บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

นับตั้งแต่ช่วงปลายทศวรรษที่ 1980 เป็นต้นมา โปรแกรมคอมพิวเตอร์ส่วนใหญ่ถูกเขียนขึ้นโดยใช้กรอบความคิดที่เรียกว่า วิธีการเขียนโปรแกรมเชิงโครงสร้าง (Structured Programming) โปรแกรมที่ถูกเขียนด้วยกรอบความคิดนี้มีลักษณะเด่นหลายประการ กล่าวคือ (1) โปรแกรมจะถูกแบ่งออกเป็นส่วนๆ โดยที่ส่วนประกอบเหล่านั้นจะทำงานภายในขอบเขตที่ชัดเจน และ (2) ลำดับการทำงานภายในส่วนประกอบไม่กระโดดข้ามไปมา (Non-spaghetti Code)

แม้วิธีการเขียนโปรแกรมเชิงโครงสร้าง จะมีผลดีต่อขั้นตอนการออกแบบและการพัฒนาโปรแกรมเป็นอย่างมาก แต่การเขียนโปรแกรมแบบนี้ก็สร้างปัญหาให้กับขั้นตอนการซ่อมบำรุงโปรแกรมไม่น้อยเช่นกัน ปัญหาที่เกิดขึ้นบ่อยๆ เมื่อซ่อมบำรุงโปรแกรมที่เขียนด้วยวิธีการดังกล่าว ได้แก่ (1) ปัญหาผลกระทบแบบลูกคลื่น (Ripple Effect)[1] (2) ปัญหาการปรับโปรแกรมให้เข้ากับเทคโนโลยีสมัยใหม่ (3) ปัญหาค่าซ่อมบำรุงที่เพิ่มขึ้นมากในช่วงปลายการใช้งาน ฯลฯ ด้วยข้อเสียที่กล่าวมาทั้งหมดนี้ จึงทำให้หน่วยงานหลายแห่งตัดสินใจทิ้งโปรแกรมเหล่านี้ไปทั้งๆ ที่ยังใช้งานได้ดี

หากพิจารณาปัญหาโดยละเอียดจะพบว่า ต้นเหตุที่แท้จริงเกิดจากการที่วิธีการเขียนโปรแกรมเชิงโครงสร้างไม่มีหลักการที่ดีพอสำหรับรองรับการซ่อมบำรุงโปรแกรมขนาดใหญ่ เทียบกับวิธีการเขียนสมัยใหม่อย่างเช่นวิธีการเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming) ซึ่งหลักการอย่างเช่นระบบอินเทอร์เฟซ (Interface) แล้วจะเห็นได้ว่าวิธีการเขียนโปรแกรมเชิงวัตถุมีกลไกที่เอื้อต่อการซ่อมบำรุงโปรแกรมมากกว่า ด้วยเหตุนี้ แนวคิดเรื่องปรับย้ายโครงสร้างโปรแกรมจากเชิงโครงสร้างให้กลายมาเป็นเชิงวัตถุจึงเริ่มเป็นที่สนใจ

การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ (Candidate Object Identification) เป็นวิธีการอย่างหนึ่ง ที่ใช้อำนวยความสะดวกการปรับย้ายโครงสร้างโปรแกรมจากแบบเชิงโครงสร้างให้เป็นแบบเชิงวัตถุ การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้มีขั้นตอนการทำงาน คือ (1) คัดแยกส่วนประกอบเดิม (Artifact) ออกจากโปรแกรมเก่า (2) หาความสัมพันธ์ระหว่างส่วนประกอบเดิม (3) จัดเรียงส่วนประกอบเดิมเป็นกลุ่ม และ (4) เลือกกลุ่มที่มีความหมาย (Semantic) มาประกอบเป็นโครงสร้างที่เรียกว่า "วัตถุซอฟต์แวร์ที่เป็นไปได้" (Candidate Object)

การนำวิธีการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้มาช่วยในงานปรับโครงสร้าง ทำให้เกิดผลดีดังนี้

1) ช่วยป้องกันไม่ให้แบบโปรแกรม (Design) ลดคุณภาพลง ขณะทำการปรับย้ายโครงสร้าง [1]
2) ช่วยอำนวยความสะดวกการศึกษาการสร้างและการแก้ไขข้อมูล รวมถึงลักษณะการออกแบบและวิธีการเขียนรหัสคำสั่งในโปรแกรมเก่า [1]
3) ช่วยอำนวยความสะดวกการกู้คืนแบบโปรแกรม (Recovering Software Representation) [2]
4) ช่วยอำนวยความสะดวกการย่อส่วนโปรแกรม (Downsizing) [3]

---

[1] ผลกระทบลูกคลื่น คือ ผลข้างเคียงที่เกิดขึ้นไม่รู้จบจากการแก้ไขคำสั่งในโปรแกรม

5) ช่วยอำนวยความสะดวกการนำส่วนประกอบเก่ากลับมาเวียนใช้ (Reusing Legacy Component) [4]

6) ช่วยในการซ่อมแซมส่วนประกอบที่เสียหาย เนื่องจากถูกซ่อมบำรุงอย่างไม่ถูกต้อง [5]

ในปัจจุบัน การวิจัยเกี่ยวกับการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้เน้นไปที่ การศึกษาผลการนำวิธีการวิเคราะห์ข้อมูลชนิดต่างๆ มาประยุกต์ใช้ วิธีการวิเคราะห์ข้อมูลถือเป็นหัวใจสำคัญของการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ เพราะวิธีการวิเคราะห์ข้อมูลช่วยให้การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้เป็นไปด้วยความสะดวกรวดเร็ว การระบุวัตถุซอฟต์แวร์ในโครงการหนึ่งๆ จะประสบความสำเร็จหรือไม่ ขึ้นอยู่กับว่าวิศวกรซอฟต์แวร์ในโครงการนั้นๆ สามารถเลือกวิธีการวิเคราะห์ข้อมูลมาประยุกต์ใช้ได้อย่างเหมาะสมเพียงใด วิธีการวิเคราะห์ข้อมูลที่ได้รับความสนใจมากที่สุดในปัจจุบันคือ วิธีการที่เรียกว่า วิธีการวิเคราะห์คอนเซ็ปต์ (Concept Analysis)

วิธีการวิเคราะห์คอนเซ็ปต์เป็นวิธีการวิเคราะห์ข้อมูลชนิดหนึ่ง ใช้สำหรับจัดเรียงวัตถุที่มีคุณสมบัติตรงกันให้อยู่ในกลุ่มเดียวกัน เมื่อนำวิธีการนี้มาประยุกต์ใช้ในการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ พบว่ามีข้อดีเกิดขึ้นหลายประการคือ (1) สามารถจัดเรียงส่วนประกอบเดิมได้ถูกต้องและรวดเร็วมากขึ้น (2) มีรายละเอียดการจัดกลุ่มแสดงให้เห็นอย่างชัดเจน (3) แสดงผลลัพธ์ในแผนภาพต้นไม้ ซึ่งเป็นรูปแบบที่ศึกษาทำความเข้าใจได้ง่าย

อย่างไรก็ตาม การนำวิธีการวิเคราะห์คอนเซ็ปต์มาประยุกต์ใช้ก็มีข้อเสียอยู่ด้วยเช่นกัน ยกตัวอย่างเช่น (1) ไม่สามารถกลั่นกรองผลการทำงานได้ สารสนเทศที่ถูกส่งออกมาจึงมีทั้งที่เป็นประโยชน์และไม่เป็นประโยชน์ต่อการทำงาน (Information Overflow) [6] (2) ส่วนประกอบเดิมที่ถูกจัดให้อยู่ในคอนเซ็ปต์หนึ่ง อาจไปปรากฏอยู่ในคอนเซ็ปต์อีกอันหนึ่งได้ ผลลัพธ์ที่ได้จึงทับซ้อน (Concept Overlapping) และไม่สามารถนำมาใช้เป็นตัวแทนของกลุ่มข้อมูลได้ [7] และ (3) เมื่อนำไปใช้กับโปรแกรมที่มีโครงสร้างแบบผูกติด วิธีการนี้จะไม่สามารถแยกแยะส่วนประกอบเดิมได้อย่างถูกต้อง (จำเป็นต้องนำเทคนิคอื่นๆเข้ามาช่วยทำงาน) [8]

วิธีการจัดกลุ่มข้อมูลแบบลำดับชั้น (Clustering Analysis) เป็นวิธีการวิเคราะห์ข้อมูลอีกชนิดหนึ่งที่ได้รับความเชื่อถือและถูกนำไปประยุกต์ใช้ในงานวิจัยด้านต่างๆ อย่างกว้างขวาง ไม่ว่าจะเป็นงานวิจัยทางเศรษฐศาสตร์ ดาราศาสตร์ และชีววิทยา Wiggerts [9] กล่าวถึงจุดเด่นของวิธีการนี้ไว้ว่า มีความยืดหยุ่นในการทำงานสูง สามารถปรับเปลี่ยนส่วนประกอบต่างๆ ได้หลากหลาย จึงเหมาะที่จะนำไปใช้เป็นเครื่องมือจัดระเบียบซอฟต์แวร์ (Software Re-modularization)

Kuipers กับ Deursen [10] เคยนำวิธีการจัดกลุ่มข้อมูลแบบลำดับชั้นและวิธีการวิเคราะห์คอนเซ็ปต์ มาทดลองประยุกต์ใช้กับการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ในโปรแกรมภาษาโคบอล ผลการทดลองของพวกเขาแสดงให้เห็นว่า วิธีการจัดกลุ่มข้อมูลแบบลำดับชั้นทำงานได้ไม่ดีเท่ากับวิธีการวิเคราะห์คอนเซ็ปต์ อย่างไรก็ตาม แม้จะมีข้อสรุปเป็นเช่นนั้น แต่ก็อาจพิจารณาได้ว่าข้อสรุปดังกล่าวไม่สมบูรณ์เท่าที่ควร เพราะพวกเขาทำการศึกษาครั้งนั้นโดยไม่ได้ทดลองปรับเปลี่ยนส่วนประกอบอีกหลายอย่าง เช่น การนิยามคุณสมบัติข้อมูล การให้ค่าน้ำหนัก ฯลฯ ที่ล้วนแต่ช่วยให้วิธีการจัดกลุ่มข้อมูลฯ ทำงานดีขึ้นได้

วิทยานิพนธ์นี้มีจุดมุ่งหมายเพื่อออกแบบวิธีการและพัฒนาเครื่องมือระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ ซึ่งนำวิธีการจัดกลุ่มข้อมูลแบบลำดับชั้นมาประยุกต์ใช้ ร่วมกับการให้ค่าน้ำหนักแก่ความสัมพันธ์ที่เกิดจากส่วนข้อมูลสตรักท์ เพื่อช่วยให้การทำงานมีความถูกต้องมากขึ้น วิธีการและเครื่องมือที่พัฒนาขึ้นจะถูกทดสอบโดย นำไปทดลองใช้กับโปรแกรมต้นฉบับภาษาซีเชิงโครงสร้างที่แตกต่างกันสามโปรแกรม ร่วมกับวิธีการอื่นอีกสองวิธี คือ วิธีการที่

ประยุกต์จากวิธีการจัดกลุ่มข้อมูลแบบลำดับชั้น (แบบปกติ) และวิธีการที่ประยุกต์จากวิธีการวิเคราะห์คอนเซ็ปต์ หลังจากนั้นจึงนำผลลัพธ์ของวิธีการทั้งสามชนิดมาศึกษาเพื่อเปรียบเทียบการทำงาน

1.2 วัตถุประสงค์

เพื่อออกแบบวิธีการและพัฒนาเครื่องมือระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ ซึ่งประยุกต์มาจากวิธีการจัดกลุ่มข้อมูลแบบลำดับชั้น สำหรับใช้ในการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ในโปรแกรมต้นฉบับเชิงโครงสร้างที่เขียนด้วยภาษาซี

1.3 ขอบเขตการวิจัย

1. ออกแบบวิธีการและพัฒนาเครื่องมือที่พัฒนาขึ้นมีขอบเขตการทำงาน คือ (1) สามารถจัดกลุ่มฟังก์ชันที่มีความเกี่ยวข้องกันตามนิยามความสัมพันธ์ที่วิศวกรซอฟต์แวร์กำหนดได้ และ (2) สามารถเพิ่มหรือลดค่าน้ำหนักเพื่อช่วยให้การจัดกลุ่มฟังก์ชันให้มีความชัดเจนมากขึ้นได้
2. สามารถนำวิธีการและเครื่องมือที่พัฒนาขึ้น ไปใช้ได้กับโปรแกรมต้นฉบับเชิงโครงสร้างที่เขียนจากภาษาซีได้
3. สามารถนำวิธีการและเครื่องมือที่พัฒนาขึ้น ไปใช้ระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ในโปรแกรมต้นฉบับที่มีลักษณะการทำงานต่างกันอย่างน้อย 3 โปรแกรมได้อย่างถูกต้อง
4. วิธีการและเครื่องมือที่พัฒนาขึ้น ควรมีความสามารถในการทำงานไม่น้อยไปกว่าวิธีการและเครื่องมือที่ประยุกต์จากวิธีการจัดกลุ่มข้อมูลที่ไม่ได้ให้ค่าน้ำหนัก
5. ชุดพัฒนาซอฟต์แวร์ที่ใช้ ได้แก่ ชุดพัฒนาโปรแกรมภาษาจาวา รุ่น 1.3