การตรวจสอบความครบถ้วนของความต้องการในการเปลี่ยนแปลงซอฟต์แวร์

Checking for Completeness of Software Change Requirements

ปัทมาภรณ์ สายสิม (Pattamaporn Saisim)* และ ทวิตีย์ เสนีวงศ์ ณ อยุธยา (Twittie Senivongse)*

บทคัดย่อ

ปัญหาที่พบในการเปลี่ยนแปลงระบบซอฟต์แวร์โดยการพัฒนาระบบใหม่บนพื้นฐานของข้อกำหนดความต้องการของระบบเดิม คือ ฟังก์ชันงานหรือข้อมูลที่เคยมีในระบบเดิมและยังต้องการคงไว้ เกิดการตกหล่นไปในการออกแบบระบบใหม่ จึงอาจส่งผลให้การส่งมอบงานล่าช้ากว่ากำหนด เนื่องจากต้องแก้ไขให้ระบบใหม่มีข้อมูลหรือฟังก์ชันงานที่ครบถ้วนถูกต้องก่อน บทความนี้นำเสนอวิธีการตรวจสอบความครบถ้วนของความต้องการในการเปลี่ยนแปลงซอฟต์แวร์ตั้งแต่ในระยะเริ่มต้นของโครงการซอฟต์แวร์ใหม่ โดยการนำแผนภาพคลาสเชิงแนวคิดของระบบเดิมที่มีอยู่แล้วและของระบบที่ออกแบบใหม่มาเปรียบเทียบกัน อัลกอริทึมที่ใช้จะพิจารณาทั้งความคล้ายกันของโครงสร้างและความคล้ายกันในเชิงความหมายของการตั้งชื่อในแผนภาพ ผลลัพธ์ของการเปรียบเทียบจะทำให้เห็นถึงความคล้ายคลึงและความแตกต่างของแผนภาพทั้งสอง อันจะช่วยให้นักวิเคราะห์ระบบและผู้ใช้สามารถตรวจสอบความครบถ้วนของความต้องการในการเปลี่ยนแปลง สำหรับการออกแบบเชิงแนวคิดของระบบใหม่ที่จะพัฒนาต่อไปได้

คำสำคัญ: ความต้องการในการเปลี่ยนแปลงซอฟต์แวร์ แผนภาพคลาสเชิงแนวคิด เวิร์ดเน็ต

Abstract

An important problem in development of a new software system based on the requirements specification of the original system is that certain functions or data that are present in the original system and should be retained are missing from the new system. This problem can delay product delivery as the new system will need to be fixed to fulfill all functional and data requirements. This paper presents a method to check for completeness of software change requirements at the early stage of the new software project by comparing the conceptual class diagram of the original system with that of the new system to be developed. The algorithm considers structural similarity and semantic similarity of names in both diagrams. The comparison result can identify similarities and differences between the two diagrams, and hence can support system analysts and users in checking for completeness of change requirements for the conceptual design of the new system.

Keyword: Software Change Requirement, Conceptual Class Diagram, WordNet.

1. บทนำ

การเปลี่ยนแปลงระบบซอฟต์แวร์มีทั้งแบบที่เป็นการเพิ่มฟังก์ชันใหม่ในระบบเดิม และการพัฒนาระบบใหม่ทั้งหมดบนพื้นฐานของข้อกำหนดความต้องการของระบบเดิม โดยอาจมีการเพิ่ม ลด แก้ไขโปรแกรม หรือปรับขั้นตอนการทำงานบางอย่าง การพัฒนาระบบใหม่ทั้งหมดมีสาเหตุสำคัญ คือ เพื่อแก้ไขข้อบกพร่องของการออกแบบเดิม หรือ เพื่อเพิ่มฟังก์ชันหรือเปลี่ยนขั้นตอนการทำงานโดยการปรับปรุงโครงสร้างซอฟต์แวร์ใหม่ให้สอดคล้องกัน

* ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ในการพัฒนาซอฟต์แวร์ระบบใหม่บนพื้นฐานของข้อกำหนดความต้องการของระบบเดิม นักวิเคราะห์ระบบจะทำการรวบรวมความต้องการจากผู้ใช้ใหม่ ร่วมกับการศึกษาข้อกำหนดความต้องการที่มีอยู่ ปัญหาที่เกิดขึ้นได้คือ ระบบใหม่ที่พัฒนาอาจมีฟังก์ชันหรือข้อมูลไม่ครบตามที่เคยมีในระบบเดิมทั้งๆ ที่เป็นฟังก์ชันหรือข้อมูลที่ต้องการคงไว้ สาเหตุเกิดจากผู้ใช้ซึ่งเป็นผู้ให้ความต้องการซอฟต์แวร์ของระบบใหม่ อาจหลงลืมและไม่ได้แจ้งให้นักวิเคราะห์ระบบใหม่ทราบ หรือผู้ใช้ไม่ทราบว่ามีข้อมูลหรือฟังก์ชันนี้ในระบบ ดังนั้นจึงส่งผลให้มีการใช้เวลาและทรัพยากรในการพัฒนาระบบมากเกินไป เนื่องจากต้องมีการเพิ่มหรือแก้ไขระบบในภายหลัง จากปัญหาดังกล่าวจึงจำเป็นต้องมีการตรวจสอบความครบถ้วนของความต้องการในการเปลี่ยนแปลงซอฟต์แวร์จากระบบเดิมมาเป็นระบบใหม่ เนื่องจากการทราบถึงความต้องการที่ขาดหายไปตั้งแต่ช่วงของการวิเคราะห์ความต้องการจะทำให้นักวิเคราะห์ระบบสามารถแก้ไขได้ทันท่วงที ก่อนที่จะมีการออกแบบระบบในขั้นต่อไป

วิธีการที่จะนำเสนอคือการนำแผนภาพคลาสเชิงแนวคิด (Conceptual Class Diagram) ของระบบเดิมที่มีอยู่แล้วและแผนภาพคลาสเชิงแนวคิดของระบบใหม่ที่ได้จากการวิเคราะห์ความต้องการมาเปรียบเทียบกัน เนื่องจากในการพัฒนาซอฟต์แวร์ในลักษณะนี้เป็นการพัฒนาบนพื้นฐานของข้อกำหนดความต้องการเดิม แต่อาจมีการเพิ่ม ลด หรือแก้ไขการทำงานบางอย่าง ดังนั้นหากนำแผนภาพคลาสเชิงแนวคิดของระบบเดิมและระบบใหม่มาเปรียบเทียบกัน ผลลัพธ์ที่ได้จะทำให้เห็นถึงความแตกต่างของแผนภาพทั้งสอง อันจะนำไปสู่การตรวจสอบความครบถ้วนของความต้องการของระบบใหม่ได้ เช่น ฟังก์ชันหรือข้อมูลที่มีในระบบเดิมแต่ไม่มีในระบบใหม่นั้นเป็นส่วนที่ผู้ใช้ต้องการลบออกจากระบบใหม่จริงหรือไม่ หรือฟังก์ชันหรือข้อมูลที่ไม่มีในระบบเดิมแต่ถูกเพิ่มหรือเปลี่ยนแปลงเข้ามาในระบบใหม่นั้นเป็นส่วนที่ผู้ใช้ต้องการเพิ่มหรือเปลี่ยนแปลงในระบบใหม่จริงหรือไม่ อัลกอริทึมของผู้วิจัยซึ่งใช้ในการเปรียบเทียบแผนภาพคลาสเชิงแนวคิดมีชื่อว่า S-UMLDiff โดยเป็นการปรับจากอัลกอริทึม UMLDiff [1] ซึ่งใช้ในการพิจารณาความคล้ายกันของโครงสร้างของสองแผนภาพคลาส แต่ผู้วิจัยได้ปรับให้มีการพิจารณาความหมายของการตั้งชื่อในแผนภาพด้วยโดยอ้างอิงจากความสัมพันธ์ของคำศัพท์ในเวิร์ดเน็ต (WordNet) [2]

ในส่วนที่ 2 ของบทความจะกล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง ส่วนที่ 3 เป็นการนำเสนอวิธีการเปรียบเทียบแผนภาพคลาสเชิงแนวคิด ส่วนที่ 4 เป็นการอธิบายผลลัพธ์ที่ได้จากการเปรียบเทียบ และส่วนที่ 5 เป็นบทสรุป

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในกระบวนการวิเคราะห์ความต้องการซอฟต์แวร์ มีการใช้แผนภาพคลาสของยูเอ็มแอล [3] เพื่อแสดงถึงส่วนประกอบหรือคลาสในระบบ ข้อมูล ฟังก์ชันการทำงาน และความสัมพันธ์ระหว่างคลาสเหล่านั้น การออกแบบแผนภาพคลาสในกระบวนการนี้เป็นการออกแบบเพียงในระดับแนวคิดหรือเรียกว่าแผนภาพคลาสเชิงแนวคิด จึงยังไม่มีรายละเอียดในเชิงตรรกะการเขียนโปรแกรม เช่น จะระบุเพียงคลาส ลักษณะประจำ (Attribute) การดำเนินการ (Operation) และความสัมพันธ์ระหว่างคลาส โดยยังไม่ระบุประเภทข้อมูล (Data Type) ของลักษณะประจำ ความสามารถในการมองเห็น (Visibility) หรือพารามิเตอร์ของการดำเนินการ เป็นต้น งานวิจัยอื่นหลายงานได้เสนอวิธีการที่แตกต่างกันในการเปรียบเทียบแผนภาพคลาส เพื่อนำไปใช้ในวัตถุประสงค์ที่แตกต่างกัน ผู้วิจัยเห็นว่าสามารถนำวิธีของงานวิจัยอื่นมาปรับใช้ได้ แต่ต้องพิจารณาวิธีที่เหมาะกับแผนภาพคลาสเชิงแนวคิด ซึ่งยังไม่มีรายละเอียดที่ซับซ้อนนักภายในแผนภาพ แต่ในขณะเดียวกันสามารถรองรับรูปแบบการเปลี่ยนแปลงที่หลากหลายซึ่งเกิดขึ้นได้ในระบบซอฟต์แวร์จริงขนาดใหญ่

ตัวอย่างเช่น งานวิจัยของ Girschick [4] ได้นำเสนอวิธีการในการหาความแตกต่างของแผนภาพคลาสเพื่อใช้ในการจัดเก็บแผนภาพคลาสต่างเวอร์ชัน โดยทำการแปลงแผนภาพคลาสเป็นเอกสารเอกซ์เอ็มแอลก่อน จากนั้นทำการเปรียบเทียบด้วยอัลกอริทึมที่นำเสนอ งานวิจัยดังกล่าวสามารถแสดงให้เห็นถึงความเปลี่ยนแปลงที่เกิดขึ้นกับแผนภาพคลาสหนึ่งจนทำให้ได้เป็นแผนภาพคลาสอีกเวอร์ชันหนึ่ง ได้แก่ การเพิ่ม ลบ เปลี่ยนชื่อ ย้ายส่วนย่อย (Element) ต่างๆ อันประกอบด้วย คลาส ลักษณะประจำ การดำเนินการ นอกจากนี้ยังสามารถตรวจหาการสำเนาการดำเนินการจากคลาสแม่ไปยังคลาสลูกหรือเรียกว่า Method Overriding ซึ่งในแผนภาพคลาสเชิงแนวคิดไม่มีการสำเนาในลักษณะนี้ วิธีนี้จึงมีความซับซ้อนเกินไปสำหรับการเปรียบ

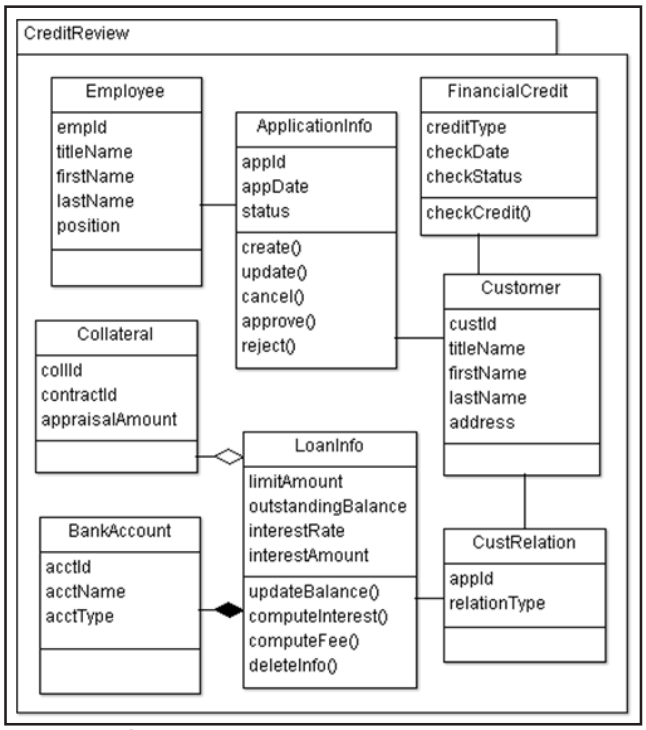
เทียบแผนภาพคลาสเชิงแนวคิด นอกจากนี้ยังมีงานวิจัยของ Auxepaules และคณะ [5] ได้นำเสนอวิธีการเปรียบเทียบแผนภาพคลาสสำหรับพัฒนาเป็นเครื่องมือที่ใช้ในการเรียนการสอน เครื่องมือดังกล่าวสามารถวิเคราะห์ความแตกต่างระหว่างแผนภาพคลาสของนักเรียนกับแผนภาพคลาสของผู้เชี่ยวชาญได้ การเปรียบเทียบโครงสร้างของแผนภาพคลาสทำโดยใช้อัลกอริทึมจับคู่กราฟทั่วไป (Generic Graph Matching) ซึ่ง Sorlin และคณะได้นำเสนอไว้ใน [6] ส่วนการเปรียบเทียบชื่อของหน่วยย่อยต่าง ๆ ได้ใช้วิธีการของ Giunchiglia และคณะ [7] ซึ่งมีการใช้เวิร์ดเน็ตสำหรับการเปรียบเทียบคำเชิงความหมายด้วย แต่ Auxepaules และคณะได้สรุปไว้ว่าวิธีการที่นำเสนอเหมาะกับแผนภาพคลาสที่ไม่ซับซ้อน วิธีนี้จึงไม่เหมาะสมนักที่จะนำมาใช้กับระบบงานจริง

ผู้วิจัยได้เลือกอัลกอริทึม UMLDiff ของ Xing [1] มาปรับใช้ เนื่องจากเป็นอัลกอริทึมที่พัฒนาขึ้นมาสำหรับการเปรียบเทียบแผนภาพคลาสโดยเฉพาะ แทนการนำวิธีการเปรียบเทียบกราฟโดยทั่วไปมาใช้ และเป็นอัลกอริทึมที่ใช้สำหรับเปรียบเทียบแผนภาพคลาสเพื่อให้เห็นวิวัฒนาการในการพัฒนาซอฟต์แวร์ งานวิจัยดังกล่าวได้ทำการสร้างแผนภาพคลาสจากรหัสต้นฉบับ (Source Code) ของโปรแกรมสองเวอร์ชัน ซึ่งแผนภาพคลาสที่ได้จะอยู่ในรูปของแบบจำลองเชิงตรรกะ (Logical Model) จากนั้นจึงนำแผนภาพคลาสมาเปรียบเทียบกันโดยแปลงเป็นกราฟก่อนที่จะทำการเปรียบเทียบกราฟด้วยอัลกอริทึม UMLDiff ผลลัพธ์ที่ได้ทำให้ทราบถึงความแตกต่างระหว่างแผนภาพคลาส ได้แก่ การเพิ่ม ลบ เปลี่ยนชื่อ ย้ายส่วนย่อย เป็นต้น

อย่างไรก็ตาม ในการออกแบบแผนภาพคลาสเชิงแนวคิดของระบบเดิมและระบบใหม่ อาจมีการกำหนดชื่อส่วนย่อยต่าง ๆ โดยใช้คำที่ไม่เหมือนกันแต่มีความหมายเดียวกัน ดังนั้นผู้วิจัยจึงได้เพิ่มการเปรียบเทียบความคล้ายกันในเชิงความหมายของการตั้งชื่อในแผนภาพ โดยอ้างอิงจากความสัมพันธ์เชิงความหมายตามโครงสร้างลำดับชั้นของคำศัพท์ในฐานข้อมูลเวิร์ดเน็ต [2] ซึ่งจะได้ค่าเป็นคะแนนความคล้ายคลึงกันในเชิงความหมาย วิธีการวัดคะแนนมีให้เลือกใช้หลายวิธี เช่น วิธี Wu & Palmer วิธี Hirst & St-Onge และวิธี Lin เป็นต้น

3. วิธีการเปรียบเทียบแผนภาพคลาสเชิงแนวคิด

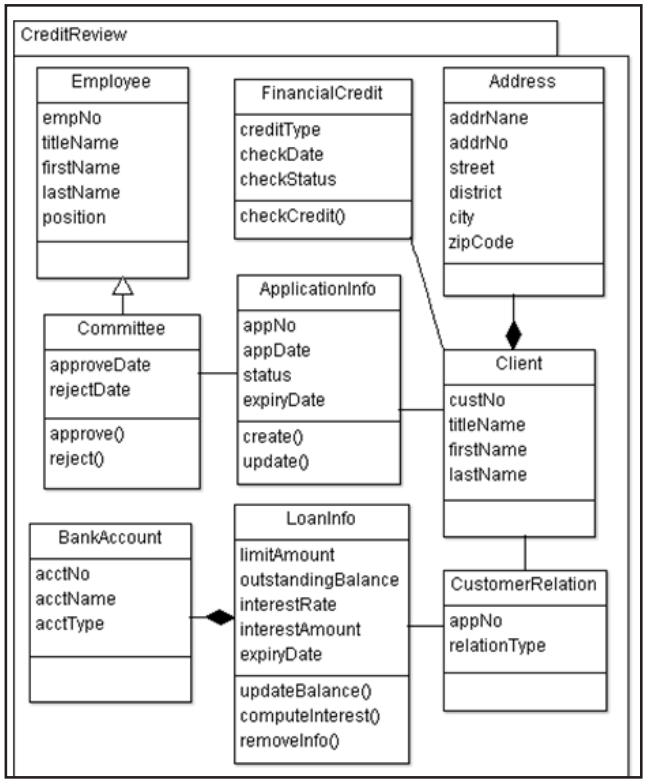
งานวิจัยนี้นำเสนออัลกอริทึม S-UMLDiff (หรือ Semantic-UMLDiff) เพื่อใช้ในการเปรียบเทียบแผนภาพคลาสเชิงแนวคิดโดยพิจารณาความหมายของคำในแผนภาพ เพิ่มเติมจากการพิจารณาโครงสร้างของคลาส อัลกอริทึม S-UMLDiff ปรับมาจากรหัสต้นฉบับของอัลกอริทึม UMLDiff ซึ่ง Xing ได้พัฒนาไว้ด้วยภาษาจาวา [1] ผู้วิจัยขอยกตัวอย่างบางส่วนของแผนภาพคลาสเชิงแนวคิดของระบบงานจริงของธนาคารแห่งหนึ่งในประเทศไทย มาใช้ในการนำเสนอ ภาพที่ 1 เป็นแผนภาพคลาสเชิงแนวคิดของระบบซอฟต์แวร์เดิม และภาพที่ 2 เป็นแผนภาพคลาสเชิงแนวคิดของระบบซอฟต์แวร์ที่จะพัฒนาใหม่



ภาพที่ 1 *แผนภาพคลาสเชิงแนวคิดของระบบเดิม*

ขั้นตอนการเปรียบเทียบแผนภาพคลาสเชิงแนวคิดเพื่อตรวจสอบความครบถ้วนของความต้องการในการเปลี่ยนแปลงซอฟต์แวร์มีดังนี้

1) นำแผนภาพคลาสเชิงแนวคิดของระบบเดิมและระบบใหม่มาแปลงเป็นเอกสารเอกซ์เอ็มไอ (XMI: XML Metadata Interchange) โดยใช้เครื่องมือสนับสนุน คือ ArgoUML [8]



ภาพที่ 2 *แผนภาพคลาสเชิงแนวคิดของระบบใหม่*

2) สกัด (Extract) ส่วนย่อยต่าง ๆ ของแผนภาพคลาสเชิงแนวคิดจากเอกสารเอกซ์เอ็มไอ ได้แก่ ชื่อคลาส ชื่อลักษณะประจำ ชื่อการดำเนินการ เป็นต้น

3) นำส่วนย่อยต่างๆ ที่สกัดได้มาสร้างข้อมูลกราฟด้วยวิธีการที่นำเสนอใน [1] คือ กำหนดให้ชื่อส่วนย่อยต่างๆ เป็นจุดต่อ (Node) หรือเอนทิตี (Entity) และกำหนดความสัมพันธ์ระหว่างส่วนย่อยเป็นเส้นเชื่อม (Edge) หรือความสัมพันธ์ (Relation) ตัวอย่างของข้อมูลเอนทิตีที่ได้จากแผนภาพคลาสเชิงแนวคิดในภาพที่ 1 แสดงดังตารางที่ 1 (ซึ่ง Virtual Root เป็นเอนทิตีที่จะถูกกำหนดให้กับทุกแผนภาพคลาสเชิงแนวคิดเสมอ) และเนื่องจาก UMLDiff เป็นอัลกอริทึมที่ใช้ในการเปรียบเทียบแผนภาพคลาสเชิงตรรกะ ดังนั้นเมื่อนำมาประยุกต์ใช้ในการเปรียบเทียบแผนภาพคลาสเชิงแนวคิดจึงจำเป็นต้องกำหนดประเภทความสัมพันธ์ของคลาสตามประเภทความสัมพันธ์ของเอนทิตีที่มีกำหนดไว้ใน UMLDiff ดังตารางที่ 2 นอกจากนี้ยังมีความสัมพันธ์อื่นๆ ที่ใช้แสดงถึงความสัมพันธ์ระหว่างเอนทิตี ได้แก่ Contain, Field Data Type และ Method Return Type ซึ่งมีตัวอย่างของข้อมูลความสัมพันธ์ที่ได้จากแผนภาพคลาสเชิงแนวคิดในภาพที่ 2 แสดงไว้ดังตารางที่ 3 แต่เนื่องจาก UMLDiff จำเป็นต้องระบุประเภทข้อมูลของลักษณะประจำ และประเภทข้อมูลที่ถูกส่งกลับจากการดำเนินการ (Method Return Type) แต่ในแผนภาพคลาสเชิงแนวคิดไม่มีข้อมูลเหล่านี้ ดังนั้นในงานวิจัยนี้จึงกำหนดให้ทุกลักษณะประจำมีประเภทข้อมูลเป็น String และทุกการดำเนินการมีประเภทข้อมูลที่ส่งกลับเป็น void

4) นำข้อมูลกราฟที่ได้มาทำการเปรียบเทียบกันโดยใช้อัลกอริทึม S-UMLDiff ซึ่งปรับปรุงจากอัลกอริทึม UMLDiff ทั้งในส่วนของการวัดความคล้ายกันของชื่อ (Name Similarity) และการวัดความคล้ายคลึงกันของโครงสร้าง (Structural Similarity)

รายละเอียดของอัลกอริทึม S-UMLDiff เป็นดังนี้

ตารางที่ 1 *ตัวอย่างเอนทิตีที่ได้จากแผนภาพระบบเดิมภาพที่ 1*

ประเภท	ชื่อเอนทิตี
Virtual Root	Virtual Root
Package	CreditReview
Class	FinancialCredit
Attribute	creditType, checkDate, checkStatus
Method	checkCredit()

ตารางที่ 2 *การกำหนดความสัมพันธ์ระหว่างคลาส*

ความสัมพันธ์ระหว่างคลาสในแผนภาพคลาสเชิงแนวคิด	ความสัมพันธ์ที่ใช้ในอัลกอริทึม UMLDiff
Extend	Extend
Association	Use
Aggregation	Implement
Composition	Implement

3.1 การพิจารณาความคล้ายกันของชื่อเอนทิตี

UMLDiff ใช้วิธี LCS (Longest Common Subsequence) ในการเปรียบเทียบความคล้ายกันของชื่อเอนทิตี แต่ในการออกแบบแผนภาพคลาสเชิงแนวคิดของระบบเดิมและระบบใหม่อาจมีการใช้คำที่เขียนไม่เหมือนกันแต่มีความหมายคล้ายกัน เช่น Customer ในภาพที่ 1 และ Client ในภาพที่ 2 ซึ่ง LCS ไม่สามารถหาความคล้ายกันเชิงความหมายได้

ตารางที่ 3 *ตัวอย่างความสัมพันธ์ที่ได้จากแผนภาพระบบใหม่ภาพที่ 2*

ประเภท	เอนทิตีที่เกียวข้อง	
Contain	Virtual Root	CreditReview
	CreditReview	ApplicationInfo
	ApplicationInfo	appNo
	ApplicationInfo	create()
Extend	Committee	Employee
Implement	LoanInfo	BankAccount
Use	LoanInfo	CustomerRelation
Field Data Type	appNo	String
Method Return Type	create()	void

ในงานวิจัยนี้จึงได้นำ WordNet::Similarity [2] มาใช้ในการเปรียบเทียบคำเชิงความหมายด้วยโดยเลือกใช้วิธี Wu & Palmer เนื่องจากเป็นวิธีที่ให้ผลลัพธ์ที่เหมาะสมที่สุดเมื่อเทียบกับผลลัพธ์ที่ได้จากวิธีต่างๆ อย่างไรก็ตาม เนื่องจากชื่อเอนทิตีมักจะประกอบด้วยคำมากกว่าหนึ่งคำหรือเป็นวลี (Phrase) ซึ่งเวิร์ดเน็ตไม่สามารถหาความหมายของวลีได้ ในงานวิจัยนี้จึงได้นำวิธีการหาความคล้ายกันเชิงความหมายของวลีจากงานวิจัย [9] มาใช้ในการเปรียบเทียบชื่อเอนทิตีต่าง ๆ ดังนั้นหากต้องการหาความคล้ายกันเชิงความหมายระหว่างวลี a ซึ่งมี m คำ กับวลี b ซึ่งมี n คำ จะสามารถหาได้ดังสมการ (1) และ (2)

$$wpSim(a,b) = \frac{\sum_{s=1}^{m} wpSim(a_s,b)}{m} \quad (1)$$

$$wpSim(a_s,b) = \max(wSim(a_s,b_1),..., wSim(a_s,b_n)) \quad (2)$$

โดยที่ $wSim(a_s, b_n)$ หมายถึง ค่าความคล้ายกันเชิงความหมายระหว่างคำที่ *s* ในวลี *a* กับคำที่ *n* ในวลี *b* ตามวิธี Wu & Palmer ซึ่งค่าจะอยู่ในช่วง [0, 1]

การที่สองวลีจะถือว่าเหมือนกันได้นั้น จะต้องมีค่า *pSim* ไม่น้อยกว่าค่าขีดแบ่งของความคล้ายคลึง (Similarity Threshold) ที่นักวิเคราะห์ระบบกำหนดในการเปรียบเทียบแผนภาพ ตัวอย่างเช่น ในการหาความคล้ายกันเชิงความหมายของชื่อการดำเนินการ deleteInfo ในภาพที่ 1 (วลี *a*) กับ removeInfo ในภาพที่ 2 (วลี *b*) โดยใช้สมการ (1) และ (2) จะได้ว่า

pSim(deleteInfo, removeInfo)

= (*wpSim*(delete,removeInfo)

+wpSim(Info,removeInfo))/2

= (*max*(*wSim*(delete, remove), *wSim*(delete, Info)) +

max(*wSim*(Info, remove), *wSim*(Info, Info)))/2

= (*max*(0.8, 0) + *max*(0.4, 1))/2 = (0.8+1)/2 = 0.9

มีข้อสังเกตว่า เนื่องจาก remove เป็นทั้งคำนามและคำกริยา ในกรณีนี้ S-UMLDiff จะถือว่าเป็นคำกริยาเพราะคำแรกของชื่อการดำเนินการจะเป็นคำกริยา หากในการเปรียบเทียบ นักวิเคราะห์ระบบกำหนดค่าขีดแบ่งไว้เป็น 0.9 ดังนั้น S-UMLDiff จะสรุปว่าชื่อการดำเนินการ deleteInfo และ removeInfo เป็นชื่อเดียวกันในเชิงความหมาย

3.2 การพิจารณาความคล้ายกันของโครงสร้าง

S-UMLDiff ใช้แนวทางของ UMLDiff ในการเปรียบเทียบความคล้ายกันในเชิงโครงสร้าง โดยการจับคู่เอนทิตีประเภทเดียวกันและมีชื่อเดียวกันหรือคล้ายกัน เพื่อเปรียบเทียบรายละเอียดภายใน ดังตารางที่ 4

ตารางที่ 4 *การเปรียบเทียบเชิงโครงสร้าง*

ประเภท	การเปรียบเทียบ
Package	Class ที่อยู่ภายใน
Class	Attribute และ Method ที่อยู่ภายในรวมทั้ง Class อื่นที่ Class นี้มีความสัมพันธ์ด้วย

อย่างไรก็ตาม ในการออกแบบแผนภาพคลาสเชิงแนวคิดของระบบใหม่ อาจมีเอนทิตีที่ถูกเปลี่ยนประเภทไป เช่น ลักษณะประจำ address ในภาพที่ 1 ถูกเปลี่ยนเป็นคลาส Address ในภาพที่ 2 ดังนั้นหากต้องการทราบถึงข้อมูลที่ขาดหายไปหรือเพิ่มเข้ามาในระบบใหม่ จึงจำเป็นต้องทำการเปรียบเทียบชื่อลักษณะประจำกับชื่อคลาสด้วย ซึ่งอัลกอริทึม UMLDiff ไม่สามารถตรวจสอบได้ ในงานวิจัยนี้จึงได้เพิ่มเติมความสามารถในการตรวจสอบการเปลี่ยนแปลงประเภทของส่วนย่อย ได้แก่ การเปลี่ยนจากลักษณะประจำเป็นคลาส รวมถึงการเปลี่ยนจากคลาสเป็นลักษณะประจำ

4. ผลการทดลอบ

จากการทดสอบพบว่า S-UMLDiff สามารถใช้ในการเปรียบเทียบแผนภาพคลาสเชิงแนวคิดได้ ประเภทของการเปลี่ยนแปลงที่สามารถตรวจหาได้ คือ การเพิ่ม ลบ เปลี่ยนชื่อ และย้ายส่วนย่อยต่าง ๆ อันประกอบด้วย แพคเกจ คลาส ลักษณะประจำ และการดำเนินการ ตัวอย่างของผลลัพธ์ที่ได้จากการเปรียบเทียบแผนภาพในภาพที่ 1 และ 2 โดยใช้อัลกอริทึม UMLDiff และ S-UMLDiff แสดงดังตารางที่ 5 UMLDiff สรุปว่า Customer และ address ถูกลบ ส่วน Client และ Address ถูกเพิ่ม แต่ S-UMLDiff สามารถสรุปได้ว่า Customer และ Client เป็นคลาสเดียวกันในเชิงความหมาย นอกจากนี้ยังสามารถตรวจสอบได้ว่าคลาส Address มีการเปลี่ยนประเภทของส่วนย่อยอีกด้วย ผู้วิจัยเห็นว่าวิธีการเช่นนี้ทำให้ผลลัพธ์ที่ได้ตรงตามวัตถุประสงค์ของการเปรียบเทียบแผนภาพคลาสเชิงแนวคิดสำหรับการพัฒนาซอฟต์แวร์ระบบใหม่บนพื้นฐานของข้อกำหนดความต้องการของระบบเดิม ซึ่งในการออกแบบอาจไม่ได้ดำเนินการโดยบุคคลเดียวกัน และอาจใช้คำต่างกันแต่มีความหมายเดียวกัน การเปรียบเทียบเชิงความหมายของชื่อส่วนย่อยและการตรวจสอบการเปลี่ยนประเภทของส่วนย่อยนี้ จะทำให้ทราบถึงสิ่งที่ขาดหายไปหรือเพิ่มเข้ามาในระบบใหม่ นักวิเคราะห์ระบบและผู้ใช้สามารถนำผลลัพธ์ที่ได้ไปใช้ในการพิจารณาว่า ส่วนที่แตกต่างกันระหว่างแผนภาพคลาสเชิงแนวคิดทั้งสองเป็นส่วนที่ต้องการเปลี่ยนแปลงซอฟต์แวร์หรือไม่

5. บทสรุป

งานวิจัยนี้ได้นำเสนออัลกอริทึม S-UMLDiff สำหรับใช้ตรวจสอบความครบถ้วนของความต้องการในการเปลี่ยนแปลงซอฟต์แวร์สำหรับการพัฒนาซอฟต์แวร์ระบบใหม่บนพื้นฐานของข้อกำหนดความต้องการของระบบเดิม โดยนำแผนภาพคลาสเชิงแนวคิดของระบบเดิมและระบบใหม่มาเปรียบเทียบกันเพื่อให้เห็นถึงฟังก์ชันและข้อมูลที่เพิ่มเข้ามาหรือขาดหายไป จากการทดสอบโดยใช้แผนภาพคลาสบางส่วนของระบบงานจริงพบว่า อัลกอริทึมสามารถตรวจหาความคล้ายกันและความแตกต่างระหว่างแผนภาพคลาสเชิงแนวคิดได้ แม้ว่าในแผนภาพนั้นจะกำหนดชื่อของส่วนย่อยแตกต่างกันหรือมีการเปลี่ยนประเภทของส่วนย่อยก็ตาม ในลำดับต่อไปผู้วิจัยจะทำการทดสอบโดยใช้แผนภาพคลาสเชิงแนวคิดของระบบงานจริงอีกระบบหนึ่งของธนาคารแห่งเดียวกันนี้ ซึ่งมีความซับซ้อนมากขึ้น และจะทำการประเมินความถูกต้องแม่นยำของอัลกอริทึม ในแง่ Precision และ Recall ของผลลัพธ์ที่ได้เมื่อเทียบกับการประเมินโดยนักวิเคราะห์ระบบและผู้ใช้ที่เป็นผู้เชี่ยวชาญ

ตารางที่ 5 *ตัวอย่างของผลลัพธ์ของการเปรียบเทียบภาพที่ 1 และ 2*

เอนทิตี		ประเภทการเปลี่ยนแปลง	
ระบบเดิม	**ระบบใหม่**	UMLDiff	S-UMLDiff
Virtual Root	Virtual Root	match	match
CreditReview	CreditReview	match	match
FinancialCredit	FinancialCredit	match	match
ApplicationInfo	ApplicationInfo	match	match
LoanInfo	LoanInfo	match	match
BankAccount	BankAccount	match	match
CustRelation	CustomerRelation	rename	rename
Employee	Employee	match	match
Customer		remove	semantic match
	Client	add	
Collateral		remove	remove
	Committee	add	add
address		remove	change type
	address	add	

6. เอกสารอ้างอิง

[1] Z. Xing. “Supporting Object-Oriented Evolutionary Development by Design Evolution Analysis.” Doctor’s Thesis, Department of Computing Science, University of Alberta, 2008. Available online at http://webdocs.cs.ualberta.ca/~stroulia/Zhenchang_Xing_Old_Home/

[2] T. Pedersen. “WordNet::Similarity.” Available online at http://wn-similarity.sourceforge.net/

[3] OMG. Unified Modeling Language. Available online

at http://www.omg.org/spec/UML/2.4.1/. 2012.

[4] M. Girschick. "Difference Detection and Visualization in UML Class Diagrams." *Technical Report TUD-CS-2006-5*, TU Darmstadt, Germany, 2006.

[5] L. Auxepaules, D. Py, and T. Lemeunier. "A Diagnosis Method that Matches Class Diagrams in a Learning Environment for Object-Oriented Modeling." *Proceedings of 8th IEEE International Conference on Advanced Learning Technologies (ICALT 2008)*, Santander, Cantabria, pp. 26-30, 2008.

[6] S. Sorlin, C. Solnon, J.-M. Jolion. "A Generic Graph Distance Measure Based on Multivalent Matchings." *Applied Graph Theory in Computer Vision and Pattern Recognition*, Studies in Computational Intelligence, Vol. 52, pp. 151-181, 2007.

[7] F. Giunchiglia, M. Yatskevich, and P. Shvaiko. "Semantic Matching: Algorithms and Implementation," *Journal on Data Semantics*, Vol. 9, pp. 1-38, 2007.

[8] Tigris.org. ArgoUML. Available online at http://argouml.tigris.org/. 2009.

[9] W. Gad and M. Kamel. "PH-SSBM: Phrase Semantic Similarity Based Model for Document Clustering." *Proceedings of 2009 Second International Symposium on Knowledge Acquisition and Modeling*, Washington, DC, USA, pp.197-200, 2009.