วิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมภาษาจาวาจากข้อกำหนดรูปนัยคาเฟโอบีเจ

นายชาติชาย ดวงสอาด

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2546

ISBN 974-17-4233-9

AN APPROACH FOR TRACKING PROGRESS OF JAVA PROGRAM DEVELOPMENT FROM CAFEOBJ SPECIFICATION

Mr. Chartchai Doungsa-ard

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2003

ISBN 974-17-4233-9

หัวข้อวิทยานิพนธ์	วิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมภาษาจาวาจากข้อกำหนดรูปนัยคาเฟโอบีเจ
โดย	นายชาติชาย ดวงสอาด
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

.....กรรมการ
(อาจารย์ ดร.ภัทรชัย ลลิตโรจน์วงศ์)

ชาติชาย ดวงสอาด : วิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมภาษาจาวาจากข้อกำหนดรูปนัยคาเฟโอบีเจ. (AN APPROACH FOR TRACKING PROGRESS OF JAVA PROGRAM DEVELOPMENT FROM CAFEOBJ SPECIFICATION) อ. ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร. ธาราทิพย์ สุวรรณศาสตร์, 149 หน้า. ISBN 974-17-4233-9.

วิทยานิพนธ์นี้มีวัตถุประสงค์ เพื่อออกแบบวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมภาษาจาวาจากข้อกำหนดรูปนัยคาเฟโอบีเจ การตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมเป็นขั้นตอนสำคัญขั้นตอนหนึ่งสำหรับการพัฒนาโปรแกรม เนื่องจากเป็นข้อมูลสำคัญสำหรับการตัดสินใจแก้ไข หรือปรับปรุงแผนการพัฒนาโปรแกรมเพื่อพัฒนาโปรแกรมให้แล้วเสร็จทันในระยะเวลาที่กำหนด การตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมนั้นต้องตรวจสอบ ว่าผู้พัฒนาโปรแกรมได้พัฒนาโปรแกรมซึ่งสอดคล้องกับเอกสารออกแบบระบบหรือไม่

วิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่เสนอในวิทยานิพนธ์นี้ประกอบด้วยขั้นตอน 3 ขั้นตอนได้แก่ ขั้นตอนการสร้างโครงภาษาจาวา ขั้นตอนการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ และขั้นตอนการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม โดยในขั้นตอนการสร้างโครงภาษาจาวา วิทยานิพนธ์นี้เสนอกราฟแสดงความสัมพันธ์ระหว่างมอดูลต่างๆ ในข้อกำหนดรูปนัยคาเฟโอบีเจ และกฎสำหรับการสร้างโครงภาษาจาวา จากส่วนประกาศมอดูล และส่วนประกาศลายเซ็นของข้อกำหนดรูปนัยคาเฟโอบีเจที่แสดงอยู่ในกราฟนั้น เพื่อส่งมอบโครงภาษาจาวาที่ได้ให้กับผู้พัฒนาพัฒนาโปรแกรมต่อไป นอกจากนี้วิทยานิพนธ์นี้ยังเสนอขั้นตอนวิธีการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบจากส่วนประกาศสัจพจน์ของข้อกำหนดรูปนัยคาเฟโอบีเจ เพื่อให้ได้แนวทางในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม และในขั้นตอนการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม วิทยานิพนธ์นี้เสนอขั้นตอนวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมโดยเปรียบเทียบระหว่างซอร์สโคดโปรแกรมที่ผู้พัฒนาพัฒนามาจากโครงภาษาจาวา กับส่วนของโปรแกรมที่คาดว่าจะพบ

ในงานวิทยานิพนธ์นี้ได้พัฒนาเครื่องมือเพื่อทดสอบวิธีการที่นำเสนอ และทดลองตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมจาก ข้อกำหนดการนับ ข้อกำหนดการทำเครื่องหมาย และข้อกำหนดการนับที่มีสวิตช์ ซึ่งผลที่ได้จากการทดลอง สรุปผลได้ว่า เครื่องมือที่พัฒนาขึ้นสามารถตรวจสอบความก้าวหน้าได้ใกล้เคียงกับการตรวจสอบโดยมนุษย์

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา .....

ปีการศึกษา 2546

4470277521 : MAJOR COMPUTER ENGINEERING

KEY WORD: FORMAL METHODD / CAFEOBJ / TRANSFORMING APPROACH / TRANSFORMATION RULE / PROGRESS TRACKING

CHARTCHAI DOUNGSA-ARD : AN APPROACH FOR TRACKING PROGRESS OF JAVA PROGRAM DEVELOPMENT FROM CAFEOBJ SPECIFICATION , THESIS ADVISOR : ASST. PROF. TARATIP SUWANNASART, Ph.D., 149 pp. ISBN 974-17-4233-9.

This thesis proposes an approach for tracking progress of Java program development from CafeOBJ specification. Tracking progress of program development is an important part to complete the software project on time. The tracking progress is a comparison if the developed source code is consistent with the design document.

The proposed approach consists of 3 parts: the Java template code generation part, the expected source code generation, and the progress tracking part. In the Java template code generation part, the transformation rules for transforming the syntactic part of CafeOBJ specification to Java template code are proposed. The Java template code is given to the developer to implement the program, which is consistent to the specificaition. The semantic part of the specification is transformed to the expected source code by the expected source code generation algorithm. The expected source code is a guideline for tracking progress. Finally, the code generation part compares the progress of the source code development between the source code, which the developer has implemented, and the expected source code using the proposed progress tracking algorithm.

The tools, which apply the proposed techniques, are developed for testing the proposed approach. The progress of program, which is designed by the COUNTER specification, the FLAG specification, and the COUNTER-WITH-SWITCH specification, is tracked using the developed tools. The results of working with the tools show that the proposed approach works almost as effectively as the manual process.

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

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

Academic year 2003

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลืออย่างดียิ่งจาก ผู้ช่วยศาสตราจารย์ ดร. ธาราทิพย์ สุวรรณศาสตร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้เสียสละเวลาให้คำแนะนำเกี่ยวกับแนวทางวิจัย และข้อคิดเห็นที่มีประโยชน์มากมาย เพื่อประกอบการวิจัยนี้มาโดยตลอด ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี ผู้ช่วยศาสตราจารย์ ดร. วิวัฒน์ วัฒนาวุฒิ และ อาจารย์ ดร. ภัทรชัย ลลิตโรจน์วงศ์ คณะกรรมการสอบวิทยานิพนธ์ ที่ท่านได้กรุณาให้คำแนะนำและข้อชี้แนะในการทำวิจัย รวมถึงการตรวจสอบ และแก้ไขวิทยานิพนธ์ฉบับนี้

ขอขอบพระคุณคณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้ความรู้และคำแนะนำในการทำวิจัย

ขอขอบพระคุณอาจารย์ โคอิชิโร โอชิมิตซุสำหรับคำแนะนำในการทำวิจัยนี้

ขอขอบคุณพี่ๆ บุคลากรภาควิชาวิศวกรรมคอมพิวเตอร์จุฬาลงกรณ์มหาวิทยาลัยที่ให้ความสะดวกและกำลังใจในการทำงานวิจัย

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ สำหรับข้อแนะนำ คำเสนอแนะ รวมทั้งกำลังใจในการทำงานวิจัย

ขอขอบคุณใครบางคนที่เป็นกำลังใจ และอยู่เคียงข้างกันตลอดระยะเวลาที่ผ่านมา

สุดท้ายนี้ ข้าพเจ้าใคร่ขอกราบขอบพระคุณบิดา มารดา ทุกๆ คนในครอบครัว ที่สนับสนุนและให้กำลังใจแก่ข้าพเจ้าเสมอมาจนสำเร็จการศึกษา

สารบัญ

หน้า

บทที่ หน้า

สารบัญ (ต่อ)

บทที่ หน้า

สารบัญตาราง

หน้า

หน้า

สารบัญภาพ

หน้า

หน้า

หน้า

หน้า

สารบัญภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การพัฒนาซอฟต์แวร์ตามหลักวิศวกรรมซอฟต์แวร์ ประกอบด้วย 4 ขั้นตอน คือ การวิเคราะห์ (Analysis) การออกแบบ (Design) การพัฒนา (Implementation) และการทดสอบ (Testing) [1] ในขั้นตอนการวิเคราะห์ นักวิเคราะห์ระบบจะเก็บความต้องการ (Requirements) ของระบบที่ต้องการพัฒนามาสร้างเอกสารความต้องการซอฟต์แวร์ (Software Requirements Specification Document) เพื่อนำมาออกแบบซอฟต์แวร์นำไปใช้ในระบบ เอกสารที่ได้จากการออกแบบนี้เรียกว่าเอกสารการออกแบบ (Design Document) หรือเอกสารข้อกำหนดซอฟต์แวร์ (Software specification document) ซึ่งแบ่งเป็น 3 ประเภท [2] ได้แก่ ข้อกำหนดอรูปนัย (Informal specification) ข้อกำหนดกึ่งรูปนัย (Semi-formal specification) และข้อกำหนดรูปนัย (Formal specification) ข้อกำหนดอรูปนัยนำภาษาธรรมชาติ (Natural language) หรือภาษาที่ใช้ตามปรกติ มาอธิบายระบบ ในขณะที่ข้อกำหนดกึ่งรูปนัยใช้รูปภาพหรือแผนภาพเพื่ออธิบายความสัมพันธ์ของระบบต่างๆ เช่นแผนภาพยูเอ็มแอล (UML Diagram) [3] และข้อกำหนดรูปนัยนำสัญลักษณ์ทางคณิตศาสตร์มาอธิบายความต้องการของระบบ เมื่อได้เอกสารข้อกำหนดซอฟต์แวร์แล้วผู้พัฒนาโปรแกรม (Programmer) จะนำข้อกำหนดซอฟต์แวร์มาเป็นแนวทางในการพัฒนาโปรแกรม และให้ผู้ทดสอบ (Tester) ทดสอบโปรแกรมนั้นว่าพัฒนาถูกต้องตามที่ระบุไว้ในเอกสารความต้องการหรือไม่

วิธีการติดตามความก้าวหน้าของการพัฒนาซอฟต์แวร์ในปัจจุบัน ติดตามจากแผนการพัฒนาซอฟต์แวร์ (SPMP: Software Project Management Plan) โดยผู้จัดการโครงการ (Project manager) จะสอบถามความก้าวหน้าจากผู้พัฒนาโปรแกรม ทั้งนี้ผู้จัดการโครงการอาจนำข้อกำหนดซอฟต์แวร์มาสร้างรายการตรวจสอบ (Checklist) หรือนำรายการงาน (Task list) หรือรายการกิจกรรม (Activity list) ซึ่งกำหนดไว้ในแผนการพัฒนาซอฟต์แวร์ [4] มาสอบถามผู้พัฒนาโปรแกรมว่าได้พัฒนาโปรแกรม หรือว่าทำงานตามรายการเหล่านั้นแล้วหรือไม่ อย่างไรก็ดี หากผู้จัดการโครงการไม่ได้ทำงานร่วมกับผู้พัฒนาโปรแกรมในสถานที่เดียวกันหรือสภาพแวดล้อมเดียวกัน การสอบถามเพื่อตรวจสอบความก้าวหน้าจะทำได้ลำบาก เนื่องจากผู้จัดการโครงการจำเป็นต้องใช้โทรศัพท์ หรือใช้การประชุมทางไกล (Teleconference) เพื่อสอบถามความก้าวหน้าของการพัฒนากับผู้พัฒนาโปรแกรม สภาพแวดล้อมที่กลุ่มของผู้พัฒนาโปรแกรมไม่ได้อยู่ร่วมกันนี้เรียกว่าสภาพแวดล้อมการทำงานแบบกระจาย (Distributed Cooperative Work Environment)

ในปัจจุบันมีการเสนอวิธีการและเครื่องมือต่างๆ เพื่อช่วยในการติดตามความก้าวหน้าในสภาพแวดล้อมการทำงานแบบกระจาย อาทิเช่น Guang Yang และ Igor Tomak [5] นำเสนอ “Team Lab: A Collaborative Environment for Team work” โดยเตรียมสภาพแวดล้อมเพื่อใช้สำหรับสภาพแวดล้อมการทำงานแบบกระจาย โดยนำ MUM (Multi Universe MOO) ควบคุมขั้นตอนการทำงานโดยให้ผู้จัดการโครงการตรวจสอบความก้าวหน้าจากการส่งงานของสมาชิกผู้พัฒนาโปรแกรม Emile Morese และ Michelle Potts Steves [6] เสนอ “CollabLoggers: A Tool for Visualizing Group at Work” เพื่อตรวจสอบความก้าวหน้าในการพัฒนาโดยเตรียมสภาพแวดล้อมการทำงานให้สามารถตรวจสอบระยะเวลาที่ผู้พัฒนาโปรแกรมทำการพัฒนา และนำเอาระยะเวลาที่พัฒนานั้นมาแสดงความก้าวหน้าที่เกิดขึ้น Hiroyuki Murakoshi, Akira Shimazu และ Koichiro Ochimizu [7] เสนอ “Construction of Deliberation Structure in E-mail Communication” ซึ่งตรวจสอบความก้าวหน้าโดยตรวจสอบจดหมายอิเล็กทรอนิกส์ที่รับส่งระหว่างกลุ่มผู้พัฒนาโดยนำเอาลักษณะทางภาษามาแบ่งส่วนเนื้อหาของจดหมายอิเล็กทรอนิกส์แล้วคัดเอาชนิดของประโยคมาสร้างต้นไม้พิจารณา (Deliberation tree) เพื่อสรุปความก้าวหน้าในหัวเรื่องต่างๆ

การติดตามความก้าวหน้าด้วยวิธีการที่เสนอมาข้างต้นยังไม่สามารถตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมได้โดยอัตโนมัติ การตรวจสอบความก้าวหน้ายังคงต้องใช้ผลสรุปจากผู้พัฒนาโปรแกรมก่อนจึงจะสามารถสรุปความก้าวหน้าที่เกิดขึ้นได้ งานวิจัยนี้จึงมีแนวความคิดที่จะเสนอวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมโดยอัตโนมัติ โดยเปรียบเทียบข้อกำหนดรูปนัยที่ผู้ออกแบบได้ออกแบบไว้กับซอร์สโคดที่ผู้พัฒนาโปรแกรมได้พัฒนาเพื่อนำไปใช้ตรวจสอบความก้าวหน้าในการพัฒนาแบบกระจายได้

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อกำหนดกฎความสัมพันธ์ระหว่างข้อกำหนดคาเฟโอบีเจ และโปรแกรมภาษาจาวา

1.2.2 เพื่อสร้างวิธีการในการติดตามความก้าวหน้าของการพัฒนาโปรแกรมโดยอัตโนมัติ

1.2.3 เพื่อพัฒนาเครื่องมือซอฟต์แวร์ เพื่อตรวจสอบความก้าวหน้าโดยอัตโนมัติ

1.3 ขอบเขตของงานวิจัย

1.3.1 เครื่องมือนี้ประกอบด้วย 2 ส่วน ได้แก่ ส่วนสร้างโครงภาษาจาวา และ ส่วนตรวจสอบความก้าวหน้า

1.3.2 ผู้ใช้เครื่องมือต้องป้อนข้อมูลเพิ่มเติมเพื่อให้เครื่องมือสามารถสร้างโครงภาษาจาวาได้ถูกต้องตามวากยสัมพันธ์ (Syntactic) ของภาษาจาวา

1.3.3 ในส่วนตรวจสอบความก้าวหน้านั้นจะนำซอร์สโคดที่ได้จากการพัฒนาโครงภาษาจาวาเพิ่มเติมมาตรวจสอบ

1.3.4 ข้อกำหนดรูปนัยคาเฟโอบีเจที่ใช้ต้องได้รับการตรวจสอบว่าถูกต้องตามความต้องการและถูกต้องตามวากยสัมพันธ์แล้ว

1.3.5 วิธีการส่งซอร์สโคด เพื่อตรวจสอบความก้าวหน้า ไม่อยู่ในงานวิจัยนี้

1.3.6 ข้อกำหนดรูปนัยคาเฟโอบีเจที่ใช้ตรวจสอบ ไม่ครอบคลุมข้อกำหนดที่มีลักษณะต่อไปนี้ คือ

- ชนิดข้อมูลที่เป็นระเบียน (Record) เช่น

```
record date {
        day : Int
        month : Int
        year : Int
        }
```

- คำสำคัญและส่วนประกอบในภาษาคาเฟโอบีเจที่ใช้สำหรับการพิสูจน์ (Prove) ข้อกำหนดว่าถูกต้องมากน้อยเพียงใด เช่น red, =*=

1.3.7 ทดสอบการใช้งานโดยใช้ข้อกำหนดคาเฟโอบีเจที่สามารถสร้างโครงภาษาจาวาได้ และทดลองสร้างโปรแกรมตามข้อกำหนดตัวอย่างนั้นๆ และทดลองตรวจสอบความก้าวหน้าทีละขั้นตอน โดยใช้ข้อกำหนดอย่างน้อย 3 ข้อกำหนด

1.3.8 ในการตรวจสอบการกำหนดค่าทำการตรวจสอบความก้าวหน้าได้เฉพาะข้อความสั่ง (Statement) ที่แสดง การกำหนดค่าสำหรับตัวแปร การบวก การลบ การคูณ และการหารเท่านั้น

1.3.9 เครื่องมือที่พัฒนาขึ้นโดยใช้ภาษาจาวา J2SDK 1.4.0 บนระบบปฏิบัติการวินโดวส์

1.4 ขั้นตอนการดำเนินงานวิจัย

1.4.1 ศึกษาวิธีรูปนัยและวากยะสัมพันธ์ของภาษาคาเฟโอบีเจ

1.4.2 หาความสัมพันธ์ระหว่างส่วนประกาศมอดูล และส่วนประกาศโอเปอเรชันกับภาษาจาวา

1.4.3 ออกแบบขั้นตอนวิธีการแปลงข้อกำหนดคาเฟโอบีเจเป็นโครงภาษาจาวา

1.4.4 พัฒนาเครื่องมือซอฟต์แวร์เพื่อสร้างโครงภาษาจาวา

1.4.5 หาความสัมพันธ์ระหว่างส่วนประกาศสัจพจน์กับโปรแกรมภาษาจาวาที่พัฒนาแล้ว

1.4.6 ออกแบบขั้นตอนวิธีการเปรียบเทียบความสัมพันธ์ระหว่างส่วนประกาศสัจพจน์กับโปรแกรมภาษาจาวาที่ทำการพัฒนาแล้ว

1.4.7 พัฒนาเครื่องมือซอฟต์แวร์เพื่อตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

1.4.8 ตรวจสอบผลการวิจัย

1.4.9 สรุปผลการวิจัย และจัดทำเอกสาร

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 เครื่องมือสนับสนุนการเขียนโปรแกรมภาษาจาวาจากข้อกำหนดคาเฟโอบีเจ

1.5.2 เครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

1.5.3 ลดความพยายามที่ใช้ในการเขียนโปรแกรมจากข้อกำหนดรูปนัย

บทที่ 2
แนวคิดและทฤษฎีที่เกี่ยวข้อง

2.1 คาเฟโอบีเจ (CafeOBJ)

คาเฟโอบีเจ [8, 9] เป็นข้อกำหนดรูปนัยที่พัฒนามาจากภาษาโอบีเจ ซึ่งอธิบายข้อกำหนดของระบบในเชิงพีชคณิต (Algebraic Language) คาเฟโอบีเจใช้สถานะของวัตถุในการอธิบายระบบ โดยแสดงสถานะของระบบเมื่อมีการกระทำใดๆ เกิดขึ้นกับระบบ โครงสร้างของภาษาคาเฟโอบีเจ ประกอบไปด้วย 3 ส่วนได้แก่

2.1.1 ส่วนประกาศมอดูล (Module declaration)

ส่วนประกาศมอดูล ใช้สำหรับการประกาศชื่อมอดูล การประกาศพารามิเตอร์ (ถ้ามี) และการประกาศมอดูลอื่นเข้า (Import module) เพื่อใช้อธิบายคุณสมบัติของมอดูลปัจจุบัน โดยมอดูลในภาษาคาเฟโอบีเจมี 2 ประเภท ขึ้นอยู่กับหน้าที่และการทำงาน ประเภทแรกคือประเภทเดียว (A unique model) หมายถึง มอดูลที่ทำงานเป็นเอกเทศ (Unique) เพราะเมื่อนำข้อกำหนดรูปนัยไปสร้างเป็นระบบที่สามารถทำงานได้นั้น ผู้พัฒนาจะสามารถทำได้เพียงหนึ่งรูปแบบ หรือหนึ่งโมเดล (Model) เท่านั้น การประกาศมอดูลประเภทนี้จะใช้คำสำคัญ module! mod! module หรือ mod และประเภทที่สองคือ ประเภทหมวดหมู่ (A class of model) หมายถึงมอดูลที่มีลักษณะเป็นคลาสของโมเดล หรือมอดูลที่มีมากกว่า 1 โมเดล การประกาศมอดูลประเภทนี้จะใช้คำสำคัญ module* หรือ mod*

นอกจากนี้ ภาษาคาเฟโอบีเจยังมีการประกาศการนำเข้า (Import declaration) ซึ่งเป็นการอ้างอิงถึงมอดูลอื่นที่ได้ประกาศไว้แล้วมาทำงานร่วมกับมอดูลปัจจุบัน โดยภาษาคาเฟโอบีเจมีการนำเข้า 3 แบบได้แก่

- การนำเข้าแบบป้องกัน (Protecting mode) การนำเข้าแบบนี้จะไม่ยอมให้มีการเพิ่มหรือยุบส่วนย่อย (Element) ของมอดูลที่นำเข้ามาโดยเด็ดขาด โดยจะประกาศด้วยสัญลักษณ์ protecting หรือ pr

- การนำเข้าแบบขยาย (Extending mode) การนำเข้าแบบนี้สามารถเพิ่มส่วนย่อยของมอดูลที่นำเข้ามาได้ แต่ไม่ยอมให้ยุบส่วนย่อย โดยจะประกาศด้วยสัญลักษณ์ extending หรือ ex

- การนำเข้าแบบการใช้ (Using mode) การนำเข้าแบบนี้ สามารถเพิ่มหรือยุบส่วนย่อยของมอดูลที่นำเข้ามาได้ตามต้องการ โดยจะประกาศด้วยสัญลักษณ์ using หรือ us

2.1.2 ส่วนประกาศลายเซ็น (Signatures declaration)

ส่วนประกาศลายเซ็นใช้ในการประกาศชนิดข้อมูล (Sort) และสัญลักษณ์ทางฟังก์ชันที่กระทำกับชนิดข้อมูลที่ได้กำหนดไว้แล้ว โดยรูปแบบของแต่ละส่วนมีดังนี้

2.1.2.1 การประกาศชนิดข้อมูล (Sort declaration)

ชนิดข้อมูลสามารถแบ่งออกได้เป็น 2 ประเภทคือ

- ชนิดข้อมูลสังเกตค่าได้ (Visible sort) เป็นชนิดข้อมูลสำหรับให้บุคคลภายนอกระบบที่ออกแบบไว้สังเกตค่าของวัตถุนั้นๆ โดยมีรูปแบบการกำหนดดังนี้

  [sort_name1 sort_name2 …]

  โดยที่

  [] ใช้สำหรับการประกาศชนิดข้อมูลสังเกตค่าได้

  sort_name1 sort_name2 … คือ รายการชื่อชนิดข้อมูลที่ประกาศ

- ชนิดข้อมูลแฝง (Hidden sort) คือชนิดข้อมูลที่ใช้เก็บสถานะของระบบ โดยปกติไม่สามารถสังเกตค่าของชนิดข้อมูลนี้ได้โดยตรง แต่สามารถพิจารณาได้จากค่าของชนิดข้อมูลสังเกตค่าได้ โดยมีรูปแบบการกำหนดดังนี้

  [hsort_name1 hsort_name2 …]

  โดยที่

  [] ใช้สำหรับการประกาศชนิดข้อมูลแฝง

  hsort_name1 hsort_name2 … คือรายการชนิดข้อมูลแฝงที่ประกาศ

ในกรณีที่มีการกำหนดลักษณะชนิดข้อมูลย่อย (Sub sort) กล่าวคือ ชนิดข้อมูลหนึ่งเป็นเซตย่อยของอีกชนิดข้อมูลหนึ่ง จะมีรูปแบบดังการกำหนด

[sort_name1 < sort_name2]

[hsort_name1 < hsort_name2]

โดยที่

sort_name1 < sort_name2 คือ sort_name1 เป็นชนิดข้อมูลย่อยของ sort_name2 สำหรับชนิดข้อมูลสังเกตค่า

hsort_name1 < hsort_name2 คือ hsort_name1 เป็น ชนิดข้อมูลย่อยของ hsort_name2 สำหรับชนิดข้อมูลแฝง

2.1.2.2 การประกาศโอเปอเรชัน (Operation declaration)

การประกาศโอเปอเรชันเป็นการกำหนดว่าภายในวัตถุมีโอเปอเรชันใดบ้าง แต่ละโอเปอเรชันมีชนิดข้อมูลใดเป็นส่วนนำเข้า (Input) และชนิดข้อมูลใดเป็นผลลัพธ์ (Output) โดยที่ส่วนนำเข้าเป็นรายการของชนิดข้อมูล เรียกว่าอาริที (Arity) และส่วนผลลัพธ์เป็นชนิดของข้อมูลเพียงชนิดเดียวเรียกว่าโคอาริที (Coarity) โดยการประกาศโอเปอเรชันมีรูปแบบดังนี้

op op_name : sort_name1 sort_name2 … -> sort

หรือ

bop op_name : sort_name1 sort_name2 … -> sort

โดยที่

op คือ คำสำคัญที่ใช้กำหนดโอเปอเรชัน

bop คือ คำสำคัญที่ใช้โอเปอเรชันเชิงพฤติกรรม (Behavioral operation)

op_name คือ ชื่อโอเปอเรชัน

sort_name1 sort_name2 ... คือ รายการชนิดข้อมูลที่เป็นอาริที

sort คือ ชนิดข้อมูลใดๆ ที่เป็นโคอาริที

2.1.3 ส่วนประกาศสัจพจน์ (Axiom declaration)

ส่วนประกาศสัจพจน์เป็นส่วนที่ใช้กำหนดพฤติกรรมของชนิดข้อมูลหรือวัตถุ โดยใช้สมการทางคณิตศาสตร์มาเป็นตัวกำหนด ซึ่งประกอบด้วยการกำหนดตัวแปรและการกำหนดสมการดังนี้

2.1.3.1 ส่วนประกาศตัวแปร (Variable declaration)

ส่วนประกาศตัวแปรใช้สำหรับการกำหนดตัวแปรทั้งหมดที่อ้างอิงในการเขียนสมการเพื่ออธิบายพฤติกรรมของชนิดข้อมูลหรือวัตถุ โดยมีรูปแบบดังนี้

var varname : sort_name

หรือ

vars var_name1, var_name2, … : sort_name

โดยที่

var vars คือ คำสำคัญที่ใช้กำหนดตัวแปร

var_name คือ ชื่อตัวแปร

var_name1, var_name2, … คือ รายการชื่อตัวแปรที่เป็นชนิดข้อมูลเดียวกัน

sort_name คือ ชนิดข้อมูล

หรือ อาจประกาศตัวแปรแบบทันทีเมื่อใช้ (On-the-fly declaration)

var_name : sort_name ภายในส่วนประกาศสมการ โดยไม่ต้องประกาศตัวแปรก่อน

โดยที่

var_name คือ ชื่อตัวแปร

sort_name คือ ชนิดข้อมูล

2.1.3.2 ส่วนประกาศสมการ (Equation declaration)

ส่วนประกาศสมการเพื่อใช้กำหนดพฤติกรรมของวัตถุ แบ่งได้เป็น 2 ประเภทคือ

- การประกาศสมการแบบไม่มีเงื่อนไข (Unconditional equation declaration) มีรูปแบบดังนี้

eq term = term

หรือ

beq term = term

โดยที่

eq คือ คำสำคัญที่ใช้ในการประกาศสมการแบบไม่มีเงื่อนไข

beq คือ คำสำคัญที่ใช้ในการประกาศสมการเชิงพฤติกรรมแบบไม่มีเงื่อนไข

term คือ พจน์ทางคณิตศาสตร์

- การประกาศสมการแบบมีเงื่อนไข (Conditional equation declaration) มีรูปแบบดังนี้

ceq term = term if boolean_term

หรือ

bceq term = term if boolean_term

โดยที่

ceq คือ คำสำคัญที่ใช้ประกาศสมการ แบบมีเงื่อนไข

bceq คือ คำสำคัญที่ใช้ประกาศสมการเชิงพฤติกรรม แบบมีเงื่อนไข

term คือ พจน์ทางคณิตศาสตร์

if คือ คำสำคัญแสดงส่วนกำหนดเงื่อนไข

boolean_term คือ พจน์เงื่อนไข หรือ term ที่ให้ผลลัพธ์เป็นจริง หรือ เท็จ

ตัวอย่างของข้อกำหนดคาเฟโอบีเจ แสดงในภาคผนวก ก

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 “Team Lab: A Collaborative Environment for Teamwork” โดย Guang Yang และ Igor Tomak [5]

งานวิจัยนี้เสนอการเตรียมเครื่องมือสำหรับการทำงานร่วมกันภายในสภาพแวดล้อมการพัฒนาแบบกระจาย โดยใช้สถาปัตยกรรมแบบระบบรับ-ให้บริการ (Client-Server system) เพื่อช่วยให้ทีมงานสามารถพัฒนาโปรแกรมไปพร้อมๆ กัน (Concurrent programming activity) และติดต่อกับส่วนเก็บโปรแกรมที่กำลังพัฒนา (Code repository) ได้ ทั้งนี้ Yang และ Tomak ได้พัฒนา MUM – a Multi-Universe MOO เพื่อเป็นส่วนจัดการในการติดต่อกันภายในทีมงาน โดยแบ่งส่วนของการทำงานออกเป็นสี่ส่วนหลักคือ Team lab agent, Team lab and registry, ส่วนเก็บโปรแกรมที่กำลังพัฒนา และเครื่องมือฝั่งผู้รับ (Client-side tools) เพื่อให้การทำงานร่วมกันเป็นไปอย่างมีประสิทธิภาพ

งานวิจัยนี้เตรียมเพียงเครื่องมือสำหรับการติดตามตรวจสอบความก้าวหน้าที่เกิดขึ้น ทั้งนี้ผู้จัดการโครงการเป็นผู้ตรวจสอบความก้าวหน้าทั้งหมดด้วยตนเอง โดยการตรวจสอบซอร์สโคดในส่วนเก็บโปรแกรมที่กำลังพัฒนาโดยตรง

2.2.2 “CollabLogger: A Tool for Visualizing Groups at Work” โดย Emile Morse และ Michelle Potts Steves [6]

งานวิจัยนี้เสนอเครื่องมือสำหรับการตรวจสอบการทำงานของทีมพัฒนา เครื่องมือที่พัฒนาขึ้นจะทำงานร่วมกับโปรแกรม Teamwave Workspace ซึ่งเป็นโปรแกรมสำหรับเตรียมสภาพแวดล้อมการทำงานแบบกระจาย CollabLogger จะตรวจสอบว่าผู้พัฒนาแต่ละคนใช้งานเครื่องมือสำหรับการพัฒนาใดบ้างและใช้งานเครื่องมือนั้นเป็นเวลาเท่าใด เพื่อนำมาสร้างเป็นกราฟแสดงระยะเวลาของผู้พัฒนาแต่ละคนว่าได้ทำงานมากน้อยเพียงใด

งานวิจัยนี้เป็นรูปแบบหนึ่งในการติดตามความก้าวหน้าโดยตรวจสอบจากเวลาที่ใช้เครื่องมือต่างๆ สำหรับการพัฒนา แล้วนำมาสร้างเป็นกราฟเพื่อให้ผู้จัดการโครงการสามารถเห็นภาพการทำงานของสมาชิกในทีมพัฒนาได้สะดวกยิ่งขึ้น

2.2.3 “Construction of Deliberation Structure in E-mail Communication” โดย Hiroyuki Murakoshi, Akira Shimazu และ Koichiro Ochimizu [7]

งานวิจัยนี้เสนอวิธีการตรวจสอบความก้าวหน้าในการทำงานโดยตรวจสอบจากจดหมายอิเล็กทรอนิกส์ที่รับส่งกันภายในกลุ่มของผู้พัฒนา ทั้งนี้ชนิดของข้อความภายในจดหมายอิเล็กทรอนิกส์ได้รับการจัดหมวดหมู่ โดยอาศัยลักษณะทางภาษา (Linguistic clue) ของข้อความนั้นๆ เช่นข้อความนั้นเป็นประโยคเริ่มต้นของเนื้อหา หรือเป็นส่วนตอบคำถามของจดหมายฉบับที่

แล้วเป็นต้น เมื่อทราบหมวดหมู่ของเนื้อหาภายในจดหมายอิเล็กทรอนิกส์แต่ละฉบับ จะรวบรวมเนื้อหาของจดหมายเหล่านั้นที่มีเนื้อหาเดียวกันมาสร้างเป็นต้นไม้พิจารณา เพื่อลำดับเนื้อหาภายในจดหมายอิเล็กทรอนิกส์แต่ละฉบับให้สะดวกในการติดตามความก้าวหน้าในการพัฒนา

งานวิจัยนี้นำเสนอวิธีการตรวจสอบความก้าวหน้าโดยอัตโนมัติจากจดหมายอิเล็กทรอนิกส์ ทั้งนี้จะสามารถตรวจสอบได้เมื่อจดหมายนั้นเขียนด้วยภาษาที่ได้สร้างลักษณะทางภาษานั้นๆ แล้วและผู้จัดการโครงการจะตรวจสอบความก้าวหน้าได้จากต้นไม้พิจารณาที่สร้างขึ้น

2.2.4 “A Vector-Based Approach to Software Size Measurement and Effort Estimation” โดย T.E Hasting [10]

งานวิจัยนี้เสนอการนำข้อกำหนดรูปนัยมาประมาณความพยายาม (Effort estimation) ที่ต้องใช้ในการพัฒนาซอฟต์แวร์ในโครงการพัฒนาซอฟต์แวร์ซึ่งเวลาที่ประเมินได้นี้จะนำไปใช้ในการวางแผนงานพัฒนาต่อไป งานวิจัยนี้เลือกเอาข้อกำหนดรูปนัยเชิงพีชคณิต (Algebraic Specification Language) มาใช้ในการประมาณความพยายาม โดยอาศัยหลักการว่าระบบใดที่มีความซับซ้อน (Complexity) มาก จะต้องใช้ความพยายามในการพัฒนามากตามไปด้วย ความซับซ้อนของระบบพิจารณาจากอัตราส่วนระหว่างจำนวนโอเปอเรชันที่ปรากฏอยู่ในส่วนความหมาย (Semantic) ซึ่งระบุสถานะที่เปลี่ยนไปของระบบเมื่อมีการเรียกโอเปอเรชันใดๆ และจำนวนโอเปอเรชันที่ปรากฏในส่วนวากยสัมพันธ์ซึ่งเป็นส่วนประกาศโอเปอเรชัน ของระบบ

งานวิจัยนี้เป็นตัวอย่างการนำข้อกำหนดรูปนัยมาใช้เป็นแนวทางในการตรวจสอบความก้าวหน้า โดยนำมาประมาณความพยายามในการพัฒนา เพื่อประเมินเวลาที่ผู้จัดการโครงการจะทำการสอบถามความก้าวหน้าจากผู้พัฒนาโปรแกรมต่อไป

2.2.5 “GENOA- A Customizable, Front-End-Retargetable Source Code Analysis Framework” โดย Premkumar T. Devanbu [11]

งานวิจัยนี้เสนอวิธีการตรวจสอบซอร์สโคดภาษา C++ โดยในการตรวจสอบจะแปลงซอร์สโคดให้อยู่ในรูปของต้นไม้โครงสร้างภาษา (Abstract syntax tree) แล้วท่อง (Traverse) ไปตามต้นไม้ เพื่อให้ได้ข้อมูลตามที่ต้องการ รูปแบบของข้อมูลที่ต้องการนั้นสามารถเปลี่ยนแปลงได้ว่าต้องการข้อมูลแบบใด โดยยังใช้รูปแบบของต้นไม้โครงภาษาเดิม ทั้งนี้รูปแบบการสอบถาม (Query) ข้อมูลกำหนดโดยภาษา GEN++

งานวิจัยนี้เป็นแนวทางในการนำซอร์สโคดมาวิเคราะห์หาข้อมูลโดยอัตโนมัติ เพื่อให้ได้ข้อมูลที่ต้องการเพื่อนำไปพิจารณาเปรียบเทียบในขั้นต่อไป

2.2.6 “Evolving Object Oriented Design to Improve Code Traceability” โดย G. Antoniol, A. Potrich, P. Tonella และ R. Fiutem [12]

งานวิจัยนี้เสนอการเปรียบเทียบระหว่างซอร์สโคด (Source Code) กับส่วนออกแบบที่ออกแบบโดยแผนภาพโอเอ็มที (OMT Diagram) โดยตรวจสอบเมท็อด (Method) และ ลักษณะประจำ (Attribute) ภายในซอร์สโคดว่าสอดคล้องกับการออกแบบภายในแผนภาพโอเอ็มที

งานวิจัยนี้นำเสนอวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม โดยตรวจสอบเฉพาะในส่วนวากยสัมพันธ์ของซอร์สโคดเปรียบเทียบกับส่วนออกแบบเท่านั้น

2.2.7 การตัดส่วนโปรแกรม (Program slicing)

การตัดส่วนโปรแกรมเสนอโดย Weiser [13] เป็นวิธีการเลือกเฉพาะส่วนของโปรแกรมที่สนใจ เพื่อใช้ในการบำรุงรักษา (Maintenance) และวิวัฒนาการ (Evolution) ตัวอย่างเช่นการปรับรูปแบบใหม่ (Re-engineering) การทดสอบ การแบ่งส่วนย่อย (Decomposition) การรวมส่วนและการดัดแปลง (Integration and Modification) การทำโปรแกรมแปลกลับ (Decompilation) การทำความเข้าใจโปรแกรม (Program comprehension) และ การแก้จุดบกพร่อง (Debugging) [14] การตัดส่วนโปรแกรมนั้นสามารถทำได้ 2 วิธี คือ การตัดส่วนแบบสถิต (Static slicing) และการตัดส่วนแบบพลวัต (Dynamic slicing) [15] Weiser [13] ได้ให้นิยามการตัดส่วนแบบสถิตว่าคือ เซตย่อย (subset) ของซอร์สโคดที่สามารถบอกพฤติกรรมของโปรแกรมนั้นๆ ในขณะที่ Korel และ Larski [16] เสนอการตัดส่วนแบบพลวัต โดยเลือกตัดส่วนของโปรแกรมเฉพาะข้อความสั่งซึ่งถูกเรียกใช้ในขณะที่เปลี่ยนแปลงค่าของตัวแปรที่กำลังสนใจ การตัดส่วนโปรแกรมแบบพลวัตจะตัดส่วนเมื่อโปรแกรมกำลังทำงานอยู่ วิธีการตัดส่วนพลวัตนี้ จะช่วยให้ขนาดของส่วนของโปรแกรมที่ได้มีขนาดเล็กลง และทำให้การทำความเข้าใจต่อโปรแกรมรวมถึงหาข้อผิดพลาดของโปรแกรมทำได้ง่ายขึ้น

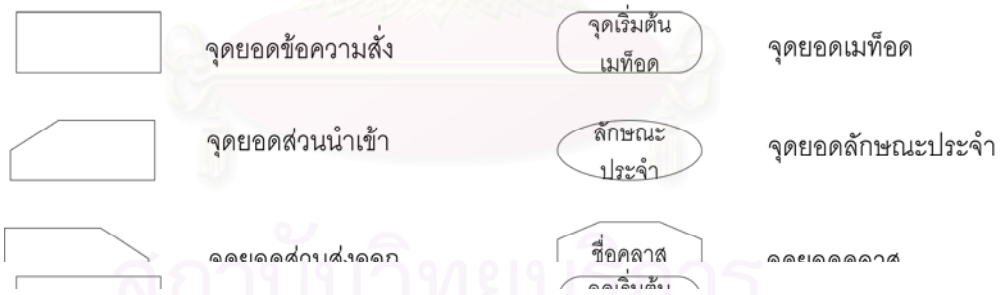
การตัดส่วนแบบสถิตทำโดยนำซอร์สโคด มาหาความสัมพันธ์ระหว่างข้อความสั่งและตัวแปรที่กำลังพิจารณา โดยที่ยังไม่ต้องสั่งให้โปรแกรมทำงาน ในขณะที่การตัดส่วนพลวัตจำเป็นต้องเรียกโปรแกรมให้ทำงานก่อนแล้วสังเกตว่าโปรแกรมใช้ข้อความสั่งใดบ้างที่เกี่ยวข้องกับตัวแปรที่กำลังพิจารณาเพื่อให้ได้ส่วนของโปรแกรมออกมา

เนื่องจากการตัดส่วนพลวัตจะตัดส่วนได้ก็ต่อเมื่อมีโปรแกรมที่สามารถทำงานได้โดยสมบูรณ์แล้ว แต่การตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมจะต้องทำตลอดช่วงเวลาที่พัฒนา ถึงแม้ว่าโปรแกรมนั้นจะยังไม่สามารถทำงานได้โดยสมบูรณ์ ดังนั้นการตัดส่วนพลวัตจึงไม่สามารถนำมาใช้ในงานวิจัยนี้ได้ ในงานวิจัยนี้จึงเลือกวิธีการตัดส่วนสถิตมาใช้

2.2.8 การตัดส่วนโปรแกรมเชิงวัตถุ (Slicing object-oriented programs)

ปัจจุบันโปรแกรมที่พัฒนาด้วยวิธีการเชิงวัตถุ (Object-oriented paradigm) ได้รับความนิยมเนื่องจากคุณสมบัติต่างๆ อาทิเช่นการแบ่งวัตถุออกเป็นมอดูล (Modularity) การนำกลับมาใช้ใหม่ (Reusability) และการพัฒนาเพิ่มเติม (Extendibility) โดยวิธีการเชิงวัตถุนำการกำหนดสาระสำคัญของวัตถุ (Object abstraction) การห่อหุ้มข้อมูล (Encapsulation) การสืบทอดสมบัติ (Inheritance) และการมีหลายรูป (Polymorphism) มาแสดงชนิดของข้อมูล ทำให้ความสัมพันธ์ระหว่างส่วนต่างๆ ภายในโปรแกรมมีมากกว่าโปรแกรมเชิงกระบวนงาน (Conventional procedure-oriented language) ดังนั้น J.L. Chen และคณะ จึงได้เสนอกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ (Object Oriented Dependency Graph หรือ OODG) [17] เพื่อแสดงความสัมพันธ์ระหว่างเอนทิตี (Entities) แต่ละเอนทิตีในโปรแกรม โดยแต่ละเอนทิตีมีความสัมพันธ์อ้างอิง (Dependencies) ระหว่างกัน

โปรแกรมเชิงวัตถุประกอบด้วยเอนทิตีหลายชนิดประกอบกันไม่ว่าจะเป็น ข้อความ (Messages) เมท็อด ลักษณะประจำ และคลาส (Class) โดยสามารถแบ่งเอ็นทิตีเหล่านี้เพื่อสร้างเป็นจุดยอด (Vertex) ภายในกราฟได้แก่ จุดยอดข้อความสั่ง จุดยอดส่วนนำเข้า (Formal-in) จุดยอดส่วนส่งออก (Formal-out) จุดยอดเมท็อด จุดยอดลักษณะประจำ และจุดยอดคลาส สัญลักษณ์ของจุดยอดแต่ละชนิดในกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ แสดงในรูปที่ 2.1

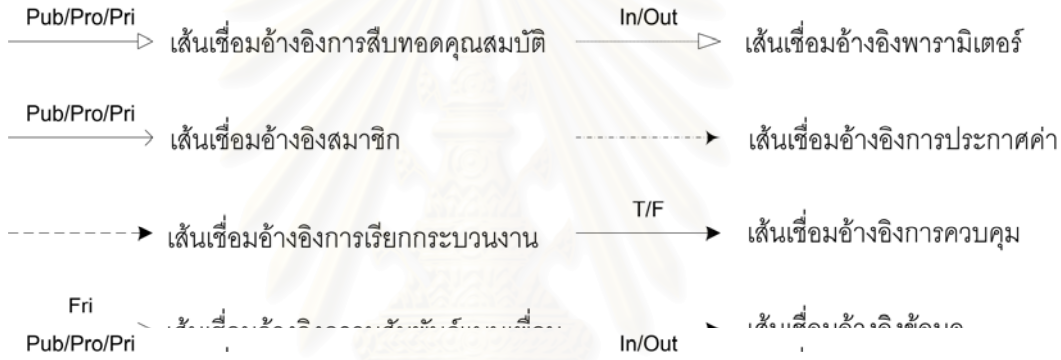


รูปที่ 2.1 สัญลักษณ์ของจุดยอดในกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ

- จุดยอดข้อความสั่ง แสดงการส่งผ่านข้อความ (Message passing) การเรียกฟังก์ชัน (Function call) และข้อความสั่งควบคุม (Control statement) ซึ่งรวมถึงข้อความเงื่อนไขและการวนลูป ภายในเมท็อด หรือฟังก์ชันในโปรแกรม
- จุดยอดส่วนนำเข้า แสดงสถานะของวัตถุที่นำเข้าสู่ฟังก์ชันหรือเมท็อด ณ จุดเริ่มต้นของเมท็อด หรือฟังก์ชัน
- จุดยอดส่วนส่งออก แสดงสถานะของวัตถุที่ส่งออกจากฟังก์ชัน หรือเมท็อด ณ จุดสิ้นสุดของเมท็อด หรือฟังก์ชัน

- จุดยอดเมท็อด แสดงจุดเริ่มต้นของเมท็อด หรือฟังก์ชัน
- จุดยอดลักษณะประจำ แสดงลักษณะประจำ
- จุดยอดคลาส แสดงส่วนหัวของคลาส (Class header)

เส้นเชื่อมจากแต่ละจุดยอดไปยังจุดยอดอื่นแสดงความสัมพันธ์อ้างอิงกันระหว่างเอนทิตีหนึ่งไปยังอีกเอนทิตีที่อ้างอิงกับเอนทิตีนั้น โดยชนิดของความสัมพันธ์ได้แก่ การสืบทอดคุณสมบัติ การเป็นสมาชิก (Membership) ความสัมพันธ์แบบเพื่อน (Friend-relationship) การเรียกกระบวนงาน (Procedure-call) ความสัมพันธ์จากพารามิเตอร์ ความสัมพันธ์จากการควบคุม (Control dependencies) และความสัมพันธ์จากข้อมูล เส้นเชื่อมแสดงความสัมพันธ์อ้างอิงแสดงในรูปที่ 2.2 โดยที่



รูปที่ 2.2 สัญลักษณ์แสดงเส้นเชื่อมในกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ

- เส้นเชื่อมอ้างอิงการสืบทอดคุณสมบัติ (Inheritance dependency edge) เชื่อมจุดยอดคลาสจากคลาสแม่ (Super class) ไปยังคลาสลูก (Sub class) โดยเส้นเชื่อมจะระบุข้อความ ‘Pub’, ‘Pro’ หรือ ‘Pri’ แสดงการสืบทอดคุณสมบัติแบบ public, protected หรือ private
- เส้นเชื่อมอ้างอิงสมาชิก (Membership dependency edge) เชื่อมจากจุดยอดคลาส ไปยังจุดยอดลักษณะประจำหรือเมท็อด โดยเส้นเชื่อมจะระบุข้อความ ‘Pub’, ‘Pro’ หรือ ‘Pri’ เพื่อแสดงว่าลักษณะประจำหรือเมท็อดนั้นเป็นสมาชิกแบบ public, protected หรือ private
- เส้นเชื่อมอ้างอิงความสัมพันธ์แบบเพื่อน (Friend-relationship dependency edge) เชื่อมจากจุดยอดคลาสหนึ่งไปยังอีกจุดยอดคลาสหนึ่งหรือจุดยอดเมท็อดของคลาสอื่น ซึ่งถูกกำหนดให้มีความสัมพันธ์แบบเพื่อน (Friend relationship) กับคลาสต้นทาง

- เส้นเชื่อมอ้างอิงการเรียกกระบวนงาน (Procedure-call dependency edge) เชื่อมจากจุดยอดข้อความไปยังจุดยอดเมท็อดที่ทำงานเนื่องจากการเรียกจากข้อความต้นทาง
- เส้นเชื่อมอ้างอิงพารามิเตอร์ (Parameter dependency edge) แบ่งเป็นสองประเภทตามที่ระบุไว้บนเส้นเชื่อมคือเข้า (In) และออก (Out) เส้นเชื่อมอ้างอิงพารามิเตอร์แบบเข้าเชื่อมจากจุดยอดข้อความสั่งไปยังจุดยอดส่วนนำเข้า ในขณะที่เส้นเชื่อมอ้างอิงพารามิเตอร์แบบออกเชื่อมจากจุดยอดส่วนส่งออกไปยังจุดยอดข้อความสั่งที่เรียกเมท็อดนั้น
- เส้นเชื่อมอ้างอิงการประกาศค่า (Declaration dependency edge) เชื่อมจากจุดยอดคลาสไปยังจุดยอดอื่นเพื่อประกาศว่าจุดยอดนั้นเป็นสมาชิกของคลาสนั้น
- เส้นเชื่อมอ้างอิงการควบคุม (Control dependency edge) แสดงทิศทางการไหลของโปรแกรม (Control flow) ภายในการส่งข้อความหรือการเรียกใช้เมท็อด โดยเชื่อมจากจุดยอดหนึ่งไปยังอีกจุดยอดหนึ่งโดยจุดยอดปลายทางจะทำงานเมื่อการตรวจสอบเงื่อนไข (Condition) ที่จุดยอดต้นทางมีผลตามที่ระบุไว้บนเส้นเชื่อมคือเป็นจริงหรือเท็จ
- เส้นเชื่อมอ้างอิงข้อมูล (Data dependency edge) แสดงการอ้างอิงข้อมูลระหว่างข้อความ การเรียกเมท็อด และการควบคุมโดยเชื่อมจากจุดยอดหนึ่งไปยังอีกจุดยอดหนึ่งซึ่งอ้างอิงข้อมูลจากจุดยอดต้นทาง

ภาคผนวก ข แสดงตัวอย่างของโปรแกรมภาษาจาวา โดยกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของโปรแกรมซึ่งแสดงในภาคผนวก ข แสดงใน ภาคผนวก ค

ทั้งนี้ Chen และคณะยังได้เสนอวิธีการตัดส่วนโปรแกรมสำหรับโปรแกรมที่พัฒนาเชิงวัตถุ [18] โดยนำกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุมาแสดงความสัมพันธ์ของส่วนของโปรแกรมที่กำลังพิจารณา ในการตัดส่วนนั้นอ้างอิงจากความสัมพันธ์ระหว่างแต่ละวัตถุ โดยสามารถเลือกได้ว่าต้องการตัดส่วนโปรแกรมแบบอิงสถานะ (State slicing) หรืออิงพฤติกรรม (Behavior slicing)

การตัดส่วนอิงสถานะใช้ตรวจสอบการเปลี่ยนแปลงที่เกิดขึ้นกับตัวแปรตัวใดตัวหนึ่งที่กำลังพิจารณา ผลลัพธ์ที่ได้จากการตัดส่วนของโปรแกรมจะเป็นชุดข้อความสั่ง ที่มีผลต่อการเปลี่ยนแปลงค่าของตัวแปรดังกล่าว

ในขณะที่การตัดส่วนอิงพฤติกรรมใช้ตรวจสอบความสัมพันธ์ระหว่างเมท็อดหรือคลาสที่กำลังพิจารณาว่ามีความสัมพันธ์กับคลาสหรือเมท็อดใดบ้าง ผลลัพธ์ที่ได้จะเป็นส่วนของโปรแกรม

ที่เป็นคลาส หรือเมท็อดที่มีความสัมพันธ์กับคลาสหรือเมท็อดที่กำลังพิจารณา โดยอาจจะเป็นคลาส หรือเมท็อดที่ถูกเรียกโดยคลาส หรือเมท็อดที่กำลังพิจารณาก็ได้

บทที่ 3
การออกแบบการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

3.1 แนวทางในการตรวจสอบความก้าวหน้าในการพัฒนาโปรแกรม

งานวิจัยนี้นำเสนอ วิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม โดยตรวจสอบจาก ซอร์สโคดที่พัฒนามาจากข้อกำหนดซอฟต์แวร์ที่ออกแบบเอาไว้ ทั้งนี้โปรแกรมที่นำมาตรวจสอบนั้นต้องได้รับการออกแบบอยู่ในรูปแบบข้อกำหนดรูปนัยคาเฟโอบีเจ และพัฒนาโดยใช้ภาษาจาวา ทั้งนี้เนื่องจากข้อกำหนดรูปนัยคาเฟโอบีเจมีคุณลักษณะทางด้านการออกแบบเชิงวัตถุที่ชัดเจน อีกทั้งยังมีเครื่องมือช่วยในการสร้างข้อกำหนด [18] อีกด้วย และภาษาจาวาถูกเลือกใช้เป็นภาษาที่นำมาตรวจสอบ เนื่องจากภาษาจาวาเป็นภาษาเชิงวัตถุ ทำให้สะดวกในการเปรียบเทียบกับข้อกำหนดที่สามารถออกแบบเชิงวัตถุได้

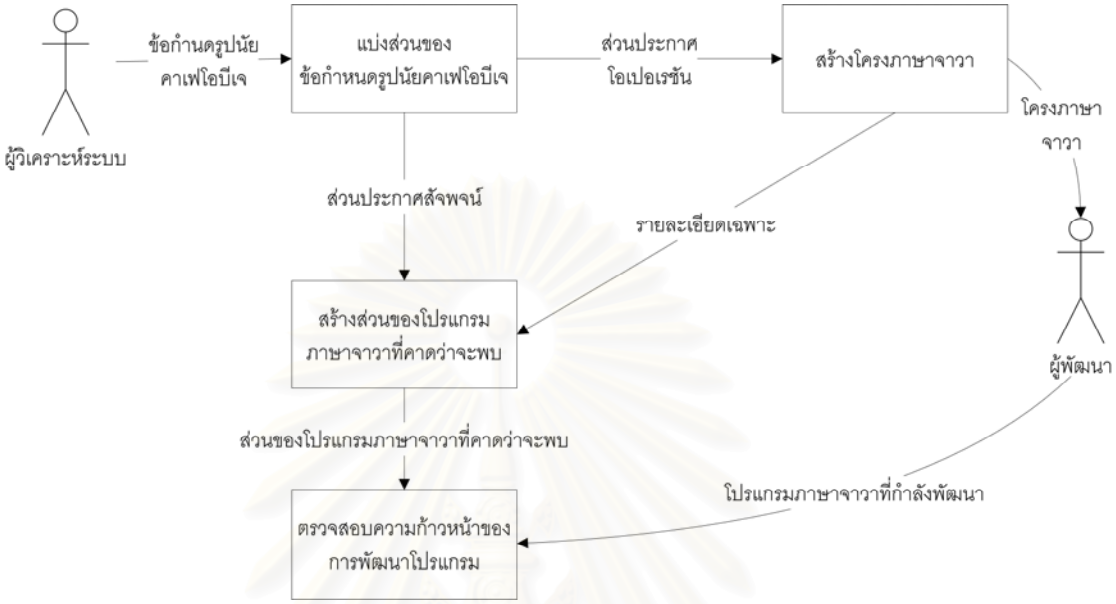
ข้อกำหนดรูปนัยคาเฟโอบีเจสามารถแบ่งออกได้เป็นสองส่วนหลัก เช่นเดียวกับภาษาโปรแกรมระดับสูง (High level programming language) ทั่วไป นั่นคือ ส่วนวากยสัมพันธ์ และส่วนความหมาย ส่วนวากยสัมพันธ์ ประกอบด้วยส่วนประกาศมอดูล และส่วนประกาศลายเซ็น ในขณะที่ส่วนความหมายประกอบด้วยส่วนกำหนดสัจพจน์

การพัฒนาโปรแกรมด้วยภาษาเชิงวัตถุ เช่นภาษาจาวา ตามข้อกำหนดรูปนัยคาเฟโอบีเจมีข้อสังเกตดังนี้

- คลาสสำหรับภาษาจาวาสร้างจากมอดูลแต่ละมอดูลในข้อกำหนดคาเฟโอบีเจ
- ตัวแปรเก็บสถานะของคลาสของจาวาถูกสร้างขึ้นเพื่อเก็บสถานะของคลาส เช่นเดียวกับชนิดข้อมูลแฝงซึ่งเก็บสถานะของมอดูลในข้อกำหนดคาเฟโอบีเจ
- เมท็อดและค่าคงที่ต่างๆ ในคลาสของภาษาจาวาพิจารณาจากโอเปอเรชันที่ประกาศไว้ในส่วนประกาศโอเปอเรชันของข้อกำหนดคาเฟโอบีเจ
- ข้อความสั่งในแต่ละเมท็อดจะเปลี่ยนสถานะของตัวแปรเก็บสถานะตามส่วนประกาศสัจพจน์ที่อ้างอิงถึงโอเปอเรชันที่คู่กับเมท็อดนั้น

โดยปกติ ผู้พัฒนาโปรแกรมจะต้องพัฒนาโปรแกรมตามข้อกำหนดรูปนัยที่ออกแบบไว้ โดยการเขียนโปรแกรมจะเริ่มจากการสร้างส่วนวากยสัมพันธ์ ที่สอดคล้องกับข้อกำหนดรูปนัย แล้วจึงเพิ่มส่วนความหมายลงไปยังส่วนของโปรแกรมที่กำลังพัฒนาขึ้นแล้ว ดังนั้นวิธีตรวจสอบความก้าวหน้าที่นำเสนอในงานวิจัยนี้จึงประกอบไปด้วยสามขั้นตอนได้แก่ การสร้างส่วน

วากยสัมพันธ์ การสร้างส่วนความหมาย และ การตรวจสอบความก้าวหน้า ภาพรวมขั้นตอนการทำงานแสดงในรูปที่ 3.1



รูปที่ 3.1 ภาพรวมของวิธีการตรวจสอบความก้าวหน้าของการพัฒนาซอฟต์แวร์ที่นำเสนอ

ขั้นตอนการทำงานเริ่มจากผู้ออกแบบออกแบบข้อกำหนดรูปนัยของโปรแกรมที่ต้องการแล้วนำส่วนวากยสัมพันธ์ของข้อกำหนดซอฟต์แวร์มาสร้างโครงภาษาจาวา (Java template Code) แล้วส่งมอบโครงภาษาจาวาให้กับผู้พัฒนาโปรแกรมเพื่อพัฒนาโปรแกรมต่อไป และส่วนความหมายของข้อกำหนดซอฟต์แวร์นำมาสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ (Expected source code) จากข้อกำหนด แล้วนำโปรแกรมภาษาจาวาที่พัฒนาแล้วมาเปรียบเทียบกับส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบมาเปรียบเทียบเพื่อตรวจสอบหาความก้าวหน้าของการพัฒนาโปรแกรมที่เกิดขึ้น

3.2 การสร้างโครงภาษาจาวา

การออกแบบอยู่ในระดับของการกำหนดสาระสำคัญ (Level of abstraction) ที่สูงกว่าการเขียนโปรแกรม นั่นคือการออกแบบจะกำหนดรายละเอียดเฉพาะของโปรแกรมที่ต้องการไว้น้อยกว่าการเขียนโปรแกรม ดังนั้นการเปรียบเทียบระหว่างข้อกำหนดซอฟต์แวร์ที่ได้จากการออกแบบ และโปรแกรมที่ได้จากการเขียนโปรแกรมจึงจำเป็นต้องเพิ่มรายละเอียดเฉพาะ ทั้งนี้ผู้พัฒนาต้องกำหนดรายละเอียดเฉพาะให้กับข้อกำหนดซอฟต์แวร์เอง อย่างไรก็ดีงานวิจัยหลายงานที่เสนอการสร้างส่วนวากยสัมพันธ์ของโปรแกรมโดยอัตโนมัติ จากข้อกำหนดซอฟต์แวร์ของ

โปรแกรมนั้นๆ [19-21] ก็ต้องเพิ่มรายละเอียดเฉพาะเพิ่มเติมสำหรับข้อกำหนดซอฟต์แวร์นั้นๆ เช่นกัน

ดังนั้นเพื่อความสะดวกสำหรับผู้พัฒนาโปรแกรม และสะดวกในการจัดเก็บรายละเอียดเฉพาะที่จำเป็นต้องใช้ในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม งานวิจัยนี้จึงนำเสนอขั้นตอนในการจัดเก็บรายละเอียดเฉพาะที่จำเป็นพร้อมทั้งสร้างโครงภาษาจาวาเพื่อให้ผู้พัฒนาโปรแกรมสามารถพัฒนาโปรแกรมได้สะดวกและครบถ้วนยิ่งขึ้น รายละเอียดของการสร้างโครงภาษาจาวาแสดงในบทที่ 4

3.3 การสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

เพื่อความสะดวกในการเปรียบเทียบระหว่างข้อกำหนดรูปนัยคาเฟโอบีเจและโปรแกรมภาษาจาวา งานวิจัยนี้เสนอวิธีเปลี่ยนส่วนความหมายของข้อกำหนดรูปนัยคาเฟโอบีเจเป็นข้อความสั่งของโปรแกรมภาษาจาวา

ดังที่ได้กล่าวมาข้างต้นว่าข้อกำหนดรูปนัยคาเฟโอบีเจมีการกำหนดรายละเอียดเฉพาะของโปรแกรมไว้น้อยกว่าภายในโปรแกรมภาษาจาวา ดังนั้นการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบจำเป็นต้องใช้ข้อมูลที่ได้รับมาจากขั้นตอนการสร้างโครงภาษาจาวาด้วย ผลลัพธ์ของขั้นตอนนี้คือข้อความสั่งของภาษาจาวาที่แสดงหรือทำให้สถานะของวัตถุในภาษาจาวามีผลลัพธ์เช่นเดียวกับสถานะที่ระบุไว้ในข้อกำหนดซอฟต์แวร์ โดยระบุตัวแปรที่สนใจ และเมท็อดที่ข้อความสั่งนี้ควรจะปรากฏอยู่ รายละเอียดของการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบแสดงในบทที่ 5

3.4 การตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

เมื่อผู้พัฒนาโปรแกรมได้พัฒนาโปรแกรมได้ระยะหนึ่งแล้ว ผู้พัฒนาสามารถนำโปรแกรมที่กำลังพัฒนาอยู่มาตรวจสอบความก้าวหน้าโดยเปรียบเทียบโปรแกรมที่พัฒนาอยู่นี้กับส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ โดยจะตรวจสอบที่เมท็อดที่ได้จากผลลัพธ์ของขั้นตอนการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

ผลลัพธ์ของขั้นตอนนี้คือจำนวนของส่วนความหมายของข้อกำหนดรูปนัยคาเฟโอบีเจที่ได้รับการพัฒนาเป็นโปรแกรมแล้ว เพื่อเป็นข้อมูลว่าการพัฒนาโปรแกรมได้ดำเนินไปถึงขั้นใดแล้ว รายละเอียดของการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมแสดงในบทที่ 6

บทที่ 4

การสร้างโครงภาษาจาวา

การสร้างโครงภาษาจาวาจากข้อกำหนดรูปนัยคาเฟโอบีเจ คือการนำส่วนวากยสัมพันธ์ของข้อกำหนดรูปนัยคาเฟโอบีเจมารวมกับรายละเอียดเฉพาะต่างๆ ที่ได้รับจากผู้พัฒนาโปรแกรม เพื่อสร้างโครงภาษาจาวาที่สอดคล้องกับข้อกำหนดรูปนัย ส่วนวากยสัมพันธ์ของข้อกำหนดรูปนัยคาเฟโอบีเจประกอบด้วย 2 ส่วนคือ ส่วนประกาศมอดูล และส่วนประกาศลายเซ็น ทั้งนี้โอเปอเรชันที่ประกาศในส่วนประกาศลายเซ็นมีหลายรูปแบบซึ่งแต่ละรูปแบบก็สามารถสร้างโครงภาษาจาวาได้แตกต่างกัน ดังนั้นงานวิจัยนี้จึงแบ่งชนิดของโอเปอเรชันเพื่อความสะดวกในการสร้างโครงภาษาจาวา

อย่างไรก็ดี ผู้ออกแบบระบบจำเป็นต้องเพิ่มรายละเอียดเฉพาะเพิ่มเติมให้กับข้อกำหนดซอฟต์แวร์ เพื่อวิเคราะห์ความสัมพันธ์ระหว่างมอดูลในข้อกำหนดรูปนัยคาเฟโอบีเจ และคลาส รวมถึงความสัมพันธ์ระหว่างโอเปอเรชันและเมท็อด เพื่อสร้างโครงภาษาจาวาให้สอดคล้องกับข้อกำหนดที่ได้ ในงานวิจัยนี้ได้เสนอการสร้างกราฟแสดงความสัมพันธ์ระหว่างมอดูล และกฎสำหรับการสร้างโครงภาษาจาวา

4.1 ชนิดของโอเปอเรชันในข้อกำหนดคาเฟโอบีเจ

ชนิดของโอเปอเรชันที่ประกาศไว้ในข้อกำหนดคาเฟโอบีเจ แบ่งตามชนิดของข้อมูลที่ปรากฏอยู่ในส่วนอาริที โดยสามารถแบ่งออกเป็นชนิดต่างๆ ได้ดังนี้

4.1.1 โอเปอเรชันที่ไม่มีอาริที

โอเปอเรชันที่ไม่มีอาริทีใช้อธิบายสถานะของมอดูลที่เกิดขึ้น เนื่องจากข้อกำหนดรูปนัยคาเฟโอบีเจอธิบายพฤติกรรมของระบบที่ออกแบบด้วยสถานะ โอเปอเรชันที่ไม่มีอาริทีแสดงสถานะที่เป็นไปได้ของมอดูลในข้อกำหนด เพื่อใช้ในการตรวจสอบความถูกต้องของข้อกำหนด

โอเปอเรชันที่ไม่มีอาริทีเป็นโอเปอเรชันที่ไม่มีส่วนนำเข้า แต่มีโคอาริทีอยู่ อย่างไรก็ดี ถ้าโคอาริทีเป็นชนิดข้อมูลสังเกตค่าได้จะถือว่าโอเปอเรชันนี้บ่งบอกสถานะของระบบ แต่ถ้าโคอาริทีเป็นชนิดข้อมูลแฝงที่ประกาศไว้ในมอดูลที่โอเปอเรชันนี้ถูกประกาศไว้ ถือว่าโอเปอเรชันนี้แสดงสถานะเริ่มต้นของมอดูล โอเปอเรชันที่ไม่มีอาริทีมีรูปแบบดังนี้

op **p**: $\longrightarrow$ **s**

โดยที่

p คือ ชื่อของโอเปอเรชันในข้อกำหนดรูปนัยคาเฟโอบีเจ

s คือ ชนิดของข้อมูล

4.1.2 โปรเจคชันโอเปอเรชันของชนิดข้อมูลสังเกตค่าได้ (Visible sort projection operation)

โปรเจคชันโอเปอเรชันของชนิดข้อมูลสังเกตค่าได้เป็นโอเปอเรชันสำหรับแสดงสถานะของมอดูลว่าขณะนี้อยู่ในสถานะใด เนื่องจากข้อกำหนดรูปนัยคาเฟโอบีเจเก็บสถานะของระบบไว้ในชนิดข้อมูลแฝง ซึ่งโดยตรรกะทางคณิตศาสตร์ที่ข้อกำหนดรูปนัยคาเฟโอบีเจใช้นั้น ไม่สามารถที่จะสังเกตสถานะจากชนิดข้อมูลแฝงได้โดยตรงจึงต้องแสดงชนิดข้อมูลแฝงนั้นออกมาเป็นชนิดข้อมูลสังเกตค่าได้เสียก่อน

โปรเจคชันโอเปอเรชันของชนิดข้อมูลสังเกตค่าได้มีอาริทีเป็นชนิดข้อมูลแฝง และมีชนิดข้อมูลสังเกตค่าได้เป็นโคอาริที โปรเจคชันโอเปอเรชันของชนิดข้อมูลสังเกตค่าได้มีรูปแบบดังนี้

op p: h $\longrightarrow$ v

โดยที่

p คือ ชื่อของโอเปอเรชันในข้อกำหนดคาเฟโอบีเจ

h คือ ชนิดข้อมูลแฝงใดๆ

v คือ ชนิดข้อมูลสังเกตค่าได้ใดๆ

4.1.3 โปรเจคชันโอเปอเรชันของชนิดข้อมูลแฝง (Hidden sort projection operation)

ในบางกรณีเมื่อมอดูลใดมอดูลหนึ่งประกาศนำเข้ามอดูลอื่น และมอดูลที่ถูกเรียกเข้ามาใช้มีสถานะเสมือนเป็นสมาชิกของมอดูลนี้ การตรวจสอบสถานะของมอดูลที่เป็นสมาชิกนี้สามารถเรียกเข้าได้โดยใช้โปรเจคชันโอเปอรชันของชนิดข้อมูลแฝง

โปรเจคชันโอเปอเรชันของชนิดข้อมูลแฝงมีอาริทีเป็นชนิดข้อมูลแฝง และมีชนิดข้อมูลแฝงอีกชนิดหนึ่งซึ่งประกาศไว้ในมอดูลที่ถูกเรียกเข้ามายังมอดูลที่ประกาศโอเปอเรชันนี้เป็นโคอาริที โปรเจคชันโอเปอเรชันของชนิดข้อมูลแฝงมีรูปแบบดังนี้

op p: h $\longrightarrow$ h_o

โดย

p คือ ชื่อของโอเปอเรชันในข้อกำหนดคาเฟโอบีเจ

h คือ ชนิดข้อมูลแฝงซึ่งประกาศไว้ในมอดูลที่ประกาศโอเปอเรชันนี้

h_o คือ ชนิดข้อมูลแฝงซึ่งประกาศไว้ในมอดูลที่ถูกเรียกเข้ามายังมอดูลที่ประกาศโอเปอเรชันนี้

4.1.4 โอเปอเรชันทั่วไป (General operation)

โอเปอเรชันทั่วไป คือโอเปอเรชันที่ไม่สามารถจัดเป็นโอเปอเรชันในข้อ 4.1.1 4.1.2 และ 4.1.3 ได้ เมื่อโอเปอเรชันทั่วไปถูกเรียก สถานะของระบบจะเปลี่ยนแปลงตามที่ได้ระบุเอาไว้ในส่วนความหมาย

โอเปอเรชันทั่วไปมีรูปแบบดังนี้

$$\text{op } p: s_1\ s_2\ \ldots\ s_m \longrightarrow s$$

โดยที่

p คือ ชื่อของโอเปอเรชันในข้อกำหนดคาเฟโอบีเจ

$s_1, s_2, \ldots s_m$ คือ รายการชนิดข้อมูลที่เป็นอาริทีของโอเปอเรชัน และ m เป็นจำนวนเต็มบวก

s คือ ชนิดข้อมูลใดๆ

4.2 การเพิ่มรายละเอียดเฉพาะสำหรับการสร้างโครงภาษาจาวา

ดังที่กล่าวไว้ข้างต้น เอกสารการออกแบบเช่น ข้อกำหนดรูปนัยคาเฟโอบีเจ แสดงรายละเอียดต่างๆ ของระบบน้อยกว่าส่วนที่ใช้ในการพัฒนาจริง ดังนั้นการสร้างส่วนที่ใช้ในการพัฒนาจากเอกสารออกแบบจึงต้องเพิ่มรายละเอียดที่เฉพาะเข้าไปด้วย รายละเอียดเฉพาะที่ต้องเพิ่มเข้าไปในขั้นตอนการสร้างโครงภาษาจาวาประกอบด้วย

4.2.1 มอดูลที่จะสร้างคลาสในโครงภาษาจาวา

ในข้อกำหนดแต่ละข้อกำหนดอาจประกอบด้วยมอดูลมากกว่า 1 มอดูล ประกอบกัน เพื่ออธิบายระบบทั้งหมด แต่ไม่จำเป็นที่ทุกมอดูลจะต้องถูกสร้างเป็นคลาส เนื่องจากมอดูลนั้นอาจจะกำหนดขึ้นมาเพื่อให้มอดูลอื่นๆ เรียกเข้าหรือเป็นมอดูลที่เก็บค่าคงที่เท่านั้น ดังนั้นผู้ออกแบบระบบต้องกำหนดก่อนว่าต้องการสร้างมอดูลใดให้เป็นคลาสบ้าง

4.2.2 ชนิดของข้อมูล (Data type) ของภาษาจาวา

คาเฟโอบีเจใช้ชนิดข้อมูลสังเกตค่าได้เพื่อแสดงสถานะของวัตถุในขณะนั้น โดยปกติชนิดข้อมูลสังเกตค่าได้ เป็นชนิดข้อมูลที่ผู้เขียนข้อกำหนดสร้างขึ้นเอง ดังนั้นจึงต้องกำหนดชนิดของข้อมูลในภาษาจาวาเพื่อใช้แสดงชนิดข้อมูลสังเกตค่าได้ เพื่อในขั้นตอนการสร้างโครงภาษาจาวาสามารถเลือกใช้ชนิดของข้อมูลในภาษาจาวาได้เหมาะสมกับชนิดข้อมูลในภาษาคาเฟโอบีเจ

4.2.3 ค่าของค่าคงที่ (Constant value) ในภาษาจาวา

บางโอเปอเรชันของข้อกำหนดคาเฟโอบีเจ อาจถูกกำหนดให้เป็นค่าคงที่สำหรับแสดงสถานะของมอดูล โดยโอเปอเรชันที่ถูกกำหนดให้เป็นค่าคงที่คือโอเปอเรชันที่ไม่มีอาริที ในการ

สร้างโครงภาษาจาวาต้องระบุว่าค่าคงที่เหล่านี้ต้องมีค่าเป็นเท่าใด โดยค่าที่กำหนดต้องสอดคล้องกับชนิดของข้อมูลของโคอาริทีตามที่ได้กำหนดในหัวข้อ 4.2.2

4.2.4 การตั้งชื่อเมท็อด และตัวแปรต่างๆ

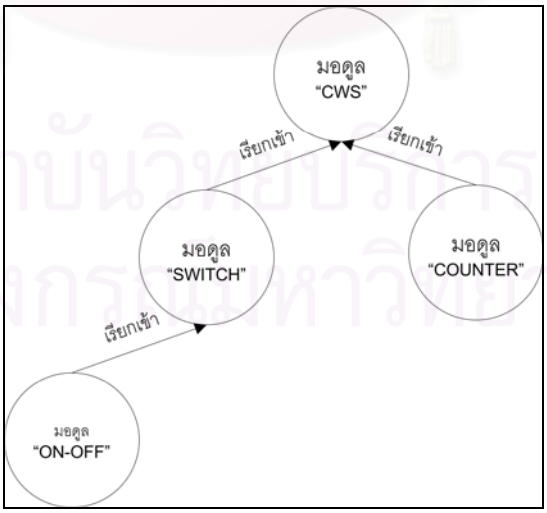
ชื่อโอเปอเรชันในภาษาคาเฟโอบีเจ อาจประกอบด้วยตัวอักษรที่ไม่สามารถใช้ตั้งเป็นชื่อเมท็อด หรือตัวแปรในภาษาจาวาได้ เช่นโอเปอเรชันชื่อ “_R*R_“ดังนั้นผู้ออกแบบระบบกำหนดชื่อเมท็อดสำหรับโอเปอเรชันเหล่านั้นให้อยู่ในรูปแบบที่สามารถใช้ในภาษาจาวาได้ รวมทั้งยังต้องมีการตั้งชื่อของตัวแปรสำหรับส่วนนำเข้าของแต่ละเมท็อดอีกด้วย

4.3 การเตรียมโครงสร้างข้อมูลสำหรับการสร้างโครงภาษาจาวา

ก่อนสร้างโครงภาษาจาวา ต้องจัดเตรียมความสัมพันธ์ระหว่างมอดูลภายในข้อกำหนดรูปนัยคาเฟโอบีเจแต่ละข้อกำหนดเสียก่อน เพื่อให้ได้ผลลัพธ์ที่ตรงกับความต้องการที่สุด ขั้นตอนการเตรียมโครงสร้างข้อมูลสำหรับการสร้างโครงภาษาจาวาประกอบด้วย

4.3.1 การสร้างกราฟแสดงความสัมพันธ์ระหว่างมอดูล

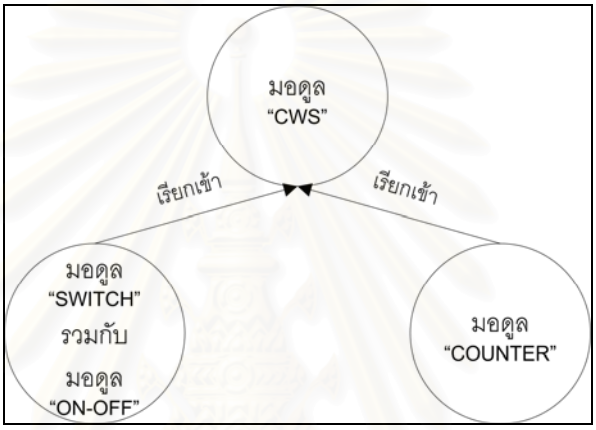
ความสัมพันธ์ระหว่างมอดูลในข้อกำหนดคาเฟโอบีเจจะเกิดขึ้นเมื่อมีมอดูลใดมอดูลหนึ่งเรียกเข้ามอดูลอื่น เข้าสู่มอดูลของตน ทั้งนี้ความสัมพันธ์นี้แสดงในรูปของกราฟแสดงความสัมพันธ์ระหว่างมอดูล ตัวอย่างกราฟแสดงความสัมพันธ์สำหรับข้อกำหนดการนับเลขที่มีสวิตช์ (Counter-with-Switch) แสดงในรูปที่ 4.1 ซึ่งข้อกำหนดนี้แสดงในภาคผนวก ก มอดูล “SWITCH“ เรียกเข้ามอดูล “ON-OFF“ และมอดูล “CWS“ เรียกเข้ามอดูล “SWITCH“ และมอดูล “COUNTER“



รูปที่ 4.1 กราฟความสัมพันธ์ระหว่างมอดูลต่างๆ ในข้อกำหนดการนับเลขที่มีสวิตช์

4.3.2 การลดรูปกราฟ

จากข้อมูลในหัวข้อ 4.2.1 มอดูลที่ไม่ถูกเลือกให้สร้างเป็นคลาสจะถูกขจัดออกไปจากกราฟ โดยข้อมูลของมอดูลนั้น จะถูกนำไปรวมอยู่ในมอดูลที่เรียกเข้ามอดูลนั้น ดังนั้นกราฟแสดงความสัมพันธ์ระหว่างมอดูลจะเหลือเพียงส่วนที่เก็บข้อมูลของมอดูลที่สร้างเป็นคลาสเท่านั้น รูปที่ 4.2 แสดงกราฟที่ลดรูปกราฟแล้ว โดยเลือกให้ มอดูล “CWS“ มอดูล “COUNTER“ และมอดูล “SWITCH“ ถูกนำไปสร้างเป็นคลาส โดยข้อมูลในมอดูล “ON-OFF“ จะถูกรวมอยู่ในมอดูลซึ่งเรียกเข้ามอดูล “ON-OFF“ นั่นคือมอดูล “SWITCH“ นั่นเอง



รูปที่ 4.2 กราฟแสดงความสัมพันธ์ที่ลดรูปแล้วของข้อกำหนดการนับเลขที่มีสวิตซ์ที่ลดรูปแล้ว

4.4 กฎในการสร้างโครงภาษาจาวา

เมื่อลดมอดูลที่ไม่จำเป็นต้องสร้างโครงภาษาจาวาแล้ว จึงเริ่มพิจารณาส่วนวากยสัมพันธ์ที่ประกาศอยู่ในมอดูลที่ลดรูปแล้ว การสร้างโครงภาษาจาวาจะสร้างทีละมอดูลที่แสดงอยู่ในกราฟแสดงความสัมพันธ์ที่ลดรูปแล้วโดย 1 มอดูลต่อ 1 คลาส ทั้งนี้การสร้างส่วนประกอบภายในคลาสพิจารณาตามกฎต่อไปนี้

4.4.1 สำหรับโปรเจคชันโอเปอเรชันของชนิดข้อมูลสังเกตค่าได้; op **p**: **h** ⟶ **v**

สามารถแปลงเป็นโครงภาษาจาวาได้ดังนี้

```
public v p(){
    }
```

โดยที่

p คือ ชื่อเมท็อดที่ถูกกำหนดให้สอดคล้องกับ **p** ซึ่งกำหนดจากข้อ 4.2.4

v คือ ชนิดของข้อมูลของภาษาจาวาที่สอดคล้องกับ **v** จากข้อ 4.2.2

จากผลลัพธ์ของกฎข้อนี้ทำให้สามารถระบุได้ว่าชนิดของข้อมูลภาษาจาวาที่สอดคล้องกับชนิดข้อมูลแฝง h คือชนิดของข้อมูลใด โดยชนิดของข้อมูลภาษาจาวาของชนิดข้อมูลแฝง จะเป็นชนิดของข้อมูลชนิดเดียวกับชนิดของข้อมูลที่สอดคล้องกับชนิดข้อมูลสังเกตค่าได้ที่ปรากฏอยู่ในโอเปอเรชันนั้น อย่างไรก็ดีชนิดข้อมูลแฝง อาจประกอบด้วยชนิดข้อมูลภาษาจาวาหลายๆ ชนิดข้อมูลได้หากมีโปรเจชันโอเปอเรชันของชนิดข้อมูลสังเกตค่าได้ที่มีอาริทีเป็นชนิดข้อมูลแฝงชนิดเดียวกัน มากกว่าหนึ่งโอเปอเรชัน

4.4.2 สำหรับโปรเจคชันโอเปอเรชันของชนิดข้อมูลแฝง; op p: h $\longrightarrow$ h_o

สามารถแปลงเป็นโครงภาษาจาวาได้ดังนี้

public *h p*(){

}

โดยที่

p คือ ชื่อเมท็อดที่ถูกกำหนดให้สอดคล้องกับ p ซึ่งกำหนดจากข้อ 4.2.4

ถ้า h_o ถูกประกาศในมอดูลเดียวกับ h

h คือ ชนิดของข้อมูลของภาษาจาวาที่สอดคล้องกับ v จากข้อ 4.2.2

มิฉะนั้น

h คือ ชื่อคลาสที่สอดคล้องกับมอดูลที่ประกาศชนิดข้อมูลแฝง h_o

4.4.3 สำหรับโอเปอเรชันที่ไม่มีอาริที; op p: $\longrightarrow$ s

ถ้าโอเปอเรชัน p ไม่ได้ถูกกำหนดให้เป็นสถานะเริ่มต้น โอเปอเรชัน p สามารถแปลงเป็นโครงภาษาจาวาได้ดังนี้

public *s p* = *a*;

โดยที่

p คือ ชื่อของค่าคงที่ที่สอดคล้องกับ p ซึ่งกำหนดจากข้อ 4.2.3

s คือ ชนิดของข้อมูลของภาษาจาวาที่สอดคล้องกับ s

a คือ ค่าคงที่ของ p ซึ่งกำหนดจากข้อ 4.2.3

4.4.4 สำหรับโอเปอเรชันทั่วไป; p: s_1 s_2 ... s_m $\longrightarrow$ s

สามารถแปลงเป็นโครงภาษาจาวาได้ดังนี้

public *type p*(s_1 n_1,s_2 n_2,...,s_n n_n){

}

โดยที่

p คือ ชื่อเมท็อดที่ถูกกำหนดให้สอดคล้องกับ p ซึ่งกำหนดจากข้อ 4.2.4

ถ้า **s** เป็นชนิดข้อมูลแฝง

type แสดงผลเป็น “void”

มิฉะนั้น

type คือ ชนิดของข้อมูลของภาษาจาวาที่สอดคล้องกับ **s**

สำหรับแต่ละชนิดข้อมูล $\mathbf{s}_i$ ในรายการอาริที

ถ้า $\mathbf{s}_i$ เป็นชนิดข้อมูลสังเกตค่าได้

s_i คือ ชนิดของข้อมูลของภาษาจาวาที่สอดคล้องกับ $\mathbf{s}_i$ และ

n_i คือ ชื่อของพารามิเตอร์นำเข้าสำหรับชนิดข้อมูล s_i

มิฉะนั้น

ไม่ต้องสร้างโครงภาษาจาวาสำหรับ $\mathbf{s}_i$

จากกฎการสร้างโครงภาษาจาวาในข้อนี้ ถ้าอาริทีเป็นชนิดข้อมูลแฝงจะไม่สร้างชนิดของข้อมูลของภาษาจาวา และสร้างพารามิเตอร์สำหรับเมท็อดที่สอดคล้องกับโอเปอเรชันนี้ เนื่องจากชนิดข้อมูลแฝงเป็นชนิดข้อมูลที่เก็บข้อมูลไว้ภายในมอดูล หรืออีกนัยหนึ่งถูกเก็บไว้ภายในมอดูลอยู่แล้วจึงไม่จำเป็นต้องรับเป็นพารามิเตอร์นำเข้าสำหรับเมท็อดนี้ ในทำนองเดียวกันถ้าชนิดข้อมูลที่ปรากฏในโคอาริทีเป็นชนิดข้อมูลแฝง ข้อมูลส่งออกของเมท็อดนี้ก็จะให้ค่า “void“ ซึ่งไม่ส่งค่าออกมา เนื่องจากสถานะผลลัพธ์ถูกเก็บไว้ภายในมอดูลแล้ว

4.4.5 สำหรับการประกาศชนิดข้อมูลแฝง; *[**x**]*

โดยที่

x เป็น ชื่อของชนิดข้อมูลแฝง

ถ้ามีโปรเจคชันโอเปอเรชันของชนิดข้อมูลสังเกตค่าได้ซึ่งมี **x** เป็นอาริทีและมี **y** เป็นโคอาริที แล้ว สามารถแปลงเป็นโครงภาษาจาวาได้ดังนี้

public *y x*;

โดยที่

y คือ ชนิดของข้อมูลของภาษาจาวาที่สอดคล้องกับ **y**

x คือ ชื่อของตัวแปรภาษาจาวาที่สอดคล้องกับ **x**

กฎข้อนี้สัมพันธ์กับกฎข้อ 4.4.1 ที่ระบุชนิดของข้อมูลภาษาจาวาที่สอดคล้องกับชนิดข้อมูลแฝงที่ประกาศไว้โดยชื่อของตัวแปรที่แสดงข้อมูลแฝงนั้นได้รับการระบุมาจากขั้นตอน 4.2.4

4.4.6 สำหรับการประกาศนำเข้ามอดูล; pr (**x**)

โดยที่ **x** เป็นชื่อมอดูล

ถ้า **x** ถูกแปลงให้เป็นคลาสของโปรแกรมภาษาจาวาและในมอดูลที่เรียกมอดูล **x** เข้ามามีโปรเจคชันโอเปอเรชันของชนิดข้อมูลแฝงซึ่งมีชนิดข้อมูลแฝงที่ประกาศใน **x** เป็นโคอาริทีแล้ว สามารถแปลงเป็นโครงภาษาจาวาได้ดังนี้

public x a;

โดยที่

x คือ ชื่อคลาสซึ่งสร้างจากมอดูล **x**

a คือ ชื่อของตัวแปรภาษาจาวาที่สอดคล้องกับ **x**

โครงภาษาจาวาที่สร้างข้อกำหนดการนับเลขที่มีตัวนับซึ่งแสดงในภาคผนวก ก แสดงอยู่ในภาคผนวก ง

บทที่ 5

การสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

การสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบทำได้โดยการวิเคราะห์ส่วนประกาศสัจพจน์ของข้อกำหนดรูปนัยคาเฟโอบีเจร่วมกับรายละเอียดเฉพาะซึ่งได้รับจากผู้ออกแบบระบบประกอบกับขั้นตอนวิธีซึ่งนำเสนอในบทนี้

5.1 การวิเคราะห์ส่วนกำหนดสัจพจน์

ส่วนกำหนดสัจพจน์ประกอบด้วยสองส่วนหลักคือ ส่วนประกาศตัวแปร และส่วนประกาศสมการ

ส่วนประกาศตัวแปรใช้กำหนดตัวแปรที่เป็นส่วนนำเข้าของโอเปอเรชันแต่ละโอเปอเรชัน การกำหนดตัวแปรสามารถกำหนดได้ตามที่แสดงไว้ในหัวข้อ 2.1.3.1

ในขั้นตอนการสร้างโครงภาษาจาวา สำหรับโอเปอเรชันทั่วไป ถ้าชนิดของอาริทีที่อยู่ในรายการอาริทีเป็นชนิดข้อมูลสังเกตค่าได้สามารถสร้างพารามิเตอร์นำเข้าของเมท็อดที่สอดคล้องกับโอเปอเรชันนั้น โดยชื่อของตัวแปรที่ทำหน้าที่พารามิเตอร์นำเข้าซึ่งถูกกำหนดโดยผู้ออกแบบ จะถูกผูก (bind) ไว้กับตัวแปรที่ทำหน้าที่เป็นส่วนนำเข้าของโอเปอเรชันนั้นๆ

ส่วนประกาศสมการเป็นการระบุสถานะที่เปลี่ยนแปลงไปเมื่อโอเปอเรชัน หรือรายการของโอเปอเรชันถูกเรียกใช้ การกำหนดสมการที่สามารถสร้างส่วนของโปรแกรมภาษาจาวาได้มีสองประเภทคือสมการแบบไม่มีเงื่อนไข และสมการแบบมีเงื่อนไขดังรายละเอียดต่อไปนี้

5.1.1 สมการแบบไม่มีเงื่อนไข

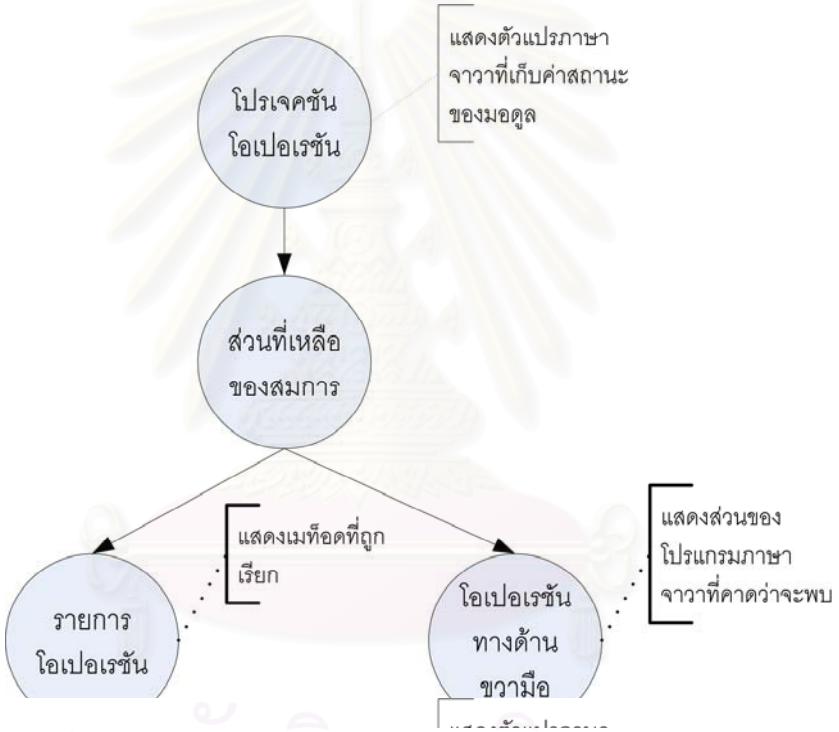
รูปแบบของสมการแบบไม่มีเงื่อนไขแสดงในหัวข้อ 2.1.3.2 สมการแบบไม่มีเงื่อนไขประกอบด้วย คำสำคัญที่ใช้ในการกำหนดสมการแบบไม่มีเงื่อนไข พจน์ทางด้านซ้ายของเครื่องหมายเท่ากับ และพจน์ทางด้านขวาของเครื่องหมายเท่ากับ พจน์คือรายการของโอเปอเรชันที่ระบุไว้ภายในสมการ สมการแบบไม่มีเงื่อนไขแสดงสถานะที่เปลี่ยน (พจน์ทางด้านขวาของเครื่องหมายเท่ากับ) เมื่อรายการโอเปอเรชัน (โอเปอเรชันที่ระบุอยู่ในพจน์ทางซ้ายของเครื่องหมายเท่ากับ) ถูกเรียกใช้

โดยปกติแล้วพจน์ทางด้านซ้ายของเครื่องหมายเท่ากับซึ่งเป็นรายการโอเปอเรชันที่ถูกเรียกประกอบด้วยโปรเจคชันโอเปอเรชัน และรายการของโอเปอเรชันที่เปลี่ยนสถานะของมอดูลโปรเจคชันโอเปอเรชันเป็นโอเปอเรชันที่ระบุได้ว่าขณะนี้การสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบสนใจการเปลี่ยนแปลงค่าของตัวแปรที่เก็บสถานะตัวใด ทั้งนี้ในขั้นตอนการสร้างโครง

ภาษาจาวาได้ระบุไว้แล้วว่าโปรเจคชันโอเปอเรชันแต่ละโอเปอเรชันจะแสดงถึงตัวแปรใดในโครงภาษาจาวา โดยพิจารณาจากอาริทีและโคอาริทีของโปรเจคชันโอเปอเรชันนั้นๆ

รายการของโอเปอเรชันที่เปลี่ยนสถานะของมอดูลแสดงเมท็อดในคลาสของโปรแกรมภาษาจาวาซึ่งเปลี่ยนสถานะของตัวแปรที่สนใจ ชื่อของเมท็อดที่เปลี่ยนสถานะได้จากการกำหนดชื่อเมท็อดที่สอดคล้องกับโอเปอเรชันในหัวข้อ 4.2.4 และรายการโอเปอเรชันในพจน์ทางขวามือของเครื่องหมายเท่ากับแสดงผลลัพธ์ที่เกิดขึ้นกับตัวแปรที่ได้จากโปรเจคชันโอเปอเรชัน

จากโครงสร้างของสมการแบบไม่มีเงื่อนไขสามารถสร้างเป็นโครงสร้างข้อมูลต้นไม้เพื่อเก็บผลลัพธ์ดังแสดงในรูปที่ 5.1



รูปที่ 5.1 โครงสร้างข้อมูลต้นไม้สำหรับสมการแบบไม่มีเงื่อนไข

5.1.2 สมการแบบมีเงื่อนไข

รูปแบบของสมการแบบมีเงื่อนไขแสดงในหัวข้อ 2.1.3.2 โดยสมการแบบมีเงื่อนไขประกอบด้วย คำสำคัญที่ใช้ในการกำหนดสมการแบบมีเงื่อนไข พจน์ทางซ้ายของเครื่องหมายเท่ากับ พจน์ทางขวาของเครื่องหมายเท่ากับ และพจน์เงื่อนไข สมการแบบมีเงื่อนไขแสดงสถานะที่เปลี่ยน (พจน์ทางด้านขวาของเครื่องหมายเท่ากับ) เมื่อรายการโอเปอเรชัน (โอเปอเรชันที่ระบุอยู่ในพจน์ทางด้านซ้ายของเครื่องหมายเท่ากับ) ถูกเรียก และพจน์เงื่อนไขมีค่าเป็นจริง

พจน์ทางด้านซ้ายของเครื่องหมายเท่ากับ และพจน์ทางด้านขวาของเครื่องหมายเท่ากับของสมการแบบมีเงื่อนไขมีลักษณะเดียวกันกับพจน์ทางด้านซ้าย และด้านขวาของเครื่องหมายเท่ากับของสมการแบบไม่มีเงื่อนไข ส่วนที่เพิ่มเข้ามาคือพจน์เงื่อนไข

พจน์เงื่อนไขเป็นพจน์ที่จะคืนค่าจริงหรือเท็จ ทั้งนี้พจน์เงื่อนไขมักจะมีเครื่องหมายเปรียบเทียบ เช่นการเปรียบเทียบการเท่ากันระหว่างพจน์สองพจน์ที่สนใจโดยใช้เครื่องหมาย “==” เป็นต้น โดยในการเปรียบเทียบจะเปรียบเทียบระหว่างส่วนที่อยู่ทางด้านซ้ายของเครื่องหมายเปรียบเทียบ และส่วนที่อยู่ทางด้านขวาของเครื่องหมายเปรียบเทียบ ส่วนที่อยู่ทั้งทางด้านซ้ายและขวาของเครื่องหมายเปรียบเทียบสามารถนำมาวิเคราะห์เพื่อหาส่วนของภาษาจาวาที่คาดว่าจะพบในส่วนข้อความสั่งเปรียบเทียบภายในโปรแกรมภาษาจาวา

ในบางโอกาสอาจมีสมการแบบมีเงื่อนไขมากกว่าหนึ่งสมการที่มีพจน์ทางซ้ายมือของเครื่องหมายเท่ากับ และส่วนที่อยู่ทางซ้ายมือของเครื่องหมายเปรียบเทียบเหมือนกันได้ ดังตัวอย่างจากส่วนกำหนดสัจพจน์ของมอดูล “COUNTER-WITH-SWITCH“ ในข้อกำหนดตัวอย่างในภาคผนวก ก ที่แสดงใน รูปที่ 5.2

```
ceq counter(put(N, C)) = add(N, counter(C))
    if state(switch(C)) == on .
ceq counter(put(N, C)) = add(-(N), counter(C))
    if state(switch(C)) == off .
```

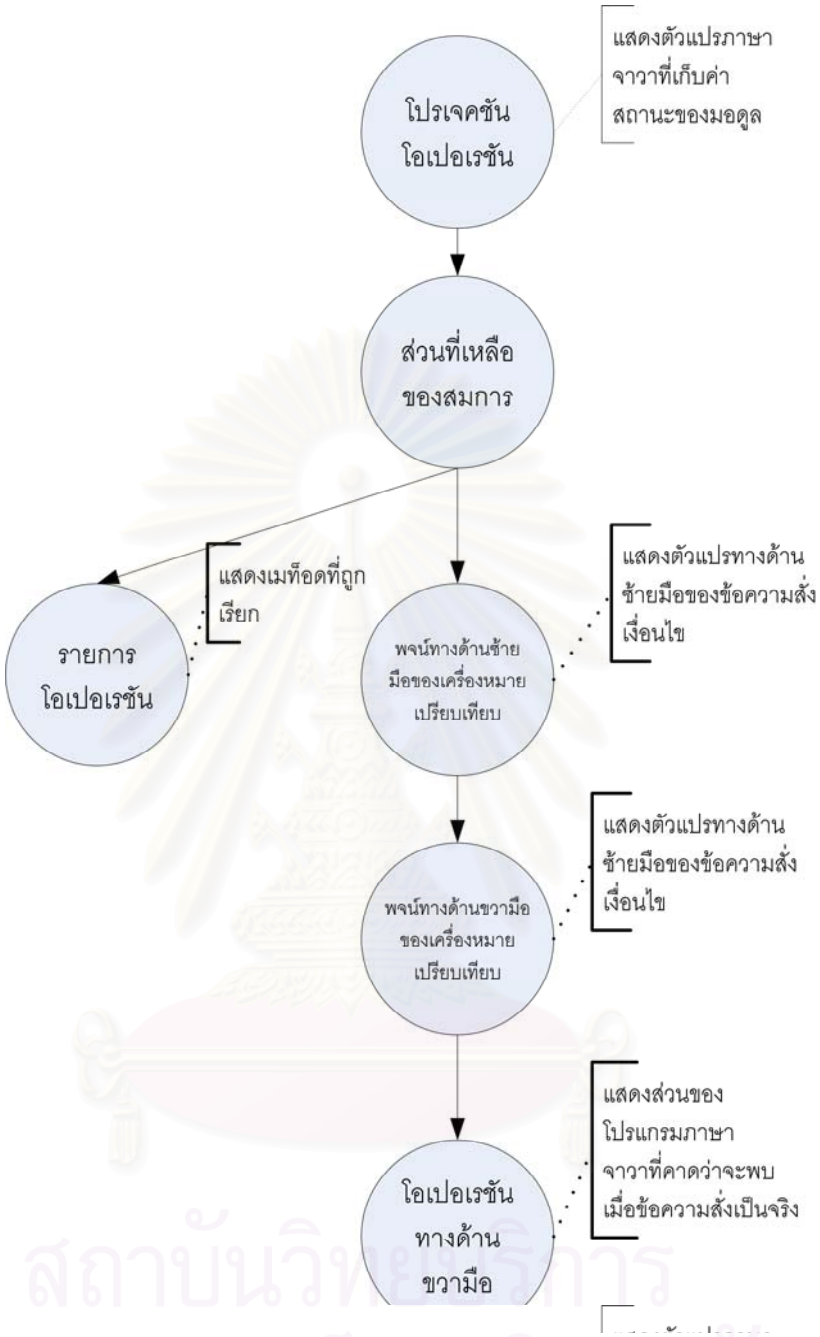
รูปที่ 5.2 สมการแบบมีเงื่อนไขที่มีบางส่วนเหมือนกัน

จากโครงสร้างของสมการแบบมีเงื่อนไขสามารถสร้างเป็นโครงสร้างข้อมูลต้นไม้เพื่อเก็บผลลัพธ์ดังแสดงในรูปที่ 5.3

5.2 ขั้นตอนวิธีการสร้างส่วนของภาษาจาวาที่คาดว่าจะพบ

ดังที่ได้กล่าวไว้ข้างต้นว่าส่วนของพจน์ด้านซ้ายของเครื่องหมายเท่ากับให้ข้อมูลเกี่ยวกับตัวแปรที่เก็บสถานะของวัตถุ และเมท็อดที่เรียกใช้ แต่ในส่วนของพจน์ด้านขวาของเครื่องหมายเท่ากับ และพจน์เงื่อนไขนั้นต้องนำมาผ่านขั้นตอนวิธีการสร้างส่วนของภาษาจาวาที่คาดว่าจะพบเสียก่อนจึงจะได้ผลลัพธ์ที่ต้องการออกมา

ในขั้นตอนวิธีการสร้างส่วนของภาษาจาวาที่คาดว่าจะพบ ต้องแบ่งชนิดของพจน์ที่นำมาพิจารณา โดยแบ่งพจน์ออกเป็นชนิดต่างๆ ตามจำนวนของโอเปอเรชันแต่ละชนิดที่มีอยู่ในพจน์นั้นๆ ดังนี้



รูปที่ 5.3 โครงสร้างข้อมูลต้นไม้สำหรับสมการแบบมีเงื่อนไข

- พจน์ชนิดไม่มีอาริที (Non arity type) คือพจน์ที่มีเพียงโอเปอเรชันที่ไม่มีอาริทีเท่านั้น
- พจน์ชนิดโปรเจคชันโอเปอเรชัน (Projection operation type) คือพจน์ที่มีโปรเจคชันโอเปอเรชันปรากฏอยู่ในพจน์นั้นๆ
- พจน์ชนิดรายการตัวแปร (Variable list type) คือพจน์ที่ไม่มีโปรเจคชันโอเปอเรชัน แต่มีตัวแปรของข้อกำหนดรูปนัยคาเฟโอบีเจ หรือชื่อของตัวแปรใน

โปรแกรมภาษาจาวา หรือพจน์ที่มีโอเปอเรชันใดๆ ร่วมกับชื่อของตัวแปรในโปรแกรมภาษาจาวา

- พจน์ชนิดรายการของโอเปอเรชัน คือพจน์ที่ไม่สามารถจัดเข้าเป็นชนิดหนึ่งชนิดใดในสามชนิดข้างต้น

ขั้นตอนวิธีการสร้างโครงภาษาจาวาที่คาดว่าจะพบแสดงในรูปที่ 5.4

```
input operation_list;
output expected_output;
Begin
        finish := false;
        left_op := operation_list;
        while (not finish)
                type_of_operation_list := check type of the left_op;
                case type_of_operation_list
                        non arity type:
                                constant_name := get the Java constant name
                                        corresponding to the non arity operation in
                                        left_op;
                                expected_output := expected_output appends
                                        constant_name;
                                finish := true;
                        // end non arity type
                        projection variable type:
                                        left_op := hanging_operator appends
                                                variable_name;
                                        hanging_operator := nil;
                                else
                                        expected_output := expected_output appends
                                                variable_name;
                                        finish := true;
                                endif
                        // end projection variable type
                        variable list type:
                                if there is a Java variable name and a operation in
                                        left_op then
                                expected_output := the Java variable name
                                        that calls the method which is
                                        corresponding to the operation in
                                        the left_op;
                                finish := true;
                                else
                                        substitutes CafeOBJ's variables in left_op
                                                with Java variable name which is
                                                related with;
                                endif
                        // end variable list type
```

รูปที่ 5.4 ขั้นตอนวิธีการสร้างโครงภาษาจาวาที่คาดว่าจะพบ

```
                        list of opereation type:
                                token := the first token from left_op;
                                left_op := left_op excepted the first operation;
                                if hanging_operator contains values then
                                        hanging_operator := hanging_operator
                                                appends token;
                                else
                                        hanging_operator := token;
                                endif
                        // end variable list type
                endcase // type mapping
        endwhile
        output expected_output;
End.
```

รูปที่ 5.4 ขั้นตอนวิธีการสร้างโครงภาษาจาวาที่คาดว่าจะพบ (ต่อ)

ส่วนของโปรแกรมที่คาดว่าจะพบคือส่วน “expected_output” ที่ได้จากขั้นตอนวิธีที่นำเสนอนี้

5.3 ลักษณะของส่วนของโปรแกรมที่คาดว่าจะพบ

ส่วนของโปรแกรมที่คาดว่าจะพบเป็นข้อความสั่งที่คาดว่าจะพบในเมท็อด ทั้งนี้ข้อความสั่งที่คาดว่าจะพบ แบ่งออกได้เป็น 3 ประเภทดังนี้

5.3.1 ข้อความสั่งแทนตัวเอง(Self declaration statement)

ข้อความสั่งแทนตัวเองจะเป็นข้อความสั่งที่มีแต่ชื่อของตัวแปรภาษาจาวาที่สนใจซึ่งได้มาจากส่วนโปรเจคชันโอเปอเรชันในพจน์ทางซ้ายของเครื่องหมายเท่ากับ ตัวอย่างของข้อความสั่งแทนตัวเองแสดงในรูปที่ 5.5

```
Switch;
```

รูปที่ 5.5 ตัวอย่างข้อความสั่งแทนตัวเอง

5.3.2 ข้อความสั่งกำหนดค่า(Assignment statement)

ข้อความสั่งกำหนดค่าคือข้อความสั่งที่เปลี่ยนค่าของตัวแปรภาษาจาวาที่สนใจด้วยค่าที่แสดงทางด้านขวามือของเครื่องหมายกำหนดค่า (Assignment operator) ตัวอย่างของข้อความสั่งกำหนดค่าแสดงในรูปที่ 5.6

```
Switch = true;
```

รูปที่ 5.6 ตัวอย่างข้อความสั่งกำหนดค่า

5.3.3 ข้อความสั่งเรียกใช้เมท็อด(Method call statement)

ข้อความสั่งเรียกใช้เมท็อดคือข้อความสั่งที่แสดงการเรียกใช้เมท็อดของตัวแปรภาษาจาวาที่สนใจ เมท็อดที่เรียกใช้ต้องเป็นเมท็อดที่ประกาศไว้ในคลาสของตัวแปรภาษาจาวาที่สนใจ ตัวอย่างของข้อความสั่งเรียกใช้เมท็อดแสดงในรูปที่ 5.7

```
Switch.On();
```

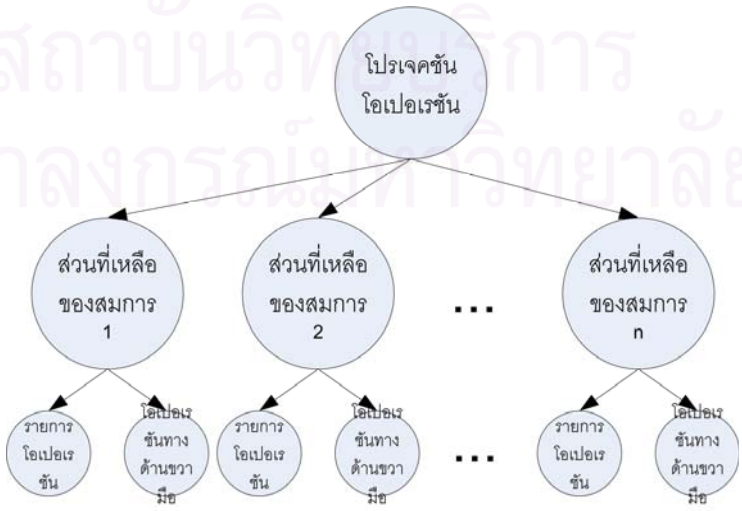
รูปที่ 5.7 ตัวอย่างข้อความสั่งเรียกใช้เมท็อด

5.4 โครงสร้างข้อมูลต้นไม้ของส่วนของภาษาจาวาที่คาดว่าจะพบในมอดูล

การออกแบบระบบโดยใช้ข้อกำหนดรูปนัยคาเฟโอบีเจแบ่งระบบออกเป็นมอดูลต่างๆ อีกทั้งการสร้างโครงภาษาจาวาก็อ้างอิงมอดูลในข้อกำหนดรูปนัยคาเฟโอบีเจ ดังนั้นเพื่อความสะดวกในการอ้างอิงในขั้นตอนการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม งานวิจัยนี้จึงนำโครงสร้างข้อมูลต้นไม้ของสมการต่างๆ รวบรวมอยู่ภายใต้โครงสร้างข้อมูลต้นไม้มอดูลเดียวกัน การรวมโครงสร้างข้อมูลต้นไม้มีขั้นตอนดังนี้

5.4.1 การรวมโครงสร้างข้อมูลต้นไม้ของสมการแบบไม่มีเงื่อนไข

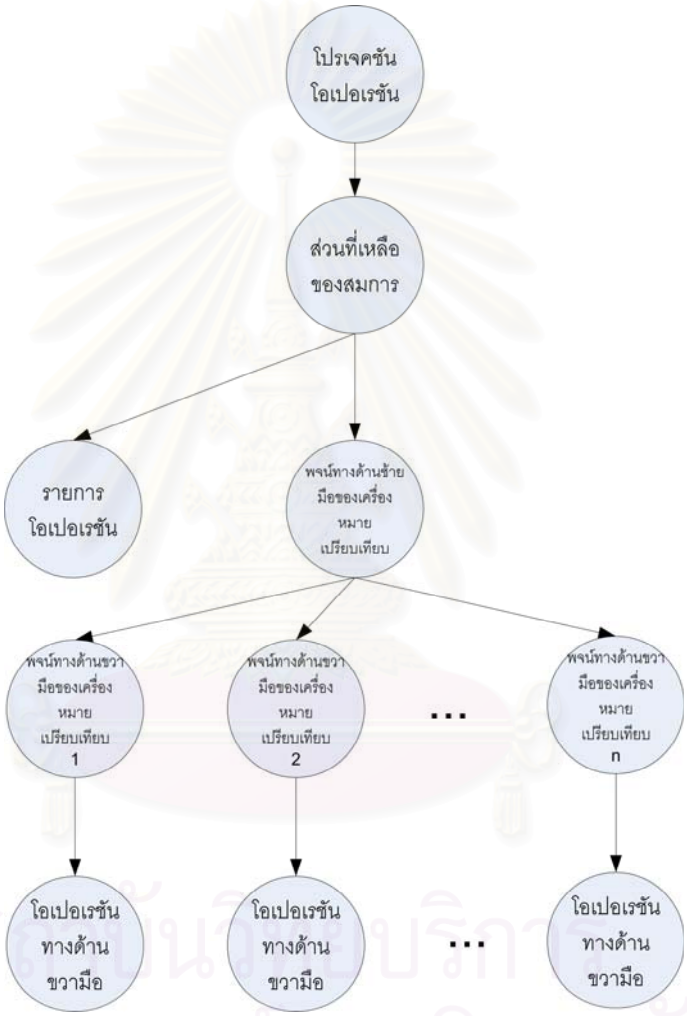
เนื่องจากโปรเจคชันโอเปอเรชันที่ปรากฏอยู่ในพจน์ทางด้านซ้ายมือของเครื่องหมายเท่ากับของสมการแบบไม่มีเงื่อนไข แสดงตัวแปรภาษาจาวาที่สนใจ ดังนั้นการรวมโครงสร้างข้อมูลต้นไม้ของสมการแบบไม่มีเงื่อนไขจึงรวมเอาโครงสร้างข้อมูลต้นไม้ของสมการแบบไม่มีเงื่อนไขที่อ้างอิงถึงตัวแปรภาษาจาวาตัวเดียวกัน นั่นคือรวมส่วนรากของโครงสร้างข้อมูลต้นไม้ที่ระบุโปรเจคชันโอเปอเรชันเดียวกันเข้าด้วยกันดังแสดงในรูปที่ 5.8



รูปที่ 5.8 โครงสร้างข้อมูลแบบต้นไม้ของรายการสมการแบบไม่มีเงื่อนไข

5.4.2 การรวมโครงสร้างข้อมูลต้นไม้ของสมการแบบมีเงื่อนไข

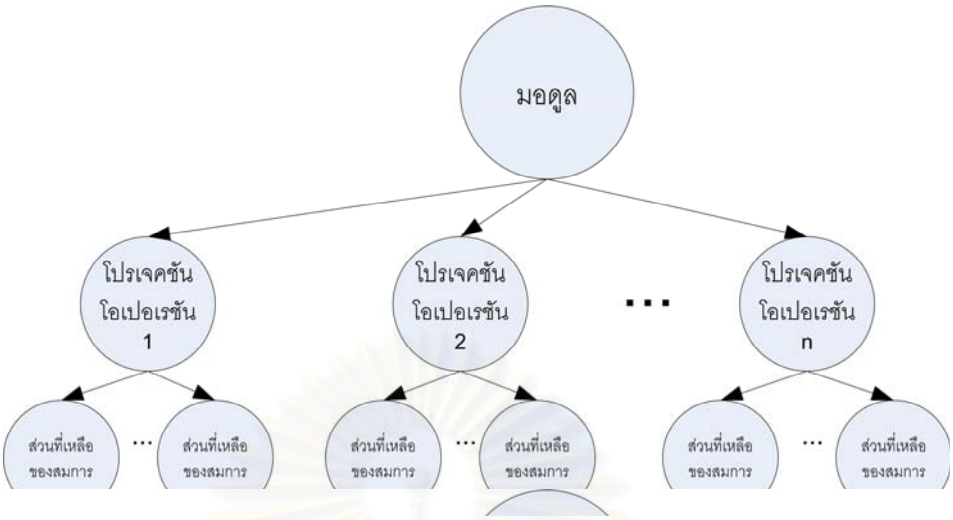
การรวมโครงสร้างข้อมูลต้นไม้ของสมการแบบมีเงื่อนไขก็เช่นเดียวกับการรวมโครงสร้างข้อมูลต้นไม้ของสมการแบบไม่มีเงื่อนไข นั่นคือ รวมข้อมูลส่วนที่มีพจน์ทางด้านซ้ายของเครื่องหมายเท่ากับและพจน์ทางด้านซ้ายของเครื่องหมายเปรียบเทียบเหมือนกันเข้าด้วยกันให้โหนดทีเหลือเป็นลูกของโหนดนั้นดังแสดงในรูปที่ 5.9



รูปที่ 5.9 โครงสร้างข้อมูลต้นไม้ของรายการสมการแบบมีเงื่อนไข

5.4.3 การรวมโครงสร้างข้อมูลต้นไม้ของมอดูล

เมื่อรวมโครงสร้างต้นไม้ของสมการแต่ละแบบเรียบร้อยแล้ว โครงสร้างข้อมูลต้นไม้ของสมการทั้งหมดจะถูกรวมอยู่ในโครงสร้างข้อมูลต้นไม้มอดูลเดียวกัน โครงสร้างข้อมูลต้นไม้ของมอดูลจะมีรากแสดงชื่อของมอดูล และมีลูกคือโครงสร้างข้อมูลต้นไม้ของแต่ละสมการมารวมกันโดยรวมโหนดของโปรเจคชันโอเปอเรชันที่อ้างอิงโปรเจคชันโอเปอเรชันชนิดเดียวกันเข้าไว้ด้วยกัน โครงสร้างข้อมูลต้นไม้ของมอดูลแสดงในรูปที่ 5.10



รูปที่ 5.10 โครงสร้างข้อมูลต้นไม้ของมอดูล

บทที่ 6
การตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

หลังจากสร้างโครงภาษาจาวาเพื่อให้ผู้พัฒนาพัฒนาโปรแกรมในส่วนความหมายเพิ่มเติมแล้ว ผู้พัฒนาจะต้องส่งโปรแกรมภาษาจาวาที่กำลังพัฒนาอยู่ เพื่อตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม โดยเปรียบเทียบกับส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ งานวิจัยนี้เสนอขั้นตอนวิธีการตรวจสอบความก้าวหน้าของการพัฒนโปรแกรมดังกล่าว โดยวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม มีขั้นตอนดังนี้

6.1 การวิเคราะห์หาเงื่อนไขของการตัดส่วนโปรแกรม

ก่อนที่จะตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมนั้น ต้องตัดส่วนโปรแกรมเพื่อแบ่งเอาเฉพาะส่วนของโปรแกรมที่เกี่ยวข้องในการตรวจสอบความก้าวหน้าเท่านั้นมาพิจารณา โดยทั่วไปการตัดส่วนโปรแกรมต้องสร้างเงื่อนไข (Criteria) ของการตัดส่วนโปรแกรมเพื่อให้ได้ส่วนของโปรแกรมที่ต้องการ เงื่อนไขของการตัดส่วนโปรแกรมประกอบด้วยชื่อของตัวแปร ชื่อคลาส และชื่อเมท็อดที่สนใจ ในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมจะตรวจสอบความก้าวหน้าที่เกิดขึ้นทีละสมการของข้อกำหนดรูปนัยคาเฟโอบีเจซึ่งระบุไว้ในขั้นตอนการสร้างส่วนของภาษาจาวาที่คาดว่าจะพบ

ในขั้นตอนการสร้างส่วนของภาษาจาวาที่คาดว่าจะพบนั้นได้สร้างโครงสร้างข้อมูลต้นไม้ของส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูลต่างๆ ซึ่งมอดูลเหล่านี้ก็คือส่วนที่ถูกนำมาสร้างเป็นคลาสนั่นเองดังนั้นจึงสามารถทราบชื่อคลาสที่นำมาใช้เป็นเงื่อนไขของการตัดส่วนจากชื่อคลาสที่สอดคล้องกับชื่อมอดูลที่ได้ระบุไว้ในขั้นตอนการสร้างโครงภาษาจาวา

เมื่อท่องโครงสร้างข้อมูลต้นไม้ของแต่ละมอดูลจากรากของโครงสร้างข้อมูลต้นไม้ลงมาที่ส่วนของโปรเจคชันโอเปอเรชัน จะทำให้ทราบตัวแปรภาษาจาวาที่สนใจ โดยตรวจสอบจากตัวแปรภาษาจาวาที่สอดคล้องกับโปรเจคชันโอเปอเรชันนั้น

จากนั้นท่องโครงสร้างข้อมูลต้นไม้ต่อลงไปยังลูกของโหนดของโปรเจคชันโอเปอเรชัน โดยไปยังโหนดที่ระบุพจน์ทางด้านซ้ายของเครื่องหมายเท่ากับเพื่อหาเมท็อดที่สอดคล้องกับโอเปอเรชันที่ประกาศไว้ในพจน์ทางด้านซ้ายมือของเครื่องหมายเท่ากับเพื่อแสดงเมท็อดที่สนใจ

เมื่อได้เงื่อนไขสำหรับการตัดส่วนโปรแกรมครบทั้งสามส่วนแล้วในขั้นตอนถัดไปคือการตัดส่วนโปรแกรมเพื่อแยกเอาเฉพาะส่วนที่สนใจในการตรวจสอบความก้าวหน้าของการพัฒนา

โปรแกรมต่อไป ทั้งนี้งานวิจัยนี้เลือกการตัดส่วนแบบสถิตเนื่องจากซอร์สโคดโปรแกรมยังไม่เสร็จสมบูรณ์ทำให้ไม่สามารถใช้วิธีการตัดส่วนแบบพลวัตได้

6.2 ขั้นตอนวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

ในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมต้องตัดส่วนโปรแกรมเพื่อให้เหลือเฉพาะส่วนที่ต้องตรวจสอบเสียก่อน ทั้งนี้การตัดส่วนโปรแกรมเพื่อตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมนี้เลือกใช้วิธีการตัดส่วนโปรแกรมสำหรับโปรแกรมที่พัฒนาเชิงวัตถุที่นำเสนอโดย Chen [18] โดยวิธีการนี้จะสร้างกราฟความสัมพันธ์เชิงวัตถุ และเลือกเอาเฉพาะจุดยอดที่เกี่ยวข้องกับเงื่อนไขการตัดส่วนโปรแกรมเพื่อนำมาตรวจสอบความก้าวหน้าต่อไป

ในขณะที่ส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบก็สามารถทราบได้จากการท่องไปยังลูกที่เป็นแสดงพจน์ทางด้านขวามือของเครื่องหมายเท่ากับของโหนดโปรเจคชันโอเปอเรชันซึ่งให้เงื่อนไขของการตัดส่วนโปรแกรม

6.2.1 การเปรียบเทียบส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบกับส่วนของโปรแกรมที่กำลังพัฒนา

ข้อความสั่งที่ได้จากการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบแสดงถึงลักษณะของส่วนของโปรแกรมที่กำลังพัฒนาดังนี้

6.2.1.1 ข้อความสั่งแทนตัวเอง

เมท็อดที่มีข้อความสั่งภาษาจาวาที่คาดว่าจะพบเป็นข้อความสั่งแทนตัวเอง เป็นเมท็อดที่ไม่มีการเปลี่ยนแปลงค่าของตัวแปรที่ปรากฏอยู่ในข้อความสั่งแทนตัวเอง นั่นคือ ถ้าเมท็อดที่ถูกตรวจสอบความก้าวหน้าไม่ได้เปลี่ยนแปลงค่าของตัวแปรที่ปรากฏในข้อความสั่งที่คาดว่าจะพบถือว่าได้พัฒนาโปรแกรมตามเอกสารการออกแบบนี้แล้ว

6.2.1.2 ข้อความสั่งกำหนดค่า

เมท็อดที่มีข้อความสั่งภาษาจาวาที่คาดว่าจะพบเป็นข้อความสั่งกำหนดค่า คือเมท็อดที่มีข้อความสั่งซึ่งเปลี่ยนแปลงค่าของตัวแปรทางด้านซ้ายของเครื่องหมายเท่ากับภายในข้อความสั่งด้วยค่าที่ประกาศทางด้านขวามือของเครื่องหมายเท่ากับภายในข้อความสั่งนั้น โดยตรวจสอบกับข้อความสั่งที่ตัดส่วนโปรแกรมมาแล้วว่ามีการกำหนดค่า (Defined value) ให้กับตัวแปรเดียวกันโดยใช้ค่า (Used value) ค่าเดียวกัน อย่างไรก็ดีการตรวจสอบความก้าวหน้าตรวจสอบเพียงการกำหนดค่าและใช้ค่าของข้อความสั่งในโปรแกรมที่ตัดส่วนแล้ว ว่าตรงกับที่ปรากฏในข้อความสั่งที่คาดว่าจะพบหรือไม่ โดยข้อความสั่งที่ปรากฏในโปรแกรมที่ตัดส่วนแล้วอาจให้ผลจากการทดสอบโปรแกรมไม่ตรงกับข้อความสั่งที่ได้จากส่วนของโปรแกรมภาษาที่คาดว่าจะพบก็ได้ แต่เมื่อ

โปรแกรมที่กำลังพัฒนาได้รับการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม ดังที่ได้กล่าวมานี้แล้ว ถือว่าได้พัฒนาโปรแกรมตามเอกสารการออกแบบนี้แล้ว

6.2.1.3 ข้อความสั่งเรียกใช้เมท็อด

เมท็อดที่มีข้อความสั่งภาษาจาวาที่คาดว่าจะพบเป็นข้อความสั่งเรียกใช้เมท็อด คือ เมท็อดที่มีการเรียกเมท็อดปรากฏในข้อความสั่งภาษาจาวาที่คาดว่าจะพบภายในเมท็อดนั้น หรือมีการเปลี่ยนแปลงค่าตัวแปรที่เก็บสถานะของวัตถุที่เรียกใช้เมท็อดนั้น ในทำนองเดียวกับเมท็อดที่ปรากฏในข้อความสั่งภาษาจาวาที่คาดว่าจะพบเปลี่ยนแปลงค่าตัวแปรนั้น

6.2.2 ขั้นตอนวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

การตรวจสอบความก้าวหน้าจะตรวจสอบทีละสมการในแต่ละมอดูลโดยวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมแสดงดังแสดงในรูปที่ 6.1

ผลลัพธ์ที่ได้จากการตรวจสอบความก้าวหน้าคือผลการตรวจสอบว่าแต่ละสมการได้รับการพัฒนาในซอร์สโคดภาษาจาวาเพื่อให้สอดคล้องกับส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบที่สอดคล้องกับสมการนั้น โดยใช้วิธีการที่นำเสนอในบทนี้

การพัฒนาโปรแกรมจะถือว่าเสร็จสมบูรณ์เมื่อทุกสมการมีส่วนของโปรแกรมที่พัฒนาแล้วสอดคล้องกับส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบของสมการนั้น ดังนั้นความก้าวหน้าของการพัฒนาโปรแกรมคืออัตราส่วนระหว่างจำนวนของสมการที่ได้รับการตรวจสอบว่าได้รับการพัฒนาแล้ว กับจำนวนสมการทั้งหมดที่ประกาศไว้ในข้อกำหนดรูปนัยคาเฟโอบีเจ

```
input axiom_declaration_tree; // the axiom_declaration_tree is module's expected source code tree
input source_code; // the developed source code which is need to track for progress
Begin
        for each axiom_equation_tree from axiom_declaration_tree do // get each equation tree form the axiom_declaration_tree
                java_variable := get Java variable name from of axiom_equation_tree;
                method_name := get method name from left side of axiom_equation_tree;
                expected_result := get expected result from right side axiom_equation_tree
                set_of_code := slicing source_code with java_variable and method_name criteria;
                if expected_result is a self-declared statement then
                        if set_of_code does not contain statements which contains java_variable as a defined attribute then
                                mark the axiom_equation_tree as a complete equation;
                        endif // end check set of code
                else //end checking a self-declared statement
                if expected_result is an assignment statement then
                        for each backward statement in set_of_code do // get each statement in a backward order
                                used_attribute := list of used attributes in backward statement;
                                def_attribute := list of defined attributes in backward statement;
                                if def_attribute contains same values as defined attributes in the expected_result then
                                        if all used_attribute contains same value as used attributes in the expected_result then
                                                mark the axiom_equation_tree as a complete equation;
                                        else // end if for all used_attributes
                                        if some used_attribute does not contains the value as used attributes defined in the expected_result then
                                                check_used_attribute_list := list of used_attribute which is not the same value as used attributes
                                                        defined in the expected_result;
                                                backward_used_attribute_list := list of used attributes in the statement;
                                                for each check_used_attribute in check_used_attribute_list do
                                                        for each check_attribute in backward_used_attribute_list do
                                                                defined_statement := a backward statement which the defined attribute contains
                                                                        check_attribute;
                                                                        end checking backward statement loop;
```

รูปที่ 6.1 ขั้นตอนวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

```
                        else
                                used_list := used attributes in defined_statement;
                                backward_used_attribute_list := backward_used_attribute_list appends
                                        used_list;
                        if check_used_attribute is contained in used attributes of defined_statement then
                                mark the check_used_attribute as a complete attribute;
                        endif // end check in defined statement
                    endfor// end check check_attribute
                endfor // end checking attribute in check_used_attribute_list
                if all attributes in check_used_attribute_list have been marked as a complete attribute then
                    mark the axiom_equation_tree as a complete equation;
                endif // check all attribute
            endif // if for some used value does not contains in use attribute of the backward statement
        endif // if for there are defined value in both backward statement and expected result
    endfor// checking backward statements
else // end an assignment statement
if expected_result is a method call statement then
    expected_caller := get the caller of the method call statement in the expected_result;
    expected_method := get the method called in the method call statement in the expected_result;
    expected_parameter := get the input parameter of the method call statement in the expected_result;
    for each backward statement in set_of_code do // get each statement in a backward order
        used_attribute := list of used attributes in this statement;
        def_attribute := list of defined attributes in this statement;
        if used_attribute contains expected_caller then
            method_name := get the method name which is call by expected_caller in statement;
            if method_name contains value and method_name is equal to expected_method then
                parameter_list := get parameters of method_name from this statement;
                if expected_parameter does not contain value and parameter_list does not contain value either then
                    mark the axiom_equation_tree as a complete equation;
                    end checking backward statement loop;
```

รูปที่ 6.1 ขั้นตอนวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม (ต่อ)

```
                    else // there is some expected paramteter check for the correct using parameter
                        if expected_parameter is same as parameter_list then
                            mark the axiom_equation_tree as a complete equation;
                            end checking backward statement loop;
                        else
                            unmatch_parameter_list := get the list of parameters which are not match with the
                                expected parameter;
                            for each check_param in unmatch_parameter_list do
                                defined_statement := a backward statement which the defined attribute
                                    contains check_param;
                                if check_param is contained in used attributes of defined_statement then
                                    mark the check_param as a complete parameter;
                                else
                                    used_list := used attributes in defined_statement;
                                    unmatch_parameter_list := unmatch_parameter_list appends
                                        used_list;
                                endif // end check in defined statement
                            endfor// end check check_attribute
                            if all parameter in unmatch_parameter_list have been marked as a complete
                                attribute then
                                mark the axiom_equation_tree as a complete equation;
                            endif // end check for all parameter
                        endif // check for correct parameter using
                    endif // check whether the expected parameter list contain value or not
                endif // check for method name
            endif // check for caller
        endfor // check backward statements
    endif // expected_result is an assignment statement
endfor // check each axiom declration
output list of axiom declaration
End.
```

รูปที่ 6.1 ขั้นตอนวิธีในการสร้างสมการสัจพจน์ (ต่อ)

บทที่ 7
การพัฒนาเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

เครื่องมือสำหรับการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมแบ่งออกเป็นสองส่วน คือเครื่องมือสร้างโครงภาษาจาวาและส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ และเครื่องมือสำหรับตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม ทั้งนี้เครื่องมือที่พัฒนาขึ้นจะทำงานตามวิธีการที่นำเสนอในงานวิจัยนี้

สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือทั้งสองเครื่องมือมีดังนี้

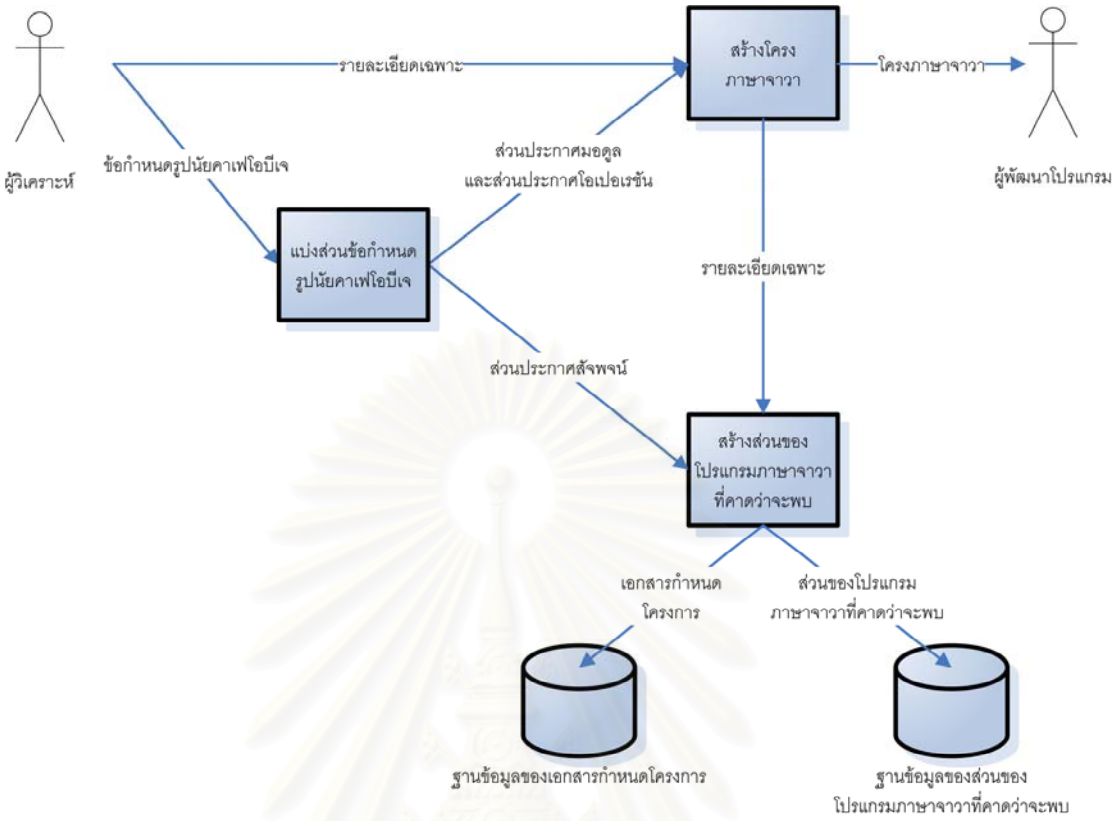
- ฮาร์ดแวร์
  - คอมพิวเตอร์พีซี Pentium 4 2.8 GHz
  - หน่วยความจำหลัก 1024 MB
  - ฮาร์ดดิสก์ความจุ 120 GB
- ซอฟต์แวร์
  - ระบบปฏิบัติการ ไมโครซอฟท์วินโดว์เอ็กพี
  - เครื่องมือพัฒนาโปรแกรมภาษาจาวาอีคลิปส์ (Eclipse IDE)

7.1 การพัฒนาเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบ

เครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบจะทำงานสองส่วนไปพร้อมๆ กัน นั่นคือการสร้างโครงภาษาจาวาเพื่อส่งให้ผู้พัฒนาพัฒนาโปรแกรมเพิ่มเติม และการสร้างส่วนของภาษาจาวาที่คาดว่าจะพบ

องค์ประกอบหลักของเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบแสดงในรูปที่ 7.1

เครื่องมือประกอบไปด้วยสองส่วนคือส่วนสร้างโครงภาษาจาวา และส่วนสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ โดยส่วนสร้างโครงภาษาจาวาจะรับข้อมูลจากส่วนประกาศมอดูล และส่วนประกาศโอเปอเรชันของข้อกำหนดรูปนัยคาเฟโอบีเจ และรับรายละเอียดเฉพาะจากผู้ใช้เครื่องมือซึ่งเป็นผู้วิเคราะห์ระบบ เพื่อสร้างโครงภาษาจาวาแล้วส่งให้ผู้พัฒนาต่อ ในขณะที่ส่วนสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบจะรับข้อมูลจากส่วนประกาศสัจพจน์ของข้อกำหนดรูปนัยคาเฟโอบีเจ และรายละเอียดเฉพาะ ซึ่งได้จากส่วนสร้างโครงภาษาจาวา เพื่อสร้าง

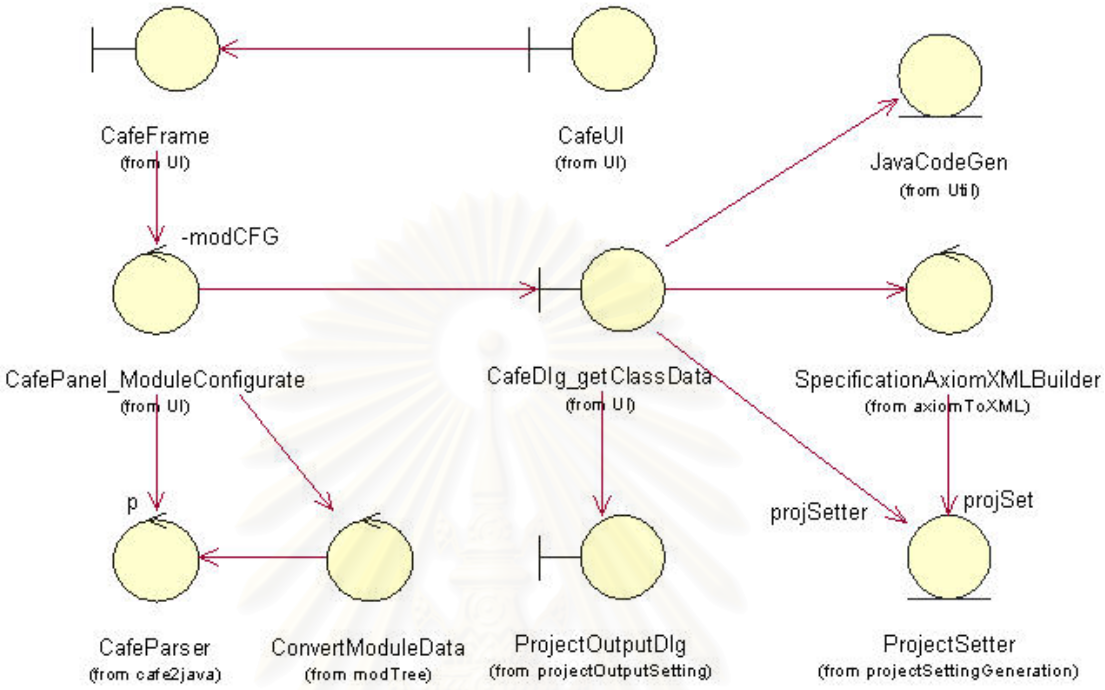


รูปที่ 7.1 องค์ประกอบหลักของเครื่องมือสร้างโครงภาษาจาวา
และส่วนของภาษาจาวาที่คาดว่าจะพบ

ส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ ส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบจะเก็บไว้ในรูปแบบเอกสารเอ็กเอ็มแอล ในขณะเดียวกันส่วนสร้างโปรแกรมภาษาจาวาที่คาดว่าจะพบจะสร้างเอกสารกำหนดโครงการเพื่อเก็บข้อมูลของการสร้างโครงภาษาจาวานี้เก็บไว้ในฐานข้อมูลด้วย

รูปที่ 7.2 แสดงแผนภาพคลาสของเครื่องมือสร้างโครงภาษาจาวา และส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ คลาสซึ่งแสดงในแผนภาพคลาสนี้ประกอบด้วย

- คลาส CafeUI เป็นคลาสหลักที่ควบคุมการทำงานของเครื่องมือนี้
- คลาส CafeFrame เป็นคลาสสำหรับควบคุมส่วนประสานผู้ใช้ทั้งหมดของเครื่องมือ
- คลาส CafePanel_ModuleConfigurate เป็นคลาสสำหรับแสดงรายละเอียดของข้อกำหนดรูปนัยคาเฟโอบีเจ โดยเมื่อคลาสนี้รับข้อกำหนดรูปนัยคาเฟโอบีเจแล้วจะนำไปแจงส่วน (parse) โดยวัตถุของคลาส CafeParser แล้วแสดงรายละเอียดของแต่ละมอดูลในข้อกำหนดซอฟต์แวร์นั้น
- คลาส CafeParser เป็นคลาสสำหรับแจงส่วนข้อกำหนดรูปนัยคาเฟโอบีเจที่รับเข้ามา



รูปที่ 7.2 แผนภาพคลาสของเครื่องมือสร้างโครงภาษาจาวา
และส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

- คลาส ConvertModuleData เป็นคลาสสำหรับรับข้อมูลจากวัตถุของคลาส CafeParser เพื่อแยกส่วนประกอบต่างๆ ของมอดูล เพื่อเตรียมข้อมูลสำหรับการสร้างโครงภาษาจาวา
- คลาส CafeDlg_getClassData เป็นคลาสสำหรับรับรายละเอียดเฉพาะจากผู้ใช้เครื่องมือ และรับข้อมูลของข้อกำหนดที่แจงส่วนแล้วจากวัตถุของคลาส CafePanel_ModuleConfigurate ทั้งนี้รายละเอียดเฉพาะที่รับจากผู้ใช้เครื่องมือต้องเป็นรายละเอียดเฉพาะที่เพิ่มเติมให้กับข้อกำหนดเท่านั้น
- คลาส ProjectOutputDlg เป็นคลาสสำหรับรับรายละเอียดเฉพาะที่เกี่ยวข้องกับโครงการสร้างโครงภาษาจาวานี้เช่น ตำแหน่งของไฟล์ที่ระบุโครงภาษาจาวา ตำแหน่งของไฟล์ผลลัพธ์จากการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ เป็นต้น
- คลาส JavaCodeGen เป็นคลาสสำหรับรับข้อมูลจากวัตถุของคลาส CafeDlg_getClassData เพื่อสร้างโครงภาษาจาวาและบันทึกไว้ในตำแหน่งที่ระบุไว้

- คลาส SpecificationAxiomXMLBuilder เป็นคลาสสำหรับสร้างส่วนของโปรแกรมจาวาที่คาดว่าจะพบโดยนำข้อมูลมาจากวัตถุของคลาส CafeDlg_getClassData
- คลาส ProjectSetter เป็นคลาสสำหรับบันทึกรายละเอียดของโครงการนี้อาทิเช่น ตำแหน่งของไฟล์ที่เก็บผลลัพธ์ของเครื่องมือ หรือรายละเอียดเฉพาะซึ่งผู้ใช้ระบุให้กับเครื่องมือ

ขั้นตอนการทำงานของเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบแสดงในแผนภาพกิจกรรม (Activity diagram) ในรูปที่ 7.3 โดยเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบ ในแต่ละขั้นตอนมีรายละเอียดดังนี้

7.1.1 ส่วนการระบุข้อมูลเฉพาะ ส่วนที่หนึ่ง

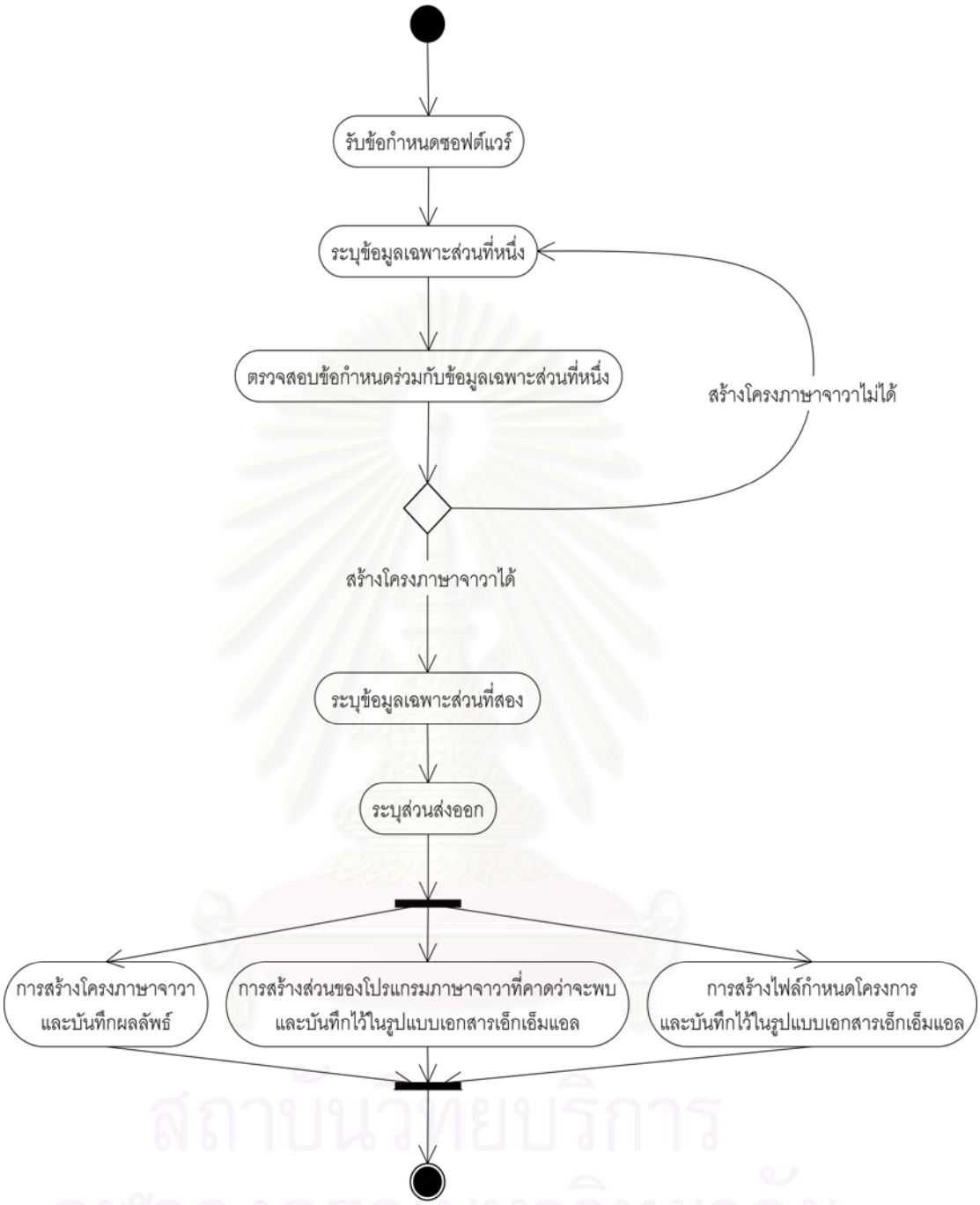
เครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบแบ่งรายละเอียดเฉพาะออกเป็น 2 ส่วน ส่วนแรกคือส่วนที่จำเป็นในขั้นตอนการเตรียมโครงสร้างข้อมูลสำหรับการสร้างโครงภาษาจาวาซึ่งอธิบายไว้ในหัวข้อ 4.3 และส่วนที่สองคือส่วนที่ใช้สำหรับการสร้างโครงภาษาจาวา รายละเอียดเฉพาะส่วนที่หนึ่งประกอบด้วย มอดูลที่ต้องการนำมาสร้างเป็นคลาส และชนิดของข้อมูลภาษาจาวาสำหรับแสดงชนิดข้อมูลสังเกตค่าได้ในข้อกำหนดรูปนัยคาเฟโอบีเจ หลังจากที่ผู้ใช้งานเลือกข้อกำหนดซอฟต์แวร์ที่ต้องการนำมาสร้างโครงภาษาจาวาแล้ว เครื่องมือจะตรวจสอบในข้อกำหนดซอฟต์แวร์ว่ามีการประกาศมอดูลใดบ้าง และมีการประกาศชนิดข้อมูลสังเกตค่าใดบ้าง เพื่อให้ผู้ออกแบบระบุมอดูลที่ต้องการสร้างคลาส และชนิดของข้อมูลของภาษาจาวาสำหรับแสดงชนิดข้อมูลสังเกตค่าได้

7.1.2 การตรวจสอบข้อกำหนดซอฟต์แวร์ร่วมกับข้อมูลเฉพาะส่วนที่หนึ่ง

ข้อมูลเฉพาะส่วนที่ 1 เป็นข้อมูลที่ใช้ในขั้นตอนการเตรียมโครงสร้างข้อมูลสำหรับการสร้างโครงภาษาจาวา ดังนั้นเมื่อระบุข้อมูลเฉพาะส่วนที่หนึ่ง แล้ว เครื่องมือจะเริ่มกระบวนการเตรียมโครงสร้างข้อมูลสำหรับสร้างโครงภาษาจาวา โดยสร้างกราฟความสัมพันธ์ระหว่างมอดูลของข้อกำหนดซอฟต์แวร์โดยวิธีการที่นำเสนอในหัวข้อ 4.3.1 จากนั้นลดรูปกราฟโดยรายละเอียดเฉพาะที่รับมาจากขั้นตอน 7.1.1 ด้วยวิธีการที่นำเสนอในหัวข้อ 4.3.2 แล้วตรวจสอบว่าแต่ละมอดูลที่ลดรูปแล้วมีชนิดข้อมูลแฝงหรือไม่ หากมีมอดูลหนึ่งมอดูลใดที่ไม่มีชนิดข้อมูลแฝง เครื่องมือจะแจ้งว่าไม่สามารถสร้างโครงภาษาจาวาได้ เนื่องจากไม่มีตัวแปรสำหรับเก็บสถานะของคลาสนั้น

7.1.3 การระบุข้อมูลเฉพาะส่วนที่สอง

หลังจากที่เครื่องมือตรวจสอบแล้วว่าสามารถสร้างโครงภาษาจาวาได้ เครื่องมือจะรับข้อมูลเฉพาะส่วนที่สองซึ่งประกอบด้วย



รูปที่ 7.3 แผนภาพกิจกรรมของเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบ

- ชื่อของตัวแปรที่เก็บสถานะ
- ชื่อและค่าของค่าคงที่
- ชื่อของเมท็อด และชื่อของตัวแปรส่วนนำเข้าของเมท็อดนั้น
- ชื่อของคลาส

ข้อมูลข้างต้นเหล่านี้จะนำมาใช้ร่วมกับกฎการสร้างโครงภาษาจาวา

7.1.4 การระบุส่วนส่งออก

ก่อนที่จะสร้างโครงภาษาจาวา และส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ ผู้ออกแบบจะต้องกำหนดตำแหน่งไฟล์ส่งออกของไฟล์โครงภาษาจาวาที่สร้างขึ้น ตำแหน่งไฟล์เก็บข้อมูลส่วนของโครงภาษาจาวาที่คาดว่าจะพบ และ ตำแหน่งไฟล์เอกสารกำหนดโครงการ

7.1.5 การสร้างโครงภาษาจาวา

ในขั้นตอนนี้จะนำกฎการสร้างโครงภาษาจาวาที่เสนอในหัวข้อ 4.4 มาใช้ร่วมกับข้อมูลเฉพาะทั้งส่วนที่หนึ่ง และส่วนที่สองโดยการสร้างโครงภาษาจาวาจะสร้างทีละโหนดในกราฟที่ลดรูปแล้ว จากนั้นจะตรวจสอบทีละโอเปอเรชันซึ่งประกาศไว้ในมอดูลว่าเข้ากับกฎข้อใดและสร้างผลลัพธ์ตามกฎนั้นแล้วบันทึกผลลัพธ์ไว้ในตำแหน่งที่ระบุ

ตัวอย่างโครงภาษาจาวาที่สร้างขึ้นจากข้อกำหนดซอฟต์แวร์ซึ่งแสดงในภาคผนวก ก แสดงในภาคผนวก ค

7.1.6 การสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

เมื่อสร้างโครงภาษาจาวาแล้ว เครื่องมือจะสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ โดยสร้างจากส่วนประกาศสัจพจน์ที่ประกาศไว้ภายในโหนดของกราฟที่ลดรูปแล้ว โดยใช้ขั้นตอนวิธีการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบซึ่งเสนอไว้ในหัวข้อ 5.2 เนื่องจากผลลัพธ์ที่ได้จากขั้นตอนนี้เป็นโครงสร้างข้อมูลต้นไม้ การพัฒนาเครื่องมือนี้ได้เลือกใช้เอกสารเอ็กเอ็มแอลเพื่อเก็บผลลัพธ์ที่ได้จากการสร้างส่วนของภาษาจาวาที่คาดว่าจะพบ โดยเอกสารเอ็กเอ็มแอลนี้มีอิลิเมนต์ต่างๆ ดังต่อไปนี้

- อิลิเมนต์ “Project” เป็นอิลิเมนต์ที่เป็นรากของเอกสารเอ็กเอ็มแอล
- อิลิเมนต์ “Module” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Project” อิลิเมนต์ “Module” เป็นอิลิเมนต์ที่มีลูกเป็นส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบทั้งหมดที่ได้จากสมการที่ประกาศไว้ในมอดูลนี้ โดยชื่อของมอดูลประกาศอยู่ในลักษณะประจำ “Name” และชื่อคลาสที่สอดคล้องกับมอดูลนี้ประกาศอยู่ในลักษณะประจำ “ClassName”
- อิลิเมนต์ “VARIABLE” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Module” อิลิเมนต์ “VARIABLE” เก็บชื่อ และชนิดข้อมูลของตัวแปรในข้อกำหนดรูปนัยคาเฟโอบีเจที่ประกาศไว้ในส่วนประกาศตัวแปรในส่วนประกาศสัจพจน์ของข้อกำหนดซอฟต์แวร์นี้
- อิลิเมนต์ “Proj_Operation” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Module” อิลิเมนต์ “Proj_Operation” แสดงโปรเจคชันโอเปอเรชันที่อยู่ในพจน์ทางด้านซ้ายมือของ

เครื่องหมายเท่ากับโดยเป็นรากของการรวมโครงสร้างข้อมูลต้นไม้ที่สร้างได้ โดยลักษณะประจำ “Operation” แสดงชื่อของโปรเจคชันโอเปอเรชันนี้ ลักษณะประจำ “arity” แสดงอาริทีของโปรเจคชันโอเปอเรชันนี้ และลักษณะประจำ “coarity” แสดงโคอาริทีของโปรเจคชันโอเปอเรชันนี้

- อิลิเมนต์ “Variable” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Proj_Operation” สำหรับอธิบายว่าตัวแปรของข้อกำหนดคาเฟโอบีเจ ซึ่งมีชนิดข้อมูลดังที่ระบุไว้ในอิลิเมนต์ “VARIABLE” แล้วนำชนิดข้อมูลนี้ไปตรวจสอบกับโปรเจคชันโอเปอเรชันว่าตรงกับตัวแปรใดในโปรแกรมภาษาจาวา อิลิเมนต์นี้มีลักษณะประจำ “Cafe_Variable” แสดงชื่อของตัวแปรของข้อกำหนดคาเฟโอบีเจ และลักษณะประจำ “Name” แสดงชื่อตัวแปรภาษาจาวาที่แสดงตัวแปรของข้อกำหนดคาเฟโอบีเจนี้
- อิลิเมนต์ “Equation” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Proj_Operation” โดยเป็นส่วนของสมการที่ต่อจากโปรเจคชันโอเปอเรชันที่แสดงอยู่ในอิลิเมนต์แม่ของอิลิเมนต์นี้ ประกอบด้วยพจน์ทางด้านซ้ายของเครื่องหมายเท่ากับและพจน์ทางด้านขวาของเครื่องหมายเท่ากับ อิลิเมนต์ “Equaiton” มีลักษณะประจำ “Name” แสดงส่วนของสมการนี้
- อิลิเมนต์ “Left_Equation” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Equation” แสดงพจน์ทางด้านซ้ายของเครื่องหมายเท่ากับของส่วนของสมการที่แสดงอยู่ในอิลิเมนต์แม่ โดยมีลักษณะประจำ “Name” แสดงส่วนของพจน์ทางด้านซ้ายของเครื่องหมายเท่ากับ ลักษณะประจำ “coarity” แสดงชนิดข้อมูลของผลลัพธ์ของรายการโอเปอเรชันในพจน์นี้ และ ลักษณะประจำ “order” บอกลำดับที่ของอิลิเมนต์นี้
- อิลิเมนต์ “Right_Equation” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Equation” ใช้สำหรับแสดงพจน์ทางด้านขวามือเมื่อสมการที่เป็นอิลิเมนต์แม่ของอิลิเมนต์นี้เป็นสมการแบบไม่มีเงื่อนไข โดยอิลิเมนต์ “Right_Equation” อาจมีหลายอิลิเมนต์ภายใต้อิลิเมนต์แม่เดียวกัน โดยแต่ละอิลิเมนต์จะถูกลำดับโดยลักษณะประจำ “order” เรียงลำดับจากน้อยไปมาก โดยในแต่ละลำดับจะแสดงขั้นตอนการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ โดยลักษณะประจำ “Name” แสดงพจน์ทางด้านขวามือที่กำลังพิจารณา ลักษณะประจำ “Hanging_Operation” แสดงส่วนที่เป็น “Hanging_Operator” ในขั้นตอนวิธีการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ ลักษณะประจำ “coarity” แสดงชนิดข้อมูลของผลลัพธ์ของรายการโอเปอเรชันที่อยู่ในลักษณะประจำ “Name” แสดงรายการโอเปอเรชันของอิลิเมนต์นี้ และ

ลักษณะประจำ “type” แสดงชนิดของพจน์ซึ่งแสดงในหัวข้อ 5.2 ของรายการโอเปอเรชันที่อยู่ในลักษณะประจำ “Name”

- อิลิเมนต์ “Guard_Equation” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Equation” ใช้สำหรับแสดงพจน์เงื่อนไขของสมการแบบมีเงื่อนไขซึ่งเป็นอิลิเมนต์แม่ของอิลิเมนต์นี้ โดยลักษณะประจำ “Name” แสดงพจน์เงื่อนไขที่อ้างถึง และลักษณะประจำ “ConditionOperand” แสดงเครื่องหมายเปรียบเทียบที่ใช้ในพจน์นี้
- อิลิเมนต์ “Left_Guard” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Guard_Equation” แสดงพจน์ที่อยู่ทางด้านซ้ายมือของเครื่องหมายเปรียบเทียบ โดยอิลิเมนต์ “Left_Guard” อาจมีหลายอิลิเมนต์ภายใต้อิลิเมนต์แม่เดียวกัน โดยแต่ละอิลิเมนต์จะถูกลำดับโดยลักษณะประจำ “order” เรียงลำดับจากน้อยไปมาก โดยในแต่ละลำดับจะแสดงขั้นตอนการสร้างส่วนของโปรแกรมจาวาที่คาดว่าจะพบในขั้นตอนการเปรียบเทียบ โดยลักษณะประจำของอิลิเมนต์นี้มีลักษณะเช่นเดียวกับอิลิเมนต์ “Right_Equation”
- อิลิเมนต์ “Right_Guard” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Guard_Equation” แสดงพจน์ที่อยู่ทางขวามือของเครื่องหมายเปรียบเทียบ โดยรายละเอียดต่างๆ ของอิลิเมนต์ “Right_Guard” จะเหมือนกับอิลิเมนต์ “Left_Guard”
- อิลิเมนต์ “Result_Guard” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Guard_Equation” แสดงพจน์ทางด้านขวามือของเครื่องหมายเท่ากับของสมการแบบมีเงื่อนไขที่เป็นอิลิเมนต์แม่ของอิลิเมนต์ “Guard_Equation” โดยรายละเอียดต่างๆ ของอิลิเมนต์ “Result_Guard” จะเหมือนกับอิลิเมนต์ “Left_Guard”
- อิลิเมนต์ “LeftOperation” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Right_Equation” อิลิเมนต์ “Left_Guard” อิลิเมนต์ “Right_Guard” และอิลิเมนต์ “Result_Guard” เป็นอิลิเมนต์ที่แสดงค่าของ “Left_Op” ซึ่งแสดงอยู่ในขั้นตอนวิธีการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ โดยมีลักษณะประจำ “Name” แสดงรายการโอเปอเรชันส่วนที่เป็น “Left_Op” และลักษณะประจำ “type” แสดงชนิดของรายการโอเปอเรชันนี้
- อิลิเมนต์ “Return” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Right_Equation” อิลิเมนต์ “Left_Guard” อิลิเมนต์ “Right_Guard” และอิลิเมนต์ “Result_Guard” เป็นอิลิเมนต์ที่แสดงส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ โดยส่วนของโปรแกรมจาวาที่คาดว่าจะพบแสดงในลักษณะประจำ “value”

โดยเอกสารเอ็กเอ็มแอลที่ใช้ในการเก็บผลลัพธ์มีรูปแบบตามที่กำหนดในเอกสารเอ็กเอ็มแอลสกีมาดังรูปที่ 7.4 ถึงรูปที่ 7.7

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="TypeLeftOperation">
    <!-- Attribute for left Operation -->
    <xsd:attribute name="Name"/>
    <xsd:attribute name="type">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="variable"/>
          <xsd:enumeration value="operation List"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <!-- Type for Left_Opeation -->
  <xsd:complexType name="Type_Left_Hand_Operation">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="leftOperation" type="TypeLeftOperation">
        <!-- End declaring attribute for left Operation -->
      </xsd:element>
    </xsd:sequence>
    <!-- Declare attribute for Left Equaiton-->
    <xsd:attribute name="Name" type="xsd:string"/>
    <xsd:attribute name="coarity" type="xsd:string"/>
    <xsd:attribute name="order" type="xsd:integer"/>
    <!-- End declare attribute for Left Equation -->
  </xsd:complexType>
```

รูปที่ 7.4 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “TypeLeftOperation” และ “Type_Left_Hand_Operation” ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้สร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

รูปที่ 7.4 แสดงการประกาศชนิดอิลิเมนต์ “TypeLeftOperation” และ “Type_Left_Hand_Operation” ชนิดข้อมูล “TypeLeftOperation” ใช้กำหนดชนิดของอิลิเมนต์ “Left_Operation” และชนิดข้อมูล “Type_Left_Hand_Operation” ใช้กำหนดชนิดของอิลิเมนต์ “Left_Equation”

รูปที่ 7.5 แสดงการประกาศชนิดอิลิเมนต์ “Type_Right_Hand_Operation” ซึ่งเป็นชนิดอิลิเมนต์ที่ใช้กำหนดอิลิเมนต์ “Right_Eqaution” “Left_Guard” “Right_Guard” และ “Result_Guard”

รูปที่ 7.6 แสดงการประกาศชนิดอิลิเมนต์ “TypeEquaiton” ซึ่งเป็นชนิดอิลิเมนต์ของอิลิเมนต์ “Equation”

```
<!-- Type for Right hand Equation -->
<xsd:complexType name="Type_Right_Hand_Operation">
  <xsd:sequence>
    <xsd:element minOccurs="0" name="leftOperation" type="TypeLeftOperation"/>
    <xsd:element minOccurs="0" name="Return">
      <xsd:complexType>
        <xsd:attribute name="value" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
  <xsd:attribute name="coarity" type="xsd:string"/>
  <xsd:attribute name="type">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="non_arity"/>
        <xsd:enumeration value="projectionOperation"/>
        <xsd:enumeration value="list_of_operation"/>
        <xsd:enumeration value="variable_list"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="order" type="xsd:integer"/>
  <xsd:attribute name="Hanging_Operation" type="xsd:string"/>
</xsd:complexType>
```

รูปที่ 7.5 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์
“Type_Right_Hand_Operation” ที่ใช้กำกับเอกสารเอ็กเอ็มแอล
ที่ใช้สร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

รูปที่ 7.7 แสดงการประกาศชนิดข้อมูล “Type_Projection_Operation” ซึ่งเป็นชนิดอิลิเมนต์ของอิลิเมนต์ “Projection_Operation” และอิลิเมนต์ “Project” ซึ่งเป็นรากของเอกสารเอ็กเอ็มแอลแสดงส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

เมื่อสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบครบถ้วนทุกสมการแล้วเอกสารเอ็กเอ็มแอลที่ได้จะถูกนำไปบันทึกในตำแหน่งที่ระบุไว้

ตัวอย่างเอกสารเอ็กเอ็มแอลของส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบที่สร้างจากข้อกำหนดการนับเลขที่มีสวิตซ์ซึ่งแสดงในภาคผนวก ก แสดงในภาคผนวก ฉ

7.1.7 การระบุไฟล์กำหนดโครงการ

การใช้เครื่องมือการสร้างโครงภาษาจาวา ผู้ออกแบบได้ระบุข้อมูลต่างๆ อาทิเช่นตำแหน่งของไฟล์ส่งออก และรายละเอียดเฉพาะต่างๆ ซึ่งรายละเอียดเหล่านี้ต้องนำไปใช้ในเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมจึงต้องสร้างไฟล์กำหนดโครงการเพื่อเก็บข้อมูลเหล่านี้สำหรับเครื่องมือตรวจสอบความก้าวหน้าต่อไป

```
<!-- Type for Equation element -->
<xsd:complexType name="TypeEquation">
  <xsd:sequence>
    <xsd:element maxOccurs="unbounded" name="Left_Equation"
type="Type_Left_Hand_Operation"/>
    <!-- End Left_Equaiton Element-->
    <xsd:choice>
      <xsd:element maxOccurs="unbounded" name="Right_Equation"
type="Type_Right_Hand_Operation"/>
      <!-- End Right_Equation Element-->
      <!-- Guard Element Use only when Condition Equation is set -->
      <xsd:element maxOccurs="unbounded" name="Guard_Equation">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element maxOccurs="unbounded" name="Left_Guard"
type="Type_Right_Hand_Operation"/>
            <xsd:element maxOccurs="unbounded"
              name="Right_Guard" type="Type_Right_Hand_Operation"/>
            <xsd:element maxOccurs="unbounded"
              name="Result_Guard" type="Type_Right_Hand_Operation"/>
          </xsd:sequence>
          <xsd:attribute name="Name" type="xsd:string" use="required"/>
          <xsd:attribute name="pathNo" type="xsd:integer"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:sequence>
  <!-- End Element in Equaiton node -->
  <xsd:attribute name="Name" type="xsd:string"/>
  <xsd:attribute default="Equation" name="type">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Equation"/>
        <xsd:enumeration value="Condition Equation"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>
<!-- End Declare all type in Element type -->
```

รูปที่ 7.6 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “TypeEquation” ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้สร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

รายละเอียดเหล่านี้จัดเก็บในรูปแบบเอกสารเอ็กเอ็มแอลซึ่งประกอบด้วยอิลิเมนต์ต่างๆ ดังนี้

- อิลิเมนต์ “Project” เป็นอิลิเมนต์รากของเอกสารกำหนดโครงการนี้
- อิลิเมนต์ “InputSpec” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Project” แสดงตำแหน่งของข้อกำหนดซอฟต์แวร์ซึ่งนำมาสร้างโครงภาษาจาวา โดยตำแหน่งของไฟล์ระบุในลักษณะประจำ “filename”

```
<!--Declare Projection operation type -->
<xsd:complexType name="Type_Projection_Operation">
  <xsd:sequence>
    <xsd:element name="Variable">
      <!-- Current java variable use for this projection operation-->
      <xsd:complexType>
        <xsd:attribute name="Name" type="xsd:string" use="required"/>
        <xsd:attribute name="Cafe_Variable" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element maxOccurs="unbounded" name="Equation"
      type="TypeEquation"/>
  </xsd:sequence>
  <!-- Delcare attributed for Proj_Operation -->
  <xsd:attribute name="Operation" type="xsd:string"/>
  <xsd:attribute name="arity" type="xsd:string"/>
  <xsd:attribute name="coarity" type="xsd:string"/>
</xsd:complexType>
<!--End Declare Projection op type -->
<xsd:element name="Project">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Module">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element maxOccurs="unbounded" name="VARIABLE">
              <xsd:complexType>
                <xsd:attribute name="Name" type="xsd:string" use="required"/>
                <xsd:attribute name="Sort" type="xsd:string" use="required"/>
              </xsd:complexType>
            </xsd:element>
            <xsd:element maxOccurs="unbounded"
              name="Proj_Operation" type="Type_Projection_Operation"/>
          </xsd:sequence>
          <!-- Declare Attribute for Module -->
          <xsd:attribute name="Name" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <!-- Declare Attribute for Project -->
    <xsd:attribute name="Name"/>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

รูปที่ 7.7 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศนิดอิลิเมนต์ “Type_Projection_Operation” และ อิลิเมนต์ “Project” ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้สร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

- อิลิเมนต์ “OutputSpecDir” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Project” แสดงตำแหน่งของเอกสารเอ็กเอ็มแอลระบุส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ โดยแยก

เป็นไฟล์ ไฟล์ละ 1 มอดูลที่ต้องการสร้างเป็นคลาส โดยตำแหน่งของไฟล์ระบุไว้ในอิลิเมนต์ลูกชื่อว่า “Path” โดยอิลิเมนต์ “Path” มีลักษณะประจำ “toDir” สำหรับบอกตำแหน่งของไฟล์

- อิลิเมนต์ “OutputSourceDir” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Project” แสดงตำแหน่งของโครงภาษาจาวาที่สร้างขึ้น โดยตำแหน่งของไฟล์ระบุไว้ในอิลิเมนต์ลูกชื่อว่า “Path” เช่นเดียวกับอิลิเมนต์ “OutputSpecDir”
- อิลิเมนต์ “ProjectSettingValue” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Project” แสดงรายละเอียดเฉพาะต่างๆ ที่ผู้ออกแบบระบุไว้
- อิลิเมนต์ “ProjOpSet” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ProjectSettingValue” แสดงรายการของโปรเจคชันโอเปอเรชัน
- อิลิเมนต์ “ProjectionOp” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ProjOpSet” แสดงรายละเอียดของโปรเจคชันโอเปอเรชันผ่านทางอิลิเมนต์ลูกได้แก่ อิลิเมนต์ “OpName” อิลิเมนต์ “ModuleName” อิลิเมนต์ “NumberOfInput” และอิลิเมนต์ “Arity” โดยอิลิเมนต์ “OpName” แสดงชื่อของโปรเจคชันโอเปอเรชันนี้ อิลิเมนต์ “ModuleName” แสดงชื่อมอดูลที่ประกาศโปรเจคชันโอเปอเรชันนี้ อิลิเมนต์ “NumberOfInput” แสดงจำนวนของอาริทีของโปรเจคชันโอเปอเรชันนี้ และอิลิเมนต์ “Arity” แสดงชนิดข้อมูลที่เป็นอาริทีของโปรเจคชันโอเปเรชันนี้
- อิลิเมนต์ “OpSet” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ProjectSettingValue” แสดงรายละเอียดของรายการของโอเปอเรชันที่ประกาศในข้อกำหนดซอฟต์แวร์
- อิลิเมนต์ “OperationDetail” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “OpSet” แสดงรายละเอียดต่างๆ ของแต่ละโอเปอเรชันผ่านทางอิลิเมนต์ลูกได้แก่ อิลิเมนต์ “OpName” อิลิเมนต์ “ModuleName” อิลิเมนต์ “NumberOfInput” และอิลิเมนต์ “Coarity” โดยอิลิเมนต์ “OpName” แสดงชื่อของโอเปอเรชันนี้ อิลิเมนต์ “ModuleName” แสดงชื่อมอดูลที่ประกาศโปรเจคชันโอเปอเรชันนี้ อิลิเมนต์ “NumberOfInput” แสดงจำนวนของอาริทีของโอเปอเรชันนี้ อิลิเมนต์ “Coarity” แสดงชนิดข้อมูลที่เป็นโคอาริทีของโอเปเรชันนี้ อิลิเมนต์ “MethodName” แสดงชื่อของเมท็อดซึ่งสอดคล้องกับโอเปอเรชันนี้ และ อิลิเมนต์ “Parameter” ซึ่งจะปรากฏเฉพาะโอเปอเรชันที่ต้องมีการกำหนดส่วนนำเข้าของโอเปอเรชันนั้น ทั้งนี้อิลิเมนต์ “Parameter” ประกอบด้วยลักษณะประจำ 2 ลักษณะประจำ ได้แก่ลักษณะประจำ “SpecName” และลักษณะประจำ “javaName” ลักษณะประจำ “SpecName” ซึ่ง

แสดงตัวแปรของข้อกำหนดรูปนัยคาเฟโอบีเจที่เป็นส่วนนำเข้าของโอเปอเรชันนี้ และลักษณะประจำ “javaName” ซึ่งแสดงตัวแปรของโปรแกรมภาษาจาวาที่เป็นส่วนนำเข้าของเมท็อดซึ่งสอดคล้องกับโอเปอเรชันนี้

- อิลิเมนต์ “VisibleSortMap” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ProjectSettingValue” แสดงรายการของชนิดของข้อมูลภาษาจาวาที่ถูกกำหนดให้แสดงชนิดข้อมูลสังเกตค่าได้ โดยอิลิเมนต์ลูกของอิลิเมนต์นี้คือ อิลิเมนต์ “vsMap” ซึ่งประกอบด้วยลักษณะประจำ “visibleSort” ซึ่งแสดงชนิดข้อมูลสังเกตค่าได้ซึ่งถูกแสดงโดยชนิดของข้อมูลภาษาจาวาซึ่งระบุไว้ในลักษณะประจำ “javaDataType”
- อิลิเมนต์ “VarNameMap” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ProjectSettingValue” แสดงรายการชื่อของตัวแปรที่เก็บสถานะภายในคลาสซึ่งสอดคล้องกับโปรเจคชันโอเปอเรชันที่ประกาศไว้ในมอดูล โดยอิลิเมนต์ลูกของอิลิเมนต์นี้คือ อิลิเมนต์ “varMap” แสดงชื่อตัวแปรภาษาจาวา ซึ่งแสดงในลักษณะประจำ “javaName” ที่สอดคล้องกับโปรเจคชันโอเปอเรชันที่ปรากฏในลักษณะประจำ “opName” ซึ่งมีอาริทีเป็นชนิดข้อมูลแฝงที่แสดงในลักษณะประจำ “hiddenSortName”
- อิลิเมนต์ “HiddenSortDecl” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ProjectSettingValue” แสดงรายการของชนิดข้อมูลแฝงซึ่งประกาศอยู่ในมอดูลต่างๆ โดยอิลิเมนต์ลูกของอิลิเมนต์นี้คือ อิลิเมนต์ “HiddenSortDecl” ประกอบด้วยลักษณะประจำ “hsName” แสดงชนิดข้อมูลแฝง และลักษณะประจำ “moduleName” แสดงมอดูลที่ประกาศชนิดข้อมูลแฝงนั้น
- อิลิเมนต์ “ClassNameMap” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ProjectSettingValue” แสดงรายการชื่อคลาสที่สอดคล้องกับชื่อมอดูลที่ต้องการสร้างคลาส โดยอิลิเมนต์ลูกของอิลิเมนต์นี้คือ อิลิเมนต์ “classNameMap” ประกอบด้วยลักษณะประจำ “moduleName” แสดงชื่อมอดูลที่ต้องการสร้างคลาส และลักษณะประจำ “className” แสดงชื่อคลาสที่สร้างจากมอดูลนี้
- อิลิเมนต์ “ImportList” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ProjectSettingValue” แสดงรายการการเรียกเข้ามอดูลทั้งหมดที่เกิดขึ้นในข้อกำหนดซอฟต์แวร์นี้
- อิลิเมนต์ “Import” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “ImportList” แสดงการเรียกเข้ามอดูล โดยลักษณะประจำ “based” แสดงชื่อมอดูลที่เป็นมอดูลหลัก โดยอิลิเมนต์ลูกของอิลิเมนต์นี้คือ อิลิเมนต์ “From” มีลักษณะประจำ “input” แสดงชื่อมอดูลที่ถูกเรียกเข้ามอดูลหลัก

- อิลิเมนต์ “IncludeModule” แสดงรายการมอดูลที่ถูกลดรูปในขั้นตอนการลดรูปกราฟ โดยอิลิเมนต์ลูกของอิลิเมนต์นี้คืออิลิเมนต์ “Include” ประกอบด้วยลักษณะประจำ 2 ลักษณะ ได้แก่ ลักษณะประจำ “based” แสดงชื่อมอดูลที่เป็นหลัก และลักษณะประจำ “include” แสดงมอดูลที่ถูกลดรูปเข้ากับมอดูลหลัก

เอกสารเอ็กเอ็มแอลสำหรับเก็บข้อมูลของการตรวจสอบข้อมูลนี้ถูกกำหนดโดยเอ็กเอ็มแอลสกีมาดังนี้

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
   <xs:element name="Project" type="ProjectType">
      <xs:annotation>
         <xs:documentation>root of Document</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:element name="HiddenSortDecl">
      <xs:annotation>
         <xs:documentation>set of hidden sort and the module which declare
them</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element name="HiddenSortDecl" minOccurs="0"
maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attribute name="hsName" type="xs:string"/>
                  <xs:attribute name="moduleName" type="xs:string"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
```

รูปที่ 7.8 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศอิลิเมนต์ “Project” และ “HiddenSortDecl” ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ

รูปที่ 7.8 แสดงเอกสารเอ็กเอ็มสกีมาซึ่งกำกับเอกสารเอ็กแอลที่ใช้กำหนดโครงการ โดยประกาศอิลิเมนต์ “Project” ซึ่งเป็นอิลิเมนต์รากของเอกสารเอ็กเอ็มแอล และอิลิเมนต์ “HiddenSortDecl” ซึ่งเป็นชนิดข้อมูลของอิลิเมนต์ “HiddenSortDecl”

รูปที่ 7.9 แสดงเอกสารเอ็กเอ็มแอลสกีมาซึ่งประกาศอิลิเมนต์ “ModuleName” “NumberOfInput” “OpName” และ “Path” โดยอิลิเมนต์ “ModuleName” อิลิเมนต์ “NumberOfInput” และ อิลิเมนต์ “OpName” เป็นอิลิเมนต์ลูกของ อิลิเมนต์ “ProjectionOp” และ

```
    <xs:element name="ModuleName">
      <xs:annotation>
        <xs:documentation>module name which stores this operation</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="NumberOfInput">
      <xs:annotation>
        <xs:documentation>amount of number of input of the
operation</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="OpName">
      <xs:annotation>
        <xs:documentation>operation name</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="Path">
      <xs:annotation>
        <xs:documentation>the directory path which the file is stored</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="toDir" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
```

รูปที่ 7.9 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศอิลิเมนต์ “ModuleName” “NumberOfInput” “OpName” และ “Path” ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ

อิลเมนต์ “OpName” ในขณะที่อิลิเมนต์ “Path” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “OutputSpecDir” และ อิลิเมนต์ “OutputSrcDir”

รูปที่ 7.10 ประกาศชนิดอิลิเมนต์ “ProjectType” ซึ่งเป็นชนิดอิลิเมนต์ของอิลิเมนต์ “Project” ซึ่งเป็นอิลิเมนต์รากของเอกสารเอ็กเอ็มแอลนี้

รูปที่ 7.11 แสดงการประกาศชนิดอิลิเมนต์ “VisibleSortMapType“ และ “VarNameMapType“โดย ชนิดอิลิเมนต์ “VisibleSortMapType“ เป็นชนิดอิลิเมนต์ของอิลิเมนต์ “VisibleSortMap” ในขณะที่ชนิดอิลิเมนต์ “VarNameType“ เป็นชนิดอิลิเมนต์ของอิลิเมนต์ “VarnameMap”

```
<xs:complexType name="ProjectType">
  <xs:annotation>
    <xs:documentation>the type of the root of document</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="InputSpec">
      <xs:annotation>
        <xs:documentation>the input fomal specification file</xs:documentation>
      </xs:annotation>
        <xs:complexType>
          <xs:attribute name="filename" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="OutputSpecDir">
      <xs:annotation>
        <xs:documentation>Output expected XML document generated from the
input specificiation</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Path" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="OutputSourceDir">
      <xs:annotation>
        <xs:documentation>Output Java source code which has been generated from
the Cafe2Java tool.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Path" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="ProjectSettingValue" type="ProjectSettingValueType">
      <xs:annotation>
        <xs:documentation>Project setting values which has been collected from user
when the user used the Cafe2Java tool</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

รูปที่ 7.10 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “ProjectType” ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ

รูปที่ 7.12 แสดงการประกาศชนิดอิลิเมนต์ “OpSetType” ซึ่งเป็นชนิดอิลิเมนต์ของอิลิเมนต์ ”OpSet”

```
<xs:complexType name="VisibleSortMapType">
  <xs:sequence>
    <xs:element name="vsMap" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="visibleSort" type="xs:string" use="required"/>
        <xs:attribute name="javaDataType" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="VarNameMapType">
  <xs:sequence>
    <xs:element name="varMap" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>the Java variable name will be matched with the
operation name. some hidden sort may have one or more projection
operation</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="opName" type="xs:string" use="required"/>
        <xs:attribute name="hiddenSortName" type="xs:string" use="required"/>
        <xs:attribute name="javaName" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

รูปที่ 7.11 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “VisibleSortMapType” และ “VarNameMapType” ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ

รูปที่ 7.13 แสดงการประกาศชนิดอิลิเมนต์ “ConstantValueType“ และ “ImportListType“ ชนิดอิลิเมนต์ “ConstantValueType“ คือชนิดอิลิเมนต์ของอิลิเมนต์ “ConstValue” ในขณะที่ชนิดอิลิเมนต์ “ImportListType“ คือชนิดอิลิเมนต์ของอิลิเมนต์ “ImportList”

รูปที่ 7.14 แสดงการประกาศชนิดอิลิเมนต์ “ProjectOpSetType“ ซึ่งเป็นชนิดอิลิเมนต์ของอิลิเมนต์ “ProjOpSet”

รูปที่ 7.15 แสดงการประกาศชนิดอิลิเมนต์ “ProjectSettingValueType” ซึ่งเป็นชนิดอิลิเมนต์ของอิลิเมนต์ “ProjectSettingValue”

หลังจากสร้างเอกสารกำหนดโครงการแล้ว เอกสารกำหนดโครงการจะถูกบันทึกไว้ในตำแหน่งที่ระบุไว้ ตัวอย่างของเอกสารกำหนดโครงการที่ใช้ในการทดลองแสดงในภาคผนวก ช

```
<xs:complexType name="OpSetType">
  <xs:sequence>
    <xs:element name="OperationDetail" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>detail of each operation</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="OpName"/>
          <xs:element ref="ModuleName"/>
          <xs:element ref="NumberOfInput"/>
          <xs:element name="Coarity">
            <xs:annotation>
              <xs:documentation>coarity name</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="name" type="xs:string" use="required"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="MethodName">
            <xs:annotation>
              <xs:documentation>module name which is set to corresponce to the
operaation name</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="name" type="xs:string" use="required"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="Parameter" minOccurs="0"
maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>parameter which the method in Java code may
contains. The parameter stores in pair of the variable in CafeOBJ speicification and the
input parameter in java source code</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="specName" type="xs:string" use="required"/>
              <xs:attribute name="javaName" type="xs:string" use="required"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

รูปที่ 7.12 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “OpSetType“ ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ

```
<xs:complexType name="ConstantValueType">
  <xs:sequence>
    <xs:element name="Constant" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="constOpName" type="xs:string" use="required"/>
        <xs:attribute name="sort" type="xs:string" use="required"/>
        <xs:attribute name="javaValue" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ImportListType">
  <xs:sequence>
    <xs:element name="Import" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>the module which is interested</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="From" maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>the module which has been imported by the
interested module</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="input" type="xs:string" use="required"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="based" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

รูปที่ 7.13 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “ConstantValueType“ และ “ImportListType“ ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ

7.2 การพัฒนาเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

องค์ประกอบหลักของเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมแสดงในรูปที่ 7.16

เครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมจะรับข้อมูลจากผู้ใช้เครื่องมือว่าต้องการตรวจสอบความก้าวหน้าของการพัฒนาโครงการใด และรับโปรแกรมภาษาจาวาที่พัฒนาแล้ว โดยนำข้อมูลรายละเอียดเฉพาะ และส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบซึ่งบันทึกไว้ในฐานข้อมูลมาใช้ในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

```
<xs:complexType name="ProjOpSetType">
  <xs:sequence>
    <xs:element name="ProjectionOp" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>each projection operation</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="OpName"/>
          <xs:element ref="ModuleName"/>
          <xs:element ref="NumberOfInput"/>
          <xs:element name="Arity">
            <xs:annotation>
              <xs:documentation>arity name of the operation</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="name" type="xs:string" use="required"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

รูปที่ 7.14 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “ProjectOpSetType“ ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ

รูปที่ 7.17 แสดงแผนภาพคลาสของเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม คลาสซึ่งแสดงในแผนภาพคลาสนี้ประกอบด้วย

- คลาส MainUIDialog เป็นคลาสหลักซึ่งทำหน้าที่ควบคุมการทำงานทั้งหมดของเครื่องมือนี้ และเป็นส่วนประสานผู้ใช้ของเครื่องมือนี้
- คลาส CompareProgress เป็นคลาสสำหรับตรวจสอบความก้าวหน้าของทุกๆ มอดูลในข้อกำหนดโดยจะเรียกใช้วัตถุของคลาส SingleModuleCompareProgress
- คลาส ProjectSetter เป็นคลาสสำหรับบันทึกรายละเอียดของโครงการนี้อาทิเช่น ตำแหน่งของไฟล์ที่เก็บผลลัพธ์ของเครื่องมือ หรือรายละเอียดเฉพาะซึ่งผู้ใช้ระบุให้กับเครื่องมือสร้างโครงภาษาจาวาและส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ
- คลาส SingleModuleCompareProgress เป็นคลาสสำหรับตรวจสอบความก้าวหน้าของแต่ละมอดูลในข้อกำหนดซอฟต์แวร์นั้น
- คลาส ResultExtractor เป็นคลาสสำหรับตรวจสอบผลลัพธ์ที่ได้จากวัตถุของคลาส SingleModuleCompareProgress เพื่อนำผลลัพธ์มาแสดงในส่วนประสานผู้ใช้

```
<xs:complexType name="ProjectSettingValueType">
  <xs:sequence>
    <xs:element name="ProjOpSet" type="ProjOpSetType">
      <xs:annotation>
        <xs:documentation>set of projection operations</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="OpSet" type="OpSetType">
      <xs:annotation>
        <xs:documentation>set of all operations include  set of projection
operations.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="VisibleSortMap" type="VisibleSortMapType">
      <xs:annotation>
        <xs:documentation>set of visible sort which map  with Java data type, set by
the user of Cafe2Java</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="VarNameMap" type="VarNameMapType">
      <xs:annotation>
        <xs:documentation>set of the projection operation and the Java variable
which is correspond to the operation</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="ConstantValue" type="ConstantValueType">
      <xs:annotation>
        <xs:documentation>set of constant value from the non arity
operation</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element ref="HiddenSortDecl"/>
    <xs:element name="ClassNameMap">
      <xs:annotation>
        <xs:documentation>set of the module name which map with its related class
name</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="classNameMap" maxOccurs="unbounded">
            <xs:complexType>
              <xs:attribute name="moduleName" type="xs:string" use="required"/>
              <xs:attribute name="className" type="xs:string" use="required"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
```

รูปที่ 7.15 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “ProjectSettingValueType“ ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ

```
        <xs:element name="ImportList" type="ImportListType">
          <xs:annotation>
            <xs:documentation>List of imported module</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="IncludeModule">
          <xs:annotation>
            <xs:documentation>the list of module which have to be embed to its base
module (the ungenerated module)</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Include" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>pair of include module , the module which will be
embed to the based module</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:attribute name="based" type="xs:string" use="required"/>
                  <xs:attribute name="include" type="xs:string" use="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
</xs:schema>
```

รูปที่ 7.15 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “ProjectSettingValueType“ ที่ใช้กำกับเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการ (ต่อ)

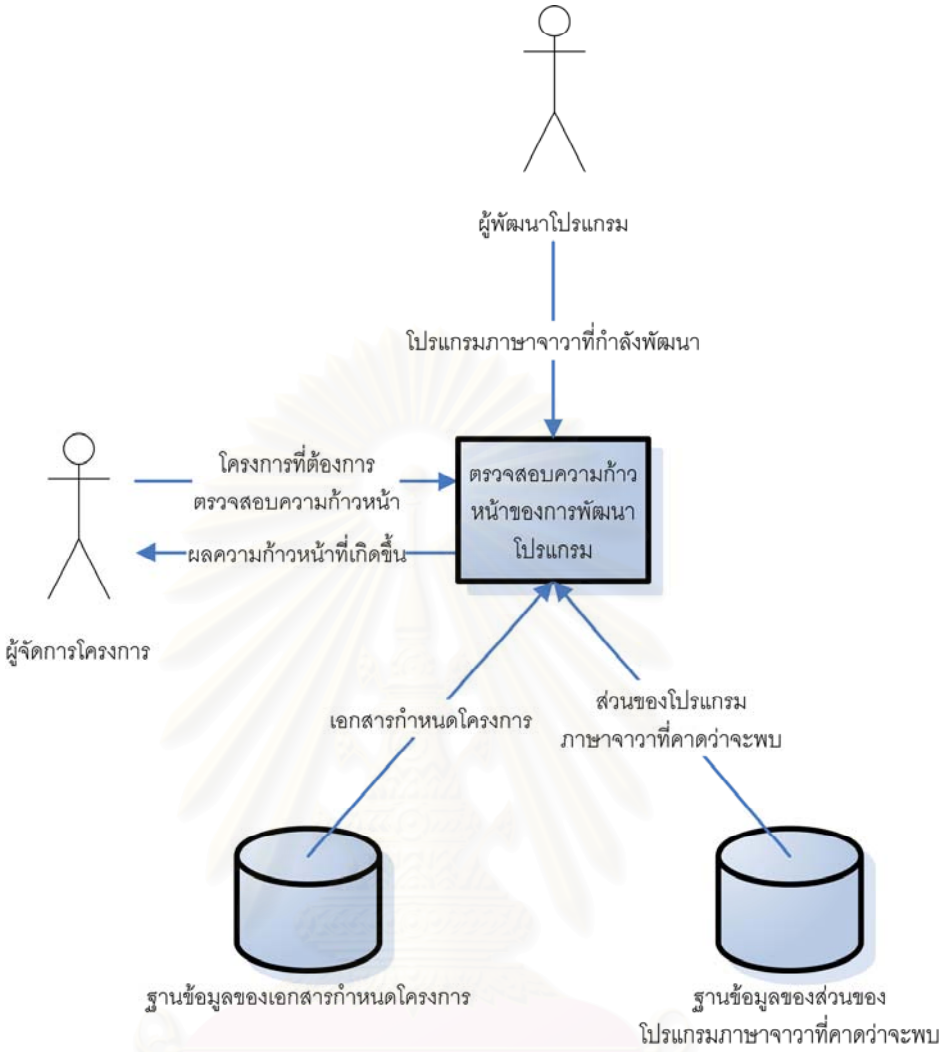
เครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมมีขั้นตอนการทำงานแสดงในแผนภาพกิจกรรมซึ่งแสดงในรูปที่ 7.18 โดยเครื่องมือตรวจสอบความก้าวหน้าในแต่ละขั้นตอนมีรายละเอียดดังนี้

7.2.1 การรับไฟล์กำหนดโครงการ

เครื่องมือตรวจสอบความก้าวหน้าต้องรับไฟล์กำหนดโครงการเพื่อระบุรายละเอียดต่างๆ ที่ใช้ในการตรวจสอบความก้าวหน้า อาทิตำแหน่งของไฟล์ของซอร์สโคดโปรแกรม ตำแหน่งของไฟล์ที่ระบุส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ และรายละเอียดเฉพาะต่างๆ ที่ผู้ออกแบบระบุเอาไว้ เพื่อนำมาตรวจสอบความก้าวหน้าที่เกิดขึ้น

7.2.2 การสร้างกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ

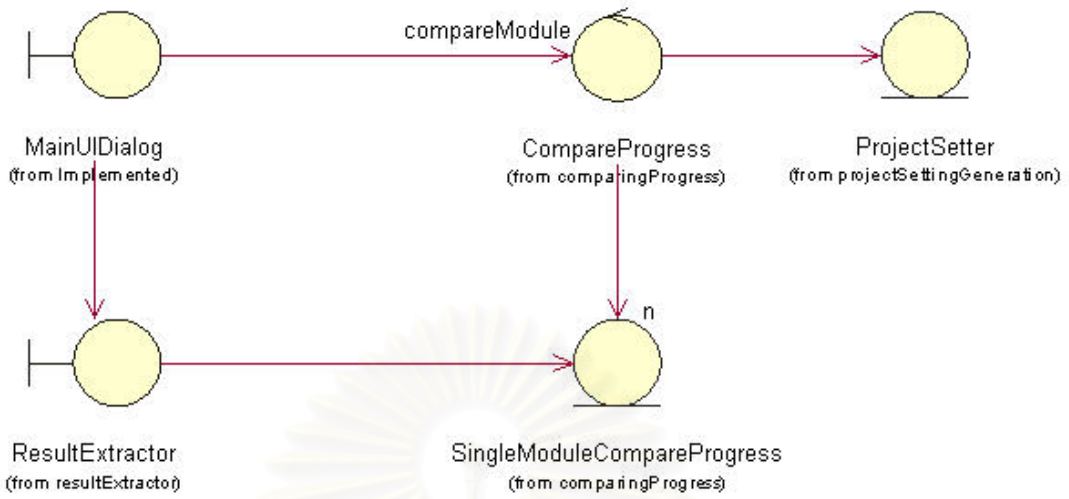
เมื่อเครื่องมือได้รับซอร์สโคดโปรแกรมที่พัฒนาแล้วจากผู้พัฒนา เครื่องมือจะสร้างกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุซึ่งแสดงไว้ในหัวข้อ 2.2.8 เพื่อเตรียมไว้ใช้ในขั้นตอนการ



รูปที่ 7.16 องค์ประกอบหลักของเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

ตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม โดยเครื่องมือเลือกใช้เอกสารเอ็กเอ็มแอลเพื่อแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ โดยเอกสารเอ็กเอ็มแอลซึ่งใช้แสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุประกอบไปด้วย

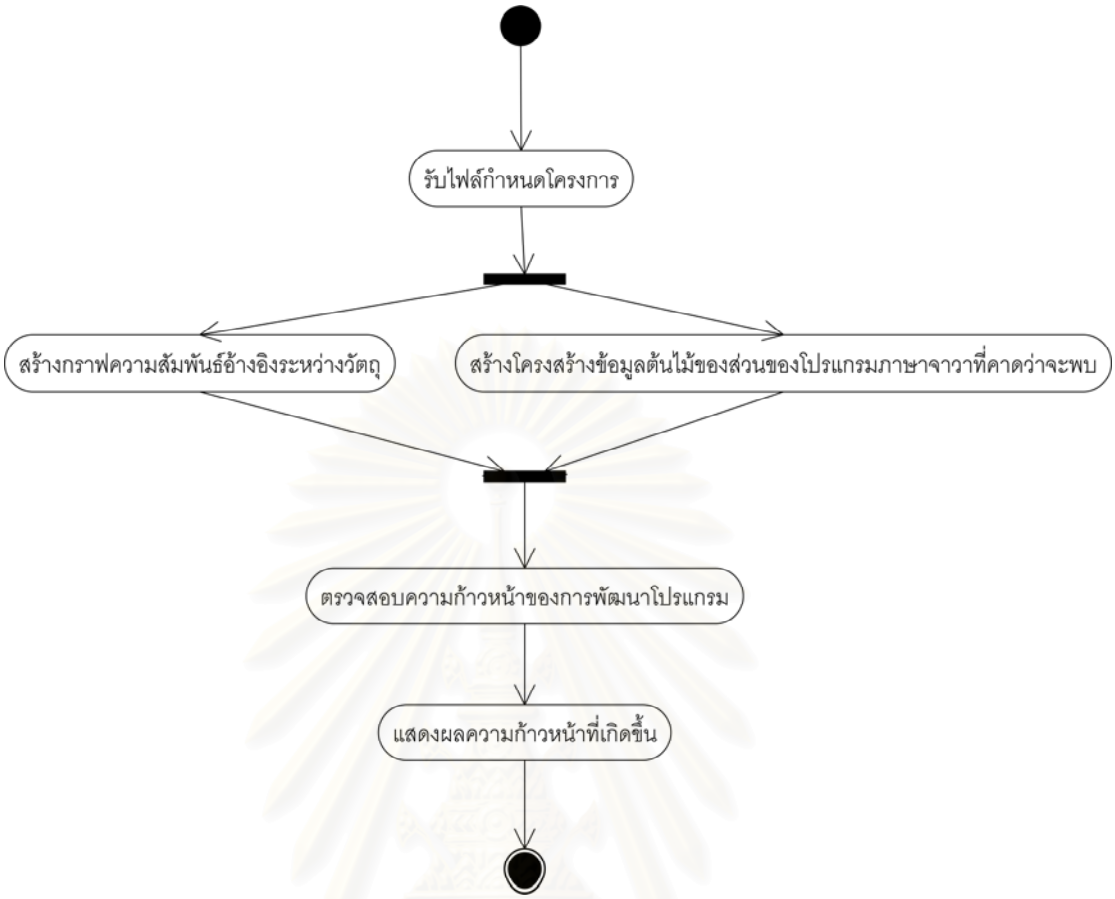
- อิลิเมนต์ “Project” ซึ่งเป็นอิลิเมนต์รากของเอกสารเอ็กเอ็มแอลซึ่งใช้แสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ
- อิลิเมนต์ “Class” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Project” อิลิเมนต์ “Class” แสดงข้อมูลของคลาสซึ่งอยู่ในซอร์สโคดโปรแกรมที่กำลังพัฒนา โดยในอิลิเมนต์ “Project” สามารถมีอิลิเมนต์ “Class” ได้หลายอิลิเมนต์ อิลิเมนต์ “Class” ประกอบด้วยลักษณะประจำ “name” แสดงชื่อของคลาสนี้ ลักษณะประจำ “modifier” แสดงลักษณะการดัดแปร (modifier) ของคลาส และลักษณะประจำ “cid” แสดงลำดับ



รูปที่ 7.17 แผนภาพคลาสของเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

ของอิลิเมนต์ “Class” ในเอกสารเอ็กเอ็มแอลนี้ อิลิเมนต์ “Class” แสดงจุดยอดคลาสในกราฟแสดงความสัมพันธ์อ้างอิงระหว่างวัตถุ

- อิลิเมนต์ “attribute” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Class” อิลิเมนต์ “attribute” แสดงลักษณะประจำที่ประกาศอยู่ภายในอิลิเมนต์ “Class” ซึ่งเป็นอิลิเมนต์แม่ของอิลิเมนต์นี้โดยอิลิเมนต์ “Class” 1 อิลิเมนต์สามารถมีอิลิเมนต์ “attribute” ได้มากกว่า 1 อิลิเมนต์ อิลิเมนต์ “attribute” ประกอบด้วยลักษณะประจำ “name” แสดงชื่อของลักษณะประจำนี้ ลักษณะประจำ “modifier” แสดงลักษณะการดัดแปรของลักษณะประจำนี้ ลักษณะประจำ “type” แสดงชนิดของข้อมูลของภาษาจาวาของลักษณะประจำนี้ ลักษณะประจำ “atid” แสดงลำดับของอิลิเมนต์ “attribute” ในเอกสารเอ็กเอ็มแอลนี้ และลักษณะประจำ “value” ซึ่งอาจมีหรือไม่มีลักษณะประจำนี้ก็ได้ ลักษณะประจำ “value” แสดงค่าที่ถูกกำหนดให้กับลักษณะประจำที่แสดงโดยอิลิเมนต์นี้ถ้าลักษณะประจำนี้ถูกกำหนดค่าในซอร์สโคดโปรแกรมภาษาจาวาที่กำลังพัฒนา อิลิเมนต์ “attribute” แสดงจุดยอดลักษณะประจำในกราฟแสดงความสัมพันธ์อ้างอิงระหว่างวัตถุ
- อิลิเมนต์ “method” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Class” อิลิเมนต์ “method” แสดงเมท็อดที่ประกาศอยู่ภายในอิลิเมนต์ “Class” ซึ่งเป็นอิลิเมนต์แม่ของอิลิเมนต์นี้ โดยอิลิเมนต์ “Class” 1 อิลิเมนต์สามารถมีอิลิเมนต์ “method” ได้มากกว่า 1 อิลิเมนต์ อิลิเมนต์ “method” ประกอบด้วย ลักษณะประจำ “name” แสดงชื่อของเมท็อดนี้



รูปที่ 7.18 แผนภาพกิจกรรมแสดงการทำงานของเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

ลักษณะประจำ “modifier” แสดงลักษณะการดัดแปรของลักษณะประจำนี้ ลักษณะประจำ “return” แสดงชนิดของข้อมูลของภาษาจาวาของส่วนส่งออกของเมท็อดนี้ และลักษณะประจำ “Eid” แสดงลำดับของอิลิเมนต์ “method” ในเอกสารเอ็กเอ็มแอลนี้ อิลิเมนต์ “method” แสดงจุดยอดเมท็อดในกราฟแสดงความสัมพันธ์อ้างอิงระหว่างวัตถุ

- อิลิเมนต์ “parameter” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “method” ซึ่งปรากฏเมื่อเมท็อดนี้มีส่วนนำเข้า โดยอิลิเมนต์ “method” 1 อิลิเมนต์สามารถมีอิลิเมนต์ “statement” ได้มากกว่า 1 อิลิเมนต์ อิลิเมนต์ “parameter” ประกอบด้วย ลักษณะประจำ “name” แสดงชื่อของส่วนนำเข้า ลักษณะประจำ “type” แสดงชนิดของข้อมูลภาษาจาวาของส่วนนำเข้านี้ และลักษณะประจำ “FI” แสดงลำดับของอิลิเมนต์ “parameter” ในเอกสารเอ็กเอ็มแอลนี้ อิลิเมนต์ “parameter” แสดงจุดยอดส่วนนำเข้าในกราฟแสดงความสัมพันธ์อ้างอิงระหว่างวัตถุ

- อิลิเมนต์ “statement” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “method” อิลิเมนต์ “statement” แสดงข้อความสั่งที่ประกาศอยู่ภายในอิลิเมนต์ “method” ซึ่งเป็นอิลิเมนต์แม่ของอิลิเมนต์นี้ โดยอิลิเมนต์ “method” 1 อิลิเมนต์สามารถมีอิลิเมนต์ “statement” ได้มากกว่า 1 อิลิเมนต์ อิลิเมนต์ “statement” ประกอบด้วย ลักษณะประจำ “name” แสดงข้อความสั่งที่แสดงอยู่ในอิลิเมนต์นี้ และลักษณะประจำ “sid” แสดงลำดับของข้อความสั่งนี้ในกรณีที่เป็นข้อความสั่งธรรมดา หรือลักษณะประจำ “FOid” แสดงลำดับของข้อความสั่งส่วนส่งออกในกรณีที่ข้อความสั่งนี้คือค่าออกจากเมท็อด ซึ่งค่าที่ปรากฏในลักษณะประจำนี้จะต้องเป็นค่าเดียวกับค่าที่ระบุไว้ในลักษณะประจำ “Eid” ของอิลิเมนต์แม่ “method” นอกจากนี้หากข้อความสั่งนั้น เป็นข้อความสั่งเชิงเงื่อนไข (condition statement) ถ้า-แล้ว (if-then-else) จะต้องเพิ่มลักษณะประจำเพื่ออธิบายอิลิเมนต์นี้เพิ่มเติม ได้แก่ อิลิเมนต์ “trueStart” อิลิเมนต์ “tureEnd” อิลิเมนต์ “falseStart” และอิลิเมนต์ “falseEnd” โดยที่ลักษณะประจำ “trueStart” แสดงลำดับของข้อความสั่งที่จะเริ่มทำงานถ้าข้อความเงื่อนไขเป็นจริง ลักษณะประจำ”trueEnd” แสดงลำดับของข้อความสั่งสุดท้ายที่ทำงานถ้าข้อความเงื่อนไขเป็นจริง ลักษณะประจำ “falseStart” แสดงลำดับของข้อความสั่งที่จะเริ่มทำงานถ้าข้อความเงื่อนไขเป็นเท็จ ลักษณะประจำ”falseEnd” แสดงลำดับของข้อความสั่งสุดท้ายที่ทำงานถ้าข้อความเงื่อนไขเป็นเท็จ ซึ่งลักษณะประจำ “falseStart” และ “falseEnd” อาจจะไม่มีก็ได้ถ้าข้อความสั่งเชิงเงื่อนไขนั้นไม่มีคำสำคัญ “else” และสำหรับข้อความสั่งที่อยู่ใต้ข้อความสั่งเชิงเงื่อนไขจะมีลักษณะประจำ “underCondition” แสดงข้อความสั่งเงื่อนไขที่ทำให้ข้อความสั่งนี้ถูกเรียกใช้ เพื่อระบุว่าข้อความสั่งนี้จะถูกทำงานเมื่ออยู่ใต้เงื่อนไขใด อิลิเมนต์ “statement” ซึ่งแสดงลักษณะประจำ “sid” แสดงจุดยอดข้อความสั่ง ในขณะที่ อิลิเมนต์ “statement” ซึ่งแสดงลักษณะประจำ “FO” แสดงจุดยอดส่วนส่งออกในกราฟแสดงความสัมพันธ์อ้างอิงระหว่างวัตถุ
- อิลิเมนต์ “Used” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “statement” แสดงตัวแปรที่ถูกใช้ในข้อความสั่งที่แสดงในอิลิเมนต์ “statement” ซึ่งเป็นอิลิเมนต์แม่ของอิลิเมนต์นี้ โดยมีอิลิเมนต์ลูก “value” แสดงค่าตัวแปรโดยอิลิเมนต์ “value” ประกอบด้วยลักษณะประจำ “name” แสดงชื่อของตัวแปรที่ถูกใช้ในข้อกำหนด โดยมีลักษณะประจำสำหรับบอกว่าตัวแปรนี้ถูกประกาศไว้ ณ ที่ใด โดยถ้าในอิลิเมนต์ “value” ปรากฏลักษณะประจำ “atid” แสดงตัวแปรที่ถูกใช้นี้เป็นตัวแปรที่ถูกประกาศเป็นลักษณะประจำของคลาสนี้โดยมีลำดับที่เป็นค่าเดียวกันกับค่าที่แสดงในลักษณะประจำนั้น

ถ้าปรากฏลักษณะประจำ “FI” แสดงว่าค่าของตัวแปรที่นำมาใช้นำมาจากส่วนนำเข้าของเมท็อดนี้ และถ้าปรากฏลักษณะประจำ “sid” แสดงว่าค่าของตัวแปรที่นำมาใช้ถูกกำหนดค่าในข้อความสั่งที่มีลำดับที่แสดงในลักษณะประจำนี้ สำหรับลักษณะประจำ “methodParam” แสดงว่าค่าที่นำมาใช้นี้รับมาจากส่วนนำเข้าของเมท็อดนี้

- อิลิเมนต์ “Def” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “statement” แสดงตัวแปรที่ถูกกำหนดค่าในข้อความสั่งที่แสดงในอิลิเมนต์แม่ “statement” โดยมีอิลิเมนต์ลูก “value” แสดงค่าตัวแปรนี้ถูกกำหนดค่าในข้อความสั่งนี้ โดยอิลิเมนต์ “value” ประกอบด้วยลักษณะประจำ “name” ซึ่งแสดงชื่อของตัวแปรที่ถูกกำหนดค่าในข้อความสั่งนี้
- อิลิเมนต์ “Call” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “statement” จะปรากฏเมื่อข้อความสั่งที่แสดงในอิลิเมนต์ “statement” ซึ่งเป็นอิลิเมนต์แม่ของอิลิเมนต์นี้มีการเรียกใช้เมท็อด โดยลักษณะประจำ “Eid” แสดงลำดับของเมท็อดที่ถูกเรียกในข้อความสั่งนี้
- อิลิเมนต์ “update” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “method” ปรากฏเมื่อเมท็อดนั้นกำหนดค่าของตัวแปรซึ่งอ้างอิงถึงชนิดข้อมูลแฝงของมอดูล โดยลักษณะประจำ “name” แสดงชื่อตัวแปรที่ถูกกำหนดค่าในเมท็อดที่แสดงในอิลิเมนต์แม่ “method”

เอกสารเอ็กเอ็มแอลที่แสดงรูปแบบของกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ กำหนดโดยเอ็กเอ็มแอลสกีมาดังรูปที่ 7.19 ถึงรูปที่ 7.22

รูปที่ 7.19 แสดงการประกาศอิลิเมนต์ “Project“ ซึ่งเป็นรากของเอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ

รูปที่ 7.20 แสดงการประกาศชนิดอิลิเมนต์ ”TypeClass” และ “TypeAttribute” ชนิดอิลิเมนต์ ”TypeCalss” เป็นชนิดอิลิเมนต์ของอิลิเมนต์ ”Class” และชนิดอิลิเมนต์ ”TypeAttribute” เป็นชนิดอิลิเมนต์ของอิลิเมนต์ ”attribute”

รูปที่ 7.21 แสดงการประกาศชนิดอิลิเมนต์ ”TypeMethod” ”TypeParameter” และอิลิเมนต์ ”value” โดยที่ ชนิดอิลิเมนต์ ”TypeMethod” เป็นชนิดอิลิเมนต์ของอิลิเมนต์ “method” ”TypeParameter” เป็นชนิดอิลิเมนต์สำหรับอิลิเมนต์ “parameter” ในขณะที่อิลิเมนต์ ”value” เป็นอิลิเมนต์ลูกของอิลิเมนต์ “Used” และ “Def”

รูปที่ 7.22 แสดงการประกาศชนิดอิลิเมนต์ “TypeStatement“ ซึ่งเป็นชนิดอิลิเมนต์ของอิลิเมนต์ “statement”

ตัวอย่างกราฟแสดงความสัมพันธ์ระหว่างวัตถุที่แสดงโดยเอกสารเอ็กเอ็มแอลของซอร์สโคดในภาคผนวก จ แสดงในภาคผนวก ซ

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:element name="Project">
      <xsd:annotation>
         <xsd:documentation>the root element</xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Class" type="TypeClass" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
```

รูปที่ 7.19 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศอิลิเมนต์ “Project“ ที่ใช้กำกับเอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ

7.2.3 การสร้างโครงสร้างข้อมูลต้นไม้ของส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ

ในไฟล์กำหนดโครงการระบุตำแหน่งของเอกสารเอ็กเอ็มแอลที่แสดงส่วนของโปรแกรมภาษาจาวาที่จะพบ ในขั้นตอนการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่นำเสนอในงานวิจัยนี้ต้องแปลงส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบในรูปแบบเอกสารเอ็กเอ็มแอลให้กลับเป็นโครงสร้างข้อมูลต้นไม้เพื่อใช้ในขั้นตอนต่อไป

7.2.4 แสดงผลความก้าวหน้าที่เกิดขึ้น

การแสดงผลความก้าวหน้าของการพัฒนาโปรแกรมที่เกิดขึ้นแบ่งเป็นสองส่วน คือ ภาพรวมความก้าวหน้าของการพัฒนา และรายละเอียดความก้าวหน้าของการพัฒนาโปรแกรมแยกตามสมการที่ระบุไว้ในส่วนประกาศสัจพจน์

ภาพรวมความก้าวหน้าของการพัฒนาที่เกิดขึ้น แสดงเป็นแท่งก้าวหน้า (Progress bar) ของแต่ละมอดูลเพื่อแสดงว่าข้อกำหนดซอฟต์แวร์ของแต่ละมอดูลได้รับการพัฒนาไปแล้วมากน้อยเพียงใด ในขณะที่รายละเอียดความก้าวหน้าของการพัฒนาโปรแกรมแยกตามสมการที่ระบุไว้ในส่วนประกาศสัจพจน์ แสดงรายละเอียดเฉพาะในระดับสมการว่าแต่ละสมการที่ประกาศไว้ในแต่ละมอดูลนั้นได้รับการพัฒนาโปรแกรมตามสมการนั้นๆ แล้วหรือไม่

7.2.5 การตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

เมื่อได้กราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของซอร์สโคดโปรแกรม และโครงสร้างข้อมูลต้นไม้ของส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบแล้วเครื่องมือก็เริ่มทำงานตามขั้นตอนวิธีตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่เสนอไว้ในบทที่ 6โดยท่องไปตามโครงสร้าง

```
<xsd:complexType name="TypeClass">
  <xsd:sequence>
    <xsd:element name="attribute" type="TypeAttribute" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="method" type="TypeMethod" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="modifier" type="xsd:string" use="optional"/>
  <xsd:attribute name="cid" type="xsd:integer" use="required"/>
</xsd:complexType>
<xsd:complexType name="TypeAttribute">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="type" type="xsd:string" use="required"/>
  <xsd:attribute name="modifier" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value=""/>
        <xsd:enumeration value="public"/>
        <xsd:enumeration value="protected"/>
        <xsd:enumeration value="private"/>
        <xsd:enumeration value="public static"/>
        <xsd:enumeration value="protected static"/>
        <xsd:enumeration value="private static"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="atid" type="xsd:integer" use="required"/>
  <xsd:attribute name="value" type="xsd:string"/>
</xsd:complexType>
```

รูปที่ 7.20 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “TypeClass“ และ “TypeAttribute“ ที่ใช้กำกับเอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ

ข้อมูลต้นไม้ของส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบเพื่อหาเงื่อนไขของการตัดส่วนโปรแกรม แล้วตัดส่วนโปรแกรมโดยใช้วิธีการตัดส่วนโปรแกรมโดยใช้กราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของซอร์สโคดโปรแกรมแล้วเปรียบเทียบความก้าวหน้าที่เกิดขึ้น

ขั้นตอนวิธีการใช้เครื่องมือที่พัฒนาขึ้นและส่วนประสานผู้ใช้ของเครื่องมือที่พัฒนาขึ้นทั้ง 2 เครื่องมือแสดงในภาคผนวก ฌ

```
<xsd:complexType name="TypeMethod">
  <xsd:sequence>
    <xsd:element name="parameter" type="TypeParameter" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="statement" type="TypeStatement" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="update" minOccurs="0">
      <xsd:complexType>
        <xsd:attribute name="name" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="return" type="xsd:string" use="required"/>
  <xsd:attribute name="modifier" type="xsd:string" use="required"/>
  <xsd:attribute name="Eid" type="xsd:integer" use="required"/>
</xsd:complexType>
<xsd:element name="value">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="atid" type="xsd:integer"/>
    <xsd:attribute name="FI" type="xsd:integer"/>
    <xsd:attribute name="methodParam" type="xsd:string" use="optional"
default="Y"/>
  </xsd:complexType>
</xsd:element>
<xsd:complexType name="TypeParameter">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="type" type="xsd:string" use="required"/>
  <xsd:attribute name="FI" type="xsd:integer" use="required"/>
</xsd:complexType>
```

รูปที่ 7.21 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ ”TypeMethod” และ ”TypeParameter” และอิลิเมนต์ ”value” ที่ใช้กำกับเอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ

```
  <xsd:complexType name="TypeStatement">
    <xsd:sequence>
      <xsd:element name="Used">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="value" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Def">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="value" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Call" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="Eid" type="xsd:integer" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="trueStart" type="xsd:integer" use="optional"/>
    <xsd:attribute name="trueEnd" type="xsd:integer" use="optional"/>
    <xsd:attribute name="falseStart" type="xsd:integer" use="optional"/>
    <xsd:attribute name="falseEnd" type="xsd:integer" use="optional"/>
    <xsd:attribute name="sid" type="xsd:integer"/>
    <xsd:attribute name="FOid" type="xsd:integer"/>
    <xsd:attribute name="underCondition" type="xsd:integer"/>
  </xsd:complexType>
</xsd:schema>
```

รูปที่ 7.22 เอกสารเอ็กเอ็มแอลสกีมาแสดงการประกาศชนิดอิลิเมนต์ “TypeStatement“ ที่ใช้กำกับเอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุ

บทที่ 8

การทดสอบ

ในบทนี้กล่าวถึง ขั้นตอนการตรวจสอบเครื่องมือที่พัฒนาขึ้นจากกฎการสร้างโครงภาษาจาวาที่เสนอในบทที่ 4 และวิธีการที่เสนอในบทที่ 5 และบทที่ 6 สภาวะที่ใช้ในการทดสอบเครื่องมือ และผลที่ได้จากการทดสอบ

8.1 ขั้นตอนการตรวจสอบเครื่องมือที่พัฒนาขึ้น

ขั้นตอนการทดสอบเครื่องมือที่พัฒนาขึ้นประกอบไปด้วยขั้นตอนต่างๆ ดังนี้

8.1.1 เลือกข้อกำหนดรูปนัยคาเฟโอบีเจที่นำมาทดสอบ ข้อกำหนดที่นำมาทดสอบมี 3 ข้อกำหนด ประกอบด้วย ข้อกำหนดการนับเลข (Counter) ข้อกำหนดการทำเครื่องหมาย (Flag) และข้อกำหนดการนับเลขที่มีสวิตซ์ (Counter-with-Switch)

8.1.2 สร้างโครงภาษาจาวาของข้อกำหนดนั้นด้วยเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบที่พัฒนาขึ้น

8.1.3 ส่งโครงภาษาจาวาที่เครื่องมือสร้างขึ้นให้กับผู้พัฒนาจำนวน 5 คน

8.1.4 รับโปรแกรมภาษาจาวาที่พัฒนาเสร็จแล้วคืนมา

8.1.5 ตรวจสอบความก้าวหน้าที่เกิดขึ้นด้วยเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่พัฒนาขึ้น

8.1.6 ตรวจสอบความก้าวหน้าที่เกิดขึ้นด้วยมือ

8.1.7 เปรียบเทียบผลที่ได้จากข้อ 8.1.5และ 8.1.6

8.2 สภาวะที่ใช้ทดสอบเครื่องมือ

เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบ มีรายละเอียดดังนี้

คอมพิวเตอร์พีซี Pentium 4 2.8 GHz

หน่วยความจำหลัก 1024 MB

ฮาร์ดดิสก์ความจุ 120 GB

8.3 ข้อกำหนดรูปนัยที่ใช้ในการทดสอบเครื่องมือ

8.3.1 ข้อกำหนดการนับเลข

ข้อกำหนดการนับเลข เป็นการออกแบบโปรแกรมเพื่อเพิ่ม หรือลดเลขจำนวนเต็ม โดยการบวก หรือการลบ ขึ้นอยู่กับค่าที่ผู้ใช้เป็นผู้ใส่ให้กับระบบ ข้อกำหนดการนับเลขมี 2 มอดูล ได้แก่ มอดูล “COUNTER“ และ “DATA“ โดยมอดูล “DATA“ กำหนดชนิดข้อมูลสังเกตุค่าได้ “Data“ และโอเปอเรชันที่ต้องใช้ ในขณะที่มอดูล “COUNTER“ รับมอดูล “DATA“ เป็นพารามิเตอร์ เพื่อเรียกใช้โอเปอเรชัน “+“ และชนิดข้อมูลสังเกตค่าได้ “Data“ ซึ่งประกาศในมอดูล “DATA“ มาใช้ในมอดูล “COUNTER“ ข้อกำหนดการนับเลขแสดงดังรูปที่ 8.1

```
-- generic data
mod* DATA {
  [ Data ]
  op _+_ : Data Data -> Data
}

-- generic counter object
mod* COUNTER(X :: DATA) {
  *[ Counter ]*

  bop add : Data Counter -> Counter    -- method
  bop read_ : Counter -> Data          -- attribute

  eq read add(N:Data, C:Counter) = read(C) + N .
}
```

รูปที่ 8.1 ข้อกำหนดการนับเลข

ในขั้นตอนการสร้างโครงภาษาจาวาของข้อกำหนดนี้ กำหนดให้สร้างมอดูล “COUNTER“ เป็นคลาสเพียงคลาสเดียว กำหนดชื่อตัวแปรภายในคลาสว่า “counter” และกำหนดให้ชนิดข้อมูลแฝง “Data” มีชนิดของข้อมูลภาษาจาวาเป็น “int” โดยเปลี่ยนชื่อเมท็อดดังตารางที่ 8.1

ตารางที่ 8.1 การกำหนดชื่อเมท็อดที่สอดคล้องกับชื่อโอเปอเรชันของข้อกำหนดการนับเลข

ชื่อโอเปอเรชัน	ชื่อเมท็อด
_+_	Plus
add	Add
read_	Read

ทั้งนี้ข้อกำหนดการนับเลขมีสมการที่ต้องตรวจสอบความก้าวหน้าทั้งสิ้น 1 สมการ

8.3.2 ข้อกำหนดการทำเครื่องหมาย

ข้อกำหนดการทำเครื่องหมาย เป็นการออกแบบโปรแกรมเพื่อสร้างเครื่องหมายเอาไว้ว่าขณะนี้วัตถุในโปรแกรมอยู่ในสถานะขึ้น (Up) หรือ ลง (Down) ข้อกำหนดการทำเครื่องหมายแสดงดังรูปที่ 8.2

```
mod* FLAG {

  *[ Flag ]*

  bops (up_) (dn_) (rev_) : Flag -> Flag  -- methods
  bop up?_ : Flag -> Bool                 -- attribute

  var F : Flag
  eq up? up F = true .
  eq up? dn F = false .
  eq up? rev F = not up? F .
}
```

รูปที่ 8.2 ข้อกำหนดการทำเครื่องหมาย

ในข้อกำหนดการนับมีการเรียกโอเปอเรชันที่ไม่มีอาริที “true” และ “false” ซึ่งมีโคอาริทีเป็นชนิดข้อมูลสังเกตค่าได้ “Bool” และไม่ได้ประกาศในมอดูล “FLAG” แต่ว่าประกาศอยู่ในมอดูล “BOOL” ซึ่งเป็นมอดูลมาตราฐานของข้อกำหนดรูปนัยคาเฟโอบีเจออยู่แล้ว ไม่จำเป็นต้องประกาศในข้อกำหนดก็ได้

ในขั้นตอนการสร้างโครงภาษาจาวาของข้อกำหนดการทำเครื่องหมาย กำหนดให้สร้างคลาสจากมอดูล “FLAG” เพียงมอดูลเดียว กำหนดชื่อตัวแปรภายในคลาสว่า “flag” และกำหนดให้ชนิดข้อมูลแฝง “Flag” มีชนิดของข้อมูลภาษาจาวาเป็น “boolean” โดยมีการกำหนดชื่อเมท็อดดังตารางที่ 8.2

ตารางที่ 8.2 การกำหนดชื่อเมท็อดที่สอดคล้องกับชื่อโอเปอเรชันของข้อกำหนดมอดูลการทำเครื่องหมาย

ชื่อโอเปอเรชัน	ชื่อเมท็อด
up?_	upProj
up_	Up
dn_	Dn
rev_	Rev

ทั้งนี้ข้อกำหนดการการทำเครื่องหมายมีสมการที่ต้องตรวจสอบความก้าวหน้าทั้งสิ้น 3 สมการ

8.3.3 ข้อกำหนดการนับเลขที่มีสวิตซ์

ข้อกำหนดการนับเลขที่มีสวิตซ์ เป็นระบบที่ใช้สำหรับการบวก และลบเลขจำนวนเต็ม เช่นเดียวกับข้อกำหนดการนับเลข แต่การบวกหรือลบนั้น มีสวิตซ์เป็นตัวควบคุม เมื่อสถานะของสวิตซ์เป็น “on“ จะเป็นการบวก แต่ถ้าสถานะของสวิตซ์เป็น “off“ ก็จะเป็นการลบ ข้อกำหนดการนับเลขที่มีสวิตซ์ประกอบด้วย 3 มอดูล โดยมีมอดูล “COUNTER-WITH-SWITCH“ ซึ่งเป็นมอดูลหลักนำเข้ามอดูล “SWITCH“ และมอดูล “COUNTER“ นอกจากนี้มอดูล “SWITCH“ ยังเรียกเข้ามอดูล “ON-OFF“ เพื่อประกาศสถานะของมอดูล “SWITCH“ ในขณะที่มอดูล “COUNTER“ นำเข้ามอดูลมาตรฐาน “INT“ หรือเลขจำนวนเต็ม เพื่อใช้ชนิดข้อมูลสังเกตค่าได้ “Int“ มาอธิบายพฤติกรรมของมอดูล “COUNTER“ ข้อกำหนดการนับเลขที่มีสวิตซ์แสดงในภาคผนวก ก

ในขั้นตอนการสร้างโครงภาษาจาวา กำหนดให้สร้างมอดูล “COUNTER“ มอดูล “SWITCH“ และมอดูล “COUNTER-WITH-SWITCH“ เป็นคลาส โดยระบุค่าต่างๆ เพิ่มเติม โดยตารางที่ 8.3 แสดงการกำหนดชื่อคลาสที่จะสร้างโครงภาษาจาวาที่สอดคล้องมอดูลในข้อกำหนดการนับเลขที่มีสวิตซ์ ตารางที่ 8.4 แสดงการกำหนดชนิดของข้อมูลภาษาจาวาสำหรับแต่ละชนิดข้อมูลสังเกตค่าได้ของข้อกำหนดการนับเลขที่มีสวิตซ์

ตารางที่ 8.3 การกำหนดชื่อคลาสที่สอดคล้องกับชื่อมอดูลของข้อกำหนดการนับเลขที่มีสวิตซ์

ชื่อมอดูล	ชื่อคลาส
SWITCH	SWITCH
COUNTER	COUNTER
COUNTER-WITH-SWITCH	COUNTER_WITH_SWITCH

ตารางที่ 8.4 การกำหนดชนิดข้อมูลของภาษาจาวาที่สอดคล้องกับชนิดข้อมูลสังเกตค่าได้ของข้อกำหนดการนับเลขที่มีสวิตซ์

ชนิดข้อมูลสังเกตค่าได้	ชนิดข้อมูลภาษาจาวา
Value	boolean
Int	int

การกำหนดชื่อของเมท็อดที่สอดคล้องกับชื่อโอเปอเรชัน และการกำหนดชื่อของค่าคงที่และค่าของค่าคงที่นั้น ของมอดูล “SWITCH“ ในข้อกำหนดการนับเลขที่มีสวิตซ์แสดงในตารางที่ 8.5 และตารางที่ 8.6 ตามลำดับ และการกำหนดชื่อตัวแปรสำหรับเก็บสถานะของคลาสที่สร้างจากมอดูล “SWITCH“ แสดงในตารางที่ 8.7

ตารางที่ 8.5 การกำหนดชื่อเมท็อดที่สอดคล้องกับชื่อโอเปอเรชันของมอดูล “SWITCH“ ในข้อกำหนดการนับเลขที่มีสวิตซ์

ชื่อโอเปอเรชัน	ชื่อเมท็อด
init	constructor
on_	On
off_	Off
state_	State

ตารางที่ 8.6 การกำหนดชื่อค่าคงที่ที่สอดคล้องกับชื่อโอเปอเรชันที่ไม่มีอาริทีในมอดูล “SWITCH“ ในข้อกำหนดการนับเลขที่มีสวิตซ์

ชื่อโอเปอเรชันที่ไม่มีอาริที	ชื่อค่าคงที่	ค่าของค่าคงที่
on	on	true
off	off	false

ตารางที่ 8.7 การกำหนดชื่อตัวแปรในมอดูล “SWITCH“ ในข้อกำหนดการนับเลขที่มีสวิตซ์

อาริทีของโปรเจคชันโอเปอเรชันของชนิดข้อมูลแฝง	โคอาริทีของโปรเจคชันโอเปอเรชันของชนิดข้อมูลแฝง	ชื่อตัวแปร
Switch	Value	Switch

ในขณะที่การกำหนดชื่อของเมท็อดที่สอดคล้องกับชื่อโอเปอเรชัน และการกำหนดชื่อของค่าคงที่ และค่าของค่าคงที่นั้น ของมอดูล “COUNTER“ ในข้อกำหนดการนับเลขที่มีสวิตซ์แสดงในตารางที่ 8.8 และ ตารางที่ 8.9 ตามลำดับ และการกำหนดชื่อตัวแปรสำหรับเก็บสถานะของคลาสที่สร้างจากมอดูล “COUNTER“ แสดงในตารางที่ 8.10

ตารางที่ 8.8 การกำหนดชื่อเมท็อดที่สอดคล้องกับชื่อโอเปอเรชันของมอดูล “COUNTER“ ในข้อกำหนดการนับเลข

ชื่อโอเปอเรชัน	ชื่อเมท็อด
init	constructor
add	Add
read	Read

ตารางที่ 8.9 การกำหนดชื่อค่าคงที่ที่สอดคล้องกับชื่อโอเปอเรชันที่ไม่มีอาริทีในมอดูล “COUNTER“ ในข้อกำหนดการนับเลขที่มีสวิตซ์

ชื่อโอเปอเรชันที่ไม่มีอาริที	ชื่อค่าคงที่	ค่าของค่าคงที่
0	zero	0

ตารางที่ 8.10 การกำหนดชื่อตัวแปรในมอดูล “COUNTER“ ในข้อกำหนดการนับเลขที่มีสวิตซ์

อาริทีของโปรเจคชัน โอเปอเรชันของชนิดข้อมูลแฝง	โคอาริทีของโปรเจคชัน โอเปอเรชันของชนิดข้อมูลแฝง	ชื่อตัวแปร
Counter	Int	counter

ตารางที่ 8.11แสดงการกำหนดชื่อเมท็อดที่สอดคล้องกับโอเปอเรชันของมอดูล “COUNTER-WITH-SWITCH“ ในข้อกำหนดการนับเลขที่มีสวิตซ์ และการกำหนดชื่อตัวแปรที่ใช้เก็บสถานะของคลาสที่สร้างจากมอดูล “COUNER-WITH-SWITCH“ โดยแยกตามอาริทีและโคอาริทีของโปรเจคชันโอเปอเรชันแสดงในตารางที่ 8.12

8.4 ผลการทดสอบ

เมื่อนำโครงภาษาจาวาให้ผู้พัฒนา 5 คนซึ่งเป็นนิสิตระดับปริญญาโทในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย พัฒนาโปรแกรมต่อแล้วนำโปรแกรมที่พัฒนาแล้วมาตรวจสอบความก้าวหน้าด้วยเครื่องมือเปรียบเทียบกับการตรวจสอบโดยผู้ทำวิจัย โดยตรวจสอบทีละคลาสที่สร้างมาจากแต่ละข้อกำหนดแล้วใช้เครื่องมือตรวจสอบว่ามีจำนวนสมการของข้อกำหนดที่เครื่องมือตรวจสอบและแจ้งว่าพัฒนาเสร็จแล้วจำนวนกี่สมการ

ตารางที่ 8.11 การกำหนดชื่อเมท็อดที่สอดคล้องกับชื่อโอเปอเรชันของมอดูล "COUNTER-WITH-SWITCH" ในข้อกำหนดการนับเลขที่มีสวิตซ์

ชื่อโอเปอเรชัน	ชื่อเมท็อด
init	constructor
put	Put
add_	Add
sub_	Sub
read	Read
counter	Counter
switch_	Switch

ตารางที่ 8.12 การกำหนดชื่อตัวแปรในมอดูล "COUNTER-WITH-SWITCH" ในข้อกำหนดการนับเลขที่มีสวิตซ์

อาริทีของโปรเจคชัน โอเปอเรชันของชนิดข้อมูลแฝง	โคอาริทีของโปรเจคชัน โอเปอเรชันของชนิดข้อมูลแฝง	ชื่อตัวแปร
CWS	Int	cwsInt
CWS	Switch	cwsSwitch
CWS	Counter	cwsCounter

ทั้งนี้ข้อกำหนดการการนับเลขที่มีสวิตซ์มีสมการที่ต้องตรวจสอบหาความก้าวหน้าแบ่งตามมอดูลที่ถูกเลือกให้สร้างเป็นคลาสดังแสดงในตารางที่ 8.13

ตารางที่ 8.13 จำนวนสมการที่ต้องการตรวจสอบของแต่ละมอดูลในข้อกำหนดการนับเลขที่มีสวิตซ์

มอดูล	จำนวนสมการที่ต้องการตรวจสอบ
COUNTER	3
SWITCH	2
COUNTER-WITH-SWITCH	10

จากนั้นให้ผู้ทำวิจัยตรวจสอบโดยวิเคราะห์จากซอร์สโคดโปรแกรมว่ามีจำนวนสมการของข้อกำหนดที่ได้รับการพัฒนาแล้วจำนวนกี่สมการ แล้วนำมาเปรียบเทียบกัน โดยผลจากการตรวจสอบคลาสที่สร้างจากมอดูล “COUNTER“ ในข้อกำหนดการนับแสดงในตารางที่ 8.14 ผลจากการตรวจสอบคลาส ที่สร้างจากมอดูล “FLAG“ ในข้อกำหนดการทำเครื่องหมายแสดงในตารางที่ 8.15 และผลจากการตรวจสอบคลาสที่สร้างจากมอดูล “COUNTER“ มอดูล “SWITCH“ และมอดูล “COUNTER-WITH-SWITCH“ ในข้อกำหนดการนับที่มีสวิตซ์แสดงในตารางที่ 8.16

ตารางที่ 8.17และตารางที่ 8.18 ตามลำดับ

ตารางที่ 8.14 ผลการทดสอบเครื่องมือเปรียบเทียบกับผู้ทำวิจัยในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่สอดคล้องกับข้อกำหนดการนับ

ผู้พัฒนาคนที่	ความก้าวหน้าที่ตรวจสอบด้วยเครื่องมือ	ความก้าวหน้าที่ตรวจสอบโดยผู้ทำการวิจัย
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1

ตารางที่ 8.15 ผลการทดสอบเครื่องมือเปรียบเทียบกับผู้ทำวิจัยในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่สอดคล้องกับข้อกำหนดการทำเครื่องหมาย

ผู้พัฒนาคนที่	ความก้าวหน้าที่ตรวจสอบด้วยเครื่องมือ	ความก้าวหน้าที่ตรวจสอบโดยผู้ทำการวิจัย
1	3	3
2	3	3
3	3	3
4	3	3
5	3	3

ตารางที่ 8.16 ผลการทดสอบเครื่องมือเปรียบเทียบกับผู้ทำวิจัยในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่สอดคล้องกับมอดูล “COUNTER“ ในข้อกำหนดการนับที่มีสวิตซ์

ผู้พัฒนาคนที่	ความก้าวหน้าที่ตรวจสอบด้วยเครื่องมือ	ความก้าวหน้าที่ตรวจสอบโดยผู้ทำการวิจัย
1	2	2
2	2	2
3	2	2
4	2	2
5	2	2

ตารางที่ 8.17 ผลการทดสอบเครื่องมือเปรียบเทียบกับผู้ทำวิจัยในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่สอดคล้องกับมอดูล “SWITCH“ ในข้อกำหนดการนับที่มีสวิตซ์

ผู้พัฒนาคนที่	ความก้าวหน้าที่ตรวจสอบด้วยเครื่องมือ	ความก้าวหน้าที่ตรวจสอบโดยผู้ทำการวิจัย
1	3	3
2	2	3
3	3	3
4	3	3
5	3	3

8.5 การวิเคราะห์ผลการทดสอบ

จากผลการทดลองตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมด้วยเครื่องมือที่พัฒนาขึ้น พบว่าเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมซึ่งใช้วิธีการตรวจสอบความก้าวหน้าที่นำเสนอในงานวิจัยนี้สามารถตรวจสอบความถูกต้องได้ใกล้เคียงกับการตรวจสอบความก้าวหน้าโดยมนุษย์ อย่างไรก็ดียังมีรูปแบบการเขียนโปรแกรมบางรูปแบบซึ่งไม่สามารถตรวจสอบได้ถูกต้อง ซึ่งสรุปรูปแบบซึ่งไม่สามารถตรวจสอบความได้ถูกต้อง ดังต่อไปนี้

ตารางที่ 8.18 ผลการทดสอบเครื่องมือเปรียบเทียบกับผู้ทำวิจัยในการตรวจสอบความก้าวหน้าในการพัฒนาโปรแกรมที่สอดคล้องกับมอดูล “COUNTER-WITH-SWITCH“ ของข้อกำหนดการนับที่มีสวิตซ์

ผู้พัฒนาคนที่	ความก้าวหน้าที่ตรวจสอบด้วยเครื่องมือ	ความก้าวหน้าที่ตรวจสอบโดยผู้ทำการวิจัย
1	9	10
2	8	8
3	9	9
4	10	10
5	8	8

8.5.1 การเรียกเมท็อดเพื่อเปลี่ยนแปลงสถานะของตัวแปรในขณะที่ส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบเป็นการเปลี่ยนแปลงสถานะของข้อมูล

จากผลการทดลองซึ่งแสดงใน

ตารางที่ 8.17 ในการพัฒนาของผู้พัฒนาคนที่ 2 เมท็อดที่ไม่สามารถตรวจสอบความก้าวหน้าได้ถูกต้องคือเมท็อดตัวสร้าง (Constructor Method) ดังแสดงดังรูปที่ 8.3

```
public SWITCH()
{
   this.Off();
}
```

รูปที่ 8.3 ส่วนของโปรแกรมที่ไม่สามารถตรวจสอบความก้าวหน้าได้ถูกต้อง

โดยส่วนของโปรแกรมภาษาภาษาจาวาที่คาดว่าจะพบแสดงดังรูปที่ 8.4

```
Switch = false;
```

รูปที่ 8.4 ส่วนของโปรแกรมที่คาดว่าจะพบในเมท็อดที่แสดงในรูปที่ 8.3

ทั้งนี้เมท็อด “Off“ ก็เปลี่ยนสถานะของตัวแปร “Switch“ ให้มีค่าเป็น “off“ ซึ่งตรงกับส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบแต่ว่าเครื่องมือยังไม่สามารถวิเคราะห์เข้าไปในส่วนของเมท็อด “Off“ ว่าทำงานอย่างไรได้

8.5.2 การเรียกตัวแปรสมาชิกของวัตถุโดยตรง ในขณะที่ส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบเป็นการเรียกโปรแกรมแสดงค่า (Ancestor method)

จากผลการทดลองซึ่งแสดงในตารางที่ 8.18 โปรแกรมที่พัฒนาโดยผู้พัฒนาคนที่ 1 เมท็อดที่ไม่สามารถตรวจสอบความถูกต้องได้แก่เมท็อดที่แสดงในรูปที่ 8.5

```
public int Read()
{
   return cwsCounter.counter;
}
```

รูปที่ 8.5 ส่วนของโปรแกรมที่ไม่สามารถตรวจสอบความก้าวหน้าได้ถูกต้อง

โดยส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบแสดงในรูป

```
cwsCounter.Read();
```

รูปที่ 8.6 ส่วนของโปรแกรมที่คาดว่าจะพบในเมท็อดที่แสดงในรูปที่ 8.5

โปรแกรมที่ผู้พัฒนาขึ้นใช้วิธีคืนค่าตัวแปรสมาชิกของวัตถุ “cwsCounter“ โดยตรงซึ่งผิดจากหลักที่ควรจะเป็นในการเขียนโปรแกรมภาษาเชิงวัตถุซึ่งไม่ควรมีการเรียกใช้สมาชิกของวัตถุโดยตรงแต่ควรเรียกใช้เมท็อดแสดงค่าของตัวแปรสมาชิกของวัตถุ อย่างไรก็ดีส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบแสดงส่วนของโปรแกรมที่เรียกใช้เมท็อดเพื่อแสดงค่าของตัวแปรสมาชิกซึ่งเครื่องมือไม่สามารถตรวจสอบความก้าวหน้าในกรณีที่ผู้พัฒนาโปรแกรมพัฒนาโปรแกรมเช่นนี้โดยมีส่วนของภาษาจาวาที่คาดว่าจะพบในลักษณะนี้ได้

บทที่ 9
สรุปผลการวิจัย

9.1 สรุปผลการวิจัย

งานวิจัยนี้ได้ออกแบบวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมภาษาจาวาจากข้อกำหนดรูปนัยคาเฟโอบีเจ โดยขั้นตอนการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมนั้นประกอบด้วยสามขั้นตอนคือ การสร้างโครงภาษาจาวา การสร้างส่วนของภาษาจาวาที่คาดว่าจะพบ และการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

โดยในงานวิจัยนี้เสนอกฎในการสร้างโครงภาษาจาวาจากส่วนวากยสัมพันธ์ของข้อกำหนดรูปนัยโอบีเจ โดยกฎในการสร้างโครงภาษาจาวาต้องอาศัยรายละเอียดเฉพาะซึ่งผู้ออกแบบกำหนดเพิ่มเติม โครงภาษาจาวาที่ได้เป็นโปรแกรมภาษาจาวาที่มีส่วนวากยสัมพันธ์สอดคล้องกับส่วนวากยสัมพันธ์ของข้อกำหนด และสอดคล้องกับรายละเอียดเฉพาะที่ผู้ออกแบบกำหนดให้ ในขณะเดียวกันงานวิจัยนี้ก็เสนอขั้นตอนวิธีการสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบจากสมการที่ประกาศไว้ในส่วนความหมายของข้อกำหนดรูปนัยคาเฟโอบีเจเพื่อสร้างส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบ ส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบเป็นข้อความสั่งซึ่งแสดงในเมท็อดที่เกี่ยวข้องกับสมการที่ประกาศอยู่ในส่วนกำหนดสัจพจน์ โดยส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบใช้ในขั้นตอนการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

ในขั้นตอนการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมที่นำเสนอในงานวิจัยนี้นำซอร์สโคดโปรแกรมที่ผู้พัฒนาพัฒนาต่อจากโครงภาษาจาวาที่ได้ในขั้นตอนการสร้างโครงภาษาจาวามาตรวจสอบความก้าวหน้า โดยในขั้นตอนการตรวจสอบความก้าวหน้าจะตรวจสอบทีละสมการที่ประกาศไว้ในส่วนประกาศสัจพจน์ ซึ่งแต่ละสมการจะประกาศสถานะของตัวแปรในโปรแกรมที่แตกต่างกัน ดังนั้นในขั้นตอนนี้จึงต้องตัดส่วนโปรแกรมเฉพาะส่วนของโปรแกรมที่เกี่ยวข้องกับสมการที่ต้องการตรวจสอบ แล้วนำส่วนของโปรแกรมที่ตัดส่วนแล้ว และส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบจากสมการเดียวกันมาผ่านขั้นตอนวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมซึ่งนำเสนอในงานาวิจัยนี้เพื่อตรวจสอบว่าสมการนั้นๆ ได้รับการพัฒนาแล้วหรือไม่

ในการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมภาษาจาวานั้นต้องมีข้อกำหนดรูปนัยคาเฟโอบีเจที่ถูกต้องตามวากยสัมพันธ์ของข้อกำหนดรูปนัยคาเฟโอบีเจ และถูกต้องตามความต้องการของระบบ เพื่อมาสร้างโครงภาษาจาวา โดยในขั้นตอนนี้ผู้ใช้เครื่องมือสร้างโครง

ภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบต้องเพิ่มรายละเอียดข้อมูลเพื่อใช้ในการสร้างโครงภาษาจาวาอีกด้วย ในขณะเดียวกันเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบจะสร้างส่วนของภาษาจาวาที่คาดว่าจะพบด้วย

เมื่อเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบสร้างโครงภาษาจาวาแล้ว โครงภาษาจาวาที่ได้จะถูกส่งไปให้กับผู้พัฒนาเพื่อพัฒนาโปรแกรมในระหว่างนั้นผู้พัฒนาสามารถส่งโปรแกรมมาตรวจสอบความก้าวหน้ากับเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาซอฟต์แวร์โดยเปรียบเทียบกับส่วนของภาษาจาวาที่คาดว่าจะพบเพื่อตรวจสอบความก้าวหน้าในระหว่างการพัฒนาโปรแกรมได้

งานวิจัยนี้ทดสอบเครื่องมือกับข้อกำหนดรูปนัยคาเฟโอบีเจ 3 ข้อกำหนด โดยสร้างโครงภาษาจาวาของข้อกำหนดรูปนัยคาเฟโอบีเจทั้ง 3 ข้อกำหนด แล้วส่งให้ผู้พัฒนาโปรแกรม 5 คน พัฒนาโปรแกรมตามข้อกำหนดรูปนัยคาเฟโอบีเจแต่ละข้อกำหนด แล้วตรวจสอบความก้าวหน้าโดยเปรียบเทียบระหว่างเครื่องมือตรวจสอบซึ่งใช้วิธีการที่นำเสนอในงานวิจัยนี้กับผู้ทำวิจัย ผลลัพธ์แสดงออกมาว่าเครื่องมือสามารถตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมได้ใกล้เคียงกับมนุษย์เป็นผู้ตรวจสอบ

9.2 ปัญหาและข้อจำกัดของระบบ

9.2.1 ในขั้นตอนการสร้างโครงภาษาจาวา โครงภาษาจาวาที่ได้อาจจะสร้างโครงภาษาจาวาที่เกินความจำเป็น เช่น ตัวแปรที่ไม่ได้ใช้งาน หรือเมท็อดที่ไม่จำเป็นต้องพัฒนาได้

9.2.2 ข้อกำหนดที่ใช้ในเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบต้องถูกต้องตามหลักวากยสัมพันธ์ของข้อกำหนดคาเฟโอบีเจ

9.2.3 โปรแกรมภาษาจาวาที่นำมาตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมต้องถูกต้องตามหลักวากยสัมพันธ์ของโปรแกรมภาษาจาวา

9.2.4 การเปรียบเทียบระหว่างส่วนของภาษาจาวาที่คาดว่าจะพบและโปรแกรมภาษาจาวาบางรูปแบบยังไม่สามารถตรวจสอบความก้าวหน้าได้อย่างถูกต้องเช่น การเรียกเมท็อดเพื่อเปลี่ยนแปลงสถานะของตัวแปรในขณะที่ส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบเป็นการเปลี่ยนแปลงสถานะของข้อมูล และการเรียกตัวแปรสมาชิกของวัตถุโดยตรง ในขณะที่ส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบเป็นการเรียกโปรแกรมแสดงค่า เป็นต้น

9.2.5 โอเปอเรชันที่นำมาใช้ในงานวิจัยนี้จะต้องประกาศชนิดข้อมูลแฝงในอาริทีของโอเปอเรชันนั้นเพียงชนิดเดียวเท่านั้น

9.2.6 การประกาศตัวแปรในข้อกำหนดรูปนัยคาเฟโอบีเจต้องไม่ใช้ชื่อตัวแปรเป็นชื่อเดียวกันไม่ว่าจะเป็นการประกาศตัวแปรแบบธรรมดา หรือประกาศตัวแปรแบบทันทีเมื่อใช้

9.2.7 เครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม สามารถตรวจสอบความก้าวหน้าของโปรแกรมที่พัฒนามาจากโครงภาษาจาวาที่มีการแก้ไขภายในเมท็อดเท่านั้น ไม่สามารถตรวจสอบความก้าวหน้าของโปรแกรมที่แก้ไขส่วนอื่นภายนอกขอบเขตของเมท็อด

9.3 ข้อเสนอแนะในการพัฒนาเพิ่มเติม

ขั้นตอนการสร้างส่วนของภาษาจาวาที่คาดว่าจะพบ สร้างส่วนของภาษาจาวาในเมท็อดของคลาสภาษาจาวา ซึ่งในบางกรณีส่วนของโปรแกรมที่คาดว่าจะพบนี้สามารถนำไปใช้ในเมท็อดของภาษาจาวาได้ทันที ดังนั้นควรมีการศึกษาว่าโอเปอเรชันลักษณะใดที่สามารถนำส่วนของโปรแกรมมาแทนได้เลยในขั้นตอนการสร้างโครงภาษาจาวาเพื่อช่วยลดความพยายามของผู้พัฒนาโปรแกรมต่อไป

โดยปกติแล้วการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมสามารถทำได้สะดวกถ้าหากว่าผู้พัฒนาและผู้จัดการโครงการอยู่ในสถานที่เดียวกัน ดังนั้นวิธีการตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมโดยอัตโนมัติสามารถช่วยอำนวยความสะดวกในการตรวจสอบความก้าวหน้าของการพัฒนาในสภาพแวดล้อมการทำงานแบบกระจายได้ อย่างไรก็ดีในงานวิจัยนี้ยังไม่ได้เสนอการนำวิธีการตรวจสอบความก้าวหน้าที่เสนอในงานวิจัยนี้มาใช้ในการตรวจสอบความก้าวหน้าของการพัฒนาในสภาพแวดล้อมการทำงานแบบกระจายแต่อย่างใดจึงเป็นแนวทางในการพัฒนาเพิ่มเติมได้

9.4 ผลงานที่เกี่ยวข้องกับการวิจัย

ส่วนหนึ่งของงานวิจัยนี้ได้รับคัดเลือกให้นำเสนอในงานประชุมวิชาการและตีพิมพ์ในเอกสาร Proceedings of the International Conference on Software Engineering Research and Practice (SERP'03) ในหัวข้อ "An Automatic Approach to Transform CafeOBJ Specifications to Java Template Code" ในระหว่างวันที่ 23-26 มิถุนายน 2546 โดยรายละเอียดแสดงอยู่ในภาคผนวก ญ

รายการอ้างอิง

1. Conception, A.I., S. Lin, and S.J. Simon. Manageing the Software Development by Using the Recursive Multi-Threaded (RMT) Software Life-Cycle. in 30th International Conference on Technology of Object-Oriented Languages and Systems - USA (TOOLS 30). Santa Barbara, California, 1999.
2. Harry, A., Formal methods Factfile. 1 ed. Chichester, England: John Wiley & Sons. 1996: 386.
3. OMG, OMG Unified Modeling Language Specification version 1.4. OMG, 2001.
4. Pressman, R.S., A Manager's Guide To Software. McGraw-Hill. 1993: 528.
5. Yang, G. and I. Tomek. Team Lab: A Collaborative Environment for Teamwork. in Groupware. Proceedings. Sixth International Workshop on. 2000. Madeira, Portugal, 2000.
6. Morse, E. and M.P. Steves. CollabLogger: A Tool for Visualizing Groups at Work. in Proceedings of WETICE , Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. 2000: IEEE Computer Society, 2000.
7. Hiroyuki Murakoshi, A.S., and Koichiro Ochimizu. Construction of deliberation structure in e-mail communication. in Computational Inteligence. Blackwell, 2000.
8. Diaconescu, R. and K. Futasugi, CafeOBJ Report. World Scientific, 1998.
9. Nakagawa, A.T., T. Sawada, and K. Futasugi, CafeOBJ User's Manual version 1.4. World Scientific, 1998.
10. Hasting, T.E. and A.S.M. Sajeev, A Vector-based Approach to Software Size Measurement and Effort Estimation. IEEE Transactions on Software Engineering, 27(4) 2001: 337 - 350.
11. Devanbu, P. GENOA—A Customizable, Front-end-Retargetable Source Code Analysis Framework. in ACM Transactions on Software Engineering and Methodology (TOSEM), 1999.
12. Antoniol, G., et al. Evolving object oriented design to improve code traceability. in Program Comprehension, Proceedings. Seventh International Workshop on, 1999.

13. Weiser, M., Program slicing. IEEE Transactions on Software Engineering, 1984. 10(4) 1984: 352-357.

14. Harman, M., et al. Pre/post conditioned slicing. in In IEEE International Conference on Software Maintenance (ICSM'01). Florence, Italy: IEEE Computer Society Press, Los Alamitos, California, USA, 2001.

15. Lucia, A.D., A.R. Fasolino, and M. Munro. Understanding function behaviors through program slicing. in Proceedings of the Fourth Workshop on Program Comprehension. Berlin, Germany, 1996.

16. Korel, B. and J. Laski, Dynamic Slicing of Computer Programs. The Journal of Systems and Software. 13(3) 1990: 187-195.

17. Chen, J.L., F.J. Wang, and Y.L. Chen. An Objected-Oriented Dependency Graph for Program Slicing. in The 24th International Conference on Technology of Object Oriented Language and Systems (TOOLS Asia). Beijing, 1997.

18. Chen, J.L., F.J. Wang, and Y.L. Chen. Slicing Object-Oriented Programs. in Proceedings of the APSEC'97. Hongkong, China, 1997.

19. Johnston, W. and G. Rose, Guidelines for the Manual Conversion of Object-Z to C++. University of Queensland, Software Verification Research Center: Queensland, Australia, 1993.

20. Miyazaki, T. and E.A. Lee. Code generation by using integer-controlled dataflow graph. in Acoustics, Speech, and Signal Processing, 1997. ICASSP-97. IEEE International Conference on. 1997.

21. Thongmak, M. and P. Muenchaisri. Design of Rules for Transforming UML Sequence Diagrams into Java code. in Ninth Asia-Pacific Software Engineering Conference (APSEC'02). Gold Coast, Australia, 2002.

ภาคผนวก

ภาคผนวก ก
ข้อกำหนดการนับเลขที่มีสวิตซ์ซึ่งใช้เป็นตัวอย่างในวิทยานิพนธ์

```
module! ON-OFF{
  [Value]

  op on : → Value
  op off : → Value
}

module* SWITCH{
  pr(ON-OFF)

  *[Switch]*

  op init-sw : → Switch
  bop on : Switch → Switch
  bop off : Switch → Switch
  bop status : Switch → Value

  var S : Switch
  eq status(init-sw) = on .
  eq status(on(S)) = on .
  eq status(off(S)) = off .
}

module* COUNTER{
  protecting (INT)
  *[Counter]*
  op Init: → Counter
  bop add: Int Counter → Counter
  bop read: Counter → Int
  var I: Int
  var C: Counter

  eq read(init) = 0.
  eq read(add(I,C)) = I + read(C).
}

module* COUNTER-WITH-SWITCH {
  Protecting(SWITCH + COUNTER)
  *[Cws]*
  op init: → Cws
  bop put: Int Cws → Cws
  bop add_: Cws → Cws
  bop sub_: Cws → Cws
  bop read: Cws → Int
  bop counter_: Cws → Counter
  bop switch_: Cws → Switch

  var N: Int
  var C: Cws
  eq read(C) = read (Counter C) .

  eq switch (init) = init .
  eq switch put(N,C) = Switch C .
  eq switch add(C) = on (switch C) .
  eq switch sub(C) = off(switch C) .

  eq counter(init) = init .
  ceq counter(put(N,C)) = add(N,counter(C))
       if state(switch(C)) == on .
  ceq counter(put(N,C))=add(-N,counter(C))
       if state(switch(C)) == off .
  eq couner add(C) = counter C .
  eq counter sub(C) = counter C.
}
```

sort declaration

operation declaration

axiom declaration

รูปที่ ก.1 ข้อกำหนดการนับเลขที่มีสวิตช์

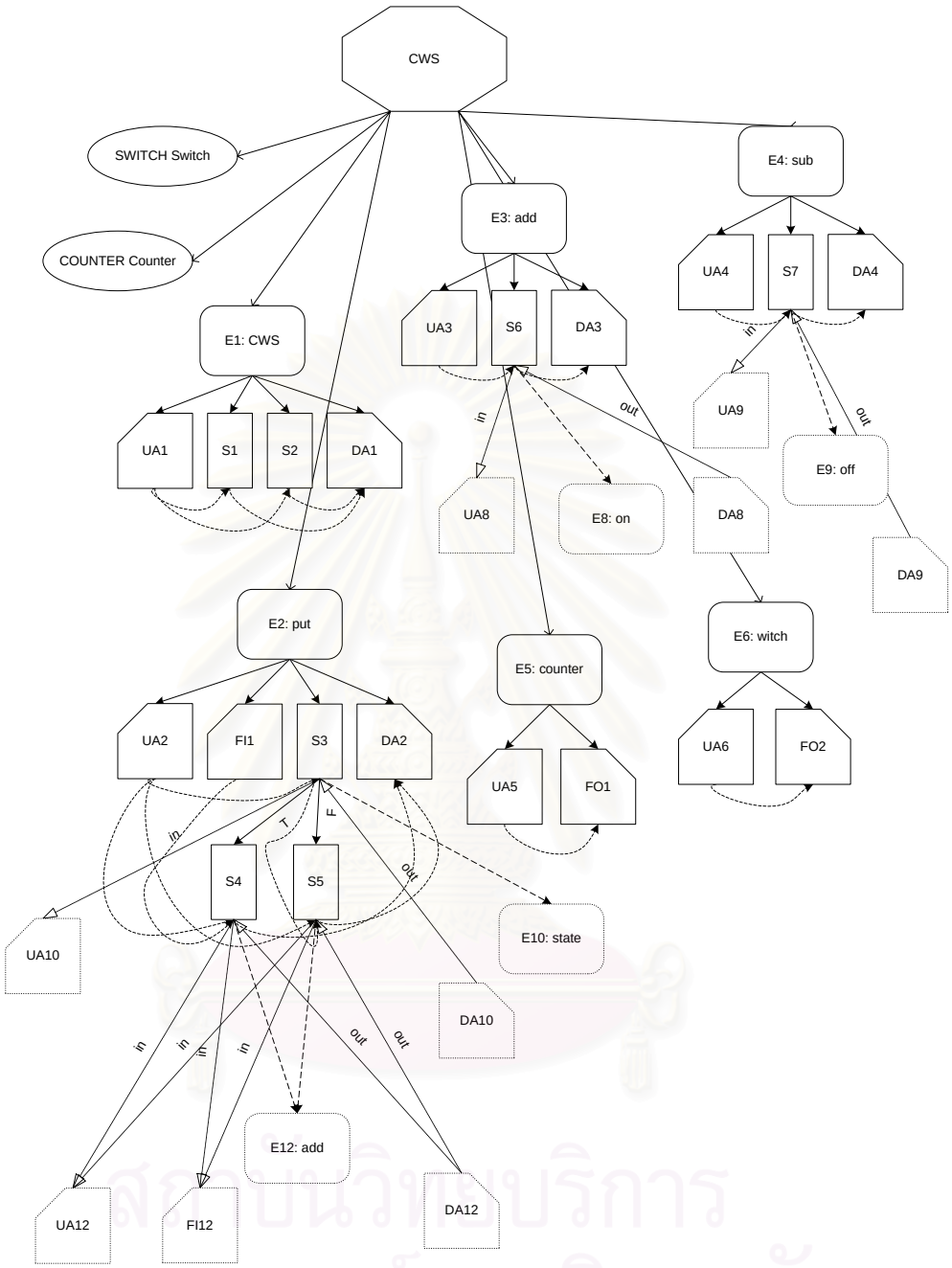
ภาคผนวก ข
ตัวอย่างโปรแกรมภาษาจาวา

```
public class SWITCH
{
	public boolean Switch;
	public boolean off = false;
	public boolean on = true;
	public SWITCH(){		// E7, UA7, DA7
		Switch = off;		// S8
	}

	public void On(){		// E8, UA8, DA8
		Switch = on;		// S9
	}
	public void Off(){		// E9, UA9, DA9
		Switch = off;
	}
	public boolean status(){	// E10, UA10
		return Switch;		// FO10
	}
};

public class COUNTER
{
	public int Counter;
	public COUNTER(){		// E11, UA11, DA11
		Counter = 0;		// S11
	}
	public void add(int c) {	// E12, UA12, FI12, DA12
		Counter +=c;		// S12
	}
	public int read(){		// E13, UA13
		return Counter;		 // FO12
	}
};

public class CWS
{
	public SWITCH Switch;
	public COUNTER Counter;
	public CWS() {			// E1, UA1, DA1
		Switch = new SWITCH();		// S1
		Counter = new COUNTER();	// S2
	}
	public void put(int c){		// E2, FI1, UA2, DA2
		if (Switch.state())		// S3
		{
				Counter.add(c);// S4
		}else
				Counter.add(-c);// S5
	}
	public void add() {		// E3, UA3, DA3
		Switch.On();		// S6
	}
	public void sub(){		// E4, UA4, DA4
	Switch.Off();			// S7
	}
	public int read(){
		return Counter.read();
	}
	public COUNTER counter(){ // E5 , UA5
		return Counter;		// FO1
	}
	public SWITCH switch(){ // E6, UA6
		return Switch;		// FO2
	}
};
```

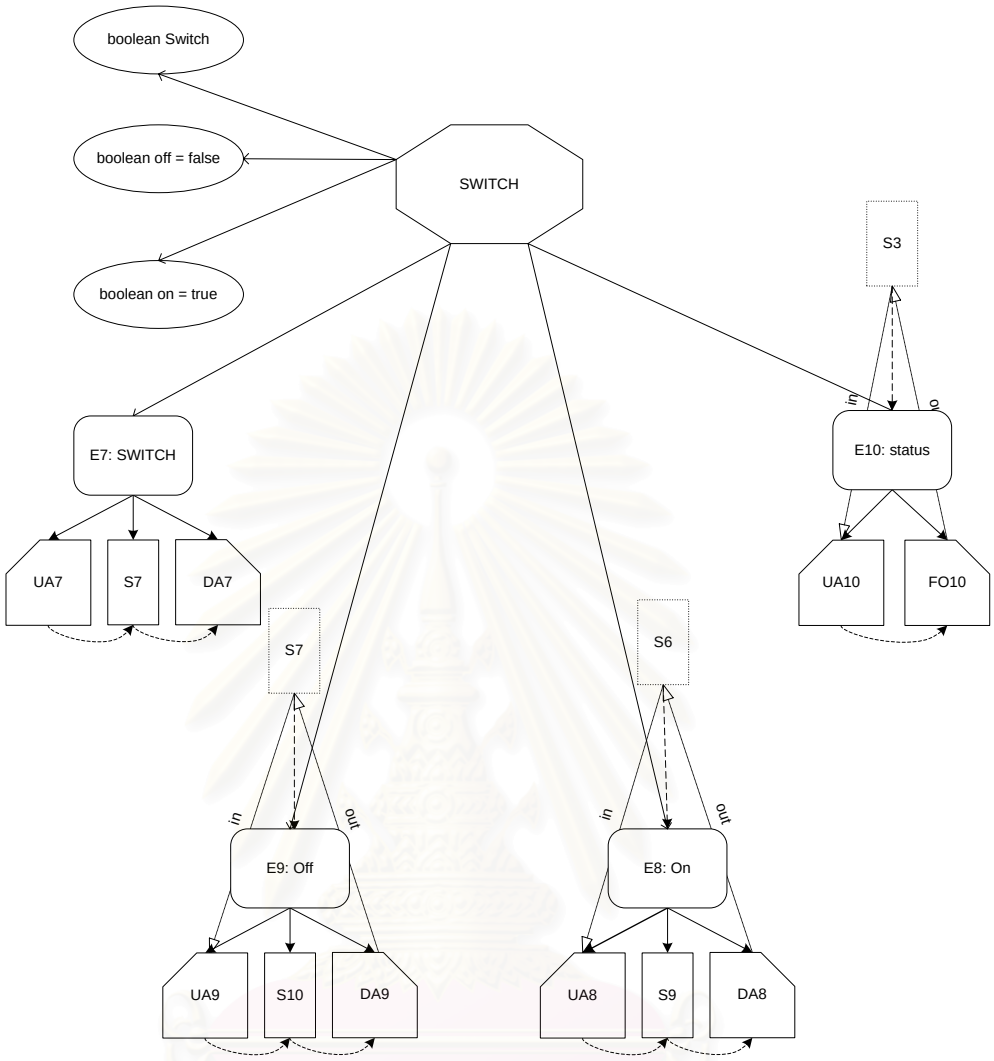
รูปที่ ข.1 ตัวอย่างโปรแกรมที่ได้ทดลองเขียนขึ้น

จากโปรแกรมที่ได้เขียนขึ้นด้านหลังของเครื่องหมาย // จะบอกจุดยอดที่ถูกสร้างขึ้นในการสร้างกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุซึ่งจะแสดงใน ภาคผนวก ค

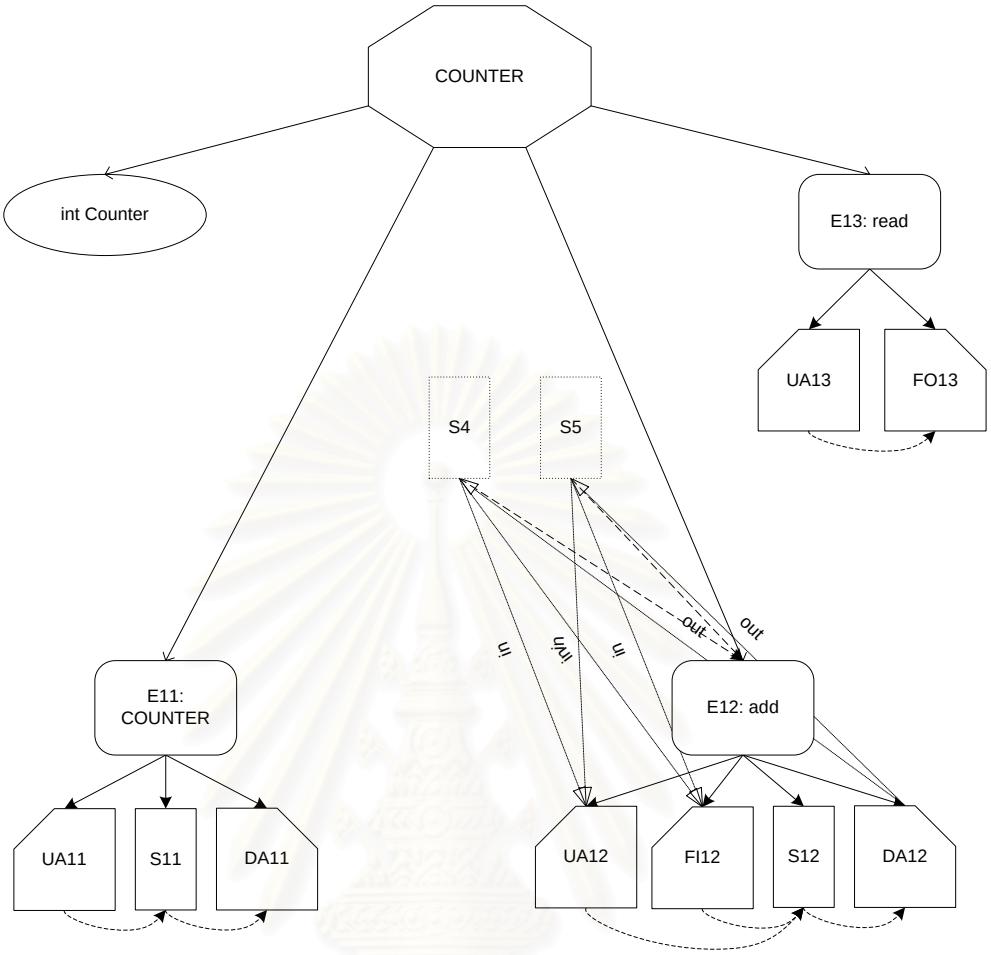
ภาคผนวก ค
กราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของโปรแกรมที่ได้พัฒนาแล้วในภาคผนวก ข



รูปที่ ค.1 กราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของคลาส “CWS“ ซึ่งอ้างอิงกับมอดูล “COUNTER-WITH-SWITCH“



รูปที่ ค.2 กราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของคลาส “SWITCH“ ซึ่งอ้างอิงกับมอดูล “SWITCH“



รูปที่ ค.3 กราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของคลาส “COUNTER“ ซึ่งอ้างอิงกับมอดูล “COUNTER“

ภาคผนวก ง
โครงภาษาจาวาที่สร้างจากข้อกำหนดรูปนัยคาเฟโอบีเจ
ในภาคผนวก ก

```
public class SWITCH{

	public static  boolean on = true;
	public static  boolean off = false;
	public boolean Switch;
	public SWITCH()
	{

	}

	public boolean State()
	{
		/**@todo: Implement this State method*/
		throw new java.lang.UnsupportedOperationException("Method State() not yet
implemented.");
	}

	public void Off()
	{
		/**@todo: Implement this Off method*/
		throw new java.lang.UnsupportedOperationException("Method Off() not yet implemented.");
	}

	public void On()
	{
		/**@todo: Implement this On method*/
		throw new java.lang.UnsupportedOperationException("Method On() not yet implemented.");
	}
```

รูปที่ ง.1 โครงภาษาจาวาของคลาส “SWITCH“ ที่สร้างจากมอดูล “SWITCH“

```
public class COUNTER{

	public static  int zero = 0;
	public int counter;
	public COUNTER()
	{

	}

	public int Read()
	{
		/**@todo: Implement this Read method*/
		throw new java.lang.UnsupportedOperationException("Method Read() not yet
implemented.");
	}

	public void Add(int i)
	{
		/**@todo: Implement this Add method*/
		throw new java.lang.UnsupportedOperationException("Method Add(int i) not yet
implemented.");
	}
```

รูปที่ ง.2 โครงภาษาจาวาของคลาส “COUNTER“ ที่สร้างจากมอดูล “COUNTER“

```
public class COUNTER_WITH_SWITCH{

	public int cwsInt;
	public COUNTER cwsCounter;
	public SWITCH cwsSwitch;
	public COUNTER_WITH_SWITCH()
	{

	}

	public void Sub()
	{
		/**@todo: Implement this Sub method*/
		throw new java.lang.UnsupportedOperationException("Method Sub() not yet
implemented.");
	}

	public COUNTER Counter()
	{
		/**@todo: Implement this Counter method*/
		throw new java.lang.UnsupportedOperationException("Method Counter() not yet
implemented.");
	}

	public int Read()
	{
		/**@todo: Implement this Read method*/
		throw new java.lang.UnsupportedOperationException("Method Read() not yet
implemented.");
	}

	public void Add()
	{
		/**@todo: Implement this Add method*/
		throw new java.lang.UnsupportedOperationException("Method Add() not yet
implemented.");
	}

	public SWITCH Switch()
	{
		/**@todo: Implement this Switch method*/
		throw new java.lang.UnsupportedOperationException("Method Switch() not yet
implemented.");
	}

	public void Put(int i)
	{
		/**@todo: Implement this Put method*/
		throw new java.lang.UnsupportedOperationException("Method Put(int i) not yet
implemented.");
	}

}
```

รูปที่ ง.3 โครงภาษาจาวาของคลาส “COUNTER_WITH_SWITCH“ ที่สร้างจากมอดูล “COUNTER-WITH-SWITCH“

ภาคผนวก จ
ตัวอย่างโปรแกรมภาษาจาวาที่พัฒนาจากโครงภาษาจาวาซึ่งแสดงใน ภาคผนวก ง

```
/**
 * SWITCH
 * @author Siros S.
 */
public class SWITCH{

	public static  boolean on = true;
	public static  boolean off = false;
	public boolean Switch;
	public SWITCH()
	{
		Switch = off;
	}

	public boolean State()
	{
		/**@todo: Implement this State method*/
		return Switch;
	}

	public void Off()
	{
		/**@todo: Implement this Off method*/
		//throw new java.lang.UnsupportedOperationException("Method Off() not yet
implemented.");
		Switch = off;
	}

	public void On()
	{
		/**@todo: Implement this On method*/
		Switch = on;
	}

}
```

รูปที่ จ.1 โปรแกรมภาษาจาวาที่พัฒนาจากโครงภาษาจาวาของคลาส “SWITCH“
ที่สร้างจากมอดูล “SWITCH“

```
/**
 * COUNTER
 * @author Siros S.
 */
public class COUNTER{

	public static  int zero = 0;
	public int counter;
	public COUNTER()
	{
		counter = zero;
	}

	public int Read()
	{
		/**@todo: Implement this Read method*/
		return counter;
	}

	public void Add(int i)
	{
		/**@todo: Implement this Add method*/
		counter = counter + i;
	}

}
```

รูปที่ จ.2 โปรแกรมภาษาจาวาที่พัฒนาจากโครงภาษาจาวาของคลาส “COUNTER“
ที่สร้างจากมอดูล “COUNTER“

```
/**
 * COUNTER_WITH_SWITCH
 * @author Siros S.
 */
public class COUNTER_WITH_SWITCH{

	public int cwsInt;
	public COUNTER cwsCounter;
	public SWITCH cwsSwitch;
	public COUNTER_WITH_SWITCH()
	{
		cwsCounter = new COUNTER();
		cwsSwitch = new SWITCH();
	}

	public void Sub()
	{
		/**@todo: Implement this Sub method*/
		// Turn switch off
		cwsSwitch.Off();
		// Do nothing with counter
	}

	public COUNTER Counter()
	{
		/**@todo: Implement this Counter method*/
		return cwsCounter;
	}

	public int Read()
	{
		/**@todo: Implement this Read method*/
		return cwsInt;
	}

	public void Add()
	{
		/**@todo: Implement this Add method*/
		// Turn switch on
		cwsSwitch.On();
		// Do nothing with counter
	}

	public SWITCH Switch()
	{
		/**@todo: Implement this Switch method*/
		return cwsSwitch;
	}

	public void Put(int i)
	{
		/**@todo: Implement this Put method*/
		// Do nothing with switch

		// If switch is on, add i to counter
		if (cwsSwitch.State() == SWITCH.on) {
			cwsCounter.Add(i);
		} else {
		// If switch is off, add (-i) to counter
			cwsCounter.Add(-i);
		}
	}

}
```

รูปที่ จ.3 โปรแกรมภาษาจาวาที่พัฒนาจากโครงภาษาจาวาของคลาส “COUNTER_WITH_SWITCH“ ที่สร้างจากมอดูล “COUNTER-WITH-SWITCH“

ภาคผนวก ฉ
เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบจากข้อกำหนดรูปนัยคาเฟโอบีเจในภาคผนวก ก

เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล COUNTER-WITH-SWITCH

```
<?xml version="1.0" encoding="UTF-8"?>
<Project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CafeSchema.xsd">
  <Module Name="COUNTER-WITH-SWITCH"
ClassName="COUNTER_WITH_SWITCH">
    <VARIABLE Name="N" Sort="Int" />
    <VARIABLE Name="C" Sort="Cws" />
    <Proj_Operation Operation="read" arity="Cws" coarity="Int">
      <Variable Cafe_Variable="C" Name="cwsInt" />
      <Equation Name="C  = read counter C">
        <Left_Equation Name="C" order="0" />
        <Right_Equation Name="read counter C" type="list_of_operation" order="0">
          <leftOperation Name="counter C" type="operation List" />
        </Right_Equation>
        <Right_Equation Hanging_Operation="read" Name="counter C" order="1"
type="projectionOperation" coarity="Counter">
          <Return value="cwsCounter" />
        </Right_Equation>
        <Right_Equation Name="read cwsCounter" order="2" type="variable_list"
coarity="Int">
          <Return value="cwsCounter.Read()" />
        </Right_Equation>
      </Equation>
    </Proj_Operation>
    <Proj_Operation Operation="switch" arity="Cws" coarity="Switch">
      <Variable Cafe_Variable="C" Name="cwsSwitch" />
      <Equation Name="init  = init">
        <Left_Equation Name="init" order="0" />
        <Right_Equation Name="init" type="non_arity" coarity="Switch" order="0">
          <Return value="cwsSwitch.constructor()" />
        </Right_Equation>
      </Equation>
      <Equation Name="put N ,  C  = switch C">
        <Left_Equation Name="put N ,  C" coarity="Cws" order="0" />
        <Right_Equation Name="switch C" type="projectionOperation" coarity="Switch"
order="0">
          <Return value="cwsSwitch" />
        </Right_Equation>
      </Equation>
```

รูปที่ ฉ.1 เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล “COUNTER-WITH-SWITCH“

```
    <Equation Name="add C  = on switch C">
      <Left_Equation Name="add C" coarity="Cws" order="0" />
      <Right_Equation Name="on switch C" type="list_of_operation" order="0">
        <leftOperation Name="switch C" type="operation List" />
      </Right_Equation>
      <Right_Equation Hanging_Operation="on" Name="switch C" order="1"
type="projectionOperation" coarity="Switch">
        <Return value="cwsSwitch" />
      </Right_Equation>
      <Right_Equation Name="on cwsSwitch" order="2" type="variable_list"
coarity="Switch">
        <Return value="cwsSwitch.On()" />
      </Right_Equation>
    </Equation>
   <Equation Name="sub C  = off switch C">
      <Left_Equation Name="sub C" coarity="Cws" order="0" />
      <Right_Equation Name="off switch C" type="list_of_operation" order="0">
        <leftOperation Name="switch C" type="operation List" />
      </Right_Equation>
      <Right_Equation Hanging_Operation="off" Name="switch C" order="1"
type="projectionOperation" coarity="Switch">
        <Return value="cwsSwitch" />
      </Right_Equation>
      <Right_Equation Name="off cwsSwitch" order="2" type="variable_list"
coarity="Switch">
        <Return value="cwsSwitch.Off()" />
      </Right_Equation>
    </Equation>
   </Proj_Operation>
   <Proj_Operation Operation="counter" arity="Cws" coarity="Counter">
    <Variable Cafe_Variable="C" Name="cwsCounter" />
    <Equation Name="init  = init">
      <Left_Equation Name="init" order="0" />
      <Right_Equation Name="init" type="non_arity" coarity="Counter" order="0">
        <Return value="cwsCounter.constructor()" />
      </Right_Equation>
    </Equation>
```

รูปที่ ฉ.1 เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล “COUNTER-WITH-SWITCH“ (ต่อ)

```
    <Equation Name="put N ,  C" type="Condition Equation">
      <Left_Equation Name="put N ,  C" coarity="Cws" order="0" />
      <Guard_Equation Name="state switch C   == on" ConditionOperand="==">
        <Left_Guard Name="state switch C" type="list_of_operation" order="0">
          <leftOperation Name="switch C" type="operation List" />
        </Left_Guard>
        <Left_Guard Hanging_Operation="state" Name="switch C" order="1"
type="projectionOperation" coarity="Switch">
          <Return value="cwsSwitch" />
        </Left_Guard>
        <Left_Guard Name="state cwsSwitch" order="2" type="variable_list"
coarity="Value">
          <Return value="cwsSwitch.State()" />
        </Left_Guard>
        <Right_Guard Name="on" type="non_arity" coarity="Value" order="0">
          <Return value="true" />
        </Right_Guard>
        <Result_Guard Name="add N ,  counter C" type="list_of_operation" order="0">
          <leftOperation Name="N ,  counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add" Name="N ,  counter C" order="1"
type="list_of_operation">
          <leftOperation Name=",  counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add N" Name=",  counter C" order="2"
type="list_of_operation">
          <leftOperation Name=" counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add N ," Name=" counter C" order="3"
type="list_of_operation">
          <leftOperation Name="counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add N , " Name="counter C" order="4"
type="projectionOperation" coarity="Counter">
          <Return value="cwsCounter" />
        </Result_Guard>
        <Result_Guard Name="add N ,  cwsCounter" order="5" type="variable_list"
coarity="Cws" />
        <Result_Guard Name="add i , cwsCounter" order="6" type="variable_list"
coarity="Cws">
          <Return value="cwsCounter.Add(i)" />
        </Result_Guard>
      </Guard_Equation>
```

รูปที่ ฉ.1 เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล “COUNTER-WITH-SWITCH“ (ต่อ)

```
      <Guard_Equation Name="state switch C   == off" ConditionOperand="==">
        <Left_Guard Name="state switch C" type="list_of_operation" order="0">
          <leftOperation Name="switch C" type="operation List" />
        </Left_Guard>
        <Left_Guard Hanging_Operation="state" Name="switch C" order="1"
type="projectionOperation" coarity="Switch">
          <Return value="cwsSwitch" />
        </Left_Guard>
        <Left_Guard Name="state cwsSwitch" order="2" type="variable_list"
coarity="Value">
          <Return value="cwsSwitch.State()" />
        </Left_Guard>
        <Right_Guard Name="off" type="non_arity" coarity="Value" order="0">
          <Return value="false" />
        </Right_Guard>
        <Result_Guard Name="add - N ,  counter C" type="list_of_operation" order="0">
          <leftOperation Name="- N ,  counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add" Name="- N ,  counter C" order="1"
type="list_of_operation">
          <leftOperation Name="N ,  counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add -" Name="N ,  counter C" order="2"
type="list_of_operation">
          <leftOperation Name=" ,  counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add - N" Name=" ,  counter C" order="3"
type="list_of_operation">
          <leftOperation Name=",  counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add - N " Name=",  counter C" order="4"
type="list_of_operation">
          <leftOperation Name=" counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add - N ," Name=" counter C" order="5"
type="list_of_operation">
          <leftOperation Name="counter C" type="operation List" />
        </Result_Guard>
        <Result_Guard Hanging_Operation="add - N , " Name="counter C" order="6"
type="projectionOperation" coarity="Counter">
          <Return value="cwsCounter" />
        </Result_Guard>
        <Result_Guard Name="add - N ,  cwsCounter" order="7" type="variable_list"
coarity="Cws" />
        <Result_Guard Name="add - i , cwsCounter" order="8" type="variable_list"
coarity="Cws">
          <Return value="cwsCounter.Add(-i)" />
        </Result_Guard>
      </Guard_Equation>
    </Equation>
```

รูปที่ ฉ.1 เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล COUNTER-WITH-SWITCH (ต่อ)

```
    <Equation Name="add C  = counter C">
      <Left_Equation Name="add C" coarity="Cws" order="0" />
      <Right_Equation Name="counter C" type="projectionOperation" coarity="Counter"
order="0">
        <Return value="cwsCounter" />
      </Right_Equation>
    </Equation>
    <Equation Name="sub C  = counter C">
      <Left_Equation Name="sub C" coarity="Cws" order="0" />
      <Right_Equation Name="counter C" type="projectionOperation" coarity="Counter"
order="0">
        <Return value="cwsCounter" />
      </Right_Equation>
    </Equation>
  </Proj_Operation>
 </Module>
</Project>
```

รูปที่ ฉ.1 เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล “COUNTER-WITH-SWITCH“ (ต่อ)

เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล “COUNTER“

```
<?xml version="1.0" encoding="UTF-8"?>
<Project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CafeSchema.xsd">
 <Module Name="COUNTER" ClassName="COUNTER">
  <VARIABLE Name="I" Sort="Int" />
  <VARIABLE Name="C" Sort="Counter" />
  <Proj_Operation Operation="read" arity="Counter" coarity="Int">
   <Variable Cafe_Variable="C" Name="counter" />
   <Equation Name="init  = 0">
    <Left_Equation Name="init" order="0" />
    <Right_Equation Name="0" type="non_arity" coarity="Int" order="0">
     <Return value="0" />
    </Right_Equation>
   </Equation>
   <Equation Name="add I ,  C   = I + read C">
    <Left_Equation Name="add I ,  C" coarity="Counter" order="0" />
    <Right_Equation Name="I + read C" type="list_of_operation" order="0">
     <leftOperation Name="+ read C" type="operation List" />
    </Right_Equation>
    <Right_Equation Hanging_Operation="I" Name="+ read C" order="1"
type="list_of_operation">
     <leftOperation Name="read C" type="operation List" />
    </Right_Equation>
    <Right_Equation Hanging_Operation="I +" Name="read C" order="2"
type="projectionOperation" coarity="Int">
     <Return value="counter" />
    </Right_Equation>
    <Right_Equation Name="I + counter" order="3" type="variable_list" coarity="Int" />
    <Right_Equation Name="i + counter" order="4" type="variable_list" coarity="Int">
     <Return value="i + counter" />
    </Right_Equation>
   </Equation>
  </Proj_Operation>
 </Module>
</Project>
```

รูปที่ ฉ.2 เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล “COUNTER“

เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล “SWITCH“

```
<?xml version="1.0" encoding="UTF-8"?>
<Project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CafeSchema.xsd">
  <Module Name="SWITCH" ClassName="SWITCH">
    <VARIABLE Name="S" Sort="Switch" />
    <Proj_Operation Operation="state" arity="Switch" coarity="Value">
      <Variable Cafe_Variable="S" Name="Switch" />
      <Equation Name="init = off">
        <Left_Equation Name="init" order="0" />
        <Right_Equation Name="off" type="non_arity" coarity="Value" order="0">
          <Return value="false" />
        </Right_Equation>
      </Equation>
      <Equation Name="on S  = on">
        <Left_Equation Name="on S" coarity="Switch" order="0" />
        <Right_Equation Name="on" type="non_arity" coarity="Value" order="0">
          <Return value="true" />
        </Right_Equation>
      </Equation>
      <Equation Name="off S  = off">
        <Left_Equation Name="off S" coarity="Switch" order="0" />
        <Right_Equation Name="off" type="non_arity" coarity="Value" order="0">
          <Return value="false" />
        </Right_Equation>
      </Equation>
    </Proj_Operation>
  </Module>
</Project>
```

รูปที่ ฉ.3 เอกสารเอ็กเอ็มแอลแสดงส่วนของภาษาจาวาที่คาดว่าจะพบของมอดูล “SWITCH”

ภาคผนวก ช
ตัวอย่างเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการของการสร้างโครงภาษาจาวาจากข้อกำหนดการนับเลขที่มีสวิตซ์ที่ใช้ในการทดลอง

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
  <InputSpec filename="E:\user\g44cds\Thesis\Cafe\Example\cws_test.mod" />
  <OutputSpecDir>
    <Path
toDir="E:\user\g44cds\Thesis\Cafe\Example\Output\cws_test\spec\SWITCH.xml" />
    <Path toDir="E:\user\g44cds\Thesis\Cafe\Example\Output\cws_test\spec\COUNTER-
WITH-SWITCH.xml" />
    <Path
toDir="E:\user\g44cds\Thesis\Cafe\Example\Output\cws_test\spec\COUNTER.xml" />
  </OutputSpecDir>
  <OutputSourceDir>
    <Path toDir="E:\user\g44cds\Thesis\Cafe\Example\Output\cws_test\src\SWITCH.java"
/>
    <Path
toDir="E:\user\g44cds\Thesis\Cafe\Example\Output\cws_test\src\COUNTER_WITH_SW
ITCH.java" />
    <Path
toDir="E:\user\g44cds\Thesis\Cafe\Example\Output\cws_test\src\COUNTER.java" />
  </OutputSourceDir>
  <ProjectSettingValue>
    <ProjOpSet>
      <ProjectionOp>
        <OpName name="state" />
        <ModuleName name="SWITCH" />
        <NumberOfInput name="1" />
        <Arity name="Switch" />
      </ProjectionOp>
      <ProjectionOp>
        <OpName name="counter" />
        <ModuleName name="COUNTER-WITH-SWITCH" />
        <NumberOfInput name="1" />
        <Arity name="Cws" />
      </ProjectionOp>
      <ProjectionOp>
        <OpName name="read" />
        <ModuleName name="COUNTER-WITH-SWITCH" />
        <NumberOfInput name="1" />
        <Arity name="Cws" />
      </ProjectionOp>
      <ProjectionOp>
        <OpName name="switch" />
        <ModuleName name="COUNTER-WITH-SWITCH" />
        <NumberOfInput name="1" />
        <Arity name="Cws" />
      </ProjectionOp>
```

รูปที่ ช.1 ตัวอย่างเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการของการสร้างโครงภาษาจาวาจากข้อกำหนดการนับเลขที่มีสวิตซ์

```
    <ProjectionOp>
      <OpName name="read" />
      <ModuleName name="COUNTER" />
      <NumberOfInput name="1" />
      <Arity name="Counter" />
    </ProjectionOp>
  </ProjOpSet>
  <OpSet>
    <OperationDetail>
      <OpName name="state" />
      <ModuleName name="SWITCH" />
      <NumberOfInput name="1" />
      <Coarity name="Value" />
      <MethodName name="State" />
    </OperationDetail>
    <OperationDetail>
      <OpName name="off" />
      <ModuleName name="SWITCH" />
      <NumberOfInput name="1" />
      <Coarity name="Switch" />
      <MethodName name="Off" />
    </OperationDetail>
    <OperationDetail>
      <OpName name="on" />
      <ModuleName name="SWITCH" />
      <NumberOfInput name="1" />
      <Coarity name="Switch" />
      <MethodName name="On" />
    </OperationDetail>
    <OperationDetail>
      <OpName name="on" />
      <ModuleName name="SWITCH" />
      <NumberOfInput name="0" />
      <Coarity name="Value" />
      <MethodName name="on" />
    </OperationDetail>
    <OperationDetail>
      <OpName name="init" />
      <ModuleName name="SWITCH" />
      <NumberOfInput name="0" />
      <Coarity name="Switch" />
      <MethodName name="constructor" />
    </OperationDetail>
    <OperationDetail>
      <OpName name="off" />
      <ModuleName name="SWITCH" />
      <NumberOfInput name="0" />
      <Coarity name="Value" />
      <MethodName name="off" />
    </OperationDetail>
```

รูปที่ ช.1 ตัวอย่างเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการของการสร้างโครงภาษาจาวาจากข้อกำหนดการนับเลขที่มีสวิตซ์ (ต่อ)

```
<OperationDetail>
  <OpName name="sub" />
  <ModuleName name="COUNTER-WITH-SWITCH" />
  <NumberOfInput name="1" />
  <Coarity name="Cws" />
  <MethodName name="Sub" />
</OperationDetail>
<OperationDetail>
  <OpName name="counter" />
  <ModuleName name="COUNTER-WITH-SWITCH" />
  <NumberOfInput name="1" />
  <Coarity name="Counter" />
  <MethodName name="Counter" />
</OperationDetail>
<OperationDetail>
  <OpName name="read" />
  <ModuleName name="COUNTER-WITH-SWITCH" />
  <NumberOfInput name="1" />
  <Coarity name="Int" />
  <MethodName name="Read" />
</OperationDetail>
<OperationDetail>
  <OpName name="add" />
  <ModuleName name="COUNTER-WITH-SWITCH" />
  <NumberOfInput name="1" />
  <Coarity name="Cws" />
  <MethodName name="Add" />
</OperationDetail>
<OperationDetail>
  <OpName name="switch" />
  <ModuleName name="COUNTER-WITH-SWITCH" />
  <NumberOfInput name="1" />
  <Coarity name="Switch" />
  <MethodName name="Switch" />
</OperationDetail>
<OperationDetail>
  <OpName name="put" />
  <ModuleName name="COUNTER-WITH-SWITCH" />
  <NumberOfInput name="2" />
  <Coarity name="Cws" />
  <MethodName name="Put" />
  <Parameter specName="N" javaName="i" />
</OperationDetail>
<OperationDetail>
  <OpName name="init" />
  <ModuleName name="COUNTER-WITH-SWITCH" />
  <NumberOfInput name="0" />
  <Coarity name="Cws" />
  <MethodName name="constructor" />
</OperationDetail>
<OperationDetail>
```

รูปที่ ช.1 ตัวอย่างเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการของการสร้างโครงภาษาจาวาจากข้อกำหนดการนับเลขที่มีสวิตซ์ (ต่อ)

```
    <OpName name="read" />
    <ModuleName name="COUNTER" />
    <NumberOfInput name="1" />
    <Coarity name="Int" />
    <MethodName name="Read" />
   </OperationDetail>
   <OperationDetail>
    <OpName name="add" />
    <ModuleName name="COUNTER" />
    <NumberOfInput name="2" />
    <Coarity name="Counter" />
    <MethodName name="Add" />
    <Parameter specName="I" javaName="i" />
   </OperationDetail>
   <OperationDetail>
    <OpName name="0" />
    <ModuleName name="COUNTER" />
    <NumberOfInput name="0" />
    <Coarity name="Int" />
    <MethodName name="zero" />
   </OperationDetail>
   <OperationDetail>
    <OpName name="init" />
    <ModuleName name="COUNTER" />
    <NumberOfInput name="0" />
    <Coarity name="Counter" />
    <MethodName name="constructor" />
   </OperationDetail>
  </OpSet>
  <VisibleSortMap>
   <vsMap visibleSort="Value" javaDataType="boolean" />
   <vsMap visibleSort="Int" javaDataType="int" />
  </VisibleSortMap>
  <VarNameMap>
   <varMap opName="state" hiddenSortName="Switch" javaName="Switch" />
   <varMap opName="read" hiddenSortName="Cws" javaName="cwsInt" />
   <varMap opName="counter" hiddenSortName="Cws" javaName="cwsCounter" />
   <varMap opName="switch" hiddenSortName="Cws" javaName="cwsSwitch" />
   <varMap opName="read" hiddenSortName="Counter" javaName="counter" />
  </VarNameMap>
  <ConstantValue>
   <Constant constOpName="on" sort="Value" javaValue="true" />
   <Constant constOpName="off" sort="Value" javaValue="false" />
   <Constant constOpName="0" sort="Int" javaValue="0" />
  </ConstantValue>
  <HiddenSortDecl>
   <HiddenSortDecl hsName="Switch" moduleName="SWITCH" />
   <HiddenSortDecl hsName="Cws" moduleName="COUNTER-WITH-SWITCH" />
   <HiddenSortDecl hsName="Counter" moduleName="COUNTER" />
  </HiddenSortDecl>
```

รูปที่ ช.1 ตัวอย่างเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการของการสร้างโครงภาษาจาวาจากข้อกำหนดการนับเลขที่มีสวิตช์ (ต่อ)

```
    <ClassNameMap>
      <classNameMap moduleName="SWITCH" className="SWITCH" />
      <classNameMap moduleName="COUNTER-WITH-SWITCH"
classNameMap="COUNTER_WITH_SWITCH" />
      <classNameMap moduleName="COUNTER" className="COUNTER" />
    </ClassNameMap>
    <ImportList>
      <Import based="SWITCH">
        <From input="ON-OFF" />
      </Import>
      <Import based="COUNTER-WITH-SWITCH">
        <From input="SWITCH" />
        <From input="COUNTER" />
      </Import>
      <Import based="COUNTER">
        <From input="INT" />
      </Import>
    </ImportList>
    <IncludeModule>
      <Include based="SWITCH" include="ON-OFF" />
      <Include based="COUNTER" include="INT" />
    </IncludeModule>
  </ProjectSettingValue>
</Project>
```

รูปที่ ช.1 ตัวอย่างเอกสารเอ็กเอ็มแอลที่ใช้กำหนดโครงการของการสร้างโครงภาษาจาวาจากข้อกำหนดการนับเลขที่มีสวิตช์ (ต่อ)

ภาคผนวก ซ
กราฟแสดงความสัมพันธ์ระหว่างวัตถุที่แสดงโดยเอกสารเอ็กเอ็มแอลของซอร์สโคดในภาคผนวก จ

เอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของโปรแกรมในภาคผนวก จ

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
  <Class name="SWITCH" modifier="public" cid="494">
    <attribute name="on" type="boolean" modifier="public static" value="true" atid="233"
/>
    <attribute name="off" type="boolean" modifier="public static" value="false"
atid="234" />
    <attribute name="Switch" type="boolean" modifier="public" atid="235" />
    <method name="SWITCH" return="" modifier="public" Eid="495">
      <statement name="Switch = off" sid="349">
        <Used>
          <value name="off" atid="234" />
        </Used>
        <Def>
          <value name="Switch" />
        </Def>
      </statement>
      <update name="Switch" />
    </method>
    <method name="State" return="boolean" modifier="public" Eid="496">
      <statement name="return Switch" FOid="496">
        <Used>
          <value name="Switch" atid="235" />
        </Used>
        <Def />
      </statement>
    </method>
    <method name="Off" return="void" modifier="public" Eid="497">
      <statement name="Switch = off" sid="350">
        <Used>
          <value name="off" atid="234" />
        </Used>
        <Def>
          <value name="Switch" />
        </Def>
      </statement>
      <update name="Switch" />
    </method>
```

รูปที่ ซ.1 เอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของโปรแกรม

```
    <method name="On" return="void" modifier="public" Eid="498">
     <statement name="Switch = on" sid="351">
      <Used>
       <value name="on" atid="233" />
      </Used>
      <Def>
       <value name="Switch" />
      </Def>
     </statement>
     <update name="Switch" />
    </method>
  </Class>
  <Class name="COUNTER_WITH_SWITCH" modifier="public" cid="499">
    <attribute name="cwsInt" type="int" modifier="public" atid="236" />
    <attribute name="cwsCounter" type="COUNTER" modifier="public" atid="237" />
    <attribute name="cwsSwitch" type="SWITCH" modifier="public" atid="238" />
    <method name="COUNTER_WITH_SWITCH" return="" modifier="public"
Eid="500">
     <statement name="cwsCounter = new COUNTER()" sid="352">
      <Used />
      <Def>
       <value name="cwsCounter" />
       <value name="counter" />
      </Def>
      <Call Eid="508" />
     </statement>
     <statement name="cwsSwitch = new SWITCH()" sid="353">
      <Used />
      <Def>
       <value name="cwsSwitch" />
       <value name="Switch" />
      </Def>
      <Call Eid="495" />
     </statement>
     <update name="cwsCounter" />
     <update name="cwsSwitch" />
    </method>
    <method name="Sub" return="void" modifier="public" Eid="501">
     <statement name="cwsSwitch.Off()" sid="354">
      <Used>
       <value name="cwsSwitch" atid="238" />
      </Used>
      <Def>
       <value name="cwsSwitch" />
       <value name="Switch" />
      </Def>
      <Call Eid="497" />
     </statement>
    </method>
```

รูปที่ ซ.1 เอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของโปรแกรม (ต่อ)

```
 <method name="Counter" return="COUNTER" modifier="public" Eid="502">
  <statement name="return cwsCounter" FOid="502">
   <Used>
    <value name="cwsCounter" atid="237" />
   </Used>
   <Def />
  </statement>
 </method>
 <method name="Read" return="int" modifier="public" Eid="503">
  <statement name="return cwsInt" FOid="503">
   <Used>
    <value name="cwsInt" atid="236" />
   </Used>
   <Def />
  </statement>
 </method>
 <method name="Add" return="void" modifier="public" Eid="504">
  <statement name="cwsSwitch.On()" sid="355">
   <Used>
    <value name="cwsSwitch" atid="238" />
   </Used>
   <Def>
    <value name="cwsSwitch" />
    <value name="Switch" />
   </Def>
   <Call Eid="498" />
  </statement>
 </method>
 <method name="Switch" return="SWITCH" modifier="public" Eid="505">
  <statement name="return cwsSwitch" FOid="505">
   <Used>
    <value name="cwsSwitch" atid="238" />
   </Used>
   <Def />
  </statement>
 </method>
```

รูปที่ ซ.1 เอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของโปรแกรม (ต่อ)

```
    <method name="Put" return="void" modifier="public" Eid="506">
      <parameter name="i" type="int" FI="59" />
      <statement name="if cwsSwitch.State() == SWITCH.on" sid="356" trueStart="357"
trueEnd="357" falseStart="358" falseEnd="358">
        <Used>
          <value name="cwsSwitch" atid="238" />
        </Used>
        <Def />
        <Call Eid="496" />
      </statement>
      <statement name="cwsCounter.Add(i)" underCondition="356" sid="357">
        <Used>
          <value name="cwsCounter" atid="237" />
          <value name="i" FI="59" methodParam="Y" />
        </Used>
        <Def>
          <value name="cwsCounter" />
          <value name="counter" />
        </Def>
        <Call Eid="510" />
      </statement>
      <statement name="cwsCounter.Add(-i)" underCondition="356" sid="358">
        <Used>
          <value name="cwsCounter" atid="237" />
          <value name="i" FI="59" methodParam="Y" />
        </Used>
        <Def>
          <value name="cwsCounter" />
          <value name="counter" />
        </Def>
        <Call Eid="510" />
      </statement>
    </method>
  </Class>
  <Class name="COUNTER" modifier="public" cid="507">
    <attribute name="zero" type="int" modifier="public static" value="0" atid="239" />
    <attribute name="counter" type="int" modifier="public" atid="240" />
    <method name="COUNTER" return="" modifier="public" Eid="508">
      <statement name="counter = zero" sid="359">
        <Used>
          <value name="zero" atid="239" />
        </Used>
        <Def>
          <value name="counter" />
        </Def>
      </statement>
      <update name="counter" />
    </method>
```

รูปที่ ซ.1 เอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของโปรแกรม (ต่อ)

```
    <method name="Read" return="int" modifier="public" Eid="509">
      <statement name="return counter" FOid="509">
        <Used>
          <value name="counter" atid="240" />
        </Used>
        <Def />
      </statement>
    </method>
    <method name="Add" return="void" modifier="public" Eid="510">
      <parameter name="i" type="int" FI="60" />
      <statement name="counter = counter + i" sid="360">
        <Used>
          <value name="counter" atid="240" />
          <value name="i" FI="60" />
        </Used>
        <Def>
          <value name="counter" />
        </Def>
      </statement>
      <update name="counter" />
    </method>
  </Class>
</Project>
```

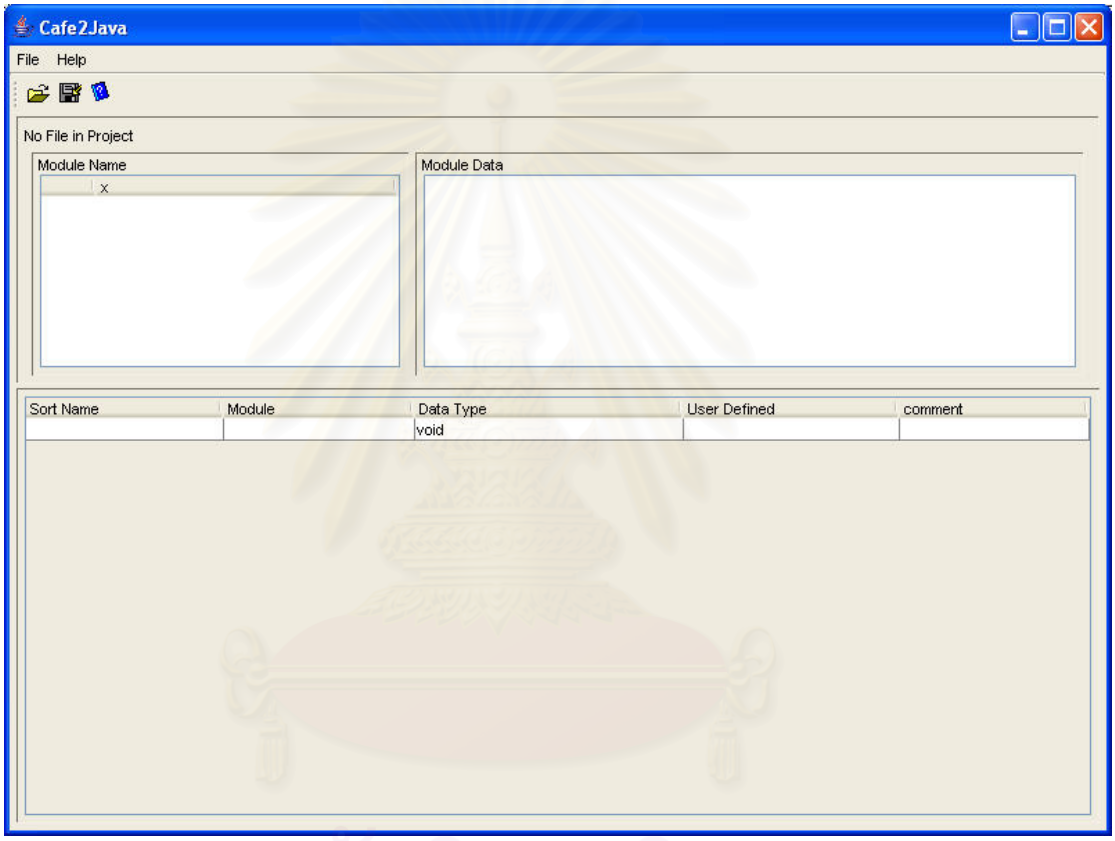
รูปที่ ซ.1 เอกสารเอ็กเอ็มแอลแสดงกราฟความสัมพันธ์อ้างอิงระหว่างวัตถุของโปรแกรม (ต่อ)

ภาคผนวก ฌ
คู่มือการใช้งานเครื่องมือ

เครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบ เป็นเครื่องมือสำหรับการสร้างโครงภาษาจาวาจากข้อกำหนดรูปนัยคาเฟโอบีเจ โดยมีวิธีการใช้งานเครื่องมือดังนี้

การเริ่มต้นใช้งานเครื่องมือ

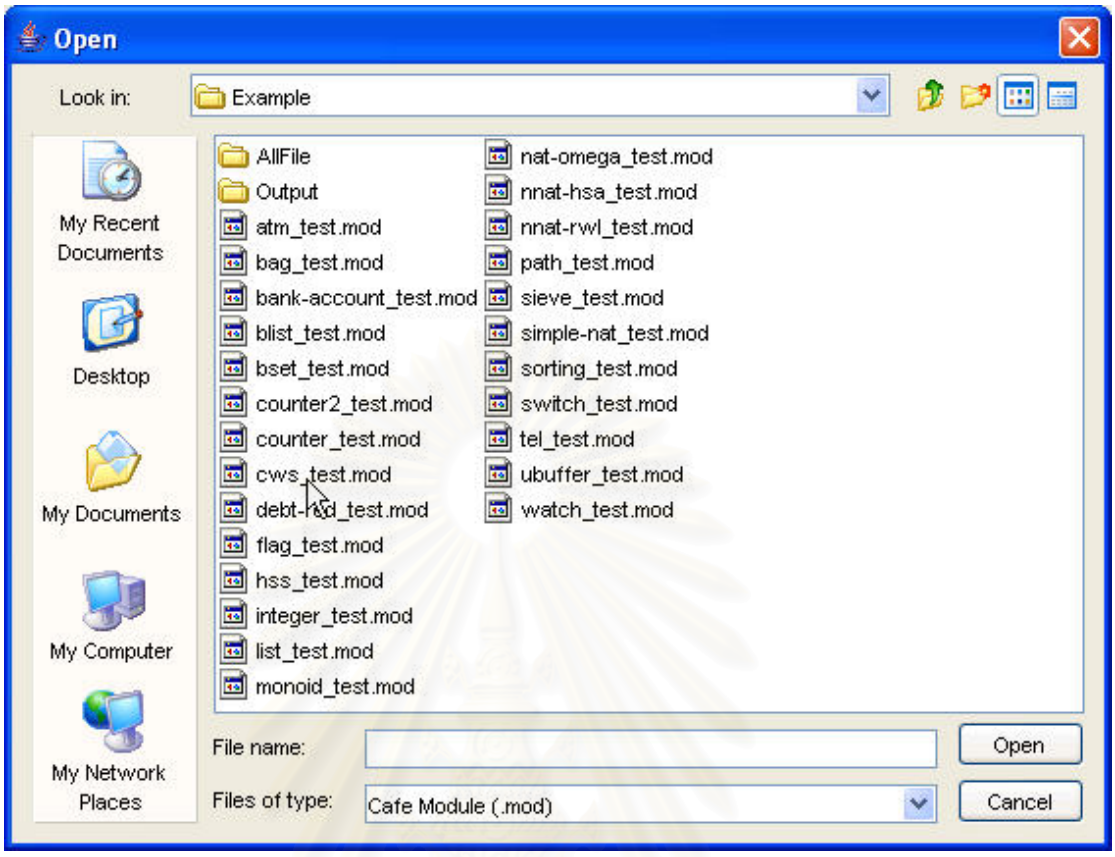
เรียกใช้เครื่องมือสร้างโครงภาษาจาวา เมื่อโปรแกรมพร้อมใช้งาน จะปรากฏหน้าจอดังรูปที่ ฌ.1



รูปที่ ฌ.1 หน้าจอเริ่มต้นการทำงานของเครื่องมือสร้างโครงภาษาจาวาและส่วนของภาษาจาวาที่คาดว่าจะพบ

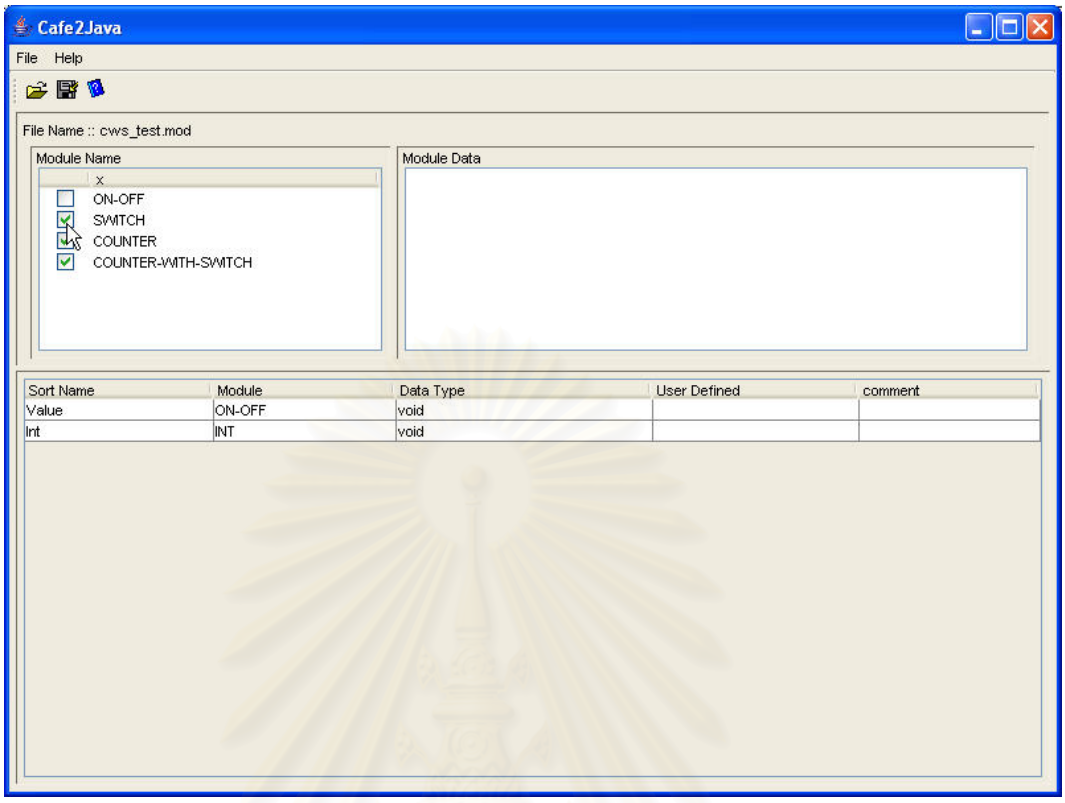
การสร้างโครงภาษาจาวา

ในการสร้างโครงภาษาจาวา ต้องเลือกไฟล์ซึ่งระบุข้อกำหนดรูปนัยคาเฟโอบีเจโดยเลือกที่ File -> New จะปรากฏหน้าจอดังรูปที่ ฌ.2

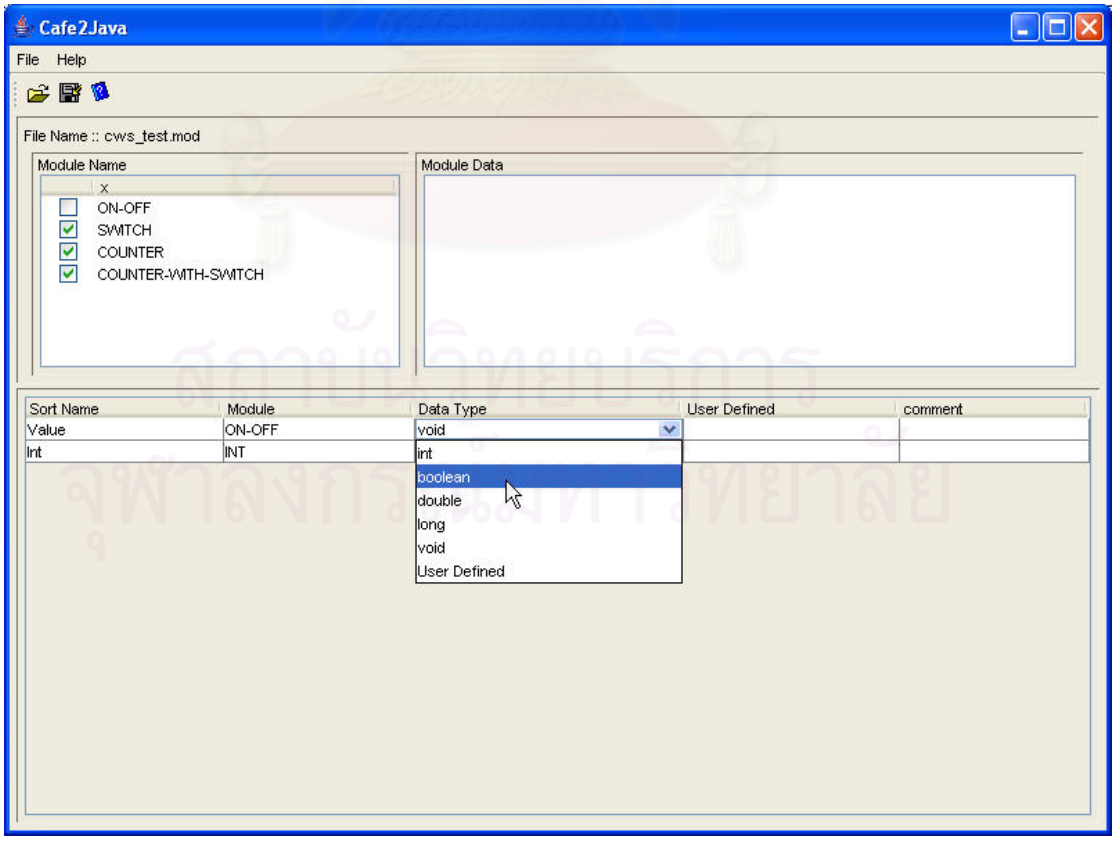


รูปที่ ฌ.2 หน้าจอเลือกข้อกำหนดรูปนัยคาเฟโอบีเจเพื่อสร้างโครงภาษาจาวา

เมื่อเลือกข้อกำหนดรูปนัยคาเฟโอบีเจแล้วจะปรากฏหน้าจอดังรูปที่ ฌ.3 หน้าจอนี้แบ่งเป็นสองส่วนได้แก่ส่วนแสดงมอดูลทั้งหมดในข้อกำหนดซึ่งแสดงในครึ่งบนของหน้าจอ และส่วนแสดงการเลือกชนิดของข้อมูลภาษาจาวากับชนิดข้อมูลสังเกตุค่าได้ของข้อกำหนดคาเฟโอบีเจซึ่งแสดงในครึ่งล่างของหน้าจอ โดยผู้ใช้เครื่องมือสามารถกำหนดมอดูลที่ต้องการสร้างเป็นคลาสโดยทำเครื่องหมายถูกหน้าชื่อมอดูล ดังแสดงในรูปที่ ฌ.3 และกำหนดชนิดของข้อมูลภาษาจาวาสำหรับชนิดข้อมูลแฝงทั้งหมดในข้อกำหนดรูปนัยคาเฟโอบีเจดังแสดงในรูปที่ ฌ.4

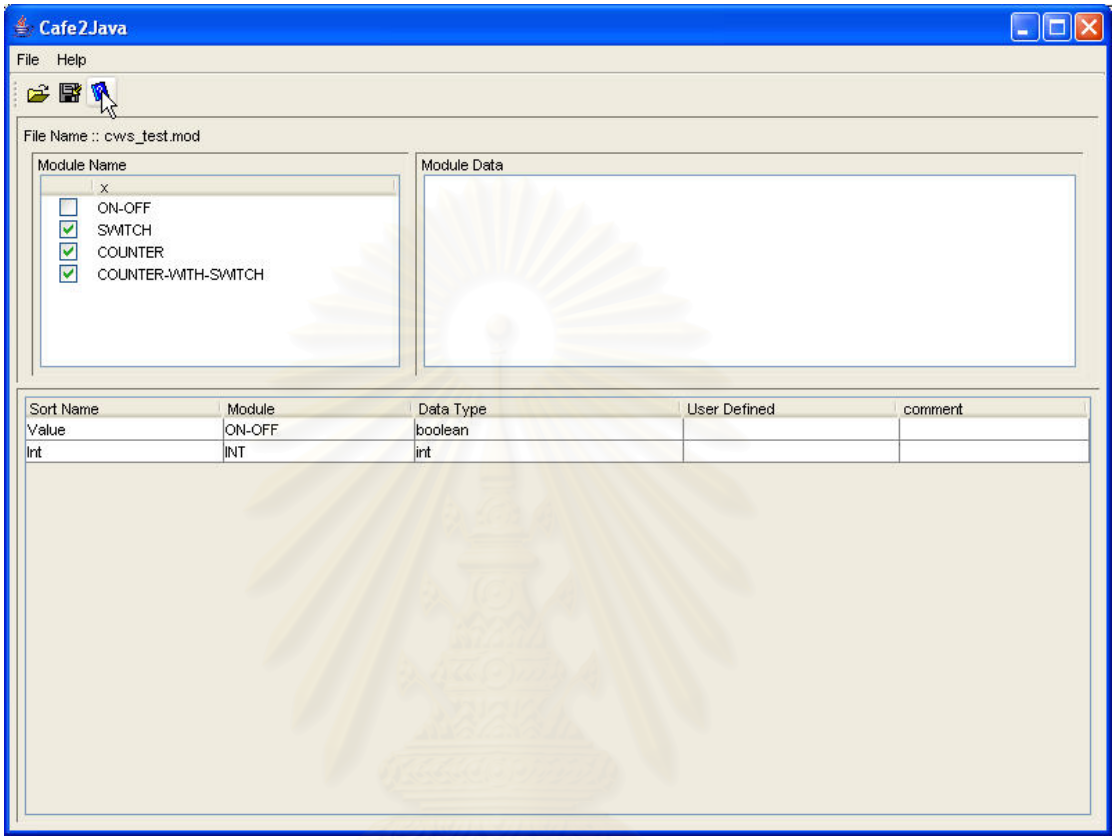


รูปที่ ฌ.3 การกำหนดมอดูลที่ต้องการสร้างเป็นคลาสในโครงภาษาจาวา



รูปที่ ฌ.4 การกำหนดชนิดของข้อมูลภาษาจาวาให้กับชนิดข้อมูลสังเกตค่าได้

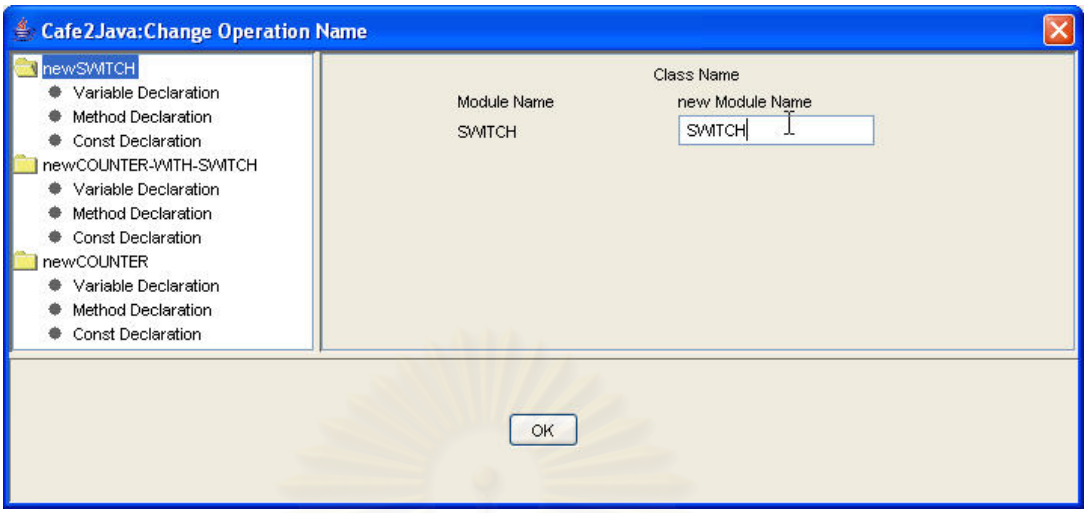
เมื่อกำหนดรายละเอียดขั้นต้นเรียบร้อยแล้วผู้ใช้กดปุ่มเพื่อทำงานในขั้นตอนถัดไปดังรูปที่ ฌ.5



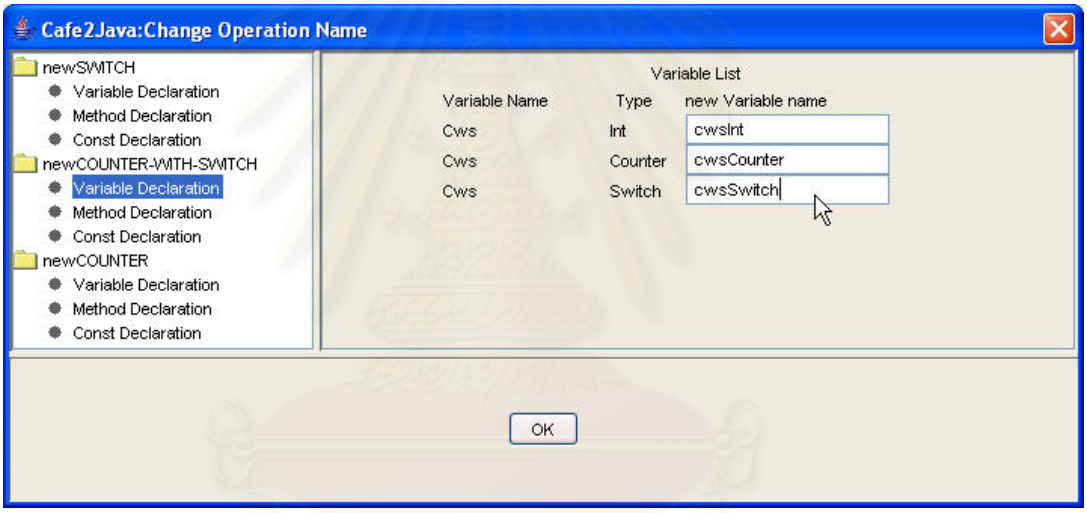
รูปที่ ฌ.5 แสดงการเลือกทำงานในขั้นตอนต่อไป

หน้าจอต่อมาเป็นหน้าจอสำหรับระบุรายละเอียดเฉพาะต่างๆ ประกอบไปด้วย

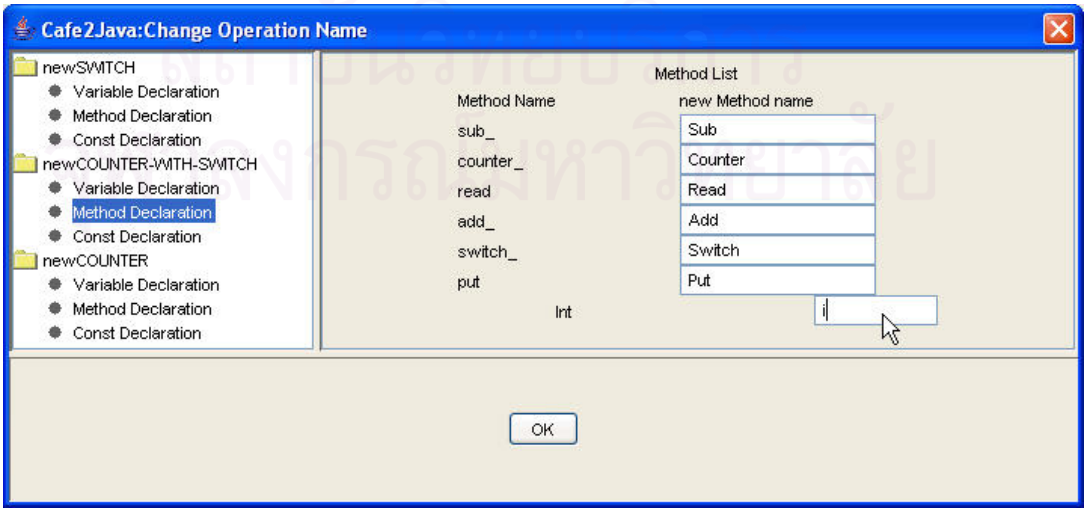
- การระบุชื่อคลาส (รูปที่ ฌ.6)
- การระบุชื่อตัวแปรภายในคลาสของโปรแกรมภาษาจาวา (รูปที่ ฌ.7)
- การระบุชื่อเมท็อดและส่วนนำเข้าของเมท็อดภายในคลาสของโปรแกรมภาษาจาวา (รูปที่ ฌ.8)
- การระบุชื่อค่าของคงที่และค่าของค่าคงที่ภายในคลาสของโปรแกรมภาษาจาวา (รูปที่ ฌ.9)



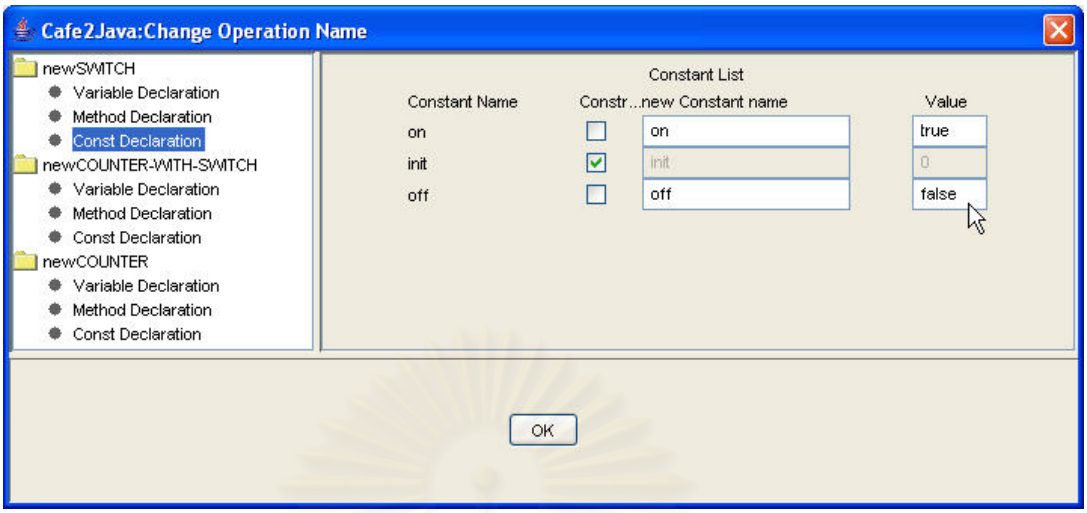
รูปที่ ฌ.6 หน้าจอแสดงการระบุชื่อคลาส



รูปที่ ฌ.7 หน้าจอแสดงการระบุชื่อตัวแปรภายในคลาส



รูปที่ ฌ.8 หน้าจอแสดงการระบุชื่อเมท็อดและชื่อส่วนนำเข้าของเมท็อด



รูปที่ ฌ.9 หน้าจอแสดงการระบุชื่อค่าคงที่และค่าของค่าที่

เมื่อระบุข้อมูลทุกส่วนครบถ้วนแล้ว ให้กดปุ่มตกลง (OK) เพื่อเตรียมข้อมูลสำหรับผลลัพธ์ของเครื่องมือโดยจะปรากฏหน้าจอดังรูปที่ ฌ.10

รูปที่ ฌ.10 หน้าจอแสดงตำแหน่งของผลลัพธ์จากการสร้างโครงภาษาจาวา

โดยหน้าจอนี้ต้องการให้ระบุ

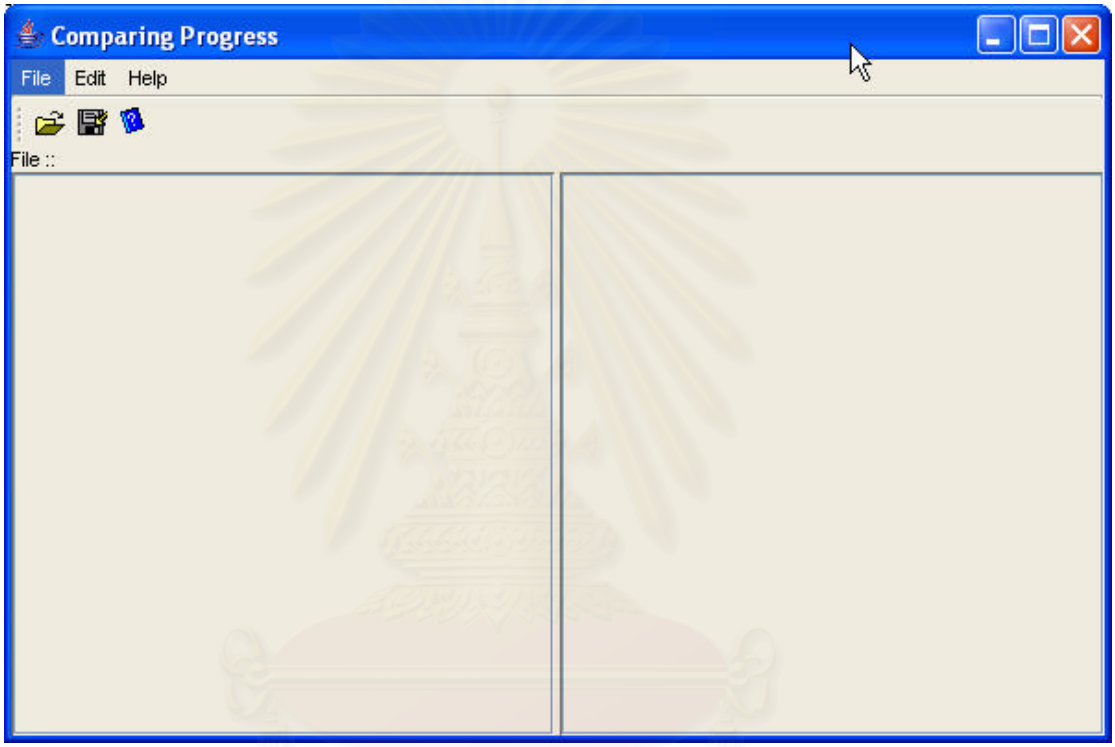
- ตำแหน่งแฟ้มของโครงภาษาจาวาที่สร้างขึ้น (Java source output directory)
- ตำแหน่งแฟ้มของส่วนของโครงภาษาจาวาที่คาดว่าจะพบ (XML Spec output Directory)
- ตำแหน่งของไฟล์เอกสารกำหนดโครงการ (Project file)

จากนั้นกดปุ่มตกลงเพื่อให้เครื่องมือสร้างโครงภาษาจาวาและส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบในตำแหน่งที่ระบุไว้ ตัวอย่างของโครงภาษาจาวาที่ได้แสดงในภาคผนวก ง และส่วนของโปรแกรมภาษาจาวาที่คาดว่าจะพบแสดงในภาคผนวก จ

เครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม เป็นเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรมซึ่งพัฒนามาจากโครงภาษาจาวาที่สร้างจากเครื่องมือภาษาจาวา มีวิธีการใช้งานเครื่องมือดังนี้

การเริ่มต้นใช้งานเครื่องมือ

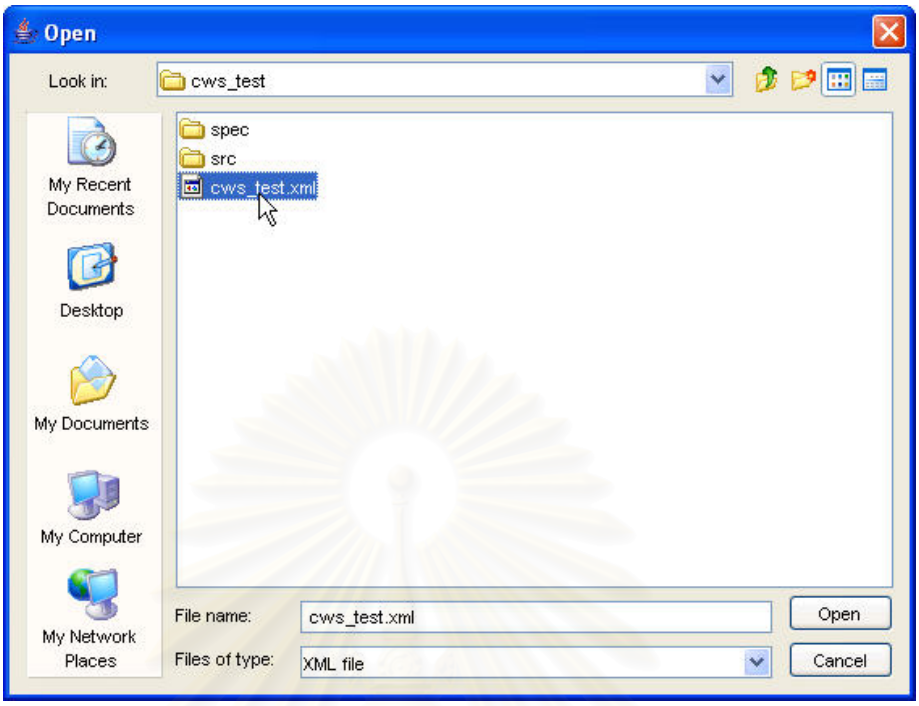
เรียกใช้เครื่องมือตรวจสอบความต้องการของการพัฒนาโปรแกรม เมื่อโปรแกรมพร้อมใช้งาน จะปรากฏหน้าจอดังรูปที่ ฌ.11



รูปที่ ฌ.11 หน้าจอเริ่มต้นการทำงานของเครื่องมือตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

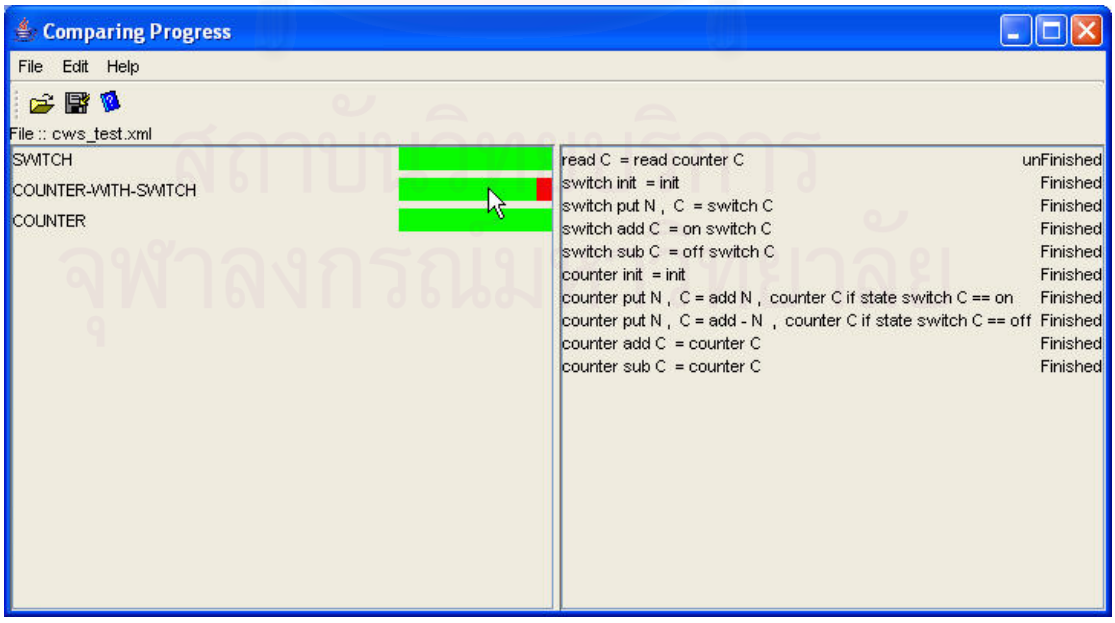
การตรวจสอบความก้าวหน้าของการพัฒนาโปรแกรม

ในการตรวจสอบโครงภาษาจาวา ต้องเลือกไฟล์เอกสารกำหนดโครงการโดยเลือก เมนู File -> New จะปรากฏหน้าจอดังรูปที่ ฌ.12



รูปที่ ฌ.12 หน้าจอแสดงการเลือกไฟล์ซึ่งระบุข้อมูลของการสร้างโครงภาษาจาวา

เมื่อเลือกไฟล์เอกสารกำหนดโครงการแล้ว เครื่องมือจะตรวจสอบโดยเปรียบเทียบจากไฟล์ที่อยู่ในแฟ้มของโครงภาษาจาวาที่กำลังพัฒนาซึ่งเก็บไว้ในแฟ้มข้อมูลของโครงภาษาจาวาที่สร้างขึ้นได้ และไฟล์ในแฟ้มของส่วนของภาษาจาวาที่คาดว่าจะพบ ซึ่งระบุไว้ในไฟล์ซึ่งระบุข้อมูลของการสร้างโครงภาษาจาวา และแสดงผลลัพธ์ดังรูปที่ ฌ.13 โดยซอร์สโคดที่นำมาตรวจสอบแสดงในภาคผนวก จ



รูปที่ ฌ.13 หน้าจอแสดงผลความก้าวหน้าที่เกิดขึ้น

โดยหน้าจอแสดงผลลัพธ์จากการเปรียบเทียบแบ่งออกเป็นสองด้าน ทางด้านซ้ายมือคือภาพรวมความก้าวหน้าที่เกิดขึ้นโดยจะแสดงในรูปแบบแท่งความก้าวหน้า (Progress Bar) โดยสีเขียวแสดงจำนวนสมการที่ได้ตรวจสอบว่าได้รับการพัฒนาแล้ว ขณะสีแดงคือจำนวนสมการที่ยังไม่ได้รับการพัฒนา ทางด้านขวามือเป็นรายละเอียดของสมการในแต่ละมอดูล ผู้ใช้งานสามารถเลือกตรวจสอบภายในแต่ละมอดูล ว่าสมการใดบ้างที่ตรวจสอบพบกว่าได้พัฒนาตามความต้องการแล้ว โดยสมการที่ตรวจสอบว่าได้รับการพัฒนาแล้วจะแสดงด้วยคำว่า “Finished” ในขณะที่สมการที่ตรวจสอบว่ายังไม่ได้รับการพัฒนาจะแสดงด้วยคำว่า “unFinished”

ภาคผนวก ญ
ผลงานตีพิมพ์ในงาน SERP'03

An Automatic Approach to Transform CafeOBJ Specifications to Java Template Code

C. Doungsa-ard and T. Suwannasart

Department of Computer Engineering

Faculty of Engineering, Chulalongkorn University

Bangkok, 10330, Thailand

Abstract

A software specification is a fundamental work product that represents user's requirements and developers can use it to further develop a software system. A software specification can be expressed by using an informal, a semi-formal, or a formal specification. Existing methodologies and tools can transform semi-formal specifications to code or programs but they have not addressed a transformation of formal specification languages. In this paper, we propose an automatic approach to transform CafeOBJ specifications to Java template code. The proposed approach consists of steps and transformation rules. We have implemented a tool called Cafe2Java to illustrate our proposed approach. We also validate the approach and experience the tool with some CafeOBJ specifications.

Keywords: CafeOBJ, Transforming approach, Formal specification, transformation rule.

1. Introduction

In software development process, the requirement phase is the phase that developers specify user's requirements. Normally, software developers can express user's requirements in any forms of specification which include informal specification, semi-formal specification, or formal specification [1]. An informal specification can be specified by natural language while we use diagrams such as UML [2] diagrams to represent user's requirements as a semi-formal specification. For a formal specification, we use mathematical notations to specify the requirements. In developing critical software or some embedded systems, developers use a formal specification language as a tool to define the requirements since we can prove if the specification is correct

and works as specified. There are numbers of formal specification languages used in software development such as CafeOBJ [3, 4], Z [5], and Object Z [6].

Currently, some commercial tools such as Rational Software [7], can transform a semi-formal specification specified by a class diagram to code such as Java code. The code generated from commercial tools consists of attributes and operations of class but it does not include other implementation details such as behavioural details for each class. We believe that if the generated code provides more details, it will help developers decrease development effort. Thus, in this paper we propose an approach to transform a formal specification language: CafeOBJ to Java template code. Java template code is a framework that developers can use it to add more implementation details. We also have implemented our approach by developing a software tool called "Cafe2Java".

In section 2, we give an overview of the CafeOBJ. An approach to transform CafeOBJ to Java Template Code is presented in section 3. Section 4 shows the implementation of the software tool that we have developed. In section 5, we show an example of the result. Finally, section 6 gives the conclusion of the paper with limitations and future work.

2. CafeOBJ

CafeOBJ is an algebraic specification language that is a direct successor of OBJ [3]. CafeOBJ supports modular specification technique. A CafeOBJ specification is written into modules. Each module consists of three main parts: sort declaration, operation declaration, and axiom declaration. In the sort declaration part, definitions of all sorts (types of objects) are declared. CafeOBJ has two types of sorts: visible sort and hidden sort. The visible sort is a sort that is visible to all modules in the system while the hidden sort stores states of the module and hide its values from other modules. As CafeOBJ supports modular specification technique, all related modules used in a module can be imported in this part as well. All possible operations are defined in the operation declaration part. There are two types of operation: op and bop (see Figure 1). "op" is an operation while "bop" is a behavioural operation. For each line in this part, an

operation name with its arity and coarity is defined. The arity is defined as an input of the operation whereas the coarity is an output. In Figure 1,“status” in module ON-OFF is a behavioural operation, where “Switch” is an arity and “Value” is a coarity. The axiom declaration part is used for defining behavioural properties or states of a module of each operation appeared in the operation part. Moreover, we can call the axiom declaration as the semantic part.

In this paper, we classify operations into three types: projection operation, hidden sort projection operation, and non-arity operation. The projection operation is an operation that has a hidden sort as arity and a visible sort as coarity. The hidden sort projection operation is an operation that its hidden sort in its coarity is not the same sort as hidden sort defined in its arity. For example, in Figure 1 the operation “counter” in COUNTER-WITH-SWITCH (CWS) module is defined as a hidden sort projection operation. “Counter” is defined as a hidden sort in its coarity whereas “Cws” is a hidden sort in its arity. “Counter” must not be the same sort as “Cws”. The non-arity operation is an operation that does not have any arity sorts but it has coarity sorts which can be either a visible sort or a hidden sort. In Figure 1, we show three types of operation in module CWS: “read” is a projection operation, “counter” is a hidden sort projection operation, and “init” is a non-arity operation. Figure 1 shows an example a CafeOBJ specification: CWS. This specification consists of four modules: ON-OFF, SWITCH, COUNTER, and CWS. ON-OFF module declares a visible sort: Value, which has two possible values: “on” and “off”. SWITCH module which imports ON-OFF module has a hidden sort: Switch, which is declared in the sort declaration part. The operation declaration part defines all operations in the SWITCH module together with its inputs and outputs of each operation. The axiom declaration part, all of the equations are defined to represent possible states changed in the Switch when each operation is called.

3. Transforming CafeOBJ specifications to Java template code

In this section we discuss a proposed approach to transform CafeOBJ specification to a Java template code. A Java template code is a java code that consists

of a class name, method names, a return type of each method, and constant values as well as behavioural variables of the class. Our generated Java Template Code does not include the semantic part or implementation details of Java code, specified in the axiom declaration part of a CafeOBJ specification. We leave this part for developers to add more details by themselves.

The approach we propose to transform a CafeOBJ specification to Java template code is shown in Figure 2. The proposed approach consists of three steps: Graph generation, Graph Reduction, and Java Template Code generation.

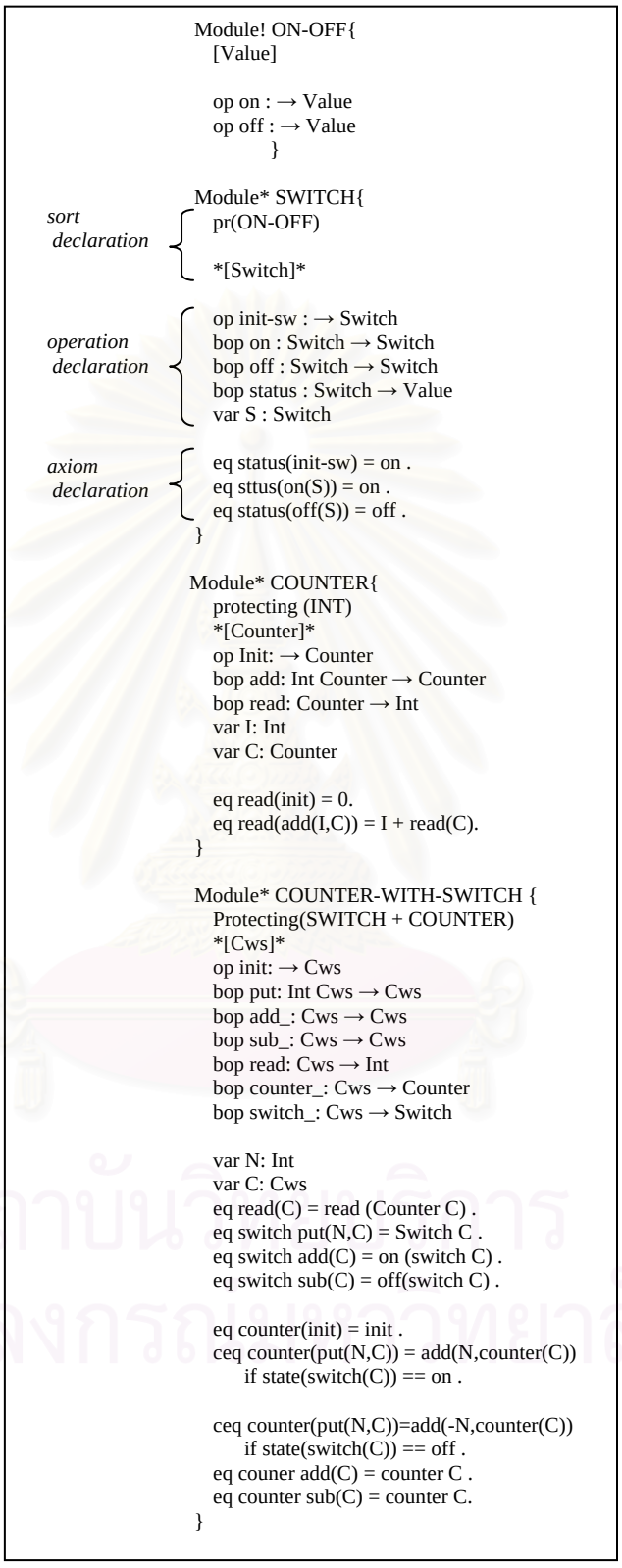


Figure 1. A CafeOBJ specification: COUNTER-WITH-SWITCH (CWS)

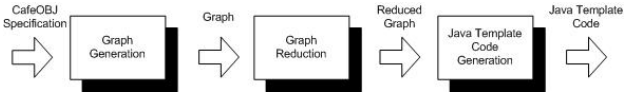


Figure 2 Transforming Approach

3.1 Graph generation

The graph generation step transforms a CafeOBJ specification to a graph structure. A graph structure represents a relationship among modules in a CafeOBJ specification. Each node in a graph represents each module in CafeOBJ specification. The graph structure of CWS is shown in Figure 3.

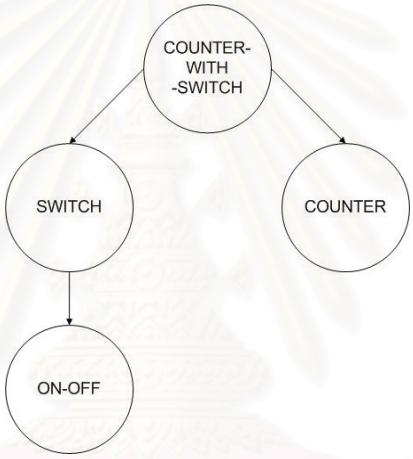


Figure 3 COUNTER-WITH-SWITCH graph structure

3.2 Graph Reduction

After a graph has been constructed, we have to reduce the graph in order to specify which module in a CafeOBJ specification will be transformed. The reduced graph contains nodes corresponding to classes, which developers want to transform them to Java template code. In the graph reduction step, developers have to select which nodes that are needed to be reduced.In the reducing process, a graph is traversed using the depth first search algorithm. If a node, which is selected to be built as a node in the reduced graph, is traversed, it will be checked to see if there is a child node. If a child node is not selected to be built as a node in the reduced graph as well, the child node will be reduced to be included in its parent node.

3.3 Java Template Code Generation

In this step, each node in the reduced graph is transformed to a Java class. To transform a node in the reduced graph we have to follow the defined processes below:

1. Define a Java data type for each visible sort specified in each CafeOBJ specification.

2. Transform operations in the declaration operation part for the operation declaration part in each node.

Since we classify types of operations into three types as described earlier in section 2, we have to check if each operation is in which types. For each operation we define transformation rules as follows:

For

p is an operation in CafeOBJ

s is a hidden sort

v is a visible sort

h is the other hidden sort, which is not **s**

t is any types of sorts

$\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ are any hidden sorts

$\mathbf{v}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ are any visible sorts

n and **m** are positive numbers

Rule 1 for a projection function; op **p**: **s** ⟶ **v**

The transformation to Java template code is below:

```
v p(){
}
```

where

p is a method name related to **p**

v is a data type in Java related to **v**, defined in process 1

Furthermore, in this rule we can define a data type for the hidden sort. The hidden sort's data type must be the same data type as the visible sort.

Rule 2 for a hidden sort projection function;

op **p**: **s** ⟶ **h**

The transformation to Java template code is below:

h p(){

}

where

p is a method name related with **p**

h is a data type in Java related with **h**, defined in the first process or a class which is generated from a node which **h** has been declared

Rule 3 for a non-arity operation; op **p**: ⟶ **t**

If a non-arity operation is selected to be an initial state of a node,

No code transformation from this operation to Java template code.

else

the transformation to Java template code is below:

public t p = a;

where

p is a constant name related with **p**

t is a data type related to **t**

a is a value of constant define by developers

Rule 4 for other operations, we have to check if an operation is one of any operation as follows

Rule 4.1 for; op **p**: $\mathbf{s}_1\ \mathbf{s}_2 \ldots \mathbf{s}_n \longrightarrow \mathbf{v}$

The transformation to Java template code is below:

public v p(){

}

where

p is a method name related to **p**

v is a data type in java related to **v**, defined in the first process

Rule 4.2 for; op p: $s_1\ s_2 \ldots s_n\ v_1\ v_2 \ldots v_m \longrightarrow v$

The transformation to Java template code is below:

public v p($v_1\ n_1$,$v_2\ n_2$,…,$v_m\ n_m$){

}

where

p is a method name related to **p**

$v,v_1,v_2,\ldots v_m$ are data types in Java related to $\mathbf{v},\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n$

$n_1,n_2,\ldots,n_m$ are any parameter names which correspond to the Java naming convention

Rule 4.3 for; op **p**: $\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_m \longrightarrow \mathbf{v}$

The transformation to Java template code is below:

public v p(v_1 n_1,v_2 n_2,…,v_m n_m){

}

where

p is a method name related with **p**

$v,v_1,v_2,\ldots v_m$ are data types in Java related to $\mathbf{v},\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n$

$n_1,n_2,\ldots,n_m$ are any parameter names which correspond to the Java naming convention

Rule 4.4 for; op **p**: $\mathbf{s}_1\ \mathbf{s}_2\ \ldots\ \mathbf{s}_n \longrightarrow \mathbf{s}$

The transformation to Java template code is below:

public void p(){

}

where

p is a method name related to **p**

Rule 4.5 for; op p: $s_1\ s_2\ \ldots\ s_n\ v_1\ v_2\ \ldots\ v_m \longrightarrow s$

The transformation to Java template code is below:

public void p(v_1 n_1,v_2 n_2,…,v_m n_m){

}

where

p is a method name related with **p**

$v,v_1,v_2,\ldots v_m$ are data types in Java related to $\mathbf{v},\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n$

$n_1,n_2,\ldots,n_m$ are any parameter names which correspond to the Java naming convention

Rule 4.6 for; op **p**: $\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_m \longrightarrow \mathbf{s}$

The transformation to Java template code is below:

public void p(v_1 n_1,v_2 n_2,…,v_m n_m){

}

where

p is a method name related to **p**

$v,v_1,v_2,\ldots v_m$ are data types in Java related to $\mathbf{v},\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n$

$n_1,n_2,\ldots,n_m$ are any parameter names which correspond to the Java naming convention

In this step, we transform hidden sorts in arity and coarity into void because variables represented for the hidden sort is already in the class, there is no need to send or receive this information outside the class by any method.

3. Transform the sort declaration part

3.1 for a hidden sort declaration: *[**x**]*

where

x is the hidden sort name

The transformation to Java template code is below:

private a **x**;

where

a is a data type in Java related to **x**, defined in the first process

x is a variable name related to **x**

3.2 for a module import declaration; pr (**x**)

where

x is the module name

If **x** has been transformed to a class in Java template code,

The transformation to Java template code is below:

private *x a*;

where

x is a class name which **x** has been transformed to

a is a variable name for **x**

4. Implementation

To illustrate the feasibility of our approach, we have implemented a tool – "Cafe2Java" to support the transformation steps proposed in Section 3. First, developers can identify a CafeOBJ specification that needs to be transformed. The tool will show modules of the identified specification at the upper-left panel shown in Figure 4 Developers can select numbers of modules to be transformed to Java template code. A selected CafeOBJ specification module is visualized in the upper-right panel. The lower panel shows all visible sorts that developers have to specify Java data types for them.

After that, the transformation steps will be automatically processed. Besides, developers can specify a constant value or variables, and methods names in the generated Java template code.

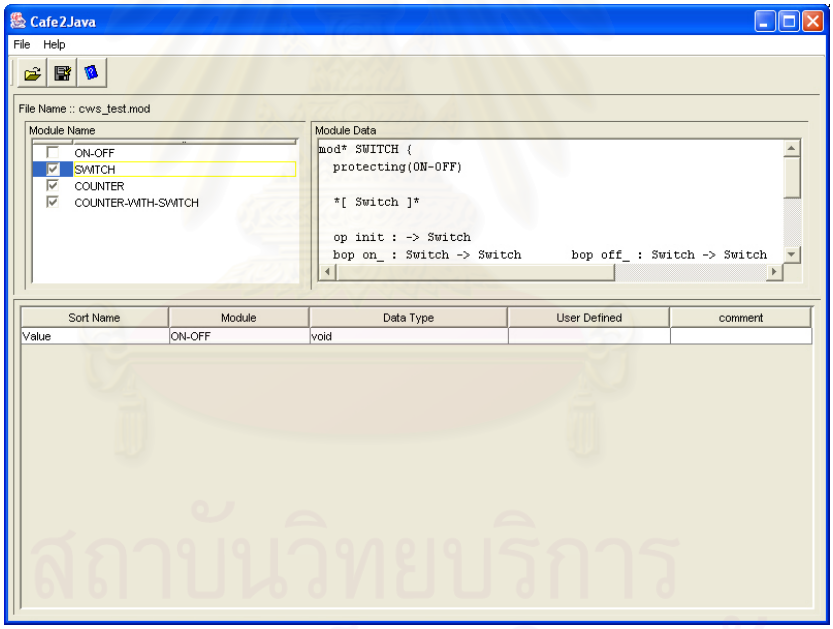


Figure 4 Cafe2Java Interface

```
public class newSwitch{

    public boolean Off = false;
    public boolean On = true;
    public boolean Switch;
    public newSwitch()
    {

    }
    public void on( )
    {
      /**@todo: Implement this on method*/
      throw new java.lang.UnsupportedOperationException("Method on( ) not yet implemented.");
    }
    public void off( )
    {
      /**@todo: Implement this off method*/
      throw new java.lang.UnsupportedOperationException("Method off( ) not yet implemented.");
    }
    public boolean status( )
    {
      /**@todo: Implement this status method*/
      throw new java.lang.UnsupportedOperationException("Method status( ) not yet implemented.");
    }
}
```

Figure 5-SWITCH". This specification consists of four modules: ON-OFF, COUNTER, SWITCH, and COUNTER-WITH-MODULE. The transformation result from our approach is shown in Figure 5.

6. Conclusion and future work

In this paper, we propose an approach to transform CafeOBJ specifications to Java template code. The approach that we proposed includes 3 steps: graph generation, graph reduction, and Java template code generation. In the Java template code generation step, we define rules for the transformation. However, the transformation rules have not supported other types of modules: parameterized modules and view modules [3] in CafeOBJ specifications. To illustrate the proposed approach, we have developed a tool that supports those transformation steps. Even though the developed tool can automate the transformation steps but developers who use this tool have to manually specified Java data type of each visible sort identified in CafeOBJ specifications.

The Java template code generated by our approach is not completed Java code due to lacking of implementation details or the axiom part specified in a CafeOBJ specification but the developers can use it as a framework for further implementation. In the future, we plan to extract the Java code developed from the Java template code to

see if the developers follow the specification. Since the size of CafeOBJ specifications, we used to test our tool, are not large and complex, so we plan to experience the proposed approach and tool with the larger and more complex modules in the future.

7. Acknowledgements

This work is partly supported by Thailand-Japan Technology Transfer Project (TJTTP-OECF) and Chulalongkorn University-Industry Linkage Research Fund. The authors gratefully acknowledge Prof. Dr. Koichiro Ochimizu from JAIST (Japan Advanced Institute of Science and Technology) for some discussions earlier.

8. References

1. Harry, A., *Formal methods Factfile*. 1 ed. 1996, Chichester, England: John Wiley & Sons. 386.

2. OMG, OMG Unified Modeling Language Specification version 1.4. 2001: OMG.

3. Diaconescu, R. and K. Futasugi, *CafeOBJ Report*. 1998: World Scientific.

4. Nakagawa, A.T., T. Sawada, and K. Futasugi, *CafeOBJ User's Manual version 1.4*. 1998: World Scientific.

5. Ben Potter, J.S., David Till, *An Introduction to Formal Speicification and Z*. 1 ed. Prentice Hall International Series in Computer Science, ed. C.A.R. Hoare. 1991, Hertfordshire: Prentice Hall International (UK)Ltd. 304.

6. Duke, R., et al. The Object-Z Specification Language. in In T. Korson, V. Vaishnavi, & B. Meyer (Eds.), Technology of Object-Oriented Languages and Systems TOOLS 5. Proc. of the 5th International Conf. 1994. 465-483.

7. Quatrani, T., *Visual Modelling with Rational Rose and UML*. The Addison-Wesley Object Technology, ed. G. Booch, I. Jacobson, and J. Rumbaugh. 1998: Addison Wesley Longman, Inc. 222.

ประวัติผู้เขียนวิทยานิพนธ์

นายชาติชาย ดวงสอาด เกิดเมื่อวันที่ 28 เมษายน พ.ศ. 2522 ที่จังหวัดเชียงใหม่ จบหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ เกียรตินิยม อันดับ 2 ภาควิชาวิศวกรรมคอมพิวเตอร์ จากคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2543 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2544