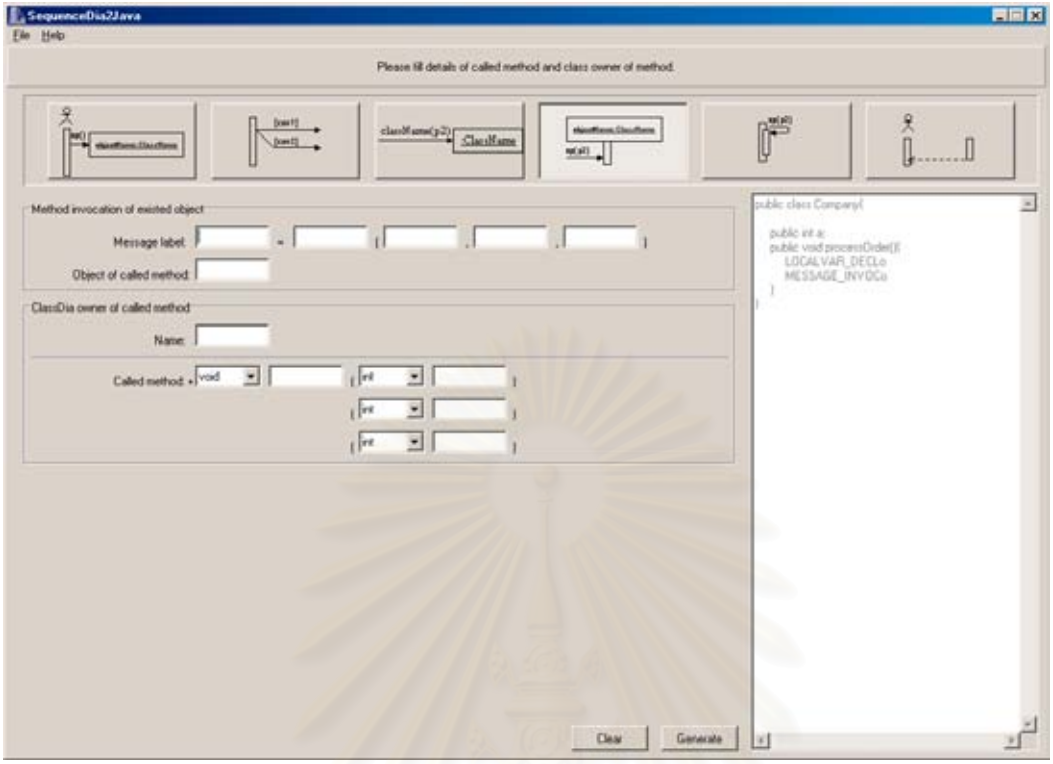
ข้อมูลที่ผู้ใช้ต้องกรอกประกอบด้วยส่วนหลักๆ คือข้อมูลการเรียกตัวสร้างตามที่วาดไว้ในซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมของตัวสร้าง ซึ่งมีข้อมูลย่อยคือ

3.1) ข้อมูลคำอธิบายข้อความ (Message label) – ชื่อตัวแปรที่ใช้รับค่าที่ส่งกลับมาจากการเรียกใช้ตัวสร้าง ชื่อตัวสร้าง และชื่อพารามิเตอร์ที่รับ

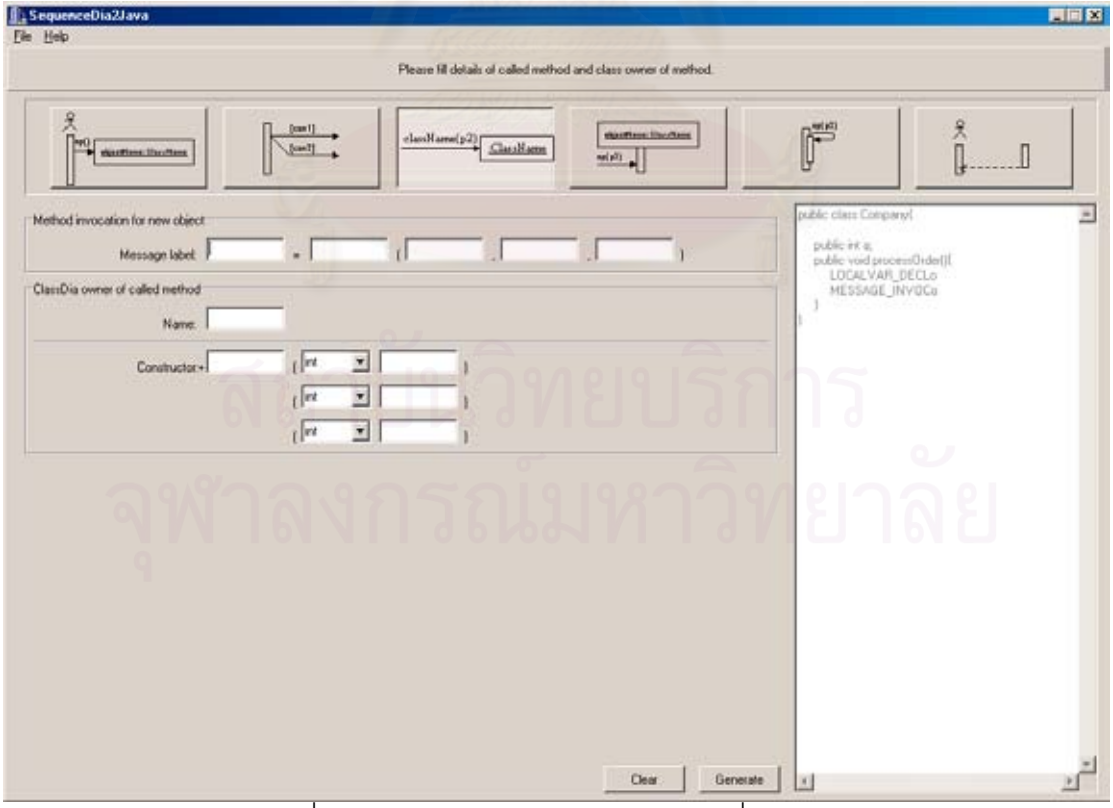
3.2) ข้อมูลชื่อคลาส (Name)

3.3) ข้อมูลเมทธอดตัวสร้าง (Constructor) – ชื่อตัวสร้าง ประเภทของพารามิเตอร์ที่รับ และชื่อพารามิเตอร์ที่รับ

4) การเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังรูปที่ 5.5 – เมื่อกดปุ่มที่มีคำอธิบาย Method invocation on existed object



รูปที่ 5.4 แสดงการสร้างวัตถุใหม่



รูปที่ 5.5 แสดงการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว

ข้อมูลที่ผู้ใช้ต้องกรอกประกอบด้วยส่วนหลักๆ คือข้อมูลการเรียกเมทธอดของวัตถุที่มีอยู่แล้วตามที่วาดไว้ในซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมของเมทธอดที่ถูกเรียก ซึ่งมีข้อมูลย่อยคือ

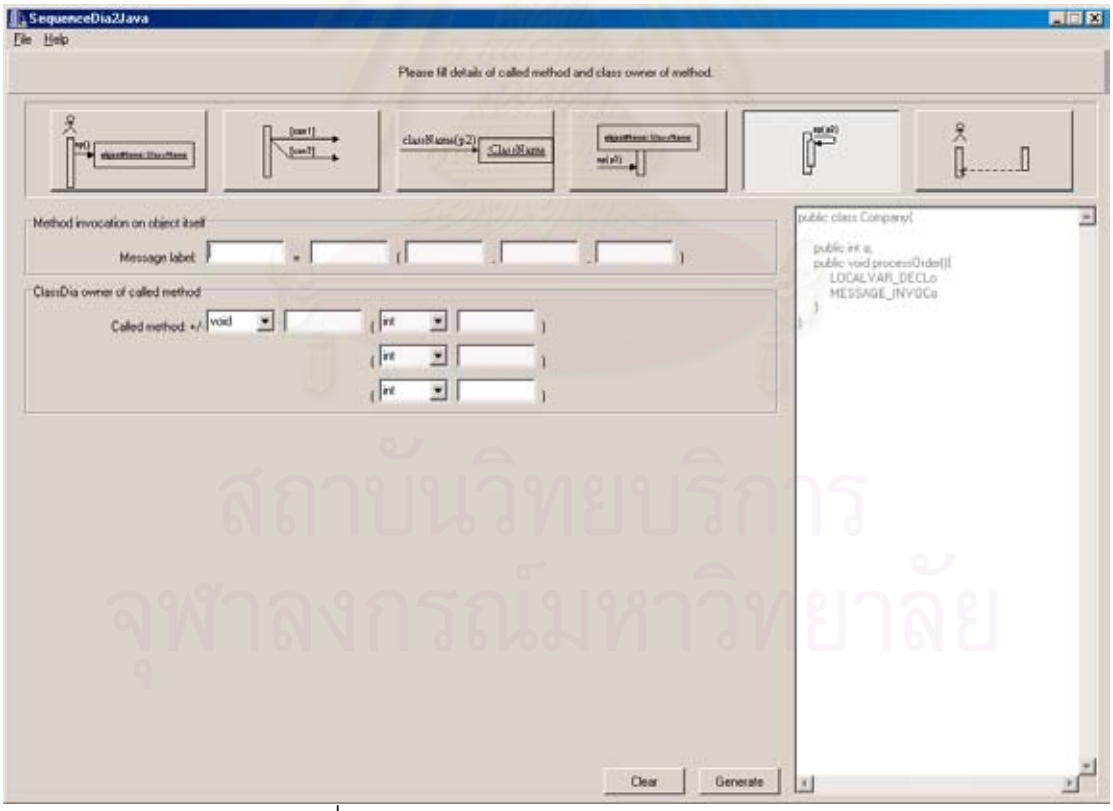
4.1) ข้อมูลคำอธิบายข้อความ (Message label) – ชื่อตัวแปรที่ใช้รับค่าที่ส่งกลับมาจากการเรียกใช้เมทธอด ชื่อเมทธอด และชื่อพารามิเตอร์ที่รับ

4.2) ข้อมูลชื่อวัตถุ (Object of called method)

4.3) ข้อมูลชื่อคลาส (Name)

4.4) ข้อมูลเมทธอดที่ถูกเรียก (Called method) – ประเภทของพารามิเตอร์ที่ส่ง ชื่อเมทธอด ประเภทของพารามิเตอร์ที่รับ และชื่อพารามิเตอร์ที่รับ

5) การเรียกเมทธอดของของตัววัตถุเองดังรูปที่ 5.6 – เมื่อกดปุ่มที่มีคำอธิบาย Method invocation on object itself



รูปที่ 5.6 แสดงการเรียกเมทธอดของตัววัตถุเอง

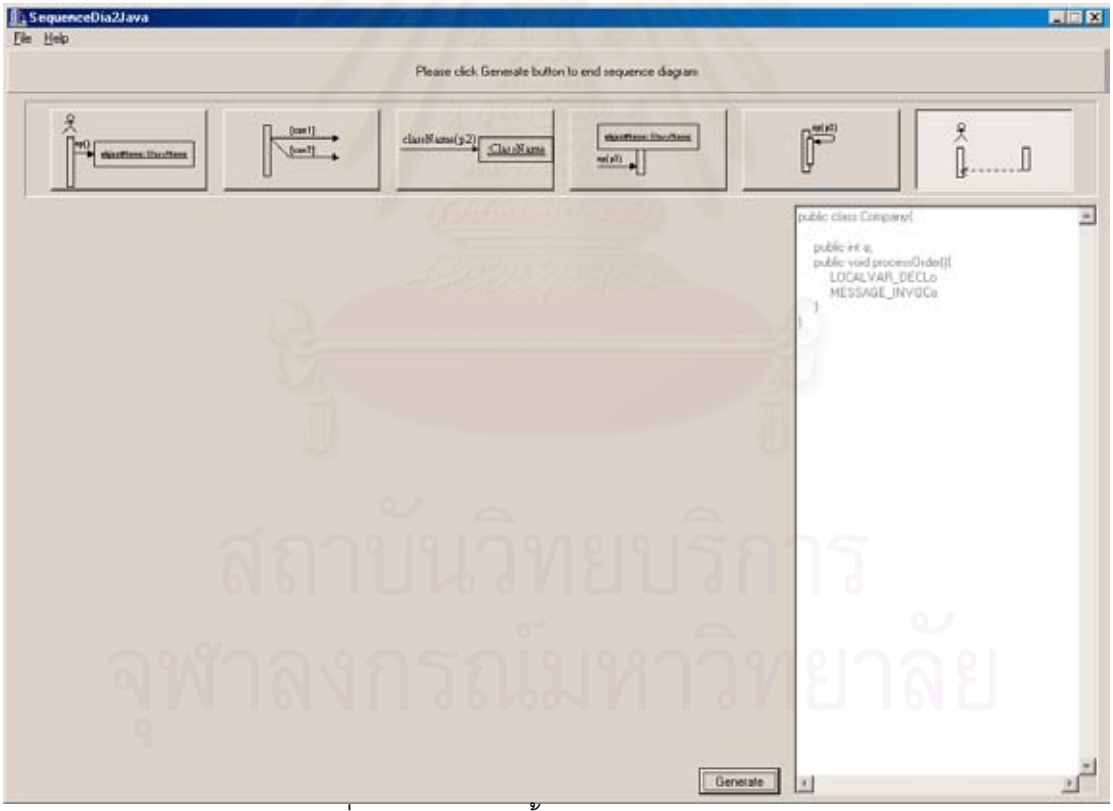
ข้อมูลที่ผู้ใช้ต้องกรอกประกอบด้วยส่วนหลักๆ คือข้อมูลการเรียกเมทธอดของตัววัตถุเองตามที่วาดไว้ในซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมของเมทธอดที่ถูกเรียก ซึ่งมีข้อมูลย่อยคือ

5.1) ข้อมูลคำอธิบายข้อความ (Message label) – ชื่อตัวแปรที่ใช้รับค่าที่ส่งกลับมาจากการเรียกใช้เมทธอด ชื่อเมทธอด และชื่อพารามิเตอร์ที่รับ

5.2) ข้อมูลชื่อคลาส (Name)

5.3) ข้อมูลเมทธอดที่ถูกเรียก (Called method) – ประเภทของพารามิเตอร์ที่ส่ง ชื่อเมทธอด ประเภทของพารามิเตอร์ที่รับ และชื่อพารามิเตอร์ที่รับ

6) สิ้นสุดซีเควนซ์ไดอะแกรมดังรูปที่ 5.7 – เมื่อกดปุ่มที่มีคำอธิบาย Finish sequence



รูปที่ 5.7 แสดงหน้าสิ้นสุดซีเควนซ์ไดอะแกรม

หลังจากกรอกข้อมูลลงในหน้าต่างๆ ผู้ใช้สามารถสร้างชุดคำสั่งภาษาจาวาของส่วนที่กรอกข้อมูลลงไปด้วยการกดปุ่มทำให้เกิด ในหน้านั้นๆ ชุดคำสั่งภาษาจาวาของส่วนที่กรอก

ข้อมูลลงไปจะถูกสร้างขึ้น เมื่อหมดส่วนต่างๆ ของซีเควนซ์ไดอะแกรม ให้เลือกหน้าสิ้นสุดซีเควนซ์ไดอะแกรม แล้วกดปุ่มทำให้เกิดอีกครั้ง เป็นอันสิ้นสุดการสร้างชุดคำสั่งภาษาจาวา

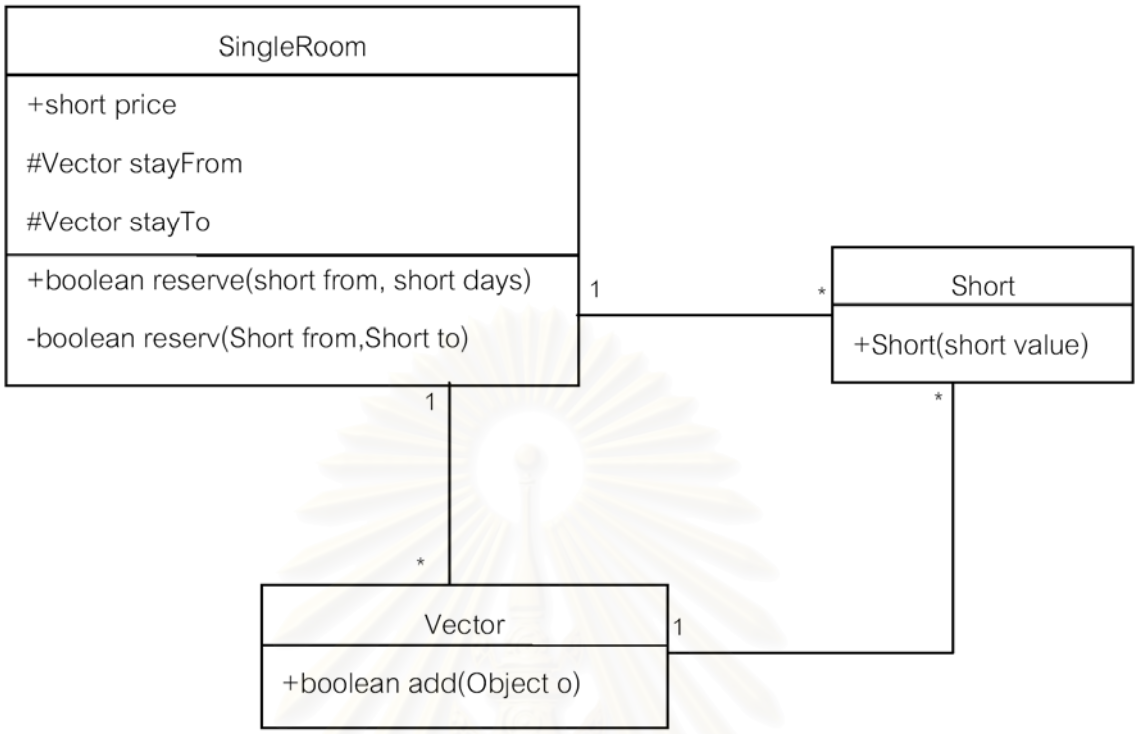
บทที่ 6
การทดสอบและการประเมินผลการออกแบบกฎแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม

ในบทนี้จะเริ่มต้นจากการทดสอบการแปลงซีเควนซ์ไดอะแกรมโดยการประยุกต์ใช้กฎ โดยทดสอบกับซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมที่จำเป็นที่เขียนขึ้นอย่างถูกต้องตรงตามมาตรฐานยูเอ็มแอล ซีเควนซ์ไดอะแกรมที่ใช้ทดสอบมี 3 ซีเควนซ์ไดอะแกรม ได้แก่ ซีเควนซ์ไดอะแกรมของเมทธอดจองของระบบห้องพัก ซีเควนซ์ไดอะแกรมของเมทธอดคืนหนังสือของระบบห้องสมุด และซีเควนซ์ไดอะแกรมของเมทธอดแสดงของระบบกองไพ่ เริ่มต้นการแปลงจะอธิบายถึงคลาส และซีเควนซ์ไดอะแกรมที่ใช้ จากนั้นจะนำเสนอการประยุกต์ใช้กฎ แล้วจึงแสดงชุดคำสั่งภาษาจาวาที่สร้างได้จากการใช้เครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม ต่อมาจะทำการประเมินผลการออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมโดยทำการเปรียบเทียบจำนวนบรรทัดคำสังที่สร้างได้และถูกนำมาใช้งานจริง กับจำนวนบรรทัดคำสั่งจากชุดคำสั่งจริง และเพื่อความง่ายในการเปรียบเทียบจำนวนบรรทัดคำสั่ง ตัวอย่างชุดคำสั่งภาษาจาวาจริงจึงถูกนำมาเขียนใหม่ให้เป็นบรรทัดย่อยๆ ก่อนการนำไปเปรียบเทียบดังจะแสดงตัวอย่างต่อไป

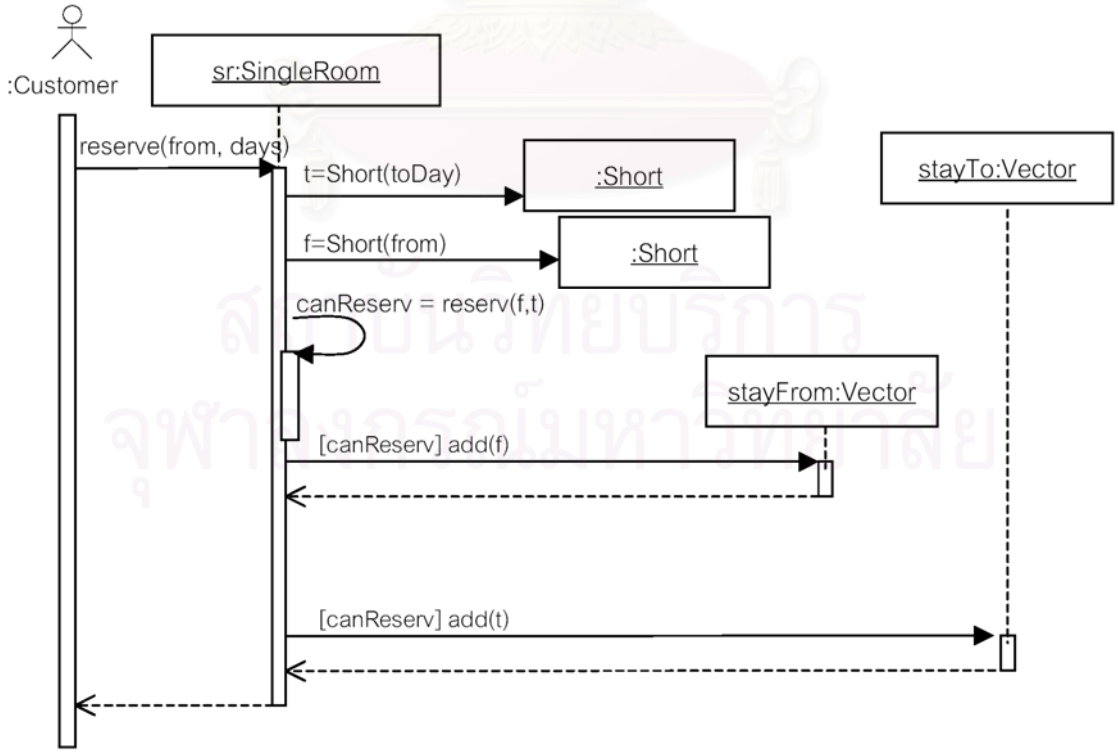
6.1 การแปลงซีเควนซ์ไดอะแกรมของเมทธอดจองของระบบห้องพัก

รูปที่ 6.1 แสดงซีเควนซ์ไดอะแกรมของเมทธอดจอง โดยเริ่มต้นจากเมื่อลูกค้าต้องการจองห้องพักจะบอกวันที่เริ่มจอง (from) และจำนวนวันที่ต้องการพัก (days) มายังระบบห้องพัก หลังจากนั้นระบบห้องพักจะทำการสร้างข้อมูลชนิดสั้นขึ้นมาใหม่เพื่อทำการเก็บวันที่ที่สิ้นสุดการจอง (toDay) ที่เกิดจากนำวันที่ที่เริ่มจองบวกจำนวนวันที่ต้องการพัก (from+days) และสร้างข้อมูลชนิดสั้นขึ้นเพื่อทำการเก็บวันที่เริ่มจอง หลังจากนั้นระบบจะทำการตรวจสอบว่าสามารถจองได้หรือไม่ (canReserv) หากจองได้จะทำการเพิ่มวันที่เริ่มจอง และจำนวนวันที่พักลงในวัตถุเริ่มพักจาก (stayFrom) และสิ้นสุดการพัก (stayTo)

รูปที่ 6.2. แสดงคลาสไดอะแกรมที่ใช้ประกอบการสร้างชุดคำสั่งของเมทธอดจอง ซึ่งห้องพัก 1 ห้อง สามารถเกี่ยวข้องกับข้อมูลเวคเตอร์ (Vector) และชนิดสั้น (Short) ได้หลายจำนวน และเวคเตอร์หนึ่งๆ สามารถเก็บข้อมูลชนิดสั้นได้หลายจำนวน



รูปที่ 6.1 แสดงคลาสไดอะแกรมที่ใช้ประกอบการสร้างชุดคำสั่งของเมทธอดจอง



รูปที่ 6.2 แสดงซีเควนซ์ไดอะแกรมของเมทธอดจองของระบบห้องพัก

การประยุกต์ใช้กฎกับซีเควนซ์ไดอะแกรมของเมทธอดจองของระบบห้องพัก และคลาสไดอะแกรมที่ใช้ประกอบการสร้างชุดคำสั่งของเมทธอดจอง มีดังต่อไปนี้

1) ประยุกต์ใช้กฎที่ 1 เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายดังรูปที่ 6.3

2) ประยุกต์ใช้กฎที่ 2 เมต้ารูลสำหรับการแบ่งซีเควนซ์ดังรูปที่ 6.4

3) ประยุกต์ใช้กฎที่ 5 เมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้ดังรูปที่ 6.5

4) ประยุกต์ใช้กฎที่ 6 เมต้ารูลสำหรับการสร้างวัตถุใหม่ดังรูปที่ 6.6

5) ประยุกต์ใช้กฎที่ 5 เมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้ดังรูปที่ 6.7

6) ประยุกต์ใช้กฎที่ 6 เมต้ารูลสำหรับการสร้างวัตถุใหม่ดังรูปที่ 6.8

7) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปรดังรูปที่ 6.9

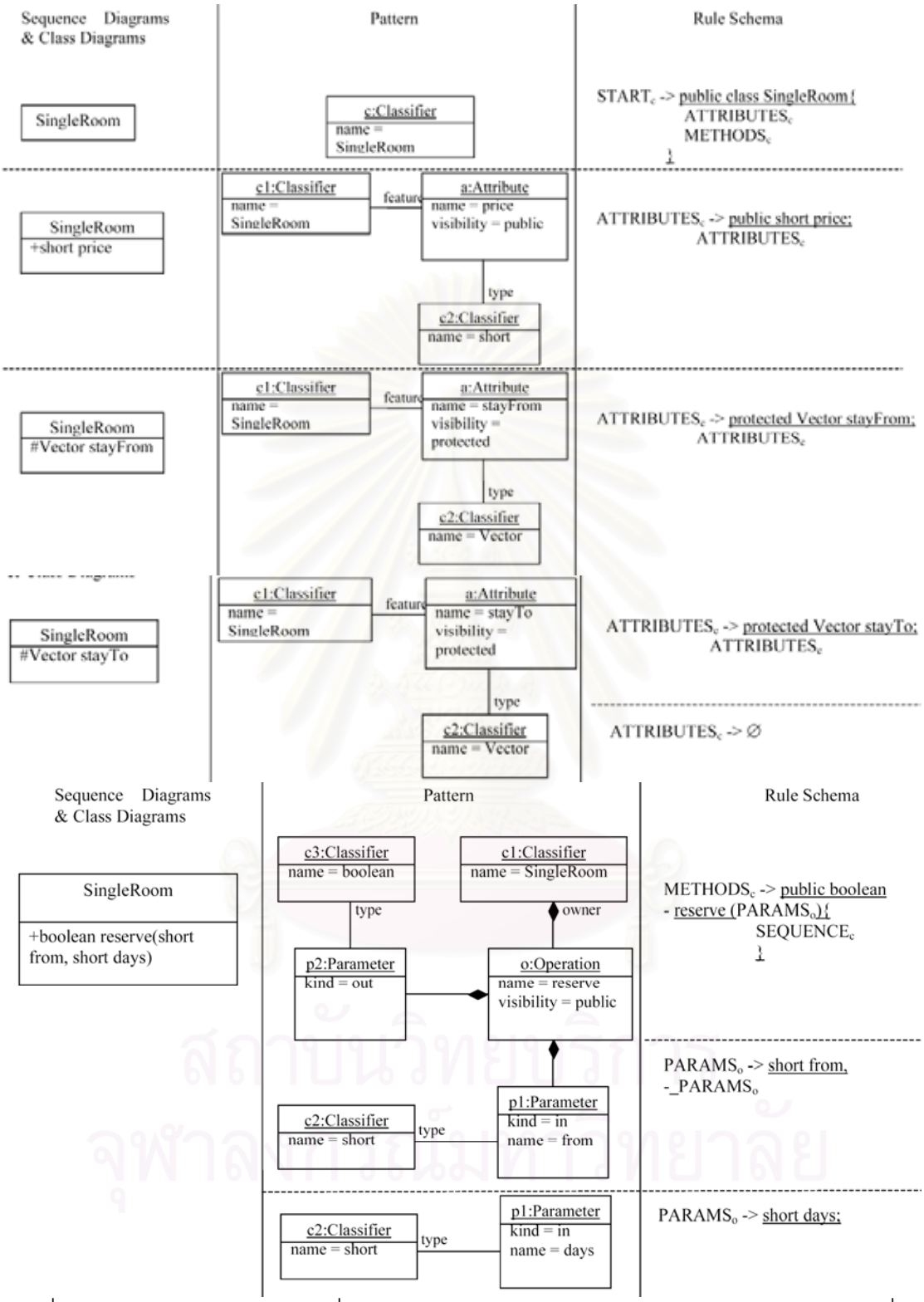
8) ประยุกต์ใช้กฎที่ 8 เมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเองดังรูปที่ 6.10

9) ประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่งดังรูปที่ 6.11

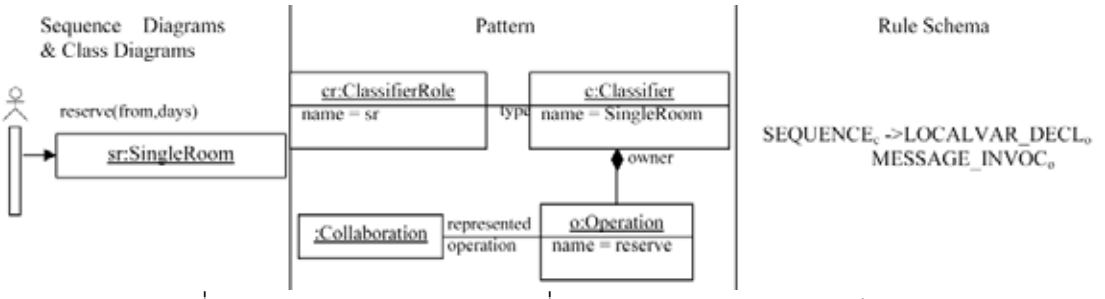
10) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังรูปที่ 6.12

11) ประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่งดังรูปที่ 6.13

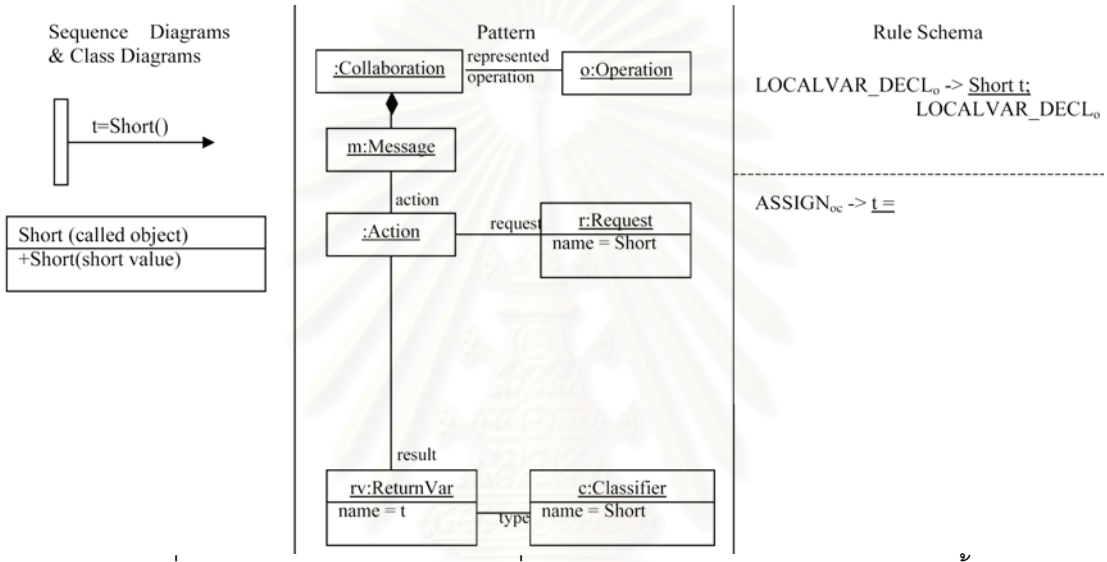
12) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังรูปที่ 6.14



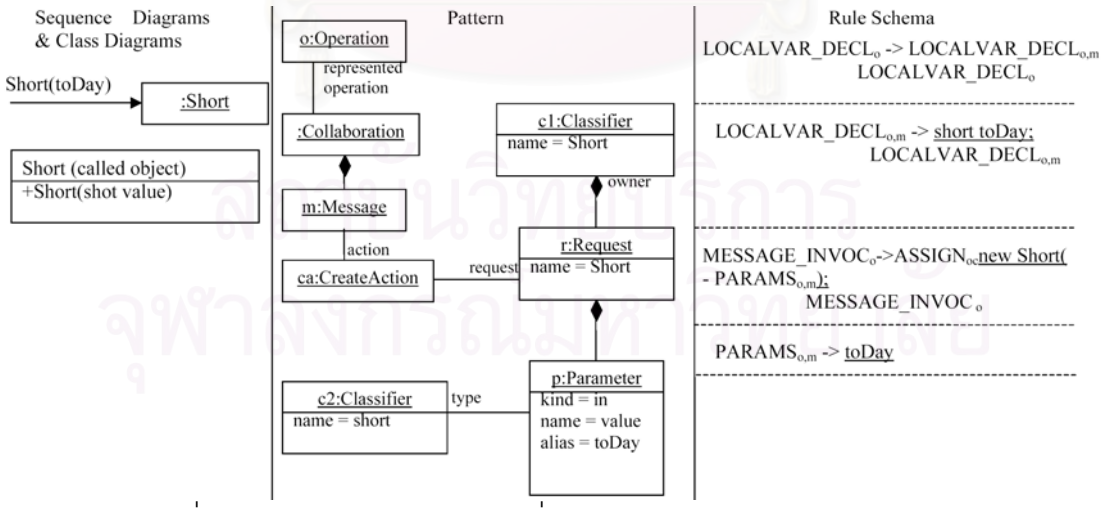
รูปที่ 6.3 แสดงการประยุกต์ใช้กฎที่ 1 เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย



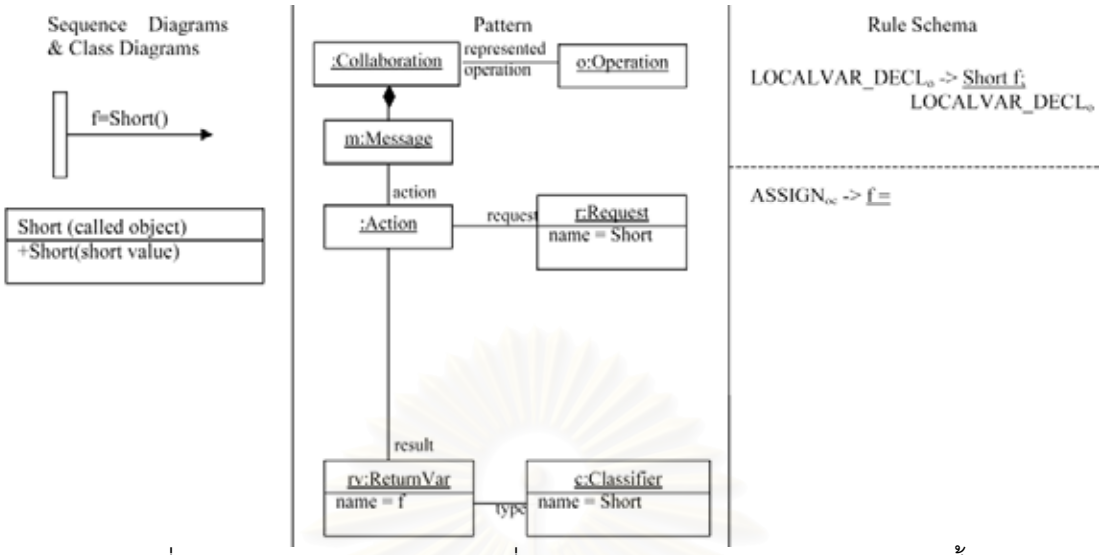
รูปที่ 6.4 แสดงการประยุกต์ใช้กฎที่ 2 เมต้ารูลสำหรับการแบ่งซีเควนซ์



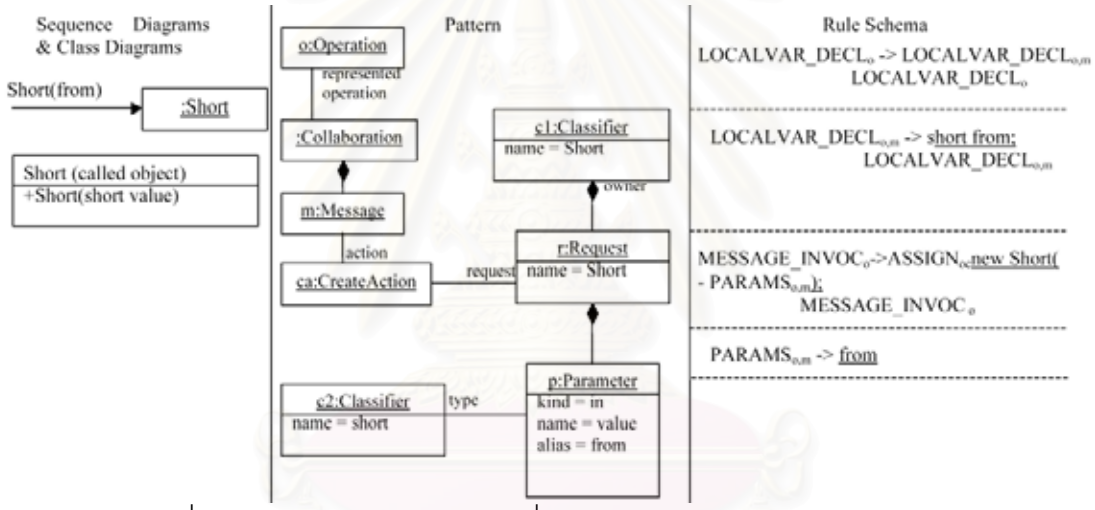
รูปที่ 6.5 แสดงการประยุกต์ใช้กฎที่ 5 เมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้



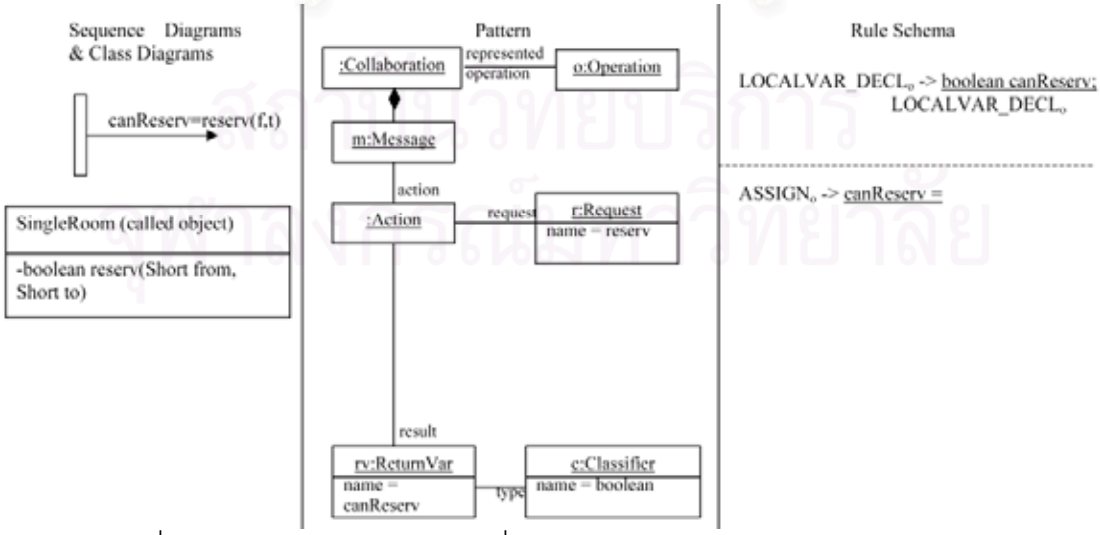
รูปที่ 6.6 แสดงการประยุกต์ใช้กฎที่ 6 เมต้ารูลสำหรับการสร้างวัตถุใหม่



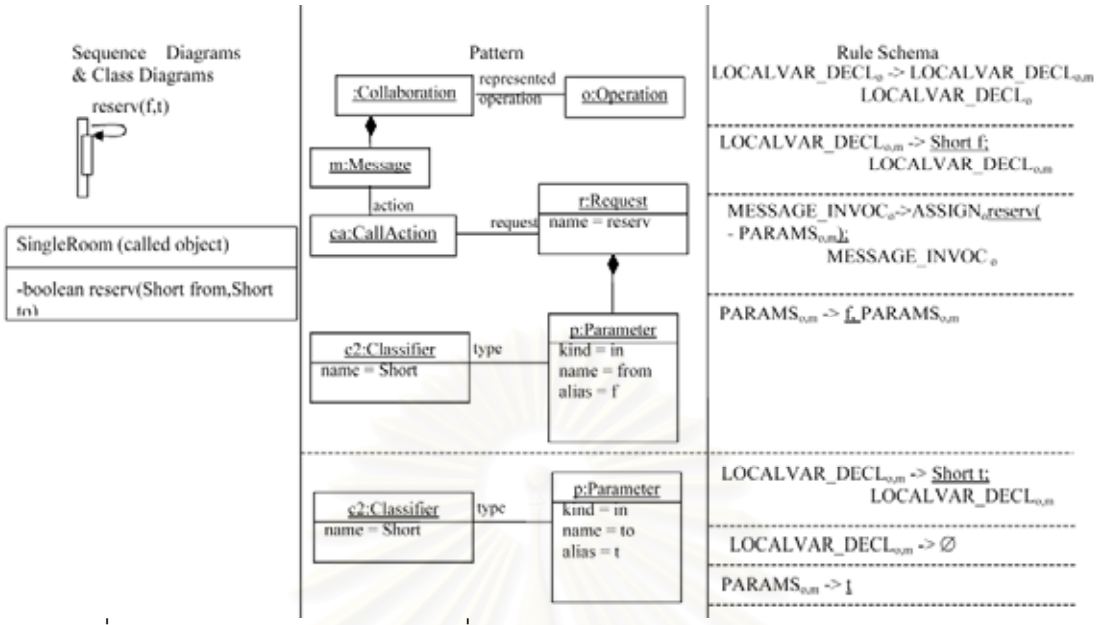
รูปที่ 6.7 แสดงการประยุกต์ใช้กฎที่ 5 เมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้



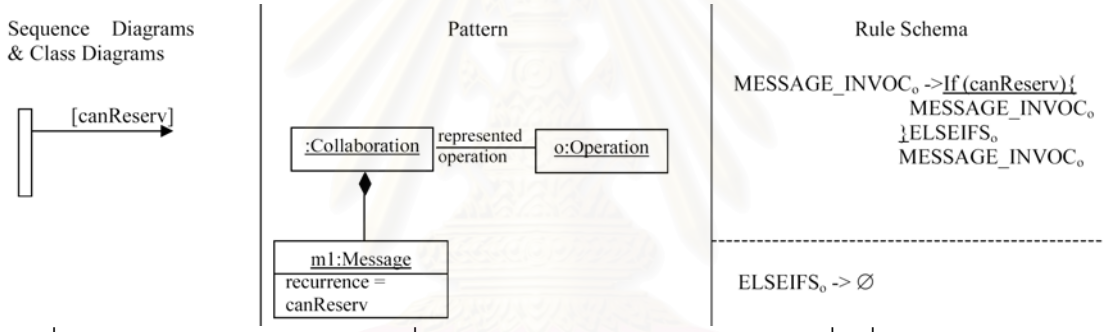
รูปที่ 6.8 แสดงการประยุกต์ใช้กฎที่ 6 เมต้ารูลสำหรับการสร้างวัตถุใหม่



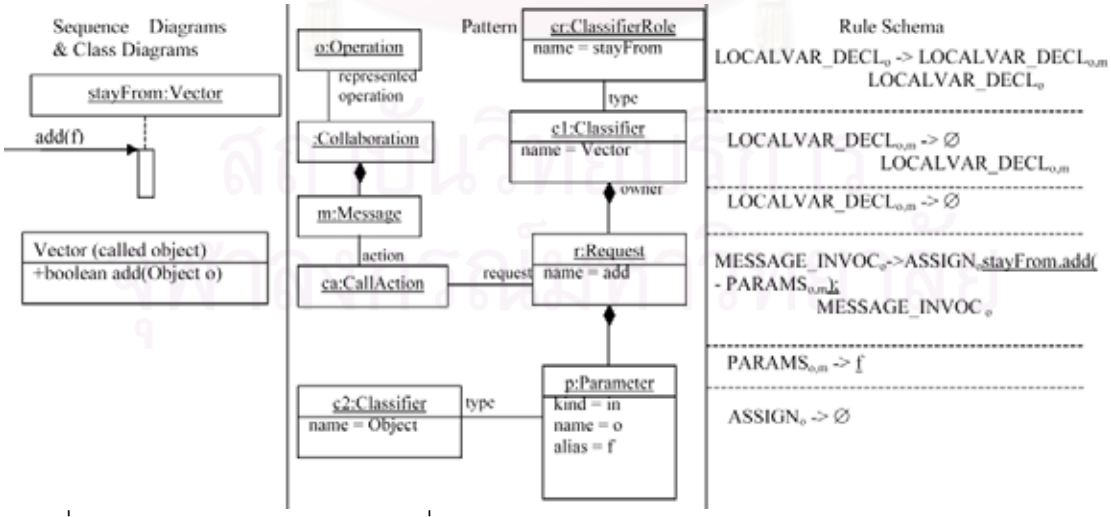
รูปที่ 6.9 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร



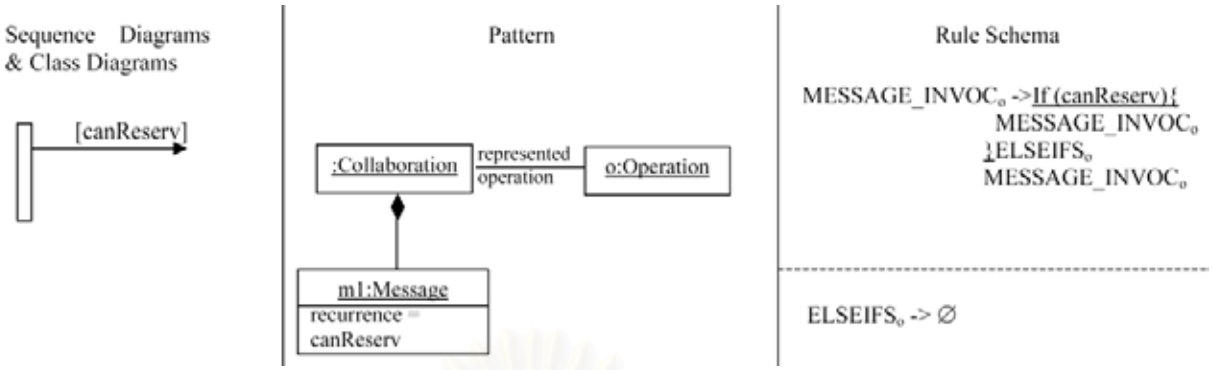
รูปที่ 6.10 แสดงการประยุกต์ใช้กฎที่ 8 เมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเอง



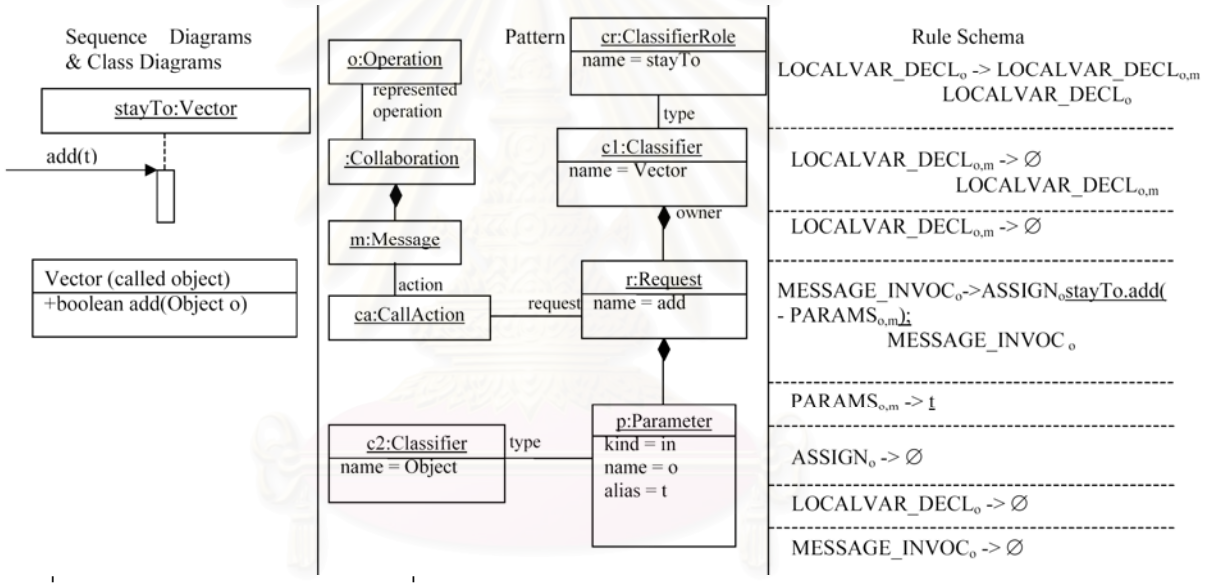
รูปที่ 6.11 แสดงการประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง



รูปที่ 6.12 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว



รูปที่ 6.13 แสดงการประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง



รูปที่ 6.14 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว

หลังจากการประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรมกับซีเควนซ์ไดอะแกรมและคลาสไดอะแกรมดังกล่าว และทำการตัดการประกาศตัวแปรที่ประกาศซ้ำซ้อน จะได้ชุดคำสั่งภาษาจาวาของเมทธอดจองดังรูปที่ 6.15

```
public class SingleRoom{
        public short price;
        protected Vector stayFrom;
        protected Vector stayTo;
        public boolean reserve(short from, shot days){
```

```
                    Short t;
                    Short f;
                    short toDay;
                    boolean canReserv;
                    t = new Short(toDay);
                    f = new Short(from);
                    canReserv = reserv(f,t);
                    if(canReserv){
                        stayFrom.add(f);
                    }
                    if(canReserv){
                        stayTo.add(t);
                    }
                }
            }
```

รูปที่ 6.15 แสดงชุดคำสั่งภาษาจาวาของเมทธอดจองที่สร้างจากเครื่องมือที่ประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรม

ตัวอย่างชุดคำสั่งภาษาจาวาจริงของเมทธอดจองของระบบห้องพักแสดงดังรูปที่ 6.16 ถูกนำมาเขียนให้เป็นบรรทัดย่อยๆ ดังรูป 6.17 เพื่อความง่ายในการเปรียบเทียบจำนวนบรรทัดคำสั่งที่สร้างได้จากการประยุกต์ใช้กฎต่อจำนวนบรรทัดคำสั่งทั้งหมด

```
import java.lang.*;
import java.util.*;
public class SingleRoom {
    public short price;
    protected Vector stayFrom;
    protected Vector stayTo;
    public boolean reserve(short from,short days){
        Short t = new Short((short)(from+days));
        Short f = new Short(from);
        boolean canReserv;
        if ((canReserv = reserv(f,t))== true){
            stayFrom.add(f);
```

```
                    stayTo.add(t);
                }
                return canReserv;
        }
}
```

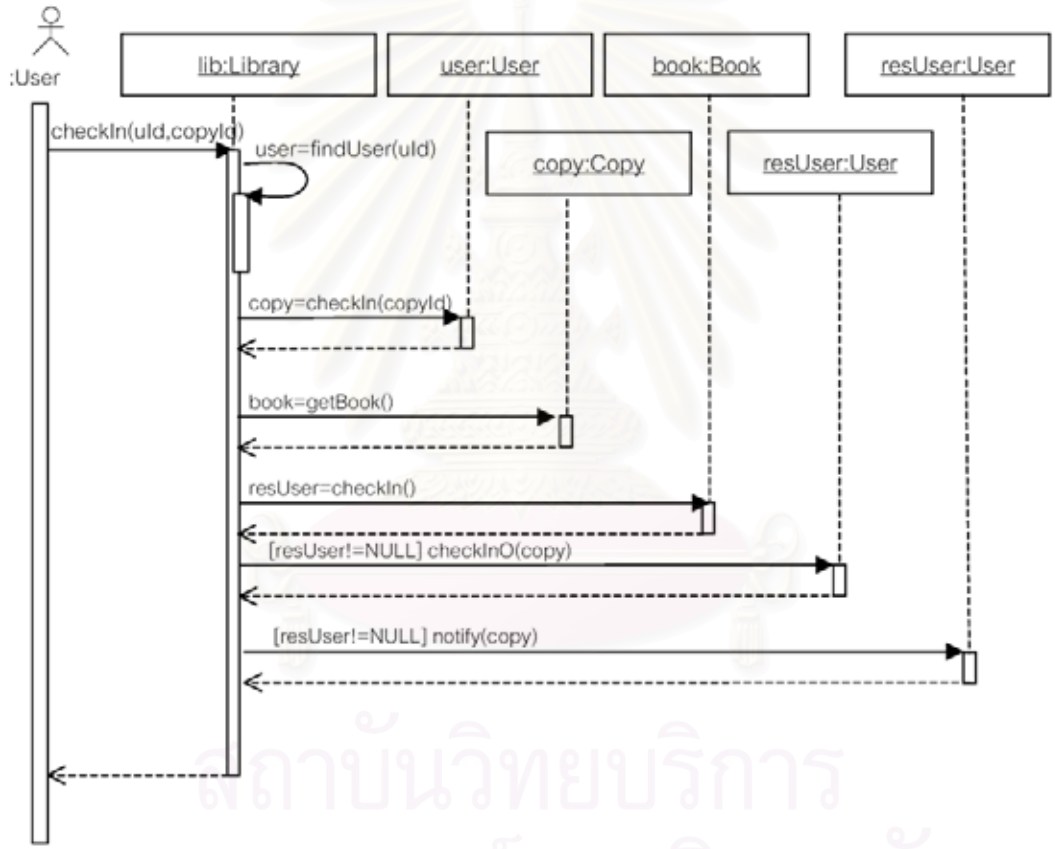
รูปที่ 6.16 แสดงตัวอย่างชุดคำสั่งภาษาจาวาของเมทธอดจองของระบบห้องพัก [11]

```
import java.lang.*;
import java.util.*;
public class SingleRoom {
        public short price;
        protected Vector stayFrom;
        protected Vector stayTo;
        public boolean reserve(short from,short days){
                Short t;
                Short f;
                boolean canReserv;
                int toDay;
                toDay = from+days;
                t = new Short((short)(toDay));
                f = new Short(from);
                canReserv = reserv(f,t);
                if(canReserv==true){
                        stayFrom.add(f);
                }
                if(canReserv==true){
                        stayTo.add(t);
                }
                return canReserv;
        }
}
```

รูปที่ 6.17 แสดงตัวอย่างชุดคำสั่งภาษาจาวาของเมทธอดจองที่ถูกนำมาเขียนใหม่

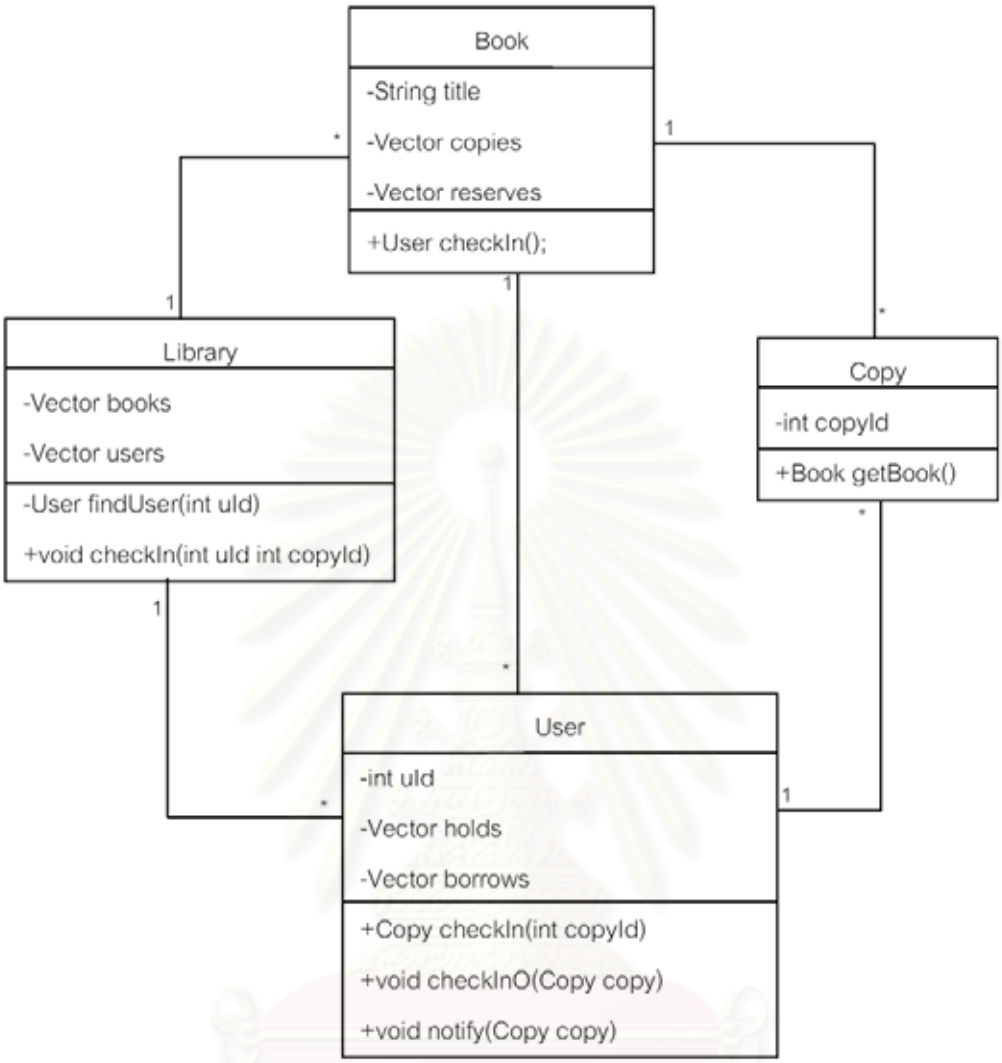
6.2 การแปลงซีเควนซ์ไดอะแกรมของเมทธอดคืนหนังสือของระบบห้องสมุด

รูปที่ 6.18 แสดงซีเควนซ์ไดอะแกรมของเมทธอดคืนหนังสือ โดยเริ่มต้นจากเมื่อผู้ใช้เอาหนังสือมาคืนจะบอกรหัสผู้ใช้ (uId) และรหัสสำเนาที่คืน (copyId) ระบบห้องสมุดจะทำการค้นผู้ใช้ เรียกเมทธอดคืนหนังสือจากวัตถุผู้ใช้ (user) แล้วจึงหาวัตถุหนังสือจากการเรียกใช้เมทธอดหาหนังสือ (getBook) จากวัตถุสำเนา (copy) หลังจากนั้นระบบเรียกเมทธอดคืนหนังสือของวัตถุหนังสือ เพื่อหาผู้จองหนังสือ (resUser) หากมีผู้จองหนังสือไว้ (resUser!=NULL) ระบบจะเรียกเมทธอดจองหนังสือ (checkInO) และทำการแจ้งเตือนผู้จองต่อไป



รูปที่ 6.18 แสดงซีเควนซ์ไดอะแกรมของเมทธอดคืนหนังสือของระบบห้องสมุด

รูปที่ 6.19 แสดงคลาสไดอะแกรมที่ใช้ประกอบการสร้างชุดคำสั่งของเมทธอดคืนหนังสือ ซึ่งห้องสมุด (Library) 1 ห้องมีหนังสือ (Book) ได้มากกว่า 1 เล่ม และผู้ใช้ (User) มากกว่า 1 คน หนังสือ 1 เล่ม สามารถมีผู้ใช้ที่ยืมได้มากกว่า 1 คน หนังสือ 1 เล่มมีได้หลายสำเนา (Copy) และผู้ใช้ 1 คนสามารถยืมหนังสือได้หลายสำเนา



รูปที่ 6.19 แสดงคลาสไดอะแกรมที่ใช้ประกอบการสร้างชุดคำสั่งของเมทธอดคืนหนังสือ

จากการประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรมกับซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมดังกล่าว (รายละเอียดการประยุกต์ใช้กฎดังแสดงรายละเอียดในภาคผนวก ค) และทำการตัดการประกาศตัวแปรที่ประกาศซ้ำซ้อน จะได้ชุดคำสั่งภาษาจาวาของเมทธอดคืนหนังสือดังรูปที่ 6.20

```
public class Library{
        private Vector books;
        private Vector users;
        public void checkIn(int uId, int copyId){
                User user;
                Copy copy;
```

```
                    Book book;
                    User resUser;
                    user = findUser(uId);
                    copy = user.checkIn(copyId);
                    book = copy.getBook()
                    resUser =book.checkIn()
                    if(resUser!=NULL){
                        resUser.checkInO(copy);
                    }
                    if(resUser!=NULL){
                        resUser.notify(copy);
                    }
                }
            }
```

รูปที่ 6.20 แสดงชุดคำสั่งภาษาจาวาของเมทธอดคืนหนังสือที่สร้างจากเครื่องมือที่ประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรม

ตัวอย่างชุดคำสั่งภาษาจาวาจริงของเมทธอดคืนหนังสือของระบบห้องสมุดแสดงดังรูปที่ 6.21 ถูกนำมาเขียนให้เป็นบรรทัดย่อยๆ ดังรูปที่ 6.22 เพื่อความง่ายในการเปรียบเทียบจำนวนบรรทัดคำสั่งที่สร้างได้จากการประยุกต์ใช้กฎต่อจำนวนบรรทัดคำสั่งทั้งหมด

```
import java.util.*;
public class Library {
    private Vector books;
    private Vector users;
    public void checkIn(int uId, int copyId) throws UserNotFoundException,
InvalidReturnCopyException {
        User user = findUser(uId);
        if (user == null)   {
            throw new UserNotFoundException();
        }
        Copy copy = user.checkIn(copyId);
        if (copy == null)  {
            throw new InvalidReturnCopyException();
```

```
                    }
                    Book book = copy.getBook();
                    User resUser = book.checkIn();
                    if (resUser != null)        {
                            resUser.checkInO(copy);
                            resUser.notify(copy);
                    }
            }
    }
```

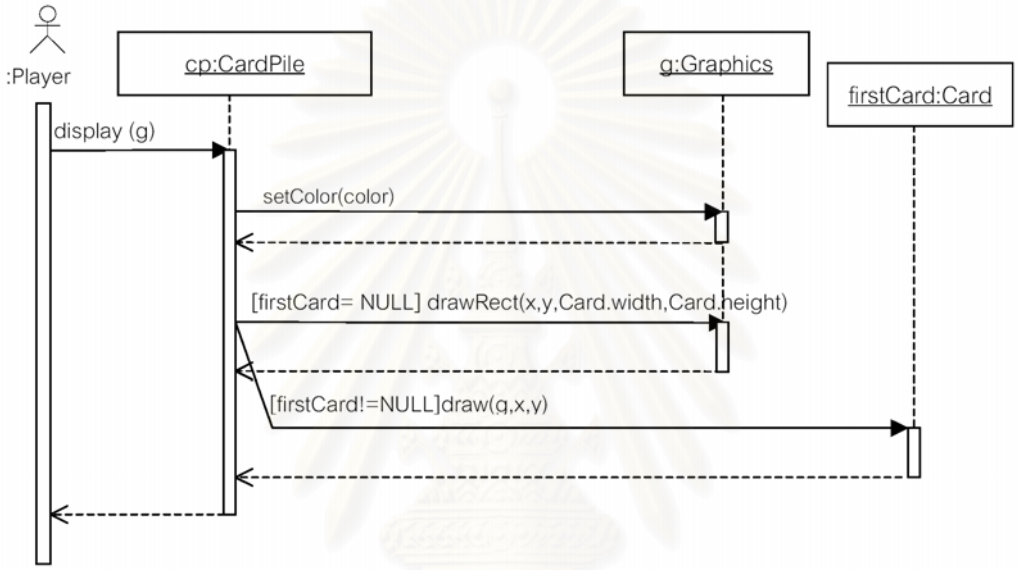
รูปที่ 6.21 แสดงตัวอย่างชุดคำสั่งภาษาจาวาของเมทธอดคืนหนังสือของระบบห้องสมุด [12]

```
    import java.util.*;
    public class Library {
            private Vector books;
            private Vector users;
            public void checkIn(int uId, int copyId) throws UserNotFoundException,
            InvalidReturnCopyException {
                    User user;
                    Copy copy;
                    Book book;
                    User resUser;
                    user = findUser(uId);
                    if (user == null)   {
                            throw new UserNotFoundException();
                    }
                    copy = user.checkIn(copyId);
                    if (copy == null){
                            throw new InvalidReturnCopyException();
                    }
                    book = copy.getBook();
                    resUser = book.checkIn();
                    if (resUser != null){
                            resUser.checkInO(copy);
                    }
                    if (resUser != null){
```

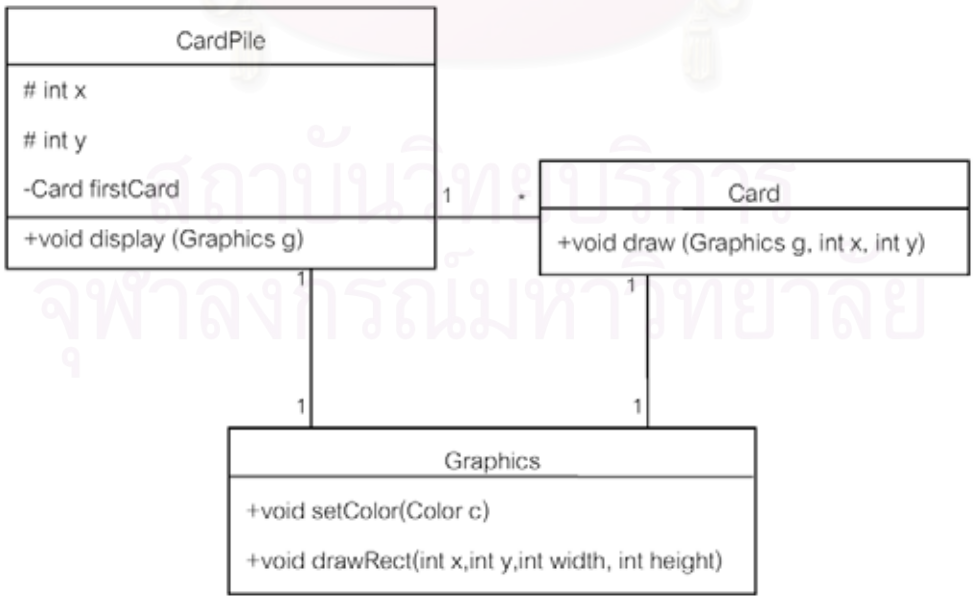
```
                    resUser.notify(copy);
                }
            }
        }
```

รูปที่ 6.22 แสดงตัวอย่างชุดคำสั่งภาษาจาวาของเมทธอดคืนหนังสือที่ถูกนำมาเขียนใหม่

6.3 การแปลงซีเควนซ์ไดอะแกรมของเมทธอดแสดงของระบบกองไพ่



รูปที่ 6.23 แสดงซีเควนซ์ไดอะแกรมของเมทธอดแสดงของระบบกองไพ่



รูปที่ 6.24 แสดงคลาสไดอะแกรมที่ใช้ประกอบการสร้างชุดคำสั่งของเมทธอดแสดง

รูปที่ 6.23 แสดงซีเควนซ์ไดอะแกรมของเมทธอดแสดง โดยเริ่มต้นจากมีการเรียกเมทธอดแสดงโดยส่งวัตถุการวาด (g) มาให้ ระบบกองไพ่จะทำการกำหนดสี (setColor) หลังจากนั้นจึงทำการตรวจสอบว่ามีไพ่ใบแรก (firstCard) หรือไม่ ถ้าไม่มี (firstCard = NULL) จะทำการวาดสี่เหลี่ยมตามสีที่กำหนดไว้ ถ้ามีไพ่ใบแรกจะทำการเรียกใช้เมทธอดวาด (draw) ของวัตถุไพ่ใบแรก

รูปที่ 6.24 แสดงคลาสไดอะแกรมที่ใช้ประกอบการสร้างชุดคำสั่งของเมทธอดแสดง ซึ่งกองไพ่ (CardPile) 1 กองมีไพ่ (Card) ได้หลายใบ ทั้งกองไพ่ และไพ่สามารถเกี่ยวข้องกับการวาด (Graphics) ได้หลายการวาด

หลังจากการประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรมกับซีเควนซ์ไดอะแกรมและคลาสไดอะแกรมดังกล่าว (รายละเอียดการประยุกต์ใช้กฎดังแสดงรายละเอียดในภาคผนวก ง) และทำการตัดการประกาศตัวแปรที่ประกาศซ้ำซ้อน จะได้ชุดคำสั่งภาษาจาวาของเมทธอดคืนหนังสือดังรูปที่ 6.25

```
public class CardPile{
        protected int x;
        protected int y;
        private Card firstCard;
        public void display(Graphics g){
                Color color;
                g.setColor(color);
                if (firstCard==NULL){
                g.drawRect(x,y,Card.width,Card.height);
                }else if(firstCard!=NULL){
                        firstCard.draw(g,x,y);
                }
        }
}
```

รูปที่ 6.25 แสดงชุดคำสั่งภาษาจาวาของเมทธอดแสดงที่สร้างจากเครื่องมือที่ประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรม

ตัวอย่างชุดคำสั่งภาษาจาวาจริงของเมทธอดแสดงของระบบกองไพ่แสดงดังรูปที่ 6.26 ถูกนำมาเขียนให้เป็นบรรทัดย่อยๆ ดังรูปที่ 6.27 เพื่อความง่ายในการเปรียบเทียบจำนวนบรรทัดคำสั่งที่สร้างได้จากการประยุกต์ใช้กฎต่อจำนวนบรรทัดคำสั่งทั้งหมด

```
public class CardPile {
```

```
        protected int x;
        protected int y;
        private Card firstCard;
        public void display (Graphics g){
                Color color = new Color(black);
                g.setColor(color);
                if(firstCard==NULL){
                        g.drawRect(x,y,Card.width,Card.height);
                }else{
                        firstCard.draw(q,x,y);
                }
        }
}
```

รูปที่ 6.26 แสดงตัวอย่างชุดคำสั่งภาษาจาวาของเมทธอดแสดงของระบบกองไพ่ [13]

```
public class CardPile {
        protected int x;
        protected int y;
        private Card firstCard;
        public void display (Graphics g){
                Color color;
                color = new Color(black);
                g.setColor(color);
                if(firstCard==NULL){
                        g.drawRect(x,y,Card.width,Card.height);
                }else{
                        firstCard.draw(q,x,y);
                }
        }
}
```

รูปที่ 6.27 แสดงตัวอย่างชุดคำสั่งภาษาจาวาของเมทธอดแสดงที่ถูกนำมาเขียนใหม่

6.4 ผลการประเมิน

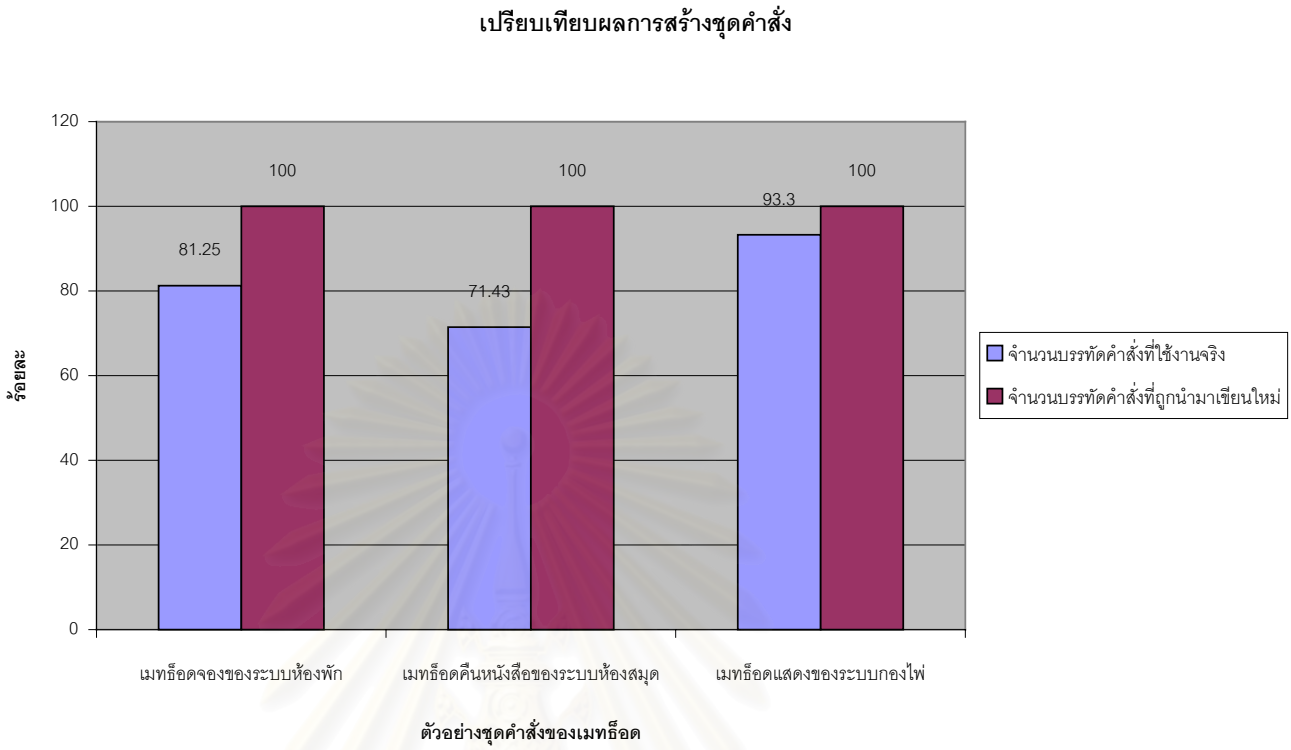
จากการเปรียบเทียบชุดคำสั่งภาษาจาวาที่สร้างจากเครื่องมือที่ประยุกต์ใช้กฎการแปลงซีเควนซ์ไดอะแกรม และชุดคำสั่งภาษาจาวาที่ถูกนำมาเขียนใหม่ จะได้สรุปผลการสร้างชุด

คำสั่งดังตารางที่ 6.1 ซึ่งแสดงถึงจำนวนบรรทัดคำสั่งที่สร้างได้ จำนวนบรรทัดคำสั่งที่นำมาใช้งานจริง และจำนวนบรรทัดคำสั่งที่ต้องเพิ่มเติม โดยการนับจำนวนบรรทัดจะไม่นับรวมบรรทัดที่ว่าง ในกรณีที่ไม่สามารถสร้างได้ทั้งบรรทัดจะนับเป็น 0.5 บรรทัด จากนั้นเมื่อนำจำนวนบรรทัดคำสั่งที่ใช้งานจริงหารจำนวนบรรทัดคำสั่งที่ถูกนำมาเขียนใหม่จากชุดคำสั่งจริงคูณด้วย 100 จะได้ร้อยละของจำนวนบรรทัดคำสั่งที่สร้างได้ และนำมาใช้งานจริงจากการประยุกต์ใช้กฎ

ตารางที่ 6.1 แสดงสรุปผลการสร้างชุดคำสั่ง

ตัวอย่างชุดคำสั่งของเมทธอด	จำนวนบรรทัดคำสั่งที่ถูกนำมาเขียนใหม่จากชุดคำสั่งจริง	จำนวนบรรทัดคำสั่งที่สร้างได้	จำนวนบรรทัดคำสั่งที่ใช้งานจริง	จำนวนบรรทัดคำสั่งที่ต้องเพิ่มเติม	ร้อยละของจำนวนบรรทัดคำสั่งที่สร้างได้และนำมาใช้งานจริงจากการประยุกต์ใช้กฎ
เมทธอดจองของระบบห้องพัก	24	20	19.5	4.5	81.25
เมทธอดคืนหนังสือของระบบห้องสมุด	28	20	20	8	71.43
เมทธอดแสดงของระบบกองไพ่	15	14	14	1	93.3

จากรูปที่ 6.28 จะพบว่าเมื่อเทียบร้อยละจำนวนบรรทัดคำสั่งที่สร้างได้ และนำมาใช้งานจริงจากการประยุกต์ใช้กฎกับร้อยละของจำนวนบรรทัดคำสั่งที่ถูกนำมาเขียนใหม่จากชุดคำสั่งจริง จำนวนบรรทัดคำสั่งที่สร้างได้ และนำมาใช้งานจริงจากการประยุกต์ใช้กฎมีสัดส่วนใกล้เคียงกับจำนวนบรรทัดคำสั่งที่ถูกนำมาเขียนใหม่จากชุดคำสั่งจริงมาก แต่เนื่องจากกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาไม่สามารถระบุส่วนการเขียนโปรแกรมในเชิงตรรกะ (Programming Logic) ได้ หรือในกรณีที่ซีเควนซ์ไดอะแกรมที่ออกแบบเขียนไว้ไม่ครอบคลุมการทำงานทั้งหมด อาทิ เช่น การจัดการความผิดพลาด (Exception Handling) เป็นต้น การแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาจะได้จำนวนบรรทัดของชุดคำสั่งน้อยกว่าจำนวนบรรทัดของชุดคำสั่งที่ใช้จริง



รูปที่ 6.28 แผนภูมิแท่งแสดงการเปรียบเทียบร้อยละของจำนวนบรรทัดคำสั่งที่สร้างได้ และนำมาใช้งานจริงจากการประยุกต์ใช้กฎกับจำนวนบรรทัดคำสั่งที่ถูกนำมาเขียนใหม่จากชุดคำสั่งจริง

บทที่ 7
สรุปผลการวิจัย และข้อเสนอแนะ

7.1 สรุปผลการวิจัย

วิทยานิพนธ์นี้ทำการออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา เพื่อให้สามารถนำกฎไปประยุกต์ใช้ในการสร้างเครื่องมืออัตโนมัติสำหรับแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา โดยทำการออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาจำนวน 8 กฎ ได้แก่

1) เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย

2) เมต้ารูลสำหรับการแบ่งซีเควนซ์

3) เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง

4) เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร

5) เมต้ารูลสำหรับการกำหนดค่าให้ตัวชี้

6) เมต้ารูลสำหรับการสร้างวัตถุใหม่

7) เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว

8) เมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเอง

หลังจากนั้นได้ทำการพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา เพื่อแสดงว่ากฎที่ได้ออกแบบไว้สามารถนำมาประยุกต์ใช้ได้จริง แล้วจึงทดสอบการแปลงซีเควนซ์ไดอะแกรมโดยใช้ซีเควนซ์ไดอะแกรมของเมทธอดของระบบ 2 ระบบ และคลาสไดอะแกรมที่เขียนขึ้นอย่างถูกต้องตามวากยสัมพันธ์ ความหมายในทางสถิต มีโครงสร้างที่ดีตรงตามมาตรฐานยูเอ็มแอล และครอบคลุมถึงกฎการแปลงทุกกฎที่ออกแบบเป็นข้อมูลนำเข้า ได้แก่ ซีเควนซ์ไดอะแกรมของเมทธอดจองของระบบห้องพัก และซีเควนซ์ไดอะแกรมของเมทธอดคืนหนังสือของระบบห้องสมุด และซีเควนซ์ไดอะแกรมของเมทธอดแสดงของระบบกองไพ่ ผลการประเมินพบว่า เครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมสามารถสร้างชุดคำสั่งของเมทธอดจองของระบบห้องพักได้ร้อยละ 81.25 เมทธอดคืนหนังสือของระบบห้องสมุดได้ร้อยละ 71.43 และเมทธอดแสดงของระบบกองไพ่ได้ร้อยละ 93.3 สำหรับประโยชน์ของการประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา คือ ช่วยให้การพัฒนาโปรแกรมประยุกต์โดยใช้ภาษาจาวาเป็นภาษาในการเขียนโปรแกรมทำได้เร็วขึ้น โดยเมื่อผู้พัฒนาโปรแกรมประยุกต์ออกแบบยูเอ็มแอลซีเควนซ์ไดอะแกรมเพื่อแสดงพฤติ

กรรมของเมทธอดนั้นๆ เรียบร้อยแล้ว สามารถทำการแปลงจากยูเอ็มแอลซีเควนซ์ไดอะแกรมไปเป็นชุดคำสั่งภาษาจาวาได้ทันที โดยที่ผู้พัฒนาโปรแกรมไม่ต้องเสียเวลาในการสร้างชุดคำสั่งด้วยตนเองทั้งหมด เพียงเพิ่มเติม และแก้ไขบางส่วนของชุดคำสั่ง อาทิเช่น การเพิ่มการเขียนโปรแกรมในเชิงตรรกะ เป็นต้น

7.2 ข้อเสนอแนะ

1) เนื่องจากกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมนี้เน้นการแปลงซีเควนซ์ไดอะแกรมที่มีลักษณะทั่วไป (General sequence diagrams) โดยให้ความสำคัญกับลูกศรพื้นฐาน (Basic type of arrow flows) เฉพาะลูกศรที่ใช้ในการส่งสารแบบธรรมดาเท่านั้น แต่ไม่ได้ทำการออกแบบกฎที่สนับสนุนข้อความที่ไม่ได้เกิดพร้อมกัน การเกิดภาวะพร้อมกัน และเงื่อนไขที่มีการแข่งขัน ดังนั้นจึงควรเพิ่มเติมการออแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมที่สนับสนุนการเขียนซีเควนซ์ไดอะแกรมที่แสดงข้อความที่ไม่ได้เกิดพร้อมกัน การเกิดภาวะพร้อมกัน และเงื่อนไขที่มีการแข่งขัน

2) เนื่องจากกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมนี้ได้นำคลาสไดอะแกรมเข้ามาใช้ร่วมในการแปลง แต่ได้ใช้เพียงข้อมูลเฉพาะแต่ละคลาสเท่านั้นไม่ได้นำความสัมพันธ์ระหว่างคลาสมาพิจารณา ดังนั้นจึงควรเพิ่มเติมการนำข้อมูลความสัมพันธ์ระหว่างคลาสมาใช้ในการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเพื่อให้สามารถแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาได้ถูกต้อง และตรงความหมายมากขึ้น

3) เนื่องจากกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเน้นการแปลงซีเควนซ์ไดอะแกรมของเมทธอดหนึ่งเมทธอดเท่านั้น ดังนั้นจึงควรเพิ่มเติมการแปลงซีเควนซ์ไดอะแกรมของทุกเมทธอดในคลาสหนึ่งคลาส โดยการรวมชุดคำสั่งของเมทธอดแต่ละเมทธอดเข้าด้วยกัน เพื่อให้ได้ชุดคำสั่งภาษาจาวาที่สมบูรณ์ยิ่งขึ้น

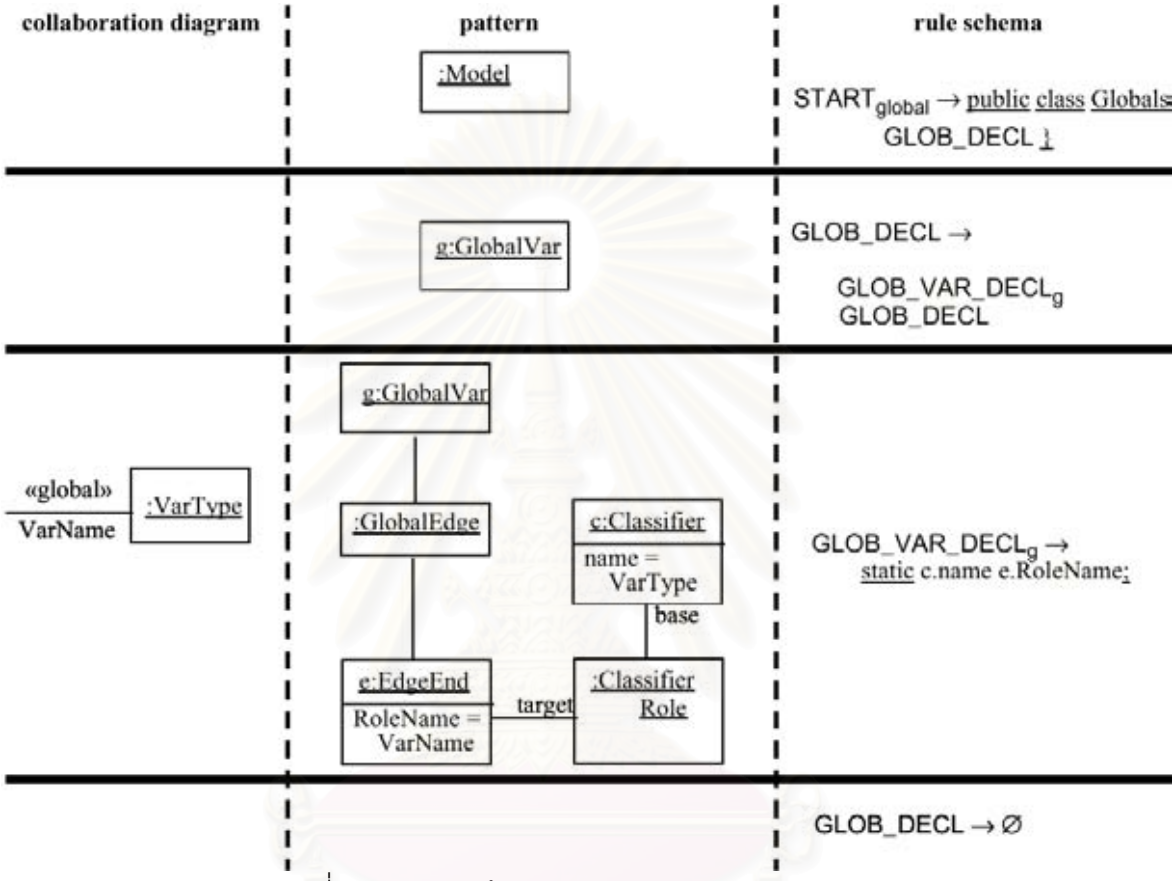
รายการอ้างอิง

1. วีระศักดิ์ ซึงถาวร. Fundamental Of Java Programming volumn1. Bangkok : Sum Publishing, 1998.
2. Martin Fowler and Kendall Scott. UML Distilled: Applying the Standard Object Modeling Language. Addison-Wesley, 1998.
3. Gregor Engels, Roland Hucking, Stefan Sauer and Annika Wagner. “UML Collaboration Diagrams and Their Transformation to Java”, Second International Conference on the Unified Modeling Language (UML’ 99), pp. 437-488, 1999.
4. Gregor Engels, Reiko Heckel and Jochen Malte Kuster. “Rule-Based Specification of Behavioral Consistency Based on the UML Meta-model”, Fourth International Conference on the Unified Modeling Language (UML2001), pp. 272-286, 2001.
5. Ken Slonneger, “Two-level Grammars”, http://www.cs.uiowa.edu/~slonnegr/plf/Book.
6. Rational Rose, “Sequence diagrams”, [online] available from http://www.rational.com/uml/resources/quick/uml_fig8.jsp. [2001, November 14]
7. OMG. UML Semantics. Version 1.1. The Object Management Group, Document ad/97-08-04, Framingham MA, 1997.
8. Neeraj Sangal, Edward Ferrel, Karl Lieberherr and David Lorenz. “Interaction Schemata: Compiling Interac,tions to Code”, the proceedings of Technology of Object-Oriented Language and Systems (TOOLS USA’ 99), pp. 268-277, 1999.
9. Dong Hyuk Park, Soo Dong Kim. “XML Rule Based Source Code Generator for UML Case Tool”, Asia-Pacific Software Engineering Conference (APSEC2001), pp. 53-60, 2001.
10. OMG. UML Notation Guide. Version 1.1. The Object Management Group, Document ad/97-08-05, Framingham MA, 1997.
11. นายชาติชาย ดวงสอาด. SingleRoom.java., 2002.
12. นายพชร จำเรียงฤทธิ์. Library.java., 2002.
13. นายวรินทร์ วัฒนพรพรหม. CardPile.java., 2002.

ภาคผนวก

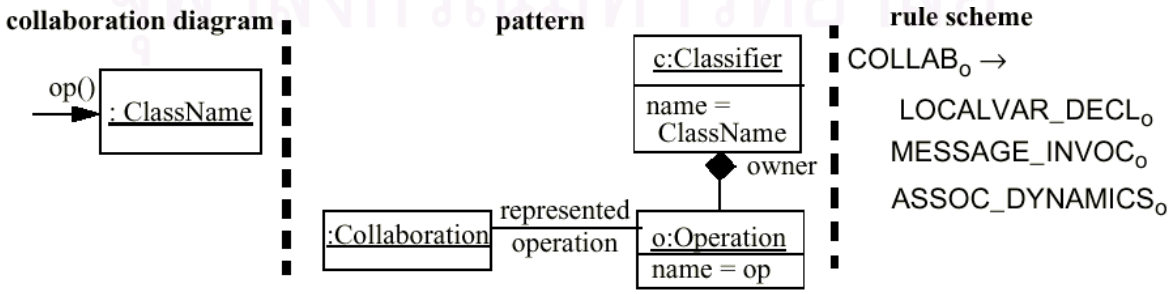
ภาคผนวก ก.
กฎการแปลงยูเอ็มแอลคอแลบบอเรชันไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

1) เมต้ารูลสำหรับตัวแปรครอบคลุม (Meta rules for global variables)



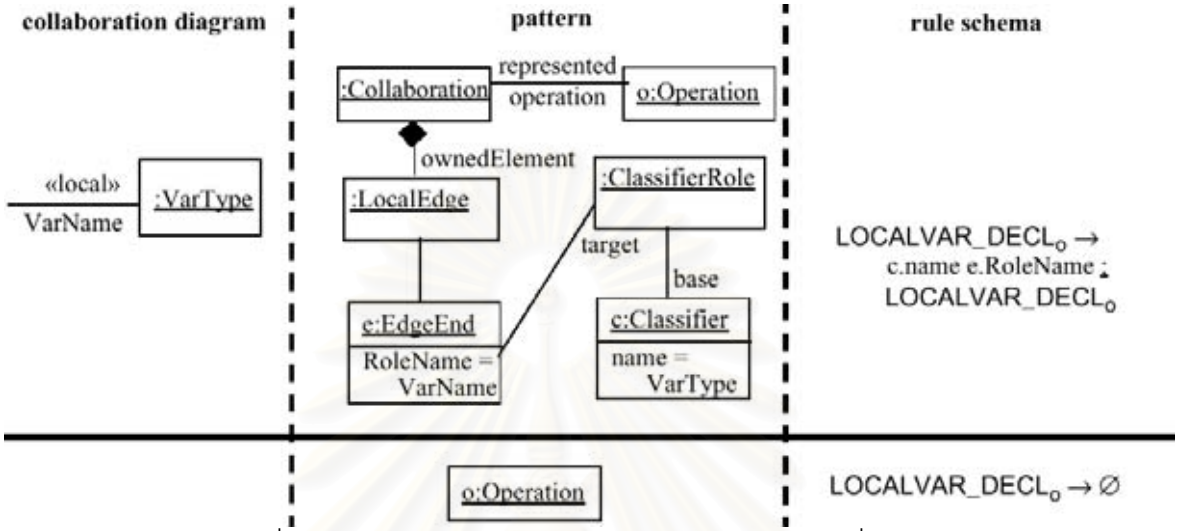
รูปที่ ก-1 แสดงเมต้ารูลสำหรับตัวแปรครอบคลุม

2) เมต้ารูลสำหรับการแบ่งคอแลบบอเรชันไดอะแกรม (Meta rules for splitting of COLLAB)



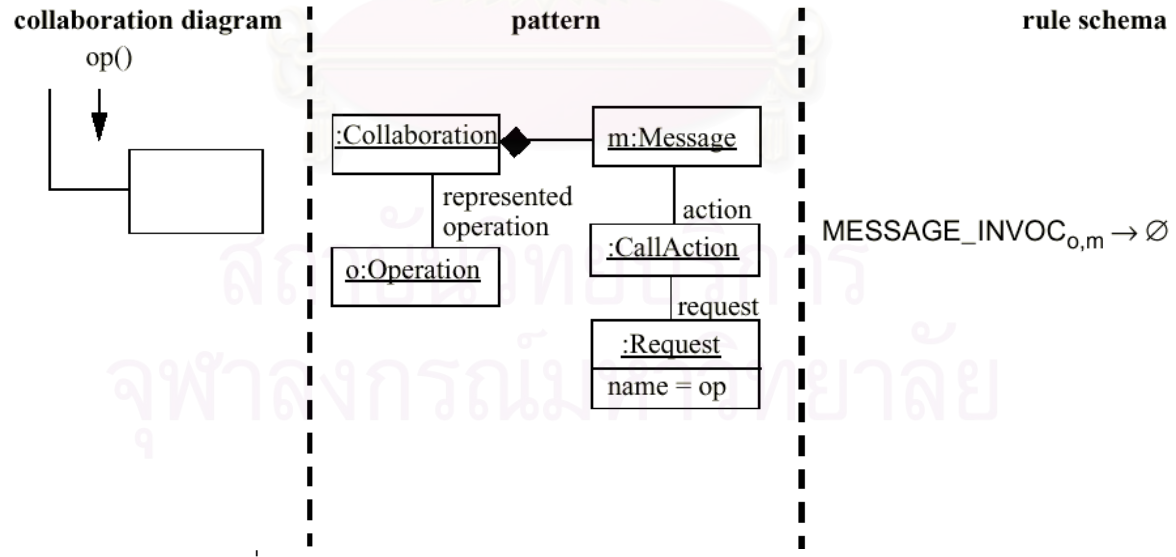
รูปที่ ก-2 แสดงเมต้ารูลสำหรับการแบ่งคอแลบบอเรชันไดอะแกรม

3) เมต้ารูลสำหรับการประกาศตัวแปรท้องถิ่น (Meta rules for local variable declaration)



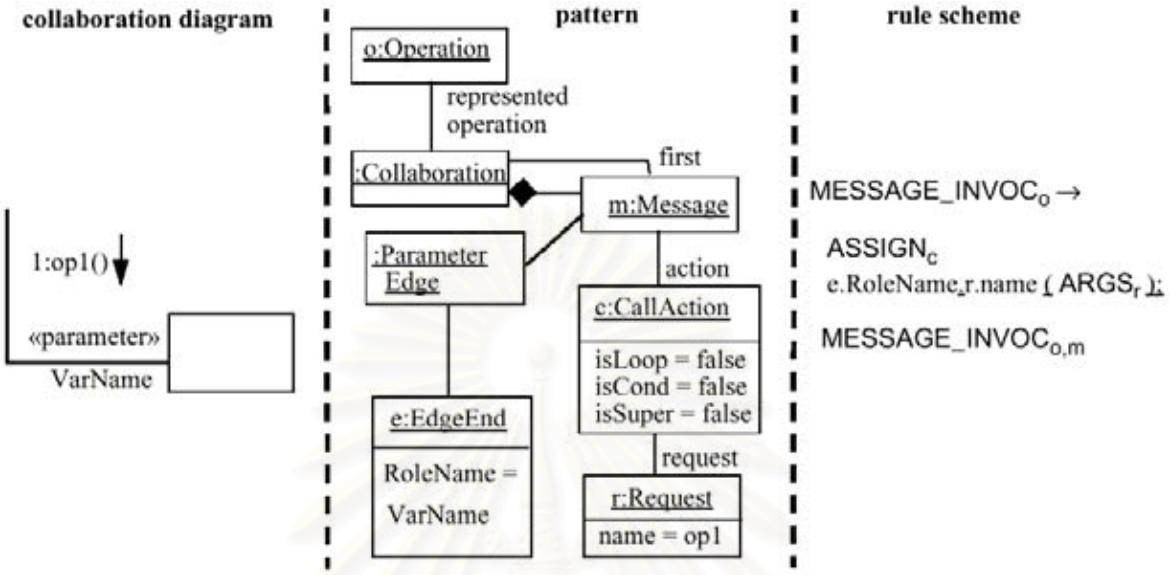
รูปที่ ก-3 แสดงเมต้ารูลสำหรับการประกาศตัวแปรท้องถิ่น

4) เมต้ารูลสำหรับการเรียกเมทธอดอย่างสมบูรณ์ (Meta rules for completing method invocation)



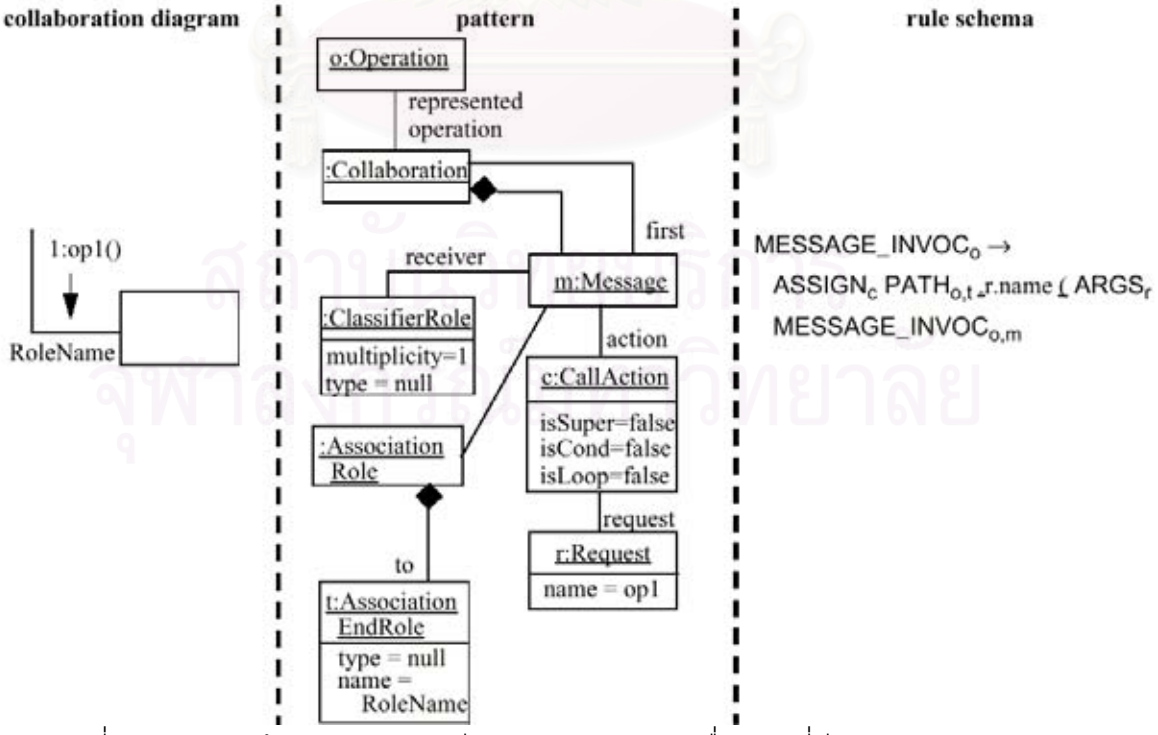
รูปที่ ก-4 แสดงเมต้ารูลสำหรับการเรียกเมทธอดอย่างสมบูรณ์

5) เมต้ารูลสำหรับการเรียกเมทธอดบนวัตถุพารามิเตอร์ (Meta rules for method invocation on parameter object)



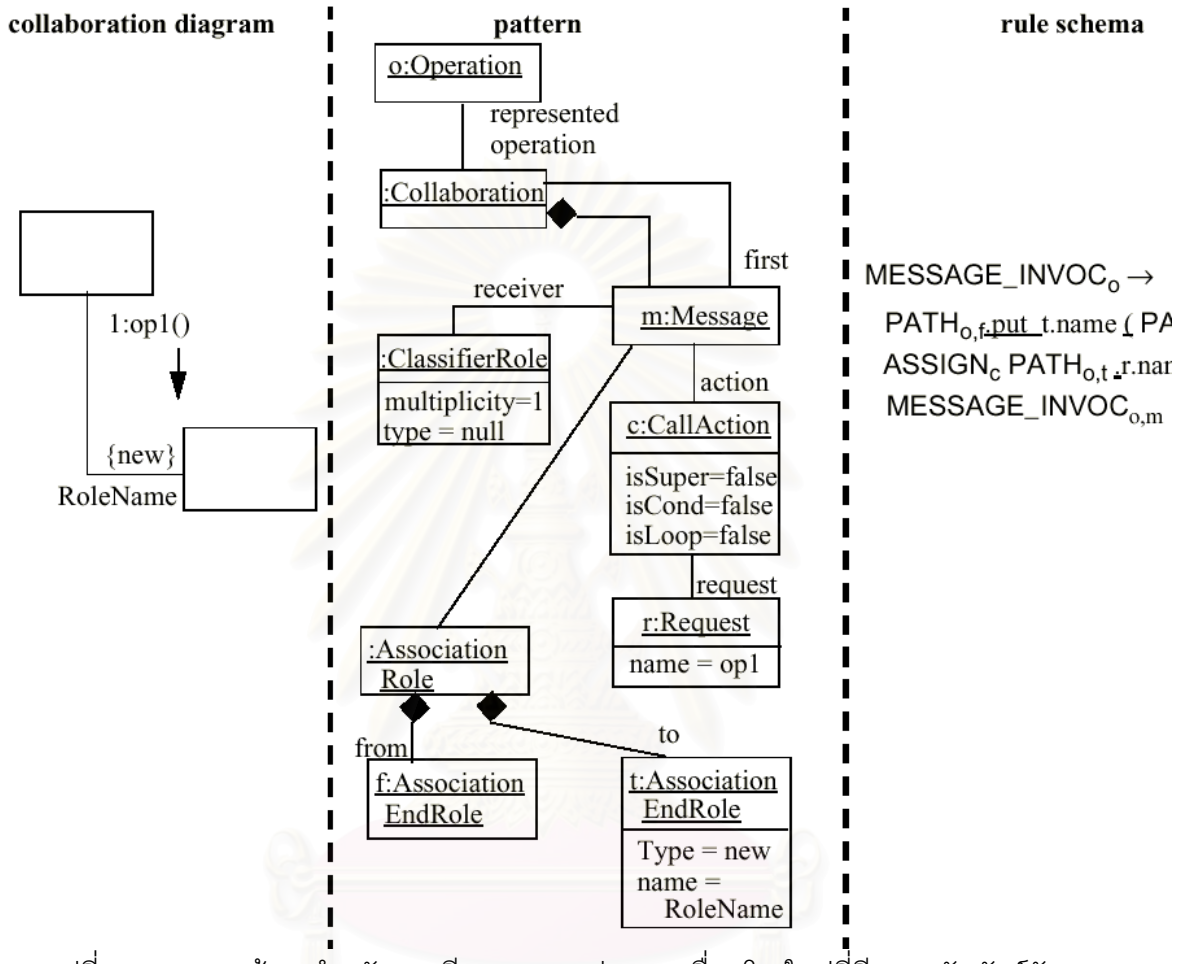
รูปที่ ก-5 แสดงเมต้ารูลสำหรับการเรียกเมทธอดบนวัตถุพารามิเตอร์

6) เมต้ารูลสำหรับการเรียกเมทธอดผ่านการเชื่อมโยงที่มีความสัมพันธ์กัน (Meta rules for method invocation via association link)



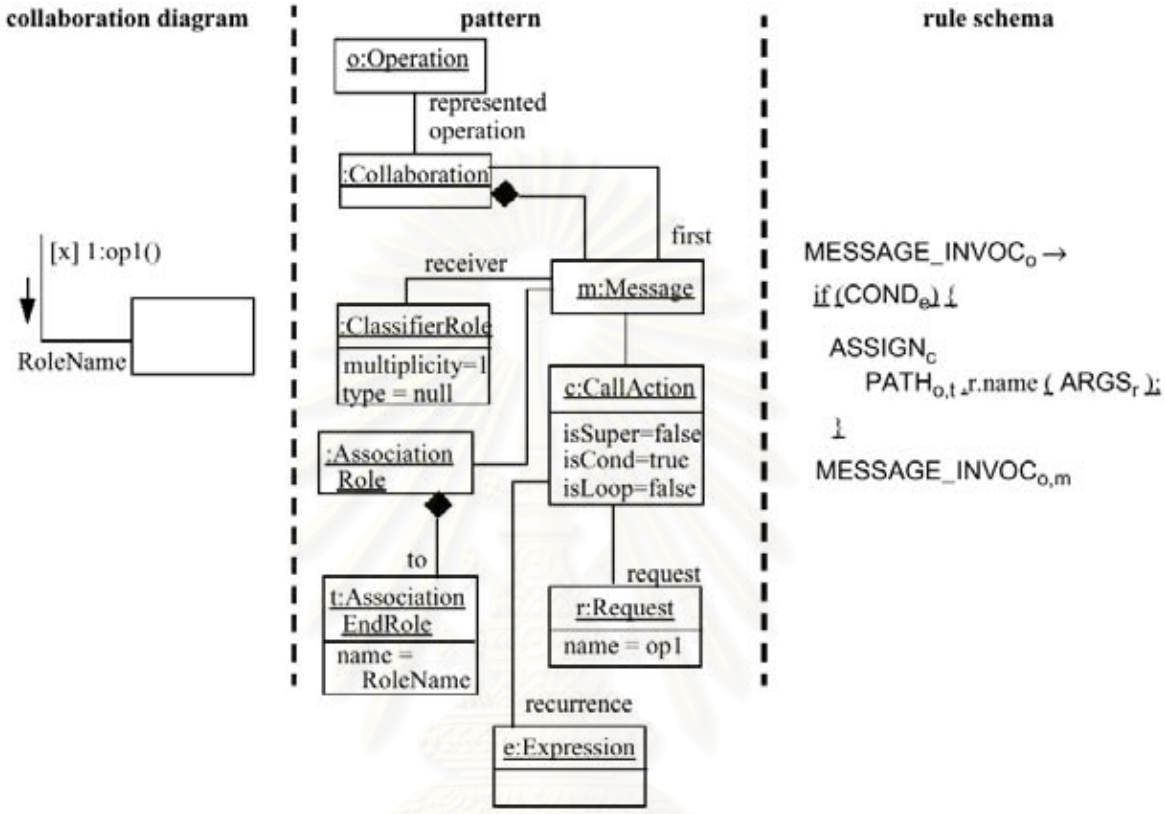
รูปที่ ก-6 แสดงเมต้ารูลสำหรับการเรียกเมทธอดผ่านการเชื่อมโยงที่มีความสัมพันธ์กัน

7) เมต้ารูลสำหรับการเรียกเมทธอดผ่านการเชื่อมโยงใหม่ที่มีความสัมพันธ์กัน (Meta rules for method invocation via a {new} association link)



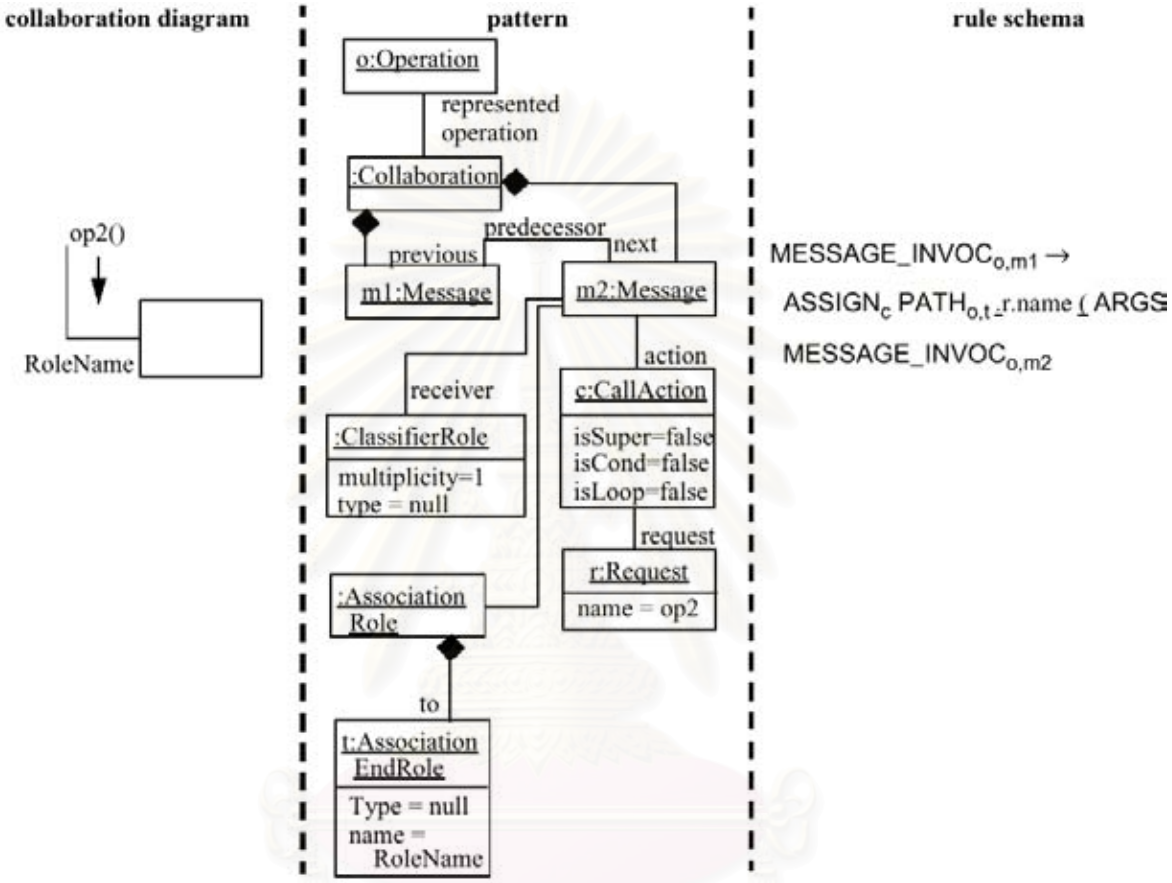
รูปที่ ก-7 แสดงเมต้ารูลสำหรับการเรียกเมทธอดผ่านการเชื่อมโยงใหม่ที่มีความสัมพันธ์กัน

8) เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไขผ่านการเชื่อมโยงที่มีความสัมพันธ์กัน (Meta rules for conditional method invocation via association link)



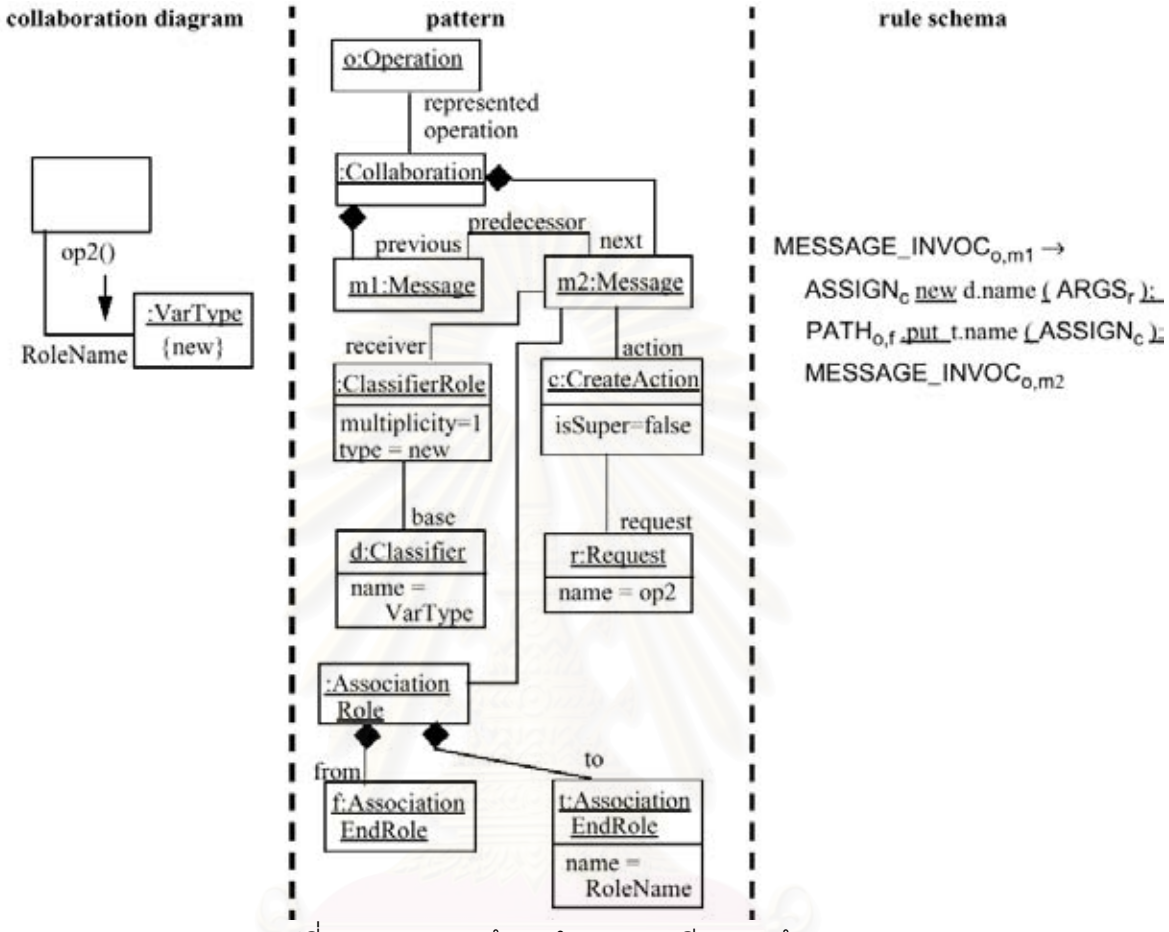
รูปที่ ก-8 แสดงเมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไขผ่านการเชื่อมโยงที่มีความสัมพันธ์กัน

9) เมต้ารูลสำหรับการผ่านลำดับของสารโดยอยู่บนพื้นฐานของการเรียกเมทธอดบนการเชื่อมโยงที่มีความสัมพันธ์กันที่มีอยู่แล้ว (Meta rules for traversing message sequences based on method invocation on existing association link)



รูปที่ ก-9 แสดงเมต้ารูลสำหรับการผ่านลำดับของสารโดยอยู่บนพื้นฐานของการเรียกเมทธอดบนการเชื่อมโยงที่มีความสัมพันธ์กันที่มีอยู่แล้ว

10) เมต้ารูลสำหรับการเรียกตัวสร้าง (Meta rules for constructor invocation)



รูปที่ ก-10 แสดงเมต้ารูลสำหรับการเรียกตัวสร้าง

ภาคผนวก ข.
การพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม

ในการพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมนี้ ผู้วิจัยได้เลือกใช้ภาษา C++ เป็นภาษาในการเขียนโปรแกรม โดยเลือกใช้เครื่องมือ Borland C++ Builder 5 ของค่ายบอร์แลนด์เป็นเครื่องมือในการพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม เนื่องจาก Borland C++ Builder 5 นี้มีวิธีการเขียนโปรแกรมแบบวิชวล (Visual) กล่าวคือการเขียนโปรแกรมด้วยส่วนประกอบต่างๆ ที่เรามองเห็นทันทีในขณะกำลังเขียนโปรแกรม ซึ่งช่วยให้ผู้ใช้สามารถเขียนโปรแกรมได้สะดวก และรวดเร็วกว่า การพัฒนาเริ่มต้นจากการออกแบบส่วนต่อประสานของโปรแกรม แล้วจึงทำการเขียนข้อความสั่ง (Statement) เพิ่มเติมเข้ากับชุดคำสั่งที่มีอยู่ เพื่อให้โปรแกรมที่พัฒนาขึ้นสามารถทำงานได้ตามต้องการ หลังจากนั้นจึงทำการทดสอบโปรแกรมเพื่อหาข้อผิดพลาด และแก้ไขต่อไป

ข-1 การออกแบบส่วนต่อประสาน

การออกแบบส่วนต่อประสานจะเริ่มต้นจากการพิจารณาข้อมูลนำเข้า ซึ่งในที่นี้คือ ซีเควนซ์ และคลาสไดอะแกรมที่เกี่ยวข้อง ผู้วิจัยออกแบบให้ส่วนต่อประสานมีปุ่ม (Button) ที่มีรูปเพื่อให้ผู้นำเข้าข้อมูลเลือกว่าส่วนของซีเควนซ์ไดอะแกรมที่ผู้นำเข้าข้อมูลกำลังพิจารณาอยู่นั้น ตรงกับรูปบนปุ่มใด โดยการนำเข้าข้อมูลผู้นำเข้าข้อมูลจะอ่านซีเควนซ์ไดอะแกรมตามลำดับจากบนลงล่าง และซ้ายมาขวา หลังจากผู้นำเข้าข้อมูลเลือกปุ่มแล้ว เครื่องมือจะนำไปสู่หน้าของการนำเข้ารายละเอียดของซีเควนซ์ และคลาสไดอะแกรมที่เกี่ยวข้อง เพื่อให้ผู้นำเข้าข้อมูลกรอกข้อมูลลงในกล่องข้อความต่อไป ผู้วิจัยออกแบบให้มีแถบแสดงคำแนะนำการใช้อยู่ด้านบนถัดลงมาจากรายการหลัก เพื่อให้คำแนะนำแก่ผู้นำเข้าข้อมูลตลอดการใช้งาน การแสดงผลลัพธ์ชุดคำสั่งที่ได้จากการกดปุ่มสร้างในแต่ละครั้งจะแสดงไว้ทางด้านขวาของส่วนนำเข้าข้อมูลผ่านข้อเขียนเตือนความจำ (Memo) โดยที่ออกแบบให้ผู้นำเข้าข้อมูลไม่สามารถแก้ไขข้อมูลในข้อเขียนเตือนความจำได้ นอกจากนี้ยังมีการออกแบบให้มีรายการหลักเพื่อรวบรวมคำสั่งหลักที่ต้องผู้นำเข้าข้อมูลต้องใช้ไว้ด้านบนสุดซ้ายมือของเครื่องมือ

ข-2 การเพิ่มเติมข้อความสั่ง

หลังจากการออกแบบส่วนต่อประสาน จึงทำการเขียนข้อความสั่งเพิ่มเติม ข้อความสั่งที่เขียนเพิ่มเติม เป็นส่วนของการดึงข้อมูลจากกล่องข้อความมาสร้างเป็นผลลัพธ์ชุดคำสั่ง เนื่องจากการสร้างชุดคำสังแต่ละครั้งจะทำโดยการแทนที่สัญลักษณ์ที่ไม่ถึงจุดจบด้วยสัญลักษณ์ที่

ถึงจุดจบ และข้อมูลจากกล่องข้อความ ชุดคำสั่งทั้งหมดของเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมแสดงดังรูปที่ ข-1 ถึง ข-4

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "MainForm.h"
#include "About.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
int countIf;
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
        Memo1->Modified = false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
        Notebook1->ActivePage = "1";
        /*Disable other SpeedButtons coz show only Start SpeedButton*/
        SpeedButton2->Enabled = false;
        SpeedButton3->Enabled = false;
        SpeedButton4->Enabled = false;
        SpeedButton5->Enabled = false;
        SpeedButton6->Enabled = false;
}

//---------------------------------------------------------------------------

void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
        Notebook1->ActivePage = "1";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
```

```
        Notebook1->ActivePage = "2";
        Generate2->Enabled = true;
        Panel1->Caption = "Please fill condition or branching conditions.";
}
//---------------------------------------------------------------------------


void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
{
        Notebook1->ActivePage = "3";
        Generate3->Enabled = true;
        Panel1->Caption = "Please fill details of called method and class"
        " owner of method.";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SpeedButton4Click(TObject *Sender)
{
        Notebook1->ActivePage = "4";
        Generate4->Enabled = true;
        Panel1->Caption = "Please fill details of called method and class"
        " owner of method.";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SpeedButton5Click(TObject *Sender)
{
        Notebook1->ActivePage = "5";
        Generate5->Enabled = true;
        Panel1->Caption = "Please fill details of called method and class"
        " owner of method.";
}
//---------------------------------------------------------------------------


void __fastcall TForm1::SpeedButton6Click(TObject *Sender)
{
        Notebook1->ActivePage = "6";
        Panel1->Caption = "Please click Generate button to end sequence"
        " diagram";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Generate1Click(TObject *Sender)
```

```
{
    /*Apply rule1:declare class*/
    Memo1->Lines->Add("STARTc");
ShowMessage("step");
    int SelPos = Memo1->Lines->Text.Pos("STARTc");
    if (SelPos > 0){
        Memo1->SelStart = SelPos - 1;
        Memo1->SelLength = 6;
        Memo1->SelText = "public class " + ClassN11->Text + "{";
        Memo1->Lines->Add("    ATTRIBUTESc");
        Memo1->Lines->Add("    METHODSc");
        Memo1->Lines->Add("}");
        SelPos = 0;
ShowMessage("step");
    }
    /*Apply rule1:declare attributes*/
    /*Case No attribute1*/
    if (AttributeN11->Text == ""){
        SelPos = Memo1->Lines->Text.Pos("ATTRIBUTESc");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 11;
            Memo1->SelText = "";
ShowMessage("step");
            SelPos = 0;
        }
    /*Case has attribute1*/
    }else{
        SelPos = Memo1->Lines->Text.Pos("ATTRIBUTESc");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 11;
            Memo1->SelText = AttributeV11->Text + " " +
            AttributeT11->Text + " " + AttributeN11->Text + ";";
            SelPos = 0;
ShowMessage("step");
            /*Find line to insert ATTRIBUTESc*/
            AnsiString LineStr;
            int i;
            for(i=0;i<Memo1->Lines->Count;i++){
                LineStr = Memo1->Lines->Strings[i];
                SelPos = LineStr.Pos(AttributeN11->Text);
                if(SelPos>0) break;
            }
            if (SelPos > 0){
```

```
                    Memo1->Lines->Insert(i+1,
                    "    ATTRIBUTESc");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }
        /*Case No attribute2*/
        if (AttributeN12->Text == ""){
            SelPos = Memo1->Lines->Text.Pos("ATTRIBUTESc");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 11;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has attribute2*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("ATTRIBUTESc");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 11;
                Memo1->SelText = AttributeV12->Text + " " +
                AttributeT12->Text + " " + AttributeN12->Text + ";";
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert ATTRIBUTESc*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos(AttributeN12->Text);
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i+1,
                    "    ATTRIBUTESc");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }
        /*Case No attribute3*/
        if (AttributeN13->Text == ""){
```

```
            SelPos = Memo1->Lines->Text.Pos("ATTRIBUTESc");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 11;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has attribute3*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("ATTRIBUTESc");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 11;
                Memo1->SelText = AttributeV13->Text + " " +
                AttributeT13->Text + " " + AttributeN13->Text + ";";
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert blank line*/
/*                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos(AttributeN13->Text);
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i+1,
                    "");
ShowMessage("step");
                    SelPos = 0;
                }*/
            }
        }
        /*Apply rule1:declare method that sequence diagram describes*/
        SelPos = Memo1->Lines->Text.Pos("METHODSc");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 8;
            Memo1->SelText = "public " + ReturnT11->Text +
            " " + MethodN11->Text + "(PARAMSo){";
            SelPos = 0;
            AnsiString LineStr;
            int i;
            for(i=0;i<Memo1->Lines->Count;i++){
```

```
                LineStr = Memo1->Lines->Strings[i];
                SelPos = LineStr.Pos("public " + ReturnT11->Text +
            " " + MethodN11->Text + "(PARAMSo){");
                if(SelPos>0) break;
            }
            if (SelPos > 0){
                Memo1->Lines->Insert(i+1, "          SEQUENCEc");
                Memo1->Lines->Insert(i+2, "     }");
ShowMessage("step");
                SelPos = 0;
            }
        }
        /*Apply rule1:declare parameters of method that
        sequence diagram describes*/
        /*Case has parameter1*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 7;
            if (ParameterN11->Text!=""){
                Memo1->SelText = ParameterT11->Text + " " +
                ParameterN11->Text + ", PARAMSo";
            }else{
        /*Case no parameter1*/
                Memo1->SelText = "";
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Case has parameter2*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 7;
            if (ParameterN12->Text!=""){
                Memo1->SelText = ParameterT12->Text + " " +
                ParameterN12->Text + ", PARAMSo";
            }else{
        /*Case no parameter2*/
                SelPos = 0;
                SelPos = Memo1->Lines->Text.Pos(", PARAMSo");
                if (SelPos >0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 9;
                    Memo1->SelText = "";
```

```
                }
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Case has parameter3*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 7;
            if (ParameterN13->Text!=""){
                Memo1->SelText = ParameterT13->Text + " " +
                ParameterN13->Text;
            }else{
        /*Case no parameter3*/
                SelPos = 0;
                SelPos = Memo1->Lines->Text.Pos(", PARAMSo");
                if (SelPos >0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 9;
                    Memo1->SelText = "";
                }
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Apply rule2*/
        SelPos = Memo1->Lines->Text.Pos("SEQUENCEc");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
            Memo1->SelText = "LOCALVAR_DECLo ";
            SelPos = 0;
            AnsiString LineStr;
            int i;
            for(i=0;i<Memo1->Lines->Count;i++){
                LineStr = Memo1->Lines->Strings[i];
                SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                if(SelPos>0) break;
            }
            if (SelPos > 0){
                Memo1->Lines->Insert(i+1,
                "        *****");
                Memo1->Lines->Insert(i+2,
                "        MESSAGE_INVOCo");
```

```
ShowMessage("step");
                    SelPos = 0;
                }
        }
        /*Disable button, enable other speedbuttons, change panel caption*/
        Generate1->Enabled = false;
        SpeedButton2->Enabled = true;
        SpeedButton3->Enabled = true;
        SpeedButton4->Enabled = true;
        SpeedButton5->Enabled = true;
        SpeedButton6->Enabled = true;
        Panel1->Caption = "Please select next part of sequence diagram.";

}
//---------------------------------------------------------------------------


void __fastcall TForm1::Generate2Click(TObject *Sender)
{

        /*Apply rule3*/
        if (ConditionS21->Text!=""){
                /*Condition1*/
                int SelPos = Memo1->Lines->Text.Pos("MESSAGE_INVOCo");
                if (SelPos > 0){
                        Memo1->SelStart = SelPos - 1;
                        Memo1->SelLength = 14;
                        Memo1->SelText = "if (" + ConditionS21->Text + "){";
                        countIf = 1;
                        SelPos = 0;
ShowMessage("step");
                        /*Find line to insert MESSAGE_INVOCo and ELSEIFSo*/
                        AnsiString LineStr;
                        int i,ii;
                        for(ii=0;ii<Memo1->Lines->Count;ii++){
                                LineStr = Memo1->Lines->Strings[ii];
                                int SelPos2 = LineStr.Pos(ConditionS21->Text);
                                if(SelPos2>0){
                                        i = ii;
                                        SelPos = SelPos2;
                                }
                        }
                        if (SelPos > 0){
                                Memo1->Lines->Insert(i+1,
```

```
                    "              MESSAGE_INVOCo");
                    Memo1->Lines->Insert(i+2,
                    "         }ELSEIFSo");
                    Memo1->Lines->Insert(i+3,
                    "         MESSAGE_INVOCo");
ShowMessage("step");
                    SelPos = 0;
               }
          }
          /*Case has Condition2*/
          SelPos = Memo1->Lines->Text.Pos("ELSEIFSo");
          if (SelPos > 0){
               Memo1->SelStart = SelPos - 1;
               Memo1->SelLength = 8;
               if (ConditionS22->Text!=""){
                    Memo1->SelText = "else if(" +
                    ConditionS22->Text + ")";
                    countIf = 2;
ShowMessage("step");
                    /*Find line to insert MESSAGE_INVOCo and
                    ELSEIFSo*/
                    AnsiString LineStr;
                    int i,ii;
                    for(ii=0;ii<Memo1->Lines->Count;ii++){
                         LineStr = Memo1->Lines->Strings[ii];
                         int SelPos2 = LineStr.Pos
                         (ConditionS22->Text);
                         if(SelPos2>0){
                              i = ii;
                              SelPos = SelPos2;
                         }
                    }
                    if (SelPos > 0){
                         Memo1->Lines->Insert(i+1,
                         "              MESSAGE_INVOCo");
                         Memo1->Lines->Insert(i+2,
                         "         }ELSEIFSo");
ShowMessage("step");
                         SelPos = 0;
                    }
               }else{
          /*Case no Condition2*/
                    Memo1->SelText = "";
               }
          }
```

```
            /*Case has Condition3*/
            SelPos = Memo1->Lines->Text.Pos("ELSEIFSo");
            if (SelPos > 0){
                  Memo1->SelStart = SelPos - 1;
                  Memo1->SelLength = 8;
                  if (ConditionS23->Text!=""){
                        Memo1->SelText = "else if(" +
                        ConditionS23->Text + ")";
                        countIf = 3;
ShowMessage("step");
                        /*Find line to insert MESSAGE_INVOCo*/
                        AnsiString LineStr;
                        int i,ii;
                        for(ii=0;ii<Memo1->Lines->Count;ii++){
                              LineStr = Memo1->Lines->Strings[ii];
                              int SelPos2 = LineStr.Pos
                              (ConditionS23->Text);
                              if(SelPos2>0){
                                    i = ii;
                                    SelPos = SelPos2;
                              }
                        }
                        if (SelPos > 0){
                              Memo1->Lines->Insert(i+1,
                              "            MESSAGE_INVOCo");
                              Memo1->Lines->Insert(i+2,
                              "        }");
ShowMessage("step");
                              SelPos = 0;
                        }
                  }else{
            /*Case no Condition2*/
                        Memo1->SelText = "";
                  }
            }

            /*Disable button, change panel caption*/
            Generate2->Enabled = false;
            Panel1->Caption =
            "Please select next part of sequence diagram.";
      }


}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Generate3Click(TObject *Sender)
{
    /*Apply rule6:splitting local var*/
    int SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo ");
    if (SelPos > 0){
        Memo1->SelStart = SelPos - 1;
        Memo1->SelLength = 15;
        Memo1->SelText = "LOCALVAR_DECLo,m";
        SelPos = 0;
ShowMessage("step");
        /*Find line to insert LOCALVAR_DECLo*/
        AnsiString LineStr;
        int i;
        for(i=0;i<Memo1->Lines->Count;i++){
            LineStr = Memo1->Lines->Strings[i];
            SelPos = LineStr.Pos("LOCALVAR_DECLo,m");
            if(SelPos>0) break;
        }
        if (SelPos > 0){
            Memo1->Lines->Insert(i+1,"        LOCALVAR_DECLo ");
        }
    }
    /*Apply rule6:declare received parameter of constructor*/
    /*Case No parameter1*/
    if (ParameterN31->Text == ""){
        SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 16;
            Memo1->SelText = "";
ShowMessage("step");
            SelPos = 0;
        }
    /*Case has parameter1*/
    }else{
        SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 16;
            int SelPos2 = Memo1->Lines->Text.Pos
            (ParameterT31->Text + " " + ParameterN31->Text);
            if (SelPos2 == 0){
                Memo1->SelText = ParameterT31->Text + " " +
                ParameterN31->Text + ";";
            }else{
```

```
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo,m*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "       LOCALVAR_DECLo,m");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }
        /*Case No parameter2*/
        if (ParameterN32->Text == ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has parameter2*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                int SelPos2 = Memo1->Lines->Text.Pos
                (ParameterT32->Text + " " + ParameterN32->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ParameterT32->Text + " " +
                    ParameterN32->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
```

```
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo,m*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "       LOCALVAR_DECLo,m");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }
        /*Case No parameter3*/
        if (ParameterN33->Text == ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has parameter3*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                int SelPos2 = Memo1->Lines->Text.Pos
                (ParameterT33->Text + " " + ParameterN33->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ParameterT33->Text + " " +
                    ParameterN33->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert blank line*/
/*                  AnsiString LineStr;
```

```
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "");
ShowMessage("step");
                    SelPos = 0;
                }*/
            }
        }
        /*Apply rule6:message_invoc part*/
        SelPos = Memo1->Lines->Text.Pos("MESSAGE_INVOCo");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 14;
            Memo1->SelText = "ASSIGNocnew " + MethodN32->Text
            + "(PARAMSo,m);";
            SelPos = 0;
ShowMessage("step");
            /*Find line to insert MESSAGE_INVOCo*/
            AnsiString LineStr;
            int i;
            for(i=0;i<Memo1->Lines->Count;i++){
                LineStr = Memo1->Lines->Strings[i];
                SelPos = LineStr.Pos("ASSIGNocnew " + MethodN32->Text
                + "(PARAMSo,m);");
                if(SelPos>0) break;
            }
            if (SelPos > 0){
                Memo1->Lines->Insert(i+1,
                "       MESSAGE_INVOCo");
            }
        }
        /*Apply rule5: declare received var*/
        /*Case has received var*/
        if (ReceivedV31->Text != ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo ");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 15;
                int SelPos2 = Memo1->Lines->Text.Pos
```

```
                (ClassN31->Text + " " + ReceivedV31->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ClassN31->Text + " " +
                    ReceivedV31->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("******");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "       LOCALVAR_DECLo ");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }

        /*Apply rule5:assign value*/
        SelPos = Memo1->Lines->Text.Pos("ASSIGNoc");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 8;
            if (ReceivedV31->Text!=""){
                Memo1->SelText = ReceivedV31->Text + " = ";
            }else{
                Memo1->SelText = "";
            }
            SelPos = 0;
        }
ShowMessage("step");
        /*Apply rule6:declare parameters of constructor*/
        /*Case has parameter1*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
```

```
            if (ParameterN31->Text!=""){
                Memo1->SelText = ParameterN31->Text +
                ", PARAMSo,m";
            }else{
        /*Case no parameter1*/
                Memo1->SelText = "";
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Case has parameter2*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
            if (ParameterN32->Text!=""){
                Memo1->SelText = ParameterN32->Text +
                ", PARAMSo,m";
            }else{
        /*Case no parameter2*/
                SelPos = 0;
                SelPos = Memo1->Lines->Text.Pos(", PARAMSo,m");
                if (SelPos >0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 11;
                    Memo1->SelText = "";
                }
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Case has parameter3*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
            if (ParameterN33->Text!=""){
                Memo1->SelText = ParameterN33->Text;
            }else{
        /*Case no parameter3*/
                SelPos = 0;
                SelPos = Memo1->Lines->Text.Pos(", PARAMSo,m");
                if (SelPos >0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 11;
```

```
                    Memo1->SelText = "";
                }
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Delete MESSAGE_INVOCo*/
        if (countIf>0){
            int SelPos = Memo1->Lines->Text.Pos("MESSAGE_INVOCo");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 14;
                Memo1->SelText = "";
                countIf = countIf-1;
            }
        }
        /*Disable button, change panel caption*/
        Generate3->Enabled = false;
        Panel1->Caption = "Please select next part of sequence diagram.";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Generate4Click(TObject *Sender)
{
        /*Apply rule7:splitting local var*/
        int SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo ");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 15;
            Memo1->SelText = "LOCALVAR_DECLo,m";
            SelPos = 0;
ShowMessage("step");
            /*Find line to insert LOCALVAR_DECLo*/
            AnsiString LineStr;
            int i;
            for(i=0;i<Memo1->Lines->Count;i++){
                LineStr = Memo1->Lines->Strings[i];
                SelPos = LineStr.Pos("LOCALVAR_DECLo,m");
                if(SelPos>0) break;
            }
            if (SelPos > 0){
                Memo1->Lines->Insert(i+1,
                "        LOCALVAR_DECLo ");
            }
```

```
        }
        /*Apply rule7:declare received parameter of called method*/
        /*Case No parameter1*/
        if (ParameterN41->Text == ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has parameter1*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                int SelPos2 = Memo1->Lines->Text.Pos
                (ParameterT41->Text + " " + ParameterN41->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ParameterT41->Text + " " +
                    ParameterN41->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo,m*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "       LOCALVAR_DECLo,m");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }
        /*Case No parameter2*/
```

```
        if (ParameterN42->Text == ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has parameter2*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                int SelPos2 = Memo1->Lines->Text.Pos
                (ParameterT42->Text + " " + ParameterN42->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ParameterT42->Text + " " +
                    ParameterN42->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo,m*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "       LOCALVAR_DECLo,m");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }
        /*Case No parameter3*/
        if (ParameterN43->Text == ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
```

```
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has parameter3*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                int SelPos2 = Memo1->Lines->Text.Pos
                (ParameterT43->Text + " " + ParameterN43->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ParameterT43->Text + " " +
                    ParameterN43->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert blank line*/
/*                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "");
ShowMessage("step");
                    SelPos = 0;
                }*/
            }
        }
        /*Apply rule7:message_invoc part*/
        SelPos = Memo1->Lines->Text.Pos("MESSAGE_INVOCo");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 14;
            Memo1->SelText = "ASSIGNo" + ObjectN41->Text + "." +
                MethodN41->Text + "(PARAMSo,m);";
```

```
            SelPos = 0;
ShowMessage("step");
            /*Find line to insert MESSAGE_INVOCo*/
            AnsiString LineStr;
            int i;
            for(i=0;i<Memo1->Lines->Count;i++){
                  LineStr = Memo1->Lines->Strings[i];
                  SelPos = LineStr.Pos("ASSIGNo" + ObjectN41->Text + "."
                  + MethodN41->Text + "(PARAMSo,m);");
                  if(SelPos>0) break;
            }
            if (SelPos > 0){
                  Memo1->Lines->Insert(i+1,
                  "         MESSAGE_INVOCo");
            }
      }
      /*Apply rule4: declare received var*/
      /*Case has received var*/
      if (ReceivedV41->Text != ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo ");
            if (SelPos > 0){
                  Memo1->SelStart = SelPos - 1;
                  Memo1->SelLength = 15;
                  int SelPos2 = Memo1->Lines->Text.Pos
                  (ReturnT41->Text + " " + ReceivedV41->Text);
                  if (SelPos2 == 0){
                        Memo1->SelText = ReturnT41->Text + " " +
                        ReceivedV41->Text + ";";
                  }else{
                        Memo1->SelText = "";
                  }
                  SelPos = 0;
ShowMessage("step");
                  /*Find line to insert LOCALVAR_DECLo*/
                  AnsiString LineStr;
                  int i;
                  for(i=0;i<Memo1->Lines->Count;i++){
                        LineStr = Memo1->Lines->Strings[i];
                        SelPos = LineStr.Pos("******");
                        if(SelPos>0) break;
                  }
                  if (SelPos > 0){
                        Memo1->Lines->Insert(i,
                        "         LOCALVAR_DECLo ");
ShowMessage("step");
```

```
                    SelPos = 0;
                }
            }
        }

        /*Apply rule4:assign value*/
        SelPos = Memo1->Lines->Text.Pos("ASSIGNo");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 7;
            if (ReceivedV41->Text!=""){
                Memo1->SelText = ReceivedV41->Text + " = ";
            }else{
                Memo1->SelText = "";
            }
            SelPos = 0;
        }
ShowMessage("step");
        /*Apply rule7:declare parameters of called method*/
        /*Case has parameter1*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
            if (ParameterN41->Text!=""){
                Memo1->SelText = ParameterN41->Text +
                ", PARAMSo,m";
            }else{
        /*Case no parameter1*/
                Memo1->SelText = "";
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Case has parameter2*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
            if (ParameterN42->Text!=""){
                Memo1->SelText = ParameterN42->Text +
                ", PARAMSo,m";
            }else{
        /*Case no parameter2*/
                SelPos = 0;
```

```
                SelPos = Memo1->Lines->Text.Pos(", PARAMSo,m");
                if (SelPos >0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 11;
                    Memo1->SelText = "";
                }
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Case has parameter3*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
            if (ParameterN43->Text!=""){
                Memo1->SelText = ParameterN43->Text;
            }else{
        /*Case no parameter3*/
                SelPos = 0;
                SelPos = Memo1->Lines->Text.Pos(", PARAMSo,m");
                if (SelPos >0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 11;
                    Memo1->SelText = "";
                }
            }
        }
ShowMessage("step");
        SelPos = 0;

        /*Delete MESSAGE_INVOCo*/
        if (countIf>0){
            int SelPos = Memo1->Lines->Text.Pos("MESSAGE_INVOCo");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 14;
                Memo1->SelText = "";
                countIf = countIf-1;
            }
        }
        /*Disable button, change panel caption*/
        Generate4->Enabled = false;
        Panel1->Caption = "Please select next part of sequence diagram.";
```

```
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Generate5Click(TObject *Sender)
{
        /*Apply rule8:splitting local var*/
        int SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo ");
        if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 15;
                Memo1->SelText = "LOCALVAR_DECLo,m";
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                        LineStr = Memo1->Lines->Strings[i];
                        SelPos = LineStr.Pos("LOCALVAR_DECLo,m");
                        if(SelPos>0) break;
                }
                if (SelPos > 0){
                        Memo1->Lines->Insert(i+1,
                        "       LOCALVAR_DECLo ");
                }
        }
        /*Apply rule8:declare received parameter of called method*/
        /*Case No parameter1*/
        if (ParameterN51->Text == ""){
                SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
                if (SelPos > 0){
                        Memo1->SelStart = SelPos - 1;
                        Memo1->SelLength = 16;
                        Memo1->SelText = "";
ShowMessage("step");
                        SelPos = 0;
                }
        /*Case has parameter1*/
        }else{
                SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
                if (SelPos > 0){
                        Memo1->SelStart = SelPos - 1;
                        Memo1->SelLength = 16;
                        int SelPos2 = Memo1->Lines->Text.Pos
                        (ParameterT51->Text + " " + ParameterN51->Text);
```

```
                if (SelPos2 == 0){
                    Memo1->SelText = ParameterT51->Text + " " +
                    ParameterN51->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo,m*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "       LOCALVAR_DECLo,m");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }
        /*Case No parameter2*/
        if (ParameterN52->Text == ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has parameter2*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                int SelPos2 = Memo1->Lines->Text.Pos
                (ParameterT52->Text + " " + ParameterN52->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ParameterT52->Text + " " +
                    ParameterN52->Text + ";";
```

```
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo,m*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "       LOCALVAR_DECLo,m");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }
        /*Case No parameter3*/
        if (ParameterN53->Text == ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                Memo1->SelText = "";
ShowMessage("step");
                SelPos = 0;
            }
        /*Case has parameter3*/
        }else{
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo,m");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 16;
                int SelPos2 = Memo1->Lines->Text.Pos
                (ParameterT53->Text + " " + ParameterN53->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ParameterT53->Text + " " +
                    ParameterN53->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
```

```
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert blank line*/
/*                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("LOCALVAR_DECLo ");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "");
ShowMessage("step");
                    SelPos = 0;
                }*/
            }
        }
        /*Apply rule8:message_invoc part*/
        SelPos = Memo1->Lines->Text.Pos("MESSAGE_INVOCo");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 14;
            Memo1->SelText = "ASSIGNo" + MethodN51->Text +
            "(PARAMSo,m);";
            SelPos = 0;
ShowMessage("step");
            /*Find line to insert MESSAGE_INVOCo*/
            AnsiString LineStr;
            int i;
            for(i=0;i<Memo1->Lines->Count;i++){
                LineStr = Memo1->Lines->Strings[i];
                SelPos = LineStr.Pos("ASSIGNo" + MethodN51->Text +
                "(PARAMSo,m);");
                if(SelPos>0) break;
            }
            if (SelPos > 0){
                Memo1->Lines->Insert(i+1,
                "        MESSAGE_INVOCo");
            }
        }
        /*Apply rule4: declare received var*/
        /*Case has received var*/
        if (ReceivedV51->Text != ""){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo ");
```

```
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 15;
                int SelPos2 = Memo1->Lines->Text.Pos
                (ReturnT51->Text + " " + ReceivedV51->Text);
                if (SelPos2 == 0){
                    Memo1->SelText = ReturnT51->Text + " " +
                    ReceivedV51->Text + ";";
                }else{
                    Memo1->SelText = "";
                }
                SelPos = 0;
ShowMessage("step");
                /*Find line to insert LOCALVAR_DECLo*/
                AnsiString LineStr;
                int i;
                for(i=0;i<Memo1->Lines->Count;i++){
                    LineStr = Memo1->Lines->Strings[i];
                    SelPos = LineStr.Pos("******");
                    if(SelPos>0) break;
                }
                if (SelPos > 0){
                    Memo1->Lines->Insert(i,
                    "       LOCALVAR_DECLo ");
ShowMessage("step");
                    SelPos = 0;
                }
            }
        }

        /*Apply rule4:assign value*/
        SelPos = Memo1->Lines->Text.Pos("ASSIGNo");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 7;
            if (ReceivedV51->Text!=""){
                Memo1->SelText = ReceivedV51->Text + " = ";
            }else{
                Memo1->SelText = "";
            }
            SelPos = 0;
        }
ShowMessage("step");
        /*Apply rule8:declare parameters of called method*/
        /*Case has parameter1*/
```

```
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
            if (ParameterN51->Text!=""){
                Memo1->SelText = ParameterN51->Text +
                ", PARAMSo,m";
            }else{
        /*Case no parameter1*/
                Memo1->SelText = "";
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Case has parameter2*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 9;
            if (ParameterN52->Text!=""){
                Memo1->SelText = ParameterN52->Text +
                ", PARAMSo,m";
            }else{
        /*Case no parameter2*/
                SelPos = 0;
                SelPos = Memo1->Lines->Text.Pos(", PARAMSo,m");
                if (SelPos >0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 11;
                    Memo1->SelText = "";
                }
            }
        }
ShowMessage("step");
        SelPos = 0;
        /*Case has parameter3*/
        SelPos = Memo1->Lines->Text.Pos("PARAMSo,m");
        if (SelPos > 0){
            Memo1->SelStart = SelPos - 1;
            Memo1->SelLength = 10;
            if (ParameterN53->Text!=""){
                Memo1->SelText = ParameterN53->Text;
            }else{
        /*Case no parameter3*/
                SelPos = 0;
```

```
                SelPos = Memo1->Lines->Text.Pos(", PARAMSo,m");
                if (SelPos >0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 11;
                    Memo1->SelText = "";
                }
            }
        }
ShowMessage("step");
        SelPos = 0;

        /*Delete MESSAGE_INVOCo*/
        if (countIf>0){
            int SelPos = Memo1->Lines->Text.Pos("MESSAGE_INVOCo");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 14;
                Memo1->SelText = "";
                countIf = countIf-1;
            }
        }
        /*Disable button, change panel caption*/
        Generate5->Enabled = false;
        Panel1->Caption = "Please select next part of sequence diagram.";

}
//---------------------------------------------------------------------------


void __fastcall TForm1::Generate6Click(TObject *Sender)
{
        /*Apply rule9:replace LOCALVAR_DECLo with empty set*/
        bool stopReplace = false;
        int SelPos;
        while (stopReplace == false){
            SelPos = Memo1->Lines->Text.Pos("LOCALVAR_DECLo ");
            if (SelPos > 0){
                Memo1->SelStart = SelPos - 1;
                Memo1->SelLength = 15;
                Memo1->SelText = "";
                SelPos = 0;
ShowMessage("step");
            }else{
                stopReplace = true;
            }
```

```
        }
        /*Apply rule9:replace MESSAGE_INVOCo with empty set*/
        stopReplace = false;
        while (stopReplace == false){
              SelPos = Memo1->Lines->Text.Pos("MESSAGE_INVOCo");
              if (SelPos > 0){
                    Memo1->SelStart = SelPos - 1;
                    Memo1->SelLength = 14;
                    Memo1->SelText = "";
                    SelPos = 0;
ShowMessage("step");
              }else{
                    stopReplace = true;
              }
        }
        /*Clear ***** */
        SelPos = Memo1->Lines->Text.Pos("*****");
        if (SelPos > 0){
              Memo1->SelStart = SelPos - 1;
              Memo1->SelLength = 5;
              Memo1->SelText = "";
              SelPos = 0;
ShowMessage("step");
        }

        /*Disable button, change panel caption*/
        Generate6->Enabled = false;
        SpeedButton1->Enabled = false;
        SpeedButton2->Enabled = false;
        SpeedButton3->Enabled = false;
        SpeedButton4->Enabled = false;
        SpeedButton5->Enabled = false;
        SpeedButton6->Enabled = false;
        Panel1->Caption = "The process of generating code is complete....."
        "Thank you.";

}
//---------------------------------------------------------------------------


void __fastcall TForm1::Clear1Click(TObject *Sender)
{

        ClassN11->Text = "";
```

```
        AttributeV11->Text = "public";
        AttributeT11->Text = "int";
        AttributeN11->Text = "";
        AttributeV12->Text = "public";
        AttributeT12->Text = "int";
        AttributeN12->Text = "";
        AttributeV13->Text = "public";
        AttributeT13->Text = "int";
        AttributeN13->Text = "";

        ReturnT11->Text = "";
        MethodN11->Text = "";
        ParameterT11->Text = "int";
        ParameterN11->Text = "";
        ParameterT12->Text = "int";
        ParameterN12->Text = "";
        ParameterT13->Text = "int";
        ParameterN13->Text = "";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Clear2Click(TObject *Sender)
{

        ConditionS21->Text = "";
        ConditionS22->Text = "";
        ConditionS23->Text = "";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Clear3Click(TObject *Sender)
{

        ReceivedV31->Text = "";
        MethodN31->Text = "";
        ParameterN31->Text = "";
        ParameterN32->Text = "";
        ParameterN33->Text = "";

        ClassN31->Text = "";

        MethodN32->Text = "";
        ParameterT31->Text = "int";
```

```
        ParameterN34->Text = "";
        ParameterT32->Text = "int";
        ParameterN35->Text = "";
        ParameterT33->Text = "int";
        ParameterN36->Text = "";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Clear4Click(TObject *Sender)
{

        ReceivedV41->Text = "";
        MethodN41->Text = "";
        ParameterN41->Text = "";
        ParameterN42->Text = "";
        ParameterN43->Text = "";
        ObjectN41->Text = "";

        ClassN41->Text = "";

        ReturnT41->Text = "void";
        MethodN42->Text = "";
        ParameterT41->Text = "int";
        ParameterN44->Text = "";
        ParameterT42->Text = "int";
        ParameterN45->Text = "";
        ParameterT43->Text = "int";
        ParameterN46->Text = "";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Clear5Click(TObject *Sender)
{

        ReceivedV51->Text = "";
        MethodN51->Text = "";
        ParameterN51->Text = "";
        ParameterN52->Text = "";
        ParameterN53->Text = "";

        ReturnT51->Text = "void";
        MethodN52->Text = "";
        ParameterT51->Text = "int";
```

```
        ParameterN54->Text = "";
        ParameterT52->Text = "int";
        ParameterN55->Text = "";
        ParameterT53->Text = "int";
        ParameterN56->Text = "";

}
//---------------------------------------------------------------------------



void __fastcall TForm1::MethodN31Change(TObject *Sender)
{
        MethodN32->Text = MethodN31->Text;
}
//---------------------------------------------------------------------------



void __fastcall TForm1::MethodN41Change(TObject *Sender)
{
        MethodN42->Text = MethodN41->Text;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::MethodN51Change(TObject *Sender)
{
        MethodN52->Text = MethodN51->Text;
}
//---------------------------------------------------------------------------


void __fastcall TForm1::New1Click(TObject *Sender)
{
        if(Memo1->Modified){
                int Result = MessageBox(NULL, "Memo is modified. Do you"
                " want to save change?", "About this Memo", MB_YESNOCANCEL|
                MB_ICONWARNING);
                if (Result == IDYES){
                        Save1Click(Sender);
                        Memo1->Clear();
                }else if (Result == IDNO){
                        Memo1->Clear();
                }else if (Result == IDCANCEL) return;
        }
```

```
Memo1->Clear();
/*Set form as initial*/
Panel1->Caption = "Welcome to SequenceDia2Java program.....Please"
" fill details of class owner of method.";
Notebook1->ActivePage = "1";
SpeedButton1->Enabled = true;
SpeedButton2->Enabled = false;
SpeedButton2->Enabled = false;
SpeedButton3->Enabled = false;
SpeedButton4->Enabled = false;
SpeedButton5->Enabled = false;
SpeedButton6->Enabled = false;
Generate1->Enabled = true;
Generate6->Enabled = true;
SaveDialog1->FileName = "";
/*Clear value*/
Memo1->Modified = false;
countIf = 0;
/*Clear all Edits*/
/*Page1*/
ClassN11->Text = "";
AttributeV11->Text = "public";
AttributeT11->Text = "int";
AttributeN11->Text = "";
AttributeV12->Text = "public";
AttributeT12->Text = "int";
AttributeN12->Text = "";
AttributeV13->Text = "public";
AttributeT13->Text = "int";
AttributeN13->Text = "";

ReturnT11->Text = "void";
MethodN11->Text = "";
ParameterT11->Text = "int";
ParameterN11->Text = "";
ParameterT12->Text = "int";
ParameterN12->Text = "";
ParameterT13->Text = "int";
ParameterN13->Text = "";
/*Page2*/
ConditionS21->Text = "";
ConditionS22->Text = "";
ConditionS23->Text = "";
/*Page3*/
ReceivedV31->Text = "";
```

```
MethodN31->Text = "";
ParameterN31->Text = "";
ParameterN32->Text = "";
ParameterN33->Text = "";

ClassN31->Text = "";

MethodN32->Text = "";
ParameterT31->Text = "int";
ParameterN34->Text = "";
ParameterT32->Text = "int";
ParameterN35->Text = "";
ParameterT33->Text = "int";
ParameterN36->Text = "";

/*Page4*/
ReceivedV41->Text = "";
MethodN41->Text = "";
ParameterN41->Text = "";
ParameterN42->Text = "";
ParameterN43->Text = "";
ObjectN41->Text = "";

ClassN41->Text = "";

ReturnT41->Text = "void";
MethodN42->Text = "";
ParameterT41->Text = "int";
ParameterN44->Text = "";
ParameterT42->Text = "int";
ParameterN45->Text = "";
ParameterT43->Text = "int";
ParameterN46->Text = "";
/*Page5*/
ReceivedV51->Text = "";
MethodN51->Text = "";
ParameterN51->Text = "";
ParameterN52->Text = "";
ParameterN53->Text = "";

ReturnT51->Text = "void";
MethodN52->Text = "";
ParameterT51->Text = "int";
ParameterN54->Text = "";
ParameterT52->Text = "int";
```

```
        ParameterN55->Text = "";
        ParameterT53->Text = "int";
        ParameterN56->Text = "";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Save1Click(TObject *Sender)
{
        if(SaveDialog1->FileName != ""){
                Memo1->Lines->SaveToFile(SaveDialog1->FileName);
                Memo1->Modified = false;
        }else{
                SaveAs1Click(Sender);
        }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SaveAs1Click(TObject *Sender)
{
        if(Memo1->Lines->Count>0){
                if(SaveDialog1->Execute()){
                        Memo1->Lines->SaveToFile(SaveDialog1->FileName);
                        Memo1->Modified = false;
                }
        }
}
//---------------------------------------------------------------------------



void __fastcall TForm1::Exit1Click(TObject *Sender)
{
        if(Memo1->Modified){
                int Result = MessageBox(NULL, "Memo is modified. Do you"
                " want to save change?", "About this Memo", MB_YESNOCANCEL|
                MB_ICONWARNING);
                if (Result == IDYES){
                        Save1Click(Sender);
                        Close();
                }
                if (Result == IDNO) Close();
                if (Result == IDCANCEL) return;
        }else{
                Close();
```

```
        }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::About1Click(TObject *Sender)
{
        Form2->ShowModal();
}
//---------------------------------------------------------------------------
```

รูปที่ ข-1 ชุดคำสั่งจากไฟล์ Mainform.cpp

```
//---------------------------------------------------------------------------

#ifndef MainFormH
#define MainFormH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <ExtCtrls.hpp>
#include <Buttons.hpp>
#include <ComCtrls.hpp>
#include <Dialogs.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:            // IDE-managed Components
        TPanel *Panel1;
        TBevel *Bevel1;
        TSpeedButton *SpeedButton6;
        TSpeedButton *SpeedButton1;
        TSpeedButton *SpeedButton2;
        TSpeedButton *SpeedButton3;
        TSpeedButton *SpeedButton4;
        TSpeedButton *SpeedButton5;
        TMainMenu *MainMenu1;
        TMenuItem *File1;
        TMenuItem *New1;
        TMenuItem *N1;
        TMenuItem *Save1;
        TMenuItem *SaveAs1;
        TMenuItem *N2;
        TMenuItem *Exit1;
```

```
TMenuItem *Help1;
TMenuItem *About1;
TGroupBox *GroupBox2;
TLabel *Label4;
TEdit *ClassN11;
TLabel *Label5;
TEdit *AttributeN11;
TComboBox *AttributeT11;
TLabel *Label6;
TEdit *AttributeN12;
TComboBox *AttributeT12;
TLabel *Label7;
TEdit *AttributeN13;
TComboBox *AttributeT13;
TComboBox *AttributeV11;
TComboBox *AttributeV12;
TComboBox *AttributeV13;
TLabel *Label1;
TEdit *MethodN11;
TLabel *Label2;
TLabel *Label3;
TComboBox *ReturnT11;
TComboBox *ParameterT11;
TEdit *ParameterN11;
TLabel *Label8;
TComboBox *ParameterT12;
TEdit *ParameterN12;
TLabel *Label9;
TLabel *Label10;
TComboBox *ParameterT13;
TEdit *ParameterN13;
TLabel *Label11;
TBevel *Bevel2;
TBevel *Bevel3;
TNotebook *Notebook1;
TGroupBox *GroupBox1;
TLabel *Label12;
TEdit *ConditionS21;
TLabel *Label13;
TEdit *ConditionS22;
TLabel *Label14;
TEdit *ConditionS23;
TGroupBox *GroupBox3;
TLabel *Label15;
TLabel *Label20;
```

```
TLabel *Label21;
TLabel *Label22;
TEdit *ReceivedV31;
TEdit *MethodN31;
TEdit *ParameterN31;
TEdit *ParameterN32;
TEdit *ParameterN33;
TLabel *Label26;
TLabel *Label27;
TGroupBox *GroupBox4;
TLabel *Label17;
TLabel *Label24;
TLabel *Label25;
TLabel *Label28;
TLabel *Label29;
TLabel *Label30;
TLabel *Label31;
TLabel *Label32;
TEdit *ClassN31;
TEdit *MethodN32;
TComboBox *ParameterT31;
TEdit *ParameterN34;
TComboBox *ParameterT32;
TEdit *ParameterN35;
TComboBox *ParameterT33;
TEdit *ParameterN36;
TLabel *Label37;
TBevel *Bevel4;
TGroupBox *GroupBox5;
TLabel *Label38;
TLabel *Label39;
TLabel *Label40;
TLabel *Label41;
TLabel *Label42;
TLabel *Label43;
TLabel *Label44;
TEdit *ReceivedV41;
TEdit *MethodN41;
TEdit *ParameterN41;
TEdit *ParameterN42;
TEdit *ParameterN43;
TEdit *ObjectN41;
TGroupBox *GroupBox6;
TLabel *Label45;
TLabel *Label53;
```

```
TLabel *Label54;
TLabel *Label55;
TLabel *Label56;
TLabel *Label57;
TLabel *Label58;
TLabel *Label59;
TBevel *Bevel6;
TEdit *ClassN41;
TComboBox *ReturnT41;
TEdit *MethodN42;
TComboBox *ParameterT41;
TComboBox *ParameterT43;
TComboBox *ParameterT42;
TEdit *ParameterN44;
TEdit *ParameterN45;
TEdit *ParameterN46;
TLabel *Label46;
TLabel *Label48;
TGroupBox *GroupBox7;
TLabel *Label49;
TLabel *Label50;
TLabel *Label51;
TLabel *Label52;
TLabel *Label60;
TLabel *Label61;
TEdit *ReceivedV51;
TEdit *MethodN51;
TEdit *ParameterN51;
TEdit *ParameterN52;
TEdit *ParameterN53;
TGroupBox *GroupBox8;
TLabel *Label63;
TLabel *Label64;
TLabel *Label65;
TLabel *Label66;
TLabel *Label67;
TLabel *Label68;
TLabel *Label69;
TLabel *Label70;
TComboBox *ReturnT51;
TEdit *MethodN52;
TComboBox *ParameterT51;
TComboBox *ParameterT53;
TComboBox *ParameterT52;
TEdit *ParameterN54;
```

```
        TEdit *ParameterN55;
        TEdit *ParameterN56;
        TButton *Clear1;
        TButton *Generate1;
        TButton *Clear2;
        TButton *Generate2;
        TButton *Clear3;
        TButton *Generate3;
        TButton *Clear4;
        TButton *Generate4;
        TButton *Clear5;
        TButton *Generate5;
        TButton *Generate6;
        TSaveDialog *SaveDialog1;
        TMemo *Memo1;
        void __fastcall SpeedButton1Click(TObject *Sender);
        void __fastcall SpeedButton2Click(TObject *Sender);
        void __fastcall SpeedButton3Click(TObject *Sender);
        void __fastcall SpeedButton4Click(TObject *Sender);
        void __fastcall SpeedButton5Click(TObject *Sender);
        void __fastcall SpeedButton6Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall Generate1Click(TObject *Sender);
        void __fastcall Clear1Click(TObject *Sender);
        void __fastcall Clear2Click(TObject *Sender);
        void __fastcall Clear3Click(TObject *Sender);
        void __fastcall Clear4Click(TObject *Sender);
        void __fastcall Clear5Click(TObject *Sender);
        void __fastcall Generate2Click(TObject *Sender);
        void __fastcall Generate3Click(TObject *Sender);
        void __fastcall Generate4Click(TObject *Sender);
        void __fastcall Generate5Click(TObject *Sender);
        void __fastcall Generate6Click(TObject *Sender);
        void __fastcall MethodN31Change(TObject *Sender);
        void __fastcall MethodN41Change(TObject *Sender);
        void __fastcall MethodN51Change(TObject *Sender);
        void __fastcall About1Click(TObject *Sender);
        void __fastcall Exit1Click(TObject *Sender);
        void __fastcall Save1Click(TObject *Sender);
        void __fastcall SaveAs1Click(TObject *Sender);
        void __fastcall New1Click(TObject *Sender);
private:    // User declarations
public:                 // User declarations
        __fastcall TForm1(TComponent* Owner);
};
```

```
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

รูปที่ ข-2 ชุดคำสั่งจากไฟล์ Mainform.h

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "About.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button1Click(TObject *Sender)
{
        Close();
}
//---------------------------------------------------------------------------
```

รูปที่ ข-3 ชุดคำสั่งจากไฟล์ About.cpp

```
//---------------------------------------------------------------------------

#ifndef AboutH
#define AboutH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:            // IDE-managed Components
        TLabel *Label1;
```

```
        TLabel *Label2;
        TLabel *Label3;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
private:    // User declarations
public:                 // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

รูปที่ ข-4 ชุดคำสั่งจากไฟล์ About.h

ข-3 การทดสอบโปรแกรม

หลังจากเพิ่มเติมข้อความสั่ง เครื่องมือที่ออกแบบไว้จะสามารถสร้างชุดคำสั่งได้ และเพื่อให้แน่ใจว่าเครื่องมือทำงานตรงตามความต้องการ ผู้วิจัยจึงทำการทดลองสร้างชุดคำสั่งจากซีเควนซ์ไดอะแกรม และคลาสไดอะแกรมอย่างน้อย 3 ชุด หากพบข้อผิดพลาด จะทำการแก้ไขข้อความสั่งที่ใช้ในการสร้างเครื่องมือแล้วทดลองสร้างชุดคำสั่งจากซีเควนซ์ และคลาสไดอะแกรมอีกครั้ง ทำเช่นนี้จนกว่าจะได้ผลลัพธ์ชุดคำสั่งที่สมบูรณ์

ภาคผนวก ค.
การประยุกต์ใช้กฎการแปลงยูเอ็มแอลคอแลบบอเรชันไดอะแกรมกับเมทธอดคืนหนังสือของระบบห้องสมุด

1) ประยุกต์ใช้กฎที่ 1 เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายดังแสดงในรูปที่ ค-1

2) ประยุกต์ใช้กฎที่ 2 เมต้ารูลสำหรับการแบ่งซีเควนซ์ดังแสดงในรูปที่ ค-2

3) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปรดังแสดงในรูปที่ ค-3

4) ประยุกต์ใช้กฎที่ 8 เมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเองดังแสดงในรูปที่ ค-4

5) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปรดังแสดงในรูปที่ ค-5

6) ประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังแสดงในรูปที่ ค-6

7) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปรดังแสดงในรูปที่ ค-7

8) ประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังแสดงในรูปที่ ค-8

9) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปรดังแสดงในรูปที่ ค-9

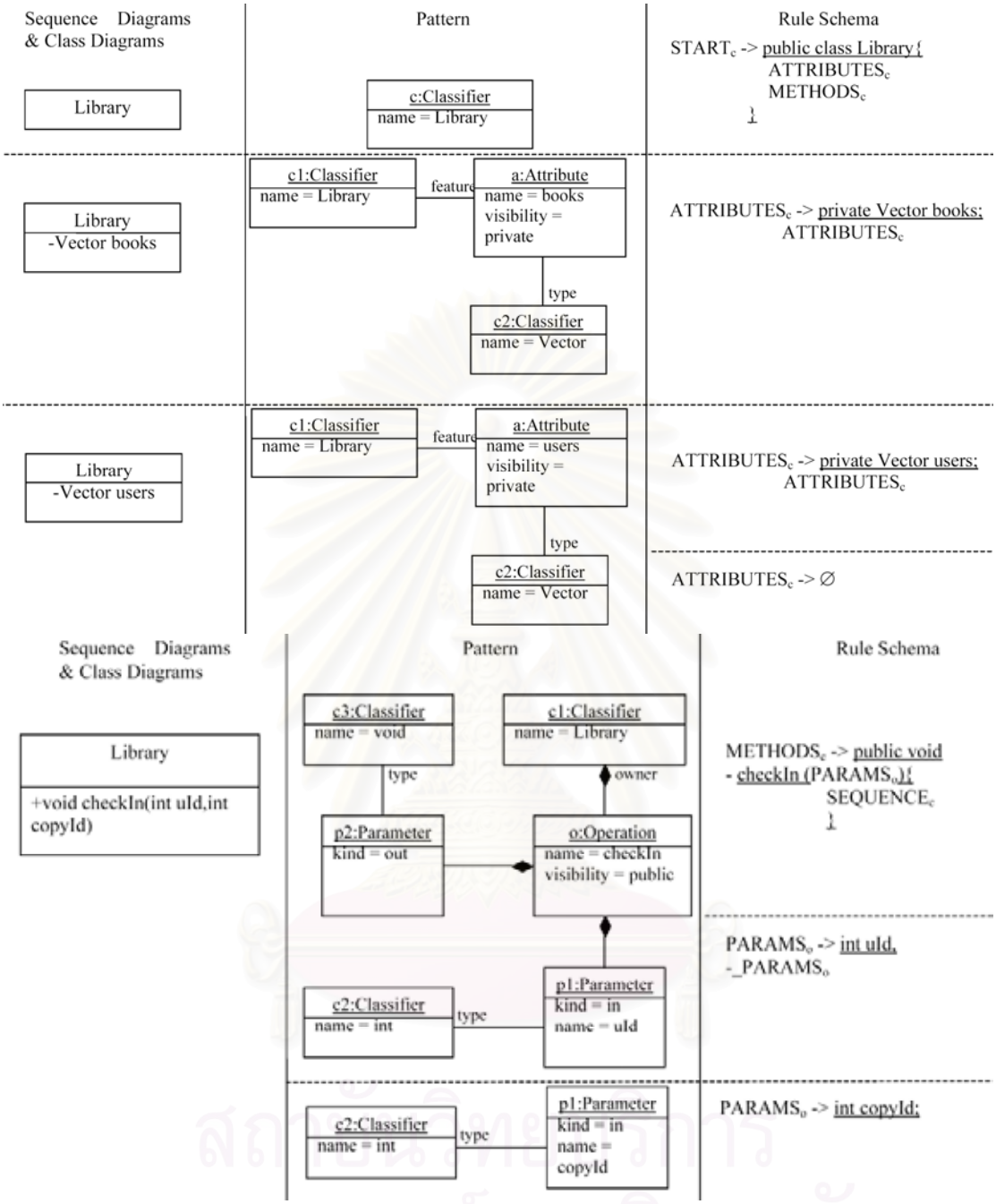
10) ประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังแสดงในรูปที่ ค-10

11) ประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่งดังแสดงในรูปที่ ค-11

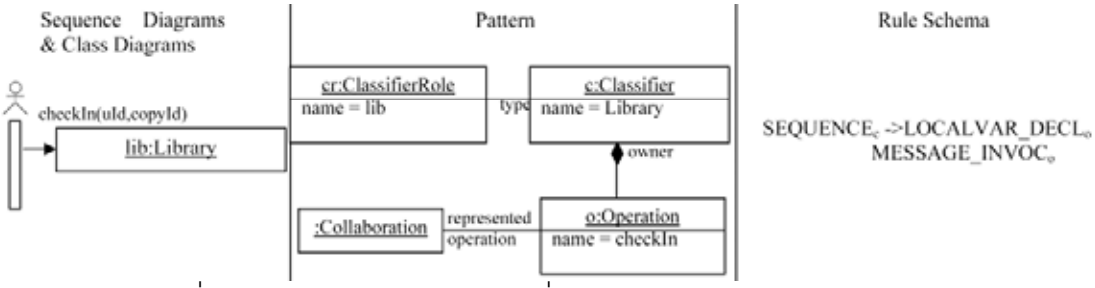
12) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังแสดงในรูปที่ ค-12

13) ประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่งดังแสดงในรูปที่ ค-13

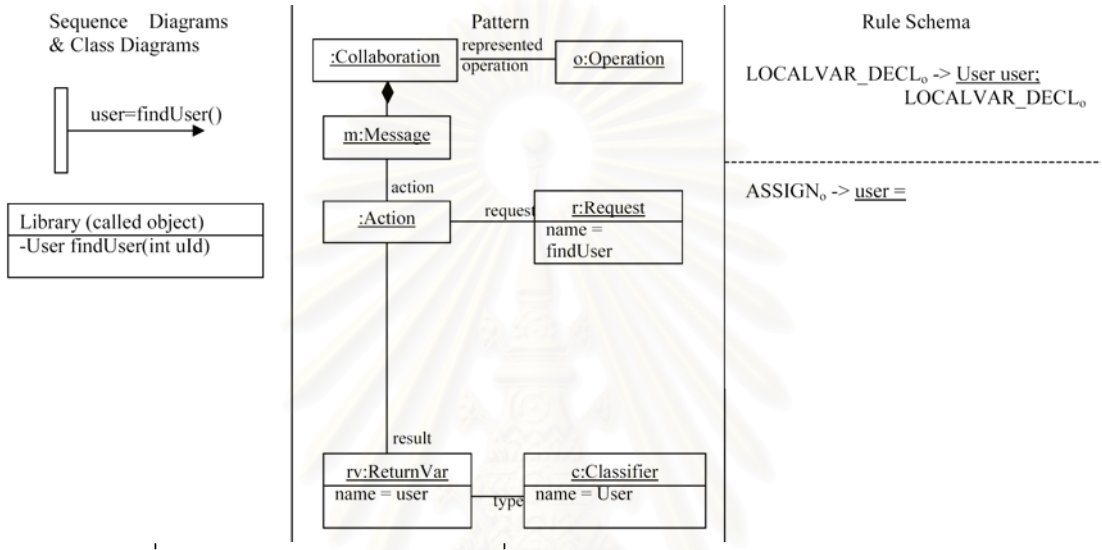
14) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังแสดงในรูปที่ ค-14



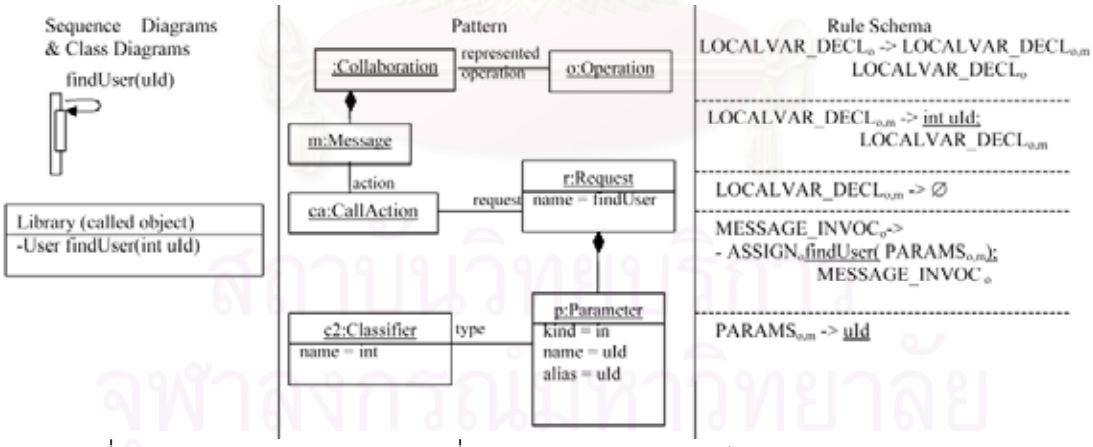
รูปที่ ค-1 แสดงการประยุกต์ใช้กฎที่ 1 เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย



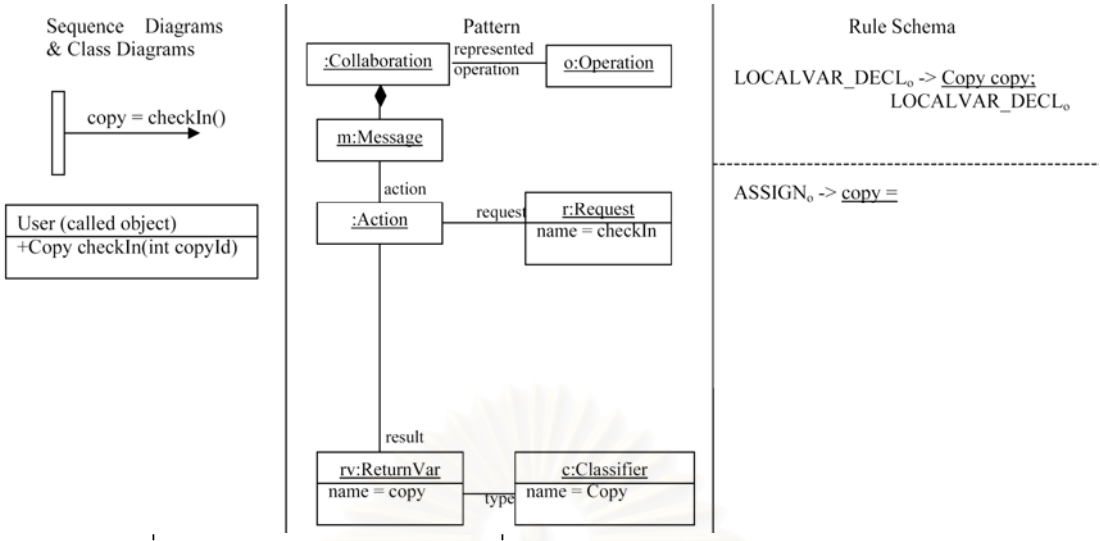
รูปที่ ค-2 แสดงการประยุกต์ใช้กฎที่ 2 เมต้ารูลสำหรับการแบ่งซีเควนซ์



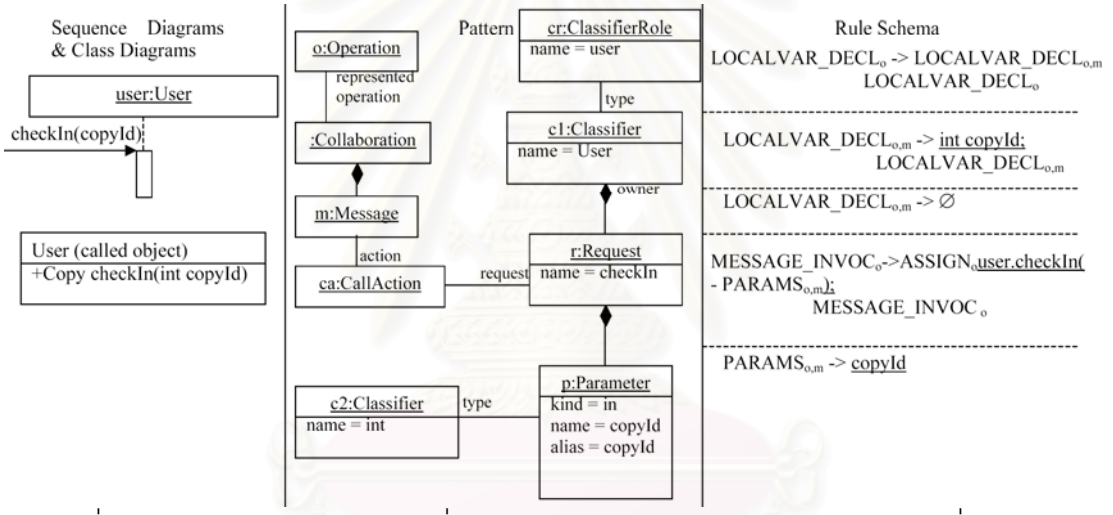
รูปที่ ค-3 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร



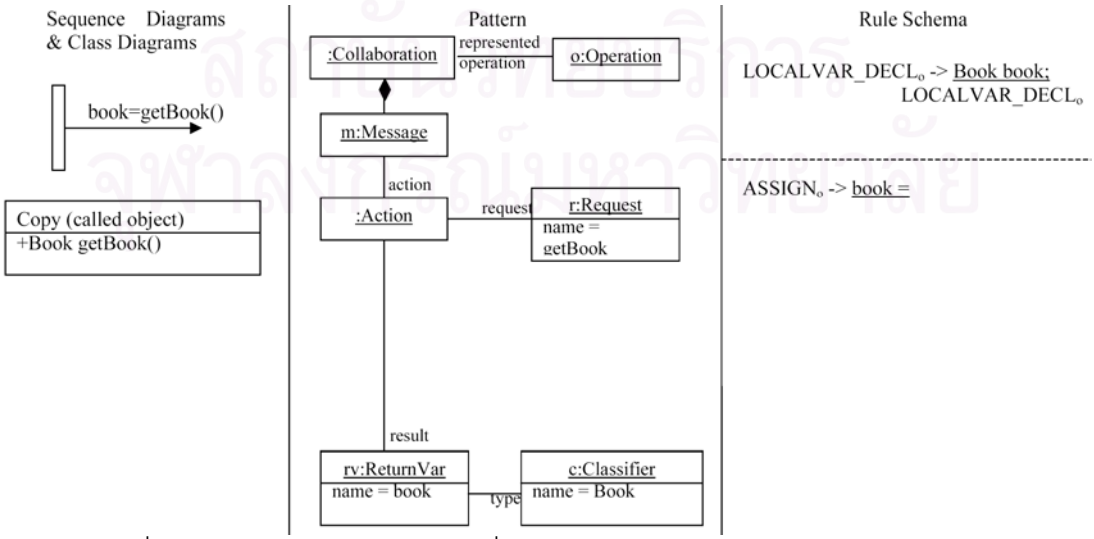
รูปที่ ค-4 แสดงการประยุกต์ใช้กฎที่ 8 เมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเอง



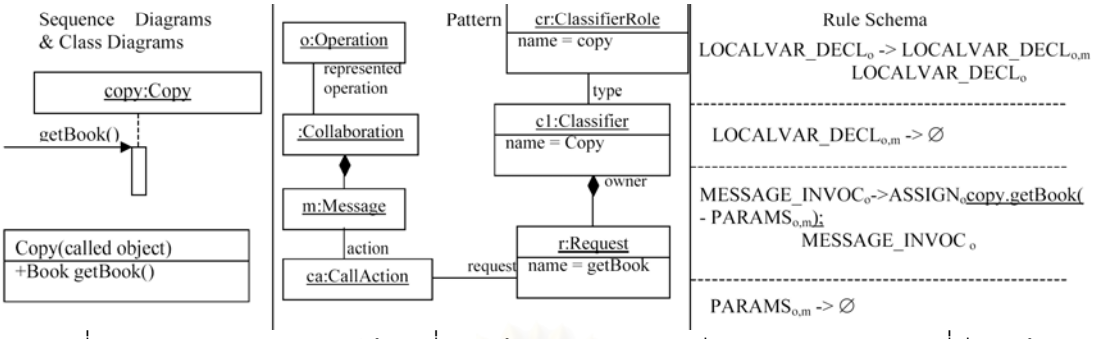
รูปที่ ค-5 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร



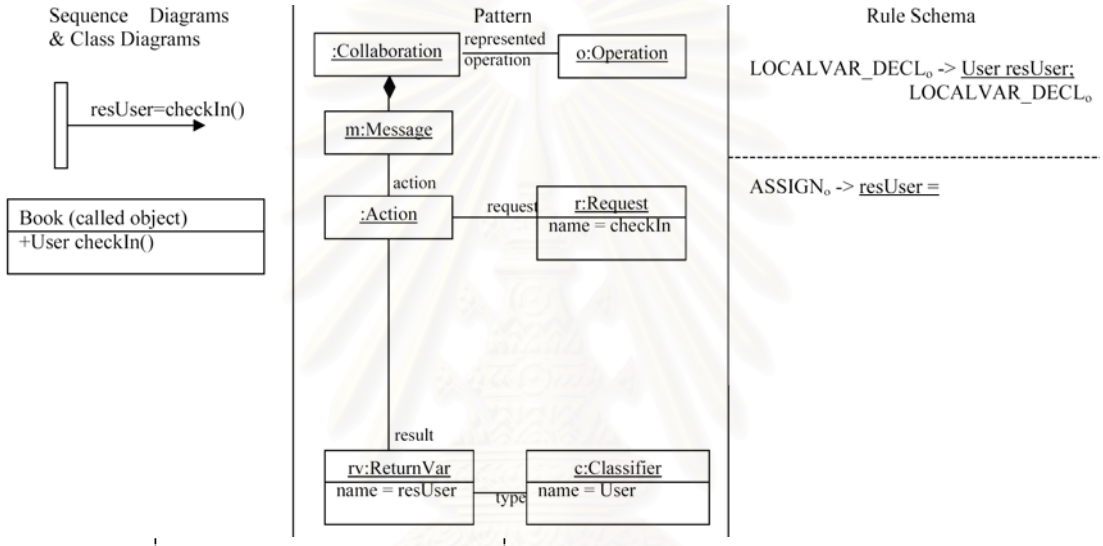
รูปที่ ค-6 แสดงการประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว



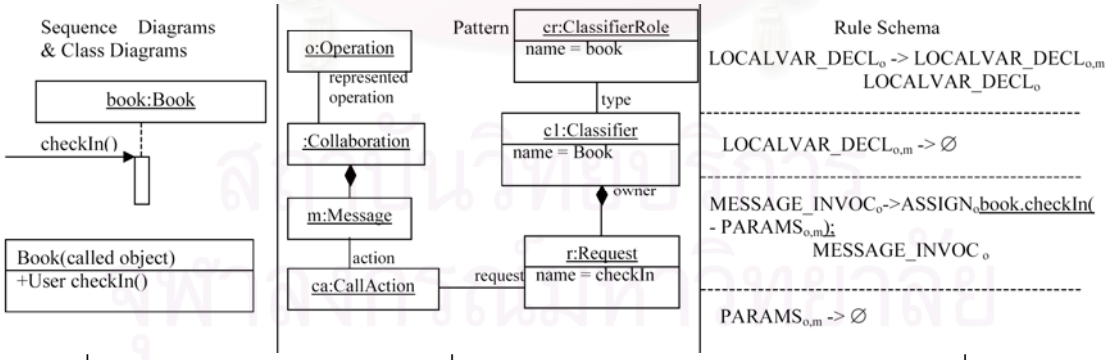
รูปที่ ค-7 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร



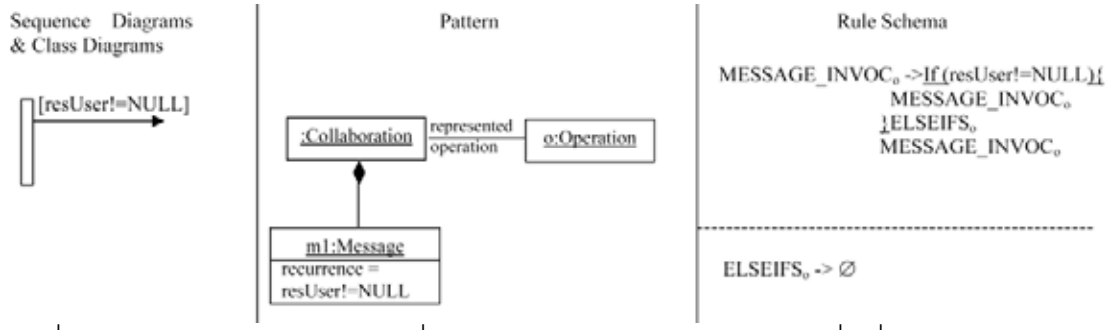
รูปที่ ค-8 แสดงการประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว



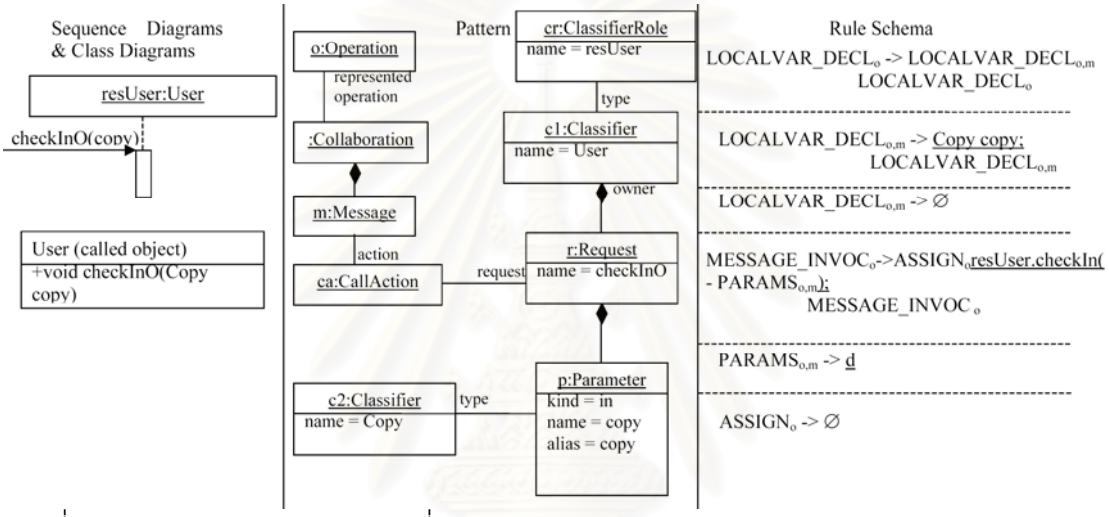
รูปที่ ค-9 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร



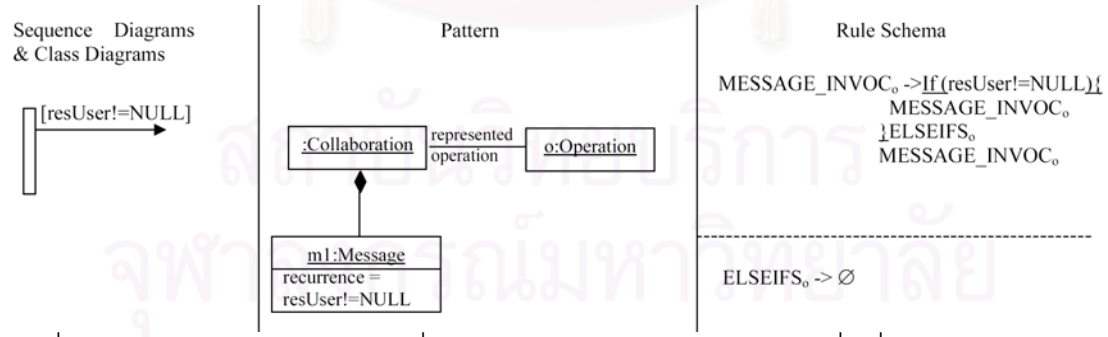
รูปที่ ค-10 แสดงการประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว



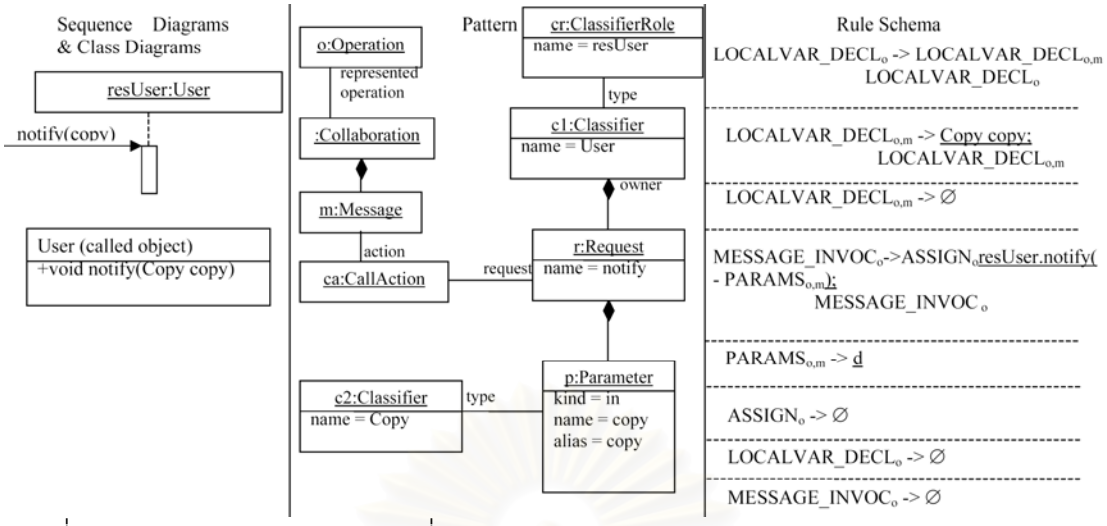
รูปที่ ค-11 แสดงการประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง



รูปที่ ค-12 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว



รูปที่ ค-13 แสดงการประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง



รูปที่ ค-14 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว

ภาคผนวก ง.
การประยุกต์ใช้กฎการแปลงยูเอ็มแอลคอแลบบอเรชันไดอะแกรมกับเมทธอดแสดงของระบบกองไพ่

1) ประยุกต์ใช้กฎที่ 1 เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบายดังแสดงในรูปที่ ง-1

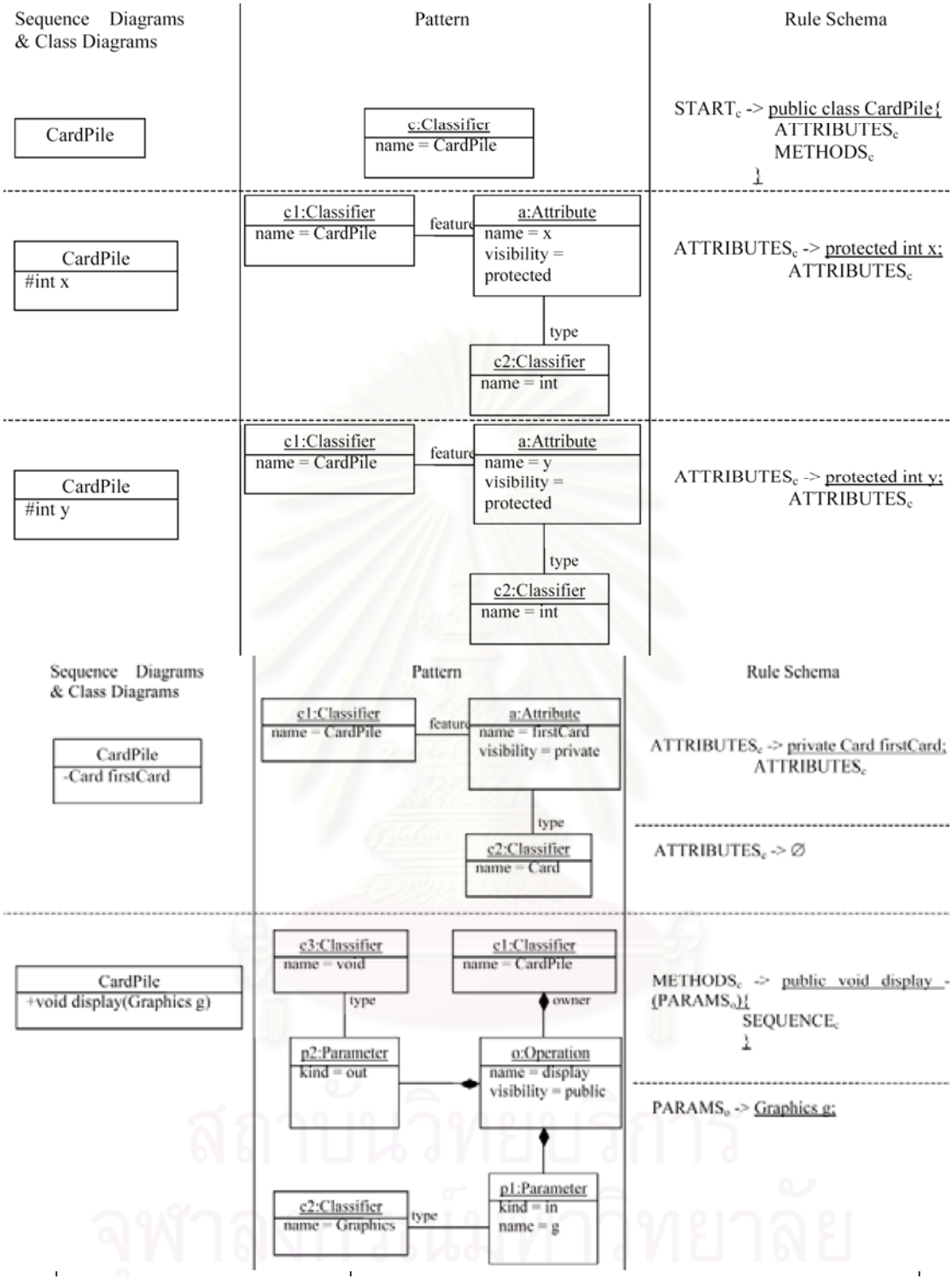
2) ประยุกต์ใช้กฎที่ 2 เมต้ารูลสำหรับการแบ่งซีเควนซ์ดังแสดงในรูปที่ ง-2

3) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังแสดงในรูปที่ ง-3

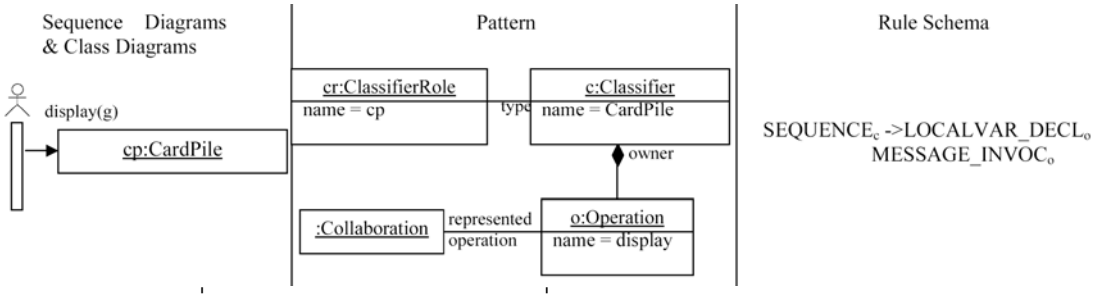
4) ประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่งดังแสดงในรูปที่ ง-4

5) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังแสดงในรูปที่ ง-5

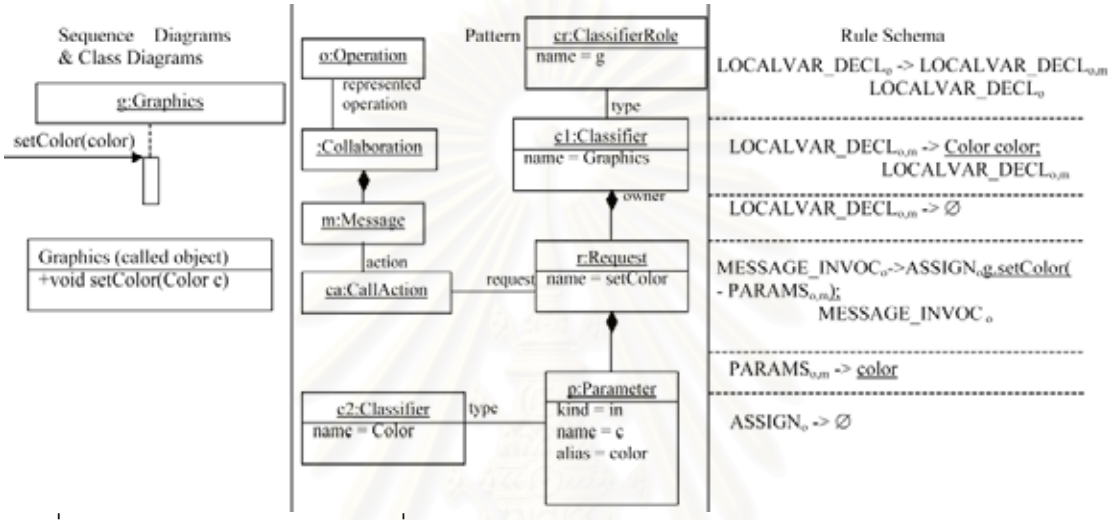
6) ประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วดังแสดงในรูปที่ ง-6



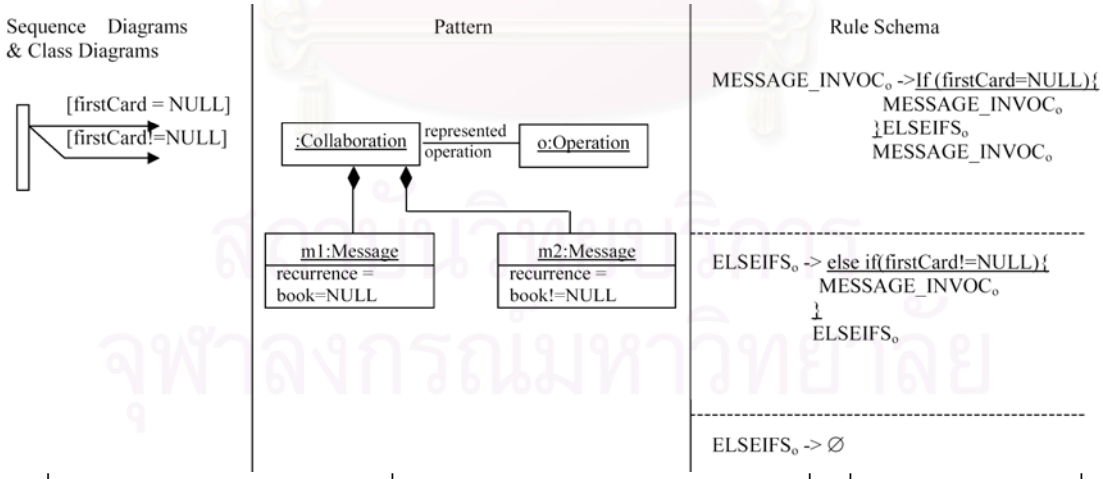
รูปที่ ง-1 แสดงการประยุกต์ใช้กฎที่ 1 เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ซีเควนซ์ไดอะแกรมอธิบาย



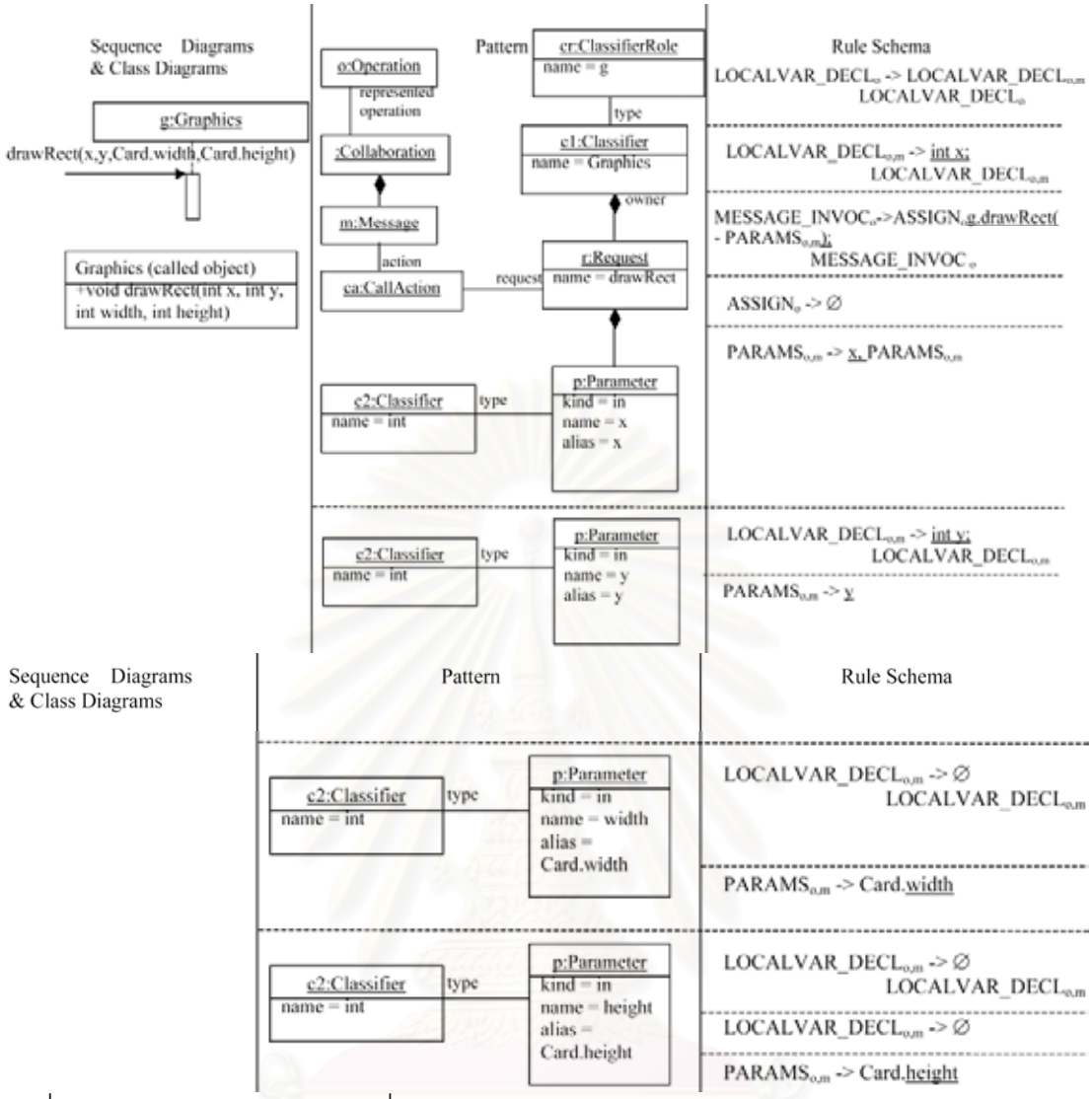
รูปที่ ง-2 แสดงการประยุกต์ใช้กฎที่ 2 เมต้ารูลสำหรับการแบ่งซีเควนซ์



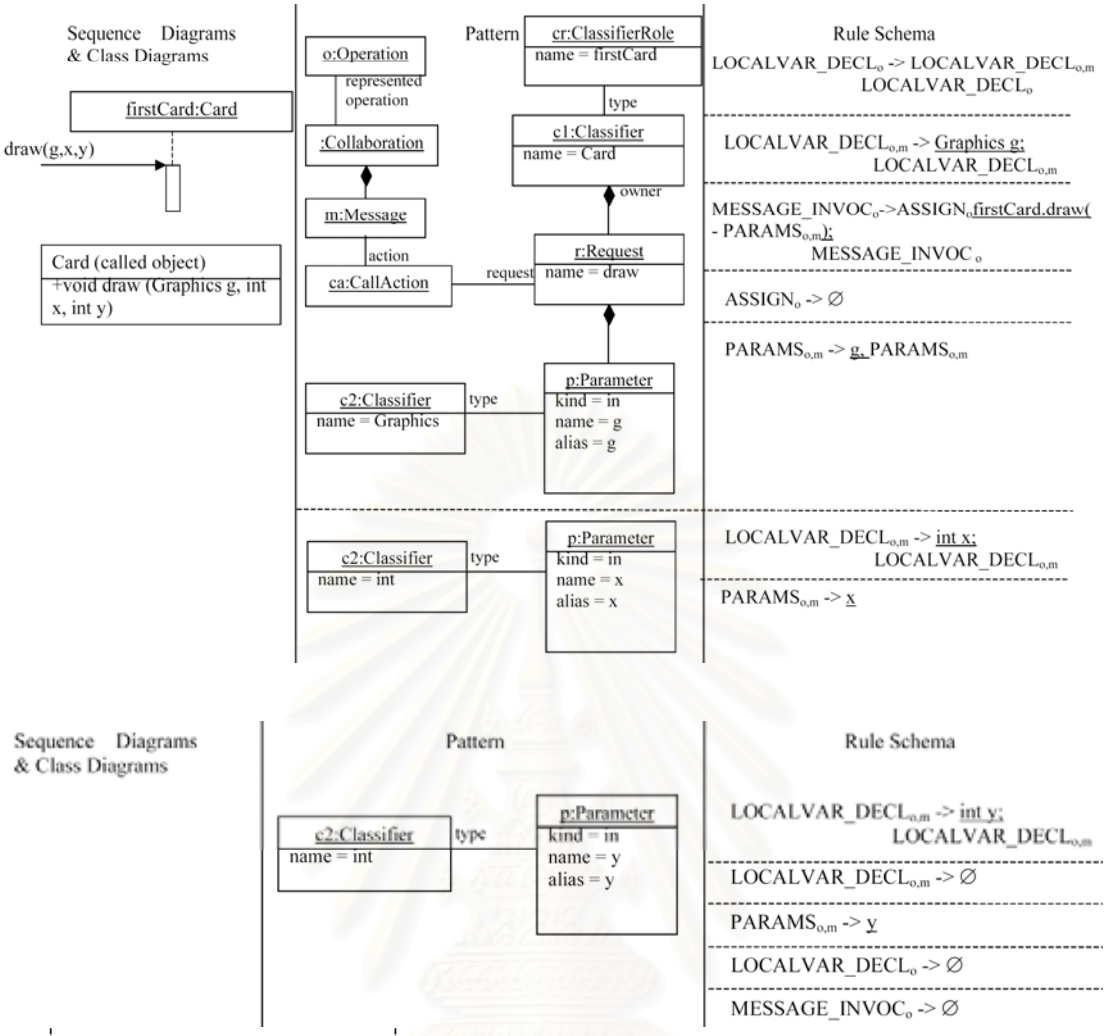
รูปที่ ง-3 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว



รูปที่ ง-4 แสดงการประยุกต์ใช้กฎที่ 3 เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง



รูปที่ ง-5 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว



รูปที่ ง-6 แสดงการประยุกต์ใช้กฎที่ 4 เมต้ารูลสำหรับการกำหนดค่าให้ตัวแปร และประยุกต์ใช้กฎที่ 7 เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้ว

ภาคผนวก จ.
ภาพรวมของกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

รายละเอียดภาพรวมของกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวาแสดงรายละเอียดในรูปที่ จ-1

$START_c$ -> public class c.name{
$ATTRIBUTES_c$
$METHODS_c$
}

$ATTRIBUTES_c$ -> a.visibility c2.name a.name;
$ATTRIBUTES_c$
-> ∅

$METHODS_c$ -> o.visibility c3.name o.name (PARAMS$_o$){
$SEQUENCE_c$
}

$PARAMS_o$ -> c2.name p1.name;
-> c2.name p1.name, PARAMS$_o$;
-> ∅

$SEQUENCE_c$ -> $LOCALVAR\_DECL_o$
$MESSAGE\_INVOC_o$

$MESSAGE\_INVOC_o$ -> if(m1.recurrence){
$MESSAGE\_INVOC_o$
}ELSEIFSo
$MESSAGE\_INVOC_o$
}
-> $ASSIGN_{oc}$ new r.name(PARAMS$_{o,m}$);
$MESSAGE\_INVOC_o$
-> $ASSIGN_{oc}$ r.name r.name(PARAMS$_{o,m}$)
$MESSAGE\_INVOC_o$
-> $ASSIGN_o$ r.name(PARAMS$_{o,m}$)
$MESSAGE\_INVOC_o$
-> ∅

$ELSEIFS_o$ -> else if(m2.recurrence){
$MESSAGE\_INVOC_o$
}$ELSEIFS_o$
-> ∅

$LOCALVAR\_DECL_o$ -> c.name rv.name;
$LOCALVAR\_DECL_o$
-> $LOCALVAR\_DECL_{o,m}$
$LOCALVAR\_DECL_o$

	->	∅
$LOCALVAR\_DECL_{o,m}$	->	c2.name p.alias;
		$LOCALVAR\_DECL_{o,m}$
	->	∅
		$LOCALVAR\_DECL_{o,m}$
	->	∅
$ASSIGN_{o}$	->	rv.name =
	->	∅
$PARAMS_{o,m}$	->	p.alias
	->	p.alias, $PARAMS_{o,m}$
	->	∅

รูปที่ จ-1 แสดงรายละเอียดภาพรวมของกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวมฑุปายาส ทองมาก เกิดวันที่ 8 พฤศจิกายน พ.ศ. 2522 สำเร็จการศึกษาปริญญาตรีบริหารธุรกิจบัณฑิตสาขาระบบสารสนเทศเพื่อการจัดการจากมหาวิทยาลัยธรรมศาสตร์ ในปีการศึกษา 2542 จากนั้นเข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต ที่คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย