การตรวจจับแบบรูปการออกแบบเชิงพฤติกรรมด้วยแผนภาพคลาส

นางสาวนุชนาถ สัตย์วินิจ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

DETECTION OF BEHAVIORAL DESIGN PATTERNS USING CLASS DIAGRAM

Miss Nutchanart Satvinij

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2011

หัวข้อวิทยานิพนธ์	การตรวจจับแบบรูปการออกแบบเชิงพฤติกรรมด้วยแผนภาพคลาส
โดย	นางสาวนุชนาถ สัตย์วินิจ
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

..... กรรมการภายนอกมหาวิทยาลัย
(ดร.เนื่องวงศ์ ทวยเจริญ)

นุชนาถ สัตย์วินิจ : การตรวจจับแบบรูปการออกแบบเชิงพฤติกรรมด้วยแผนภาพคลาส. (DETECTION OF BEHAVIORAL DESIGN PATTERNS USING CLASS DIAGRAM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.วิวัฒน์ วัฒนาวุฒิ, 107 หน้า

แผนภาพคลาสยูเอ็มแอลเป็นแผนภาพที่ใช้ในการแสดงโครงสร้างและความสัมพันธ์กันของโครงสร้างต่างๆภายในระบบ ทำให้ผู้ออกแบบสามารถมองเห็นภาพรวมของระบบ โดยแผนภาพคลาสนี้สามารถนำมาประยุกต์ใช้ร่วมกับแนวทางที่ใช้ในการแก้ปัญหาในการออกแบบระบบที่เรียกว่า แบบรูปการออกแบบ ซึ่งแบบรูปการออกแบบจะถูกนำไปประยุกต์ใช้ในการออกแบบเพื่อช่วยให้การออกแบบเป็นไปในทิศทางเดียวกันกรณีที่มีผู้ออกแบบหลายคน และช่วยป้องกันการเกิดปัญหาตามมาเมื่อระบบมีการขยายขนาดให้ใหญ่มากขึ้น

งานวิจัยนี้นำเสนอขั้นตอนวิธีและเครื่องมือในการการตรวจจับแบบรูปการออกแบบด้วยแผนภาพคลาสโดยเน้นที่การตรวจจับแบบรูปการออกแบบเชิงพฤติกรรม ซึ่งสามารถตรวจจับแบบรูปการออกแบบพฤติกรรมได้ทั้งสิ้น 7 รูปแบบ ซึ่งเป็นแบบรูปที่นิยมใช้ในการออกแบบระบบ โดยการตรวจจับแบบรูปการออกแบบนี้จะมีการตรวจจับแบบรูปทั้งที่เป็นแบบรูปตามแบบมาตรฐานและแบบรูปที่ดัดแปลงจากมาตรฐานที่มีการเพิ่มเติมบางส่วนเข้าไปแต่ยังคงเป็นแบบรูปอยู่ ซึ่งข้อมูลนำเข้าของระบบที่ออกแบบนี้คือ ข้อมูลที่อยู่ในรูปแบบของแฟ้มเอ็กซ์เอ็มไอ โดยการตรวจจับจะทำการตรวจสอบหาโหนดหลัก (Anchor Node) เพื่อเป็นการหาจุดเริ่มต้นของแต่ละแบบรูปการออกแบบ และจะทำการตรวจหารายละเอียดของแบบรูปการออกแบบเฉพาะส่วนที่มีความสัมพันธ์กับโหนดหลักเท่านั้นเพื่อเป็นการลดการทำงาน การทดสอบจะทดสอบจากระบบซึ่งเป็นระบบที่ใช้งานจริง ซึ่งมีการใช้แบบรูปเชิงพฤติกรรมในการออกแบบ และจากกรณีทดสอบที่สร้างขึ้นที่มีการเพิ่มเติมบางส่วนเข้าไป ผลลัพธ์จะแสดงให้เห็นถึงชนิด และจำนวนของแบบรูปการออกแบบทั้งหมดที่พบ และระบุชุดของคลาส และความสัมพันธ์ ของแต่ละแบบรูปการออกแบบ การวัดความถูกต้องจะใช้ผู้เชี่ยวชาญทำการตรวจสอบข้อมูลผลลัพธ์ที่ได้

ภาควิชา .....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....

สาขาวิชา.....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา.....2554.....

5270361621 : MAJOR COMPUTER ENGINEERING
KEYWORD : DESIGN PATTERN / CLASS DIAGRAM / BEHAVIORAL PATTERN

NUTCHANART SATVINIJ : DETECTION OF BEHAVIORAL DESIGN PATTERNS USING CLASS DIAGRAM. ADVISOR : ASSOC. PROF. WIWAT VATANAWOOD, Ph.D., 107 pp.

The UML class diagram is the system design tool that helps the developer to see an overview of the system. It can be used with the design patterns to guide the direction in problem solving in the system design. The design patterns are applied to solve the difficult situations that may occur in the system development, to guide the designer to the same direction, and to prevent the problem of the system when the developer expands the system to large scale.

This research proposes an alternative algorithm and a tool to detect the Behavioral design patterns using exhaustive pattern matching scheme. It can detect seven patterns that are frequently used. The input of this tool is a UML class diagram, which is expected in XMI file format, and the output is the detected behavioral patterns. The algorithm will find the anchor node, that use to be the starting class for finding the detail of the design patterns. The algorithm provides pattern matching rules for each specific pattern, not only for exact design pattern generic schema but also modified design pattern schema, which is more practical to the real world. The modified design pattern schema means the original design pattern schema polished with the cascading inheritance and multiple occurrences of implementing concrete classes. The test cases of this algorithm are the generate test case and the real system. The accuracy of the proposed tool is verified by a specialist.

Department:.....Computer Engineering..... Student's Signature .....
Field of Study: .....Computer Engineering..... Advisor's Signature .....
Academic Year: .....2011.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาอย่างยิ่งจากรองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ อาจารย์ที่ปรึกษาหลัก ซึ่งได้ให้โอกาสและแนวคิดในการทำวิทยานิพนธ์ ตลอดจนทักษะ แนวทางการแก้ไขปัญหาและความอดทนให้การวิจัยลุล่วงและประสบความสำเร็จมาโดยตลอดระยะเวลาการศึกษาและการวิจัย ขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ด้วย

ขอขอบพระคุณ รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์ และอาจารย์ ดร.เนื่องวงศ์ ทวยเจริญ กรรมการสอบวิทยานิพนธ์ที่ได้ให้คำแนะนำและชี้แนะแนวทางที่เป็นประโยชน์ต่อการทำวิทยานิพนธ์ในครั้งนี้

ขอขอบพระคุณคณาจารย์ทุกท่านที่ อบรม สั่งสอน ให้ความรู้ต่างๆ มากมายจนมีวันนี้

ขอขอบพระคุณคณาจารย์และเพื่อนร่วมงานทุกท่านที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์ ที่คอยให้คำปรึกษา ติดตาม และเป็นกำลังใจในการทำวิทยานิพนธ์อย่างเสมอมา

ขอกราบขอบพระคุณคุณแม่ คุณพ่อ ครอบครัวอันเป็นที่รัก และญาติๆทุกคน ที่คอยให้ความห่วงใย ทำให้มีความสุขทั้งกายและใจ และเป็นกำลังใจในการดำเนินชีวิตมาโดยตลอด

ขอขอบคุณทุนพัฒนาอาจารย์จากมหาวิทยาลัยธุรกิจบัณฑิตย์ ที่ได้ให้เงินสนับสนุนด้านการศึกษาและการทำวิทยานิพนธ์ ทำให้ผู้วิจัยมีพลังในการสร้างสรรค์ผลงานเพื่อให้ได้มาซึ่งวิทยานิพนธ์ที่สมบูรณ์ในที่สุด

ขอขอบคุณ นายอนุศิษฐ์ ดิษดำ พี่ชายและครอบครัวที่คอยให้คำแนะนำ และช่วยเหลือในการทำวิทยานิพนธ์มาโดยตลอด

ขอบคุณเพื่อนๆ พี่ๆ น้องๆ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกคน ที่ร่วมทุกข์ร่วมสุข แลกเปลี่ยนความรู้ แง่คิดต่างๆ ตลอดระยะเวลาที่ดำเนินการวิจัย ซึ่งให้ความสนุกสนาน และความอบอุ่นตลอดระยะเวลาที่อยู่ด้วยกัน

สารบัญ

หน้า

บทที่ หน้า

บทที่ หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

ภาพที่ หน้า

ภาพที่ หน้า

ภาพที่ หน้า

ภาพที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันการออกแบบพัฒนาระบบโดยใช้แนวคิดการพัฒนาโปรแกรมเชิงวัตถุ (OOP: Object-Oriented Programming) เป็นที่นิยมอย่างสูง ซึ่งการออกแบบโดยใช้แนวคิดนี้มีเครื่องมือในการแสดงภาพรวมทั้งโครงสร้าง และรายละเอียดการทำงานของระบบ เพื่ออำนวยความสะดวกต่อผู้พัฒนา คือ แผนภาพยูเอ็มแอล (UML: Unified Modeling Language) โดยแผนภาพยูเอ็มแอลมีอยู่หลายรูปแบบขึ้นกับการใช้งานของผู้ใช้ว่าต้องการใช้แผนภาพแสดงภาพรวมของสิ่งใดหนึ่งในแผนภาพยูเอ็มแอลที่นิยมสำหรับผู้ออกแบบระบบ คือ แผนภาพคลาส (Class Diagram) โดยแผนภาพคลาสจะแสดงจำนวนคลาสและความสัมพันธ์ระหว่างคลาสที่มี ซึ่งแสดงให้เห็นภาพรวมของโครงสร้างของระบบ ซึ่งเป็นประโยชน์ในการพัฒนาระบบต่อไป การออกแบบระบบที่ดีควรออกแบบโดยคำนึงถึงระบบเดิมที่มีอยู่ และการเปลี่ยนแปลงของระบบในอนาคต สำหรับระบบเดิมที่มีอยู่ระบบที่พัฒนาขึ้นควรจะต่อเชื่อมเข้ากับระบบเดิม และสามารถทำงานร่วมกันได้ สำหรับการเปลี่ยนแปลงของระบบในอนาคต การเปลี่ยนแปลง เพิ่มหรือลดโครงสร้างหรือการทำงาน ควรเป็นไปได้ง่ายและไม่กระทบต่อส่วนอื่นๆที่เกี่ยวข้องในระบบ จากแนวคิดในการออกแบบระบบจึงมีการพัฒนาแบบรูปการออกแบบ (Design Patterns) เข้ามาใช้ในขั้นตอนการวิเคราะห์ และออกแบบระบบ

โดยในการพัฒนาระบบ ขั้นตอนในการพัฒนาจะเริ่มต้นจากการรวบรวมความต้องการของระบบจากผู้ใช้ จากนั้นนำความต้องการที่รวบรวมมาทำการวิเคราะห์ และจะเข้าสู่ขั้นตอนการออกแบบ ในขั้นตอนการออกแบบนี้ แบบรูปการออกแบบจะถูกนำเข้ามาใช้งาน เพื่อช่วยให้การพัฒนาระบบเป็นไปในทิศทางเดียวกัน เนื่องมาจากในการออกแบบระบบอาจมีจำนวนนักออกแบบหลายคน ซึ่งแต่ละคนอาจมีลักษณะแนวคิดในการวิเคราะห์ปัญหา และออกแบบที่แตกต่างกัน หากนักออกแบบทราบถึงปัญหา และสามารถนำแบบรูปการออกแบบมาช่วยในการแก้ปัญหาได้ จะส่งผลให้การออกแบบเป็นไปในทิศทางเดียวกัน ส่งผลให้การพัฒนาระบบเป็นไปได้รวดเร็วยิ่งขึ้น และการที่มีแบบรูปการออกแบบยังช่วยให้การเชื่อมต่อระบบเดิม และระบบใหม่ การเพิ่มหรือลดโครงสร้างต่างๆเป็นไปได้ง่าย เนื่องจากแบบรูปการออกแบบมีโครงสร้างที่อำนวยความสะดวกในการแก้ไขระบบโดยไม่ส่งผลต่อส่วนอื่นๆ

แบบรูปการออกแบบเป็นการอธิบายแนวทางหรือโครงสร้างที่สามารถนำไปประยุกต์ใช้ในสถานการณ์ต่างๆที่อาจเกิดขึ้นในการพัฒนาระบบ แบบรูปการออกแบบจะอธิบายแบบรูปอยู่ในรูปของแผนภาพคลาส และแผนภาพลำดับ (Sequence Diagram) โดยแผนภาพคลาสจะแสดงโครงสร้างของคลาส และความสัมพันธ์ของแต่ละแบบรูป ส่วนแผนภาพลำดับจะแสดงขั้นตอนการสร้างวัตถุ และการติดต่อสื่อสารกันของคลาสที่มี

นักออกแบบระบบส่วนใหญ่มักใช้แบบรูปการออกแบบในการป้องกันปัญหาในการออกแบบระบบที่อาจจะขยายวงกว้างขึ้น เช่น การเพิ่มหรือลดโครงสร้างต่างๆ หากระบบนั้นออกแบบไม่ดีการจะเพิ่มหรือลดอาจส่งผลให้ต้องทำการแก้ไขโครงสร้างทั้งหมดที่ได้ออกแบบไว้แล้ว เป็นต้น แบบรูปการออกแบบเป็นแบบแผนหรือแนวทางทั่วไปที่ใช้เพื่อแก้ไขปัญหาที่เกิดขึ้นบ่อยๆในการออกแบบระบบ โดยนักออกแบบมักจะแทนที่ส่วนที่มองออกว่าเป็นแบบรูปการออกแบบประเภทใดแล้วแทนที่แบบรูปการออกแบบนั้นๆ ลงไป ทำให้การเกิดการแยกส่วนของปัญหาออกมาเป็นส่วนๆ ประโยชน์ของแบบรูปการออกแบบต่อการพัฒนาระบบ คือ ช่วยให้การออกแบบระบบทำได้รวดเร็ว และมีประสิทธิภาพเพิ่มขึ้น และเป็นการเตรียมการสำหรับปัญหาต่างๆที่อาจจะไม่พบในขั้นตอนของการออกแบบ แต่อาจพบในการนำไปใช้งานจริง

แบบรูปการออกแบบที่มักพบในการใช้งาน คือ แบบรูปเชิงพฤติกรรม (Behavioral Pattern) เนื่องจากแบบรูปเชิงพฤติกรรมจะใช้ในการป้องกันหรือแก้ปัญหาการติดต่อสื่อสารระหว่างคลาสที่มีในระบบ จากประโยชน์ของแบบรูปการออกแบบดังกล่าว งานวิจัยหลายงานวิจัยจึงมีการพัฒนาเครื่องมือหรือขั้นตอนวิธี (Algorithm) ในการตรวจจับแบบรูปการออกแบบที่มีอยู่ในแผนภาพคลาส เพื่อประโยชน์ในการเข้าใจโครงสร้างของระบบที่มีอยู่ซึ่งส่งผลต่อการเชื่อมต่อระบบเข้าด้วยกัน

ดังนั้น งานวิจัยชิ้นนี้จึงเป็นการพัฒนาขั้นตอนวิธี และเครื่องมือในการตรวจจับแบบรูปการออกแบบเชิงพฤติกรรม โดยใช้แผนภาพคลาสเพื่อตรวจสอบโครงสร้างของระบบที่สนใจว่า ภายในโครงสร้างนั้นมีส่วนของแบบรูปของการออกแบบซ่อนอยู่หรือไม่ เพื่อเป็นการหาแบบรูปการออกแบบ และรายงานผลแก่ผู้ออกแบบ เพื่อให้ผู้ออกแบบจัดการกับแผนภาพคลาสเหล่านั้นต่อไป และเพื่อช่วยในการพิจารณาคุณสมบัติอื่นๆที่ผู้ออกแบบต้องการจากระบบ (Non-functional Requirement)

1.2 วัตถุประสงค์ของการวิจัย

เพื่อออกแบบขั้นตอนวิธีและพัฒนาเครื่องมือที่ใช้ในการตรวจจับแบบรูปการออกแบบเชิงพฤติกรรมของแผนภาพคลาส

1.3 ขอบเขตของการวิจัย

1) สร้างกฎที่ใช้ในการตรวจจับแบบรูปการออกแบบซึ่งจะครอบคลุมลักษณะของแบบรูปตามแบบมาตรฐาน และแบบรูปที่ดัดแปลงจากมาตรฐาน (Modified from Standard Design Pattern)

2) การตรวจจับแบบรูปการออกแบบจะครอบคลุมแบบรูปการออกแบบพฤติกรรมดังต่อไปนี้

- Chain of Responsibility
- Command
- Iterator
- Memento
- Observer
- Strategy
- Template Method

3) เครื่องมือใช้ข้อมูลนำเข้าเป็นแผนภาพคลาสที่อยู่ในรูปแบบแฟ้มเอ็กซ์เอ็มไอ โดยเครื่องมือที่พัฒนาขึ้นจะใช้ภาษาจาวาในการพัฒนา

4) เครื่องมือในการการตรวจจับแบบรูปการออกแบบจะสามารถแจกแจงชนิด และจำนวนของแบบรูปการออกแบบทั้งหมดที่พบ และระบุชุดของคลาส และชนิดของแต่ละแบบรูปการออกแบบ

5) การตรวจสอบความแม่นยำของเครื่องมือจะมีการใช้ข้อมูล 2 ประเภท คือ ข้อมูลที่สร้างขึ้นเอง และข้อมูลจากระบบจริงจำนวน 2 ระบบ โดยข้อมูลที่สร้างขึ้นจะทำการสร้างให้ครอบคลุมแบบรูปทั้งหมดตามข้อ 2)

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) นำเสนอทางเลือกของขั้นตอนวิธีที่ใช้ในการตรวจจับแบบรูปการออกแบบโดยไม่ต้องทำการแบ่งความสัมพันธ์ของแผนภาพคลาส
2) ได้เครื่องมือที่ใช้ในการตรวจจับแบบรูปการออกแบบเชิงพฤติกรรม
3) สามารถระบุตำแหน่ง และชนิดของแบบรูปการออกแบบที่มีในแผนภาพคลาส เพื่อให้นักออกแบบสามารถนำข้อมูลที่ได้ไปใช้ในการพัฒนาระบบต่อไป

1.5 วิธีดำเนินการวิจัย

1) ศึกษาข้อมูลเอกสารและงานวิจัยที่เกี่ยวข้องกับหัวข้อการตรวจจับแบบรูปการออกแบบ
2) ศึกษาความหมายของสัญลักษณ์ต่างๆในแผนภาพคลาสตามมาตรฐานของยูเอ็มแอล
3) ศึกษารูปแบบของการแปลงแผนภาพคลาสให้อยู่ในรูปของ XMI
4) ศึกษาแบบรูปการออกแบบ และหาลักษณะเฉพาะของแต่ละแบบรูปการออกแบบ เพื่อนำมาสร้างกฎสำหรับแบบรูปการออกแบบ
5) ออกแบบขั้นตอนวิธีตรวจจับแบบรูปการออกแบบสำหรับแต่ละแบบรูปการออกแบบ
6) ออกแบบเครื่องมือในการตรวจจับแบบรูปการออกแบบ รวมถึงรูปแบบของข้อมูลนำเข้า และข้อมูลส่งออก
7) พัฒนาเครื่องมือ และขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบตามที่ได้ออกแบบไว้
8) ทดสอบเครื่องมือ และขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบตามขอบเขตที่กำหนดไว้
9) สรุปผลการวิจัย และข้อเสนอแนะ
10) จัดทำวิทยานิพนธ์ฉบับสมบูรณ์

1.6 ผลงานตีพิมพ์

งานวิจัยนี้ได้รับการตีพิมพ์ในงานประชุมวิชาการ *The 15th International Annual Symposium on Computational Science and Engineering (ANSCSE2011)* ในวันที่ 30 มีนาคม - 1 เมษายน 2554 จัดที่มหาวิทยาลัยกรุงเทพ ประเทศไทย ชื่อผลงานตีพิมพ์ Detection of Behavioral Design Patterns ในเอกสารประกอบการประชุมวิชาการหน้าที่ 569-574 [1]

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่เกี่ยวข้องอันเป็นประโยชน์ในการทำวิจัยแบ่งออกเป็น 3 ส่วน ได้แก่ แผนภาพคลาสยูเอ็มแอล แบบรูปการออกแบบ และเอ็กซ์เอ็มแอล

2.1.1 แผนภาพคลาสยูเอ็มแอล (UML Class Diagram) [2]

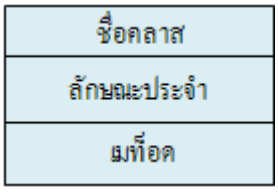
แผนภาพคลาส เป็นแผนภาพที่ใช้แสดงชุดของคลาส และความสัมพันธ์ของคลาสทั้งหมดที่มี และยังสามารถแสดงให้เห็นถึงลักษณะประจำ (Attribute) และเมท็อด (Method) หรือการดำเนินการ (Operation) ของคลาสอีกด้วยด้วย แผนภาพคลาสเป็นแผนภาพที่พบมากที่สุดในการสร้างแบบจำลองเชิงวัตถุ ซึ่งสามารถระบุภาพรวมของการออกแบบของระบบ

โดยแผนภาพคลาสประกอบไปด้วย คลาส ลักษณะประจำ เมท็อด และความสัมพันธ์ระหว่างคลาส [3]

2.1.1.1 คลาส

คลาสใช้อธิบายโครงสร้างและพฤติกรรมของวัตถุที่มีลักษณะ และความหมายเดียวกัน คลาสเป็นองค์ประกอบหลักของแผนภาพคลาส โครงสร้างของคลาสจะถูกอธิบายโดยลักษณะประจำ และพฤติกรรมของคลาสจะถูกอธิบายโดยเมท็อด

คลาสแต่ละคลาสจะแทนด้วยรูปสี่เหลี่ยม ซึ่งจะถูกแบ่งออกเป็นสามส่วน คือ ส่วนบนสุดใช้สำหรับแสดงชื่อคลาส ส่วนตรงกลางใช้สำหรับแสดงลักษณะประจำของคลาส และส่วนล่างสุดใช้แสดงเมท็อดของคลาส โดยส่วนของลักษณะประจำ และเมท็อดสามารถที่จะมีการซ่อนไม่แสดงได้



ภาพที่ 2.1 แสดงองค์ประกอบของคลาส

ชนิดของคลาสโดยทั่วไปสามารถแบ่งออกได้เป็น 3 ชนิด คือ

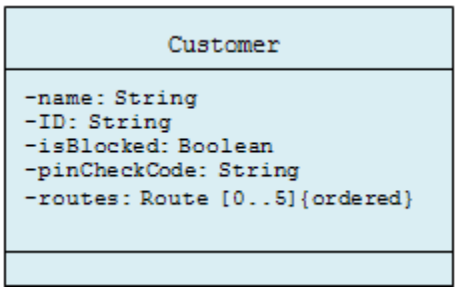
- คลาสรูปธรรม (Concrete Class) เป็นคลาสทั่วๆไปที่สามารถสร้างวัตถุ (Object) ได้
- คลาสนามธรรม (Abstract Class) เป็นคลาสที่สร้างขึ้นเพื่อเป็นซุปเปอร์คลาส (Superclass) โดยเฉพาะ ไม่สามารถสร้างวัตถุได้ และยังสามารถประกาศเมท็อดนามธรรม (Abstract Method) ไว้ได้ โดยทุกๆ คลาสย่อยที่สืบทอดจากคลาสนี้จะต้องทำการอิมพลีเมนต์ (Implement) ทุกเมท็อดนามธรรมที่คลาสนามธรรมมี คลาสนามธรรมสามารถใช้อธิบายถึงคุณลักษณะ และการดำเนินงาน โดยมีจุดมุ่งหมายเพื่อประกาศสมาชิกร่วมบางส่วนของคลาสย่อย (Subclasses) โดยยังไม่นิยามรายละเอียดการทำงาน คลาสย่อยจะทำหน้าที่นิยามวิธีการทำงานแตกต่างกันไปตามความมุ่งหมายพิเศษของแต่ละคลาสย่อย
- คลาสอินเตอร์เฟซ (Interface Class) จะมีเพียงลักษณะประจำ และเมท็อดนามธรรมเท่านั้น ไม่สามารถสร้างวัตถุจากคลาสอินเตอร์เฟซ และคลาสย่อยของอินเตอร์เฟซคลาสได้ จะสามารถสร้างวัตถุได้จากการอิมพลีเมนต์ของคลาสอินเตอร์เฟซเท่านั้น โดยที่ทุกๆการอิมพลีเมนต์จากคลาสอินเตอร์เฟซจะต้องทำการอิมพลีเมนต์ทุกเมท็อดนามธรรม

2.1.1.2 ลักษณะประจำ

ลักษณะประจำใช้ในการอธิบายถึงลักษณะคุณสมบัติโครงสร้างของคลาส ซึ่งประกอบไปด้วยความสามารถในการมองเห็นได้ (Visibility) ชนิด (Type) และ Multiplicity โครงสร้างของคลาสจะถูกกำหนดโดยคุณลักษณะ ซึ่งเป็นนามธรรมของแบบจำลองเทียบกับโลกแห่งความจริง ลักษณะประจำจะสะท้อนให้เห็นถึงโครงสร้างที่เกี่ยวข้องกับแบบจำลอง ในยูเอ็มแอลกำหนดชนิดของความสามารถในการมองเห็นได้ไว้ 4 รูปแบบ คือ

- Public แทนด้วยสัญลักษณ์ “+” หมายถึง การอนุญาตให้ทุกวัตถุสามารถเข้าถึงหรือมองเห็นค่าของลักษณะประจำนี้ได้
- Private แทนด้วยสัญลักษณ์ “-” หมายถึง วัตถุอื่น ๆ ไม่สามารถเข้าถึงหรือมองเห็นค่าของลักษณะประจำนี้ได้ มีเพียงคลาสตัวเองเท่านั้นที่สามารถเข้าถึงค่าของลักษณะประจำนี้ได้
- Package แทนด้วยสัญลักษณ์ “~” หมายถึง การอนุญาตให้วัตถุที่อยู่ภายใต้แพ็คเก็จเดียวกันกับคลาสสามารถเข้าถึงหรือมองเห็นค่าของลักษณะประจำนี้ได้เท่านั้น

- Protected แทนด้วยเครื่องหมาย “#” หมายถึง การอนุญาตให้เฉพาะวัตถุของคลาสย่อยเข้าถึงหรือมองเห็นค่าของลักษณะประจำนี้ได้เท่านั้น

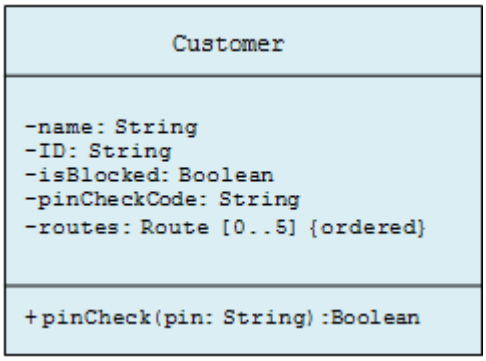


ภาพที่ 2.2 แสดงคลาสซึ่งประกอบด้วยชื่อคลาส และลักษณะประจำ

2.1.1.3 เมท็อดหรือการดำเนินการ

เมท็อดใช้ในการอธิบายถึงคุณสมบัติทางพฤติกรรมของคลาส คือกิจกรรมหรือฟังก์ชันการทำงานของคลาสที่สามารถกระทำได้ ซึ่งประกอบไปด้วยความสามารถในการมองเห็นได้ (Visibility) ชื่อ (Name) พารามิเตอร์ (Parameter) และชนิดการส่งค่ากลับ (Return Type) พฤติกรรมของคลาสจะถูกกำหนดโดยผลรวมของเมท็อดของคลาส

พารามิเตอร์ของคลาสจะมีการใช้สัญลักษณ์คล้ายกับลักษณะประจำ หากคลาสมีพารามิเตอร์จำนวนมากจะใช้สัญลักษณ์ “,” ในการแบ่งพารามิเตอร์ และความสามารถในการมองเห็นของคลาสก็มีลักษณะเหมือนกับความสามารถในการมองเห็นของลักษณะประจำ



ภาพที่ 2.3 แสดงคลาสซึ่งประกอบด้วยชื่อคลาส ลักษณะประจำ และเมท็อด

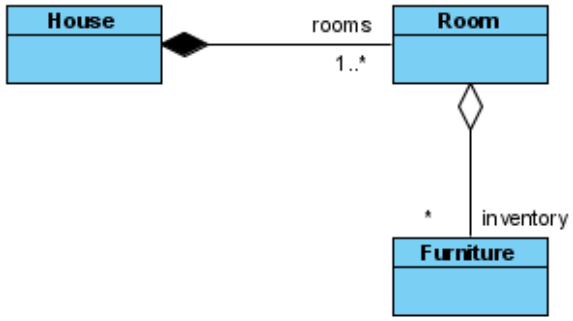
2.1.1.4 **ความสัมพันธ์ระหว่างคลาส**

ความสัมพันธ์ระหว่างคลาสใช้แสดงความสัมพันธ์ระหว่างคลาสต่างๆที่มีภายในแผนภาพคลาส โดยจะถูกแสดงด้วยเส้นที่ลากเชื่อมระหว่างคลาส [4, 5]

- ความสัมพันธ์แบบแอสโซซิเอชัน (Association) เป็นความสัมพันธ์โครงสร้างระหว่างคลาสสองคลาส ในลักษณะที่วัตถุหนึ่งสามารถอ้างอิงถึงวัตถุอื่น ๆ ได้ และสามารถเรียกใช้ เมท็อดของวัตถุอื่น ๆ ได้ ซึ่งมี Multiplicity เพื่อบ่งบอกปริมาณ

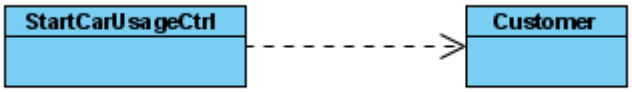
ภาพที่ 2.4 แสดงตัวอย่างความสัมพันธ์แบบแอสโซซิเอชัน

- ความสัมพันธ์แบบอะกรีเกรชัน (Aggregation) และคอมโพสิชัน (Composition) ใช้ในการอธิบายโครงสร้างแบบส่วนใหญ่ส่วนย่อย (Whole - Part Hierarchy) เป็นความสัมพันธ์ในลักษณะของการรวมกันหรือเป็นส่วนประกอบกันของคลาสหรือวัตถุ ความสัมพันธ์แบบอะกรีเกรชันมักรู้จักในรูปแบบของความสัมพันธ์แบบ Has-a (Has-a Relationship) เนื่องมาจากส่วนใหญ่ (Whole) จะมีส่วนย่อย (Part) โดยความสัมพันธ์ลักษณะนี้ส่วนย่อยเป็นส่วนประกอบของส่วนใหญ่ เมื่อส่วนใหญ่ถูกทำลายไปส่วนย่อยก็ยังคงอยู่ได้ ส่วนความสัมพันธ์แบบคอมโพสิชันมักรู้จักในรูปแบบของความสัมพันธ์แบบ Contains-a (Contains-a Relationship) เนื่องมาจากส่วนใหญ่ประกอบด้วยส่วนย่อย โดยความสัมพันธ์ในลักษณะนี้ส่วนย่อยเป็นส่วนประกอบของส่วนใหญ่ แต่เมื่อส่วนใหญ่ถูกทำลายหายไป ส่วนย่อยก็ถูกทำลายหายไปด้วย



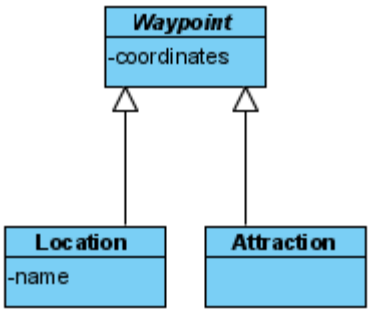
ภาพที่ 2.5 แสดงตัวอย่างความสัมพันธ์แบบอะกรีเกรชันและคอมโพสิชัน

- ความสัมพันธ์แบบดีเพนเดนซี (Dependency) เป็นความสัมพันธ์ระหว่างองค์ประกอบสององค์ประกอบโดยที่องค์ประกอบอันหนึ่งต้องการให้องค์ประกอบอื่นทำการอิมพลีเมนต์ โดยหากองค์ประกอบหลักมีการเปลี่ยนแปลงองค์ประกอบอื่นก็ต้องการการเปลี่ยนแปลงด้วย



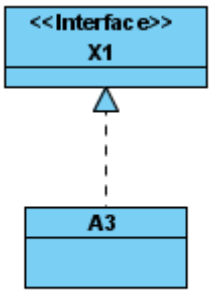
ภาพที่ 2.6 แสดงตัวอย่างความสัมพันธ์แบบดีเพนเดนซี

- ความสัมพันธ์แบบเจนเนอรัลไลเซชัน (Generalization) เป็นความสัมพันธ์ของการแบ่งหมวดหมู่ (Taxonomic Relationship) จากคลาสพิเศษ (Special Class) ไปยังคลาสทั่วไป (General Class) ในการพัฒนาซอฟต์แวร์จะใช้คำว่า การสืบทอด (Inheritance) มากกว่าเจนเนอรัลไลเซชัน บางครั้งใช้คำว่า “เป็นประเภทหนึ่งของ” (is a kind of) โดยความสัมพันธ์แบบเจนเนอรัลไลเซชันเป็นการใช้วิธีการสร้างคลาสจากการสืบทอดลักษณะประจำและเมท็อดมาจากคลาสอื่น ทำการสร้างซูเปอร์คลาสที่มีลักษณะประจำและเมท็อดพื้นฐาน ซึ่งลักษณะประจำและเมท็อดพื้นฐานนี้จะถูกนำมาใช้ในคลาสย่อยด้วย โดยคลาสย่อยจะสืบทอดลักษณะประจำและเมท็อดมาจากซูเปอร์คลาส และยังสามารถมีลักษณะประจำและเมท็อดเพิ่มเติมเป็นของตัวเองได้



ภาพที่ 2.7 แสดงตัวอย่างความสัมพันธ์แบบเจนเนอรัลไลเซชัน

- ความสัมพันธ์แบบรีไลเซชัน (Realization) เป็นความสัมพันธ์ที่องค์ประกอบหลักจะมีการสนับสนุนการทำงานขององค์ประกอบอื่นๆ โดยองค์ประกอบอื่นๆถูกสร้างให้มีคุณสมบัติขององค์ประกอบหลัก เป็นความสัมพันธ์เชิงความหมาย ระหว่าง อินเตอร์เฟซกับคลาส



ภาพที่ 2.8 แสดงตัวอย่างความสัมพันธ์แบบรีไลเซชัน

2.1.2 แบบรูปการออกแบบ (Design Pattern)

แบบรูปการออกแบบ คือ วิธีในการแก้ปัญหาในการเขียนโปรแกรมโดยการปรับปรุงจากเทคนิคการออกแบบที่ดี [6] แบบรูปแต่ละแบบรูปนั้นจะอธิบายปัญหาที่มักจะเกิดขึ้นซ้ำๆ และอธิบายถึงส่วนสำคัญของวิธีการแก้ปัญหา โดยทั่วไปแล้วแบบรูปการออกแบบจะประกอบไปด้วยส่วนสำคัญ 4 ส่วน [7] คือ

- ชื่อของแบบรูป (Pattern name) เป็นส่วนที่แสดงชื่อของแบบรูป ซึ่งเป็นสิ่งที่ทำให้ผู้ใช้สามารถทราบได้ว่าแบบรูปที่จะนำมาใช้นั้นเป็นเช่นไร

- ปัญหา (Problem) เป็นส่วนที่ใช้ในการอธิบายว่าเมื่อใดควรมีการใช้แบบรูปการออกแบบ ซึ่งส่วนนี้จะอธิบายถึงปัญหา และรายละเอียดของปัญหา
- วิธีในการแก้ปัญหา (Solution) เป็นส่วนที่ใช้ในการอธิบายองค์ประกอบที่ใช้ในการออกแบบ ความสัมพันธ์ หน้าที่ (Responsibilities) และการทำงานร่วมกัน (Collaborations) ขององค์ประกอบเหล่านั้น
- ผลที่ตามมา (Consequences) เป็นส่วนที่ใช้ในการแสดงผลลัพธ์ที่ได้หลังจากมีการประยุกต์ใช้แบบรูปการออกแบบ

แบบรูปการออกแบบถูกแบ่งออกเป็น 3 กลุ่มตามวัตถุประสงค์การใช้งาน [7, 8] คือ

- แบบรูปการสร้างวัตถุ (Creational Patterns) คือ แบบรูปที่เกี่ยวข้องกับกระบวนการการสร้างวัตถุ
- แบบรูปเชิงโครงสร้าง (Structural Patterns) คือ แบบรูปที่เกี่ยวข้องกับส่วนประกอบของคลาสหรือวัตถุ
- แบบรูปเชิงพฤติกรรม (Behavioral Patterns) คือ แบบรูปที่บ่งถึงวิธีทางที่คลาสหรือวัตถุโต้ตอบและแบ่งหน้าที่กัน

2.1.2.1 **แบบรูปการสร้างวัตถุ**

แบบรูปโครงสร้างเป็นแบบรูปที่ใช้สำหรับแก้ปัญหาในการสร้างวัตถุ เป็นแบบรูปการออกแบบที่จัดการกับกลไกการสร้างวัตถุ พยายามที่จะสร้างวัตถุในลักษณะที่เหมาะสมกับสถานการณ์ จัดการกับรูปแบบพื้นฐานของการสร้างวัตถุที่อาจทำให้เกิดปัญหาในการออกแบบหรือการเพิ่มความซับซ้อนในการออกแบบ แบบรูปการสร้างวัตถุแก้ปัญหานี้โดยควบคุมการสร้างวัตถุ แบบรูปในกลุ่มนี้ประกอบไปด้วย

- Abstract Factory เป็นแบบรูปที่มีการเตรียมอินเตอร์เฟซ สำหรับสร้างกลุ่มของวัตถุ (Product) ที่เกี่ยวข้องหรือขึ้นต่อกันโดยไม่ต้องกำหนดคลาสรูปธรรม (Product Class) ของวัตถุนั้นๆ
- Builder เป็นแบบรูปที่ทำการแบ่งคอนสตรัคเตอร์ (Constructor) ของวัตถุที่มีความซับซ้อน โดยแบบรูป Builder นั้นสรางขึ้นมาเพื่อแยกส่วนของข้อมูลออกจากการแสดงผล
- Factory Method เป็นแบบรูปที่นิยามอินเตอร์เฟซสำหรับสร้างวัตถุแต่ไม่ได้เจาะจงชนิดที่แน่นอน คลาสย่อยเป็นผู้กำหนดเจาะจงว่าจะสร้างวัตถุของคลาสใด ใน

ความหมายโดยทั่วไปคือเมท็อดที่ทำหน้าที่สร้างวัตถุ วัตถุถูกสร้างผ่านการสืบทอด แนวคิดของ Factory Method คือมีวัตถุพิเศษทำหน้าที่เป็นโรงงานเพื่อสร้างวัตถุจากคลาสย่อยต่างๆที่มาจากซุปเปอร์คลาสเดียวกัน

- Prototype ช่วยในการลดจำนวนของคลาส วัตถุถูกสร้างผ่านตัวแทน แนวคิดของแบบรูป Prototype คือให้วัตถุตัวหนึ่งเป็นต้นแบบ (Prototype) เพื่อทำหน้าที่สร้างสำเนาที่เหมือนวัตถุต้นแบบทุกประการ ทั้งค่าของลักษณะประจำ และเมท็อดต่างๆ
- Singleton เป็นแบบรูปที่คลาสนั้นๆจะมีอินสแตนซ์ (Instance) เพียงอินสแตนซ์เดียวเท่านั้น และ แบบรูปนี้จะต้องสร้างทางสำหรับเข้าถึง โดยทุกๆวัตถุจะใช้อินสแตนซ์ของคลาสเดียวกันเสมอ ซึ่งประโยชน์ของแบบรูปนี้คือ ควบคุมการเข้าถึง และลดการใช้เนมสเปซ (Namespace)

2.1.2.2 แบบรูปโครงสร้าง

แบบรูปโครงสร้างเป็นแบบรูปที่ใช้สำหรับแก้ปัญหาเกี่ยวกับการวางโครงสร้างความสัมพันธ์ระหว่างคลาส ซึ่งเกี่ยวข้องกับการควบคุมความสัมพันธ์ระหว่างส่วนต่างๆ ในระบบ แบบรูปในกลุ่มนี้ประกอบไปด้วย

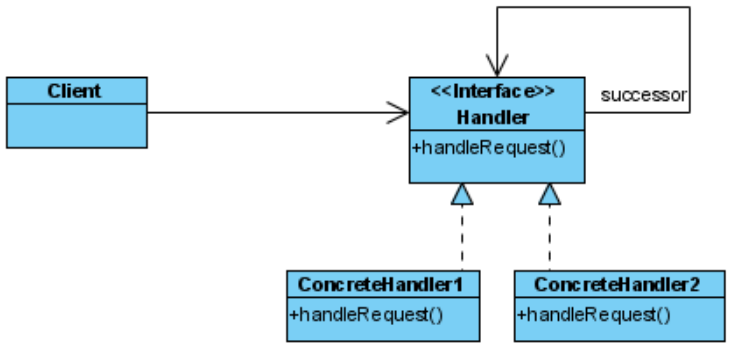
- Adapter เป็นแบบรูปที่ใช้ในการแปลงอินเตอร์เฟซหรือคลาสหนึ่งๆให้มีความสามารถเหมือนกับคลาสอื่น ซึ่งมีการใช้เมื่อต้องการให้คลาสสองคลาสที่ไม่ได้มีความเกี่ยวข้องสัมพันธ์กันให้สามารถติดต่อทำงานร่วมกันได้
- Bridge เป็นแบบรูปที่สร้างขึ้นเพื่อแยกการกำหนดสาระสำคัญ (Abstraction) และการอิมพลีเมนต์ (Implementation) ให้ไม่ขึ้นต่อกัน
- Composite เป็นแบบรูปที่จัดโครงสร้างของวัตถุที่มีลักษณะการทำงานคล้ายกันและเป็นส่วนประกอบย่อยๆ ของระบบใหญ่ให้อยู่ในรูปแบบโครงสร้างต้นไม้
- Decorator เป็นแบบรูปที่ทำให้สามารถเพิ่มหรือปรับแต่งคลาสได้อย่างอิสระในขณะทำงาน (Run Time) เป็นการขยายความสามารถของฟังก์ชันเดิมด้วยการสร้างคลาสย่อย
- Façade เป็นแบบรูปที่ทำหน้าที่ในการเตรียมอินเตอร์เฟซที่ทำให้สามารถติดต่อกับระบบที่ซับซ้อนโดยง่าย โดยไม่จำเป็นต้องเข้าใจถึงความซับซ้อนของระบบ เป็นการสร้างวัตถุตัวกลางที่ใช้ในการติดต่อระบบเพื่อให้ได้ผลลัพธ์ที่ต้องการ
- Flyweight เป็นแบบรูปที่รวมวัตถุที่มีลักษณะร่วมเหมือนกัน และทำการสร้างตัวแทนของวัตถุที่มีลักษณะเฉพาะแบบต่างๆเพื่ออ้างอิงจากภายนอก

- Proxy เป็นแบบรูปที่ใช้วัตถุตัวหนึ่งทำหน้าที่เป็นตัวแทนของวัตถุอื่น โดยวัตถุที่เป็นตัวแทนนี้จะทำหน้าที่เก็บสิ่งที่ใช้อ้างอิง (Reference) ไปยังวัตถุอื่น และทำการสร้างวัตถุนั้นเมื่อถึงเวลาที่จำเป็น

2.1.2.3 แบบรูปพฤติกรรม

แบบรูปพฤติกรรมเป็นแบบรูปที่ใช้สำหรับแก้ปัญหาพฤติกรรมของวัตถุ และการปฏิสัมพันธ์ระหว่างวัตถุ ซึ่งเกี่ยวข้องกับสถานะ (State) และการไหล (Flow) ของการทำงานของระบบ คลาสในแบบรูปพฤติกรรมใช้การสืบทอดในการกระจายพฤติกรรมระหว่างคลาส แบบรูปนี้จะบ่งถึงลักษณะพฤติกรรมของการควบคุมการไหลของการทำงานที่ยากที่จะติดตามขณะทำงาน (run-time) เป็นแบบรูปการออกแบบ โดยแบบรูปกลุ่มนี้จะเพิ่มความยืดหยุ่นในการดำเนินการสื่อสารที่ระบุรูปแบบการสื่อสารร่วมกันระหว่างวัตถุ แบบรูปในกลุ่มนี้ประกอบไปด้วย [9, 10, 11, 12, 13]

- Chain of Responsibility เป็นแบบรูปที่มีการสร้างลูกโซ่ (Chain) ของวัตถุเพื่อใช้ในการตรวจสอบการร้องขอ (Request) หรือข้อความ (Message) แต่ละวัตถุจะมีการตรวจสอบการร้องขอ และจัดกระทำ (Handle) กับการร้องขอนั้นหรือส่งต่อไปยังวัตถุอื่นๆที่อยู่ในลูกโซ่ เป็นการเรียกใช้เมท็อดแบบเป็นทอดๆผ่านทางอินเตอร์เฟซเดียวกัน



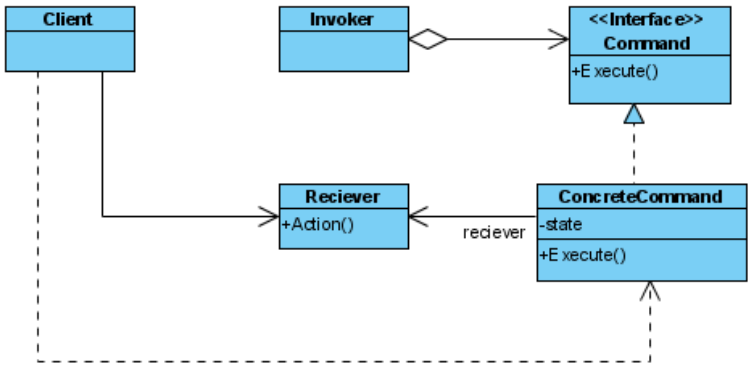
ภาพที่ 2.9 แสดงโครงสร้างของแบบรูป Chain of Responsibility

องค์ประกอบของแบบรูป Chain of Responsibility

- Handler เป็นคลาสอินเตอร์เฟซมีหน้าที่ในการประกาศเมท็อดที่ต้องมีในการจัดกระทำกับข้อความและการร้องขอต่างๆ

- ConcreteHandler เป็นอิมพลีเมนต์คลาส (Implementation Class) ของคลาส Handler ที่ทำหน้าที่ประมวลผลเมท็อดของวัตถุ ซึ่งในคลาสนี้จะมี successor ที่ทำหน้าที่เป็นผู้รับ (Receiver) ของข้อความของคลาสตนเอง
- Client เป็นคลาสที่ทำหน้าที่จัดลำดับ และกำหนดว่าผู้ส่ง (Sender) ตัวใดจะส่งข้อความหรือการร้องขอไปยังผู้รับใด

- Command เป็นแบบรูปที่ใช้หลักการของการห่อหุ้ม (Encapsulate) เพื่อซ่อนรายละเอียด ความซับซ้อนของคำสั่งต่างๆไว้ภายในวัตถุ โดยส่งคำสั่งไปยังวัตถุเพื่อให้วัตถุเรียกใช้การทำงานที่ต้องการโดยไม่ต้องทราบรายละเอียดของการทำงานนั้นๆ วัตถุจะมีความรับผิดชอบในการจัดการกับคำสั่ง โดยมีตัวควบคุมที่จะจัดการกับคำสั่งและวัตถุนั้น



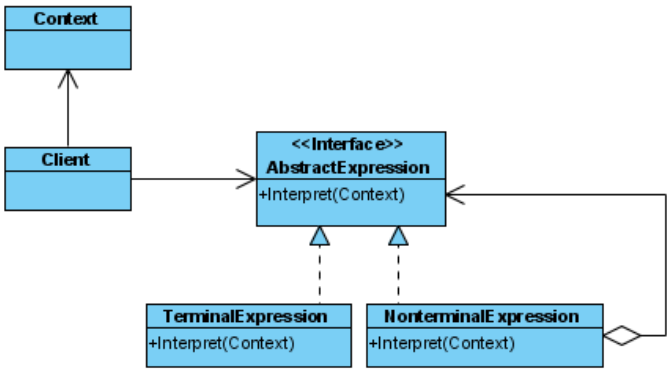
ภาพที่ 2.10 แสดงโครงสร้างของแบบรูป Command

องค์ประกอบของแบบรูป Command

- Command เป็นคลาสอินเตอร์เฟซสำหรับวัตถุที่มีหน้าที่ประมวลผลการทำงาน
- ConcreteCommand เป็นอิมพลีเมนต์คลาสของคลาส Command ทำหน้าที่ในการอิมพลีเมนต์การประมวลผลการทำงานของคลาสคอมมานต์ และจับคู่ระหว่างคำสั่งกับวัตถุและเมท็อดที่ควรจะรับผิดชอบในการทำงานกับคำสั่งนั้นๆ
- Client เป็นคลาสที่มีหน้าที่ในการสร้าง ConcreteCommand และกำหนดผู้รับที่เหมาะสมให้แก่ ConcreteCommand

- Invoker เป็นคลาสที่ทำหน้าที่เรียกคำสั่งหนึ่งๆให้ทำงาน
- Receiver เป็นคลาสที่ทำหน้าที่ในการจัดเตรียมการทำงานที่จำเป็นสำหรับการเรียกใช้จากภายนอก

- Interpreter เป็นแบบรูปที่ใช้หลักการเชิงวัตถุในการแปลสัญลักษณ์ (Symbol) หรือประมวลผลบริบท (Context) อย่างหนึ่งให้เป็นสัญลักษณ์อีกอย่างหนึ่ง

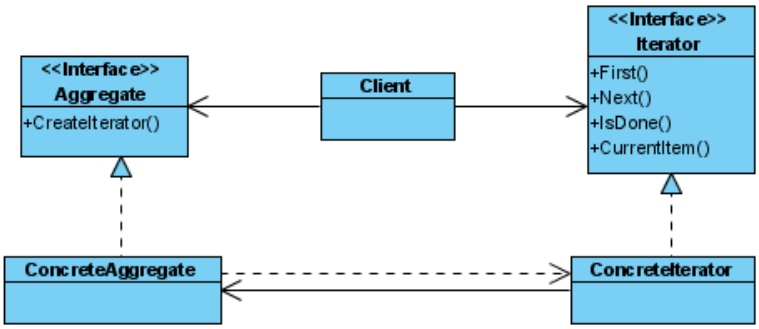


ภาพที่ 2.11 แสดงโครงสร้างของแบบรูป Interpreter

องค์ประกอบของแบบรูป Interpreter

- AbstractExpression เป็นคลาสอินเตอร์เฟซซึ่งทำหน้าที่ในการประกาศ เมท็อดที่ใช้ในการแปลหรือประมวลผล Context
- TerminalExpression เป็นอิมพลีเมนต์คลาสของคลาส AbstractExpression ซึ่งเป็นองค์ประกอบของผลลัพธ์สุดท้ายในการแปล
- NonterminalExpression เป็นอิมพลีเมนต์คลาสของคลาส AbstractExpression ซึ่งเป็นเนื้อหาที่เก็บอยู่ใน Context และเป็นองค์ประกอบของผลลัพธ์ระหว่างการแปล
- Context เป็นคลาสที่จัดเก็บเนื้อหาที่จะรับการประมวลผล
- Client เป็นคลาสที่ทำหน้าที่ในการสร้างกฎของการแปล และสั่งให้มีกระบวนการแปลเกิดขึ้น

- Iterator เป็นแบบรูปที่ทำให้สามารถเข้าถึงสมาชิกแต่ละตัวในคอลเลคชัน (Collection) โดยที่ไม่ต้องเข้าใจโครงสร้างของสมาชิก และไม่มีการเปิดเผยโครงสร้างของสมาชิกในคอลเลคชันให้ภายนอกรับรู้

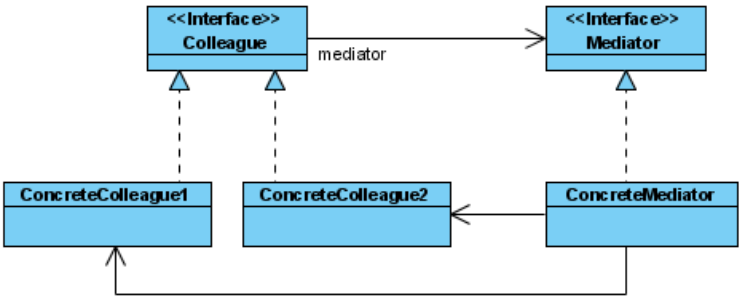


ภาพที่ 2.12 แสดงโครงสร้างของแบบรูป Iterator

องค์ประกอบของแบบรูป Iterator

- Iterator เป็นคลาสอินเตอร์เฟซที่ทำหน้าที่ประกาศเมท็อดที่จำเป็นในการทราเวิร์ส (Traverse) สมาชิกของคลาส Aggregate
- ConcreteIterator เป็นอิมพลีเมนต์คลาสของคลาส Iterator ทำหน้าที่ในการจดจำตำแหน่งปัจจุบันของสมาชิกของคลาส ConcreteAggregate ที่ถูกเข้าถึง
- Aggregate เป็นคลาสอินเตอร์เฟซที่ใช้ในการประกาศเมท็อดที่จำเป็นเพื่อใช้สร้างวัตถุของ Iterator
- ConcreteAggregate เป็นอิมพลีเมนต์คลาสของคลาส Aggregate โดยมีหน้าที่ในการตัดสินใจเลือกที่จะสร้าง ConcreteIterator

- Mediator เป็นแบบรูปที่ใช้ใช้แนวคิดในการมีตัวกลาง (Media) ที่ทำหน้าที่ส่งผ่านข้อความจากผู้ส่งไปยังผู้รับ โดยผู้ส่งไม่จำเป็นต้องถูกกำหนดว่าตนเองจะส่งข้อความไปให้ใคร และผู้รับก็ไม่จำเป็นต้องรู้ว่าข้อความนั้นมาจากวัตถุใด ตัวกลางนี้จะมีหน้าที่ในการห่อหุ้มข้อความจากวัตถุหนึ่งส่งไปยังวัตถุหนึ่ง ซึ่งสามารถเปลี่ยนแปลงผู้ส่ง และผู้รับได้อย่างอิสระ

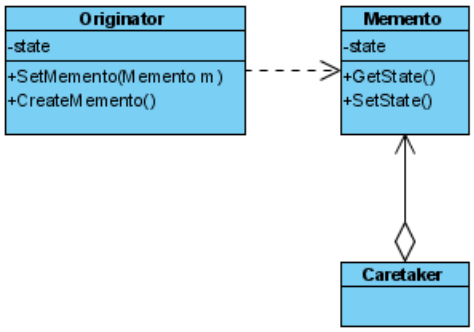


ภาพที่ 2.13 แสดงโครงสร้างของแบบรูป Mediator

องค์ประกอบของแบบรูป Mediator

- Mediator เป็นคลาสอินเตอร์เฟซที่มีหน้าที่ประกาศเมท็อดที่จำเป็นในการเป็นตัวกลางในการติดต่อระหว่างวัตถุ
- ConcreteMediator เป็นอิมพลีเมนต์คลาสของคลาส Mediator โดยจะรู้จัก และจัดการวัตถุที่ต้องทำการส่งผ่านว่าวัตถุใดเป็นผู้ส่ง วัตถุใดเป็นผู้รับ
- Colleague เป็นกลุ่มของคลาสที่จะมีการติดต่อสื่อสารกัน โดยที่ Colleague จะมีหน้าที่ในการรู้ว่าจะติดต่อกับวัตถุอื่นๆผ่านทาง Mediator ใด

- Memento เป็นแบบรูปที่ใช้แนวคิดในการจัดเก็บสถานะภายใน (Internal State) ของวัตถุในแต่ละช่วงเวลา เพื่อประโยชน์ในการนำสถานะดังกล่าวกลับมาเป็นของวัตถุได้ในภายหลัง โดยมีเงื่อนไขว่าการจัดเก็บดังกล่าวยังคงมีคุณสมบัติในการเป็นสถานะภายในซึ่งไม่สามารถเข้าถึงได้โดยตรงจากภายนอก

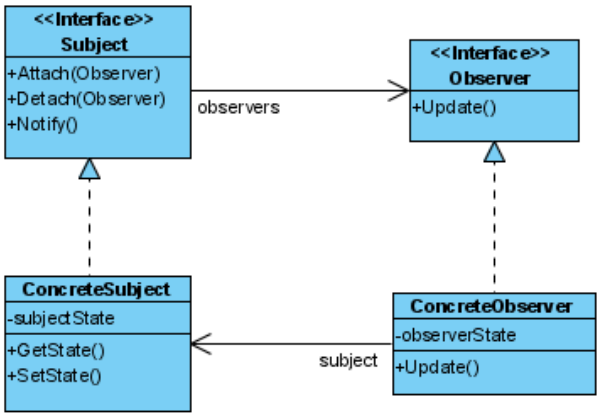


ภาพที่ 2.14 แสดงโครงสร้างของแบบรูป Memento

องค์ประกอบของแบบรูป Memento

- Memento เป็นคลาสที่มีหน้าที่ในการจัดเก็บสถานะของวัตถุที่สนใจ โดยสามารถตัดสินใจว่าจะจัดเก็บสถานะใดของวัตถุ และมีหน้าที่ในการป้องกันไม่ให้วัตถุของคลาสอื่นสามารถเข้าถึงสถานะที่เก็บไว้ได้
- Originator เป็นวัตถุที่สามารถเปลี่ยนสถานะได้ สามารถสร้างวัตถุของ Memento เพื่อจัดเก็บสถานะ และสามารถนำสถานะจากมี Memento กลับมาเพื่อเป็นสถานะปัจจุบันได้
- Caretaker เป็นคลาสซึ่งทำหน้าที่ในการรักษา Memento ของวัตถุไว้ โดยไม่เปลี่ยนแปลงเนื้อหาใดๆ และทำหน้าที่ในการให้บริการในการสร้างและนำส่ง Memento ให้แก่ Originator

- Observer เป็นแบบรูปที่ใช้แจ้งเหตุการณ์หรือข้อมูลข่าวสารจากคลาสต้นกำเนิดไปยังคลาสที่สมัครเป็นผู้รับข่าวสาร เป็นการกำหนดการอ้างอิงแบบหนึ่งต่อจำนวนมาก (one-to-many) ระหว่างวัตถุ เพื่อที่ว่าเมื่อวัตถุหนึ่งมีการเปลี่ยนแปลง วัตถุทั้งหมดที่ขึ้นต่อวัตถุนี้จะได้รับแจ้ง และทำการปรับปรุง (Update) โดยอัตโนมัติ

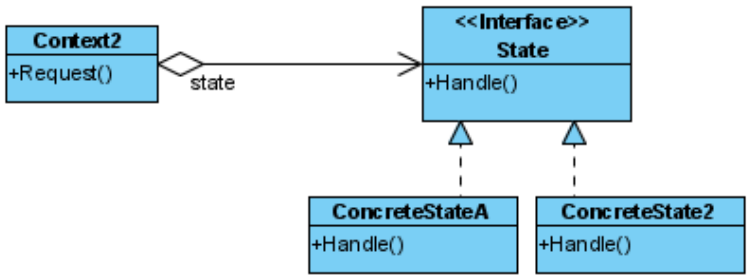


ภาพที่ 2.15 แสดงโครงสร้างของแบบรูป Observer

องค์ประกอบของแบบรูป Observer

- Subject เป็นคลาสอินเตอร์เฟซสำหรับวัตถุ ซึ่งจะประกาศเมท็อดต่างๆที่จำเป็นสำหรับการกำหนดว่าวัตถุตัวใดของ Observer ทำหน้าที่ในการสังเกตการเปลี่ยนแปลงสถานะของตนเอง สามารถที่จะเพิ่ม ลดผู้สังเกตของตนเองได้
- Observer เป็นคลาสอินเตอร์เฟซที่ประกาศเมท็อดที่จำเป็นสำหรับการกระตุ้นให้เกิดการตรวจสอบสถานะของวัตถุของคลาส ConcreteSubject
- ConcreteSubject เป็นอิมพลีเมนต์คลาสของ Subject มีหน้าที่ในการส่งข้อความเพื่อแจ้งแก่ Observer เมื่อมีการเปลี่ยนแปลงของสถานะเกิดขึ้น
- ConcreteObserver เป็นอิมพลีเมนต์คลาสของคลาส Observer ซึ่งจะทำหน้าที่อ้างอิงไปยัง Subject ที่สนใจ เพื่อตรวจสอบ และจดจำสถานะของวัตถุ

- State เป็นแบบรูปที่ทำให้สามารถจัดการเมื่อมีการเปลี่ยนแปลงสถานะ (State) ของวัตถุได้อย่างเป็นระบบ ช่วยให้วัตถุในการเปลี่ยนแปลงพฤติกรรมเมื่อมีการเปลี่ยนแปลงภายในเกิดขึ้น ซึ่งจะใช้หลักการสร้างวัตถุขึ้น เพื่อทำหน้าที่เป็นตัวแทนของสถานะของวัตถุใดวัตถุหนึ่ง



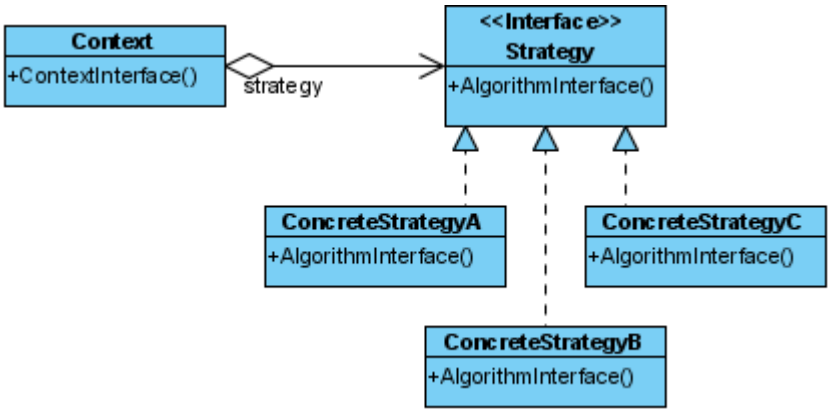
ภาพที่ 2.16 แสดงโครงสร้างของแบบรูป State

องค์ประกอบของแบบรูป State

- Context เป็นวัตถุที่ทำหน้าที่เก็บวัตถุของคลาส State
- State เป็นคลาสอินเตอร์เฟซที่ทำหน้าที่ในการประกาศเมท็อดที่จำเป็นสำหรับวัตถุที่ทำหน้าที่แสดงสถานะของวัตถุ

- ConcreteState เป็นอิมพลีเมนต์คลาสของคลาส State ทำหน้าที่ดำเนินการแสดง จัดการการเปลี่ยนสถานะ กำหนดกิจกรรมที่คลาส Context ต้องทำเมื่ออยู่ในสถานะหนึ่งๆ

- Strategy เป็นแบบรูปที่ใช้แนวคิดในการห่อหุ้มขั้นตอนวิธีที่แตกต่างกัน แต่ทำงานอย่างเดียวกันไว้ในคลาส โดยคลาสอื่นสามารถนำคลาสเหล่านี้ไปใช้งาน และสามารถเปลี่ยนแปลงการเลือกใช้ได้อิสระตลอดเวลา

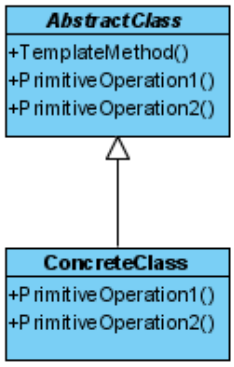


ภาพที่ 2.17 แสดงโครงสร้างของแบบรูป Strategy

องค์ประกอบของแบบรูป Strategy

- Strategy เป็นคลาสอินเตอร์เฟซที่ทำการประกาศเมท็อดที่มีหลายขั้นตอนวิธีในการทำงาน
- ConcreteStrategy เป็นอิมพลีเมนต์คลาสของคลาส Strategy ซึ่ง ConcreteStrategy แต่ละตัวจะมีขั้นตอนวิธีที่ต่างกันสำหรับแต่ละเมท็อดที่ประกาศไว้
- Context เป็นคลาสที่มีเมท็อดในการเรียกใช้ ConcreteStrategy โดยพฤติกรรมการทำงานของเมท็อดในคลาส Context จะถูกกำหนดโดย ConcreteStrategy

- Template Method เป็นแบบรูปที่ใช้กำหนดแม่แบบหรือโครงหลักของขั้นตอนวิธี โดยไม่ได้กำหนดรายละเอียดของขั้นตอนต่างๆ คลาสย่อยสามารถนิยามรายละเอียดของแต่ละขั้นตอนย่อยได้ตามความต้องการ

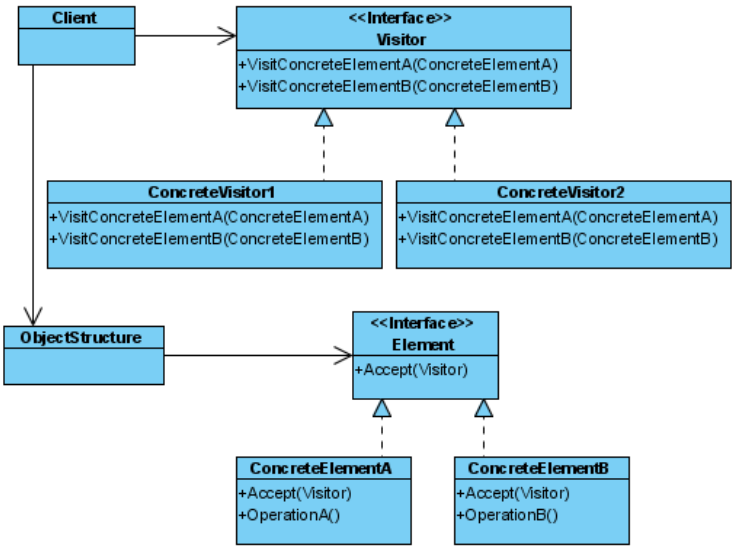


ภาพที่ 2.18 แสดงโครงสร้างของแบบรูป Template Method

องค์ประกอบของแบบรูป Template Method

- Abstract Class เป็นผู้กำหนดโครงหลักในการทำงาน กำหนดขั้นตอนย่อยของการทำงานโดยไม่กำหนดรายละเอียด โดยสร้างเป็นเมท็อดนามธรรม (Abstract method) หรือกำหนดการทำงานพื้นฐานไว้ และสร้าง Template Method ที่มีโครงของอัลกอริทึมและเรียกใช้ขั้นตอนย่อยที่ได้นิยามไว้แล้วเป็นเมท็อดนามธรรม
- ConcreteClass คลาสย่อยของคลาสนามธรรม กำหนดรายละเอียดของขั้นตอนย่อยที่ถูกนิยามในซูปเปอร์คลาส

- Visitor เป็นแบบรูปที่มีการสร้างวัตถุตัวหนึ่งขึ้นเพื่อทำการเข้าถึงสมาชิกแต่ละตัวและเลือกการดำเนินการให้เหมาะสมกับชนิดของวัตถุ โดยการเข้าถึงนี้จะเข้าถึงสมาชิกทุกตัวในคอลเลคชัน วัตถุที่ทำหน้าที่ในการเข้าถึงสมาชิกนี้เรียกว่า Visitor



ภาพที่ 2.19 แสดงโครงสร้างของแบบรูป Visitor

องค์ประกอบของแบบรูป Visitor

- Visitor เป็นคลาสอินเตอร์เฟซที่ประกาศเมท็อดในการเข้าถึงสมาชิกแต่ละตัวในโครงสร้างข้อมูลของวัตถุ (Object Structure) โดยชื่อและลายเซ็น (Signature) ของเมท็อดจะเป็นสิ่งที่กำหนดว่าเมท็อดนั้นๆใช้เพื่อดำเนินการกับวัตถุของคลาสใด
- ConcreteVisitor เป็นอิมพลีเมนต์คลาสของคลาส Visitor มีหน้าที่ในการเข้าถึงสมาชิกแต่ละตัวในโครงสร้างข้อมูลของวัตถุที่สนใจ
- Element เป็นคลาสอินเตอร์เฟซที่ทำหน้าที่ประกาศเมท็อดในการยินยอมให้มีการเข้าถึงได้
- ConcreteElement เป็นอิมพลีเมนต์คลาสของคลาส Element ซึ่งจะทำการอิมพลีเมนต์เมท็อดในคลาส Element
- ObjectStructure เป็นคอลเลคชันคลาสของคลาส Element มีหน้าที่ในการจัดเตรียมเมท็อดเพื่อให้ Visitor สามารถเข้าถึง Element แต่ละตัวได้

2.1.3 **เอ็กซ์เอ็มแอล** (XML) [16]

เอ็กซ์เอ็มแอล (XML: Extensible Markup Language) คือ ชุดของกฎสำหรับกำหนดแท็ก (Tag) ความหมายเพื่อใช้ในการแบ่งเอกสารออกเป็นส่วน และระบุส่วนต่างๆของเอกสาร เอ็กซ์เอ็มแอลเป็นภาษามาร์กอัปสำหรับการใช้งานทั่วไป เพื่อใช้ในการแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่แตกต่างกัน มาร์กอัพของเอ็กซ์เอ็มแอลใช้ในการบรรยายโครงสร้าง และความหมายของเอกสาร จุดประสงค์หลักของเอ็กซ์เอ็มแอลคือการแยกส่วนของข้อมูลเพื่อประโยชน์ในการแสดงผล โดยยึดเอาข้อมูลที่มีใจความเหมือนกัน แต่สามารถนำเอาไปแสดงผลเมื่อมองผ่านอุปกรณ์แสดงผลลัพธ์ที่ต่างกัน ซึ่งเอ็กซ์เอ็มแอลพยายามลดข้อจำกัดของความแตกต่างทางด้านระบบคอมพิวเตอร์ เพื่อทำการสร้างภาษาที่ใช้กำหนดโครงสร้างของเอกสารให้สามารถเข้าใจกันได้ในทุกๆระบบ เอ็กซ์เอ็มแอลเป็นภาษาที่เป็นภาษาเมต้า (Meta Language) ซึ่งเป็นรูปแบบการรวบรวมข้อมูลข่าวสาร ที่นำมาจากแหล่งข้อมูลข่าวสารอื่นๆแต่ละหัวข้อในบทอ้างอิงจึงเป็น ข้อมูลที่ได้รวบรวมมาจากข้อมูลในเนื้อหา บทอ้างอิงจะสามารถบอกข้อมูลเกี่ยวกับสมาชิก (Element) และแอตทริบิวต์ (Attribute) ซึ่งจะมีเนื้อหาต่อไปได้

ส่วนประกอบของเอ็กซ์เอ็มแอล มีดังนี้

- แท็ก ซึ่งแท็กในเอ็กซ์เอ็มแอลมีความหมายในลักษณะเดียวกับแท็กในเฮชทีเอ็มแอล (HTML) คือข้อความที่อยู่ระหว่างสัญลักษณ์ < และ >
- อีลิเมนต์ (Element) หมายถึงส่วนของข้อมูลที่ประกอบไปด้วยแท็กเปิดและแท็กปิดรวมกัน เป็นโครงสร้างหลักของเอ็กซ์เอ็มแอล
- แอตทริบิวต์ คือข้อมูลความหมายเพิ่มเติม
- เนื้อความ (Content) คือ ข้อมูลเพื่อใช้ในการแสดงให้ผู้อ่านเอกสารได้เห็น
- เอนทิตี้ (Entity) คือ สัญลักษณ์ที่ไม่ใช่ข้อความที่ต้องการแสดงในเอกสาร

โดยเอ็กซ์เอ็มแอลสามารถนำมาใช้กับแผนภาพยูเอ็มแอลได้โดยการแปลงให้อยู่ในรูปของเอ็กซ์เอ็มไอ (XML Metadata Interchange) ซึ่งใช้ในการแปลงแผนภาพให้อยู่ในรูปของภาษาเมต้า

2.1.3.1 **เอ็กซ์เอ็มไอ** (XMI) [17]

เอ็กซ์เอ็มไอ (XML Metadata Interchange) คือ การกำหนดรูปแบบข้อมูลที่ใช้ในการแปลงแบบจำลองต่างๆให้อยู่ในรูปของข้อความ โดยเอ็กซ์เอ็มไอถูกสร้างขึ้นเพื่อจุดประสงค์ในการแลกเปลี่ยนข้อมูลที่อยู่ในรูปของภาษาเมต้าระหว่างเครื่องมือที่ใช้ในการออกแบบแบบจำลองต่างๆของยูเอ็มแอล โดยเน้นมาตรฐานของโอเอ็มจี (OMG: Object Management Group)

เอ็กซ์เอ็มไอรวมมาตรฐานอุตสาหกรรมที่สำคัญสามอย่าง คือ เอ็กซ์เอ็มแอล ยูเอ็มแอล และเอ็มโอเอฟ (MOF) การบูรณาการเทคโนโลยีทั้งสามเข้าสู่เอ็กซ์เอ็มไอช่วยให้นักพัฒนาของระบบสามารถกระจายแบบจำลองวัตถุของตนเองไปยังนักพัฒนาคนอื่นๆได้ โดยรูปแบบของข้อมูลเอ็กซ์เอ็มไอจะอยู่ในรูปของเอ็กซ์เอ็มแอล เอ็กซ์เอ็มไอได้ถูกพัฒนามาหลายรุ่นจนถึงรุ่นในปัจจุบัน คือ เอ็กซ์เอ็มไอ 2.4

```
<Foundation.Core.Namespace.ownedElement>
      <Foundation.Extension_Mechanisms.Stereotype xmi.id="BHpHQ9SGAqACIgQn">
            <Foundation.Core.ModelElement.name>Interface</Foundation.Core.ModelElement.name>
      </Foundation.Extension_Mechanisms.Stereotype>
      <Foundation.Core.Class xmi.id="iHpHQ9SGAqACIgQl">
            <Foundation.Core.ModelElement.name>X1</Foundation.Core.ModelElement.name>
            <Foundation.Core.ModelElement.visibility xmi.value="public"/>
            <Foundation.Core.GeneralizableElement.isRoot xmi.value="false"/>
            <Foundation.Core.GeneralizableElement.isLeaf xmi.value="false"/>
            <Foundation.Core.GeneralizableElement.isAbstract xmi.value="false"/>
            <Foundation.Core.Class.isActive xmi.value="false"/>
            <XMI.extension xmi.extender="Visual Paradigm for UML">
                  <businessModel xmi.value="false"/>
                  <modelType xmi.value="Class"/>
            </XMI.extension>
            <Foundation.Core.ModelElement.stereotype>
                  <Foundation.Extension_Mechanisms.Stereotype xmi.idref="BHpHQ9SGAqACIgQn">
                        <Foundation.Core.ModelElement.name>Interface</Foundation.Core.ModelElement.name>
                  </Foundation.Extension_Mechanisms.Stereotype>
            </Foundation.Core.ModelElement.stereotype>
      </Foundation.Core.Class>
      <Foundation.Core.Class xmi.id="flIfQ9SGAqACIgjZ">
            <Foundation.Core.ModelElement.name>A3</Foundation.Core.ModelElement.name>
            <Foundation.Core.ModelElement.visibility xmi.value="public"/>
            <Foundation.Core.GeneralizableElement.isRoot xmi.value="false"/>
            <Foundation.Core.GeneralizableElement.isLeaf xmi.value="false"/>
            <Foundation.Core.GeneralizableElement.isAbstract xmi.value="false"/>
            <Foundation.Core.Class.isActive xmi.value="false"/>
            <XMI.extension xmi.extender="Visual Paradigm for UML">
                  <businessModel xmi.value="false"/>
                  <modelType xmi.value="Class"/>
            </XMI.extension>
      </Foundation.Core.Class>
```

ภาพที่ 2.20 แสดงตัวอย่างของแฟ้มเอ็กซ์เอ็มไอที่ได้จากการแปลงแผนภาพคลาส

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.2.1 งานวิจัยของ Dirk Heuzeroth และคณะ [18]

งานวิจัยนี้จะเป็นการตรวจจับแบบรูปอัตโนมัติโดยใช้การวิเคราะห์เชิงสถิตย์ (Static Analysis) และการวิเคราะห์เชิงพลวัต (Dynamic Analysis) โดยการวิเคราะห์เชิงสถิตย์จะพิจารณาจากรหัสต้นฉบับ (Source Code) โดยข้อมูลที่ใช้ในการวิเคราะห์ประกอบไปด้วยคลาส ลักษณะประจำ เมท็อด และพารามิเตอร์ (parameter) โดยนำข้อมูลหล่านี้มาเปรียบเทียบกับกฎที่สร้างขึ้น โดยกฎที่ใช้ในการวิเคราะห์ได้มาจากการพิจารณาข้อมูลของแบบรูปการออกแบบ ผลลัพธ์ที่ได้จากการวิเคราะห์เชิงสถิตย์จะได้เป็นกลุ่มของคลาส (Candidate) ที่เป็นแบบรูปการออกแบบ จากนั้นจะนำกลุ่มของคลาสเหล่านี้มาทำการประมวลผล เพื่อทำการวิเคราะห์เชิงพลวัต โดยการวิเคราะห์เชิงพลวัตจะเป็นการพิจารณาจากรหัสต้นฉบับที่ทำการประมวลผล ซึ่งจะพิจารณาการเรียกใช้เมท็อด การสร้างวัตถุของคลาส โดยการวิเคราะห์เชิงพลวัตจะมีการสร้างกฎเช่นเดียวกับการวิเคราะห์เชิงสถิตย์ ข้อจำกัดของงานวิจัยนี้คือ การวิเคราะห์เชิงสถิตย์จะขึ้นกับภาษาที่ใช้ โดยวิเคราะห์ได้เพียงโปรแกรมที่เขียนด้วยภาษาจาวา (Java language) เท่านั้น ส่วนการวิเคราะห์เชิงพลวัตจะพบปัญหาเมื่อบางคลาสหรือเมท็อดไม่มีการถูกเรียกใช้งาน ความเชื่อมั่นของเครื่องมือนี้สามารถตรวจสอบด้วยมือ (Hand-checking) เพื่อยืนยันความถูกต้อง โดยนำข้อมูลที่วิเคราะห์ได้มาเปรียบเทียบกับเอกสารของระบบที่พัฒนาขึ้น งานวิจัยที่เกี่ยวข้องชิ้นนี้เป็นที่มาของงานวิจัยที่ทำขึ้นในเรื่องของการสร้างกฎของแบบรูปการออกแบบขึ้นมา เพื่อใช้ในการวิเคราะห์โครงสร้างของคลาส สิ่งที่ต่างระหว่างงานวิจัยที่เกี่ยวข้องชิ้นนี้กับงานวิจัยที่ทำขึ้น คือ จะใช้การวิเคราะห์เชิงสถิตย์เท่านั้น และการวิเคราะห์เชิงสถิตย์ที่ใช้จะไม่ได้ทำการวิเคราะห์จากรหัสต้นฉบับ แต่จะทำการวิเคราะห์จากแผนภาพคลาส ซึ่งแผนภาพคลาสนั้นสามารถแสดงข้อมูลที่ต้องใช้ในการวิเคราะห์ได้เหมือนกับรหัสต้นฉบับ

2.2.2 งานวิจัยของ Nikolaos Tsantalis และคณะ [14,19]

งานวิจัยชิ้นนี้เป็นการตรวจจับแบบรูปการออกแบบโดยใช้ข้อมูลเป็นแผนภาพคลาส โดยจะทำการแบ่งกลุ่ม (Segment) ของแผนภาพคลาสของระบบ และแผนภาพของแบบรูปด้วยความสัมพันธ์ชนิดต่างๆของแผนภาพคลาส จากนั้นจะทำการแปลงแผนภาพคลาสให้อยู่ในรูปของกราฟ (Graph) ซึ่งกราฟที่ใช้ในงานวิจัยชิ้นนี้มีทั้งสิ้น 2 ส่วน คือ กราฟของระบบที่ต้องการตรวจสอบ และกราฟของแบบรูปการออกแบบ และทำการสร้างเมตริกซ์ความสัมพันธ์ของจุด

(Vertices) ของกราฟ โดยเมตริกซ์นี้จะเป็นเมตริกซ์ของความสัมพันธ์ของระบบ และแบบรูปการออกแบบ จากนั้นใช้ขั้นตอนวิธีการคำนวณหาคะแนนความคล้ายคลึง (Similarity Scoring Algorithm) ซึ่งขั้นตอนวิธีคำนวณหาคะแนนความคล้ายคลึงจะเป็นคำนวนหาคะแนนความคล้ายระหว่างจุดของกราฟจากเมตริกซ์ที่สร้างขึ้น หลังจากได้คะแนนในแต่ละความสัมพันธ์แล้ว จะทำการนำเมตริกต์ของคะแนนเหล่านั้นของระบบมาคำนวณหาคะแนนรวมของความคล้ายกับแบบรูปการออกแบบที่สนใจ งานวิจัยที่เกี่ยวข้องชิ้นนี้เป็นที่มาของงานวิจัยที่ทำขึ้นในเรื่องการใช้แผนภาพคลาสมาทำการวิเคราะห์เทียบกับแบบรูปการออกแบบ สิ่งที่ต่างระหว่างงานวิจัยที่เกี่ยวข้องชิ้นนี้กับงานวิจัยที่ทำขึ้น คือ ในงานวิจัยนี้จะใช้การแปลงข้อมูลให้อยู่ในรูปของเมตริกซ์และมีการใช้ขั้นตอนวิธีการคำนวณหาคะแนนความคล้ายคลึง แต่ในงานวิจัยที่ทำขึ้นจะเป็นการสร้างกฎสำหรับแต่ละแบบรูปการออกแบบ

2.2.3 **งานวิจัยของ** Jing Dong **และคณะ** [15]

งานวิจัยนี้เป็นการตรวจจับแบบรูปการออกแบบโดยใช้ข้อมูลเป็นแผนภาพคลาส และใช้วิธีการเทมเพลทแมตชิ่ง (Template Matching) ในการตรวจจับแบบรูปการออกแบบซึ่งวิธีการที่แตกต่างกับงานวิจัย [14] คือ แผนภาพคลาสของแบบรูปการออกแบบ และระบบที่ต้องการตรวจสอบจะถูกแปลงให้อยู่ในรูปของเมตริกซ์ (Matrix) ซึ่งเป็นเมตริกซ์ผลรวมของทุกๆคลาสในแผนภาพคลาส โดยใช้ผลคูณสหสัมพันธ์ (Cross Correlation) จากนั้นประยุกต์ใช้ขั้นตอนวิธีการคำนวณหาคะแนนความคล้ายคลึงทำการหาคะแนนความคล้ายคลึงของจุดทั้งหมด งานวิจัยที่เกี่ยวข้องชิ้นนี้เป็นที่มาของงานวิจัยที่ทำขึ้นในเรื่องการใช้แผนภาพคลาสมาทำการวิเคราะห์เทียบกับแบบรูปการออกแบบ และทำให้เห็นว่าสามารถที่จะไม่แบ่งความสัมพันธ์ของแผนภาพคลาสออกเป็นส่วนๆได้

งานวิจัยทั้ง 3 นี้สามารถตรวจจับไม่ใช่เพียงการจับคู่แบบแม่นตรง (Exact Matching) เท่านั้น แต่สามารถตรวจจับแบบรูปที่มีการเปลี่ยนแปลง (Inexact Matching) แต่ยังคงถือว่าเป็นแบบรูปอยู่ได้ และการทดสอบกฎหรือขั้นตอนวิธีที่สร้างขึ้นจะมีการประเมินความถูกต้องของผลลัพธ์ในการตรวจจับแบบรูปการออกแบบโดยนำเอกสารของระบบมาเปรียบเทียบ ซึ่งต่างจากงานวิจัยนี้ คือ งานวิจัยนี้สามารถตรวจจับแบบรูปการออกแบบที่มีการเพิ่มหรือลดความสัมพันธ์บางประการลงได้ โดยงานวิจัยที่เกี่ยวข้องสามารถตรวจจับได้เพียงกรณีที่มีการเพิ่มความสัมพันธ์ เช่น การสืบทอดเท่านั้น แต่ไม่สามารถตรวจจับในกรณีที่ลดความสัมพันธ์ระหว่างคลาส หรือกรณี

ที่เป็นความสัมพันธ์แบบยกเว้นบางประการ เช่น เปลี่ยนจากความสัมพันธ์แบบอะกรีเกชันเป็นความสัมพันธ์แบบแอสโซซิเอชัน

บทที่ 3
แนวคิดและขั้นตอนวิธี

งานวิจัยนี้เป็นการออกแบบ และพัฒนาเครื่องมือในการตรวจจับแบบรูปการออกแบบ ซึ่งขั้นตอนวิธี (Algorithm) ที่ใช้ในการตรวจสอบแบบรูปการออกแบบได้มาจากการวิเคราะห์แบบรูปการออกแบบในแต่ละแบบรูปเพื่อหาจุดเด่นของแต่ละแบบรูปการออกแบบ และทำการสรุปการทำงาน และรูปแบบการเขียนแบบรูปการออกแบบ แล้วนำข้อมูลเหล่านั้นมาสร้างเป็นกฎสำหรับแต่ละแบบรูป และนำกฎที่ได้นี้มาแปลงเป็นขั้นตอนวิธีสำหรับแต่ละแบบรูปการออกแบบ

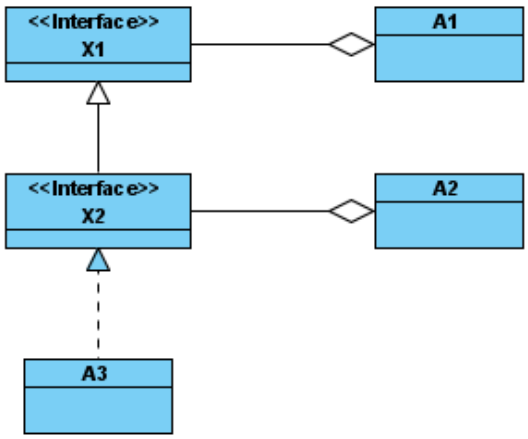
3.1 นิยามของคำศัพท์ที่ใช้ในงานวิจัย

3.1.1 โหนดหลัก (Anchor node)

ในงานวิจัยชิ้นนี้มีการกำหนดโหนดหลักขึ้นมาโดยโหนดหลัก คือ โหนดที่ใช้เป็นจุดเริ่มต้นของการค้นหารายละเอียดของแบบรูปการออกแบบ สาเหตุที่ต้องทำการกำหนดโหนดหลักนั้นเพื่อทำให้การตรวจสอบทำได้รวดเร็วขึ้น และหากมีการเปลี่ยนแปลงรูปแบบการวาดแผนภาพคลาสก็จะไม่ส่งผลต่อการตรวจจับแบบรูป เนื่องจากทำการกำหนดจุดเริ่มต้นของการค้นหา แล้วทำการค้นหารายละเอียดของแบบรูปโดยเริ่มจากโหนดหลักเท่านั้น ไม่ต้องทำการค้นหาจากคลาสทั้งหมด

3.1.2 แบบรูปดัดแปลงจากมาตรฐาน (Modified Design Pattern)

แบบรูปที่ดัดแปลงจากมาตรฐาน เป็นแบบรูปที่มีการดัดแปลงลักษณะบางประการของแบบรูป เนื่องมาจากบางครั้งในการนำไปใช้งานจริง แบบรูปมาตรฐานอาจมีการเปลี่ยนแปลงโครงสร้างบางส่วนโดยนักออกแบบ เพื่อให้โครงสร้างของแบบรูปที่นำมาใช้เหมาะสมกับข้อจำกัดของระบบ ซึ่งการดัดแปลงจะดัดแปลงโครงสร้างบางประการ เช่น เพิ่มส่วนของการคลาสที่มีการสืบทอด มีการเพิ่มหรือลดความสัมพันธ์ระหว่างคลาส มีการเปลี่ยนแปลงความสัมพันธ์บางประการเพื่อให้เหมาะสมกับงาน เป็นต้น ดังแสดงในภาพที่ 3.1



ภาพที่ 3.1 แสดงตัวอย่างแบบรูปที่ดัดแปลงจากมาตรฐานของแบบรูป Strategy [15]

โดยแบบรูปในลักษณะนี้ยังคงมีความหมายเป็นไปตามแบบรูปมาตรฐานเดิม ดังที่กล่าวถึงในงานวิจัย

3.2 ขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบพฤติกรรม

ขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบสามารถแบ่งได้ออกเป็น 2 ส่วน ดังนี้

1) การค้นหาโหนดหลัก (anchor node) : การค้นหาโหนดหลักของแบบรูปการออกแบบทำได้โดยทำการค้นหาจากคลาสที่มีทั้งหมด เปรียบเทียบหาคุณสมบัติของโหนดหลักแต่ละโหนดของแต่ละแบบรูปการออกแบบ และนำโหนดที่ค้นหาได้ใส่ลงในลิสต์ของโหนดหลักของแต่ละแบบรูปการออกแบบ ซึ่งจะส่งผลให้ในแต่ละแบบรูปการออกแบบจะมีโหนดหลักมากกว่า 1 และโหนดหลักบางโหนดอาจปรากฏได้ในหลายแบบรูปการออกแบบ แต่จะถูกนำไปค้นหาอย่างละเอียดในขั้นตอนต่อไป

2) การค้นหาแบบรูปการออกแบบจากโหนดหลัก : หลังจากค้นหาโหนดหลักเรียบร้อยแล้วจะใช้กฎเพื่อทำการค้นหาคลาสอื่นๆที่มีความสัมพันธ์กับโหนดหลักนั้นซึ่งจะมีลักษณะตรงกับแบบรูปการออกแบบ โดยการค้นหาคลาสประกอบอื่นๆเหล่านี้จะมีการค้นหาทั้งแบบที่เป็นแบบรูปมาตรฐาน และแบบรูปที่ดัดแปลงจากมาตรฐาน ซึ่งแบบรูปดัดแปลงจากมาตรฐานได้มาจากการวิเคราะห์ความสัมพันธ์ต่างๆที่ใช้งานกันโดยทั่วไป

ขั้นตอนวิธีจะถูกออกแบบเพื่อตรวจจับแบบรูปการออกแบบทั้งสิ้น 7 แบบรูปการออกแบบ ประกอบไปด้วยแบบรูปการออกแบบ Chain of Responsibility แบบรูปการออกแบบ Command แบบรูปการออกแบบ Iterator แบบรูปการออกแบบ Memento แบบรูปการออกแบบ Observer แบบรูปการออกแบบ Strategy และแบบรูปการออกแบบ Template method

3.2.1 แบบรูป Chain of responsibility

3.1.1.1 โครงสร้างของแบบรูป Chain of responsibility

1) แบบรูปตามมาตรฐาน

   a. ประกอบไปด้วยคลาสที่ทำหน้าที่เป็น Handler ซึ่งมีชนิดของคลาสเป็นคลาสอินเตอร์เฟส (Interface) คลาสที่ทำหน้าที่เป็น Client ซึ่งมีชนิดของคลาสเป็นคลาสรูปธรรม (Concrete) และคลาสที่ทำหน้าที่เป็น ConcreteHandler ซึ่งมีชนิดของคลาสรูปธรรม

   b. คลาสที่ทำหน้าที่เป็น Handler ต้องมีอิมพลีเมนต์คลาสอย่างน้อย 1 คลาส คือ ConcreteHandler และ Handler ต้องมีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเอง

   c. คลาสที่ทำหน้าที่เป็น Client ต้องมีความสัมพันธ์แบบแอสโซซิเอชันกับ Handler และห้ามมีความสัมพันธ์กับ ConcreteHandler

2) แบบรูปดัดแปลงจากมาตรฐาน

   a. คลาสที่ทำหน้าที่เป็น Handler สามารถมีชนิดของคลาสเป็นคลาสนามธรรม (Abstract) ได้ โดยหากมีชนิดเป็นคลาสนามธรรมจะต้องมีความสัมพันธ์แบบเจนเนอรัลไลเซชันกับ ConcreteHandler

   b. หากคลาสที่ทำหน้าที่เป็น Handler ไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเอง คลาสที่ทำหน้าที่เป็น ConcreteHandler ต้องมีความสัมพันธ์แบบอะกรีเกชันกับ Handler

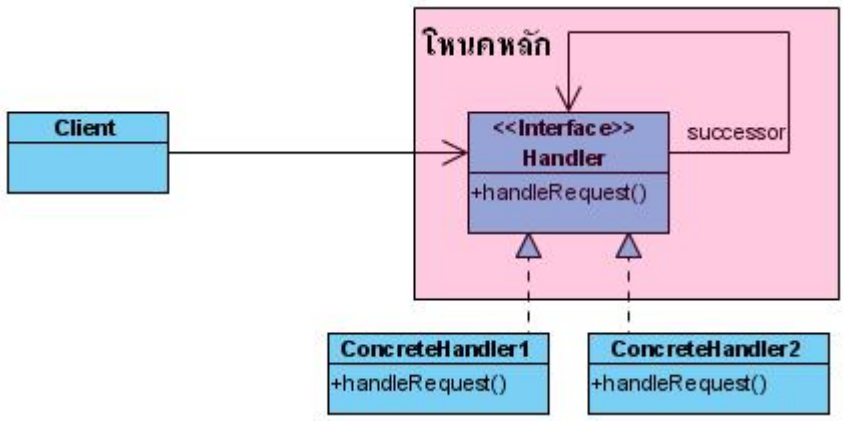
c. กรณีที่คลาสที่ทำหน้าที่เป็น Handler มีชนิดของคลาสเป็นอินเตอร์เฟส คลาสสามารถมีการสืบทอดได้ แต่คลาสที่สืบทอดคลาสสุดท้ายจะต้องมีอิมพลีเมนต์คลาสซึ่งทำหน้าที่เป็น ConcreteHandler

โดยทั้งแบบรูปตามมาตรฐาน และแบบรูปดัดแปลงจากมาตรฐาน หากอิมพลีเมนต์คลาสของ Handler (ConcreteHandler) มีการสืบทอดจะไม่ถือว่าคลาสที่สืบทอดต่อไปเหล่านั้นเป็นแบบรูปการออกแบบ

จากโครงสร้างของแบบรูปการออกแบบที่กล่าวมาข้างต้น ทำให้กำหนดคุณสมบัติของคลาสที่เป็นโหนดหลักของแบบรูป Chain of Responsibility ได้ดังต่อไปนี้

1) เป็นคลาสที่มีชนิดของคลาสได้ใน 2 รูปแบบคือ คลาสอินเตอร์เฟส หรือ คลาสนามธรรม
2) เป็นคลาสที่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือมีอิมพลีเมนต์คลาสที่มีความสัมพันธ์แบบอะกรีเกชันมายังคลาสตนเอง
3) เป็นคลาสที่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสรูปธรรมใดๆ

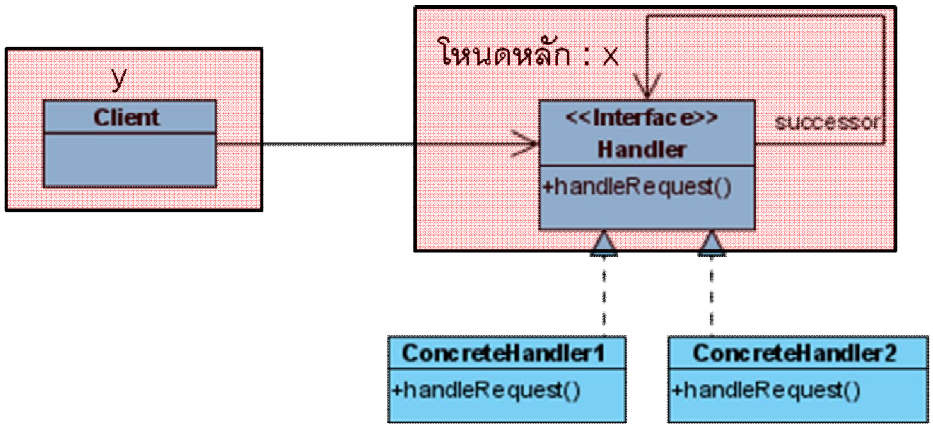
จากคุณสมบัติดังกล่าวจะเห็นได้ว่า โหนดหลัก คือ คลาสที่ทำหน้าที่เป็น Handler นั่นเอง



ภาพที่ 3.2 ตัวอย่างโหนดหลักของแบบรูปการออกแบบ Chain of Responsibility

สามารถอธิบายขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบได้ดังต่อไปนี้

ตามขั้นตอนวิธี A1 ซึ่งแสดงในภาพที่ 3.4 ในหัวข้อ 3.1.1.2 เป็นขั้นตอนวิธีในการค้นหาโหนดหลักของแบบรูปการออกแบบ Chain of Responsibility โดยเริ่มต้นการทำงานด้วยการค้นหาคลาสที่ทำหน้าที่เป็น Handler ซึ่งเป็นคลาสที่มีชนิดของคลาสเป็นคลาสอินเตอร์เฟซหรือคลาสนามธรรม เมื่อได้คลาสที่มีชนิดของคลาสดังกล่าวแล้วจะทำการตรวจสอบว่า คลาสเหล่านี้มีคลาสใดบ้างที่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือมีความสัมพันธ์แบบอะกรีเกชัน โดยให้คลาสที่มีคุณสมบัติเหล่านี้เก็บไว้ใน TempAnchor จากนั้นจะนำคลาสที่เก็บใน TempAnchor มาทำการตรวจสอบว่า มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสอื่นๆหรือไม่ กรณีที่มีจะทำการเก็บคลาสนี้เป็นโหนดหลักและเก็บข้อมูลของคลาสที่มีความสัมพันธ์ไว้ด้วย โดยเก็บไว้เป็นคู่ลำดับ (x,y) ซึ่ง x จะแทนคลาสที่เป็นโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Handler และ y แทนคลาสที่มีความสัมพันธ์กับโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Client



ภาพที่ 3.3 ตัวอย่างผลลัพธ์จากขั้นตอนวิธี A1ของแบบรูปการออกแบบ Chain of Responsibility

ตามขั้นตอนวิธี A2 ซึ่งแสดงในภาพที่ 3.5 3.6 และ 3.7 ในหัวข้อ 3.1.1.2 หลังจากได้ชุดของโหนดหลัก (คู่ลำดับ x และ y) แล้ว จะนำชุดข้อมูล (x, y) นี้ไปทำการตรวจสอบเพื่อทำการค้นหารายละเอียดของแบบรูปการออกแบบโดยจะทำการตรวจสอบข้อมูลของ y ก่อน โดยจะทำการตรวจสอบว่า y มีชนิดของคลาสเป็นคลาสรูปธรรมใช่หรือไม่ ซึ่งหากไม่ใช่จะทำการหยุดตรวจสอบชุดข้อมูลนี้ทันทีเนื่องจากไม่เป็นไปตามโครงสร้างของแบบรูปการออกแบบ Chain of Responsibility แต่ถ้าใช่จะทำการตรวจสอบในขั้นต่อไป

ขั้นตอนถัดไปเป็นขั้นตอนในการตรวจสอบหาอิมพลีเมนต์คลาสของโหนดหลัก โดยทำการตรวจสอบหาคลาสที่มีความสัมพันธ์รีไลเซชันกับโหนดหลักหากมีจะทำการตรวจสอบที่โหนดหลักว่าโหนดหลักมีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือไม่ ถ้าใช่จะตรวจสอบว่าอิมพลีเมนต์คลาสเหล่านี้มีความสัมพันธ์กับ y หรือไม่ ถ้าไม่มีจะทำให้ได้ชุดของคลาสที่เป็นแบบรูปการออกแบบ Chain of Responsibility โดย x คือ คลาสที่ทำหน้าที่เป็น Handler ส่วน y คือ คลาสที่ทำหน้าที่เป็น Client และ อิมพลีเมนต์คลาสของโหนดหลักทั้งหมด คือคลาสที่ทำหน้าที่เป็น ConcreteHandler ในกรณีที่โหนดหลักไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองจะทำการตรวจสอบหาความสัมพันธ์ระหว่างโหนดหลักกับอิมพลีเมนต์ของโหนดหลักว่า อิมพลีเมนต์ของโหนดหลักนั้นมีความสัมพันธ์อะกรีเกชันกับโหนดหลักหรือไม่ โดยถ้ามีจะตรวจสอบว่าอิมพลีเมนต์คลาสเหล่านี้มีความสัมพันธ์กับ y หรือไม่ ถ้าไม่มีจะทำให้ได้ชุดของคลาสที่เป็นแบบรูปการออกแบบ Chain of Responsibility โดย x คือ คลาสที่ทำหน้าที่เป็น Handler ส่วน y คือ คลาสที่ทำหน้าที่เป็น Client และ อิมพลีเมนต์คลาสของโหนดหลักทั้งหมด คือคลาสที่ทำหน้าที่เป็น ConcreteHandler ซึ่งในกรณีนี้จะถือว่าเป็นแบบรูปที่มีการดัดแปลงจากมาตรฐาน

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานอีกกรณีหนึ่งคือ การที่โหนดหลักเป็นคลาสอินเตอร์เฟสแต่แทนที่จะมีความสัมพันธ์แบบรีไลเซชันกลับมีความสัมพันธ์แบบเจนเนอรัลไลเซชัน ในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบคลาสสุดท้ายที่มีการสืบทอดว่าเป็นอิมพลีเมนต์คลาสหรือไม่ ถ้าใช่ก็จะทำการตรวจสอบการเรียกใช้ตัวเองของโหนดหลักและตรวจสอบความสัมพันธ์แบบอะกรีเกชันของโหนดหลักและอิมพลีเมนต์คลาส ถ้ามีความสัมพันธ์ในลักษณะที่กล่าวมาในข้างต้นและอิมพลีเมนต์คลาสไม่มีความสัมพันธ์กับ y จะได้ว่าคลาสทั้งหมดเป็นแบบรูปการออกแบบ

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานกรณีสุดท้ายคือ การที่โหนดหลักเป็นคลาสนามธรรมแทนที่จะตรวจสอบหาความสัมพันธ์แบบรีไลเซชัน จะต้องทำการตรวจสอบหาความสัมพันธ์แบบเจนเนอรัลไลเซชันแทนโดยในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบการเรียกใช้ตัวเองของโหนดหลักและตรวจสอบความสัมพันธ์แบบอะกรีเกชันของโหนดหลักและคลาสสุดท้ายของการสืบทอด ถ้ามีความสัมพันธ์ในลักษณะที่กล่าวมาในข้างต้นและคลาสที่มีการสืบทอดมาจากโหนดหลักไม่มีความสัมพันธ์กับ y จะได้ว่าคลาสทั้งหมดเป็นแบบรูปการออกแบบ

3.1.1.2 ขั้นตอนวิธีในการตรวจจับแบบรูป Chain of responsibility

A1: The "Anchor Node" finder Algorithm for **Chain of responsibility** Pattern

For Each Class ***C*** in Class Diagram ***CD***

  If Class Type of ***C*** equals "Interface" or "Abstract" then

    For Each Relation ***R*** of ***C***

      If Type of ***R*** equals "SelfAssociation" or "AggregationC" then

        Append ***x*** to ***TempAnchor*** where x is an Interface class

For Each Class ***C*** in ***TempAnchor***

  For Each Relation ***R*** of ***C***

    If Type of ***R*** equals "AssociationP" or "AggregationC"then

      Append (***x***,***y***) to ***AnchorChain*** where x is an interface class and y is related class with x

ภาพที่ 3.4 ขั้นตอนวิธี A1 ในการตรวจจับแบบรูปการออกแบบ Chain of Responsibility

A2: The "Design Pattern" finder Algorithm for **Chain of responsibility** Pattern

For Each (***x***,***y***) in ***AnchorChain***

  Read File for Getting Data ***attr*** of ***y***

  Append ***attr*** to ***YAttr***

  If Class Type of ***YAttr*** equals "Concrete" then

    For Each Relation ***R*** of ***x***

      If Type of ***R*** equals "SelfAssociation" then

        Append ***x*** to ***ListTemp***

        Append ***y*** to ***ListTemp***

        countClass++

ภาพที่ 3.5 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Chain of Responsibility

For Each Relation ***R2*** of ***x***

If Type of ***R2*** equals "ImplementationP" then

Append ***impX*** to ***ListTemp*** where impX is implementation class of x

If Type of ***R2*** equals "GeneralizationP" then

Call Perform Function ***GetCascadingInheritance(x)***

Append ***iClass*** to ***TempInherit*** where iClass contains all of the Cascading Inheritance classes returned from Function GetCascadingInheritance()

Append ***TempInherit*** to ***ListTemp***

If Type of ***R*** equals "AggregationC" then

Read File for Getting Data ***D*** of Related Class ***Rt***

For Each Relation ***R2*** of ***D***

If Type of ***R2*** equals "ImplementationC" or "GeneralizationC" and Related Class is ***x*** then

Append ***impX*** to ***CallInf*** where impX is implementation class or interit class of x

For Each Class ***C*** in ***CallInf***

For Each Relation ***R2*** of ***C***

If Type of ***R2*** equals "AggregationP" and Related Class is ***x*** then

countClass++

Append ***C*** to ***TempImp***

If ***countClass*** >= 1 then

Append ***x*** to ***ListTemp***

Append ***y*** to ***ListTemp***

Append ***TempImp*** to ***ListTemp***

If Type of ***R*** equals "GenaralizationP" then

Call Perform Function ***GetCascadingInheritance(x)***

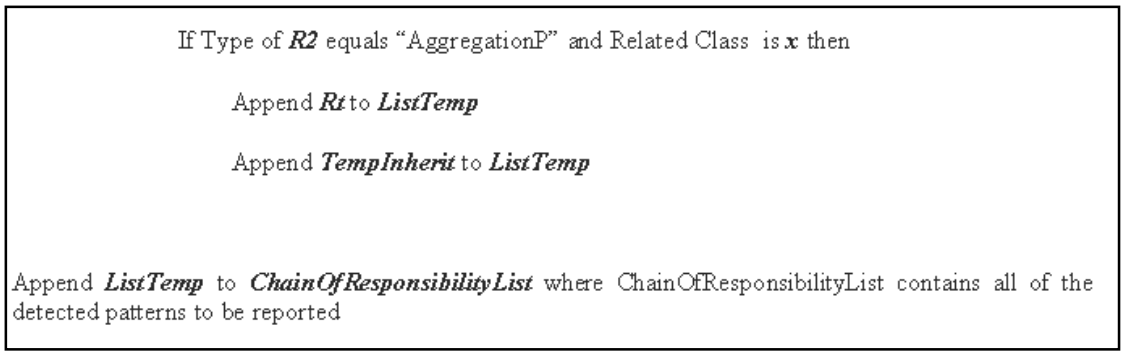
Append ***iClass*** to ***TempInherit*** where iClass contains all of the Cascading Inheritance classes returned from Function GetCascadingInheritance()

Read File for Getting Data ***D*** of Last Index of ***TempInherit***

If Class Type of ***x*** equals "Interface" then

For Each Relation ***R2*** of ***D***

ภาพที่ 3.6 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Chain of responsibility (ต่อ)



ภาพที่ 3.7 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Chain of responsibility (ต่อ)

3.2.2 แบบรูป Command

3.1.2.1 โครงสร้างของแบบรูป Command

1) แบบรูปตามมาตรฐาน

a. ประกอบไปด้วยคลาสที่ทำหน้าที่เป็น Command ซึ่งมีชนิดของคลาสเป็นคลาสอินเตอร์เฟส (Interface) คลาสที่ทำหน้าที่เป็น Invoker มีชนิดของคลาสเป็นคลาสรูปธรรม (Concrete) คลาสที่ทำหน้าที่เป็น ConcreteCommand มีชนิดของคลาสเป็นคลาสรูปธรรม คลาสที่ทำหน้าที่เป็น Receiver มีชนิดของคลาสเป็นคลาสรูปธรรม และคลาสที่ทำหน้าที่เป็น Client มีชนิดของคลาสเป็นคลาสรูปธรรม

b. คลาสที่ทำหน้าที่เป็น Command ต้องมีอิมพลีเมนต์คลาสอย่างน้อย 1 คลาส คือ ConcreteCommand ต้องไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเอง และไม่มีความสัมพันธ์กับ ConcreteCommand ในรูปแบบอื่นๆนอกเหนือจากรีไลเซชันและเจนเนอรัลไลเซชัน

c. คลาสที่ทำหน้าที่เป็น Invoker ต้องมีความสัมพันธ์แบบอะกรีเกชันกับ Command และห้ามมีความสัมพันธ์กับ ConcreteCommand

d. คลาสที่ทำหน้าที่เป็น ConcreteCommand จะมีความสัมพันธ์แบบแอสโซซิเอชันไปยังคลาสที่ทำหน้าที่เป็น Receiver โดยหาก

ConcreteCommand มีมากกว่า 1 คลาส ทุกคลาสจะต้องมีความสัมพันธ์กับ Receiver

e. คลาสที่ทำหน้าที่เป็น Receiver จะมีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสที่ทำหน้าที่เป็น Client โดยที่คลาสที่ทำหน้าที่เป็น Client นี้จะต้องมีความสัมพันธ์แบบดีเพนเดนซีกับ ConcreteCommand ด้วย

2) แบบรูปดัดแปลงจากมาตรฐาน

a. คลาสที่ทำหน้าที่เป็น Command สามารถมีชนิดของคลาสเป็นคลาสนามธรรม (Abstract) ได้ โดยหากมีชนิดเป็นคลาสนามธรรมจะต้องมีความสัมพันธ์แบบเจนเนอรัลไลเซชันกับ ConcreteCommand

b. คลาสที่ทำหน้าที่เป็น Client สามารถมีความสัมพันธ์แบบแอสโซซิเอชันกับ ConcreteCommand แทนความสัมพันธ์แบบดีเพนเดนซีได้

c. ไม่มีคลาสที่ทำหน้าที่เป็น Client ได้

d. กรณีที่คลาสที่ทำหน้าที่เป็น Command มีชนิดของคลาสเป็นอินเตอร์เฟสคลาสสามารถมีการสืบทอดได้ แต่คลาสที่สืบทอดคลาสสุดท้ายจะต้องมีอิมพลีเมนต์คลาสซึ่งทำหน้าที่เป็น ConcreteCommand

โดยทั้งแบบรูปตามมาตรฐาน และแบบรูปดัดแปลงจากมาตรฐาน หากอิมพลีเมนต์คลาสของ Command (ConcreteCommand) มีการสืบทอดจะไม่ถือว่าคลาสที่สืบทอดต่อไปเหล่านั้นเป็นแบบรูปการออกแบบ

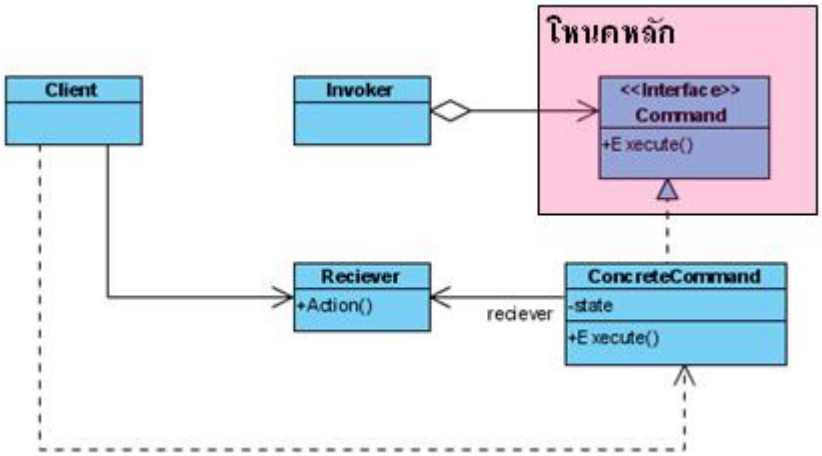
จากโครงสร้างของแบบรูปการออกแบบที่กล่าวมาข้างต้น ทำให้กำหนดคุณสมบัติของคลาสที่เป็นโหนดหลักของแบบรูป Command ได้ดังต่อไปนี้

1) เป็นคลาสที่มีชนิดของคลาสได้ใน 2 รูปแบบคือ คลาสอินเตอร์เฟสหรือคลาสนามธรรม

2) เป็นคลาสที่ไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือกับอิมพลีเมนต์คลาส

3) เป็นคลาสที่มีความสัมพันธ์แบบอะกรีเกชันกับคลาสรูปธรรมใดๆ

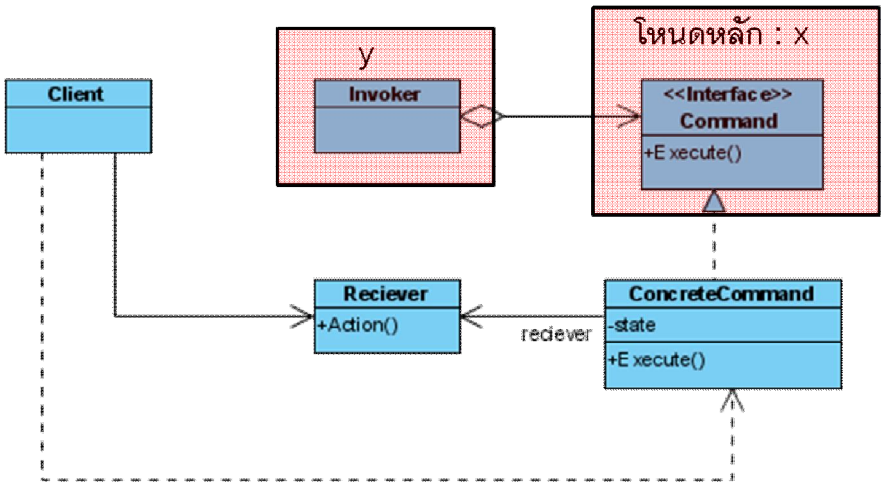
จากคุณสมบัติดังกล่าวจะเห็นได้ว่า โหนดหลัก คือ คลาสที่ทำหน้าที่เป็น Command นั่นเอง



ภาพที่ 3.8 ตัวอย่างโหนดหลักของแบบรูปการออกแบบ Command

สามารถอธิบายขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบได้ดังต่อไปนี้

ตามขั้นตอนวิธี A1 ซึ่งแสดงในภาพที่ 3.10 ในหัวข้อ 3.1.2.2 เป็นขั้นตอนวิธีในการค้นหาโหนดหลักของแบบรูปการออกแบบ Command โดยเริ่มต้นการทำงานด้วยการค้นหาคลาสที่ทำหน้าที่เป็น Command ซึ่งเป็นคลาสที่มีชนิดของคลาสเป็นคลาสอินเตอร์เฟซหรือคลาสนามธรรม เมื่อได้คลาสที่มีชนิดของคลาสดังกล่าวแล้วจะทำการตรวจสอบว่า คลาสเหล่านี้มีคลาสใดบ้างที่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองโดยหากพบจะไม่ทำการพิจารณาคลาสนั้นต่อ หากคลาสใดไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองจะทำการตรวจสอบหาความสัมพันธ์แบบอะกรีเกชันกับคลาสอื่นๆ กรณีที่มีจะทำการเก็บคลาสนี้เป็นโหนดหลักและเก็บข้อมูลของคลาสที่มีความสัมพันธ์ไว้ด้วย โดยเก็บไว้เป็นคู่ลำดับ (x,y) ซึ่ง x จะแทนคลาสที่เป็นโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Command และ y แทนคลาสที่มีความสัมพันธ์แบบอะกรีเกชันกับโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Invoker



ภาพที่ 3.9 ตัวอย่างผลลัพธ์จากขั้นตอนวิธี A1ของแบบรูปการออกแบบ Command

ตามขั้นตอนวิธี A2 ซึ่งแสดงในภาพที่ 3.11 และ 3.12 ในหัวข้อ 3.1.2.2 หลังจากได้ชุดของโหนดหลัก (คู่ลำดับ x และ y) แล้ว จะนำชุดข้อมูล (x, y) นี้ไปทำการตรวจสอบเพื่อทำการค้นหารายละเอียดของแบบรูปการออกแบบโดยจะทำการตรวจสอบข้อมูลของ y ก่อน โดยจะทำการตรวจสอบว่า y มีชนิดของคลาสเป็นคลาสรูปธรรมใช่หรือไม่ ซึ่งหากไม่ใช่จะทำการหยุดตรวจสอบชุดข้อมูลนี้ทันทีเนื่องจากไม่เป็นไปตามโครงสร้างของแบบรูปการออกแบบ Command แต่ถ้าใช่จะทำการตรวจสอบในขั้นต่อไป

ขั้นตอนถัดไปเป็นขั้นตอนในการตรวจสอบหาอิมพลีเมนต์คลาสของโหนดหลักหรือหา ConcreteCommand โดยทำการตรวจสอบหาคลาสที่มีความสัมพันธ์รีไลเซชันกับโหนดหลักหากมีจะทำการตรวจสอบว่าคลาสนี้มีความสัมพันธ์กับโหนดหลักนอกเหนือจากความสัมพันธ์แบบรีไลเซชันหรือไม่ถ้าไม่มีจะตรวจสอบว่าอิมพลีเมนต์คลาสเหล่านี้มีความสัมพันธ์กับ y หรือไม่ ถ้าไม่มีคลาสนี้จะถูกกำหนดว่าอาจมีความเป็นไปได้ที่จะเป็น ConcreteCommand หลังจากนั้นจะทำการตรวจสอบที่ ConcreteCommand ว่ามีความสัมพันธ์กับคลาสรูปธรรมอื่นๆที่ไม่ใช่ x และ y หรือไม่ถ้ามีจะทำการกำหนดคลาสที่มีความสัมพันธ์กับ ConcreteCommand ว่าอาจมีความเป็นไปได้ที่คลาสนั้นจะเป็น Receiver จากนั้นจะทำการตรวจสอบคลาส Receiver ว่ามีคลาสรูปธรรมอื่นๆมามีความสัมพันธ์แบบแอสโซซิเอชันหรือไม่ถ้ามีจะทำการตรวจสอบว่าคลาสที่มีความสัมพันธ์กับ Receiver นี้มีความสัมพันธ์แบบดีเพนเดนซีกับ ConcreteCommand หรือไม่โดยถ้ามีจะทำให้ได้

ชุดของคลาสที่เป็นแบบรูปการออกแบบ Command โดย x คือ คลาสที่ทำหน้าที่เป็น Command ส่วน y คือ คลาสที่ทำหน้าที่เป็น Invoker อิมพลีเมนต์คลาสของโหนดหลักทั้งหมด คือคลาสที่ทำหน้าที่เป็น ConcreteCommand คลาสที่มีความสัมพันธ์กับ ConcreteCommand คือ คลาสที่ทำหน้าที่เป็น Reciever และคลาสที่มีความสัมพันธ์กับ Receiver และ ConcreteCommand คือ คลาสที่ทำหน้าที่เป็น Client โดยการตรวจสอบแบบนี้จะเป็นการตรวจสอบแบบรูปแบบมาตรฐาน กรณีที่คลาสที่มีความสัมพันธ์กับ Receiver นั้นมีความสัมพันธ์แบบแอสโซซิเอชันกับ ConcreteCommand ถือว่าเป็นแบบรูป Command เช่นกัน โดยถือเป็นแบบรูปที่มีการดัดแปลงจากมาตรฐาน และกรณีที่คลาสที่มีความสัมพันธ์กับ Receiver นั้นไม่มีความสัมพันธ์กับ ConcreteCommamd จะถือเป็นแบบรูปที่มีการดัดแปลงจากมาตรฐานกรณีที่ไม่มีคลาสที่มีหน้าที่เป็น Client โดยแบบรูป Command ในลักษณะนี้จะมี x คือ คลาสที่ทำหน้าที่เป็น Command ส่วน y คือ คลาสที่ทำหน้าที่เป็น Invoker อิมพลีเมนต์คลาสของโหนดหลักทั้งหมด คือคลาสที่ทำหน้าที่เป็น ConcreteCommand คลาสที่มีความสัมพันธ์กับ ConcreteCommand คือ คลาสที่ทำหน้าที่เป็น Receiver เท่านั้น

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานอีกกรณีหนึ่งคือ การที่โหนดหลักเป็นคลาสอินเตอร์เฟสแต่แทนที่จะมีความสัมพันธ์แบบรีไลเซชันกลับมีความสัมพันธ์แบบเจนเนอรัลไลเซชัน ในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบคลาสสุดท้ายที่มีการสืบทอดว่าเป็นอิมพลีเมนต์คลาสหรือไม่ ถ้าใช่ก็จะทำการตรวจสอบคลาสนี้ในลักษณะของการตรวจสอบหา ConcreteCommand

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานกรณีสุดท้ายคือ การที่โหนดหลักเป็นคลาสนามธรรมแทนที่จะตรวจสอบหาความสัมพันธ์แบบรีไลเซชัน จะต้องทำการตรวจสอบหาความสัมพันธ์แบบเจนเนอรัลไลเซชันแทนโดยในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบความสัมพันธ์แบบอะกรีเกชันของโหนดหลักและคลาสสุดท้ายของการสืบทอด ถ้าไม่มีความสัมพันธ์ความสัมพันธ์แบบอะกรีเกชันกันจะทำการตรวจสอบคลาสสุดท้ายนี้ในลักษณะของการตรวจสอบหา ConcreteCommand

3.1.2.2 ขั้นตอนวิธีในการตรวจจับแบบรูป Command

A1: The "Anchor Node" finder Algorithm for Command Pattern

For Each Class ***C*** in Class Diagram ***CD***

If Class Type of ***C*** equals "Interface" or "Abstract" then

For Each Relation ***R*** of ***C***

If Type of ***R*** equals "AggregationC" then

For Each Relation ***R2*** of ***C***

If Type of ***R2*** is not equals "SelfAssociation"

Append (***x,y***) to ***AnchorCom*** where x is an interface class and y is related class with x

ภาพที่ 3.10 ขั้นตอนวิธี A1 ในการตรวจจับแบบรูปการออกแบบ Command

A2: The "Design Pattern" finder Algorithm for Command Pattern

For Each (***x,y***) in ***AnchorCom***

Read File for Getting Data ***attr*** of ***y***

Append ***attr*** to ***YAttr***

If Class Type of ***YAttr*** equals "Concrete" then

For Each Relation ***R*** of ***x***

If Type of ***R*** equals "GeneralizationP" then

Call Perform Function ***GetCascadingInheritance(x)***

Append ***iClass*** to ***TempInherit*** where iClass contains all of the Cascading Inheritance classes returned from Function GetCascadingInheritance()

Read File for Getting Data ***D*** of Last Index of ***TempInherit***

If Class Type of ***x*** equals "Interfece" then

For Each Relation ***R2*** of ***D***

If Type of ***R2*** equals "ImplementationP" then

Read File for Getting Data ***D1*** of Related Class ***Rt***

ภาพที่ 3.11 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Command

If Class Type of ***Rt*** equals "Concrete" then

Append ***Rt*** to ***TempImp***

Else If Class Type of ***x*** is equals "Abstract"

Append ***D*** to ***TempImp***

If Type of ***R*** equals "ImplementationP" then

Append ***Rt*** to ***TempImp*** where Rt is implementation class of x

For Each Class ***C*** in ***TempImp***

For Each Relation ***R*** of ***C***

If Type of ***R*** equals "AssociationC" and Related Class is not ***x*** then

Read File for Getting Data ***D*** of of Related Class ***Rt***

Append ***Rt*** to ***TempCallClass***

Call Perform Function ***CheckCallClass(TempCallClass)*** where Function CheckCallClass return true or false

If ***CheckCallClass*** == true then

Append ***x*** to ***ListTemp***

Append ***y*** to ***ListTemp***

Append ***TempImp*** to ***ListTemp***

Append ***TempInherit*** to ***ListTemp***

Append ***TempCallClass*** to ***ListTemp***

Read File for Getting Data ***D*** of ***TempCallClass***

For Each Relation ***R2*** of ***D***

If Type of ***R2*** equals "AssociationC" and Related Class is not equals ***TempImp*** then

Append ***Rt*** to ***ListTemp***

Append ***ListTemp*** to ***CommandList*** where CommandList contains all of the detected patterns to be reported.

ภาพที่ 3.12 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Command (ต่อ)

3.2.3 แบบรูป Iterator

3.1.3.1 โครงสร้างของแบบรูป Iterator

1) แบบรูปตามมาตรฐาน

a. ประกอบไปด้วยคลาสที่ทำหน้าที่เป็น Iterator ซึ่งมีชนิดของคลาสเป็นคลาสอินเตอร์เฟส (Interface) คลาสที่ทำหน้าที่เป็น ConcreteIterator มีชนิดของคลาสเป็นคลาสรูปธรรม (Concrete) คลาสที่ทำหน้าที่เป็น Aggregate มีชนิดของคลาสเป็นคลาสอินเตอร์เฟส คลาสที่ทำหน้าที่เป็น ConcreteAggregate มีชนิดของคลาสเป็นคลาสรูปธรรม และคลาสที่ทำหน้าที่เป็น Client มีชนิดของคลาสเป็นคลาสรูปธรรม

b. คลาสที่ทำหน้าที่เป็น Iterator ต้องมีอิมพลีเมนต์คลาสอย่างน้อย 1 คลาส คือ ConcreteIterator ต้องไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเอง และไม่มีความสัมพันธ์กับ ConcreteCommand ในรูปแบบอื่นๆนอกเหนือจากรีไลเซชันและเจนเนอรัลไลเซชัน

c. คลาสที่ทำหน้าที่เป็น ConcreteIterator ต้องมีความสัมพันธ์แอสโซซิเอชันและดีเพนเดนซีกับ ConcreteAggregate

d. คลาสที่ทำหน้าที่เป็น ConcreteAggregate จะต้องเป็นอิมพลีเมนต์ของคลาสที่ทำหน้าที่เป็น Aggregate

e. คลาสที่ทำหน้าที่เป็น Aggregate ห้ามมีความสัมพันธ์ใดๆกับ ConcreteIterator

f. คลาสที่ทำหน้าที่เป็น Client ต้องมีความสัมพันธ์แบบแอสโซซิเอชันกับทั้งคลาสที่ทำหน้าที่เป็น ConcreteIterator และคลาสที่ทำหน้าที่เป็น ConcreteAggregate

2) แบบรูปดัดแปลงจากมาตรฐาน

a. คลาสที่ทำหน้าที่เป็น Iterator และ Aggregate สามารถมีชนิดของคลาสเป็นคลาสนามธรรม (Abstract) ได้โดยหากมีชนิดเป็นคลาสนามธรรมจะต้องมีความสัมพันธ์แบบเจนเนอรัลไลเซชันกับ ConcreteIterator และ ConcreteAggregate ตามลำดับ

b. คลาสที่ทำหน้าที่เป็น ConcreteIterator สามารถมีความสัมพันธ์แบบแอสโซซิเอชันกับ ConcreteAggregate อย่างเดียวได้

c. ไม่มีคลาสที่ทำหน้าที่เป็น Client ได้

d. กรณีที่คลาสที่ทำหน้าที่เป็น Iterator และ Aggregate มีชนิดของคลาสเป็นอินเตอร์เฟสคลาสสามารถมีการสืบทอดได้ แต่คลาสที่สืบทอดคลาสสุดท้ายจะต้องมีอิมพลีเมนต์คลาสซึ่งทำหน้าที่เป็น ConcreteIterator และ ConcreteAggregate ตามลำดับ

โดยทั้งแบบรูปตามมาตรฐาน และแบบรูปดัดแปลงจากมาตรฐาน หากอิมพลีเมนต์คลาสของ Iterator (ConcreteIterator) และ Aggregate (ConcreteCommand) มีการสืบทอดจะไม่ถือว่าคลาสที่สืบทอดต่อไปเหล่านั้นเป็นแบบรูปการออกแบบ

จากโครงสร้างของแบบรูปการออกแบบที่กล่าวมาข้างต้น ทำให้กำหนดคุณสมบัติของคลาสที่เป็นโหนดหลักของแบบรูป Iterator ได้ดังต่อไปนี้

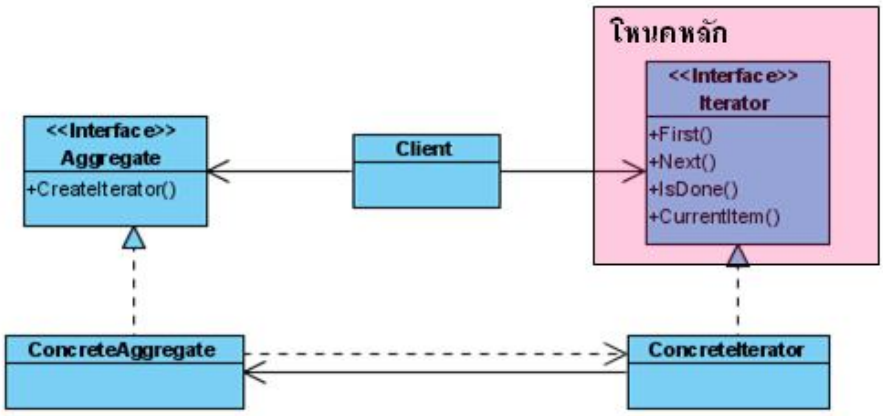
1) เป็นคลาสที่มีชนิดของคลาสได้ใน 2 รูปแบบคือ คลาสอินเตอร์เฟสหรือคลาสนามธรรม

2) เป็นคลาสที่มีความสัมพันธ์แบบรีไลเซชันหรือเจนเนอรัลไลเซชัน

3) เป็นคลาสที่ไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือกับอิมพลีเมนต์คลาส

4) เป็นคลาสที่อิมพลีเมนต์คลาสมีความสัมพันธ์แบบแอสโซซิเอชันโดยมีหน้าที่เรียกใช้

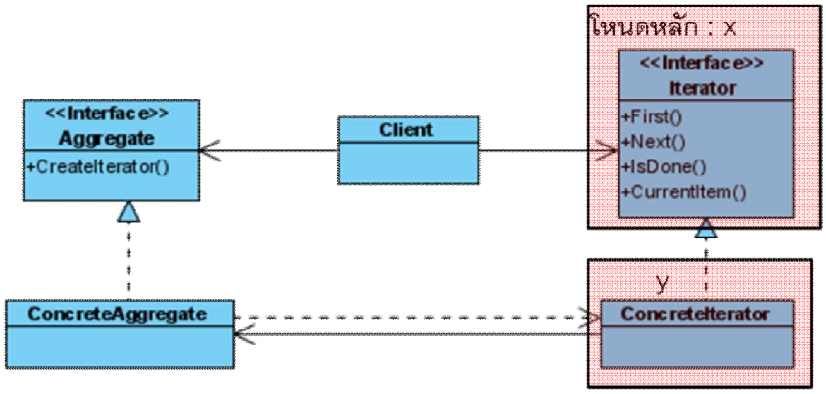
จากคุณสมบัติดังกล่าวจะเห็นได้ว่า โหนดหลัก คือ คลาสที่ทำหน้าที่เป็น Iterator นั่นเอง



ภาพที่ 3.13 ตัวอย่างโหนดหลักของแบบรูปการออกแบบ Iterator

สามารถอธิบายขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบได้ดังต่อไปนี้

ตามขั้นตอนวิธี A1 ซึ่งแสดงในภาพที่ 3.15 ในหัวข้อ 3.1.3.2 เป็นขั้นตอนวิธีในการค้นหาโหนดหลักของแบบรูปการออกแบบ Command โดยเริ่มต้นการทำงานด้วยการค้นหาคลาสที่ทำหน้าที่เป็น Iterator ซึ่งเป็นคลาสที่มีชนิดของคลาสเป็นคลาสอินเตอร์เฟซหรือคลาสนามธรรม เมื่อได้คลาสที่มีชนิดของคลาสดังกล่าวแล้วจะทำการตรวจสอบว่า คลาสเหล่านี้มีคลาสใดบ้างที่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองโดยหากพบจะไม่ทำการพิจารณาคลาสนั้นต่อ หากคลาสใดไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองจะทำการตรวจสอบหาความสัมพันธ์แบบรีไลเซชันหรือเจนเนอรัลไลเซชันกับคลาสอื่นๆ จากนั้นจะทำการตรวจสอบคลาสที่มีความสัมพันธ์แบบรีไลเซชันนี้ว่ามีความสัมพันธ์แบบแอสโซซิเอชัน โดยต้องเป็นกรณีที่ไปเรียกใช้หรือไม่ กรณีที่มีจะทำการเก็บคลาสนี้เป็นโหนดหลักและเก็บข้อมูลของคลาสที่มีความสัมพันธ์ไว้ด้วย โดยเก็บไว้เป็นคู่ลำดับ (x,y) ซึ่ง x จะแทนคลาสที่เป็นโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Iterator และ y แทนคลาสที่มีความสัมพันธ์แบบรีไลเซชันกับโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น ConcreteIterator



ภาพที่ 3.14 ตัวอย่างผลลัพธ์จากขั้นตอนวิธี A1ของแบบรูปการออกแบบ Iterator

ตามขั้นตอนวิธี A2 ซึ่งแสดงในภาพที่ 3.16 และ 3.17 ในหัวข้อ 3.1.3.2 หลังจากได้ชุดของโหนดหลัก (คู่ลำดับ x และ y) แล้ว จะนำชุดข้อมูล (x, y) นี้ไปทำการตรวจสอบเพื่อทำการค้นหารายละเอียดของแบบรูปการออกแบบโดยจะทำการตรวจสอบข้อมูลของ y ก่อน โดยจะทำการตรวจสอบว่า y มีความสัมพันธ์กับ x นอกเหนือความสัมพันธ์แบบรีไลเซชันหรือเจนเนอรัลไลซันหรือไม่ ซึ่งหากใช่จะทำการหยุดตรวจสอบชุดข้อมูลนี้ทันทีเนื่องจากไม่เป็นไปตามโครงสร้างของแบบรูปการออกแบบ Itearator แต่ถ้าไม่ใช่จะทำการตรวจสอบในขั้นต่อไป

ขั้นตอนถัดไปเป็นขั้นตอนในการตรวจสอบหา ConcreteAggregate โดยจะทำการตรวจสอบ y ว่ามีความสัมพันธ์แบบแอสโซซิเอชันในลักษณะเรียกใช้กับคลาสใดบ้าง หากมีจะทำการกำหนดคลาสที่ y มีความสัมพันธ์ด้วยว่าอาจเป็น ConcreteAggregate จากนั้นจะทำการตรวจสอบชนิดของคลาส ConcreteAggregate ว่ามีชนิดของคลาสเป็นคลาสรูปธรรมใช่หรือไม่ หากใช่ จะทำการตรวจสอบความสัมพันธ์ของ ConcreteAggregate นี้ว่าเป็นอิมพลีเมนต์คลาสของคลาสใด และกำหนดคลาสที่ ConcreteAggregate เป็นอิมพลีเมนต์คลาสว่าอาจเป็นคลาสที่เป็น Aggregate แล้วทำการตรวจสอบคลาส Aggregate ว่ามีความสัมพันธ์กับ y หรือไม่ ถ้าไม่มีจะตรวจสอบต่อว่ามีคลาสใดมามีความสัมพันธ์แบบแอสโซซิเอชันกับ Aggregate บ้างหากมีจะกำหนดคลาสที่มีความสัมพันธ์กับ Aggregate ว่าเป็น Client และนำ Client มาทำการตรวจสอบว่า Client มีความสัมพันธ์แบบแอสโซซิเอชันกับโหนดหลัก (x) หรือไม่ หากมีจะทำให้ได้ชุดของคลาสที่เป็นแบบรูปการออกแบบ Itarator โดย x คือ คลาสที่ทำหน้าที่เป็น Itearator ส่วน y คือ คลาสที่ทำหน้าที่เป็น ConcreteIterator คลาสที่มีความสัมพันธ์กับ ConcreteIterator โดยถูก ConcreteIterator เรียกใช้ คือคลาสที่ทำหน้าที่เป็น ConcreteAggragate คลาสที่

ConcreteAggragate เป็นอิมพลีเมนต์คลาส คือคลาสที่ทำหน้าที่เป็น Aggregate และคลาสที่มีความสัมพันธ์กับ Aggregate และ Iterator คือ คลาสที่ทำหน้าที่เป็น Client โดยการตรวจสอบแบบนี้จะเป็นการตรวจสอบแบบรูปแบบมาตรฐาน กรณีที่ไม่มีคลาสที่มีความสัมพันธ์กับ Aggregate และ Iterator หรือไม่มีคลาสที่ทำหน้าที่เป็น Client ก็ยังคงถือว่าเป็นแบบรูป Command เช่นกัน โดยจะเป็นแบบรูปที่มีการดัดแปลงจากมาตรฐาน โดยแบบรูป Ierator ในลักษณะนี้จะมีเพียงคลาสที่ทำหน้าที่เป็น Itearator คลาสที่ทำหน้าที่เป็น ConcreteIterator คลาสที่ทำหน้าที่เป็น ConcreteAggragate และคลาสที่ทำหน้าที่เป็น Aggregate เท่านั้น

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานอีกกรณีหนึ่งคือ การที่โหนดหลักและคลาสที่ทำหน้าที่เป็น Aggregate เป็นคลาสอินเตอร์เฟสแต่แทนที่จะมีความสัมพันธ์แบบรีไลเซชันกลับมีความสัมพันธ์แบบเจนเนอรัลไลเซชัน ในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบคลาสสุดท้ายที่มีการสืบทอดว่าเป็นอิมพลีเมนต์คลาสหรือไม่ ถ้าใช่ก็จะทำการตรวจสอบคลาสนี้ในลักษณะของการตรวจสอบคลาส y และทำการตรวจสอบขั้นอื่นๆต่อไปตามที่ได้กล่าวมาแล้ว

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานกรณีสุดท้ายคือ การที่โหนดหลักและคลาสที่ทำหน้าที่เป็น Aggregate เป็นคลาสนามธรรม โดยหากเป็นกรณีนี้จะทำการตรวจสอบหาความสัมพันธ์แบบเจนเนอรัลไลเซชันแทน และจะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบความสัมพันธ์แบบอะกรีเกชันของโหนดหลักและคลาสสุดท้ายของการสืบทอด ถ้าไม่มีความสัมพันธ์ความสัมพันธ์แบบอะกรีเกชันกันจะทำการตรวจสอบคลาสสุดท้ายนี้ในลักษณะของการตรวจสอบคลาส y และทำการตรวจสอบขั้นอื่นๆต่อไปตามที่ได้กล่าวมาแล้ว

3.1.3.2 ขั้นตอนวิธีในการตรวจจับแบบรูป Iterator

A1: The "Anchor Node" finder Algorithm for Iterator Pattern

For Each Class ***C*** in Class Diagram ***CD***

  If Class Type of ***C*** equals "Interface" or "Abstract" then

    For Each Relation ***R*** of ***C***

      If Type of ***R*** equals "ImplementationP" or "GeneralizationP" and then

        For Each Relation ***R2*** of ***C***

          If Type of ***R2*** is not equals "SelfAssociation"

            Append ***(x,y)*** to ***AnchorIt*** where x is an interface class and y is related class with x

ภาพที่ 3.15 ขั้นตอนวิธี A1 ในการตรวจจับแบบรูปการออกแบบ Iterator

A2: The "Design Pattern" finder Algorithm for Iterator Pattern

For Each (***x,y***) in ***AnchorIt***

For Each Relation ***R*** of ***x***

If Type of ***R*** equals "GenralizationP" and Related Class is ***y*** then

Call Perform Function ***GetCascadingInheritance(x)***

Append ***iClass*** to ***TempInherit*** where iClass contains all of the Cascading Inheritance classes returned from Function GetCascadingInheritance()

Read File for Getting Data ***D*** of Last Index of ***TempInherit***

If Class Type of ***x*** equals "Interface" then

For Each Relation ***R2*** of ***D***

If Type of ***R2*** equals "ImplementationP" then

Read File for Getting Data ***D1*** of Related Class ***Rt***

For Each Relation ***R3*** of ***Rt***

If Type of ***R3*** equals "AssociationC" then

Append ***Rt*** to ***TempImp***

Else If Class Type of ***x*** equals "Abstract" then

If Type of ***R2*** equals "AssociationC" then

Append ***D*** to ***TempImp***

If Type of ***R*** equals "ImplementationP" and Related Class is ***y*** then

Append ***Rt*** to ***TempImp*** where Rt is implementation class of x

For Each Class ***C*** in ***TempImp***

For Each Relation ***R*** of ***C***

If Type of ***R*** equals "AssociationC" then

Read File for Getting Data ***D*** of Related Class ***Rt***

Append ***Rt*** to ***TempCallClass***

ภาพที่ 3.16 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Iterator

For Each Class ***C*** in ***TempCallClass***

For Each Relation ***R*** of ***C***

If Type of ***R*** equals "ImplementationC" then

Read File for Getting Data ***D*** of Related Class ***Rt***

If Class Type of ***Rt*** equals "Interface" then

Append ***x*** to ***ListTemp***

Append ***y*** to ***ListTemp***

Append ***TempCallClass*** to ***ListTemp***

Append ***TempInherit*** to ***ListTemp***

Append ***Rt*** to ***ListTemp***

Append ***Rt*** to ***CheckCall***

If Type of ***R*** equals "GeneralizationC" then

Read File for Getting Data ***D*** of Related Class ***Rt***

Call Perform Function ***GetParentCascadingInheritance(Rt)***

Append ***iClass*** to ***TempInheritP*** where iClass contains all of the Parent classes of Genaeraliztion returned from Function GetParentCascadingInheritance()

Append ***x*** to ***ListTemp***

Append ***y*** to ***ListTemp***

Append ***TempCallClass*** to ***ListTemp***

Append ***TempInherit*** to ***ListTemp***

Append ***Rt*** to ***ListTemp***

Append ***TempInheritP*** to ***ListTemp***

Append Last Index of ***TempInheritP*** to ***CheckCall***

If Type of ***R*** equals "AssociationP" and Related Class is not ***y*** then

Read File for Getting Data ***D*** of Related Class ***Rt***

For Each Relation **R2** of ***Rt***

If Type of ***R2*** equals "AssociationC" and Related Class ***Rt2*** is not ***x*** then

If ***Rt2*** equals ***CheckCall***

Append ***Rt2*** to ***ListTemp***

Append ***ListTemp*** to ***IteratorList*** where IteratorList contains all of the detected patterns to be reported

ภาพที่ 3.17 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Iterator (ต่อ)

3.2.4 แบบรูป Memento

3.1.4.1 โครงสร้างของแบบรูป Memento

1) แบบรูปตามมาตรฐาน

   a. ประกอบไปด้วยคลาสที่ทำหน้าที่เป็น Memento คลาสที่ทำหน้าที่เป็น Caretaker และคลาสที่ทำหน้าที่เป็น Originator โดยทั้งหมดมีชนิดของคลาสเป็นคลาสรูปธรรม

   b. คลาสที่ทำหน้าที่เป็น Memento ต้องมีความสัมพันธ์แบบอะกรีเกชันกับคลาสที่ทำหน้าที่เป็น Caretaker และมีความสัมพันธ์แบบดีเพนเดนซีกับคลาสที่ทำหน้าที่เป็น Originator

   c. คลาสที่ทำหน้าที่เป็น Caretaker และ Originator ห้ามมีความสัมพันธ์ใดๆต่อกัน

2) แบบรูปดัดแปลงจากมาตรฐาน

   a. คลาสที่ทำหน้าที่เป็น Memento สามารถมีความสัมพันธ์แบบแอสโซซิเอชันกับ Originator แทนที่จะเป็นความสัมพันธ์แบบดีเพนเดนซีได้

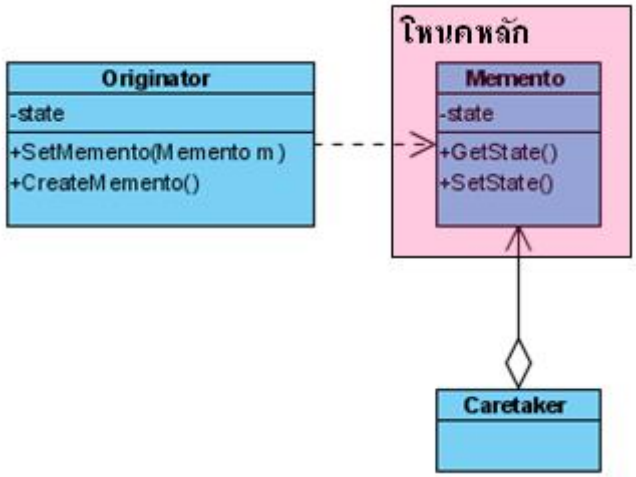
โดยทั้งแบบรูปตามมาตรฐาน และแบบรูปดัดแปลงจากมาตรฐาน หากคลาสที่ทำหน้าที่เป็น Originator และ Caretaker มีการสืบทอดจะไม่ถือว่าคลาสที่สืบทอดต่อไปเหล่านั้นเป็นแบบรูปการออกแบบ

จากโครงสร้างของแบบรูปการออกแบบที่กล่าวมาข้างต้น ทำให้กำหนดคุณสมบัติของคลาสที่เป็นโหนดหลักของแบบรูป Memento ได้ดังต่อไปนี้

1) เป็นคลาสที่มีชนิดของคลาสเป็นคลาสรูปธรรม

2) เป็นคลาสที่มีความสัมพันธ์แบบอะกรีเกชันกับคลาสใดๆ

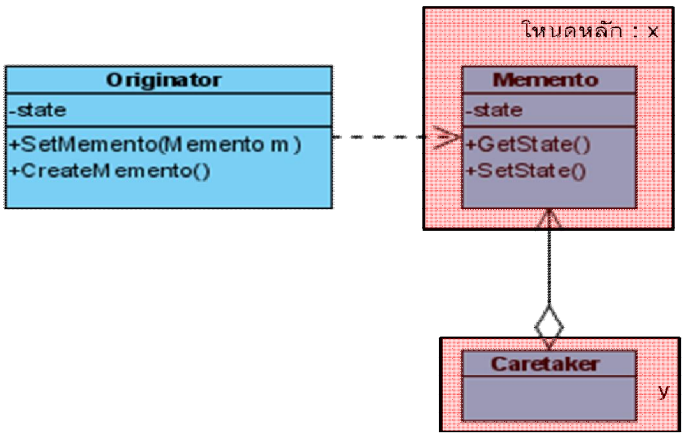
จากคุณสมบัติดังกล่าวจะเห็นได้ว่า โหนดหลัก คือ คลาสที่ทำหน้าที่เป็น Memento นั่นเอง



ภาพที่ 3.18 ตัวอย่างโหนดหลักของแบบรูปการออกแบบ Memento

สามารถอธิบายขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบได้ดังต่อไปนี้

ตามขั้นตอนวิธี A1 ซึ่งแสดงในภาพที่ 3.20 ในหัวข้อ 3.1.4.2 เป็นขั้นตอนวิธีในการค้นหาโหนดหลักของแบบรูปการออกแบบ Memento โดยเริ่มต้นการทำงานด้วยการค้นหาคลาสที่ทำหน้าที่เป็น Memento ซึ่งเป็นคลาสที่มีชนิดของคลาสเป็นคลาสรูปธรรม เมื่อได้คลาสที่มีชนิดของคลาสดังกล่าวแล้วจะทำการตรวจสอบว่า คลาสเหล่านี้มีคลาสใดบ้างที่มีความสัมพันธ์แบบอะกรีเกชันกับคลาสอื่นๆ กรณีที่มีจะทำการเก็บคลาสนี้เป็นโหนดหลักและเก็บข้อมูลของคลาสที่มีความสัมพันธ์ไว้ด้วย โดยเก็บไว้เป็นคู่ลำดับ (x,y) ซึ่ง x จะแทนคลาสที่เป็นโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Memento และ y แทนคลาสที่มีความสัมพันธ์แบบอะกรีเกชันกับโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Caretaker



ภาพที่ 3.19 ตัวอย่างผลลัพธ์จากขั้นตอนวิธี A1ของแบบรูปการออกแบบ Memento

ตามขั้นตอนวิธี A2 ซึ่งแสดงในภาพที่ 3.21ในหัวข้อ 3.1.4.2 หลังจากได้ชุดของโหนดหลัก (คู่ลำดับ x และ y) แล้ว จะนำชุดข้อมูล (x, y) นี้ไปทำการตรวจสอบเพื่อทำการค้นหารายละเอียดของแบบรูปการออกแบบโดยจะทำการตรวจสอบข้อมูลของ y ก่อน โดยจะทำการตรวจสอบว่า y มีชนิดของคลาสเป็นคลาสรูปธรรมหรือไม่ หากถูกต้องจะทำการตรวจสอบความสัมพันธ์ระหว่าง x และ y ว่าเป็นความสัมพันธ์แบบอะกรีเกชันหรือไม่ถ้าหากมีความสัมพันธ์แบบอะกรีเกชันระหว่าง x และ y จากขั้นตอนนี้จะทำให้ได้คลาสที่มีสมบัติเป็น Memento และ Caretaker และทำงานในขั้นตอนถัดไป

ขั้นตอนถัดไปเป็นขั้นตอนในการตรวจสอบหา Originator โดยจะทำการตรวจสอบจากโหนดหลัก หรือ x ว่ามีความสัมพันธ์แบบดีเพนเดนซีกับคลาสใดบ้าง หากมีจะทำการตรวจสอบว่าคลาสที่ได้จากการตรวจสอบนี้มีความสัมพันธ์กับ y หรือไม่ ถ้าไม่มีจะได้แบบรูปการออกแบบ Memento ซึ่ง x คือ คลาสที่ทำหน้าที่เป็น Memento ส่วน y คือ คลาสที่ทำหน้าที่เป็น Caretaker และ คลาสที่มีความสัมพันธ์กับ x ในแบบดีเพนเดนซี คือ คลาสที่ทำหน้าที่เป็น Originator ซึ่งผลลัพธ์ในลักษณะนี้คือ ผลลัพธ์ของแบบรูปมาตรฐาน

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐาน ทำได้โดยทำการตรวจสอบจากโหนดหลักว่ามีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสใด ถ้ามีจะทำการตรวจสอบว่าคลาสที่ได้จากการตรวจสอบนี้มีความสัมพันธ์กับ y หรือไม่ ถ้าไม่มีจะได้แบบรูปการออกแบบ Memento ซึ่ง x คือ คลาสที่ทำหน้าที่เป็น Memento ส่วน y คือ คลาสที่ทำหน้าที่เป็น Caretaker และ คลาสที่มีความสัมพันธ์กับ x ในแบบแอสโซซิเอชัน คือ คลาสที่ทำหน้าที่เป็น Originator

3.1.4.2 ขั้นตอนวิธีในการตรวจจับแบบรูป Memento

```
A1: The "Anchor Node" finder Algorithm for Memento Pattern
For Each Class C in Class Diagram CD
  If Class Type of C equals "Concrete" then
    For Each Relation R of C
          If Type of R equals "DependencyP" or "AssociationP" then
            For Each Relation R2 of C
              If Type of R2 is not equals "SelfAssociation"
                Append (x,y) to AnchorMe where x is concrete class and y is related class with x
```

ภาพที่ 3.20 ขั้นตอนวิธี A1 ในการตรวจจับแบบรูปการออกแบบ Memento

A2: The "Design Pattern" finder Algorithm for Memento Pattern

For Each (*x*,*y*) in ***AnchorMe***

  Read File for Getting Data ***attr*** of ***y***

  Append ***attr*** to ***YAttr***

  If Class Type of ***YAttr*** equals "Concrete" then

    Append ***y*** to ***ListTemp***

    For Each Relation ***R*** of ***x***

      If Type of ***R*** equals "AggregationC" then

          Read File for Getting Data ***D*** of Related Class ***Rt***

          If Class Type of ***Rt*** equals "Concrete" then

              Append ***x*** to ***ListTemp***

              Append ***Rt*** to ***ListTemp***

Append ***ListTemp*** to ***MementoList*** where MementoList contains all of the detected patterns to be reported

ภาพที่ 3.21 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Memento

3.2.5 แบบรูป Observer

3.1.5.1 โครงสร้างของแบบรูป Observer

1) แบบรูปตามมาตรฐาน

a. ประกอบไปด้วยคลาสที่ทำหน้าที่เป็น Observer ซึ่งมีชนิดของคลาสเป็นคลาสอินเตอร์เฟส (Interface) คลาสที่ทำหน้าที่เป็น Subject มีชนิดของคลาสเป็นคลาสอินเตอร์เฟส คลาสที่ทำหน้าที่เป็น ConcreteObserver มีชนิดของคลาสเป็นคลาสรูปธรรม (Concrete) และคลาสที่ทำหน้าที่เป็น ConcreteSubject มีชนิดของคลาสเป็นคลาสรูปธรรม

b. คลาสที่ทำหน้าที่เป็น Observer และ Subject ต้องมีอิมพลีเมนต์คลาสของคลาสตนเองอย่างน้อย 1 คลาส คือ ConcreteObserver และ ConcreteSubject ตามลำดับ โดยต้องไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเอง และไม่มีความสัมพันธ์กับอิมพลีเมนต์คลาสในรูปแบบอื่นๆนอกเหนือจากรีไลเซชันและเจนเนอรัลไลเซชัน

c. คลาสที่ทำหน้าที่เป็น Observer และ Subject ต้องมีความสัมพันธ์กันแบบแอสโซซิเอชัน

d. คลาสที่ทำหน้าที่เป็น ConcreteObserver และ ConcreteSubject ต้องมีความสัมพันธ์กันแบบแอสโซซิเอชัน

2) แบบรูปดัดแปลงจากมาตรฐาน

a. คลาสที่ทำหน้าที่เป็น Observer และ Subject สามารถมีชนิดของคลาสเป็นคลาสนามธรรม (Abstract) ได้โดยหากมีชนิดเป็นคลาสนามธรรมจะต้องมีความสัมพันธ์แบบเจนเนอรัลไลเซชันกับ ConcreteObserver และ ConcreteSubject ตามลำดับ

b. คลาสที่ทำหน้าที่เป็น Subject สามารถเป็นคลาสรูปธรรมได้ โดยหากคลาสที่ทำหน้าที่เป็น Subject เป็นคลาสรูปธรรม จะต้องมีความสัมพันธ์แบบแอสโซซิเอชันกับ ConcreteObserver

c. กรณีที่คลาสที่ทำหน้าที่เป็น Observer และ Subject มีชนิดของคลาสเป็นอินเตอร์เฟสคลาสสามารถมีการสืบทอดได้ แต่คลาสที่สืบทอดคลาสสุดท้ายจะต้องมีอิมพลีเมนต์คลาสซึ่งทำหน้าที่เป็น ConcreteObserver และ ConcreteSubject ตามลำดับ

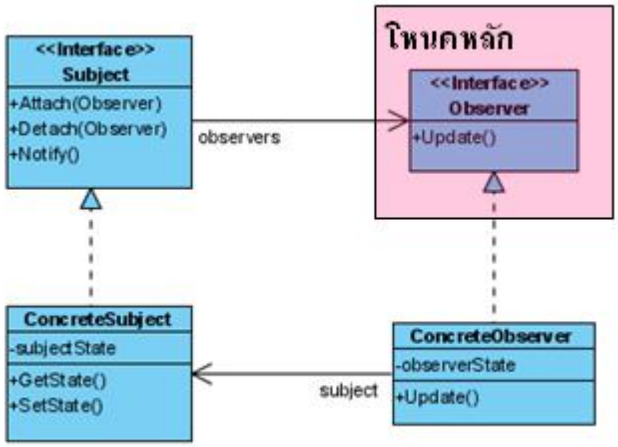
d. คลาสที่ทำหน้าที่เป็น Observer และ Subject โหนดหลักต้องมีอิมพลีเมนต์คลาสอย่างน้อย 1 คลาส คือ มีความสัมพันธ์แบบรีไลเซชันในกรณีที่เป็นคลาสอินเตอร์เฟซหรือมีความสัมพันธ์แบบเจนเนอรัลไลเซชันในกรณีที่เป็นคลาสนามธรรม

โดยทั้งแบบรูปตามมาตรฐาน และแบบรูปดัดแปลงจากมาตรฐาน หากอิมพลีเมนต์คลาสของ Observer (ConcreteObserver) และ Subject (ConcreteSubject) มีการสืบทอดจะไม่ถือว่าคลาสที่สืบทอดต่อไปเหล่านั้นเป็นแบบรูปการออกแบบ

จากโครงสร้างของแบบรูปการออกแบบที่กล่าวมาข้างต้น ทำให้กำหนดคุณสมบัติของคลาสที่เป็นโหนดหลักของแบบรูป Observer ได้ดังต่อไปนี้

1) เป็นคลาสที่มีชนิดของคลาสได้ใน 2 รูปแบบคือ คลาสอินเตอร์เฟสหรือคลาสนามธรรม
2) เป็นคลาสที่ไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือกับอิมพลีเมนต์คลาสเป็นคลาสที่มีชนิดของคลาสเป็นคลาสอินเตอร์เฟซหรือคลาสนามธรรม
3) เป็นคลาสที่มีความสัมพันธ์แบบอะกรีเกชันกับคลาสรูปธรรมใดๆ

จากคุณสมบัติดังกล่าวจะเห็นได้ว่า โหนดหลัก คือ คลาสที่ทำหน้าที่เป็น Observer นั่นเอง

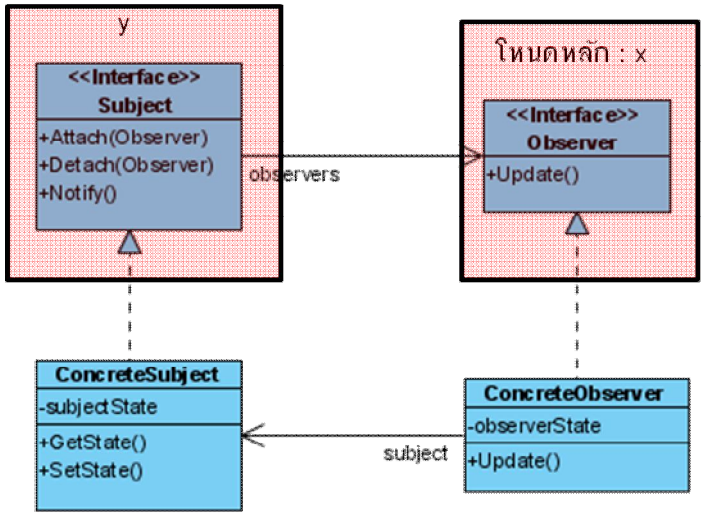


ภาพที่ 3.22 ตัวอย่างโหนดหลักของแบบรูปการออกแบบ Observer

สามารถอธิบายขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบได้ดังต่อไปนี้

ตามขั้นตอนวิธี A1 ซึ่งแสดงในภาพที่ 3.24 ในหัวข้อ 3.1.5.2 เป็นขั้นตอนวิธีในการค้นหาโหนดหลักของแบบรูปการออกแบบ Observer โดยเริ่มต้นการทำงานด้วยการค้นหาคลาสที่ทำ

หน้าที่เป็น Observer ซึ่งเป็นคลาสที่มีชนิดของคลาสเป็นคลาสอินเตอร์เฟซหรือคลาสนามธรรม เมื่อได้คลาสที่มีชนิดของคลาสดังกล่าวแล้วจะทำการตรวจสอบว่า คลาสเหล่านี้มีคลาสใดบ้างที่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองโดยหากพบจะไม่ทำการพิจารณาคลาสนั้นต่อ หากคลาสใดไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองจะทำการตรวจสอบหาความสัมพันธ์แบบแอสโซซิเอชันกับคลาสอื่นๆ กรณีที่มีจะทำการเก็บคลาสนี้เป็นโหนดหลักและเก็บข้อมูลของคลาสที่มีความสัมพันธ์ไว้ด้วย โดยเก็บไว้เป็นคู่ลำดับ (x,y) ซึ่ง x จะแทนคลาสที่เป็นโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Observer และ y แทนคลาสที่มีความสัมพันธ์แบบแอสโซซิเอชันกับโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Subject



ภาพที่ 3.23 ตัวอย่างผลลัพธ์จากขั้นตอนวิธี A1ของแบบรูปการออกแบบ Observer

ตามขั้นตอนวิธี A2 ซึ่งแสดงในภาพที่ 3.25 3.26 และ 3.27 ในหัวข้อ 3.1.5.2 หลังจากได้ชุดของโหนดหลัก (คู่ลำดับ x และ y) แล้ว จะนำชุดข้อมูล (x, y) นี้ไปทำการตรวจสอบเพื่อทำการค้นหารายละเอียดของแบบรูปการออกแบบโดยจะทำการตรวจสอบข้อมูลของ y ก่อน โดยจะทำการตรวจสอบว่า y มีชนิดของคลาสเป็นคลาสชนิดใด หาก y มีชนิดของคลาสเป็นคลาสอินเตอร์เฟสจะทำการตรวจสอบหาอิมพลีเมนต์คลาสของโหนดหลักหรือหา ConcreteObserver โดยทำการตรวจสอบหาคลาสที่มีความสัมพันธ์รีไลเซชันกับโหนดหลักหากมีจะทำการตรวจสอบว่าคลาสนี้มีความสัมพันธ์กับโหนดหลักนอกเหนือจากความสัมพันธ์แบบรีไลเซชันหรือไม่ถ้าไม่มีจะ

ตรวจสอบว่าอิมพลีเมนต์คลาสเหล่านี้มีความสัมพันธ์กับคลาสอื่นๆในลักษณะความสัมพันธ์แบบแอสโซซิเอชัน โดยต้องเป็นแอสโซซิเอชันในลักษณะที่เรียกใช้คลาสอื่นๆ ถ้ามีคลาสนี้จะถูกกำหนดว่าอาจมีความเป็นไปได้ที่จะเป็น ConcreteObserver และคลาสที่ถูก ConcreteObserver เรียกใช้จะถูกกำหนดว่ามีความเป็นไปได้ที่จะเป็น ConcreteSubject จากนั้นจะทำการตรวจสอบ ConcreteSubject ว่าเป็นอิมพลีเมนต์คลาสของคลาสอื่นๆหรือไม่ ถ้าใช่จะการตรวจสอบว่าเป็น y ใช่หรือไม่หรือไม่ ซึ่งถ้าใช่จะทำให้ได้ชุดของคลาสที่เป็นแบบรูปการออกแบบ Observer โดย x คือ คลาสที่ทำหน้าที่เป็น Observer ส่วน y คือ คลาสที่ทำหน้าที่เป็น Subject อิมพลีเมนต์คลาสของโหนดหลักทั้งหมด คือคลาสที่ทำหน้าที่เป็น ConcreteObserver คลาสที่มีความสัมพันธ์กับ ConcreteObserver และ y คือ คลาสที่ทำหน้าที่เป็น ConcreteSubject โดยการตรวจสอบแบบนี้จะเป็นการตรวจสอบแบบรูปแบบมาตรฐาน

กรณีที่ y มีชนิดของคลาสเป็นคลาสรูปธรรม จะทำการตรวจสอบหาอิมพลีเมนต์คลาสของโหนดหลักหรือหา ConcreteObserver โดยทำการตรวจสอบหาคลาสที่มีความสัมพันธ์รีไลเซชันกับโหนดหลักหากมีจะทำการตรวจสอบว่าคลาสนี้มีความสัมพันธ์กับโหนดหลักนอกเหนือจากความสัมพันธ์แบบรีไลเซชันหรือไม่ถ้าไม่มีจะตรวจสอบว่าอิมพลีเมนต์คลาสเหล่านี้มีความสัมพันธ์กับ y ในลักษณะความสัมพันธ์แบบแอสโซซิเอชัน โดยต้องเป็นแอสโซซิเอชันในลักษณะที่เรียกใช้หรือไม่ ถ้ามีจะทำให้ได้ชุดของคลาสที่เป็นแบบรูปการออกแบบ Observer โดย x คือ คลาสที่ทำหน้าที่เป็น Observer ส่วน y คือ คลาสที่ทำหน้าที่เป็น Subject อิมพลีเมนต์คลาสของโหนดหลักทั้งหมที่มีความสัมพันธ์กับ y คือคลาสที่ทำหน้าที่เป็น ConcreteObserver โดยการตรวจสอบแบบนี้จะเป็นการตรวจสอบแบบรูปที่ดัดแปลงจากมาตรฐาน

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานอีกกรณีหนึ่งคือ การที่โหนดหลัก (x) เป็นคลาสอินเตอร์เฟสแต่แทนที่จะมีความสัมพันธ์แบบรีไลเซชันกลับมีความสัมพันธ์แบบเจนเนอรัลไลเซชัน ในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบคลาสสุดท้ายที่มีการสืบทอดว่าเป็นอิมพลีเมนต์คลาสหรือไม่ ถ้าใช่ก็จะทำการตรวจสอบคลาสนี้ในลักษณะของการตรวจสอบหา ConcreteObserver และทำการตรวจสอบคลาสอื่นๆดังที่ได้กล่าวมาแล้ว

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานกรณีสุดท้ายคือ การที่โหนดหลัก (x) และ y เป็นคลาสนามธรรมแทนที่จะตรวจสอบหาความสัมพันธ์แบบรีไลเซชัน จะต้องทำการตรวจสอบหาความสัมพันธ์แบบเจนเนอรัลไลเซชันแทนโดยในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบความสัมพันธ์แบบอะกรีเกชันของโหนดหลักและคลาสสุดท้าย

ของการสืบทอด ถ้าไม่มีความสัมพันธ์ความสัมพันธ์แบบอะกรีเกชันกันจะทำการตรวจสอบคลาสสุดท้ายนี้ในลักษณะของการตรวจสอบหา ConcreteObserver และทำการตรวจสอบคลาสอื่นๆ ดังที่ได้กล่าวมาแล้ว

3.1.5.2 ขั้นตอนวิธีในการตรวจจับแบบรูป Observer

```
A1: The "Anchor Node" finder Algorithm for Observer Pattern
For Each Class C in Class Diagram CD
  If Class Type of C equals "Interface" or "Abstract" then
    For Each Relation R of C
        If Type of R equals "AssociationP" then
          For Each Relation R2 of C
            If Type of R is not equals "SelfAssociation"
              Append (x,y) to AnchorOb where x is an interface class and y is related class with x
```

ภาพที่ 3.24 ขั้นตอนวิธี A1 ในการตรวจจับแบบรูปการออกแบบ Observer

```
A2: The "Design Pattern" finder Algorithm for Observer Pattern
For Each (x,y) in AnchorOb
  Read File for Getting Data attr of y
  Append attr to YAttr
  If Class Type of YAttr equals "Interface" then
    For Each Relation R of y
      If Type of R equals "AssociationP" and Related Class is x then
        For Each Relation R2 of y
          If Type of R2 equals "ImplementationP" then
            Append y to ListTemp
            Append Rt to CheckCallClass where Rt is implementation class of x
          If Type of R equals "GeneralizationP" then
              Call Perform Function GetCascadingInheritance(x)
                Append iClass to TempInherit where iClass contains all of the Cascading
Inheritance classes returned from Function GetCascadingInheritance()
          Read File for Getting Data D of Last Index of TempInherit
              For Each Relation R3 of D
                If Type of R3 equals "ImplementationP" then
```

ภาพที่ 3.25 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Observer

Append ***y*** to ***ListTemp***

Append ***Rt*** to ***CheckCallClass*** where Rt is implementation class of D

Else If Class Type of ***YAttr*** equals "Abstract" then

For Each Relation ***R*** of ***y***

If Type of ***R*** equals "AssociationP" and Related Class is ***x*** then

For Each Relation ***R2*** of ***y***

If Type of ***R2*** equals "GeneralizationP" then

Append ***y*** to ***ListTemp***

Call Perform Function ***GetCascadingInheritance(x)***

Append ***iClass*** to ***TempInherit*** where iClass contains all of the Cascading Inheritance classes returned from Function GetCascadingInheritance()

Read File for Getting Data ***D*** of Last Index of TempInherit2 ***LastD***

Append ***LastD*** to ***CheckCallClass***

For Each Relation ***R*** of ***x***

If Type of ***R*** equals "GeneralizationP" then

Call Perform Function ***GetCascadingInheritance(x)***

Append ***iClass*** to ***TempInherit2*** where iClass contains all of the Cascading Inheritance classes returned from Function GetCascadingInheritance()

Read File for Getting Data ***D*** of Last Index of ***TempInherit***

If Class Type of ***x*** is "Interface" then

For Each Relation ***R2*** of ***D***

If Type of ***R2*** equals "ImplementationP" then

Read File for Getting Data ***D1*** of Related Class ***Rt***

If Class Type of ***Rt*** equals "Concrete" then

Append ***x*** to ***ListTemp***

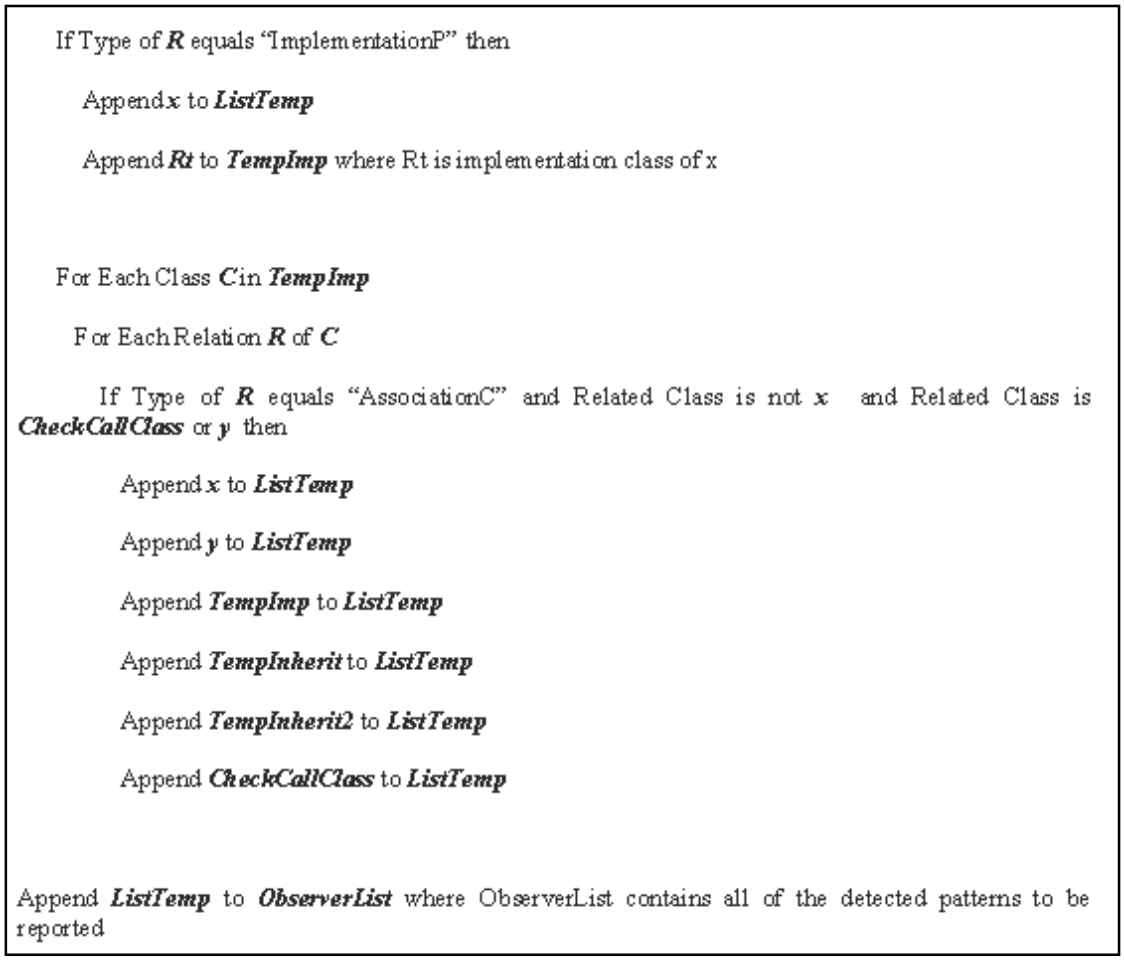
Append ***Rt*** to ***TempImp***

Else If Class Type of ***x*** is "Abstract"

Append ***x*** to ***ListTemp***

Append ***D*** to ***TempImp***

ภาพที่ 3.26 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Observer (ต่อ)



ภาพที่ 3.27 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Observer (ต่อ)

3.2.6 แบบรูป Strategy

3.1.6.1 โครงสร้างของแบบรูป Strategy

1) แบบรูปตามมาตรฐาน

a. ประกอบไปด้วยคลาสที่ทำหน้าที่เป็น Strategy ซึ่งมีชนิดของคลาสเป็นคลาสอินเตอร์เฟส (Interface) คลาสที่ทำหน้าที่เป็น ConcreteStrategy มีชนิดของคลาสเป็นเป็นคลาสรูปธรรม (Concrete) และคลาสที่ทำหน้าที่เป็น Context มีชนิดของคลาสเป็นคลาสรูปธรรม

b. คลาสที่ทำหน้าที่เป็น Strategy ต้องมีอิมพลีเมนต์คลาสของคลาสตนเองอย่างน้อย 1 คลาส คือ ConcreteStrategy โดยต้องไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเอง และไม่มีความสัมพันธ์กับอิมพลีเมนต์คลาสในรูปแบบอื่นๆนอกเหนือจากรีไลเซชันและเจนเนอรัลไลเซชัน

c. คลาสที่ทำหน้าที่เป็น Strategy และ Context ต้องมีความสัมพันธ์กันแบบอะกรีเกชัน

d. คลาสที่ทำหน้าที่เป็น ConcreteStrategy และ Context ต้องไม่มีความสัมพันธ์ใดๆต่อกัน

2) แบบรูปดัดแปลงจากมาตรฐาน

a. คลาสที่ทำหน้าที่เป็น Strategy สามารถมีชนิดของคลาสเป็นคลาสนามธรรม (Abstract) ได้โดยหากมีชนิดเป็นคลาสนามธรรมจะต้องมีความสัมพันธ์แบบเจนเนอรัลไลเซชันกับ ConcreteStrategy

b. กรณีที่คลาสที่ทำหน้าที่เป็น Strategy มีชนิดของคลาสเป็นอินเตอร์เฟสคลาสสามารถมีการสืบทอดได้ แต่คลาสที่สืบทอดคลาสสุดท้ายจะต้องมีอิมพลีเมนต์คลาสซึ่งทำหน้าที่เป็น ConcreteStrategy

c. คลาสที่ทำหน้าที่เป็น Strategy และ Context สามารถมีความสัมพันธ์กันแบบแอสโซซิเอชันได้

โดยทั้งแบบรูปตามมาตรฐาน และแบบรูปดัดแปลงจากมาตรฐาน หากอิมพลีเมนต์คลาสของ Strategy (ConcreteStrategy) มีการสืบทอดจะไม่ถือว่าคลาสที่สืบทอดต่อไปเหล่านั้นเป็นแบบรูปการออกแบบ

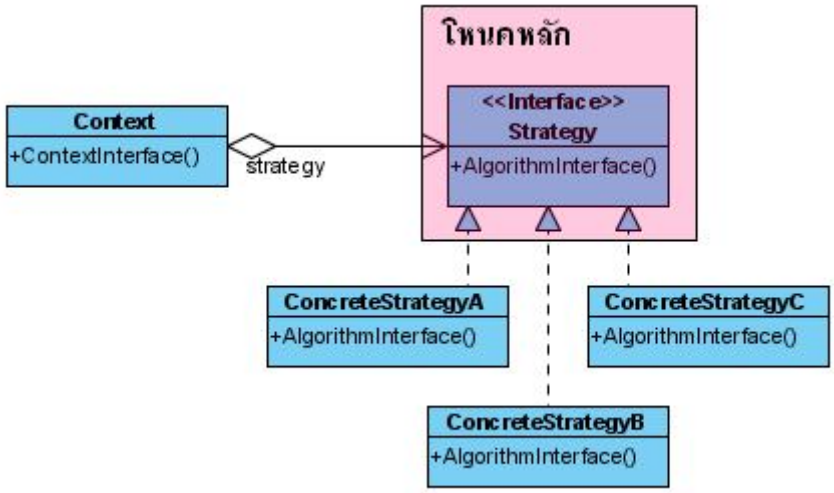
จากโครงสร้างของแบบรูปการออกแบบที่กล่าวมาข้างต้น ทำให้กำหนดคุณสมบัติของคลาสที่เป็นโหนดหลักของแบบรูป Strategy ได้ดังต่อไปนี้

1) เป็นคลาสที่มีชนิดของคลาสได้ใน 2 รูปแบบคือ คลาสอินเตอร์เฟสหรือคลาสนามธรรม

2) เป็นคลาสที่ไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือกับอิมพลีเมนต์คลาสเป็นคลาสที่มีชนิดของคลาสเป็นคลาสอินเตอร์เฟซหรือคลาสนามธรรม

3) เป็นคลาสที่มีความสัมพันธ์แบบอะกรีเกชันหรือแอสโซซิเอชันกับคลาสรูปธรรมใดๆ

จากคุณสมบัติดังกล่าวจะเห็นได้ว่า โหนดหลัก คือ คลาสที่ทำหน้าที่เป็น Strategy นั่นเอง

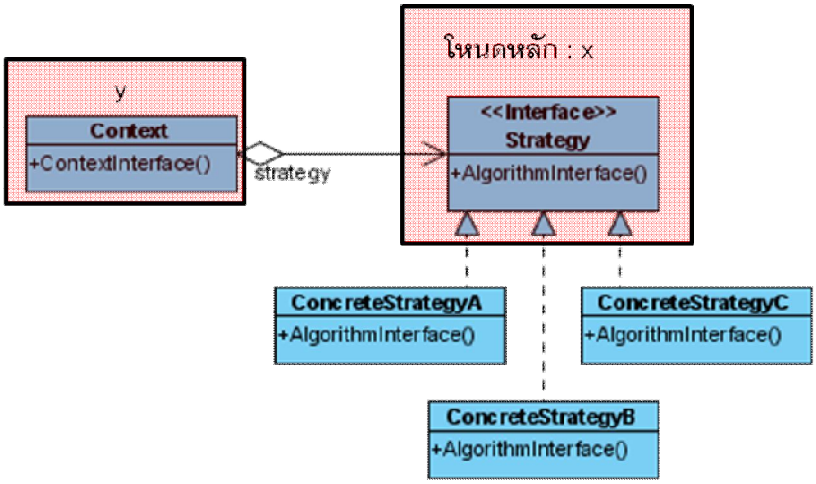


ภาพที่ 3.28 ตัวอย่างโหนดหลักของแบบรูปการออกแบบ Strategy

สามารถอธิบายขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบได้ดังต่อไปนี้

ตามขั้นตอนวิธี A1 ซึ่งแสดงในภาพที่ 3.30 ในหัวข้อ 3.1.6.2 เป็นขั้นตอนวิธีในการค้นหาโหนดหลักของแบบรูปการออกแบบ Strategy โดยเริ่มต้นการทำงานด้วยการค้นหาคลาสที่ทำหน้าที่เป็น Strategy ซึ่งเป็นคลาสที่มีชนิดของคลาสเป็นคลาสอินเตอร์เฟซหรือคลาสนามธรรม เมื่อได้คลาสที่มีชนิดของคลาสดังกล่าวแล้วจะทำการตรวจสอบว่า คลาสเหล่านี้มีคลาสใดบ้างที่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองโดยหากพบจะไม่ทำการพิจารณาคลาสนั้นต่อ หากคลาสใดไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองจะทำการตรวจสอบหาความสัมพันธ์แบบอะกรีเกชันหรือแอสโซซิเอชันกับคลาสอื่นๆ กรณีที่มีจะทำการเก็บคลาสนี้เป็น

โหนดหลักและเก็บข้อมูลของคลาสที่มีความสัมพันธ์ไว้ด้วย โดยเก็บไว้เป็นคู่ลำดับ (x,y) ซึ่ง x จะแทนคลาสที่เป็นโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Strategy และ y แทนคลาสที่มีความสัมพันธ์แบบอะกรีเกชันหรือแอสโซซิเอชันกับโหนดหลักหรือคลาสที่มีความน่าจะเป็นว่าจะทำหน้าที่เป็น Context



ภาพที่ 3.29 ตัวอย่างผลลัพธ์จากขั้นตอนวิธี A1ของแบบรูปการออกแบบ Strategy

ตามขั้นตอนวิธี A2 ซึ่งแสดงในภาพที่ 3.31 และ 3.32 ในหัวข้อ 3.1.6.2 หลังจากได้ชุดของโหนดหลัก (คู่ลำดับ x และ y) แล้ว จะนำชุดข้อมูล (x, y) นี้ไปทำการตรวจสอบเพื่อทำการค้นหารายละเอียดของแบบรูปการออกแบบโดยจะทำการตรวจสอบข้อมูลของ y ก่อน โดยจะทำการตรวจสอบว่า y มีชนิดของคลาสเป็นคลาสรูปธรรมจะทำการตรวจสอบขั้นต่อไป แต่หากไม่ใช่จะไม่ทำการตรวจสอบคู่ลำดับนี้แล้ว เนื่องจากไม่ถูกต้องตามโครงสร้างของแบบรูปการออกแบบ Strategy

การตรวจสอบขั้นตอนต่อมา คือ การตรวจสอบหาอิมพลีเมนต์คลาสของโหนดหลักหรือหา ConcreteStrategy โดยทำการตรวจสอบหาคลาสที่มีความสัมพันธ์รีไลเซชันกับโหนดหลักหากมีจะทำการตรวจสอบว่าคลาสนี้มีความสัมพันธ์กับโหนดหลักนอกเหนือจากความสัมพันธ์แบบรีไลเซชันหรือไม่ถ้าไม่มีจะตรวจสอบว่าอิมพลีเมนต์คลาสเหล่านี้ว่ามีความสัมพันธ์กับ y หรือไม่ ถ้าไม่มีจะทำให้ได้ชุดของคลาสที่เป็นแบบรูปการออกแบบ Strategy โดย x คือ คลาสที่ทำหน้าที่เป็น Strategy ส่วน y คือ คลาสที่ทำหน้าที่เป็น Context อิมพลีเมนต์คลาสของโหนดหลักทั้งหมด คือ

คลาสที่ทำหน้าที่เป็น ConcreteStrategy โดยการตรวจสอบแบบนี้จะเป็นการตรวจสอบแบบรูปแบบมาตรฐาน

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานคือ การที่โหนดหลัก (x) เป็นคลาสอินเตอร์เฟสแต่แทนที่จะมีความสัมพันธ์แบบรีไลเซชันกลับมีความสัมพันธ์แบบเจนเนอรัลไลเซชัน ในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบคลาสสุดท้ายที่มีการสืบทอดว่าเป็นอิมพลีเมนต์คลาสหรือไม่ ถ้าใช่ก็จะทำการตรวจสอบคลาสนี้ในลักษณะของการตรวจสอบหา ConcreteStrategy และทำการตรวจสอบคลาสอื่นๆดังที่ได้กล่าวมาแล้ว

การตรวจสอบหาแบบรูปที่มีการดัดแปลงจากมาตรฐานกรณีสุดท้ายคือ การที่โหนดหลัก (x) เป็นคลาสนามธรรมแทนที่จะตรวจสอบหาความสัมพันธ์แบบรีไลเซชัน จะต้องทำการตรวจสอบหาความสัมพันธ์แบบเจนเนอรัลไลเซชันแทนโดยในกรณีนี้จะทำการเก็บข้อมูลของคลาสที่มาสืบทอดทั้งหมด และตรวจสอบความสัมพันธ์แบบอะกรีเกชันของโหนดหลักและคลาสสุดท้ายของการสืบทอด ถ้าไม่มีความสัมพันธ์ความสัมพันธ์แบบอะกรีเกชันกันจะทำการตรวจสอบคลาสสุดท้ายนี้ในลักษณะของการตรวจสอบหา ConcreteStrategy และทำการตรวจสอบคลาสอื่นๆดังที่ได้กล่าวมาแล้ว

3.1.6.2 ขั้นตอนวิธีในการตรวจจับแบบรูป Strategy

```
A1: The "Anchor Node" finder Algorithm for Strategy Pattern
For Each Class C in Class Diagram CD
  If Class Type of C equals "Interface" or "Abstract" then
    For Each Relation R of C
        If Type of R equals "AggregationC" or "AssociationP" then
          For Each Relation R2 of C
            If Type of R2 is not equals "SelfAssociation"
              Append (x,y) to AnchorSt where x is an interface class and y is related class with x
```

ภาพที่ 3.30 ขั้นตอนวิธี A1 การตรวจจับแบบรูปการออกแบบ Strategy

A2: The "Design Pattern" finder Algorithm for Strategy Pattern

For Each (***x,y***) in ***AnchorSt***

Read File for Getting Data ***attr*** of ***y***

Append ***attr*** to ***YAttr***

If Class Type of ***YAttr*** equals "Concrete" then

For Each Relation ***R*** of ***x***

If Type of ***R*** equals "ImplementationP" then

Read File for Getting Data ***D*** of Related Class ***Rt***

For Each Relation ***R***2 of ***Rt***

If Type of ***R2*** equals "Aggregation" or "Association" and Related Class is not ***x*** or ***y*** then

Read File for Getting Data ***D*** of Related Class ***Rt2***

For Each Relation ***R***3 of ***Rt2***

If Type of ***R3*** is not equals "GeneraliztionC" or "ImplementationC" then

Append ***Rt*** to ***CheckCall*** where Rt is implementation class of x

If Type of ***R*** equals "GeneralizationP" then

Call Perform Function ***GetCascadingInheritance(x)***

Append ***iClass*** to ***TempInherit*** where iClass contains all of the Cascading Inheritance classes returned from Function GetCascadingInheritance()

Read File for Getting Data ***D*** of Last Index of ***TempInherit***

If Class Type of ***x*** equals "Interface" then

For Each Relation ***R2*** of ***D***

If Type of ***R2*** equals "ImplementationP" then

Read File for Getting Data ***D1*** of Related Class ***Rt***

For Each Relation ***R***3 of ***Rt***

If Type of ***R3*** equals "Aggregation" or "Association" and Related Class is not ***x*** or ***y*** then

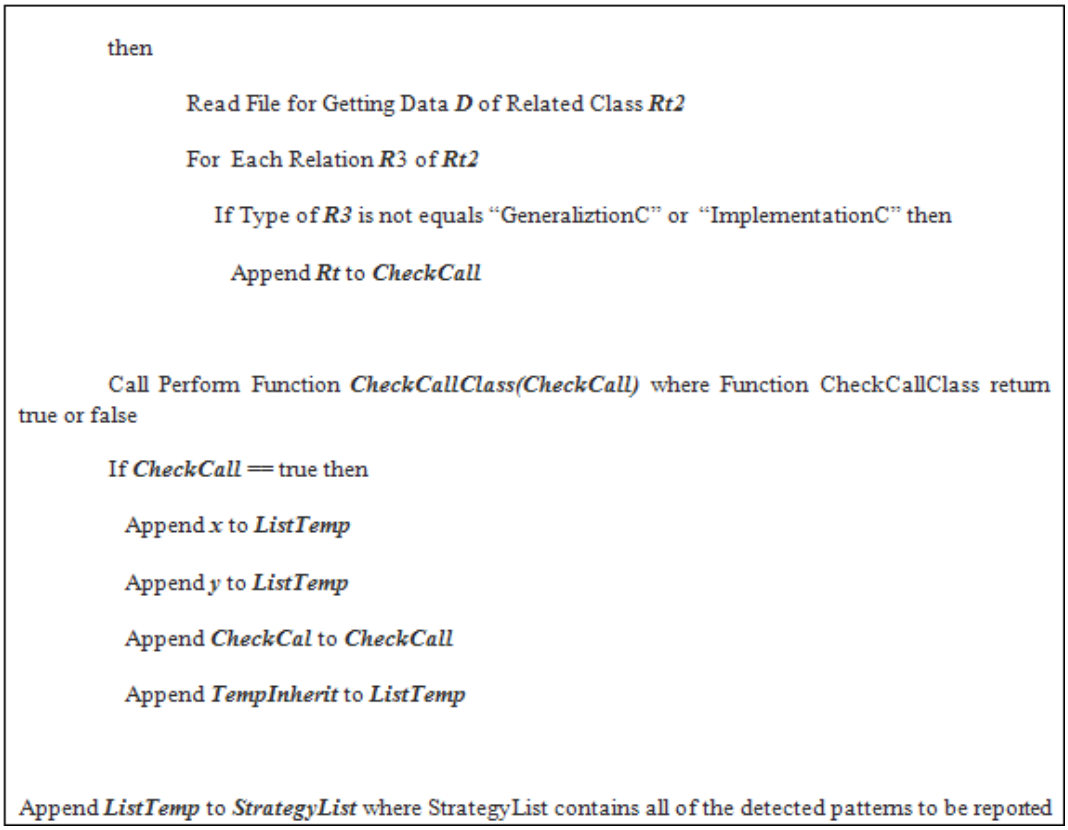
Append ***Rt*** to ***CheckCall***

If Class Type of ***x*** equals "Abstract" then

For Each Relation ***R2*** of ***D***

If Type of ***R2*** equals "Aggregation" or "Association" and Related Class is not ***x*** or ***y***

ภาพที่ 3.31 ขั้นตอนวิธี A2 การตรวจจับแบบรูปการออกแบบ Strategy



ภาพที่ 3.32 ขั้นตอนวิธี A2 การตรวจจับแบบรูปการออกแบบ Strategy (ต่อ)

3.2.7 แบบรูป Template method

3.1.7.1 โครงสร้างของแบบรูป Template method

1) แบบรูปตามมาตรฐาน

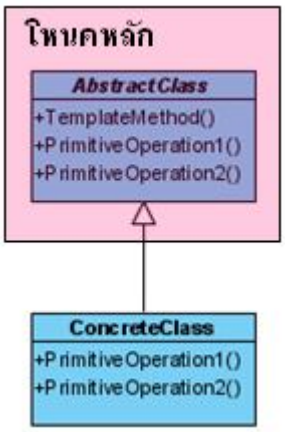
a. ประกอบไปด้วยคลาสที่ทำหน้าที่เป็น AbstractClass ซึ่งมีชนิดของคลาสเป็นคลาสนามธรรม (Abstract) และคลาสที่ทำหน้าที่เป็น ConcreteClass มีชนิดของคลาสเป็นคลาสรูปธรรม (Concrete)

b. คลาสที่ทำหน้าที่เป็น AbstractClass ต้องมีคลาสที่สืบทอดคลาสตนเองอย่างน้อย 1 คลาส คือ ConcreteClass โดยต้องไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือไม่มีคลาสที่สืบทอดตนเองมาเรียกใช้

จากโครงสร้างของแบบรูปการออกแบบที่กล่าวมาข้างต้น ทำให้กำหนดคุณสมบัติของคลาสที่เป็นโหนดหลักของแบบรูป Template method ได้ดังต่อไปนี้

1) เป็นคลาสที่มีชนิดของคลาสได้เพียงรูปแบบเดียว คือ คลาสนามธรรม
2) เป็นคลาสที่ไม่มีความสัมพันธ์แบบแอสโซซิเอชันกับคลาสตนเองหรือกับคลาสที่สืบทอด
3) เป็นคลาสที่มีความสัมพันธ์แบบเจนเนอรัลไลเซชัน

จากคุณสมบัติดังกล่าวจะเห็นได้ว่า โหนดหลัก คือ คลาสที่ทำหน้าที่เป็น AbstractClass นั่นเอง



ภาพที่ 3.33 ตัวอย่างโหนดหลักของแบบรูปการออกแบบ Template method

สามารถอธิบายขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบได้ดังต่อไปนี้

ตามขั้นตอนวิธี A1 ซึ่งแสดงในภาพที่ 3.34 ในหัวข้อ 3.1.7.2 เป็นขั้นตอนวิธีในการค้นหาโหนดหลักของแบบรูปการออกแบบ Template method โดยเริ่มต้นการทำงานด้วยการค้นหาคลาสที่ทำหน้าที่เป็น AbstractClass ซึ่งเป็นคลาสที่มีชนิดของคลาสเป็นคลาสนามธรรม และทำการเก็บคลาสนี้เป็นโหนดหลัก

ตามขั้นตอนวิธี A2 ซึ่งแสดงในภาพที่ 3.35 ในหัวข้อ 3.1.7.2 หลังจากได้ชุดของโหนดหลักแล้ว จะนำชุดของโหนดหลักนี้ไปทำการตรวจสอบคลาสที่มีความสัมพันธ์กับโหนดหลักเพื่อทำการค้นหาคลาสที่ทำหน้าที่เป็น ConcreteClass โดยตรวจสอบว่าโหนดหลักมีความสัมพันธ์แบบเจนเนอรัลไลเซชันหรือไม่ถ้ามีจะตรวจสอบว่าชนิดของคลาสที่มีความสัมพันธ์กับโหนดหลักนี้เป็น

คลาสรูปธรรมหรือไม่ถ้าใช่ จะได้ว่าคลาสทั้งหมดเป็นแบบรูปการออกแบบ Template method โดยสามารถมีการสืบทอดต่อกันไปได้เรื่อยๆเพียงคลาสสุดท้ายของการสืบทอดต้องมีชนิดของคลาสเป็นคลาสรูปธรรม โดยจะได้ว่าทุกคลาสที่สืบทอดต่อเนื่องกันไปนั้นถือเป็นสมาชิกของคลาสในแบบรูปการออกแบบ Template method

3.1.7.2 ขั้นตอนวิธีในการตรวจจับแบบรูป Template method

A1: The "Anchor Node" finder Algorithm for Template method Pattern

For Each Class ***C*** in Class Diagram ***CD***

  If Class Type of ***C*** equals "Abstract" then

    Append ***x*** to ***AnchorTe*** where x is an abstract class

ภาพที่ 3.34 ขั้นตอนวิธี A1 ในการตรวจจับแบบรูปการออกแบบ Template method

A2: The "Design Pattern" finder Algorithm for Template method Pattern

For Each ***x*** in ***AnchorTe***

  For Each Relation ***R*** of ***x***

    If Type of ***R*** equals "GeneralizationP"

      Call Perform Function ***GetCascadingInheritance(x)***

      Append ***iClass*** to ***TempInherit*** where iClass contains all of the Cascading Inheritance classes returned from Function GetCascadingInheritance()

      Read File for Getting Data ***D*** of Last Index of ***TempInherit***

      If Class Type of ***D*** equals "Concrete"

        For Each Relation ***R2*** of ***D***

          If Type of ***R2*** equals "Aggregation" or "Association" and Related Class is not ***x*** then

            Append ***x*** to ***ListTemp***

            Append ***D*** to ***ListTemp***

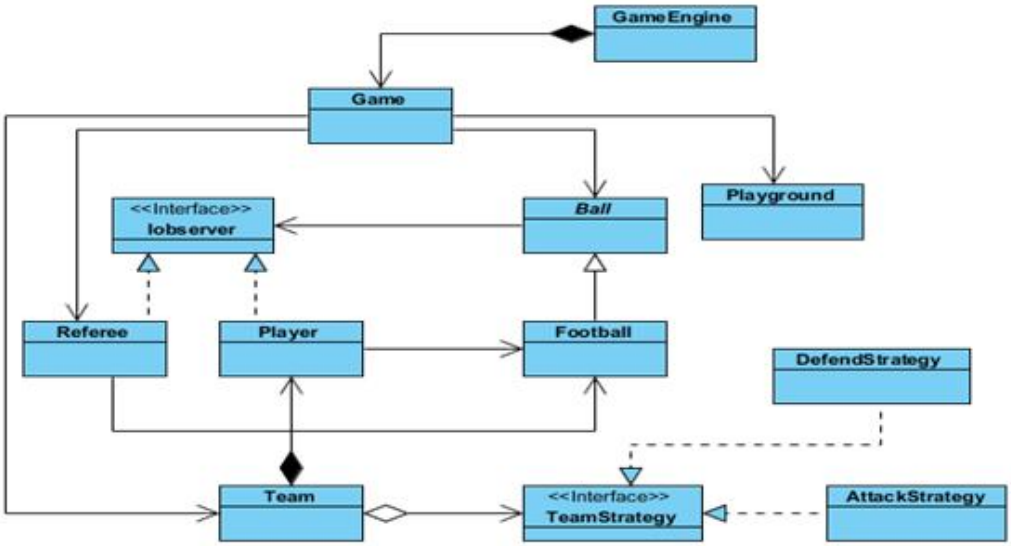
            Append ***iClass*** to ***ListTemp***

Append ***ListTemp*** to ***TemplateList*** where TemplateList contains all of the detected patterns to be reported

ภาพที่ 3.35 ขั้นตอนวิธี A2 ในการตรวจจับแบบรูปการออกแบบ Template method

3.3 ตัวอย่างการทำงานของขั้นตอนวิธีการตรวจจับแบบรูปการออกแบบ

จากขั้นตอนวิธีการตรวจจับแบบรูปการออกแบบทั้ง 7 ขั้นตอนวิธี สามารถแสดงขั้นตอนการตรวจจับรูปแบบได้ดังนี้



ภาพที่ 3.36 ตัวอย่างแผนภาพคลาสในการตรวจจับแบบรูปการออกแบบ [20]

จากตัวอย่างแผนภาพคลาสในภาพที่ 3.36 ทำการส่งออกแผนภาพนี้ให้อยู่ในรูปของแฟ้มเอ็กซ์เอ็มไอ และจากแฟ้มเอ็กซ์เอ็มไอที่ได้จะนำไปผ่าน XMI Parser เพื่อทำการเลือกในส่วนของข้อมูลที่ต้องการ โดยข้อมูลจะประกอบไปด้วย หมายเลขคลาส ชื่อคลาส ชนิดของคลาส ความสัมพันธ์ และคลาสที่มีความสัมพันธ์ ดังแสดงในภาพ 3.37 โดยจะนำข้อมูลเหล่าไปใช้ตรวจสอบกับกฎเพื่อทำการค้นหาโหนดหลัก และรายละเอียดของแบบรูปการออกแบบ

```
ListofClass = {(id, className, classType, {Relation}, {RelatedTo})}

ListofClass = {("FXWWJ2yGAqACJwgn", "TeamStrategy", "Interface",
{"aggregationP","implementP", "implementP"}, {"Team", "AttackStrategy",
"DefendStratyegy"}) ,
("dOJWJ2yGAqACJwhO", "DefendStrategy", "Concrete", {"implementC"},
{"TeamStrategy"}),
("PtsmJ2yGAqACJwdM", "Iobserver", "Interface", {"associationP", "implementP",
"implementP"}, {"Ball", "Player" , "Referee"}) , (...) , (...) , ...}
```

ภาพที่ 3.37 ตัวอย่างข้อมูลที่ถูกกรองจาก XMI Parser จากตัวอย่างแผนภาพคลาส

ข้อมูลของคลาสที่ได้จาก XMI Parser จะถูกนำไปค้นหาโหนดหลัก ในแต่ละแบบรูปการออกแบบจากขั้นตอนวิธี A1 ซึ่งจากตัวอย่างจะได้โหนดหลักของแบบรูปการออกแบบดังแสดงในภาพที่ 3.38

```
AnchorChain = {}

AnchorCom = {("TeamStrategy", "Team")}

AnchorIt = {("Iobserver", "Referee"), ("Iobserver", "Player"), ("Ball", "Football"),
("TeamStrategy", "DefendStrategy"), ("TeamStrategy", "AttackStrategy")}

AnchorMe = {("Football", "Referee"), ("Referee", "Game"), ("Football" ,"Player"),
("Team", "Game"), ("Referee", "Game"), ("Ball", "Game"), ("Playground", "Game")}

AnchorOb = {("Ball", "Game"), ("Iobserver", "Ball")}

AnchorSt = {("Iobserver", "Ball"), ("Ball", "Game"), ("TeamStrategy", "Team")}

AnchorTe = {("Ball", "Football")}
```

ภาพที่ 3.38 โหนดหลักของแบบรูปการออกแบบจากตัวอย่างแผนภาพคลาส

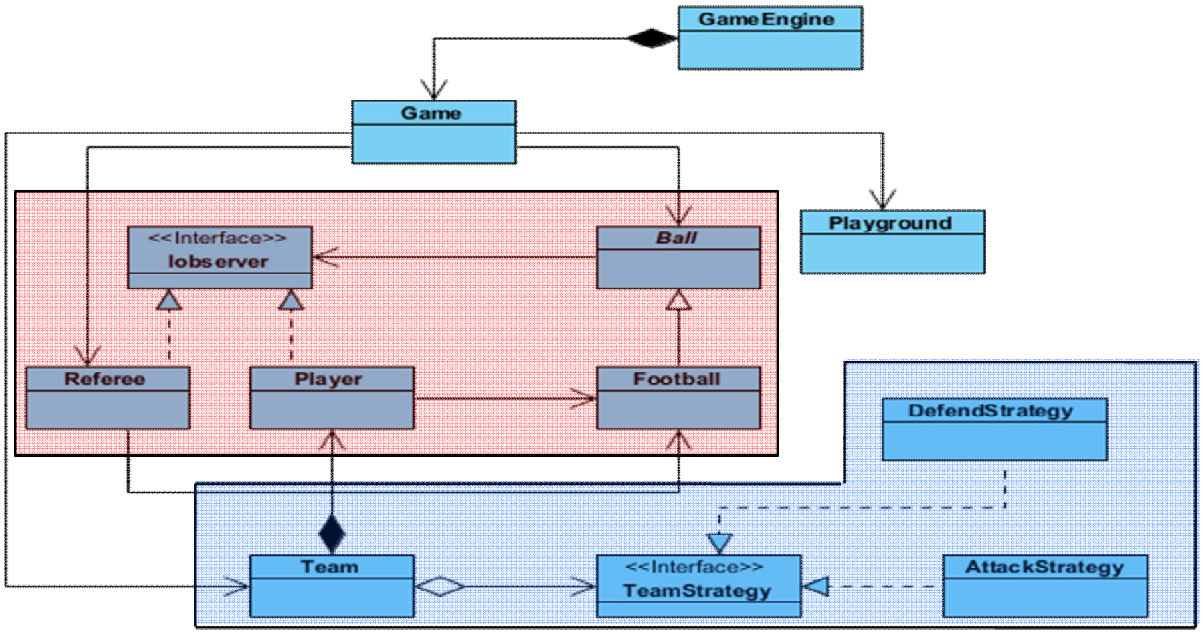
จากนั้นข้อมูลของโหนดหลักที่ได้จากขั้นตอนวิธี A1 จะถูกนำมาพิจารณาต่อ เพื่อนำไปค้นหารายละเอียดว่าเป็นแบบรูปการออกแบบหรือไม่ โดยผ่านขั้นตอนวิธี A2 โดยจะทำการแยกกันค้นหาในแต่ละแบบรูป

ตัวอย่างการค้นหาแบบรูปการออกแบบสำหรับแบบรูป Observer จะพิจารณาจากคู่อันดับ IObserver (Observer) และ Ball (Subject) โดยจะเริ่มจากการพิจารณาข้อมูลของคลาส Ball ว่ามีคุณสมบัติชนิดของคลาสเป็นคลาสอินเตอร์เฟสหรือคลาสนามธรรม ซึ่งจะเห็นได้ว่าคลาส Ball มีชนิดของคลาสเป็นนามธรรม จากนั้นจะทำการพิจารณาความสัมพันธ์แต่ละความสัมพันธ์ของคลาส Ball กับคลาสอื่นๆ พบว่า คลาส Ball มีความสัมพันธ์แบบแอสโซซิเอชันโดยเรียกใช้คลาส IObserver และคลาส Ball มีความสัมพันธ์แบบเจเนอรัลไลเซชันกับคลาส Football จึงทำการกำหนดว่าคลาส Football มีความน่าจะเป็นที่จะเป็น ConcreteSubject

จากนั้นจึงทำการตรวจสอบคลาส IObserver ต่อโดยทำการหาความสัมพันธ์ว่ามีรีไลเซชันหรือเจนเนอรัลไลเซชัน ซึ่ง IObserver มีความสัมพันธ์แบบรีไลเซชันกับคลาส Player และ Referee จึงทำการกำหนดว่าคลาสทั้งสองมีความน่าจะเป็นที่จะเป็นคลาสที่ทำหน้าที่เป็น ConcreteObserver จากนั้นจะทำการตรวจสอบว่าคลาสที่ทำหน้าที่เป็น ConcreteObserver มีความสัมพันธ์กับคลาสที่ทำหน้าที่เป็น ConcreteSubject ในลักษณะที่เรียกใช้หรือไม่ ซึ่งจากตัวอย่างปรากฏว่าใช่ ดังนั้นจะทำการเก็บข้อมูลของ IObserver Ball Player Referee และ Football ลงใน ListTemp และทำการนำข้อมูลจาก ListTemp เก็บใน ObserverList ซึ่งหากมีข้อมูลใน AnchorOb อยู่อีกก็จะทำการตรวจสอบในลักษณะที่กล่าวมาไปเรื่อยๆจนครบทุกคู่ลำดับ ซึ่งจากตัวอย่างจะได้ข้อมูลใน ObserverList คือ {{IObserver, Ball, Player, Referee, Football}}

สำหรับแบบรูป Strategy จะพิจารณาจากคู่อันดับ TeamStrategy และ Team โดยจะเริ่มจากการพิจารณาข้อมูลของคลาส Team ว่ามีคุณสมบัติชนิดของคลาสเป็นคลาสรูปธรรมหรือไม่ ซึ่งจะเห็นได้ว่า คลาส Team มีชนิดของคลาสเป็นคลาสรูปธรรม จากนั้นจะทำการตรวจสอบคลาส TeamStrategy ว่าความสัมพันธ์ว่ามีรีไลเซชันหรือเจนเนอรัลไลเซชัน ซึ่ง TeamStrategy มีความสัมพันธ์แบบรีไลเซชันกับสองคลาส คือ AttackStrategy และ DefendStrategy จากนั้นจะทำการตรวจสอบว่าทั้ง AttackStrategy และ DefendStrategy นั้นไม่มีความสัมพันธ์แบบเรียกใช้คลาส TeamStrategy จริงหรือไม่ ซึ่งหากจริงจะทำการเก็บข้อมูลของ TeamStrategy Team AttackStrategy และ DefendStrategy ลงใน ListTemp และทำการนำข้อมูลจาก ListTemp เก็บใน StrategyList ซึ่งหากมีข้อมูลใน AnchorSt อยู่อีกก็จะทำการตรวจสอบในลักษณะที่กล่าวมาไปเรื่อยๆจนครบทุกคู่ลำดับ ดังนั้นจากตัวอย่างซึ่งมีเพียงหนึ่งคู่ลำดับใน AnchorSt จึงได้ข้อมูลใน StrategyList คือ {{ TeamStrategy, Team, AttackStrategy, DefendStrategy }}

ผลลัพธ์สุดท้ายจากตัวอย่างของแผนภาพคลาส คือ แบบรูป Observer และ แบบรูป Strategy ซึ่งมีทั้งสิ้น 2 รูป ดังแสดงในภาพที่ 3.39

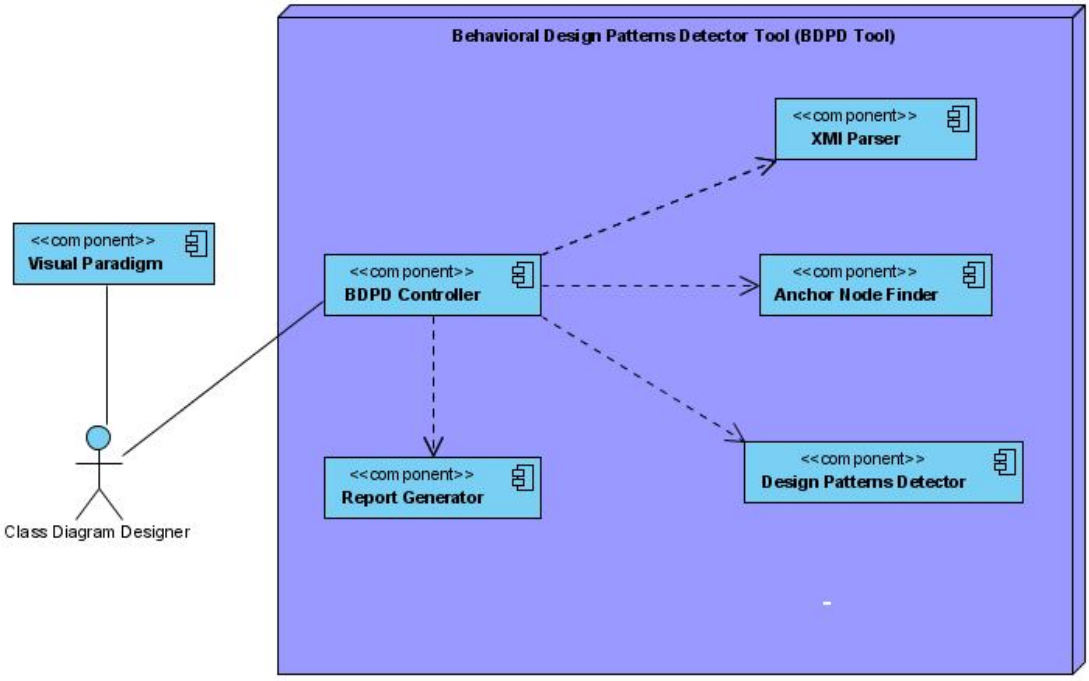


ภาพที่ 3.39 ตัวอย่างผลในการใช้ขั้นตอนวิธีตรวจจับแบบรูปการออกแบบ

บทที่ 4

การวิเคราะห์และออกแบบพัฒนาเครื่องมือ

งานวิจัยนี้เป็นการออกแบบ และพัฒนาเครื่องมือในการตรวจจับแบบรูปการออกแบบ ซึ่งมีรายละเอียดในการทำงานของแบ่งออกเป็น 5 ส่วน ดังแสดงในภาพที่ 4.1



ภาพที่ 4.1 แสดงภาพรวมของเครื่องมือในการตรวจจับแบบรูปการออกแบบแบบอัตโนมัติ

ซึ่งสามารถอธิบายการทำงานของเครื่องมือได้ดังนี้ เมื่อนักออกแบบทำการออกแบบแผนภาพคลาสด้วยเครื่องมือที่ใช้ในการสร้างแผนภาพยูเอ็มแอลเรียบร้อยแล้ว นักออกแบบต้องทำการส่งออก (Export) แผนภาพคลาสให้อยู่ในรูปแบบของแฟ้มเอ็กซ์เอ็มไอ และทำการส่งข้อมูลนี้เข้าสู่เครื่องมือในการตรวจจับแบบรูปการออกแบบเชิงพฤติกรรม (Behavioral Design Pattern Detector Tool) ซึ่งเครื่องมือในการตรวจจับแบบรูปการออกแบบเชิงพฤติกรรมจะแบ่งการทำงานออกเป็น 5 ส่วน

4.1 ภาพรวมของเครื่องมือ

4.1.1 BDPD Controller (Behavioral Design Pattern Detection)

เป็นส่วนที่ใช้ในการควบคุมการทำงานต่างๆของเครื่องมือ เปรียบเสมือนเมท็อดหลัก (Main Method) ใช้ในการจัดการการทำงานต่างๆของเครื่องมือให้เป็นระบบ โดยทำการรับ และส่งข้อมูลต่างๆที่จำเป็นระหว่างส่วนประกอบ (Component) อื่นๆของเครื่องมือ

4.1.2 XMI Parser

เป็นส่วนที่ใช้ในการอ่านแฟ้มเอ็กซ์เอ็มไอของแผนภาพคลาสที่นักออกแบบได้ทำการออกแบบ โดย XMI Parser จะทำการอ่าน และจัดเก็บข้อมูลสำคัญต่างๆที่ใช้ในการตรวจจับแบบรูปการออกแบบ โดยตัดข้อมูลอื่นๆซึ่งไม่จำเป็นในการทำงานทิ้งไป ซึ่งข้อมูลที่ XMI Parser จะอ่านและนำมาจัดเก็บไว้ประกอบไปด้วยข้อมูลต่างๆ ดังนี้ หมายเลขคลาส (Class ID) ชื่อคลาส (Class Name) ชนิดของคลาส (Class Type) ความสัมพันธ์ของคลาส (Class Relation) และคลาสอื่นๆที่มีความสัมพันธ์ (Related Class) โดยหลังจาก XMI Parser ทำการรวบรวมข้อมูลทั้งหมดนี้เรียบร้อยแล้ว จะทำการส่งข้อมูลกลับไปยัง BDPD Controller เพื่อให้ BDPD Controller ทำการส่งข้อมูลไปยังส่วนต่อไป ซึ่งก็คือ Anchor Node Finder

4.1.3 Anchor Node Finder

เป็นส่วนที่ใช้ในการค้นหาโหนดหลัก (Anchor Node) สำหรับแต่ละแบบรูปการออกแบบ โดยการค้นหาโหนดหลักนี้จะทำการสร้างกฎสำหรับค้นหาโหนดหลักสำหรับแต่ละแบบรูปการออกแบบ ซึ่งกฎในการค้นหานี้ได้มาจากลักษณะเฉพาะ และองค์ประกอบของแบบรูปการออกแบบ

โดยโหนดหลักของแต่ละแบบรูปการออกแบบจะถูกจัดเก็บไว้ในรายการ (List) ในรูปแบบของคู่ลำดับ เพื่อนำไปพิจารณาในส่วนต่อไปของระบบ โดยรายการของคู่ลำดับของโหนดหลักนี้จะถูกส่งกลับไปยัง BDPD Controller เพื่อให้ BDPD Controller ทำการส่งข้อมูลไปยังส่วนของ Design Pattern Detector

4.1.4 Design Pattern Detector

เป็นส่วนที่ใช้ในการค้นหาแบบรูปการออกแบบอย่างละเอียด โดยการค้นหาจะไม่ทำการค้นหาทั้งหมดของแผนภาพคลาส แต่จะเริ่มทำการค้นหาจากส่วนที่เป็นคู่ลำดับของโหนดหลัก ซึ่งได้มาจากส่วนของ Anchor Node Finder ซึ่งจะใช้ขั้นตอนวิธีตรวจจับแบบรูปการออกแบบ

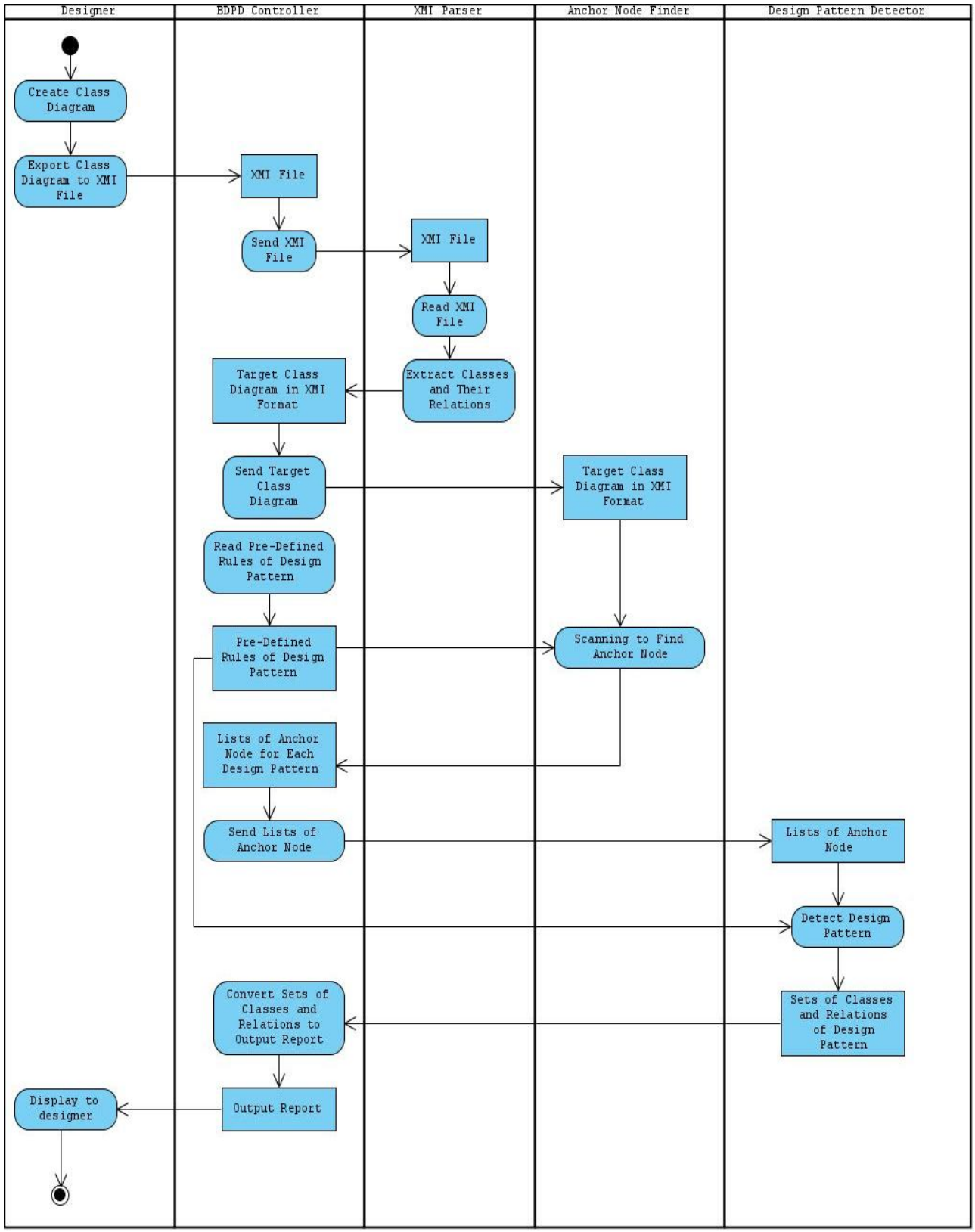
(Design Pattern Detection Algorithm) ซึ่งขั้นตอนวิธีตรวจจับแบบรูปการออกแบบจะมีขั้นตอนวิธีเฉพาะสำหรับแต่ละแบบรูปการออกแบบดังนั้นจะมีการตรวจสอบแผนภาพภาพจำนวนหลายครั้ง เพื่อให้ครบถ้วนกับจำนวนแบบรูปการออกแบบทั้งหมด จากข้อมูลในส่วนนี้จะได้รายการของแบบรูปการออกแบบในแต่ละแบบรูป ซึ่งรายการนี้จะถูกส่งกลับไปยัง BDPD Controller เพื่อให้ BDPD Controller ทำการส่งข้อมูลไปยังส่วนของการออกรายงาน

4.1.5 Report Generator

เป็นส่วนที่ใช้ในการออกรายงานของแบบรูปการออกแบบที่พบจากแผนภาพคลาสที่ออกแบบไว้ โดยรายงานนี้จะทำการแจกแจงชนิด และจำนวนของแบบรูปการออกแบบทั้งหมดที่พบ พร้อมระบุชุดของคลาส และความสัมพันธ์ ของแต่ละแบบรูปการออกแบบ

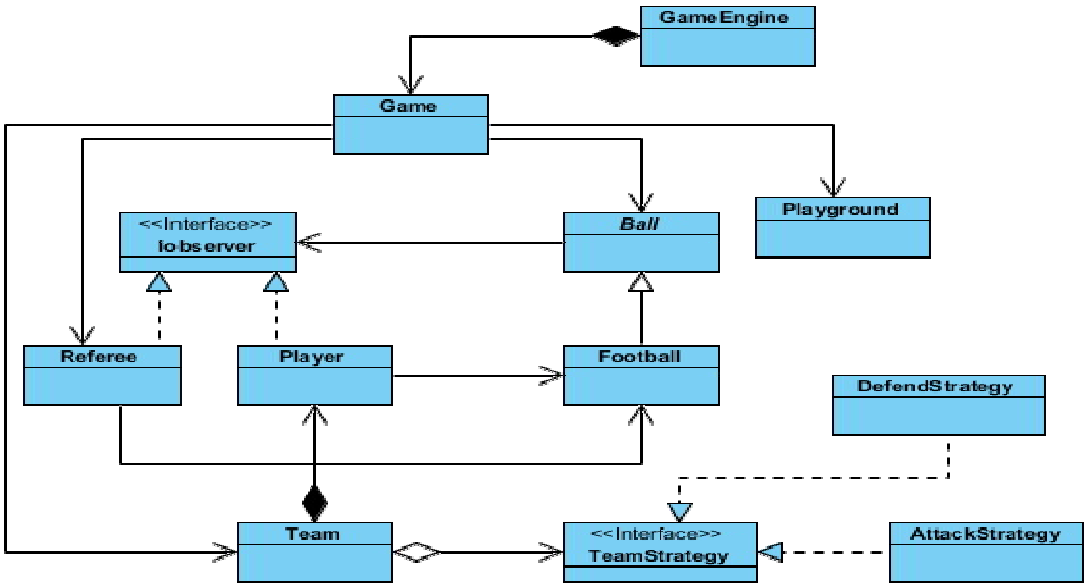
4.2 การทำงานของเครื่องมือ

จากรายละเอียดการทำงานในข้างต้น สามารถแสดงการทำงานของเครื่องมือในการตรวจจับแบบรูปการออกแบบในรูปแบบของแผนภาพกิจกรรมได้ดังภาพที่ 4.2 โดยเครื่องมือที่พัฒนาจะสามารถตรวจจับแบบรูปการออกแบบพฤติกรรมจำนวน 7 แบบรูป สาเหตุที่เลือกแบบรูปเชิงพฤติกรรมจำนวน 7 แบบรูปนี้ เนื่องมาจาก มีการใช้งานแบบรูปเหล่านี้ในการออกแบบระบบค่อนข้างสูง ซึ่งการทำงานของเครื่องมือจะเริ่มจากนักออกแบบทำการออกแบบแผนภาพคลาส จากนั้นทำการแปลงแผนภาพคลาสให้อยู่ในรูปของแฟ้มเอ็กซ์เอ็มไอ แล้วทำการส่งแฟ้มเอ็กซ์เอ็มไอให้กับตัวควบคุม (Controller) ซึ่งตัวควบคุมจะทำการส่งแฟ้มเอ็กซ์เอ็มไอให้กับ XMI Parser เพื่อทำการอ่านและกรองข้อมูลต่างๆที่จำเป็นต่อการตรวจจับแบบรูปการออกแบบ ซึ่งในที่นี้จะเรียกข้อมูลที่จำเป็นเหล่านี้ว่า Target Class Diagram จากนั้นจะทำการตรวจสอบ Target Class Diagram เพื่อทำการค้นหาโหนดหลัก และส่งรายการของโหนดหลักกลับไปยังตัวควบคุม เพื่อเข้าสู่ขั้นตอนการค้นหารายละเอียดของแบบรูปการออกแบบ ซึ่งจะทำการตรวจสอบทีละรายการไปจนจบรายการของโหนดหลักทั้งหมด และทำการออกรายงานผลการตรวจจับให้แก่นักออกแบบ



ภาพที่ 4.2 แผนภาพกิจกรรมของเครื่องมือในการตรวจจับแบบรูปการออกแบบ

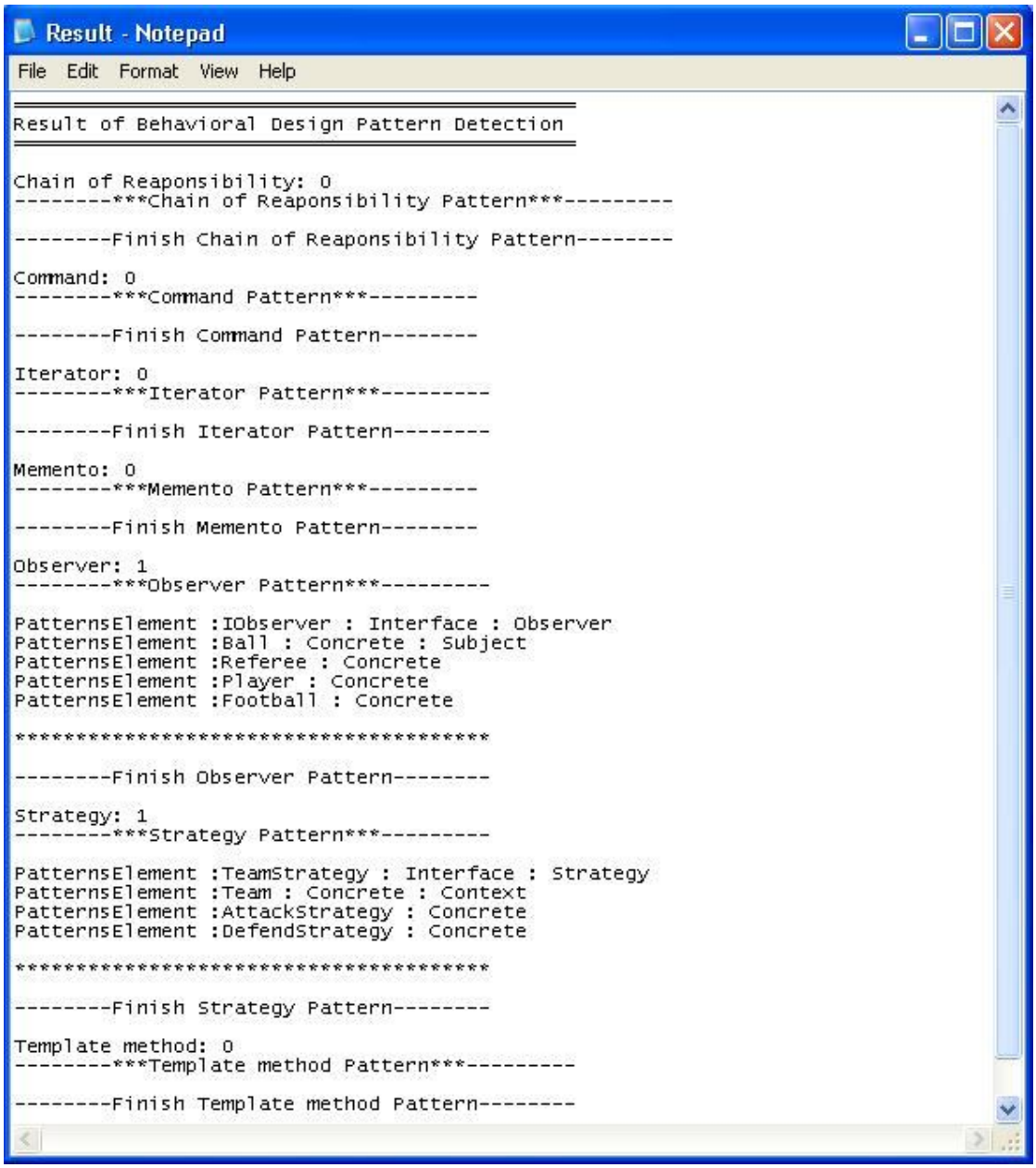
จากการการวิเคราะห์และออกแบบพัฒนาเครื่องมือในการตรวจจับแบบรูปการออกแบบ สามารถแสดงตัวอย่างการทำงานของเครื่องมือได้ดังนี้



ภาพที่ 4.3 ตัวอย่างแผนภาพคลาสที่ต้องการนำมาตรวจสอบ [19]

จากตัวอย่างแผนภาพคลาสในภาพที่ 4.3 เมื่อผู้ใช้สร้างแผนภาพคลาสเรียบร้อยแล้ว ผู้ใช้จะต้องทำการส่งออกแผนภาพนี้ให้อยู่ในรูปของแฟ้มเอ็กซ์เอ็มไอ และจากแฟ้มเอ็กซ์เอ็มไอที่ได้จากการแปลงนี้นำไปผ่าน XMI Parser เพื่อทำการเลือกในส่วนของข้อมูลที่ต้องการ โดยข้อมูลเหล่านี้จะถูกส่งเข้าสู่ขั้นตอนวิธีในการตรวจจับแบบรูปการออกแบบซึ่งได้กล่าวมาแล้วในบทที่ 3

โดยจากแผนภาพคลาสตัวอย่าง เมื่อนำไปเป็นข้อมูลเข้าของโปรแกรมจะทำให้สามารถค้นหาแบบรูปได้ทั้งสิ้น 2 แบบรูป และแสดงผลออกมาในรูปของแฟ้มข้อความ (Text File) ดังแสดงในภาพที่ 4.4



ภาพที่ 4.4 ผลลัพธ์จากการตรวจสอบของเครื่องมือตรวจจับแบบรูปการออกแบบ

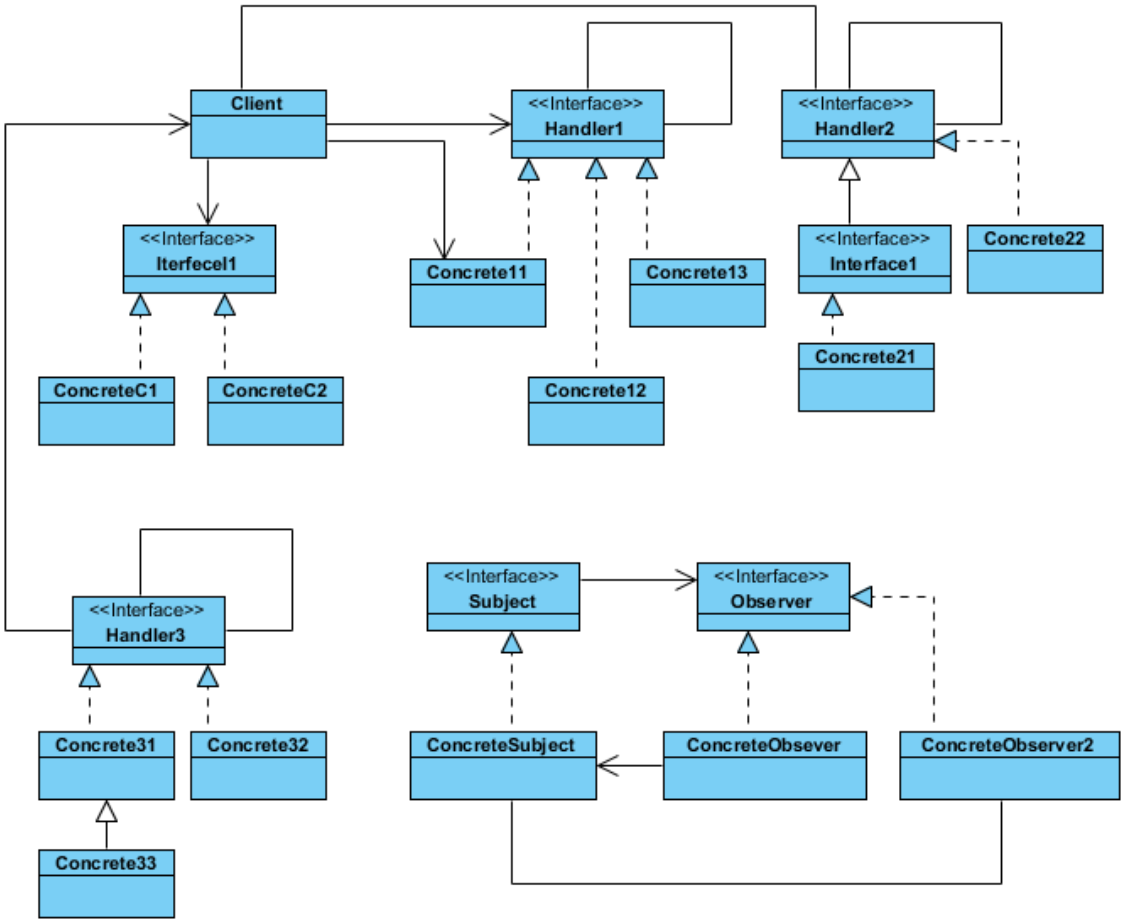
บทที่ 5

การทดสอบระบบ

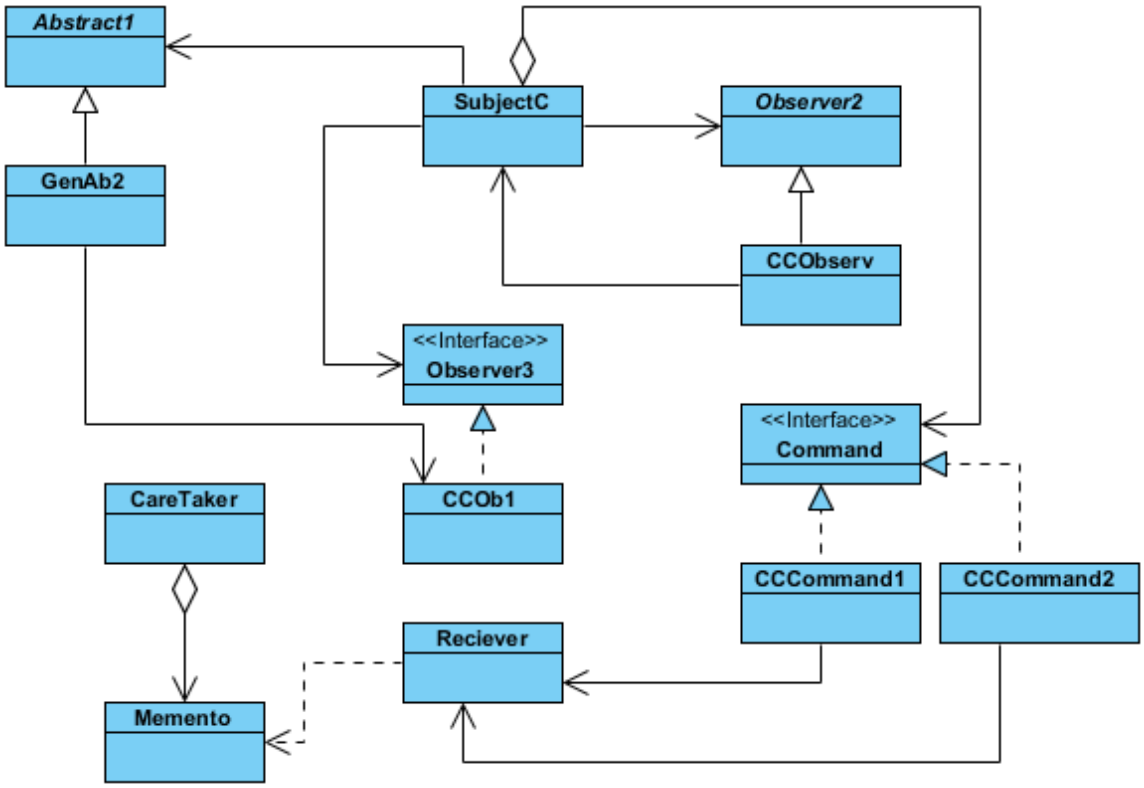
งานวิจัยนี้เป็นการออกแบบ และพัฒนาเครื่องมือในการตรวจจับแบบรูปการออกแบบ โดยการทดสอบระบบจะมีการใช้ข้อมูล 2 ประเภท คือ กรณีศึกษาที่สร้างขึ้นเอง และข้อมูลจากระบบจริงจำนวน 2 ระบบ โดยข้อมูลที่สร้างขึ้นจะทำการสร้างให้คลอบคลุมแบบรูปทั้งหมดตามขอบเขตการวิจัย

5.1 กรณีศึกษาที่สร้างขึ้น

ข้อมูลของแผนภาพคลาสที่สร้างขึ้นเพื่อใช้ในการทดสอบความแม่นยำของเครื่องมือ โดยข้อมูลนี้จะประกอบด้วยแผนภาพของแบบรูปทั้ง 7 แบบรูปตามขอบเขตการวิจัยโดยประกอบด้วยแบบรูปมาตรฐาน และแบบรูปที่ดัดแปลงจากมาตรฐาน ซึ่งสามารถแสดงแผนภาพคลาสซึ่งใช้เป็นข้อมูลนำเข้าของระบบได้ดังภาพที่ 5.1-5.2



ภาพที่ 5.1 แผนภาพคลาสของกรณีศึกษาที่สร้างขึ้น



ภาพที่ 5.2 แผนภาพคลาสของกรณีศึกษาที่สร้างขึ้น (ต่อ)

ผลลัพธ์จากการตรวจจับแบบรูปการออกแบบสามารถแสดงได้ ดังตารางที่ 5.1

ตารางที่ 5.1 ผลลัพธ์จากการตรวจจับแบบรูปการออกแบบของกรณีศึกษาที่สร้างขึ้น

Design Pattern	Number of Design Pattern	Design Pattern Element	Expected Number of Design Pattern	%
Chain of Responsibility	2	{{Handler1, Client, Concrete11, Concrete12, Concrete13}, {Handler2, Client, Interface1, Concrete21, Concrete22}}	2	100
Command	1	{{Command, SubjectC, CCCommand1, CCCommand2, Reciever}}	1	100
Iterator	1	{{Abstract1, GenAb2, Observer3, CCOb1, SubjectC}}	1	100
Memento	1	{{Memento, Reciever, CareTaker}}	1	100
Observer	2	{{Observer, Subject, ConcreteObserver, ConcreteObserver2, ConcreteSubject}, {Observer2, CCObserv, SubjectC}}	2	100
Strategy	1	{{Interface1, Client, concreteC1, ConcreteC2}}	1	100
Template method	2	{{Observer2, CCObserv}, {Abstract1, GenAb2}}	2	100

5.2 ข้อมูลจากระบบจริง

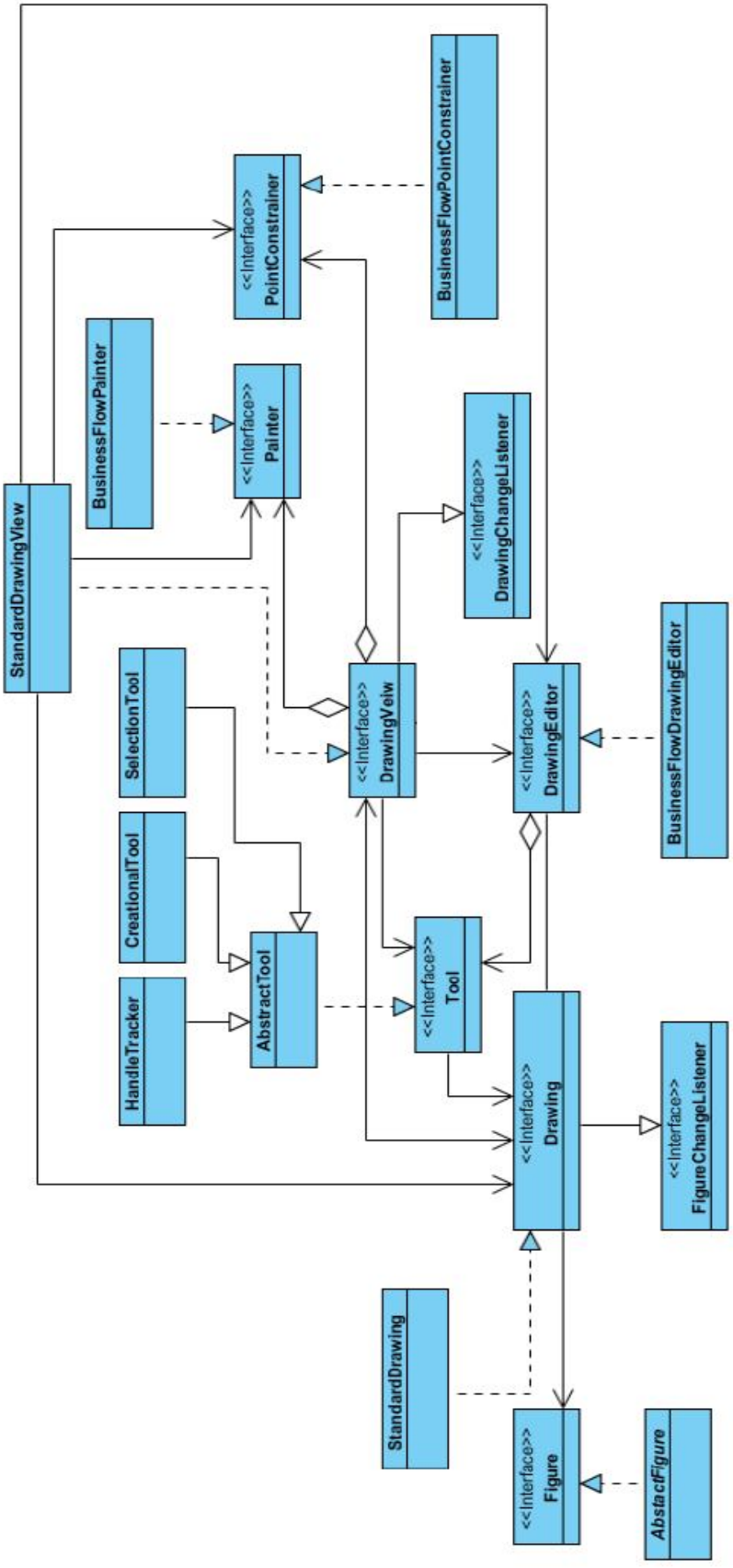
เป็นข้อมูลที่ได้มาจากระบบจริงที่มีการออกแบบระบบโดยใช้แบบรูปการออกแบบในการออกแบบและพัฒนาโปรแกรม โดยข้อมูลจากระบบจริงนี้ คือข้อมูลที่ได้มาจากการออกแบบ JHotDraw [21, 22] การออกแบบ Soccer Engine [20] และการออกแบบโปรแกรม Ubiquitous Video [23]

JHotDraw คือ แอพลิเคชั่นเฟรมเวิร์คที่ใช้ในการวาดแผนภาพโครงสร้างต่างๆทางด้านซอฟต์แวร์ โดย JHotDraw สามารถแบ่งเป็นการทำงานย่อยๆได้หลายการทำงาน โดยเลือกการทำงานของส่วนของ Drawing and DrawingView classes และ The DrawingEditor classes [24, 25, 26]

Soccer Engine เป็นโปรแกรมที่ใช้แบบรูปการออกแบบในการแก้ปัญหาด้านการจำลองการเล่นเกมฟุตบอล

โปรแกรม Ubiquitous Video เป็นโปรแกรมที่ใช้ในการแสดงภาพจากกล้องวีดีโอให้ได้ภาพในลักษณะที่มีมุมมองต่างๆกัน

5.2.1 แผนภาพคลาสของข้อมูลจากโปรแกรม JHotDraw



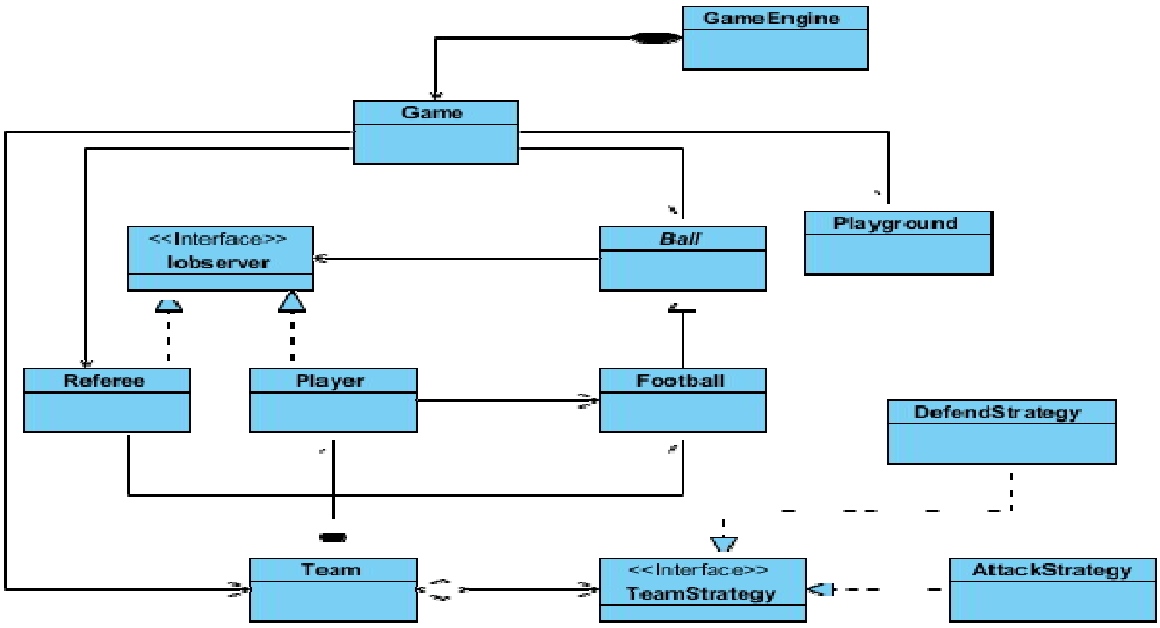
ภาพที่ 5.3 แผนภาพคลาสของ JHotDraw [24, 26]

ผลลัพธ์จากการตรวจจับแบบรูปการออกแบบสามารถแสดงได้ ดังตารางที่ 5.2

ตารางที่ 5.2 ผลลัพธ์จากการตรวจจับแบบรูปการออกแบบของโปรแกรม JHotDraw

Design Pattern	Number of Design Pattern	Design Pattern Element	Expected Number of Design Pattern	%
Chain of Responsibility	0	-	0	100
Command	0	-	0	100
Iterator	0	-	0	100
Memento	0	-	0	100
Observer	1	{{Drawing, DrawingView, StandardDrawing, StandardDrawing View}}	1	100
Strategy	4	{{Painter, StandardDrawingView, BusinessFlowPainter, DrawingView}, {PointConstrainer, StandardDrawingView, BusinessFlow PointConstrainer, DrawingView}, {DrawingEditor, StandardDrawingView, BusinessFlow DrawingEditor, DrawingView} {Drawing, StandardDrawingView, StandardDrawing, DrawingView }}	4	100
Template method	1	{{AbstractTool, HandleTracker, CreationalTool, SelectionTool}}	1	100

5.2.2 แผนภาพคลาสของข้อมูลจาก Soccer Engine



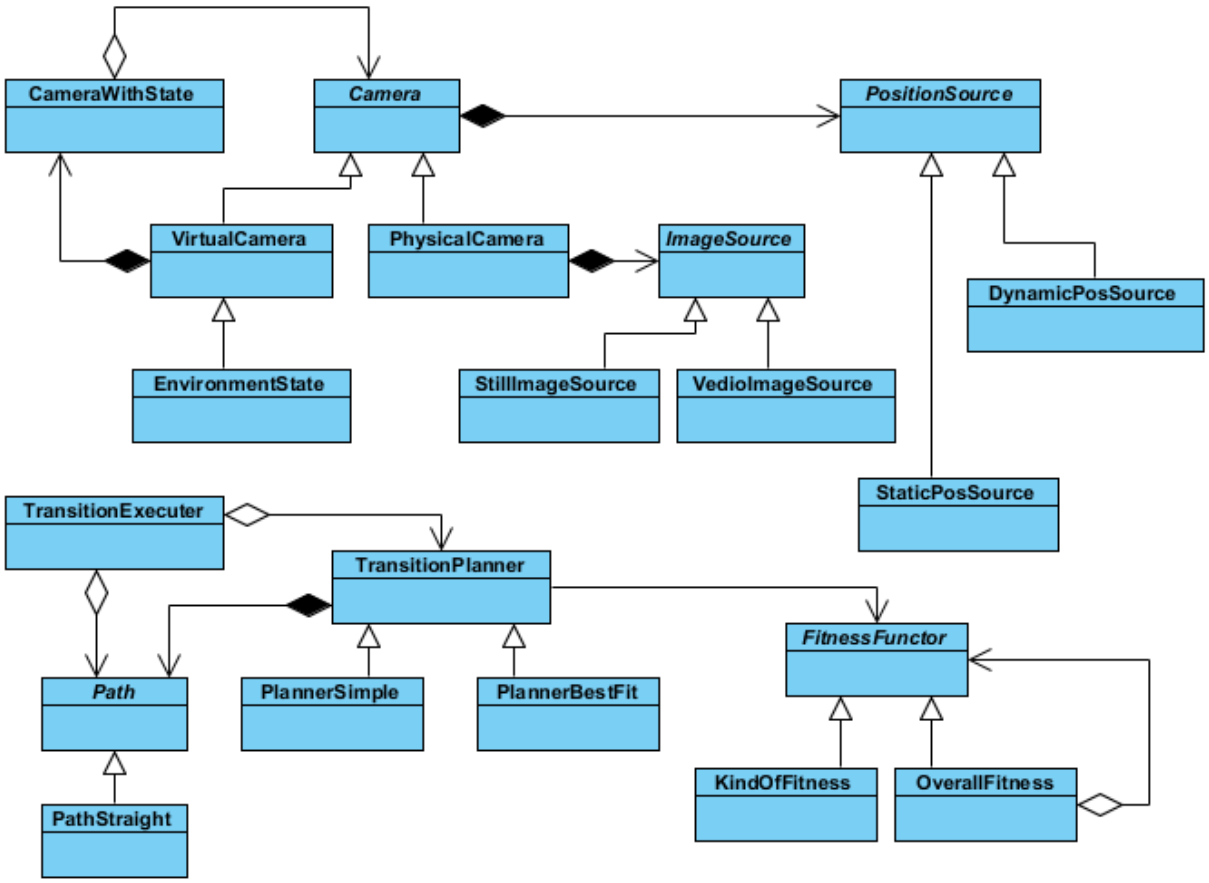
ภาพที่ 5.4 แผนภาพคลาส Soccer Engine [20]

ผลลัพธ์จากการตรวจจับแบบรูปการออกแบบสามารถแสดงได้ ดังตารางที่ 5.3

ตารางที่ 5.3 ผลลัพธ์จากการตรวจจับแบบรูปการออกแบบของ Soccer Engine

Design Pattern	Number of Design Pattern	Design Pattern Element	Expected Number of Design Pattern	%
Chain of Responsibility	0	-	0	100
Command	0	-	0	100
Iterator	0	-	0	100
Memento	0	-	0	100
Observer	1	{{IObserver, Ball, Player, Referee, Football}}	1	100
Strategy	1	{{TeamStrategy, Team, AttackStrategy, DefendStrategy}}	1	100
Template method	1	{{Ball, Football}}	1	100

5.2.3 แผนภาพคลาสของข้อมูลจากโปรแกรม Ubiquitous Video



ภาพที่ 5.5 แผนภาพคลาสโปรแกรม Ubiquitous Video [23]

ผลลัพธ์จากการตรวจจับแบบรูปการออกแบบสามารถแสดงได้ ดังตารางที่ 5.4

ตารางที่ 5.4 ผลลัพธ์จากการตรวจจับแบบรูปการออกแบบของโปรแกรม Ubiquitous Video

Design Pattern	Number of Design Pattern	Design Pattern Element	Expected Number of Design Pattern	%
Chain of Responsibility	1	{{FitnessFunctor, TransitionPlaner, KindOfFitness, OverallFitness}}	1	100
Command	0	-	0	100
Iterator	0	-	0	100
Memento	0	-	0	100
Observer	0	-	0	100
Strategy	1	{{Path, TransitionExecuter, PathStraight}}	1	100
Template method	4	{{Camera, VirtualCamera, PhysicalCamera, EnvironmentState}, {PositionSource, StaticPosSource, DynamicPosSource}, {Path, PathStraight}, {ImageSource, StillImageSource, VideoImageSource}}	4	100

ซึ่งผลลัพธ์ที่ได้จากการทดสอบขั้นตอนวิธีและเครื่องมือตรวจจับแบบรูปการออกแบบของระบบทั้ง 3 ได้แสดงไว้ในภาคผนวก ก – ค

5.3 วิเคราะห์ผลการทดสอบระบบ

จากการทดสอบระบบเพื่อเปรียบเทียบการตรวจจับแบบรูปการออกแบบพฤติกรรมของขั้นตอนวิธีและเครื่องมือที่ได้ออกแบบ กับงานวิจัยที่เกี่ยวข้อง [14, 15] สามารถตรวจจับแบบรูปการออกแบบเชิงพฤติกรรม ดังแสดงในตารางที่ 5.5

ตารางที่ 5.5 แสดงจำนวนแบบรูปการออกแบบเชิงพฤติกรรมที่สามารถตรวจจับได้

Behavioral Design Pattern	**Detection of Behavioral Design Pattern**	Automatic Design Pattern Detection	Design Pattern Detection Using Similarity Scoring	Design Pattern Detection Using Template Matching
Chain of Responsibility	**Yes**	No	No	Yes
Command	**Yes**	No	Yes	No
Interpreter	**No**	No	No	No
Iterator	**Yes**	No	No	Yes
Mediator	**No**	Yes	No	Yes
Memento	**Yes**	No	No	No
Observer	**Yes**	No	Yes	No
State	**No**	No	} Yes	} Yes
Strategy	**Yes**	No		
Template method	**Yes**	No	Yes	Yes
Visitor	**No**	Yes	Yes	Yes
Support Pattern	**7/11**	4/11	6/11	6/11

จากตารางจะเห็นได้ว่า การตรวจจับแบบรูปการออกแบบเชิงพฤติกรรมของงานวิจัยที่เกี่ยวข้องของแบบรูป State และ Strategy ไม่สามารถแยกการทำงานออกจากกันได้เนื่องจากแบบรูปทั้งสองมีลักษณะที่ใกล้เคียงกัน ดังนั้นหากกลุ่มของคลาสใดที่สามารถตรวจจับได้ว่าเป็นแบบรูป Strategy ก็จะแสดงเป็นแบบรูป State ด้วยเช่นกัน

ข้อมูลนำเข้าและข้อมูลส่งออกของขั้นตอนวิธีและเครื่องมือที่ได้ออกแบบ และงานวิจัยที่เกี่ยวข้องสามารถแสดงได้ ดังตารางที่ 5.6

ตารางที่ 5.6 ข้อมูลนำเข้า และข้อมูลส่งออกของขั้นตอนวิธีและเครื่องมือที่ได้ออกแบบ และงานวิจัยที่เกี่ยวข้อง

	Input	Output
Detection of Behavioral Design Pattern	**Class Diagram**	**Text File**
Automatic Design Pattern Detection	Source Code	Text File
Design Pattern Detection Using Similarity Scoring	Source Code to Class Diagram	Text File
Design Pattern Detection Using Template Matching	Source Code to Class Diagram	Text File

จากตารางจะเห็นได้ว่างานวิจัยที่เกี่ยวข้องจะมีข้อมูลนำเข้าเป็นรหัสต้นฉบับ (Source code) โดยในงานวิจัย [14, 15] จะต้องนำรหัสต้นฉบับที่จะนำไปตรวจสอบมาทำการแปลงให้เป็นแผนภาพคลาสเสียก่อน ซึ่งทำให้การทำงานมีความซับซ้อนเนื่องจากการตรวจสอบหาแบบรูปการออกแบบของขั้นตอนวิธีของงานวิจัยทั้งสองใช้แผนภาพคลาสในการตรวจสอบ แต่ในงานวิจัยที่พัฒนาขึ้นนี้สามารถใช้งานได้สะดวกกว่างานวิจัยที่เกี่ยวข้องเนื่องจากใช้แผนภาพคลาสในการตรวจจับแบบรูปการออกแบบโดยไม่ต้องทำการ สร้างรหัสต้นแบบก่อนแล้วจึงตรวจจับ

การทำงานของงานวิจัย [14] จะใช้ข้อมูลแผนภาพคลาสในการตรวจจับแบบรูปการออกแบบ โดยจะทำการแบ่งกลุ่ม (Segment) ของแผนภาพคลาสของระบบ และแผนภาพของแบบรูปด้วยความสัมพันธ์ชนิดต่างๆของแผนภาพคลาส จากนั้นจะทำการแปลงแผนภาพคลาสให้อยู่ในรูปของกราฟ (Graph) ซึ่งกราฟที่ใช้ในงานวิจัยชิ้นนี้มีทั้งสิ้น 2 ส่วน คือ กราฟของระบบที่ต้องการตรวจสอบ และกราฟของแบบรูปการออกแบบ จากนั้นจะนำข้อมูลของกราฟมาทำการวิเคราะห์เทียบกับแบบรูปการออกแบบ โดยเทียบทั้งลักษณะของคลาส และความสัมพันธ์ ทำให้งานวิจัยนี้มีค่าสัญกรณ์โอใหญ่ (Big O) เท่ากับ O (n^m) โดยที่ n คือ จำนวนคลาสทั้งหมด และ m คือ จำนวนความสัมพันธ์ระหว่างคลาสในแบบรูปการออกแบบ

การทำงานของงานวิจัย [15] จะใช้ข้อมูลเป็นแผนภาพคลาสในการตรวจจับแบบรูปการออกแบบ โดยจะทำการนำแผนภาพคลาสของแบบรูปการออกแบบ และระบบที่ต้องการตรวจสอบมาแปลงให้อยู่ในรูปของเมตริกซ์ (Matrix) ซึ่งเป็นเมตริกซ์ผลรวมของทุกๆคลาสในแผนภาพคลาสโดยใช้ผลคูณสหสัมพันธ์ (Cross Correlation) จากนั้นประยุกต์ใช้ขั้นตอนวิธีการคำนวณหาคะแนนความคล้ายคลึง เพื่อทำการหาคะแนนความคล้ายคลึงของจุดทั้งหมด ทำให้งานวิจัยนี้มีค่าสัญกรณ์โอใหญ่เท่ากับ O (n^m) โดยที่ n คือ จำนวนคลาสทั้งหมด และ m คือ จำนวนคลาสสมาชิกของแบบรูปการออกแบบและความสัมพันธ์ระหว่างสมาชิกทั้งหมด

การทำงานของขั้นตอนวิธีและเครื่องมือที่ออกแบบในงานวิจัยนี้ จะใช้ข้อมูลเป็นแผนภาพคลาส และทำการค้นหาโหนดหลักเพื่อใช้เป็นจุดเริ่มต้นในการค้นหารายละเอียดของแบบรูปการออกแบบทำให้งานวิจัยนี้มีค่าสัญกรณ์โอใหญ่เท่ากับ O (n) +O (m· (s+t+u)) โดยที่ n คือ จำนวนคลาสทั้งหมดที่นำมาค้นหาโหนดหลัก m คือ จำนวนโหนดหลัก s t และ u คือ จำนวนความสัมพันธ์ของโหนดหลักที่ต้องทำการตรวจสอบ โดยหากกำหนดให้ k แทน ค่าของ s+t+u จะทำให้ได้ค่าสัญกรณ์โอใหญ่เท่ากับ O (n) +O (m·k) ซึ่งค่าของสัญกรณ์ O (m·k) มีค่ามากกว่า จึงได้ค่าสัญกรณ์โอ

ใหญ่ของขั้นตอนวิธีเท่ากับ O (m·k) ซึ่งสามารถแสดงตารางเปรียบเทียบค่าสัญกรณ์โอใหญ่ได้ดังตารางที่ 5.7

ตารางที่ 5.7 เปรียบเทียบค่าสัญกรณ์โอใหญ่ของขั้นตอนวิธีและเครื่องมือที่ได้ออกแบบกับงานวิจัยที่เกี่ยวข้อง

	Big O
Detection of Behavioral Design Pattern	**O(m·k)**
Design Pattern Detection Using Similarity Scoring	$O(n^m)$
Design Pattern Detection Using Template Matching	$O(n^m)$

จากตารางจะเห็นได้ว่า ขั้นตอนวิธีและเครื่องมือที่ออกแบบในงานวิจัยนี้มีค่าสัญกรณ์โอใหญ่ (Big O) ที่น้อยกว่างานวิจัยที่เกี่ยวข้อง ซึ่งค่าสัญกรณ์โอใหญ่นี้ส่งผลต่อการทำงานของเครื่องมือตรวจจับแบบรูปการออกแบบ

บทที่ 6
บทสรุป

ในบทนี้จะกล่าวถึงสรุปผลการวิจัย ปัญหาและข้อจำกัดที่พบจากการวิจัย และข้อเสนอแนะ จากการตรวจจับแบบรูปการออกแบบพฤติกรรมทั้งสิ้น 7 แบบรูปการออกแบบ

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอขั้นตอนวิธี และเครื่องมือในการตรวจจับแบบรูปการออกแบบพฤติกรรม โดยสามารถตรวจจับแบบรูปการออกแบบเชิงพฤติกรรมจำนวน 7 แบบรูป จากจำนวน 11 แบบรูป ซึ่งมีการใช้งานค่อนข้างบ่อยในการออกแบบและพัฒนาระบบ [27] ซึ่งประกอบไปด้วย

1) แบบรูปการออกแบบ Chain of Responsibility
2) แบบรูปการออกแบบ Command
3) แบบรูปการออกแบบ Iterator
4) แบบรูปการออกแบบ Memento
5) แบบรูปการออกแบบ Observer
6) แบบรูปการออกแบบ Strategy
7) แบบรูปการออกแบบ Template Method

โดยเครื่องมือในการตรวจจับแบบรูปการออกแบบจะทำการรับข้อมูลนำเข้าที่อยู่ในรูปของไฟล์เอ็กซ์เอ็มไอ ดังนั้นผู้ใช้ต้องทำการสร้างแผนภาพคลาสจากเครื่องมือที่สามารถนำข้อมูลส่งออกให้อยู่ในรูปของไฟล์เอ็กซ์เอ็มไอ เครื่องมือสามารถตรวจจับแบบรูป และแสดงข้อมูลของแบบรูปที่สามารถตรวจจับได้ให้อยู่ในรูปของแฟ้มข้อความ โดยข้อมูลที่แสดงได้จะแสดงจำนวนของแบบรูปการออกแบบที่สามารถตรวจจับ ชื่อคลาส ชนิดของคลาส โดยการตรวจจับแบบรูปแต่ละแบบจะมีขั้นตอนวิธีในการตรวจจับเป็นเฉพาะของแต่แบบรูปการออกแบบ ซึ่งการตรวจจับแบบรูปการออกแบบจะทำการแบ่งการตรวจสอบออกเป็นสองส่วนคือ การค้นหาโหนดหลัก และการค้นหารายละเอียดของแบบรูปการออกแบบ โดยสาเหตุที่ต้องทำการค้นหาโหนดหลักนั้นเพื่อ ทำให้การตรวจสอบทำได้รวดเร็วขึ้น และหากการวาดแผนภาพมีการเปลี่ยนแปลงลำดับการวาดก็จะไม่ส่งผลต่อการตรวจจับแบบรูป เนื่องจากทำการกำหนดโหนดหลัก และค้นหารายละเอียดของแบบรูปโดยเริ่มจากโหนดหลักเท่านั้น ไม่ต้องทำการค้นหาจากคลาสทั้งหมด ดังนั้นขั้นตอนวิธีในการ

ตรวจจับแบบรูปการออกแบบสำหรับแต่ละแบบรูปจะถูกแบ่งขั้นตอนวิธีออกเป็น 2 ส่วน คือ ส่วนของการค้นหาโหนดหลัก และส่วนการค้นหารายละเอียด โดยจะเริ่มจากส่วนการค้นหาโหนดหลักก่อน และตามด้วยการค้นหารายละเอียด ซึ่งในขั้นตอนการหาโหนดหลักนั้นจะทำการค้นหาทุกคลาสที่มี แต่การค้นหารายละเอียดแบบรูปการออกแบบจะทำการค้นหา โดยเริ่มจากโหนดหลักและคลาสที่มีความสัมพันธ์กับโหนดหลักนั้นๆ

จากการทดสอบขั้นตอนวิธีและเครื่องมือที่ได้ออกแบบไว้สามารถนำไปประยุกต์ใช้งานได้จริง โดยได้ทำการทดลองโดยใช้ข้อมูลในสองรูปแบบ คือ กรณีศึกษาที่สร้างขึ้นโดยข้อมูลนี้จะเป็นการสร้างแผนภาพคลาสที่มีจำนวนของแบบรูปการออกแบบครบถ้วนทั้ง 7 แบบรูป และข้อมูลจากระบบจริงซึ่งจะมีแบบรูปการออกแบบกระจายกันไปในแต่ละกรณีทดสอบ จากการทดสอบจะเห็นได้ว่า สามารถตรวจจับและแสดงแบบรูปการออกแบบได้ ซึ่งในแบบรูปการออกแบบเชิงพฤติกรรมขั้นตอนวิธีที่สร้างขึ้นสามารถแยกแบบรูปการออกแบบออกจากกันได้ และสามารถค้นหาแบบรูปการออกแบบที่ดัดแปลงจากมาตรฐาน ซึ่งมีการเพิ่มหรือลดความสัมพันธ์ได้อีกด้วย แต่หากมีแบบรูปการออกแบบในหมวดอื่นที่ไม่ใช่แบบรูปการออกแบบเชิงพฤติกรรมแต่มีลักษณะคล้ายกันจะมีการตรวจจับออกมาด้วย ซึ่งการแก้ปัญหานี้อาจต้องมีการใช้แผนภาพลำดับเพื่อใช้ในการพิจารณาประกอบกัน

และเมื่อเปรียบเทียบขั้นตอนวิธีและเครื่องมือที่ได้ออกแบบกับงานวิจัยที่เกี่ยวข้อง [14] จะเห็นได้ว่า ขั้นตอนวิธีและเครื่องมือที่ได้ออกแบบ และงานวิจัยที่เกี่ยวข้องสามารถตรวจจับแบบรูปเชิงพฤติกรรมได้จำนวนมากกว่างานวิจัยที่เกี่ยวข้อง มีข้อมูลนำเข้าที่สามารถใช้งานได้สะดวกกว่า เนื่องมาจากใช้แผนภาพคลาสซึ่งเป็นเครื่องมือที่นักพัฒนาใช้ในการออกแบบระบบก่อนที่จะเขียนรหัสต้นฉบับ และมีค่าสัญกรณ์โอใหญ่ (Big O) ที่มีค่าน้อยกว่า ซึ่งส่งผลต่อการทำงานของเครื่องมือตรวจจับแบบรูปการออกแบบ

6.2 ปัญหาและข้อจำกัดที่พบจากการวิจัย

1) ไฟล์ข้อมูลนำเข้าซึ่งเป็นไฟล์เอ็กซ์เอ็มไอใช้ได้เพียงเอ็กซ์เอ็มไอรุ่น 1.0 เท่านั้น
2) แบบรูปการออกแบบเชิงพฤติกรรมมีจำนวนทั้งสิ้น 11 แบบรูป แต่ในงานวิจัยนี้สามารถตรวจจับแบบรูปเชิงพฤติกรรมได้เพียง 7 แบบรูป
3) แบบรูปการออกแบบพฤติกรรมหลายแบบรูปมีลักษณะที่คล้ายคลึงกับแบบรูปการออกแบบในหมวดอื่นๆ ทำให้บางครั้งขั้นตอนวิธีจำตรวจจับแบบรูปการออกแบบในหมวดอื่นๆและแสดงว่าเป็นแบบรูปการออกแบบพฤติกรรม

6.3 ข้อเสนอแนะ

งานวิจัยนี้สามารถพัฒนาเพิ่มเติมในหลายด้าน ดังนี้

1) พัฒนาให้สามารถค้นหาแบบรูปการออกแบบให้ครอบคลุมแบบรูปการออกแบบของ GOF ทั้งหมดซึ่งมีจำนวนทั้งสิ้น 23 แบบรูปซึ่งครอบคลุมทั้งแบบรูปเชิงโครงสร้าง แบบรูปเชิงพฤติกรรม และแบบรูปการสร้างวัตถุ

2) พัฒนาขั้นตอนวิธีให้สามารถวิเคราะห์รายละเอียดต่างๆซึ่งเป็นคุณสมบัติของคลาส ซึ่งประกอบด้วยลักษณะประจำ และการดำเนินการเพื่อนำมาประกอบในการวิเคราะห์ให้แม่นยำมากขึ้น

3) ประยุกต์ใช้ขั้นตอนวิธีในการค้นหาจากแผนภาพคลาสร่วมกับแผนภาพลำดับ เพื่อความแม่นยำที่มากขึ้น

4) พัฒนาขั้นตอนวิธีในการค้นหาโหนดหลัก โดยพัฒนาให้ค้นหาโหนดหลักที่มีความใกล้เคียงกันแล้วทำการค้นหารายละเอียด เนื่องจากโหนดหลักของแบบรูปบางประเภทมีลักษณะที่ใกล้เคียงกัน

รายการอ้างอิง

[1] Nutchanart Satvinij, and Wiwat Vatanawoo. Detection of Behavioral Design Patterns.Proceedings of the 15th International Annual Symposium on Computational Science and Engineering (ANSCSE2011), pp. 569-574. March 30 – April 2, 2011. Bangkok University Press, 2011.

[2] Booch, G., Rumbaugh, J., and Jacobson, I., The Unified Modeling Language User Guide. USA: Addison Wesley, 1998

[3] Weilkiens, T. Systems engineering with SysML/UML: modeling, analysis, design. USA: Morgan Kaufmann, 2007.

[4] Miles, R., and Hamilton, K. Learning UML2.0. USA: O'Reilly, 2006.

[5] O'Docherty, M. Object-oriented analysis and design: understanding system development with UML 2.0. USA: John Wiley & Sons,2005.

[6] Holzner, S. Design Patterns For Dummies. USA: Wiley Publishing, 2006.

[7] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns Elements of Reusable Object-Oriented Software. Professional Computing Series. USA: Addison-Wesley,1994.

[7] Shalloway, A., and Trott, J.R. Design Pattern Explained: A New Perspective on Object-Oriented Design. Software Patterns Series. USA: Addison-Wesley Professional, 2000.

[9] Shen W., and Kim, D., ICER: A Tool for Finding Errors in a UML Model [Online].2008.Availablefrom : http://www.cs.wmich.edu/~OODA/patterns/ [2010, 12 December].

[10] Object Oriented Design. Design Patterns [Online]. 2009. Available from: http://www.oodesign.com/ [2010, 27 November].

[11] Sourcemaking. Design Patterns [Online]. 2009. Available from: http://sourcemaking.com/design_patterns [2010, 27 November].

[12] javacamp. About Design Pattern [Online]. 2011. Available from: http://www.javacamp.org/designPattern/ [2012, 16 January].

[13] Huston, V. Design Patterns [Online]. 2003. Available from: http://www.vincehuston.org/dp [2011, 10 January].

[14] Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., and Halkidis, S.T. Design Pattern Detection Using Similarity Scoring. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 32, NO. 11. pp. 896-909. IEEE Press Piscataway, NJ, USA, 2006.

[15] Dong, J., Sun, Y., and Zhao, Y. Design Pattern Detection by Template Matching. Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC'08), Fortaleza, Ceará, Brazil, March 16-20, 2008.

[16] Harold, E. R. XML 1.1 Bible, 3rd Edition. USA: Wiley Publishing, 2004

[17] OASIS. XML Metadata Interchange (XMI) [Online].2002. Available from: http://xml.coverpages.org/xmi.html [2012, 2 April]

[18] Heuzeroth, D., Holl, T., Högström, G., and Löwe, W. Automatic Design Pattern Detection. Proceedings of the 11th IEEE International Workshop on Program Comprehension (IWPC'03), IEEE Computer Society Washington, DC, USA, 2003.

[19] Tsantalis, N. Design Pattern detection using Similarity [Online].2006. Available from:http://java.uom.gr/~nikos/pattern-detection.html[2010,27 November]

[20] Madhusudanan, A. Design Your Soccer Engine, and Learn How To Apply Design Patterns (Observer, Decorator, Strategy, and Builder Patterns) [Online].2005. Available from: http://www.codeproject.com/Articles /12194/Design-Your-Soccer-Engine-and-Learn-How-To-Apply-D [2010, 7 December].

[21] JHotDraw. JHotDraw as Open-Source Project [Online]. 2007. Available from: http://www.jhotdraw.org/ [2010, 7 December].

[22] Kaiser, W. Become a programming Picasso with JHotDraw. [Online]. 2001. Available from: http://www.javaworld.com/javaworld/jw-02-2001/jw-0216-jhotdraw.html [2010, 27 November].

[23] McCurdy, N.J., and Griswold, W.G. A Systems Architecture for Ubiquitous Video. Proceedings of the 3rd international conference on Mobile systems, applications, and services(MobiSys '05), pp. 1-14. ACM New York, NY, USA, 2005.

[24] Riehle, D. Case Study: The JHotDraw Framework [Online]. 2007. Available from: http://www.riehle.org/computer-science/research/dissertation/chapter-8.html. [2010, 7 December].

[25] Chatzigeorgiou, A., Tsantalis, N., and Stephanides, G. Application of Graph Theory to OO Software Engineering. the 2nd International Workshop on Interdisciplinary Software Engineering Research (WISER'2006), Shanghai, China, May 20, 2006.

[26] Jolita Savolskyte. Project Work . Review of the JHotDraw framework [Online]. Available from: www.sts.tu-harburg.de/pw-and-m-theses/.../savo04.pdf [2010, 27 November].

[27] Maioriello, J. A Survey of Common Design Patterns [Online]. 2002. Available from: http://www.developer.com/design/article.php/10925_1502691_4/A-Survey-of-Common-Design-Patterns.htm [2010, 27 November].

ภาคผนวก

ภาคผนวก ก
ผลลัพธ์ที่ได้จากการทดสอบขั้นตอนวิธีและเครื่องมือตรวจจับแบบรูปการออกแบบของโปรแกรม JHotDraw

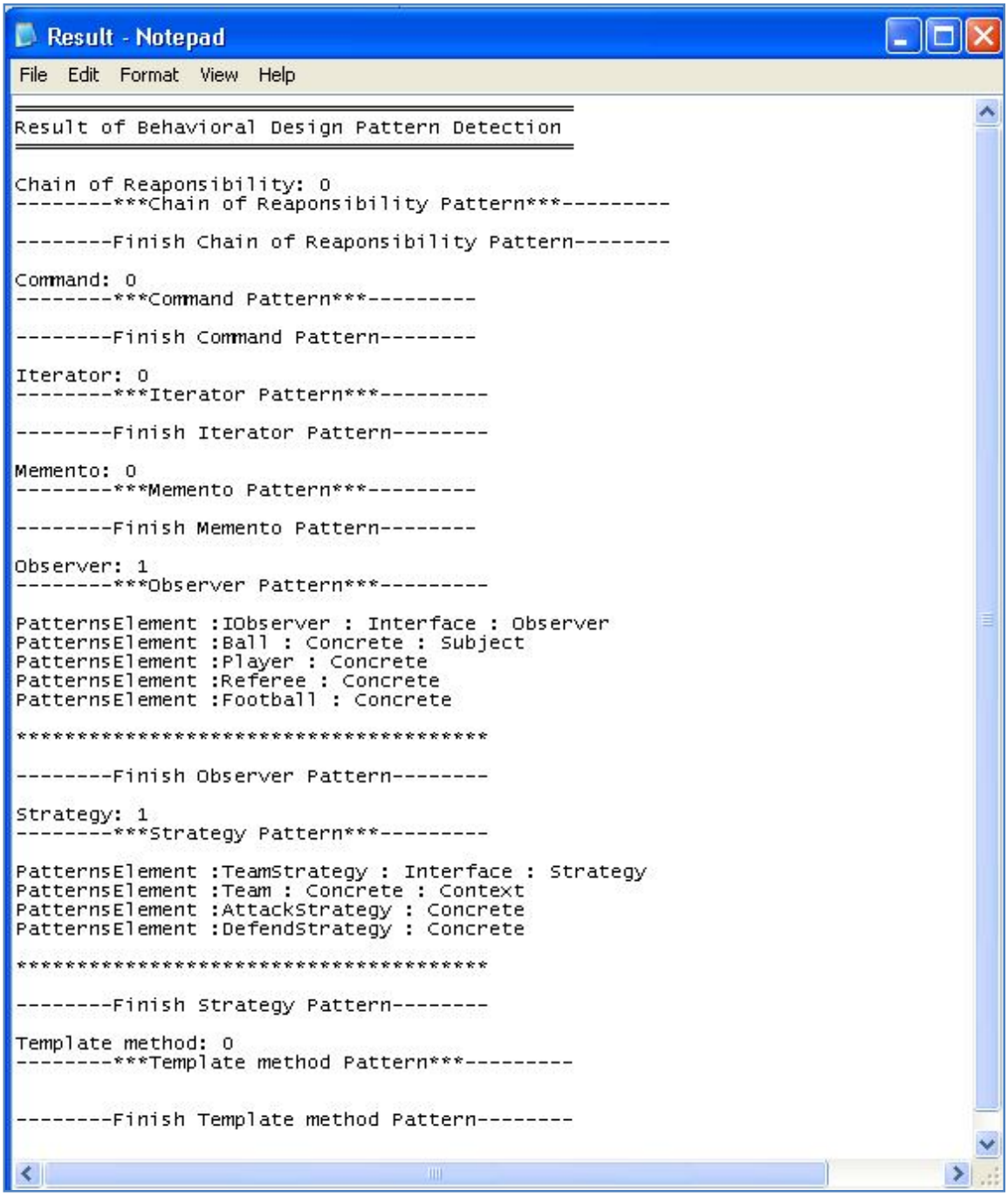
ผลลัพธ์ที่ได้จากการทดสอบขั้นตอนวิธีและเครื่องมือตรวจจับแบบรูปการออกแบบของโปรแกรม JhotDraw

ภาพที่ ก.1 ผลลัพธ์การตรวจจับแบบรูปการออกแบบโปรแกรม JHotDraw

ภาคผนวก ข

ผลลัพธ์ที่ได้จากการทดสอบขั้นตอนวิธีและเครื่องมือตรวจจับแบบรูปการออกแบบของ Soccer Engine

ผลลัพธ์ที่ได้จากการทดสอบขั้นตอนวิธีและเครื่องมือตรวจจับแบบรูปการออกแบบของ Soccer Engine



ภาพที่ ข.1 ผลลัพธ์การตรวจจับแบบรูปการออกแบบ Soccer Engine

ภาคผนวก ค

ผลลัพธ์ที่ได้จากการทดสอบขั้นตอนวิธีและเครื่องมือตรวจจับแบบรูปการออกแบบของโปรแกรม Ubiquitous Video

ผลลัพธ์ที่ได้จากการทดสอบขั้นตอนวิธีและเครื่องมือตรวจจับแบบรูปการออกแบบของโปรแกรม Ubiquitous Video

ภาพที่ ค.1 ผลลัพธ์การตรวจจับแบบรูปการออกแบบโปรแกรม Ubiquitous Video

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวนุชนาถ สัตย์วินิจ เกิดเมื่อวันที่ 28 มิถุนายน พ.ศ. 2528 ที่จังหวัดราชบุรี สำเร็จการศึกษาระดับปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ เกียรตินิยมอันดับหนึ่ง จากคณะวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์ ในปีการศึกษา 2549 และเข้าทำงานเป็นอาจารย์ประจำภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์ ในปีการศึกษา 2550 จากนั้นเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ณ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552 ปัจจุบันเป็นอาจารย์ประจำภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์