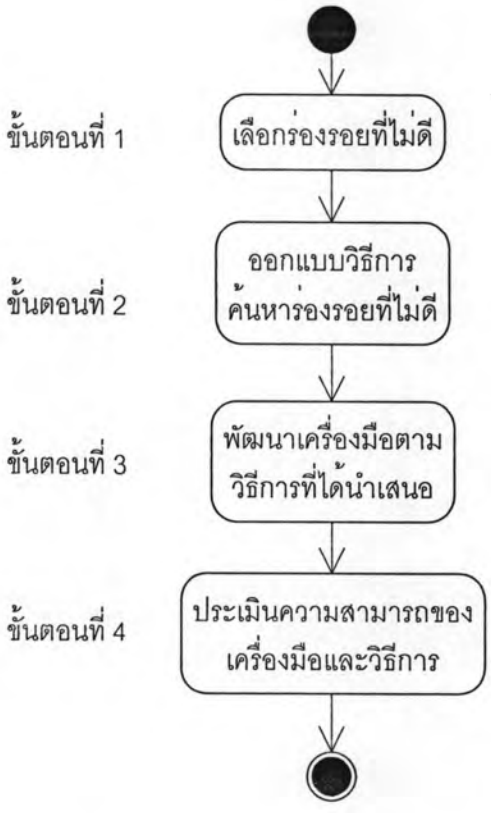
บทที่ 3

วิธีการค้นหาร่องรอยที่ไม่ดี

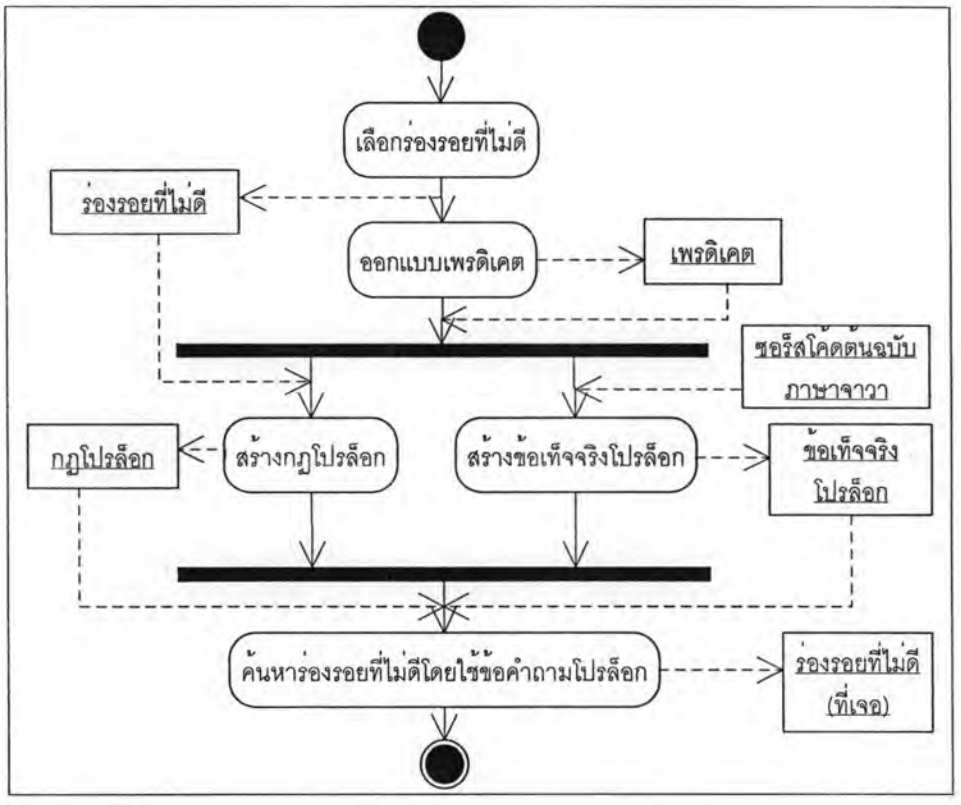
งานวิจัยนี้ได้แบ่งขั้นตอนการทำงานออกเป็น 4 ขั้นตอน ได้แก่ เลือกร่องรอยที่ไม่ดี ออกแบบวิธีการค้นหาร่องรอยที่ไม่ดี ออกแบบ และ พัฒนาเครื่องมือค้นหาร่องรอยที่ไม่ดี และทดสอบความน่าเชื่อถือของเครื่องมือ โดยสามารถแสดงด้วยแผนภาพแอคทิวิตี (Activity Diagram) ดังแสดงในรูปที่ 3.1 ในบทนี้จะอธิบายถึงขั้นตอนที่ 1 และ 2 ซึ่งเป็นรายละเอียดเกี่ยวกับวิธีการค้นหาร่องรอยที่ไม่ดี ส่วนขั้นตอนที่ 3 และ 4 รายละเอียดจะอยู่ในบทที่ 4 และ 5 ตามลำดับ



รูปที่ 3.1 แผนภาพขั้นตอนการวิจัย

การค้นหาร่องรอยที่ไม่ดีที่มีอยู่ในซอร์สโค้ดในงานวิจัยนี้จะใช้วิธีการพิจารณาจากโครงสร้างของโปรแกรม โดยอาศัยความรู้พื้นฐาน คือ ตรรกะเพรดิเคต กฎการอนุมาน และโปรล็อก

ตรรกะเพรดิเคตจะใช้แทนความรู้ของความสัมพันธ์ และ องค์ประกอบในภาษาจาวา จากนั้นจะเอาความสัมพันธ์เหล่านี้มาสร้างเป็นกฎของโปรล็อก และ จะใช้โปรล็อกในการค้นหาร่องรอยที่ไม่ดี โปรล็อกเป็นภาษาคอมพิวเตอร์ชนิดหนึ่งที่ใช้ในการแก้ปัญหาทางด้านสัญลักษณ์ โดยใช้พื้นฐานของตรรกะเพรดิเคต ซึ่งจะมีจุดเด่น คือ การจัดการเกี่ยวกับเรื่องของความสัมพันธ์ของสัญลักษณ์ต่างๆ จากข้อดีตรงจุดนี้เองที่ทำให้ผู้วิจัยเลือกที่จะใช้โปรล็อกในการแก้ปัญหาในเรื่องนี้ ขั้นตอนการค้นหาร่องรอยที่ไม่ดี แสดงดังรายละเอียดในรูปที่ 3.2



รูปที่ 3.2 แผนภาพขั้นตอนการค้นหาร่องรอยที่ไม่ดี

ขั้นตอนการค้นหาร่องรอยที่ไม่ดีในรูปที่ 3.2 จะแบ่งออกเป็น 5 ส่วนด้วยกัน คือ การเลือกร่องรอยที่ไม่ดี การออกแบบเพรดิเคต การสร้างกฎโปรล็อกจากร่องรอยที่ไม่ดี การสร้างข้อเท็จจริง และการค้นหาร่องรอยที่ไม่ดีโดยใช้ข้อคำถามโปรล็อก ซึ่งมีรายละเอียดดังนี้

3.1 เลือกร่องรอยที่ไม่ดี

ในงานวิจัยนี้จะนำเสนอวิธีการค้นหาร่องรอยที่ไม่ดี 4 ประเภท คือ Feature Envy, Message Chains, Middle Man, และ Inappropriate Intimacy (General Form) เนื่องจากวิธีการค้นหาร่องรอยที่ไม่ดีที่ผู้วิจัยนำเสนอเป็นการใช้หลักการของภาษาโปรล็อก ซึ่งเป็นการหาคำตอบจากสัมพันธ์ที่มีอยู่ ดังนั้น ร่องรอยที่ไม่ดีทั้ง 4 ประเภทนี้จึงมีเหมาะสมกับวิธีการที่ผู้วิจัยได้นำเสนอมากที่สุด

3.2 ออกแบบเพรดิเคต

เป็นการนำรายละเอียดของการออกแบบ และ การเขียนโปรแกรมภาษาจาวามาวิเคราะห์เพื่อหาความสัมพันธ์ขององค์ประกอบที่มีอยู่ในโปรแกรม ซึ่งจะแทนโปรแกรมเชิงวัตถุด้วยสัญลักษณ์ต่างๆ ดังนี้

P = (V, R, Ø, n) โดย

V = Class U Field U Method U Attribute U Expression U Field Access

R $\subseteq$ Class X Field U Class X Method U Method X Attribute U Method X Expression X Method X Field Access

Ø(v): V $\rightarrow$ รายละเอียดขององค์ประกอบ เช่น

Ø(c) แทน Class $\rightarrow$ Name X Begin Line X End Lind

Ø(f) แทน Field $\rightarrow$ Name X Type X Begin Line X End Line

Q(m) แทน Method $\rightarrow$ Name X Return Type X Parameter List X Begin Line X End Line

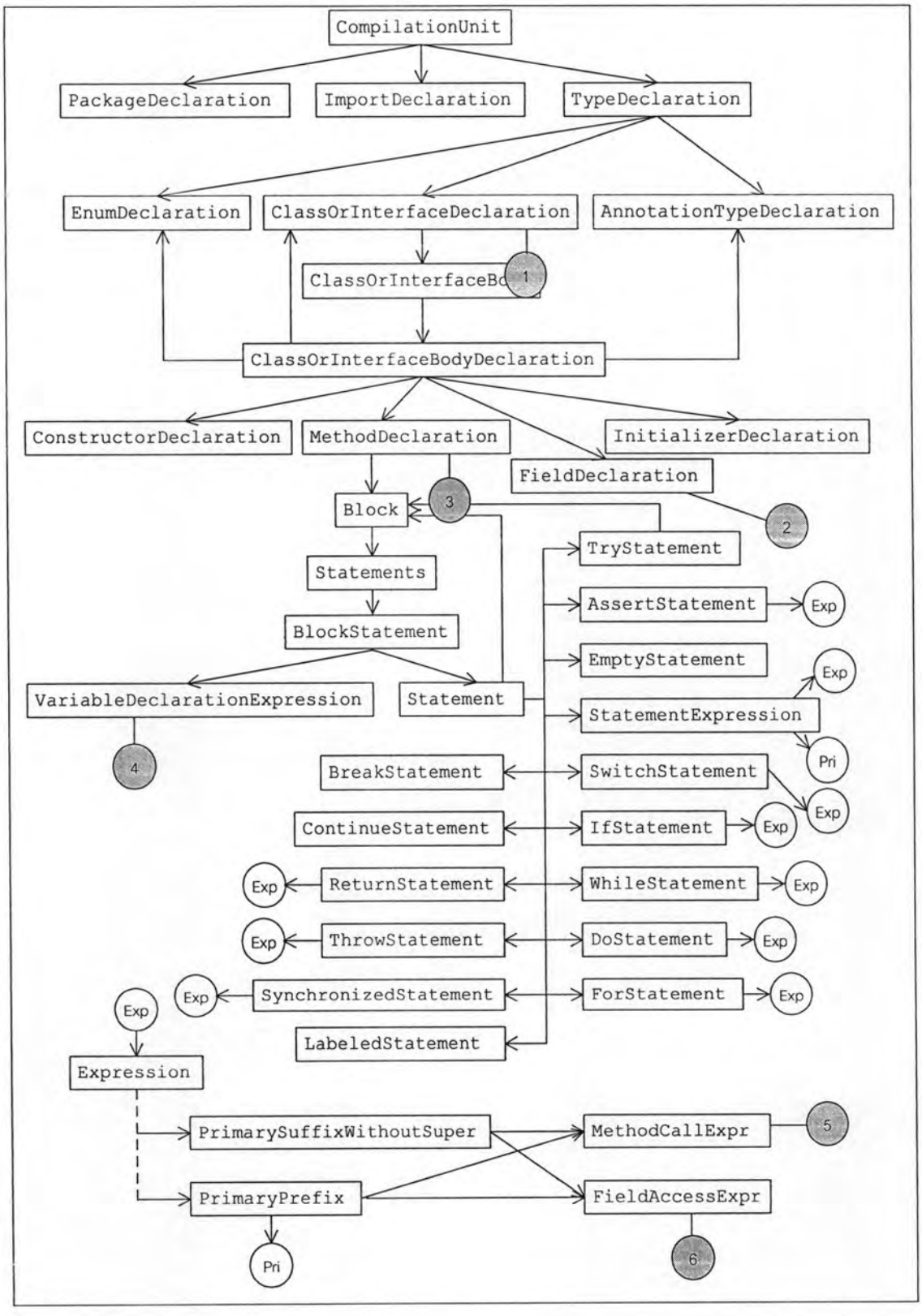
Q(a) แทน Attribute $\rightarrow$ Name X Type X Begin Line X End Line

Q(e) แทน Expression $\rightarrow$ Name X Arguments List X Scope X Begin Line X End Line

Q(fa) แทน FieldAccess $\rightarrow$ Name X Scope X Begin Line X End Line

n: R $\rightarrow$ Text

Text คือ ชนิดของความสัมพันธ์ เช่น owns, calls เป็นต้น



รูปที่ 3.3 แผนภาพต้นไม้แสดงองค์ประกอบของภาษาจาวา ที่ได้มาจากการสร้างซินแท็กซ์ทรีของคลาสจาวาพาร์เซอร์

ออกแบบเพรดิเคต โดยการนิยาม และ แปลความหมายของแต่ละเพรดิเคต โดยใช้สัญลักษณ์ที่ได้ออกแบบไว้ข้างต้น และ รายละเอียดของโปรแกรมภาษาจาวาตาม รูปที่ 3.3 เพรดิเคตที่สร้างขึ้นเป็นการหาได้โดยตรงจากโปรแกรมภาษาจาวา ซึ่งมีรายละเอียดดังนี้

จากรูปที่ 3.3 โหนดที่ (1) คือ คลาส ClassOrInterfaceDeclaration คุณลักษณะที่ต้องใช้ในสัญลักษณ์ Ø(c) คือ name, line และ token.endLine หมายถึง ชื่อคลาส บรรทัดเริ่มต้น และ บรรทัดสุดท้าย ตามลำดับ แต่ต้องตรวจสอบค่าของคุณลักษณะ isInterface ก่อนเสมอ เพราะคุณลักษณะนี้จะบอกว่าองค์ประกอบที่ส่งเข้ามาเป็น Interface หรือ Class ซึ่งถ้าองค์ประกอบเป็น Class คุณลักษณะ isInterface จะให้ค่าความจริงเป็น False ดังนั้น เมื่อนำมาสร้างเป็นเพรดิเคต จะได้เป็นสูตรที่ 1 ตามตารางที่ 3.1

โหนดที่ (2) คือ คลาส FieldDeclaration คุณลักษณะที่ต้องใช้ในสัญลักษณ์ Ø(f) คือ variables, type, line และ token.endLine หมายถึง ชื่อคุณลักษณะ ชนิดของคุณลักษณะ บรรทัดเริ่มต้น และ บรรทัดสุดท้าย ตามลำดับ เมื่อนำมาสร้างเป็นเพรดิเคต จะได้เป็นสูตรที่ 2 ตามตารางที่ 3.1

โหนดที่ (3) คือ คลาส MethodDeclaration คุณลักษณะที่ต้องใช้ในสัญลักษณ์ Ø(m) คือ name, type, parameters, line และ token.endLine หมายถึง ชื่อเมทธอด ชนิดข้อมูลของเมทธอด พาราเมเตอร์ บรรทัดเริ่มต้น และ บรรทัดสุดท้าย ตามลำดับ เมื่อนำมาสร้างเป็นเพรดิเคต จะได้เป็นสูตรที่ 3 ตามตารางที่ 3.1

โหนดที่ (4) คือ คลาส VariableDeclarationExpression คุณลักษณะที่ต้องใช้ในสัญลักษณ์ Ø(a) คือ name, type, line และ token.endLine หมายถึง ชื่อคุณลักษณะ ซึ่งเป็นคุณลักษณะระดับคลาส ชนิดข้อมูล บรรทัดเริ่มต้น และ บรรทัดสุดท้าย ตามลำดับ เมื่อนำมาสร้างเป็นเพรดิเคต จะได้เป็นสูตรที่ 4 ตามตารางที่ 3.1

โหนดที่ (5) คือ คลาส MethodCallExpr คุณลักษณะที่ต้องใช้ในสัญลักษณ์ Ø(e) คือ name, args, ret, ret.getBeginLine() และ token.endLine หมายถึง คำสั่งการเรียกใช้เมทธอด พาราเมเตอร์ คลาสที่อ้างอิงถึง บรรทัดเริ่มต้น และ บรรทัดสุดท้าย ตามลำดับ เมื่อนำมาสร้างเป็นเพรดิเคต จะได้เป็นสูตรที่ 5 ตามตารางที่ 3.1

คุณลักษณะ ret หรือ คลาสที่อ้างอิงถึง ของคลาส MethodCallExpr สามารถสร้างเพรดเคต เพื่อเป็นส่วนขยายของเพรดิเคตตามสูตรที่ 5 ได้ คือ เพรดิเคตตามสูตรที่ 7 และ สูตรที่ 8 ตามตารางที่ 3.1

โหนดที่ (6) คือ คลาส FieldAccessExpr คุณลักษณะที่ต้องใช้ในสัญลักษณ์ Ø(fa) คือ name, ret, ret.getBeginLine() และ token.endLine หมายถึง คำสั่งการเรียกใช้คุณลักษณะ

คลาสที่อ้างอิงถึง บรรทัดเริ่มต้น และ บรรทัดสุดท้าย ตามลำดับ เมื่อนำมาสร้างเป็นเพรดิเคต จะได้เป็นสูตรที่ 6 ตามตารางที่ 3.1

จากรูปที่ 3.3 โหนดที่ (2) อยู่ภายใน โหนดที่ (1) ดังนั้น สามารถสร้างเพรดิเคต ตามสูตรที่ 9 ตามตารางที่ 3.1

เพรดิเคตทั้งหมดสามารถสรุปได้ตาม ตารางที่ 3.1

ตารางที่ 3.1 แสดงสรุปเพรดิเคต

สูตรที่	เพรดิเคต
1	jclass(name, beginLine,endLine).
2	jfield(name,type, beginLine,endLine).
3	jmethod(name,returnType,paraList,beginLine,endLine).
4	jattribute(name,type,beginLine,endLine).
5	jexpression(name,paraList,scope,beginLine,endLine).
6	jfieldaccess(name,scope,beginLine,endLine).
7	jscope(type,name).
8	jscope(type1,type2,name).
9	owns(oper1,oper2).
10	number(n,fun).
11	jequals(m,s).

สูตรที่ 1

สัญลักษณ์ : jclass(name,beginLine,endLine).

ความหมาย : บอกว่าเป็นคลาส พร้อมรายละเอียดเกี่ยวกับคลาส

โดยที่ :

jclass แทน คลาส ซึ่งเป็นสัญลักษณ์เพรดิเคต

name แทน ชื่อของคลาส ซึ่งเป็นอาร์กิวเมนต์ตัวที่หนึ่ง

beginLine แทน บรรทัดเริ่มต้น ซึ่งเป็นอาร์กิวเมนต์ตัวที่สอง

endLine แทน บรรทัดสุดท้าย ซึ่งเป็นอาร์กิวเมนต์ตัวที่สาม

สูตรที่ 2

สัญลักษณ์ : jfield(name,type,beginLine,endLine).

ความหมาย : บอกว่าเป็นคุณลักษณะพร้อมรายละเอียดเกี่ยวกับคุณลักษณะ

โดยที่ :

jfield แทนคุณลักษณะ ซึ่งเป็นสัญลักษณ์เพรดิเคต

name แทน ชื่อคุณลักษณะ ซึ่งเป็นอาร์กิวเมนต์ตัวที่หนึ่ง

type แทน ชนิดของตัวแปร ซึ่งเป็นอาร์กิวเมนต์ตัวที่สอง

beginLine แทน บรรทัดเริ่มต้น ซึ่งเป็นอาร์กิวเมนต์ตัวที่สาม

endLine แทน บรรทัดสุดท้าย ซึ่งเป็นอาร์กิวเมนต์ตัวที่สี่

สูตรที่ 3

สัญลักษณ์ : jmethod(name,returnType,paraList,beginLine,endLine).

ความหมาย : บอกว่าเป็นเมทธอด พร้อมรายละเอียดเกี่ยวกับเมทธอด

โดยที่ :

jmethod แทน เมทธอด ซึ่งเป็นสัญลักษณ์เพรดิเคต

name แทน ชื่อของเมทธอด ซึ่งเป็นอาร์กิวเมนต์ตัวที่หนึ่ง

returnType แทน return type ซึ่งเป็นอาร์กิวเมนต์ตัวที่สอง

paramList แทน พารามิเตอร์ ซึ่งเป็นอาร์กิวเมนต์ตัวที่สาม

beginLine แทน บรรทัดเริ่มต้น ซึ่งเป็นอาร์กิวเมนต์ตัวที่สี่

endLine แทน บรรทัดสุดท้าย ซึ่งเป็นอาร์กิวเมนต์ตัวที่ห้า

สูตรที่ 4

สัญลักษณ์ : jattribute(name,type,beginLine,endLine).

ความหมาย : บอกว่าเป็นแอตทิบิวต์พร้อมรายละเอียดเกี่ยวกับแอตทิบิวต์

โดยที่ :

jattribute แทนแอตทิบิวต์ซึ่งเป็นสัญลักษณ์เพรดิเคต

name แทน ชื่อแอตทิบิวต์ซึ่งเป็นอาร์กิวเมนต์ตัวที่หนึ่ง

type แทน ชนิดของตัวแปร ซึ่งเป็นอาร์กิวเมนต์ตัวที่สอง

beginLine แทน บรรทัดเริ่มต้น ซึ่งเป็นอาร์กิวเมนต์ตัวที่สาม

endLine แทน บรรทัดสุดท้าย ซึ่งเป็นอาร์กิวเมนต์ตัวที่สี่

สูตรที่ 5

สัญลักษณ์ : jexpression(name,paraList,scope,beginLine,endLine).

ความหมาย : บอกว่ามีการเรียกใช้เมทธอด

โดยที่ :

jexpression แทน expression ในการเรียกใช้เมทธอด ซึ่งเป็นสัญลักษณ์เพรดิเคต

name ชื่อของเมทธอดที่ถูกเรียก ซึ่งเป็นอาร์กิวเมนต์ตัวที่หนึ่ง

paraList แทนพารามิเตอร์ ซึ่งเป็นอาร์กิวเมนต์ตัวที่สอง

scope แทนคลาสที่ใช้สร้าง expression ซึ่งเป็นอาร์กิวเมนต์ตัวที่สาม

beginLine แทน บรรทัดเริ่มต้น ซึ่งเป็นอาร์กิวเมนต์ตัวที่สี่

endLine แทน บรรทัดสุดท้าย ซึ่งเป็นอาร์กิวเมนต์ตัวที่ห้า

สูตรที่ 6

สัญลักษณ์ : jfieldaccess(name,scope,beginLine,endLine).

ความหมาย : บอกว่ามีการเข้าถึงหรือเรียกใช้คุณลักษณะ

โดยที่ :

jfieldaccess แทน expression ในการเรียกใช้คุณลักษณะ ซึ่งเป็นสัญลักษณ์เพรดิเคต

name ชื่อของคุณลักษณะที่ถูกเรียก ซึ่งเป็นอาร์กิวเมนต์ตัวที่หนึ่ง

scope แทนคลาสที่ใช้สร้าง expression ซึ่งเป็นอาร์กิวเมนต์ตัวที่สอง

beginLine แทน บรรทัดเริ่มต้น ซึ่งเป็นอาร์กิวเมนต์ตัวที่สาม

endLine แทน บรรทัดสุดท้าย ซึ่งเป็นอาร์กิวเมนต์ตัวที่สี่

สูตรที่ 7

สัญลักษณ์ : jscope(type,name).

ความหมาย : บอกว่า expression ของการเรียกใช้เมทธอด หรือ คุณลักษณะนั้น ๆ เป็นการเรียกใช้เมทธอด หรือ คุณลักษณะของคลาสใด พร้อมรายละเอียด

โดยที่ :

jscope แทนสัญลักษณ์เพรดิเคต

type แทนชื่อคลาส ซึ่งเป็นอาร์กิวเมนต์ตัวที่หนึ่ง

name แทน การเรียกใช้เมทธอด หรือ คุณลักษณะ ซึ่งเป็นอาร์กิวเมนต์ตัวที่สอง

สูตรที่ 8

สัญลักษณ์ : jscope(type1,type2,name).

ความหมาย : บอกว่า expression ของการเรียกใช้เมทธอด หรือ คุณลักษณะนั้น ๆ เป็นการเรียกใช้เมทธอด หรือ คุณลักษณะของคลาสใดซึ่งจะคล้ายกับสูตรที่ 10 แต่จะเพิ่มอาร์กิวเมนต์ขึ้นมาอีกหนึ่งตัว พร้อมรายละเอียด

โดยที่ :

jscope แทนสัญลักษณ์เพรดิเคต

type1 แทน คลาสตัวที่หนึ่ง ซึ่งเป็นอาร์กิวเมนต์ตัวที่หนึ่ง

type2 แทน คลาสตัวที่สอง ซึ่งเป็นอาร์กิวเมนต์ตัวที่สอง

name แทน การเรียกใช้เมทธอด หรือ คุณลักษณะ ซึ่งเป็นอาร์กิวเมนต์ตัวที่สาม

สูตรที่ 9

สัญลักษณ์ : owns(oper1,oper2).

ความหมาย : บอกว่า oper1 เป็นเจ้าของ oper2 หรือ oper2 อยู่ใน oper1

โดยที่ :

owns เป็นสัญลักษณ์เพรดิเคต

oper1,oper2 เป็นอาร์กิวเมนต์ตัวที่หนึ่งและสองตามลำดับ

สูตรที่ 10

สัญลักษณ์ : number(n,fun).

ความหมาย : บอกว่ามีการเรียกใช้ฟังก์ชัน f จำนวน n ครั้ง

โดยที่ :

number เป็นสัญลักษณ์เพรดิเคต

n เป็นอาร์กิวเมนต์ตัวที่หนึ่ง ซึ่งแทนจำนวนครั้ง

fun เป็นอาร์กิวเมนต์ตัวที่สอง ซึ่งแทนชื่อฟังก์ชัน

สูตรที่ 11

สัญลักษณ์ : jequals(m,s).

ความหมาย : บอกว่า m เท่ากับ s ใช้ร่วมกับ scope jclass และ jexpression เพื่อบอกว่ามีการเรียกเมทธอดของคลาสใด

โดยที่ :

jequals เป็นสัญลักษณ์เพรดิเคต

m เป็นอาร์กิวเมนต์ตัวที่หนึ่ง

s เป็นอาร์กิวเมนต์ตัวที่สอง

3.3 สร้างกฎโปรล็อก

เป็นการนำร่องรอยที่ไม่ดีมาสร้างเป็นกฎโปรล็อก โดยจะใช้เพรดิเคตที่ได้ออกแบบไว้ เพื่อสร้างเป็นกฎโปรล็อกที่มีความสอดคล้องกับร่องรอยที่ไม่ดีแต่ละประเภท มีรายละเอียดดังนี้

Feature Envy

ร่องรอยที่ไม่ดีประเภท Feature Envy จะเกิดขึ้น ใน 2 ลักษณะด้วยกัน คือ

1. Feature Envy ที่เกิดจากการเรียกใช้เมทธอด หมายถึง การที่มีเมทธอดไปเรียกใช้ หรือถูกเรียกใช้โดยคลาสอื่นมากกว่าคลาสที่เป็นเจ้าของเมทธอดนั้น สามารถเขียนกฎโปรล็อกที่ใช้สำหรับการเรียกใช้เมทธอด ดังแสดงในรูปที่ 3.4

```
mfeatureenvy(A, B, m, s) :-
    jclass(A,_,_),
    jclass(B,_,_),
    jmthod(m,_,_,_),
    jexpression(s,_,_,_,_),
    jscope(A, s),
    jequals(m, s),
    owns(B, s),
    owns(A, m);
```

รูปที่ 3.4 กฎโปรล็อกของการเรียกใช้เมทธอด

โดยที่ mfeatureenvy(A, B, m, s) เป็นส่วนหัวของกฎโปรล็อก [11] และ มีอาร์กิวเมนต์ ที่รับเข้ามา 4 ตัว คือ A, B, m, s

ส่วนลำตัวของกฎ [11] หมายถึง ส่วนที่อยู่หลังเครื่องหมาย :- ซึ่งจะอ้างอิงกับเพรดิเคต จากตารางที่ 3.1 และ อาร์กิวเมนต์ที่เป็นเครื่องหมาย "_" หมายความว่าจะเป็นอะไรก็ได้

ในที่นี้กฎโปรล็อกในรูปที่ 3.4 หมายความว่า คลาส "B" มีการเรียกใช้ เมทธอด "m" ที่อยู่ในคลาส "A"

ในการค้นหา Feature Envy ที่เกิดจากการเรียกใช้เมทธอด เป็นการนำผลที่ได้จากการค้นหา โดยใช้กฎโปรล็อกตามรูปที่ 3.4 มาพิจารณา ซึ่งสามารถทำได้ 2 วิธี คือ โดยใช้เพรดิเคต number(n,func) ในที่นี้ "func" คือ กฎโปรล็อกตามรูปที่ 3.4 ที่มีอาร์กิวเมนต์ต่างๆ กัน อีกวิธี คือ นับผลที่ได้ โดยตรงจากโปรแกรม ซึ่งในขั้นตอนการสร้างเครื่องมือผู้วิจัยเลือกใช้ วิธีที่ 2

2. Feature Envy ที่เกิดจากการเรียกใช้คุณลักษณะ หมายถึง การที่มีคุณลักษณะถูกเรียกใช้โดยคลาสอื่น มากกว่าคลาสที่เป็นเจ้าของคุณลักษณะนั้น สามารถสร้างกฎโปรล็อกได้ ดังแสดงในรูปที่ 3.5

```
afeatureenvy(A, B, f, s) :-
    jclass(A,_,_),
    jclass(B,_,_),
    jfield(f,_,_,_),
    jfieldaccess(s,_,_,_),
    jscope(A, s),
    jequals(f, s),
    owns(B, s),
    owns(A, f);
```

รูปที่ 3.5 กฎโปรล็อกของการเรียกใช้คุณลักษณะ

จากกฎโปรล็อกตามรูปที่ 3.5 การทำงานจะเหมือนกับ กฎโปรล็อกในรูปที่ 3.4 แต่จะต่างกันที่เป็นการเรียกใช้ คุณลักษณะแทนการเรียกใช้เมทธอด และ การค้นหาจะใช้วิธีการนับผลการเรียกใช้คุณลักษณะ แล้ว มาเปรียบเทียบว่าคลาสใดมีการเรียกใช้มากกว่า

Message Chains

หมายถึง ลักษณะที่เมทธอดของคลาส มีการเรียกใช้อินแสตนท์ ของคลาสใดๆ ผ่านอินสแตนท์ของคลาสอื่น โดยมีการเรียกใช้อินสแตนท์ของคลาสอื่นต่อไปเรื่อย ๆ จะทำให้คลาสมีความขึ้นต่อกัน ในชุดของการเรียกใช้งานสูง

ในงานวิจัยนี้ ผู้วิจัยจะพิจารณา ร่องรอยที่ไม่ดีประเภท Message Chains ที่มีความสัมพันธ์กันจำนวน 3 คลาส สร้างกฎโปรล็อกได้ ดังแสดงในรูปที่ 3.6

```
messagechains(A, B, C, m, s) :-
    jclass(A,_,_),
    jclass(B,_,_),
    jclass(C,_,_),
    jmthod(m,_,_,_),
    jexpression(s,_,_,_,_),
    jscope(B,C,s),
    owns(m, s),
    owns(A, m);
```

รูปที่ 3.6 กฎโปรล็อกสำหรับร่องรอยที่ไม่ดีประเภท Message Chains

จากรูปที่ 3.6 กฎโปรล็อกสำหรับร่องรอยที่ไม่ดีประเภท Message Chains ที่มีความสัมพันธ์กัน 3 คลาส ส่วนหัวของกฎโปรล็อก มีอาร์กิวเมนต์ 5 ตัว คือ A, B, C, m, s และ ส่วนลำตัวของกฎโปรล็อก คือ การแทนด้วยเพรดิเคตตามตารางที่ 3.1

กฎโปรล็อกในรูป 3.6 หมายความว่า เมทธอด "m" ที่อยู่ในคลาส "A" มีการเรียกใช้เมทธอดที่อยู่ในคลาส C โดยผ่านอินสแตนท์ของคลาส "B"

Middle Man

หมายถึง ลักษณะที่คลาสใด ๆ มีเมทธอดในการทำหน้าที่มอบหมายงานให้กับคลาสอื่น สร้างกฎโปรล็อกได้ ดังแสดงในรูปที่ 3.7

```
middleman(A, B, C, m1, m2, s1, s2) :-
    jclass(A,_,_),
    jclass(B,_,_),
    jclass(C,_,_),
    jmthod(m1,_,_,_),
    jmthod(m2,_,_,_),
    jexpression(s1,_,_,_,_),
    jexpression(s2,_,_,_,_),
    jscope(B,s1),
    jscope(C,s2),
    jequals(m2, s1),
    owns(m1, s1),
    owns(m2, s2),
    owns(A, m1),
    owns(B, m2);
```

รูปที่ 3.7 กฎโปรล็อกสำหรับร่องรอยที่ไม่ดีประเภท Middle Man

จากรูปที่ 3.7 กฎโปรล็อกสำหรับร่องรอยที่ไม่ดีประเภท Middle Man ส่วนหัวของกฎโปรล็อก มีอาร์กิวเมนต์ 7 ตัว คือ A, B, C, m1, m2, s1, s2 และ ส่วนลำตัว ของกฎโปรล็อก คือ การแทนด้วยเพรดิเคต ตามตารางที่ 3.1

กฎโปรล็อกในรูปที่ 3.7 หมายความว่า เมทธอด "m1" ที่อยู่ในคลาส "A" มีการเรียกใช้เมทธอด "m2" ที่อยู่ในคลาส "B" จากนั้นเมทธอด "m2" มีการเรียกเมทธอด ที่อยู่ในคลาส "C" ซึ่งจะเห็นว่าเมทธอด "m1" ควรจะเรียกใช้เมทธอดที่อยู่ในคลาส "C" โดยตรงได้เลยโดยที่ไม่ต้องผ่านเมทธอด "m2" ที่อยู่ในคลาส "B"

Inappropriate Intimacy (General Form)

หมายถึง ลักษณะที่มีการพยายามเรียกใช้ส่วนที่เป็นไพรเวท (Private) ระหว่างคลาส สำหรับในงานวิจัยนี้ ผู้วิจัยจะพิจารณา ในส่วนของการพยายามเรียกใช้คุณลักษณะระหว่างคลาส ซึ่งตามหลักการของโปรแกรมเชิงวัตถุแล้วคุณลักษณะน่าจะเป็นไพรเวท และ ไม่สามารถเรียกใช้งานได้โดยตรงจากคลาสอื่น จาก [10] ได้แบ่ง Inappropriate Intimacy ออกเป็น Inappropriate Intimacy (General Form) และ Inappropriate Intimacy (Subclass Form) ซึ่ง Inappropriate

Intimacy (Subclass Form) คือ การเข้าถึงข้อมูลหรือเรียกใช้ข้อมูลระหว่างคลาสที่ Inheritance กัน หรือ ระหว่าง Parent และ Child

Inappropriate Intimacy (General Form) จะมีความคล้ายกับ Feature Envy ที่พิจารณาในส่วนของการเรียกใช้คุณลักษณะ แต่จะต่างกันที่ ถ้าเป็น Inappropriate Intimacy (General Form) จะเกิดขึ้นทุกกรณีที่มีการเรียกใช้คุณลักษณะจากคลาสอื่น โดยไม่มีการพิจารณาจำนวนครั้งของการเรียกใช้ สร้างกฎโปรล็อกได้ ดังแสดงในรูปที่ 3.8

```
inappropriateintimacy(A, B, f, s) :-
    jclass(A,_,_),
    jclass(B,_,_),
    jfield(f,_,_,_),
    jfieldaccess(s,_,_,_),
    jscope(A,s),
    jscope(f,s),
    owns(B, s),
    owns(A, f);
```

รูปที่ 3.8 กฎโปรล็อกสำหรับร่องรอยที่ไม่ดีประเภท Inappropriate Intimacy (General Form)

จากรูปที่ 3.8 ส่วนหัวของกฎโปรล็อก มีอาร์กิวเมนต์ 4 ตัว คือ A, B, f, s และ ส่วนลำตัวของกฎโปรล็อก คือ การแทนด้วยเพรดิเคตตาม ตารางที่ 3.1

กฎโปรล็อกในรูปที่ 3.8 หมายความว่า คลาส "B" มีการเรียกใช้คุณลักษณะ "f" ซึ่ง อยู่ในคลาส "A"

3.4 สร้างข้อเท็จจริง

เป็นการนำซอร์สโค้ดต้นฉบับภาษาจาวามาแทนด้วยเพรดิเคตที่เราได้ออกแบบไว้ ซึ่งจะเรียกว่าข้อเท็จจริงของโปรล็อก ตัวอย่างของ ซอร์สโค้ดต้นฉบับภาษาจาวา และ การสร้างข้อเท็จจริง มีรายละเอียดดังรูปที่ 3.9 และ รูปที่ 3.10 ตามลำดับ ส่วนข้อเท็จจริงของโปรแกรมที่ใช้ทดสอบทั้งหมดแสดงไว้ใน ภาคผนวก ค

```
1  class Movie {
2      public static final int CHILDRENS = 2;
3      private String _title;
4
5      public Movie() {
6
7      }
8
9      public String getTitle() {
10             return _title;
11     };
12 }
13
14 class Rental {
15
16     Movie _movie = new Movie();
17
18     public Rental() {
19
20     }
21
22     public String getTitleMovie() {
31             return _movie.getTitle();
24     }
25
26 }
```

รูปที่ 3.9 ตัวอย่างซอร์สโค้ดต้นฉบับภาษาจาวา

```
jclass(Movie, , 1, 12);
jclass(Rental, , 14, 26);
jfield(CHILDRENS, int, 2, 2);
owns(Movie, CHILDRENS);
jfield(_title, String, 3, 3);
owns(Movie, _title);
jfield(_movie, Movie, 16, 16);
owns(Rental, _movie);
jmethod(getTitle[], String, null, 9, 11);
owns(Movie, getTitle[]);
jmethod(getTitleMovie[], String, null, 22, 24);
owns(Rental, getTitleMovie[]);
jexpression(_movie.getTitle[], null, _movie, 31);
owns(getTitleMovie[], _movie.getTitle[]);
owns(Rental, _movie.getTitle[]);
jscope(Movie, _movie.getTitle[]);
jequals(getTitle[], _movie.getTitle[]);
jconstructor(Movie, null, 5, 7);
owns(Movie, Movie);
jconstructor(Rental, null, 18, 20);
owns(Rental, Rental);
```

รูปที่ 3.10 ตัวอย่างข้อเท็จจริงที่แปลงมาจากซอร์สโค้ดต้นฉบับภาษาจาวา

จากรูปที่ 3.10 ข้อเท็จจริงที่ได้เป็นการแปลงจากซอร์สโค้ดต้นฉบับภาษาจาวา ในรูปที่ 3.9 มาให้อยู่ในรูปของเพรดิเคตที่ออกแบบไว้ ตามตารางที่ 3.1

3.5 ข้อคำถาม

ข้อคำถาม หรือ ข้อคำถามโปรล็อก จะใช้ในการค้นหาร่อยรอยที่ไม่ดี ซึ่งข้อคำถามของแต่ละร่อยรอยที่ไม่ดีก็จะแตกต่างกันออกไป ข้อคำถามโปรล็อกของร่องรอยที่ไม่ดีทั้งหมดมีรายละเอียดดังนี้

1. ข้อคำถามสำหรับร่องรอยที่ไม่ดีประเภท Feature Envy ที่พิจารณาจากคุณลักษณะ คือ afeatureenvy(A,B,f,s). โดยที่ มีอาร์กิวเมนต์ 4 ตัว คือ "A" และ "B" แทนคลาส "f" แทนคุณลักษณะ และ ตัวสุดท้าย "s" แทน statement ของการเข้าถึงคุณลักษณะ หรือ การเรียกใช้คุณลักษณะ
2. ข้อคำถามสำหรับร่องรอยที่ไม่ดีประเภท Feature Envy ที่พิจารณาจากเมทธอด คือ mfeatureenvy(A,B,m,s). โดยที่ มีอาร์กิวเมนต์ 4 ตัว คือ "A" และ "B" แทนคลาส "m" แทนเมทธอด และ ตัวสุดท้าย "s" แทน statement ของการเรียกใช้เมทธอด
3. ข้อคำถามสำหรับร่องรอยที่ไม่ดีประเภท Message Chains คือ messagechains(A,B,C,m,s). โดยที่ มีอาร์กิวเมนต์ 5 ตัว คือ "A" "B" และ "C" แทนคลาส "m" แทนเมทธอด และตัวสุดท้าย "s" แทน statement ของการเรียกใช้เมทธอด
4. ข้อคำถามสำหรับร่องรอยที่ไม่ดีประเภท Middle Man คือ middleman(A,B,C,m1,m2,s1,s2). โดยที่ มีอาร์กิวเมนต์ 7 ตัว คือ "A" "B" และ "C" แทนคลาส "m1" และ "m2" แทนเมทธอด และ ตัวสุดท้าย "s1" และ "s2" แทน statement ของการเรียกใช้เมทธอด
5. ข้อคำถามสำหรับร่องรอยที่ไม่ดีประเภท Inappropriate Intimacy (General Form) คือ inappropriateintimacy(A,B,f,s). โดยที่ มีอาร์กิวเมนต์ 4 ตัว คือ "A" และ "B" แทนคลาส "f" แทนคุณลักษณะ และ ตัวสุดท้าย "s" แทน statement ของการเรียกใช้คุณลักษณะ