บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

กระบวนการพัฒนาซอฟต์แวร์ปัจจุบันประกอบด้วยขั้นตอนต่างๆหลายขั้นตอน ซึ่งขั้นตอนที่สำคัญอย่างหนึ่งก็คือ ขั้นตอนของการวิเคราะห์ความต้องการของระบบ และนำผลที่ได้จากการวิเคราะห์มาออกแบบระบบ ซึ่งปัจจุบัน การพัฒนาซอฟต์แวร์โดยส่วนใหญ่จะมุ่งไปในแนวทางการพัฒนาซอฟต์แวร์เชิงวัตถุ (Object-Oriented Software Development) โดยแนวความคิดหนึ่งของการพัฒนาซอฟต์แวร์เชิงวัตถุนั้น ก็คือการนำเสนอส่วนประกอบของซอฟต์แวร์ให้อยู่ในรูปของวัตถุสำหรับโดเมนที่สนใจและแสดงความสัมพันธ์ระหว่างวัตถุ ซึ่งโดยทั่วไปมักจะเรียกว่า อ็อบเจกต์โมเดล (Object Model) นั่นหมายความว่าการที่จะพัฒนาซอฟต์แวร์ขึ้นมาได้นั้นจะต้องทำการสร้างโมเดลที่สอดคล้องกับความเป็นจริงของโดเมนนั้นๆ เมื่อเข้าสู่ขั้นตอนของการออกแบบอ็อบเจกต์โมเดล ผู้ออกแบบมักพบกับปัญหาในการออกแบบอยู่บ่อยครั้ง อีกทั้งปัญหาเหล่านั้นก็มักเกิดขึ้นซ้ำบ่อยๆ E. Gamma และคณะ[1] จึงได้กำหนดดีไซน์แพทเทิร์น (Design Pattern) เพื่อแก้ปัญหาการออกแบบ นักออกแบบสามารถนำแพทเทิร์นมาใช้เพื่อแก้ปัญหาที่คล้ายคลึงกันได้

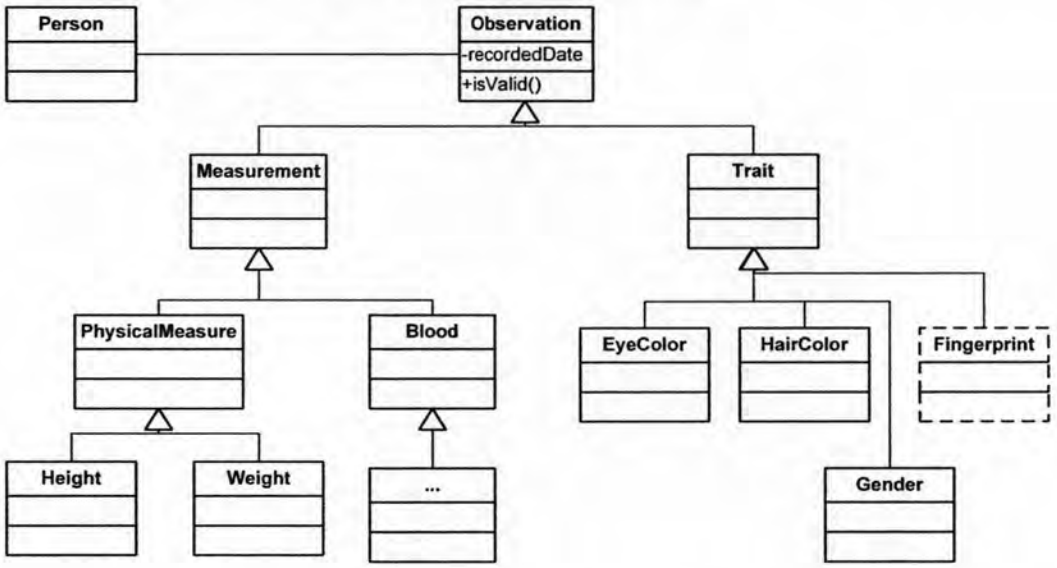
อย่างไรก็ตาม ในปัจจุบันความต้องการของระบบนั้นมักเกิดการเปลี่ยนแปลงอยู่บ่อยครั้ง ทำให้ระบบที่สร้างขึ้นมาไม่สามารถรองรับความต้องการที่เปลี่ยนแปลงได้ เนื่องจากอ็อบเจกต์โมเดล เป็นโครงสร้างที่กำหนดขึ้นมาแบบคงที่ (Static Model) การเปลี่ยนแปลงโครงสร้างของอ็อบเจกต์โมเดลจะมีผลกระทบต่อระบบ ดังนั้น จึงได้มีการนำเสนออ็อบเจกต์โมเดลที่สามารถรองรับการเปลี่ยนแปลงที่เกิดขึ้น ซึ่งหลายงานวิจัยได้เรียกว่าเอโอเอ็ม (Adaptive Object Model; AOM)

เอโอเอ็ม [2] เป็นแนวทางในการออกแบบให้โมเดลของระบบสามารถรองรับความเปลี่ยนแปลงที่เกิดขึ้นได้โดยที่ไม่ต้องทำการแก้ไขโปรแกรม แม้กระทั่งตอนรันไทม์ (Run-time) ในการออกแบบอ็อบเจกต์โมเดลให้สามารถปรับเปลี่ยนได้นั้น ทำได้โดยการนำเอาดีไซน์แพทเทิร์น หลายๆ ประเภทมาประกอบกัน เช่น ไทป์อ็อบเจกต์ (Type-Object) [3], พรอบเพอร์ตี(Property) [4], สตราทิจิ (Strategy) [1], อินเตอร์พรีทเตอร์ (Interpreter) [1], คอมโพสิท (Composite) [1], แอ็คเคาน์ทะบิลิตี (Accountability) [5] เป็นต้น และนำเสนอในรูปแบบของ Meta-Architecture การเปลี่ยนแปลงอ็อบเจกต์โมเดลทำได้โดยการแก้ไขค่าที่กำหนดไว้ และสามารถทำได้โดยผู้เชี่ยวชาญในโดเมนนั้นๆ (Domain Expert) ซึ่งไม่จำเป็นต้องเป็นนักพัฒนาซอฟต์แวร์

ถึงแม้ว่าเอโอเอ็มเป็นแนวทางที่จะเข้ามาช่วยในการแก้ปัญหาความต้องการที่เปลี่ยนแปลงบ่อยได้ในระดับหนึ่ง แต่ก็ยังมีข้อเสีย คือ โมเดลเอโอเอ็มไม่ได้สะท้อนถึงความหมายหรือสาระสำคัญ

ของธุรกิจโดยตรง (Business Abstraction) ซึ่งทำให้เข้าใจได้ยาก (Understandability) ทำให้ทีมพัฒนาหรือโปรแกรมเมอร์ต้องใช้เวลานาน ในการพัฒนา การบำรุงรักษา (Maintenance) การขยายระบบ (Extension)

ตัวอย่างของแผนภาพคลาสของระบบเฝ้าสังเกต (Observation Model) แสดงในรูปที่ 1.1 ถูกออกแบบเพื่อบันทึกข้อมูลการเฝ้าสังเกตและตรวจสอบเกี่ยวกับรูปร่างลักษณะของผู้อพยพ จะเห็นได้ว่าคลาสแต่ละคลาสเป็นตัวแทนของวัตถุในโลกของความเป็นจริงอย่างชัดเจน ซึ่งจะทำให้นักพัฒนาสามารถทำความเข้าใจในเรื่องนั้นๆได้ง่าย แต่ในกรณีที่ต้องการเพิ่มคลาสใหม่ เช่น คลาส Fingerprint โปรแกรมจะต้องถูกนำมาคอมไพล์และติดตั้งใหม่เสมอ ต้องอาศัยนักพัฒนาในการทำงาน ทำให้ไม่สามารถตอบสนองต่อความต้องการที่เปลี่ยนแปลงอย่างรวดเร็วได้

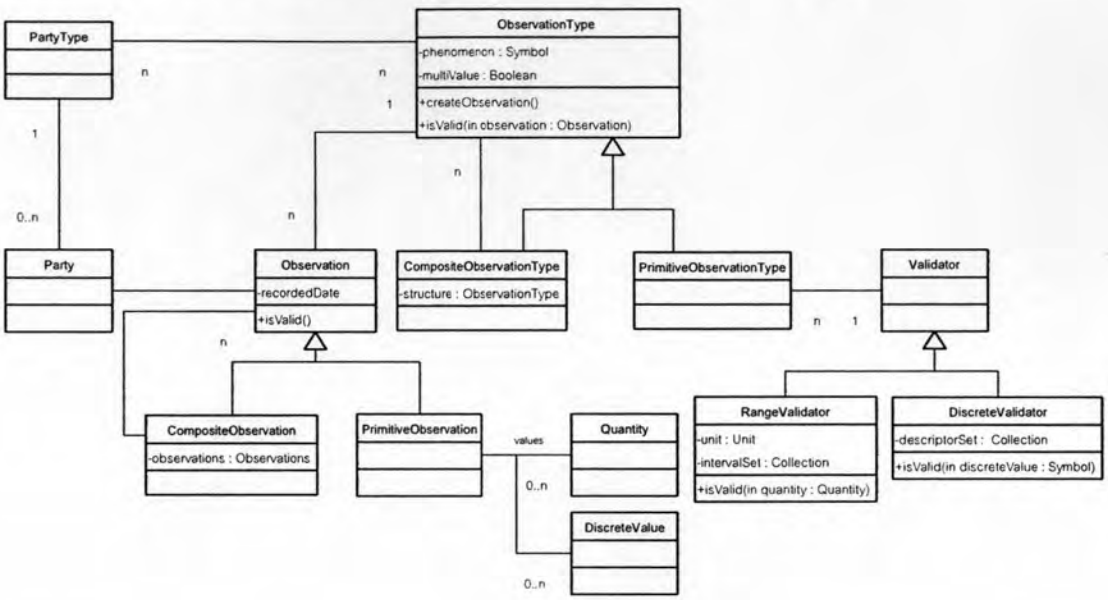


รูปที่ 1.1 แผนภาพคลาสของระบบเฝ้าสังเกต [4]

เมื่อทำการออกแบบแผนภาพคลาสของระบบเฝ้าสังเกตใหม่โดยใช้เอโอเอ็ม ดังแสดงในรูปที่ 1.2 ทำให้ระบบมีความยืดหยุ่นมากขึ้นคือ สามารถเพิ่มประเภทของข้อมูลใหม่ได้ เพิ่มกฎเกณฑ์ในการตรวจสอบได้ โดยที่ไม่ต้องทำการคอมไพล์โปรแกรมใหม่ เช่น การเพิ่มอินสแตนท์ของคลาส ObservationType ที่เป็นตัวแทนของคลาส Fingerprint เดิม แทนที่จะต้องเพิ่มคลาส Fingerprint เป็นคลาสใหม่ ซึ่งจะทำให้โปรแกรมสามารถรองรับประเภทของข้อมูลเฝ้าสังเกตใหม่ๆได้ทันที

อย่างไรก็ตามคลาสส่วนใหญ่ของเอโอเอ็มไม่ได้เป็นตัวแทนที่สื่อความหมายของวัตถุโดยตรง คลาสและความสัมพันธ์ระหว่างคลาสที่แสดง ถูกกำหนดโดยการนำเอาดีไซน์แพทเทิร์นแต่ละแบบมาใช้งานร่วมกัน เช่น ในรูปที่1.2 นั้นเป็นการนำเอา ไทป์อ็อบเจกต์และพรอบเพอร์ตีแพทเทิร์นมาใช้ร่วมกัน (ได้แก่ คลาส Party, PartyType, ObservationType และ Observation) เพื่อให้สามารถเพิ่มลดประเภทของข้อมูลการเฝ้าสังเกตของผู้อพยพ และนำเอาสตราทิจิแพทเทิร์น (คลาส Validator,

RangeValidator, DiscreteValidator) เพื่อให้สามารถเพิ่มลดวิธีการตรวจสอบข้อมูลการเฝ้าสังเกตแต่ละแบบได้ นอกจากนั้นยังมีแอ็คเคาน์ทะบิลิตีแพทเทิร์นที่จะนำมาใช้ในการปรับเปลี่ยนความสัมพันธ์ระหว่างอ็อบเจกต์ ซึ่งทำให้เอโอเอ็มเข้าใจได้ยากกว่าอ็อบเจกต์โมเดลแบบเดิม



รูปที่ 1.2 แผนภาพคลาสของระบบเฝ้าสังเกตแบบเอโอเอ็ม [4]

งานวิจัยนี้จึงจะนำเสนอวิธีในการปรับปรุงให้การออกแบบและพัฒนาซอฟต์แวร์แบบเอโอเอ็มสามารถทำได้ง่ายขึ้น โดยใช้เทคนิคการนำเสนอโครงสร้างของดีไซน์แพทเทิร์นด้วยยูนิตโมเดล (Unit Model) [6] มาประยุกต์ใช้ในการออกแบบเอโอเอ็ม

1.2 วัตถุประสงค์ของการวิจัย

เพื่อปรับปรุงวิธีการออกแบบอ็อบเจกต์โมเดลที่ปรับเปลี่ยนได้หรือเอโอเอ็ม โดยนำเสนอวิธีการและใช้โมเดลที่สามารถทำความเข้าใจได้ง่ายขึ้นต่อการพัฒนาระบบ

1.3 ขอบเขตการวิจัย

1. งานวิจัยนี้ปรับปรุงการออกแบบเอโอเอ็มด้วยการนำเสนอวิธีการแปลงดีไซน์แพทเทิร์นที่ใช้ในเอโอเอ็ม 3 แพทเทิร์น ซึ่งประกอบด้วย ไทป์อ็อบเจกต์, พรอบเพอร์ตี และ แอ็คเคาน์ทะบิลิตี ให้อยู่ในรูปของยูนิตโมเดล

2. ออกแบบและพัฒนาระบบเอโอเอ็มทั้งแบบดั้งเดิมและแบบยูนิตโมเดล ด้วยการพัฒนาโปรแกรมจำนวนอย่างน้อย 3 โปรแกรมที่ต่างโดเมนกัน นั่นคือ มีอย่างน้อย 3 โปรแกรมสำหรับเอโอเอ็มแบบเดิม และอย่างน้อย 3 โปรแกรมสำหรับเอโอเอ็มแบบยูนิต
3. เปรียบเทียบผลการออกแบบโมเดล โดยการใช้มาตรวัดสำหรับวัดค่าความซับซ้อนของโครงสร้างสำหรับโมเดลทั้งสองแบบเพื่อเปรียบเทียบกัน
4. การวัดค่าความซับซ้อนของโครงสร้างจะวัดจาก แผนภาพคลาส แผนภาพแพ็กเกจ และยูนิตโมเดลเท่านั้น
5. ทำการสร้างเครื่องมือที่ช่วยในการออกแบบโปรแกรมเอโอเอ็มด้วยยูนิตโมเดล
6. งานวิจัยนี้ไม่ได้ทำการสร้างเครื่องมือหรือโปรแกรมในการแปลงจากเอโอเอ็มไปเป็นยูนิตโมเดล

1.4 ขั้นตอนการวิจัย

1. ศึกษาดีไซน์แพทเทิร์นที่ใช้ในการแก้ปัญหาของการพัฒนาซอฟต์แวร์เชิงวัตถุ
2. ศึกษาเทคนิคในการออกแบบอ็อบเจกต์โมเดลที่ปรับเปลี่ยนได้หรือเอโอเอ็ม
3. ศึกษาโมเดลเอโอเอ็ม ที่มีอยู่ เพื่อศึกษาปัญหาที่เกิดขึ้นในการออกแบบและพัฒนา
4. ศึกษาหลักการและแนวคิดในการพัฒนาซอฟต์แวร์ที่ใช้ยูนิตโมเดลในการออกแบบ
5. ออกแบบโมเดลที่ใช้การนำเสนอใหม่ โดยใช้เทคนิคในการดึงเอาข้อมูลของโครงสร้างที่อยู่ในดีไซน์แพทเทิร์นมานำเสนอในรูปแบบของยูนิตโมเดล พร้อมทั้งประยุกต์ใช้เทคนิคอื่นเพื่อปรับให้มีความเหมาะสมต่อการทำความเข้าใจและพัฒนา
6. ทำการออกแบบและพัฒนาโปรแกรมทดสอบ
7. วิเคราะห์และประเมินผลเปรียบเทียบโมเดลใหม่ที่ได้โดยการทดลอง
8. สรุปผลการวิจัย และจัดทำวิทยานิพนธ์

1.5 ประโยชน์ที่จะได้รับ

ได้แนวทางในการปรับปรุงอ็อบเจกต์โมเดลที่ปรับเปลี่ยนได้หรือเอโอเอ็มด้วยยูนิตโมเดล ซึ่งจะช่วยให้นักพัฒนาซอฟต์แวร์สามารถทำความเข้าใจโมเดลได้ง่ายขึ้น ทำให้ช่วยลดทรัพยากรที่ต้องใช้ในการบำรุงรักษาได้