การเพิ่มประสิทธิภาพและประสิทธิผลในการใช้เทคนิคการอ่านซอฟต์แวร์โอโออาร์ทีในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ

นางสาวปรียาภรณ์ บุญพยนต์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2550

IMPROVING EFFICIENCY AND EFFECTIVENESS IN USING OBJECT-ORIENTED READING TECHNIQUE (OORT) IN SOFTWARE INSPECTION FOR OBJECT-ORIENTED DESIGN DOCUMENT

Miss Preeyaporn Boonpayont

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Business Software Development
Department of Statistics
Faculty of Commerce and Accountancy
Chulalongkorn University
Academic Year 2007

หัวข้อวิทยานิพนธ์	การเพิ่มประสิทธิภาพและประสิทธิผลในการใช้เทคนิคการอ่านซอฟต์แวร์โอโออาร์ทีในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ
โดย	นางสาวปรียาภรณ์ บุญพยนต์
สาขาวิชา	การพัฒนาซอฟต์แวร์ด้านธุรกิจ
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. อัษฎาพร ทรัพย์สมบูรณ์

คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะพาณิชยศาสตร์และการบัญชี
(รองศาสตราจารย์ ดร. อรรณพ ตันละมัย)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. ชัชพงศ์ ตั้งมณี)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร. อัษฎาพร ทรัพย์สมบูรณ์)

.....กรรมการ
(อาจารย์ ดร. จันทร์เจ้า มงคลนาวิน)

ปรียาภรณ์ บุญพยนต์: การเพิ่มประสิทธิภาพและประสิทธิผลในการใช้เทคนิคการอ่านซอฟต์แวร์โอโออาร์ทีในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ. (IMPROVING EFFICIENCY AND EFFECTIVENESS IN USING OBJECT-ORIENTED READING TECHNIQUE (OORT) IN SOFTWARE INSPECTION FOR OBJECT-ORIENTED DESIGN DOCUMENT) อ. ที่ปรึกษา : ผศ.ดร. อัษฎาพร ทรัพย์สมบูรณ์, 170 หน้า.

วัตถุประสงค์ของงานวิจัยนี้ คือ (1) เพื่อปรับปรุงเทคนิคโอโออาร์ทีโดยการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง จากการศึกษาลักษณะการเกิดของข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์โดยใช้แผนภาพยูเอ็มแอล (2) เพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล

งานวิจัยนี้แบ่งการดำเนินการออกเป็น 2 ส่วนเพื่อตอบวัตถุประสงค์ของงานวิจัย โดยในส่วนที่ 1 ดำเนินการเพื่อตอบวัตถุประสงค์ข้อที่ 1 โดยศึกษาข้อบกพร่องที่มีในโครงงานการวิเคราะห์และออกแบบระบบด้านธุรกิจด้วยแผนภาพยูเอ็มแอล จัดทำโดยนิสิตในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ ในวิชาการออกแบบระบบงานด้านธุรกิจ โดยผลการศึกษาที่ได้จะนำไปปรับปรุงเทคนิคโอโออาร์ที และในส่วนที่ 2 เป็นการทดลองเพื่อตอบวัตถุประสงค์ข้อที่ 2 โดยให้นิสิตระดับมหาบัณฑิตที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุแล้ว ตรวจสอบเอกสารการออกแบบซอฟต์แวร์โดยแผนภาพยูเอ็มแอล โดยใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

ผลการศึกษาในส่วนที่ 1 สามารถปรับปรุงเทคนิคโอโออาร์ทีได้ และในส่วนที่ 2 พบว่าเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพสูงกว่าเทคนิคโอโออาร์ทีเดิม แต่ไม่สูงกว่าเทคนิคซีบีอาร์ แต่เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิผลไม่แตกต่างกับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ทั้งนี้ถ้าไม่คำนึงถึงข้อบกพร่องของเอกสารอธิบายคลาสจากการเปรียบเทียบแผนภาพคลาสเทียบกับเอกสารอธิบายคลาส พบว่า เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพสูงกว่าทั้งเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ และเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิผลสูงกว่าเทคนิคโอโออาร์ทีเดิม แต่ไม่แตกต่างกับเทคนิคซีบีอาร์

ภาควิชา.....สถิติ..... ลายมือชื่อนิสิต.....ปรียาภรณ์ บุญพยนต์.....

สาขาวิชา...การพัฒนาซอฟต์แวร์ด้านธุรกิจ..... ลายมือชื่ออาจารย์ที่ปรึกษา.....อัษฎาพร ทรัพย์สมบูรณ์.....

ปีการศึกษา.....2550.....

##4882217226 : MAJOR BUSINESS SOFTWARE DEVELOPMENT
KEY WORD: SOFTWARE INSPECTION / OBJECT-ORIENTED READING TECHNIQUE / OORT

PREEYAPORN BOONPAYONT : IMPROVING EFFICIENCY AND EFFECTIVENESS IN USING OBJECT-ORIENTED READING TECHNIQUE (OORT) IN SOFTWARE INSPECTION FOR OBJECT-ORIENTED DESIGN DOCUMENT. THESIS ADVISOR : ASST. PROF. ASSADAPORN SAPSOMBOON, 170 pp.

The objective of the thesis are (1) To improve OORT software reading technique with reduced scenarios based on the distribution of defect types occurring in object-oriented design document with UML (2) To compare the efficiency and the effectiveness of software reading techniques : the improved OORT, the ordinary OORT, and the checklist based reading technique (CBR) in inspection of object-oriented design document with UML.

The thesis is separated into 2 sections. The first section supports the first objective of the thesis in which object-oriented designs of small business systems as student projects in the Business System Design class in the Master of Science Program in Information Technology in Business are inspected. The defect found are analyzed for its distribution. In the other section, an experiment on current graduate students in software development related program with a prerequisite in object-oriented analysis and design course performed software inspection using the improved OORT on the designated design documents.

The first section of the study results in the improved OORT. In the second section, it is found that using the improved OORT is more efficient than using the ordinary OORT, but using the CBR technique is still the most efficient. Nevertheless, using the improved OORT is not more effective than neither the ordinary OORT, nor the CBR technique. After neglecting the defects in class description according to class diagram, using the improved OORT is the most efficient, but using the improved OORT is not more effective than the CBR technique.

Department : .....Statistics..... Student's Signature : Preeyaporn Boonpayont

Field of Study : ...Business Software Development... Advisor's Signature : Assadaporn Sapsomboon

Academic Year : .....2007.....

กิตติกรรมประกาศ

ผู้วิจัยขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร. อัษฎาพร ทรัพย์สมบูรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้การชี้แนะแนวทางต่างๆ ให้กับผู้วิจัยจนสำเร็จเป็นวิทยานิพนธ์ฉบับนี้ และขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร. ชัชพงศ์ ตั้งมณี ประธานกรรมการวิทยานิพนธ์ และอาจารย์ ดร. จันทร์เจ้า มงคลนาวิน กรรมการวิทยานิพนธ์ ที่ช่วยชี้แนะสิ่งต่างๆ และอาจารย์ทุกท่านที่ให้ความรู้และอบรมสิ่งต่างๆ ให้กับผู้วิจัย และขอขอบคุณโครงงานการออกแบบระบบจากนิสิตในรายวิชาการออกแบบระบบงานด้านธุรกิจ และนิสิตสาขาการพัฒนาซอฟต์แวร์ด้านธุรกิจ และสาขาเทคโนโลยีสารสนเทศทางธุรกิจ ทุกท่านที่ช่วยสนับสนุนการวิจัยของผู้วิจัย

ที่สำคัญที่สุดขอขอบพระคุณคุณพ่อ คุณแม่ และพี่สาวที่ให้การสนับสนุน และเป็นกำลังใจที่ดีมาโดยตลอด สุดท้ายขอขอบคุณนางสาวอุมาพร นิลเอวะ และเพื่อนๆทุกคนที่คอยช่วยเหลือและให้คำปรึกษาที่ดี

สารบัญ

หน้า

หน้า

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การผลิตซอฟต์แวร์ในปัจจุบันคำนึงถึงคุณภาพของซอฟต์แวร์มากยิ่งขึ้น โดยคุณภาพของซอฟต์แวร์ หมายถึง การสร้างซอฟต์แวร์ได้ถูกต้องตรงตามความต้องการของลูกค้าหรือผู้ใช้งาน และ ต้องมีคุณสมบัติของคุณภาพตามที่กำหนดไว้ด้วย (บงกชธร เพ็ชรนิจินดา, 2548) โดยคุณภาพของซอฟต์แวร์ที่ได้ขึ้นอยู่กับการควบคุมคุณภาพของซอฟต์แวร์เพื่อให้ได้ซอฟต์แวร์ที่ตรงตามความต้องการของลูกค้า การควบคุมคุณภาพของซอฟต์แวร์ ประกอบด้วย (1) การทดสอบซอฟต์แวร์ (Software Testing) และ (2) การทบทวนซอฟต์แวร์ (Software Review) โดยการทบทวนซอฟต์แวร์ที่ได้กำหนดขั้นตอนการดำเนินการและหน้าที่ของผู้ที่เกี่ยวข้องที่ชัดเจน เรียกว่า การตรวจสอบซอฟต์แวร์ (Software Inspection) การทดสอบซอฟต์แวร์เป็นการตรวจสอบคุณภาพของซอฟต์แวร์ การทดสอบจะทำหลังจากการสร้างซอฟต์แวร์เสร็จเรียบร้อยแล้ว (บงกชธร เพ็ชรนิจินดา, 2548) การตรวจสอบซอฟต์แวร์เป็นการระบุข้อบกพร่องของสิ่งที่ตรวจสอบ เพื่อทำให้เกิดความเชื่อมั่นว่าจะถูกต้องหลังจากตรวจสอบและแก้ไขแล้ว (O'Regan, 2003)

การตรวจสอบซอฟต์แวร์สามารถทำได้ในตอนท้ายของแต่ละขั้นตอนการพัฒนาซอฟต์แวร์ เพื่อให้ผู้พัฒนาซอฟต์แวร์มั่นใจว่าการพัฒนาแต่ละขั้นตอนมีคุณภาพตรงตามความต้องการของลูกค้า วัตถุประสงค์สำคัญของการตรวจสอบซอฟต์แวร์ คือ การค้นหาข้อบกพร่อง (Defect Detection) จากสิ่งที่ตรวจสอบ (O'Regan, 2003) รูปแบบของการตรวจสอบซอฟต์แวร์มีหลายรูปแบบ แตกต่างกันที่กระบวนการในการดำเนินการตรวจสอบ ทำให้รูปแบบของการตรวจสอบซอฟต์แวร์มีลักษณะความเป็นทางการของขั้นตอนการดำเนินการไม่เท่ากัน

วิธีการตรวจสอบซอฟต์แวร์ที่มีรูปแบบที่เป็นทางการที่มีชื่อเสียงคือ วิธีตรวจสอบซอฟต์แวร์ของ Fagan (1976) ที่มีหกขั้นตอนในการดำเนินงานดังนี้ (1) การวางแผน (Planning) (2) การประชุมแนะนำ (Overview) (3) การจัดเตรียม (Preparation) (4) การประชุมตรวจสอบ (Inspection Meeting) (5) การแก้ไขใหม่ (Re-work) และ (6) การติดตาม (Follow Up) ขั้นตอนการจัดเตรียมเป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนดำเนินการตรวจสอบซอฟต์แวร์เฉพาะบุคคล (Individual Inspection) เพื่อค้นหาข้อบกพร่อง โดยมีเครื่องมือเพื่อช่วยผู้ตรวจสอบค้นหาข้อบกพร่องของซอฟต์แวร์หลายรูปแบบที่เรียกว่า เทคนิคการอ่านซอฟต์แวร์ (Software Reading Technique) ที่มีขั้นตอนการทำงานแตกต่างกันให้ผู้ตรวจสอบเลือกใช้ (Frankovich, 1995) เพื่อให้เหมาะสมกับซอฟต์แวร์ที่ตรวจสอบ

เทคนิคการอ่านซอฟต์แวร์เป็นการเพิ่มประสิทธิภาพของการตรวจสอบซอฟต์แวร์ (Travassos และคณะ, 2002) โดยเทคนิคการอ่านซอฟต์แวร์ประกอบด้วยคำแนะนำในการตรวจสอบซอฟต์แวร์เพื่อใช้เป็นแนวทางในการตรวจสอบเฉพาะบุคคลเพื่อให้ผู้ตรวจสอบใช้พิจารณาสิ่งที่ตรวจสอบและค้นหาข้อบกพร่อง (Travassos และคณะ, 2002) เทคนิคการอ่านซอฟต์แวร์มีหลายประเภท โดยแต่ละประเภทจะมีคำแนะนำในการตรวจสอบซอฟต์แวร์ต่างกัน ในอดีตเทคนิคการอ่านซอฟต์แวร์ถูกสร้างขึ้นเพื่อให้มีความเหมะสมกับการพัฒนาซอฟต์แวร์โดยวิธีโครงสร้าง (Travassos และคณะ, 2002)

ในปัจจุบันการพัฒนาซอฟต์แวร์โดยวิธีเชิงวัตถุได้รับความนิยมมากขึ้น การออกแบบซอฟต์แวร์ของการพัฒนาซอฟต์แวร์โดยวิธีเชิงวัตถุนิยมใช้ภาษายูเอ็มแอลในการออกแบบ (Dennis และคณะ, 2005) โดยภาษายูเอ็มแอลใช้สัญลักษณ์รูปภาพเพื่อแสดงถึงความต้องการของลูกค้าและการออกแบบซอฟต์แวร์ แทนการออกแบบของการพัฒนาโดยวิธีโครงสร้างที่อยู่ในรูปแบบเดิมที่เป็นลักษณะเป็นข้อความที่บรรยายถึงลักษณะของการออกแบบ ทำให้ระบบที่ออกแบบโดยภาษายูเอ็มแอลมีความชัดเจนและเข้าใจง่ายขึ้น (Dennis และคณะ, 2005) ดังนั้นเทคนิคการอ่านซอฟต์แวร์ในแบบเดิมอาจจะไม่เหมาะสมที่จะใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล (Laitenberger และคณะ,1999) ดังนั้นจึงปรับปรุงเทคนิคการอ่านซอฟต์แวร์ให้มีความเหมาะสมกับการตรวจสอบเอกสารการออกแบบของการพัฒนาซอฟต์แวร์โดยวิธีเชิงวัตถุประกอบด้วยเทคนิค ดังนี้

1. เทคนิคเฉพาะกิจ (Ad-hoc) เป็นวิธีที่ไม่เตรียมคำแนะนำในการตรวจสอบให้กับผู้ตรวจสอบว่าต้องดำเนินการอย่างไร และต้องหาข้อบกพร่องตรงส่วนใดบ้าง (Laitenberger และคณะ, 2000) วิธีนี้ผู้ตรวจสอบต้องมีประสบการณ์เพื่อทราบว่าต้องทำอย่างไรจึงตรวจพบข้อบกพร่อง

2. เทคนิคการอ่านบนพื้นฐานของเช็กลิสต์ (Checklist-Based Reading) หรือเรียกโดยย่อว่าเทคนิคซีบีอาร์ (CBR) เป็นวิธีที่ได้รับความนิยมมาก (Parnas และคณะ,2003) ที่ได้ดำเนินการปรับปรุงให้เหมาะกับการพัฒนาซอฟต์แวร์เชิงวัตถุแล้ว โดยจะมีแนวทางในการค้นหาข้อบกพร่องให้ผู้ตรวจสอบ อยู่ในรูปแบบของรายการคำถามแบบใช่ หรือ ไม่ใช่ โดยรายการคำถามจะตั้งขึ้นเพื่อรองรับการค้นหาข้อบกพร่องประเภทที่เคยพบในเอกสารรูปแบบเดียวกันในอดีต ดังนั้นเทคนิคนี้อาจจะมองข้ามข้อบกพร่องที่ไม่เคยพบในอดีตได้

3. เทคนิคการอ่านบนพื้นฐานของสถานการณ์ (Scenario-Based Reading) หรือเรียกโดยย่อว่าเทคนิคเอสบีอาร์ (SBR) เป็นวิธีที่จัดเตรียมคำแนะนำในการตรวจสอบซอฟต์แวร์ให้กับผู้ตรวจสอบโดยจะอยู่ในรูปแบบของรายการสถานการณ์ (Scenario) เพื่อบอกว่าต้อง

ตรวจสอบอะไร และตรวจสอบอย่างไรในเอกสารที่ตรวจสอบ (Basili และคณะ, 1996) มีรูปแบบของสถานการณ์ที่ต่างกัน 5 รูปแบบ ดังนี้

ก. เทคนิคการอ่านบนพื้นฐานของข้อบกพร่อง (Defect-Based Reading) หรือเรียกโดยย่อว่าเทคนิคดีบีอาร์ (DBR) ขั้นตอนการดำเนินการจะพิจารณาจากประเภทของข้อบกพร่องที่ได้กำหนดไว้ และเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์อยู่ในรูปแบบของรายการสถานการณ์ที่รองรับการค้นหาข้อบกพร่องตามประเภทที่กำหนดไว้ โดยรายการสถานการณ์จะมุ่งเน้นไปที่การค้นหาข้อบกพร่องตามประเภทที่กำหนดไว้ (Porter และคณะ, 1995)

ข. เทคนิคการอ่านบนพื้นฐานของสถานการณ์ที่อยู่บนพื้นฐานของฟังก์ชันพอยต์ (Scenario-based Reading Technique Based on Function Points) การกำหนดขั้นตอนการดำเนิน การมีพื้นฐานมาจากการวิเคราะห์ด้วยวิธีฟังก์ชันพอยต์ (Cheng และคณะ, 1996)

ค. เทคนิคการอ่านบนพื้นฐานของมุมมอง (Perspective Based Reading) หรือเรียกโดยย่อว่าเทคนิคพีบีอาร์ (PBR) ขั้นตอนการดำเนินการจะรองรับการตรวจสอบเอกสารตามมุมมองของผู้เกี่ยวข้องแต่ละหน้าที่ โดยมีวัตถุประสงค์เพื่อค้นหาข้อบกพร่องที่ต่างมุมมองกัน (Basili และคณะ, 1996)

ง. เทคนิคการอ่านบนพื้นฐานของการใช้ (Usage-Based Reading) หรือเรียกโดยย่อว่าเทคนิคยูบีอาร์ (UBR) เป็นเทคนิคที่รายการสถานการณ์มุ่งประเด็นไปที่การพิจารณาถึงแรงงานที่ใช้ในการค้นหาข้อบกพร่อง และด้วยเหตุผลที่ว่าข้อบกพร่องแต่ละประเภทมีความสำคัญแตกต่างกัน จึงจัดลำดับความสำคัญในการค้นหาข้อบกพร่องว่าข้อบกพร่องใดที่มีความวิกฤต โดยดูจากมุมมองของผู้ใช้ เพื่อมุ่งไปที่การค้นหาข้อบกพร่องประเภทนั้นก่อน (Thelin และคณะ, 2003)

จ. เทคนิคการอ่านบนพื้นฐานของความสามารถติดตาม (Traceability-Based Reading) หรือเรียกโดยย่อว่าเทคนิคทีบีอาร์ (TBR) หรือเรียกอีกชื่อหนึ่งว่า เทคนิคการอ่านเชิงวัตถุ (Object-Oriented Reading Technique) หรือเรียกโดยย่อว่า เทคนิคโอโออาร์ที (OORT) เป็นวิธีที่ใช้สำหรับค้นหาข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ของการพัฒนาซอฟต์แวร์ด้วยวิธีเชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอลโดยเฉพาะ เนื่องจากปัจจุบันการพัฒนาซอฟต์แวร์โดยวิธีเชิงวัตถุได้รับความนิยมมาก ดังนั้น Travassos และคณะ จึงได้พัฒนาเทคนิคโอโออาร์ทีขึ้นเพื่อให้มีความเหมาะสมต่อการพัฒนาซอฟต์แวร์เชิงวัตถุโดยเฉพาะ (Travassos และคณะ, 2002)

วัตถุประสงค์สำคัญของการสร้างเทคนิคโอโออาร์ทีมี 2 ประการ คือ (1) เอกสารการออกแบบต่างประเภทกันสามารถอธิบายระบบได้สอดคล้องกันหรือไม่ และ (2) เอกสารการออกแบบถูกต้องตรงตามความต้องการที่กำหนดไว้หรือไม่ (Shull และคณะ, 1999) เทคนิคโอโอ

อาร์ทีสามารถแบ่งออกเป็น 2 ส่วน คือ การอ่านแนวนอน (Horizontal Reading) เป็นการอ่านเพื่อเปรียบเทียบเอกสารการออกแบบ โดยคำนึงถึงความสอดคล้องกันของเอกสารในขั้นตอนเดียวกัน และการอ่านแนวตั้ง (Vertical Reading) เป็นการเปรียบเทียบเอกสารที่ได้จากขั้นตอนการพัฒนาที่ต่างกัน เช่น เอกสารอธิบายยูสเคส (Use Case Description) เปรียบเทียบกับ แผนภาพซีเควนซ์ (Sequence Diagram) เพื่อเป็นการคำนึงถึงความถูกต้องของการออกแบบ (Conradi และคณะ, 2003)

จากประเภทของเทคนิคการอ่านซอฟต์แวร์ที่กล่าวมา เทคนิคซีบีอาร์เป็นเทคนิคหนึ่งที่น่าสนใจ เนื่องจากเป็นเทคนิคที่ได้รับความนิยมนำมาใช้ในการตรวจสอบซอฟต์แวร์ (Porter และคณะ, 1995) และอีกเทคนิคที่น่าสนใจก็คือ เทคนิคโอโออาร์ที เนื่องจากเป็นเทคนิคที่ถูกสร้างเพื่อรองรับการพัฒนาซอฟต์แวร์โดยวิธีเชิงวัตถุโดยตรง และ อุมาพร นิลเอวะ (2549) ได้เปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคการอ่านซอฟต์แวร์ทั้ง 2 เทคนิค และพบว่า เทคนิคโอโออาร์ทีไม่ได้มีประสิทธิภาพและประสิทธิผลมากกว่าเทคนิคซีบีอาร์ ด้วยระยะเวลาในการตรวจสอบที่จำกัด เทคนิคซีบีอาร์จึงอาจจะดูเหมือนมีประสิทธิภาพและประสิทธิผลมากกว่าเทคนิคโอโออาร์ทีที่มีจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่ใช้ค้นหาข้อบกพร่องมากกว่าเทคนิคซีบีอาร์มาก จึงอาจจะทำให้ผู้ตรวจสอบต้องใช้เวลาในการค้นหาข้อบกพร่องนานกว่าเทคนิคซีบีอาร์

จากที่กล่าวมาแล้วว่าเทคนิคโอโออาร์ทีเป็นเทคนิคที่สร้างขึ้นเพื่อการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ที่พัฒนาด้วยวิธีเชิงวัตถุโดยตรง เพื่อค้นหาข้อบกพร่องที่เกิดขึ้นด้วยการพิจารณาเอกสารการออกแบบเทียบกับเอกสารความต้องการ เพื่อตรวจสอบความถูกต้องของการออกแบบ และค้นหาข้อบกพร่องที่เกิดขึ้นเมื่อเปรียบเทียบเอกสารการออกแบบที่ต่างประเภทกัน เพื่อคำนึงถึงความสอดคล้องกันของเอกสารการออกแบบ (Travassos และคณะ, 2002) การค้นหาข้อบกพร่องให้ได้มากที่สุดเหมือนเป็นหัวใจหลักของเทคนิคการอ่านซอฟต์แวร์ทุกเทคนิค โดยข้อบกพร่องที่เกิดในการพิจารณาเพื่อตรวจสอบความสอดคล้องกันของเอกสารการออกแบบเป็นสิ่งที่ทุกเทคนิคการอ่านซอฟต์แวร์ให้ความสำคัญ แต่การพิจารณาความถูกต้องของการออกแบบเมื่อเทียบกับเอกสารความต้องการยังไม่ได้รับความสนใจมากนัก (Travassos และคณะ, 2002) แต่ว่าจำนวนของข้อบกพร่องในส่วนนี้เป็นอีกส่วนที่มีความสำคัญที่ควรคำนึงถึง เนื่องจากเป็นส่วนที่แสดงว่าการออกแบบถูกต้องตรงตามความต้องการของผู้ใช้งานหรือไม่ โดยข้อบกพร่องในส่วนนี้เป็นข้อบกพร่องที่เทคนิคโอโออาร์ทีให้ความสำคัญด้วย ดังนั้นเทคนิคโอโออาร์ทีก็ยังคงเป็นเทคนิคที่น่าสนใจ

เทคนิคโอโออาร์ทีมีเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่มีขั้นตอนการตรวจสอบในการค้นหาข้อบกพร่องมาก จึงต้องใช้เวลาในการตรวจสอบมาก ถ้าพิจารณาจากขั้นตอนการตรวจสอบทั้งหมด บางข้อบกพร่องสามารถตรวจพบได้ในหลายขั้นตอน (Conradi และ

คณะ, 2003) ดังนั้นงานวิจัยนี้จึงต้องการปรับปรุงเทคนิคโอโออาร์ที โดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง แต่ว่ายังคงมีประสิทธิภาพและประสิทธิผลเทียบเท่ากับเทคนิคโอโออาร์ทีเดิม โดยในการปรับปรุงจะให้ความสำคัญกับการค้นหาข้อบกพร่องที่เกิดขึ้นทั้งในส่วนของการเปรียบเทียบระหว่าง (1) เอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ต่างประเภทกัน และ (2) เอกสารการออกแบบซอฟต์แวร์เชิงวัตถุเทียบกับเอกสารความต้องการ เป็นข้อบกพร่องที่เทคนิคการอ่านซอฟต์แวร์วิธีอื่นไม่ได้ให้ความสำคัญ แต่เป็นข้อบกพร่องที่ไม่ควรละเลยเพราะอาจจะทำให้การออกแบบผิดไปจากความต้องการที่แท้จริงได้ (Travassos และคณะ, 2002) โดยจะเรียกเทคนิคโอโออาร์ทีที่ปรับปรุงโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ว่าเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

งานวิจัยนี้เริ่มจากการศึกษาประเภทของข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่เคยมีรายงานในอดีต หลังจากนั้นผู้วิจัยจะกำหนดผู้ตรวจสอบให้ดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยวิธีเชิงวัตถุ โดยใช้เทคนิคโอโออาร์ทีในการค้นหาข้อบกพร่องที่มี เพื่อวิเคราะห์การกระจายของข้อบกพร่องที่พบกับประเภทข้อบกพร่องจากงานวิจัยในอดีต เพื่อพิจารณาประเภทข้อบกพร่องที่ตรวจพบว่ามีการกระจายตัวของประเภทข้อบกพร่องเหมือนกับที่พบในงานวิจัยในอดีตหรือไม่ เพื่อเป็นการพิจารณาว่าเอกสารการออกแบบซอฟต์แวร์ที่เลือกมาตรวจสอบเพื่อปรับปรุงเทคนิคโอโออาร์ที เป็นตัวแทนที่ดีของเอกสารการออกแบบซอฟต์แวร์ทั้งหมด และพิจารณาว่าแต่ละขั้นตอนของเทคนิคโอโออาร์ทีสามารถตรวจพบข้อบกพร่องได้จำนวนเท่าใด และพิจารณาจำนวนข้อบกพร่องที่มีในเอกสารแสดงความต้องการและเอกสารการออกแบบซอฟต์แวร์แต่ละประเภทที่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ที เพื่อใช้เป็นข้อมูลในการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีลง

แล้วจึงทดสอบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว โดยเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วกับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ที่เป็นเทคนิคที่นิยมใช้และเคยมีผู้นำเทคนิคทั้งสองมาเปรียบเทียบกันแล้ว แต่ด้วยระยะเวลาในการเปรียบเทียบที่จำกัดจึงทำให้เทคนิคโอโออาร์ทีมีประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์น้อยกว่าเทคนิคซีบีอาร์ เนื่องจากเทคนิคโอโออาร์ทีมีเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่ขั้นตอนมากกว่า การเปรียบเทียบประสิทธิภาพจะพิจารณาถึงจำนวนข้อบกพร่องที่ผู้ตรวจสอบแต่ละคนตรวจพบจากการใช้เทคนิคการอ่านซอฟต์แวร์ที่กำหนดไว้ เทียบกับระยะเวลาในการตรวจสอบของผู้ตรวจสอบแต่ละคนที่ใช้เทคนิคการอ่านซอฟต์แวร์ที่กำหนดไว้ แล้วนำมาเปรียบเทียบกันระหว่างเทคนิคการอ่านซอฟต์แวร์ที่ต่างกัน และประสิทธิผลจะพิจารณาถึงจำนวน

ข้อบกพร่องที่พบโดยผู้ตรวจสอบแต่ละคน ที่ใช้เทคนิคการอ่านซอฟต์แวร์ที่กำหนดให้ เทียบกับจำนวนข้อบกพร่องทั้งหมดที่มีในเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่กำหนดไว้

1.2 วัตถุประสงค์งานวิจัย

1.2.1 เพื่อวิเคราะห์การกระจายตัวของข้อบกพร่องที่พบจากการใช้เทคนิคโอโออาร์ทีในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยวิธีเชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล

1.2.2 เพื่อเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล

1.2.3 เพื่อเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล

1.3 ขอบเขตงานวิจัย

1.3.1 งานวิจัยนี้ผู้ตรวจสอบจะค้นหาประเภทของข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบด้วยแผนภาพยูเอ็มแอล โดยจะพิจารณาทั้งข้อบกพร่องที่ได้จากการเปรียบเทียบเอกสารการออกแบบซอฟต์แวร์ที่ต่างประเภทกัน และที่ได้จากการเปรียบเทียบเอกสารการออกแบบซอฟต์แวร์เทียบกับเอกสารความต้องการ ในการค้นหาจะให้ความสำคัญกับประเภทของข้อบกพร่องที่เกิดจากการเปรียบเทียบเอกสารการออกแบบและเอกสารความต้องการเนื่องจากเป็นข้อบกพร่องที่เทคนิคการอ่านซอฟต์แวร์ประเภทต่างๆไม่ให้ความสำคัญ แต่เทคนิคโอโออาร์ทีให้ความสำคัญด้วย

1.3.2 เทคนิคการอ่านซอฟต์แวร์ที่นำมาเปรียบเทียบเพื่อหาประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ ประกอบด้วย 3 เทคนิค คือ เทคนิคซีบีอาร์ เทคนิคโอโออาร์ทีเดิมและเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

1.3.3 การตรวจสอบเอกสารการออกแบบซอฟต์แวร์ที่กระทำในงานวิจัยนี้ แบ่งออกเป็น 2 ส่วน คือ การตรวจสอบเพื่อนำผลไปใช้ปรับปรุงเทคนิคโอของการตรวจโออาร์ที โดยจะดำเนินการตรวจสอบในขั้นตอนการประชุมแนะนำ ขั้นตอนการจัดเตรียม และขั้นตอนการประชุมตรวจสอบ และส่วนที่สองที่เปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคการอ่านซอฟต์แวร์ โดยจะดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ในขั้นตอนการประชุมแนะนำและขั้นตอนการจัดเตรียมของขั้นตอนการตรวจสอบซอฟต์แวร์

1.3.4 งานวิจัยนี้จะดำเนินการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ทางด้านธุรกิจ ที่ออกแบบโดยแผนภาพยูเอ็มแอล ประกอบด้วยแผนภาพยูสเคส (Use Case Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) แผนภาพสถานะ (State Machine Diagram) และแผนภาพคลาส (Class Diagram) รวมทั้งเอกสารอธิบายคลาส (Class Description) เอกสารอธิบายยูสเคส (Use Case Description) และเอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) เท่านั้น

1.3.5 เอกสารแสดงการออกแบบซอฟต์แวร์ที่นำมาใช้ในการทดลอง เพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ เป็นแผนภาพที่มีความถูกต้องและน่าเชื่อถือ เนื่องจากเป็นเอกสารเดียวกับงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่ผ่านการทดสอบและใช้งานจริงในงานวิจัยนั้นแล้ว

1.4 ขั้นตอนการดำเนินงานวิจัย

1.4.1 ศึกษาการใช้แผนภาพยูเอ็มแอลในการออกแบบซอฟต์แวร์เชิงวัตถุ

1.4.2 ศึกษาประเภทข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์

1.4.3 ศึกษากระบวนการตรวจสอบซอฟต์แวร์ เพื่อใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล

1.4.4 ศึกษากระบวนการดำเนินการของเทคนิคการอ่านซอฟต์แวร์ที่ใช้ในงานวิจัยนี้ คือ เทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์

1.4.5 เลือกเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุ เพื่อใช้ในการตรวจสอบซอฟต์แวร์โดยใช้เทคนิคโอโออาร์ทีในการค้นหาข้อบกพร่อง

1.4.6 ดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ โดยใช้เทคนิคโอโออาร์ทีในการค้นหาข้อบกพร่อง

1.4.7 นำผลการตรวจสอบที่ได้จากการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที มาวิเคราะห์การกระจายตัวของประเภทข้อบกพร่องที่ศึกษาได้จากงานวิจัยในอดีต เพื่อพิจารณาว่าเอกสารการออกแบบซอฟต์แวร์ที่เลือกมาตรวจสอบ เพื่อปรับปรุงเทคนิคโอโออาร์ที เป็นตัวแทนที่ดีของเอกสารการออกแบบซอฟต์แวร์ทั้งหมด และพิจารณาจำนวนข้อบกพร่องที่สามารถตรวจพบในแต่ละขั้นตอนของเทคนิคโอโออาร์ที และพิจารณาจำนวนข้อบกพร่องที่มีในเอกสารแสดงความต้องการและเอกสารการออกแบบซอฟต์แวร์แต่ละประเภทที่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ที เพื่อใช้เป็นข้อมูลในการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีลง

1.4.8 ทดสอบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว โดยการเปรียบเทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล

1.4.9 นำข้อมูลที่ได้จากการทดสอบมาใช้ในการตอบวัตถุประสงค์ของงานวิจัย

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ผู้ที่ต้องการตรวจสอบซอฟต์แวร์เชิงวัตถุในส่วนของเอกสารการออกแบบ ที่ออกแบบโดยแผนภาพยูเอ็มแอลสามารถนำงานวิจัยนี้ไปใช้เป็นแนวทางในการตรวจสอบได้

1.5.2 ถ้าเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์มากกว่าเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ จะทำให้ผู้ที่สนใจในเรื่องของการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอลนิยมใช้วิธีนี้มากขึ้น

1.5.3 ผู้ที่มีความสนใจที่จะใช้เทคนิคโอโออาร์ทีในการตรวจสอบซอฟต์แวร์ในระยะเวลาจำกัด สามารถนำงานวิจัยนี้ไปใช้ประโยชน์ต่อไปได้

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 ความนำ

งานวิจัยนี้เป็นการวิจัยเพื่อ (1) วิเคราะห์การกระจายตัวของประเภทข้อบกพร่องที่พบโดยใช้เทคนิคโอโออาร์ที (OORT) ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยวิธีเชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล (UML) และ (2) เปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ (Software Inspection) โดยใช้เทคนิคการอ่านซอฟต์แวร์ (Software Reading Technique) 3 เทคนิค คือ เทคนิคซีบีอาร์ (CBR) และ เทคนิคโอโออาร์ที (OORT) โดยเทคนิคโอโออาร์ทีที่ใช้ในการตรวจสอบ ประกอบด้วย 2 ประเภท คือ เทคนิคโอโออาร์ทีเดิม และเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

โดยนำผลของการวิเคราะห์ข้อมูล โดยพิจารณาจำนวนข้อบกพร่องที่สามารถตรวจพบในแต่ละขั้นตอนของเทคนิคโอโออาร์ที และพิจารณาจำนวนข้อบกพร่องที่มีในเอกสารการออกแบบซอฟต์แวร์แต่ละประเภทที่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ที เพื่อใช้เป็นข้อมูลในการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีลง โดยจะนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ไปตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ ที่ออกแบบโดยแผนภาพยูเอ็มแอล และนำผลการตรวจสอบที่ได้ไปเปรียบเทียบกับผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ที่ได้จากงานวิจัยของ อุมาพร นิลเอวะ (2549)

โดยทฤษฎีที่เกี่ยวข้องกับงานวิจัยมีดังต่อไปนี้ คุณภาพซอฟต์แวร์ (Software Quality) การควบคุมคุณภาพซอฟต์แวร์ (Software Quality Control) การทบทวนซอฟต์แวร์ (Software Review) การตรวจสอบซอฟต์แวร์ (Software Inspection) เทคนิคอ่านซอฟต์แวร์ (Software Reading Technique) ประเภทของข้อบกพร่องที่มีในเอกสารการออกแบบซอฟต์แวร์ และภาษายูเอ็มแอล

2.2 คุณภาพซอฟต์แวร์ (Software Quality)

มีผู้ให้คำนิยามความหมายของคุณภาพซอฟต์แวร์ไว้แตกต่างกัน โดยขึ้นอยู่กับมุมมองของแต่ละคน ดังนี้ คุณภาพซอฟต์แวร์ คือ (1) การผลิตซอฟต์แวร์ให้ตรงตามความต้องการ (United States Department of Defense, 1988) (2) การผลิตซอฟต์แวร์ให้เหมาะสมต่อการใช้งาน (Schulmeyer และ Mcmanus, 1992) (3) การผลิตซอฟต์แวร์ให้มีความสอดคล้องกันในเรื่องของหน้าที่การทำงานและประสิทธิภาพในการทำงาน รวมถึงเอกสารในการพัฒนาที่มีมาตรฐาน

(Pressman, 1997) สำหรับงานวิจัยนี้จะนิยามความหมายของ คุณภาพซอฟต์แวร์ หมายถึง การพัฒนาซอฟต์แวร์ได้อย่างถูกต้องตรงตามความต้องการของลูกค้าหรือผู้ใช้งาน และควรมีคุณสมบัติของคุณภาพตามที่กำหนดไว้ด้วย (บงกชธร เพ็ชรนิจินดา, 2548)

2.2.1 คุณสมบัติของคุณภาพซอฟต์แวร์ เป็นสิ่งที่บ่งบอกว่าซอฟต์แวร์ที่ได้มีคุณภาพประกอบด้วย (บงกชธร เพ็ชรนิจินดา, 2548)

1. ความปลอดภัย (Security) หมายถึง ซอฟต์แวร์ต้องสามารถป้องกันการเข้าใช้งานโดยไม่ได้รับอนุญาตได้

2. ประสิทธิภาพสูง (High Performance) หมายถึง ซอฟต์แวร์สามารถทำงานได้ตามที่กำหนดไว้ในเอกสารความต้องการ

3. เชื่อถือได้ (Reliability) หมายถึง ซอฟต์แวร์สามารถทำงานตามหน้าที่ได้อย่างถูกต้องในสิ่งแวดล้อมที่กำหนดไว้

4. ความถูกต้อง (Correctness) หมายถึง ข้อมูลต่างๆที่ใช้ในระบบมีความถูกต้องและซอฟต์แวร์สามารถตอบสนองความต้องการของลูกค้าหรือผู้ใช้ได้อย่างถูกต้อง

5. ความสามารถใช้งานได้ (Usability) หมายถึง การที่ผู้ใช้งานสามารถเข้าใจการทำงานและสามารถใช้งานซอฟต์แวร์ได้ง่าย

6. การนำกลับมาใช้ใหม่ได้ (Reusability) หมายถึง ซอฟต์แวร์สามารถนำไปประยุกต์ใช้งานร่วมกับโปรแกรมอื่นได้ หรือสามารถนำไปใช้งานในสิ่งแวดล้อมอื่นได้

7. บำรุงรักษาได้โดยง่าย (Maintainability) หมายถึง ซอฟต์แวร์ที่สามารถปรับปรุงได้เมื่อนำไปติดตั้งใช้งานจริงแล้ว

8. ทำงานร่วมกับระบบอื่นได้ (Interoperability) หมายถึง ซอฟต์แวร์สามารถทำงานร่วมกับระบบอื่นได้

9. สามารถโยกย้ายได้ (Portability) หมายถึง สามารถโยกย้ายซอฟต์แวร์เพื่อไปใช้งานในสิ่งแวดล้อมอื่นได้อย่างสะดวก

10. เปลี่ยนแปลงแก้ไขได้ (Flexibility) หมายถึง ซอฟต์แวร์สามารถเปลี่ยนแปลงแก้ไขได้

11. รองรับปริมาณความต้องการใช้งานได้ (Robustness) หมายถึง ซอฟต์แวร์สามารถทำตามความต้องการของผู้ใช้งานได้

12. สามารถทดสอบได้ (Testability) หมายถึง ซอฟต์แวร์สามารถทดสอบและสร้างกรณีทดสอบได้เพื่อรับประกันการทำงานของซอฟต์แวร์

2.3 การควบคุมคุณภาพซอฟต์แวร์ (Software Quality Control)

การควบคุมคุณภาพซอฟต์แวร์ หมายถึง การตรวจสอบกระบวนการพัฒนาซอฟต์แวร์ เพื่อควบคุมให้ซอฟต์แวร์ที่พัฒนาได้ตรงตามความต้องการและเหมาะสมต่อการใช้งาน เพื่อให้ซอฟต์แวร์มีคุณภาพตามที่กำหนดไว้ โดยการควบคุมคุณภาพซอฟต์แวร์ประกอบด้วย 2 กิจกรรมหลัก คือ (O'Regan, 2003)

1. การทบทวนซอฟต์แวร์ (Software Review) เป็นขั้นตอนสำคัญในการควบคุมคุณภาพซอฟต์แวร์ โดยเป็นการตรวจสอบซอฟต์แวร์เพื่อค้นหาและระบุข้อบกพร่อง การทบทวนซอฟต์แวร์สามารถทำได้ในตอนท้ายของแต่ละขั้นตอนการพัฒนาซอฟต์แวร์ โดยตรวจสอบสิ่งที่ได้จากแต่ละขั้นตอนในการพัฒนา เช่น เอกสารอธิบายความต้องการ เอกสารการออกแบบ โปรแกรม เป็นต้น โดยคำนึงถึงความถูกต้องตรงตามความต้องการและความสอดคล้องกันของสิ่งที่ตรวจสอบ (O'Regan, 2003)

2. การทดสอบซอฟต์แวร์ (Software Testing) เป็นอีกหนึ่งขั้นตอนในการควบคุมคุณภาพซอฟต์แวร์ เป็นกระบวนการที่ทำให้เกิดความถูกต้อง สมบูรณ์ และปลอดภัย เพื่อให้เกิดคุณภาพของซอฟต์แวร์ โดยการทดสอบจะตรวจสอบข้อบกพร่องให้มั่นใจว่าซอฟต์แวร์ที่พัฒนามีคุณภาพตรงตามความต้องการ โดยการทดสอบจะทำหลังจากซอฟต์แวร์ถูกพัฒนาเรียบร้อยแล้ว (บงกชธร เพ็ชรนิจินดา, 2548)

2.4 การทบทวนซอฟต์แวร์ (Software Review)

การทบทวนซอฟต์แวร์เป็นกระบวนการสำคัญที่กระทำระหว่างการพัฒนาซอฟต์แวร์ โดยเป็นกระบวนการในการตรวจสอบเพื่อค้นหาและระบุข้อบกพร่องที่เกิดขึ้นระหว่างการพัฒนาซอฟต์แวร์ เพื่อให้ได้ซอฟต์แวร์ที่ตรงตามความต้องการที่กำหนดไว้ โดยการทบทวนซอฟต์แวร์มีหลายรูปแบบที่แตกต่างกันที่กระบวนการในการทบทวน โดยสามารถแบ่งได้ 3 ประเภท ดังนี้ (Horch, 1996)

1. การทบทวนระดับเดียวกัน (Peer Review) เป็นการทบทวนซอฟต์แวร์ที่มีความเป็นทางการน้อยที่สุด เป็นการตั้งคำถามเกี่ยวกับผลที่ได้จากการพัฒนาซอฟต์แวร์ที่ต้องการทบทวน เพื่อตรวจสอบให้มั่นใจในความถูกต้องของซอฟต์แวร์ โดยผลที่ได้จากการทบทวนระดับเดียวกันมักจะไม่บันทึกเป็นเอกสาร จะเป็นคำพูดที่ได้จากการสอบถามเป็นส่วนใหญ่ (Horch, 1996)

2. การตรวจตลอด (Walkthrough) เป็นวิธีทบทวนซอฟต์แวร์ที่ผู้ออกแบบซอฟต์แวร์ (designer) หรือ ผู้เขียนโปรแกรม (Programmer) นำสมาชิกในคณะพัฒนาหรือผู้ที่มีส่วนเกี่ยวข้องที่มีความสนใจมาตรวจสอบสิ่งที่กำลังพัฒนาในขั้นตอนนั้น อาจจะเป็นโปรแกรมหรือเอกสารต่างๆที่

ใช้ในการพัฒนาซอฟต์แวร์ เช่น เอกสารอธิบายกระบวนการในการพัฒนา เอกสารการออกแบบ หรือ เอกสารที่ใช้ในการทดสอบ เป็นต้น จะใช้วิธีถามคำถามและผู้ตรวจสอบสามารถแสดงความเห็นเกี่ยวกับข้อบกพร่องที่เป็นไปได้ เช่น การละเมิดมาตรฐานการพัฒนา หรือปัญหาอื่นๆ โดยตรงต่อผู้พัฒนา วิธีนี้ผู้ตรวจสอบต้องมีความเข้าใจรายละเอียดของซอฟต์แวร์ก่อนที่จะตรวจสอบ การตรวจตลอดเป็นวิธีที่ยังไม่มีขั้นตอนในการตรวจสอบที่ชัดเจน มีเพียงกระบวนการหลักๆ คือ การค้นหาและบันทึกข้อบกพร่องที่พบ อาจจะไม่สามารถนำไปใช้ในการปรับปรุงการทำงานในการพัฒนาซอฟต์แวร์ครั้งต่อๆไปได้ (IEEE, 1997)

3. การตรวจสอบซอฟต์แวร์ (Software Inspection) เป็นวิธีที่มีความเป็นทางการในกระบวนการทบทวนมากที่สุด มีตำแหน่งหน้าที่ในการทำงานของคณะตรวจสอบอย่างชัดเจน โดยคณะตรวจสอบทุกคนต้องมีความเข้าใจในงานที่รับผิดชอบและต้องทราบถึงกระบวนการต่างๆที่ใช้ในการตรวจสอบด้วย ขั้นตอนในการตรวจสอบจะถูกแบ่งอย่างชัดเจน และแต่ละขั้นตอนจะมีวัตถุประสงค์ในการทำงานของตนเอง (IEEE, 1997)

2.5 การตรวจสอบซอฟต์แวร์ (Software Inspection)

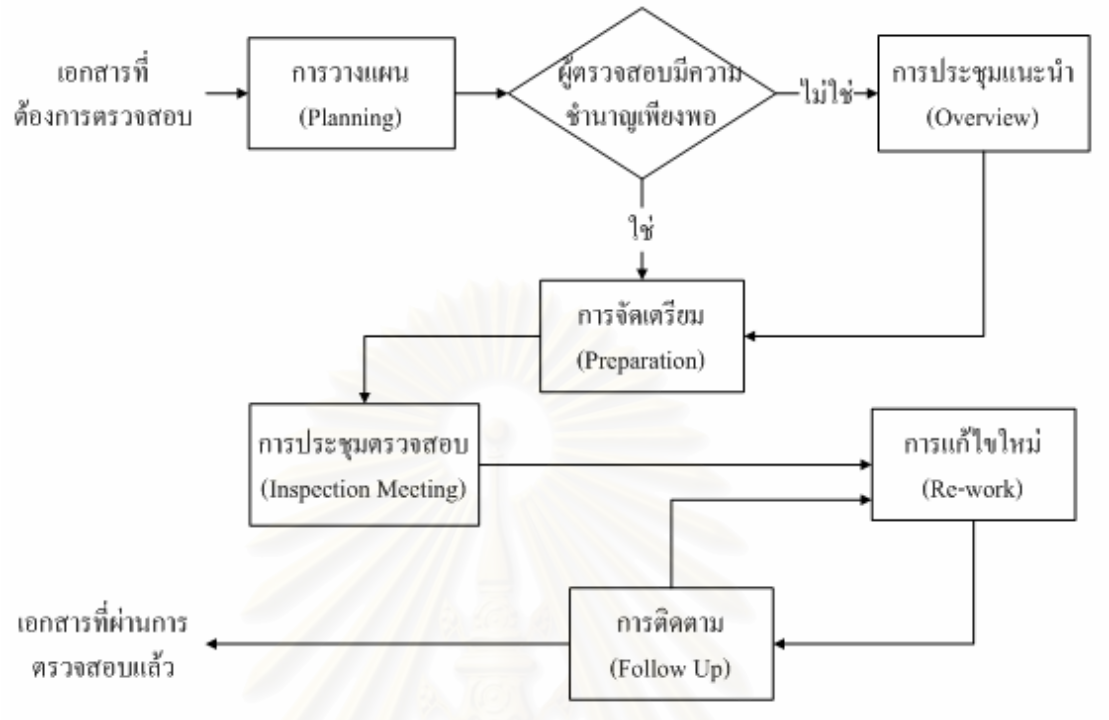
การตรวจสอบซอฟต์แวร์ได้รับการเผยแพร่ครั้งแรกโดย Fagan (1976) และได้ใช้งานกันอย่างแพร่หลายเรื่อยมา การตรวจสอบซอฟต์แวร์มีประโยชน์ในการพัฒนาซอฟต์แวร์อย่างมาก โดยมีส่วนสำคัญในการเพิ่มคุณภาพของซอฟต์แวร์ โดยการตรวจสอบซอฟต์แวร์ช่วยลดค่าใช้จ่ายในการพัฒนาด้วย เนื่องจากการตรวจสอบซอฟต์แวร์เป็นการค้นหาข้อบกพร่องของขั้นตอนการพัฒนาซอฟต์แวร์ก่อนที่จะดำเนินการพัฒนาในขั้นตอนต่อไป ทำให้สามารถช่วยลดจำนวนข้อบกพร่องอื่นๆที่จะตามมาได้และยังช่วยลดค่าใช้จ่ายในการแก้ไขข้อบกพร่องด้วย (Laitenberger และคณะ, 2000) โดยสามารถลดได้เท่าใดขึ้นอยู่กับขั้นตอนที่ตรวจพบข้อบกพร่อง ได้มีผู้วิจัยและสรุปผลว่าค่าเฉลี่ยในการตรวจสอบโปรแกรมสามารถช่วยลดค่าใช้จ่ายในการค้นหาข้อบกพร่องได้ 39% และการตรวจสอบเอกสารการออกแบบสามารถช่วยลดค่าใช้จ่ายในการค้นหาข้อบกพร่องได้ 44% (Laitenberger และคณะ, 2000) การตรวจสอบซอฟต์แวร์สามารถค้นหาและแก้ไขข้อบกพร่องของเอกสารซอฟต์แวร์ได้ 50% ถึง 90% ของข้อบกพร่องในเอกสารซอฟต์แวร์ (Fagan, 1976; Gilb and Graham, 1993) ในการตรวจสอบซอฟต์แวร์จะตรวจสอบสิ่งที่ได้จากแต่ละขั้นตอนการพัฒนาซอฟต์แวร์ เช่น เอกสารอธิบายความต้องการ เอกสารการออกแบบ โปรแกรมที่ได้จากการพัฒนาหรือเอกสารต่างๆที่ใช้ในการทดสอบซอฟต์แวร์ โดยวัตถุประสงค์สำคัญของการตรวจสอบซอฟต์แวร์คือการค้นหาข้อบกพร่องจากสิ่งที่ตรวจสอบ (O'Regan, 2003) งานวิจัยนี้จะให้ความสำคัญกับการตรวจสอบซอฟต์แวร์ในขั้นตอนการออกแบบ โดยในการตรวจสอบจะ

ตรวจสอบความสอดคล้องกันของเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ และตรวจสอบความถูกต้องของเอกสารการออกแบบโดยเปรียบเทียบกับเอกสารอธิบายความต้องการที่ได้จากขั้นตอนการวิเคราะห์ความต้องการ

2.5.1 คณะตรวจสอบซอฟต์แวร์ (Software Inspection Team) คณะตรวจสอบซอฟต์แวร์เป็นบุคคลต่างๆที่ทำงานร่วมกันในการวิเคราะห์ผลลัพธ์ของแต่ละขั้นตอนในการพัฒนาซอฟต์แวร์เพื่อค้นหาและแก้ไขข้อบกพร่อง คณะตรวจสอบซอฟต์แวร์ประกอบด้วย (Frankovich, 1995)

1. ผู้เขียนเอกสาร (Author) คือ ผู้พัฒนาซอฟต์แวร์และจัดทำเป็นเอกสารขึ้น โดยต้องมีความเข้าใจในหลักการในการพัฒนาซอฟต์แวร์และเข้าใจซอฟต์แวร์ที่พัฒนาอย่างชัดเจน และมีหน้าที่อธิบายเอกสารที่นำมาตรวจสอบและตอบข้อสงสัยต่างๆของผู้ตรวจสอบ และแก้ไขเอกสารตามที่การตรวจสอบระบุไว้
2. ผู้ดำเนินรายการ (Moderator) เป็นผู้ที่ควบคุมให้การดำเนินการตรวจสอบเป็นไปตามขั้นตอนที่กำหนดไว้ รวมถึงการควบคุมให้ผู้อื่นในคณะตรวจสอบทำหน้าที่ของตนอย่างเต็มความสามารถ หน้าที่สำคัญของผู้ดำเนินรายการ คือ ต้องตรวจสอบเอกสารที่นำมาตรวจสอบให้ครบถ้วนและถูกต้อง จัดเตรียมสถานที่และกำหนดวันเวลาที่ใช้ในการตรวจสอบ กำหนดหน้าที่ความรับผิดชอบของคณะตรวจสอบ และตรวจสอบเอกสารที่แก้ไขแล้วว่าถูกต้องหรือไม่
3. ผู้อ่าน (Reader) เป็นผู้ทำหน้าที่อ่านเอกสารที่ตรวจสอบโดยละเอียด เพื่ออธิบายให้ผู้ที่อยู่ในคณะตรวจสอบเข้าใจถึงสิ่งที่ต้องตรวจสอบในระยะเวลาอันสั้น
4. ผู้บันทึก (Recorder) เป็นผู้ที่บันทึกข้อบกพร่องทั้งหมดที่สรุปได้จากขั้นตอนการประชุมตรวจสอบ ต้องบันทึกด้วยว่าข้อบกพร่องที่พบเป็นข้อบกพร่องประเภทใดและตรวจพบที่ใดของเอกสาร
5. ผู้ตรวจสอบ (Inspector) เป็นผู้ที่ทำหน้าที่ในการตรวจสอบซอฟต์แวร์ ถือว่าเป็นอิสระจากหน้าที่อื่น คือผู้ที่ทำหน้าที่อื่นสามารถทำหน้าที่นี้ด้วยได้ ผู้ตรวจสอบมีหน้าที่ในการวิเคราะห์ ค้นหาและระบุข้อบกพร่องในเอกสารที่ตรวจสอบ

2.5.2 ขั้นตอนการตรวจสอบซอฟต์แวร์ Fagan (1976) กำหนดขั้นตอนในการตรวจสอบซอฟต์แวร์ไว้ 6 ขั้นตอน (อ้างถึงใน Frankovich, 1995) ผู้วิจัยได้ปรับปรุงโดยเพิ่มเงื่อนไขในการเลือกดำเนินการหลังขั้นตอนที่ 1 (การวางแผน (Planning)) เพื่อให้มีความชัดเจนยิ่งขึ้น



รูปที่ 2-1 แสดงขั้นตอนการตรวจสอบซอฟต์แวร์

1. การวางแผน (Planning) ในการตรวจสอบซอฟต์แวร์การวางแผนการดำเนินการเป็นสิ่งที่ต้องกระทำอย่างระมัดระวัง เพื่อให้การตรวจสอบซอฟต์แวร์สามารถดำเนินการได้ตามขั้นตอนและระยะเวลาที่เหมาะสม รวมถึงการกำหนดและอธิบายบทบาทหน้าที่ให้คณะตรวจสอบทุกคนด้วย (Frankovich, 1995)

การวางแผนเริ่มเมื่อผู้เขียนจัดทำเอกสารที่ต้องการตรวจสอบเสร็จเรียบร้อยแล้วและแจ้งต่อผู้ดำเนินรายการ ผู้ดำเนินรายการจะตรวจเอกสารที่ต้องการตรวจสอบและเอกสารต่างๆที่เกี่ยวข้องว่าพร้อมที่จะดำเนินการตรวจสอบซอฟต์แวร์หรือไม่ ถ้าเอกสารไม่ครบผู้ดำเนินรายการจะแจ้งผู้เขียนถึงความไม่พร้อมของเอกสาร เพื่อให้จัดเตรียมเพิ่มเติมสำหรับการตรวจสอบซอฟต์แวร์ การตรวจเอกสารให้ถูกต้องเป็นสิ่งที่สำคัญมากเพราะส่งผลต่อเวลาและค่าใช้จ่ายต่างๆที่ใช้ในการตรวจสอบซอฟต์แวร์ หลังจากตรวจเอกสารเรียบร้อยแล้วผู้ดำเนินรายการจะคัดเลือกคณะตรวจสอบและกำหนดวันเวลาและสถานที่ที่ใช้ในการตรวจสอบทุกขั้นตอน และพิจารณาว่าผู้ตรวจสอบมีความรู้และความเข้าใจในขั้นตอนการตรวจสอบและเอกสารที่ตรวจสอบดีแล้วหรือไม่ เพื่อพิจารณาว่าต้องทำขั้นตอนการประชุมแนะนำหรือไม่

กระบวนการดำเนินการในขั้นตอนการวางแผนมีดังนี้

1 ตรวจเอกสารที่ต้องการตรวจสอบ

2 เลือกสมาชิกในคณะตรวจสอบและกำหนดหน้าที่ความรับผิดชอบ และพิจารณาว่าต้องทำขั้นตอนการประชุมแนะนำหรือไม่

3 กำหนดวันเวลาและสถานที่ที่ใช้ในการตรวจสอบ

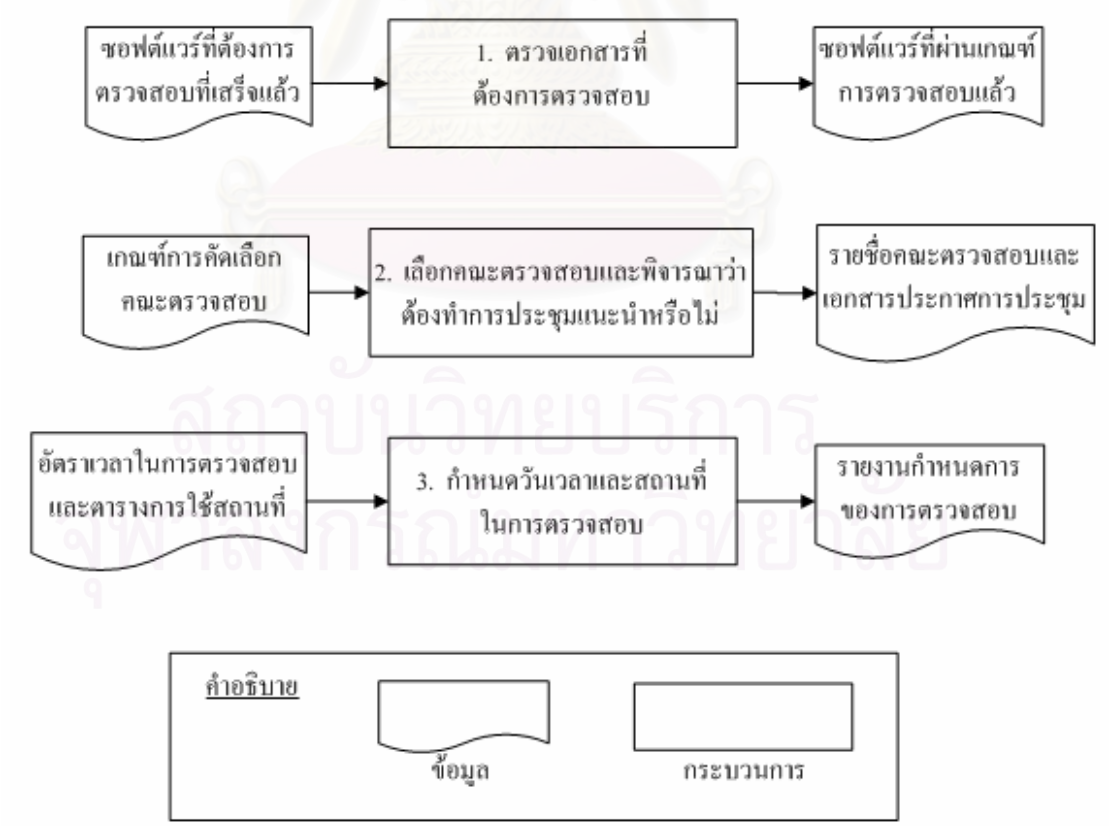
คณะตรวจสอบที่มีความเกี่ยวข้องในขั้นตอนการวางแผน ประกอบด้วย ผู้ดำเนินรายการ และผู้เขียนเอกสาร โดยมีหน้าที่ความรับผิดชอบดังนี้

ผู้ดำเนินรายการ มีหน้าที่ดังนี้

- ตรวจเอกสารที่ต้องการตรวจสอบ
- เลือกสมาชิกในคณะตรวจสอบ
- ตัดสินใจว่าต้องทำขั้นตอนการประชุมแนะนำหรือไม่
- จัดเตรียมกำหนดการในการตรวจสอบ

ผู้เขียนเอกสาร มีหน้าที่ดังนี้

- ทบทวนเอกสารที่จะตรวจสอบร่วมกับผู้ดำเนินรายการ
- ช่วยผู้ดำเนินรายการในการเลือกสมาชิกในคณะตรวจสอบ



รูปที่ 2-2 แสดงขั้นตอนการวางแผนของการตรวจสอบซอฟต์แวร์ (Frankovich, 1995)

2. การประชุมแนะนำ (Overview) วัตถุประสงค์หลักของการตรวจสอบซอฟต์แวร์ คือการวิเคราะห์เอกสารเพื่อค้นหาและกำจัดข้อบกพร่องให้ได้มากที่สุด โดยจะทำได้ต้องมีความรู้และความเข้าใจในเอกสารที่ตรวจสอบอย่างดี และถ้าผู้ดำเนินรายการคิดว่าผู้ตรวจสอบยังมีความรู้และความเข้าใจในซอฟต์แวร์ไม่พอ จึงสมควรต้องทำขั้นตอนการประชุมแนะนำเพื่อให้ข้อมูลกับผู้ตรวจสอบ โดยผู้เขียนเอกสารจะอธิบายเอกสารที่จะตรวจสอบโดยละเอียดถึงหน้าที่ของเอกสารนั้นและเทคโนโลยีที่ใช้เพื่อให้ผู้ตรวจสอบเข้าใจ

กระบวนการดำเนินการในขั้นตอนการประชุมแนะนำมีดังนี้

1 ผู้ดำเนินรายการจัดการประชุมแนะนำ

2 ผู้เขียนเอกสารอธิบายรายละเอียดซอฟต์แวร์

คณะตรวจสอบที่มีความเกี่ยวข้องในขั้นตอนการประชุมแนะนำ ประกอบด้วย ผู้ดำเนินรายการ ผู้เขียนเอกสาร และผู้ตรวจสอบ โดยมีหน้าที่ความรับผิดชอบดังนี้

ผู้ดำเนินรายการ มีหน้าที่ดังนี้

- จัดการประชุมแนะนำโดยต้องแลกเปลี่ยนข้อมูลเพื่อให้ผู้ตรวจสอบเข้าใจในเอกสารที่ตรวจสอบมากที่สุด

ผู้เขียนเอกสาร มีหน้าที่ดังนี้

- อธิบายรายละเอียดต่างๆของซอฟต์แวร์
- ตอบข้อสงสัยต่างๆที่เกิดในการประชุมแนะนำที่เกี่ยวกับซอฟต์แวร์

ผู้ตรวจสอบ มีหน้าที่ดังนี้

- ถามคำถามเพื่อให้เข้าใจซอฟต์แวร์ที่จะตรวจสอบทั้งหมด

รูปที่ 2-3 แสดงขั้นตอนการประชุมแนะนำของการตรวจสอบซอฟต์แวร์ (Frankovich, 1995)

3. การจัดเตรียม (Preparation) เป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนตรวจสอบซอฟต์แวร์ เพื่อค้นหาข้อบกพร่อง เป็นขั้นตอนที่สามารถนำเทคนิคการอ่านซอฟต์แวร์มาช่วยค้นหาและระบุข้อบกพร่องได้ โดยเทคนิคการอ่านซอฟต์แวร์มีหลายรูปแบบที่มีขั้นตอนการทำงานที่แตกต่างกันให้ผู้ตรวจสอบเลือก เพื่อให้เหมาะสมกับซอฟต์แวร์ที่ต้องการตรวจสอบ

กระบวนการดำเนินการในขั้นตอนการจัดเตรียมมีดังนี้

1. ศึกษาซอฟต์แวร์ที่จะตรวจสอบ
2. ตรวจสอบซอฟต์แวร์ โดยค้นหาและระบุข้อบกพร่องของซอฟต์แวร์ตามเทคนิคที่เลือกใช้ แล้วบันทึกข้อบกพร่องและระยะเวลาที่ใช้ในการตรวจสอบในเอกสารบันทึกรายการข้อบกพร่อง

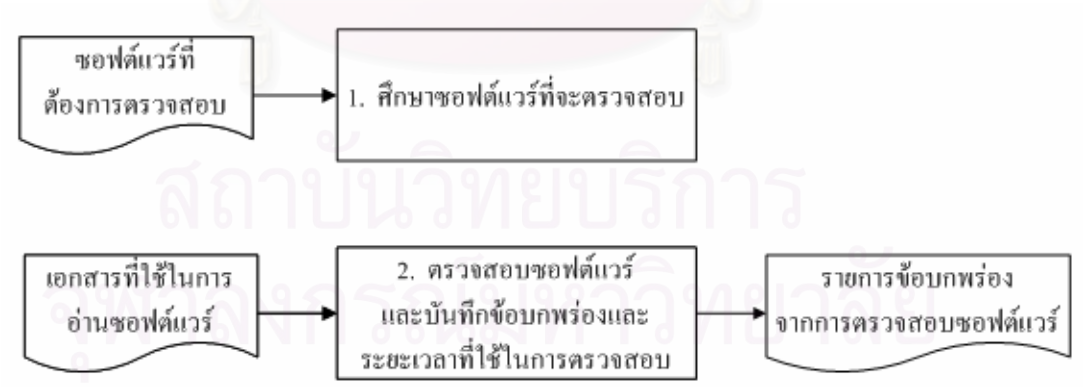
คณะตรวจสอบที่มีความเกี่ยวข้องในขั้นตอนการจัดเตรียม ประกอบด้วย ผู้ดำเนินรายการ และผู้ตรวจสอบ โดยมีหน้าที่ความรับผิดชอบดังนี้

ผู้ดำเนินรายการ มีหน้าที่ดังนี้

- ควบคุมการจัดเตรียมให้สามารถดำเนินการได้อย่างเรียบร้อย

ผู้ตรวจสอบ มีหน้าที่ดังนี้

- ศึกษาซอฟต์แวร์ที่จะตรวจสอบ
- ระบุข้อบกพร่องที่มีในซอฟต์แวร์ตามเทคนิคที่เลือกใช้
- บันทึกข้อบกพร่องและระยะเวลาที่ใช้ในการตรวจสอบ



รูปที่ 2-4 แสดงขั้นตอนการจัดเตรียมของการตรวจสอบซอฟต์แวร์ (Frankovich, 1995)

4. การประชุมตรวจสอบ (Inspection Meeting) เป็นขั้นตอนที่ทุกคนในคณะตรวจสอบต้องเข้าประชุมร่วมกัน เพื่ออภิปรายเกี่ยวกับข้อบกพร่องที่ผู้ตรวจสอบแต่ละคนสามารถตรวจพบได้จากขั้นตอนการจัดเตรียม โดยเริ่มจากผู้ดำเนินรายการตรวจสอบเอกสารทั้งหมดที่ได้

จากขั้นตอนการจัดเตรียม แล้วผู้อ่านเอกสารอ่านรายละเอียดของเอกสารที่ใช้ในการตรวจสอบทั้งหมด เพื่อให้ผู้ตรวจสอบแต่ละคนพิจารณาและแจ้งกับสมาชิกคนอื่นเมื่อถึงส่วนที่ตนตรวจพบข้อบกพร่อง และเมื่อผู้ตรวจสอบบอกถึงข้อบกพร่องเกิดขึ้น ทุกคนพิจารณาข้อบกพร่องนั้นและอภิปรายว่าเป็นข้อบกพร่องที่แท้จริงหรือไม่

กระบวนการดำเนินการในขั้นตอนการประชุมตรวจสอบ มีดังนี้

1. อ่านเอกสารที่ตรวจสอบ
2. ระบุและบันทึกข้อบกพร่องในเอกสารบันทึกรายการข้อบกพร่อง
3. ทบทวนข้อบกพร่องทั้งหมดในเอกสารบันทึกรายการข้อบกพร่อง

คณะตรวจสอบที่มีความเกี่ยวข้องในขั้นตอนการประชุมตรวจสอบ ประกอบด้วย ผู้ดำเนินรายการ ผู้เขียนเอกสาร ผู้อ่าน ผู้ตรวจสอบ และผู้บันทึกโดยมีหน้าที่ความรับผิดชอบดังนี้

ผู้ดำเนินรายการ มีหน้าที่ดังนี้

- ควบคุมการประชุมตรวจสอบให้สามารถดำเนินการได้อย่างเรียบร้อย

ผู้เขียนเอกสาร มีหน้าที่ดังนี้

- ฟังการอธิบายถึงข้อบกพร่องที่ผู้ตรวจสอบพบ
- ร่วมอภิปรายเกี่ยวกับข้อบกพร่องที่ไม่น่าจะเป็นข้อบกพร่องจริง

ผู้อ่าน มีหน้าที่ดังนี้

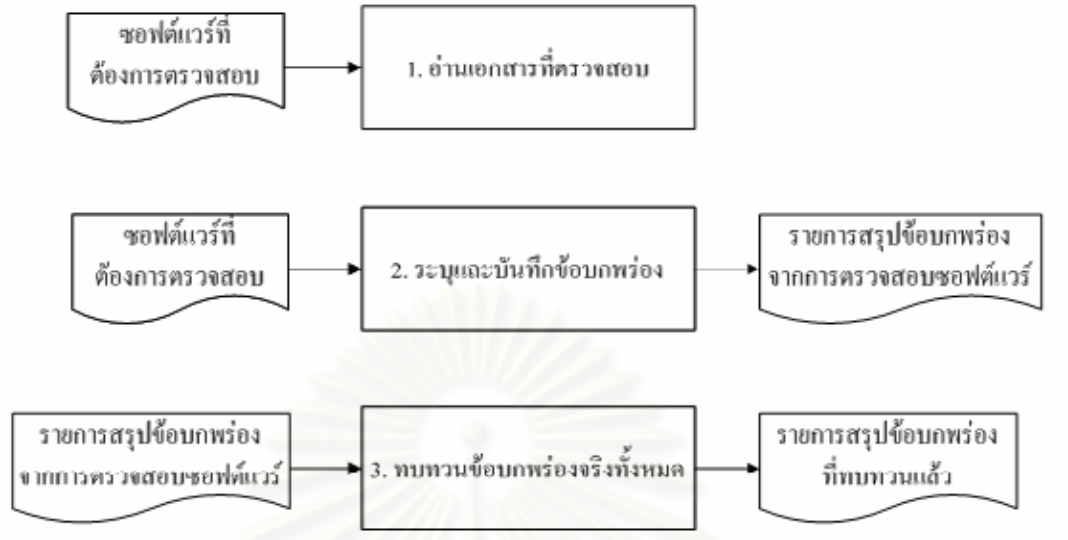
- เลือกการเรียงลำดับการอ่านเอกสารให้เหมาะสม
- ตอบคำถามต่างๆจากคณะตรวจสอบ

ผู้ตรวจสอบ มีหน้าที่ดังนี้

- ระบุข้อบกพร่องที่มีในซอฟต์แวร์ที่ตรวจพบได้อย่างถูกต้อง
- อภิปรายเกี่ยวกับข้อบกพร่องที่พบได้

ผู้บันทึก มีหน้าที่ดังนี้

- จดบันทึกรายละเอียดทั้งหมดของข้อบกพร่องที่ได้จากการประชุมตรวจสอบ ในเอกสารบันทึกรายการข้อบกพร่อง



รูปที่ 2-5 แสดงขั้นตอนการประชุมตรวจสอบของการตรวจสอบซอฟต์แวร์ (Frankovich, 1995)

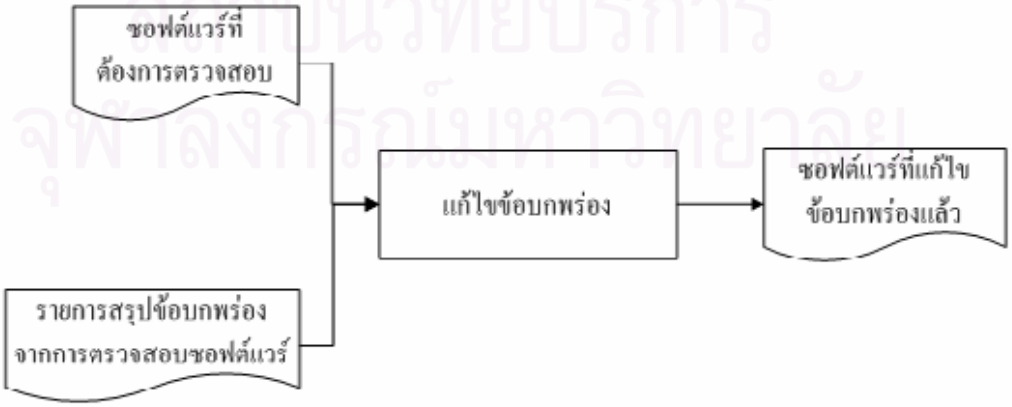
5. การแก้ไขใหม่ (Re-work) เป็นขั้นตอนที่ผู้เขียนเอกสารแก้ไขข้อบกพร่องที่ได้ที่บันทึกในเอกสารสรุปรายการข้อบกพร่องจากการตรวจสอบซอฟต์แวร์

กระบวนการดำเนินการในขั้นตอนการแก้ไขใหม่ มีดังนี้

1. แก้ไขข้อบกพร่องที่บันทึกในเอกสารสรุปรายการข้อบกพร่องจากการตรวจสอบซอฟต์แวร์

คณะตรวจสอบที่มีความเกี่ยวข้องในขั้นตอนการแก้ไขใหม่ ได้แก่ ผู้เขียนเอกสาร โดยมีหน้าที่ความรับผิดชอบดังนี้

- แก้ไขข้อบกพร่อง
- แจ้งผู้ดำเนินรายการเมื่อแก้ไขข้อบกพร่องเสร็จแล้ว



รูปที่ 2-6 แสดงขั้นตอนการแก้ไขใหม่ของการตรวจสอบซอฟต์แวร์ (Frankovich, 1995)

6. การติดตาม (Follow Up) เมื่อซอฟต์แวร์ทั้งหมดที่ผู้เขียนเอกสารแก้ไขเสร็จแล้ว จะได้รับการตรวจสอบว่าแก้ไขข้อบกพร่องครบถ้วนและถูกต้องหรือไม่

กระบวนการดำเนินการในขั้นตอนการติดตาม มีดังนี้

1. ผู้ดำเนินรายการตรวจสอบซอฟต์แวร์ว่าได้แก้ไขข้อบกพร่องที่บันทึกในเอกสารบันทึกรายการข้อบกพร่องครบถ้วนและถูกต้องหรือไม่ ถ้าซอฟต์แวร์แก้ไขเสร็จแล้วก็จะถือว่าผ่านการตรวจสอบ แต่ถ้ายังแก้ไขข้อบกพร่องไม่ครบถ้วนต้องส่งไปให้ขั้นตอนการแก้ไขใหม่แก้ไขอีกครั้ง

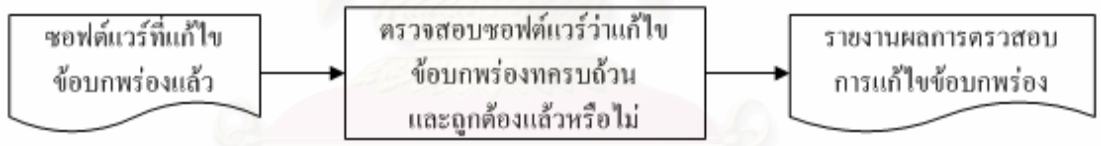
คณะตรวจสอบที่มีความเกี่ยวข้องในขั้นตอนการติดตาม ประกอบด้วย ผู้ดำเนินรายการ และผู้เขียนเอกสาร โดยมีหน้าที่ความรับผิดชอบดังนี้

ผู้ดำเนินรายการ มีหน้าที่ดังนี้

- พิจารณาซอฟต์แวร์ว่าแก้ไขครบถ้วนและถูกต้องตามที่ระบุไว้หรือไม่
- บันทึกว่าซอฟต์แวร์ได้ผ่านการตรวจสอบเรียบร้อยแล้ว

ผู้เขียนเอกสาร มีหน้าที่ดังนี้

- เตรียมตัวในการแก้ไขงานใหม่ ถ้ายังคงมีข้อบกพร่องอยู่



รูปที่ 2-7 แสดงขั้นตอนการติดตามของการตรวจสอบซอฟต์แวร์ (Frankovich, 1995)

2.6 เทคนิคการอ่านซอฟต์แวร์ (Software Reading Technique)

การตรวจสอบซอฟต์แวร์เป็นกระบวนการที่ช่วยเพิ่มประสิทธิภาพของซอฟต์แวร์ หัวใจที่สำคัญที่สุดประการหนึ่งของการตรวจสอบซอฟต์แวร์อยู่ที่ขั้นตอนการจัดเตรียม (Preparation) (Laitenberger และคณะ, 2000) โดยเป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนค้นหาข้อบกพร่องในเอกสารที่ต้องการตรวจสอบ ที่เป็นวัตถุประสงค์หลักของการตรวจสอบซอฟต์แวร์เพื่อให้เอกสารที่ตรวจสอบมีความถูกต้องตามความต้องการ มีความสอดคล้องกันกับเอกสารอื่นที่นำเสนอระบบเดียวกัน ได้อธิบายอย่างชัดเจน และมีความสมบูรณ์ วิธีที่ช่วยในการค้นหาข้อบกพร่องในขั้นตอนการจัดเตรียม คือ การใช้เทคนิคการอ่านซอฟต์แวร์ในการค้นหาข้อบกพร่องของซอฟต์แวร์ (Travassos และคณะ, 2002) เทคนิคการอ่านซอฟต์แวร์สามารถช่วยเพิ่มประสิทธิภาพในการ

ตรวจสอบซอฟต์แวร์ได้ โดยจะมีคำแนะนำในการตรวจสอบซอฟต์แวร์ (Guideline) ให้ผู้ตรวจสอบเพื่อใช้เป็นแนวทางในการค้นหาและระบุข้อบกพร่องในขั้นตอนการจัดเตรียม ไม่จำกัดเฉพาะการตรวจสอบโปรแกรมเท่านั้นแต่สามารถใช้ในการตรวจสอบเอกสารอย่างอื่นได้ด้วย (Porter และคณะ, 1995; Basili และคณะ, 1996; Fusaro และคณะ, 1997; Shull , 1998; Zhang และคณะ, 1998) โดยจะมีคำอธิบายถึงขั้นตอนในการค้นหาข้อบกพร่อง และวิธีในการตรวจสอบว่าควรจะสนใจในส่วนใดเป็นพิเศษ และจะมีคำถามให้ผู้ตรวจสอบเพื่อพิจารณาข้อมูลที่มีในการค้นหาและระบุข้อบกพร่อง เพื่อให้ผู้ตรวจสอบสามารถค้นหาข้อบกพร่องให้ได้มากที่สุด (Basili และคณะ, 1996)

2.6.1 ประเภทของเทคนิคการอ่านซอฟต์แวร์ สามารถแบ่งได้หลายประเภทตามรูปแบบคำแนะนำที่แตกต่างกัน ดังนี้

1. เทคนิคเฉพาะกิจ (Ad-hoc) เป็นวิธีที่ง่ายที่สุด โดยไม่เตรียมคำแนะนำให้ผู้ตรวจสอบว่าต้องตรวจสอบอย่างไรหรือต้องพิจารณาส่วนใดเป็นพิเศษระหว่างการตรวจสอบ โดยผู้ตรวจสอบจะต้องใช้วิจารณญาณ ความรู้และประสบการณ์ของตนเองในการพิจารณาว่าควรจะค้นหาข้อบกพร่องอย่างไรในสิ่งที่ตรวจสอบ (Laitenberger และคณะ, 2000)

2. เทคนิคการอ่านบนพื้นฐานของเช็กลิสต์ (Checklist-Based Reading) หรือเรียกโดยย่อว่าเทคนิคซีบีอาร์ (CBR) เป็นเทคนิคที่นิยมนำมาใช้ในการตรวจสอบซอฟต์แวร์ (Laitenberger และคณะ, 2000) โดยเทคนิคนี้ได้จัดเตรียมคำแนะนำที่อยู่ในรูปแบบของเช็กลิสต์ (Checklist) เป็นรายการคำถามแบบใช่ หรือ ไม่ใช่ (Yes/No Questions) เพื่อช่วยให้ผู้ตรวจสอบทราบว่าจะต้องมองหาข้อบกพร่องที่ส่วนใดในเอกสาร (Chernak, 1996) โดยรายการคำถามถูกตั้งขึ้นเพื่อรองรับการค้นหาข้อบกพร่องประเภทที่เคยพบในเอกสารรูปแบบเดียวกันในอดีต เช็กลิสต์ที่ใช้มีหลายรูปแบบแตกต่างกันที่คำถามที่ใช้ในการตรวจสอบซอฟต์แวร์ การตรวจสอบซอฟต์แวร์ต้องเลือกเช็กลิสต์ที่เหมาะกับสิ่งที่ต้องการตรวจสอบด้วย เนื่องจากคำถามในเช็กลิสต์เกิดจากการพิจารณาข้อบกพร่องในอดีต ดังนั้นถ้าประเภทสิ่งที่ต้องการตรวจสอบแตกต่างกันต้องเลือกเช็กลิสต์ที่ต่างกันด้วย โดยการตอบคำถามถ้าคำตอบเป็นปฏิเสธแสดงว่ามีข้อบกพร่อง และผู้ตรวจสอบจะต้องใส่รายละเอียดเกี่ยวกับข้อบกพร่องที่พบลงในเอกสารบันทึกรายการข้อบกพร่อง สำหรับลักษณะของคำถามในเช็กลิสต์ เป็นคำถามแบบทั่วไปไม่มีความเฉพาะเจาะจงสำหรับแต่ละสภาพแวดล้อม

ถึงแม้ว่าเทคนิคซีบีอาร์จะเป็นเทคนิคการอ่านซอฟต์แวร์ที่มีประสิทธิภาพมากกว่าเทคนิคเฉพาะกิจ แต่ Parnas และคณะ (2003) ได้กล่าวไว้ว่าเทคนิคซีบีอาร์ยังมีข้อบกพร่องอยู่ด้วยกัน 4 ข้อ ดังนี้ (1) จากที่ได้กล่าวมาแล้วว่าคำถามในเช็กลิสต์ได้มาจากการพิจารณาข้อบกพร่องที่พบในอดีต ดังนั้นผู้ตรวจสอบจะไม่ได้ให้ความสำคัญกับส่วนอื่นที่อยู่นอกเหนือจากในเช็กลิสต์ ทำให้ข้อบกพร่องบางข้อถูกละเลยไป (2) เช็กลิสต์มักประกอบด้วยรายการคำถามที่มากเกินไป และ

ไม่ได้บอกวิธีในการตอบคำถาม ดังนั้นจึงไม่ชัดเจนสำหรับผู้ตรวจสอบในการนำไปใช้ (3) จากข้อบกพร่องข้อที่ 2 จะเห็นว่าผู้ตรวจสอบอาจจะบันทึกคำตอบของคำถามได้ไม่ถูกต้องหรือไม่ครบถ้วน ทำให้ในขั้นตอนของการประชุมตรวจสอบที่จะนำผลของการตรวจสอบมาอภิปรายอาจจะทำได้ไม่ถูกต้อง และ (4) ผู้ตรวจสอบต้องตรวจสอบข้อมูลทุกอย่างในเอกสารเพื่อหาข้อบกพร่องที่อาจเกิดขึ้นได้ตามคำถามในเช็กลิสต์ เป็นสาเหตุให้ผู้ตรวจสอบรับภาระจากรายละเอียดที่ไม่จำเป็นจำนวนมาก จากปัญหาที่เช็กลิสต์ประกอบด้วยรายการคำถามที่มากเกินไป Gilb และ Graham (1993) จึงได้แนะนำว่าเช็กลิสต์ไม่ควรมีความยาวเกิน 1หน้ากระดาษ นั่นหมายถึงจำนวนคำถามไม่ควรมีเกิน 25 ข้อ

3. เทคนิคการอ่านบนพื้นฐานของสถานการณ์ (Scenario-Based Reading) หรือเรียกโดยย่อว่าเทคนิคเอสบีอาร์ (SBR) เป็นวิธีที่เตรียมคำแนะนำให้กับผู้ตรวจสอบโดยจะแสดงเป็นสถานการณ์ (Scenario) เพื่อบอกว่าต้องตรวจสอบอะไร และตรวจสอบอย่างไรในเอกสารที่ตรวจสอบ (Basili และคณะ, 1996) โดยสถานการณ์ (Scenario) จะประกอบด้วย 3 ส่วน (Porter และ Votta, 1994) คือ (1) คำแนะนำเกี่ยวกับหน้าที่ความรับผิดชอบ และสิ่งที่ต้องเกี่ยวข้องในแต่ละหน้าที่ (2) วิธีในการคัดเลือกข้อมูลที่สำคัญออกมาจากเอกสารที่ตรวจสอบ (3) คำถามที่สามารถตอบได้ด้วยข้อมูลที่คัดออกมา โดยเทคนิคเอสบีอาร์สามารถแบ่งออกเป็น 5 รูปแบบตามสถานการณ์ที่ต่างกัน ดังนี้

ก. เทคนิคการอ่านบนพื้นฐานของข้อบกพร่อง (Defect-Based Reading) หรือเรียกโดยย่อว่าเทคนิคดีบีอาร์ (DBR) Porter และคณะ (1995) อธิบายสถานการณ์บนพื้นฐานของการจัดกลุ่มข้อบกพร่องที่กำหนดขึ้นสำหรับใช้ในการตรวจสอบซอฟต์แวร์ โดยสถานการณ์เหล่านี้ได้มาจากประเภทของข้อบกพร่องที่กำหนดไว้ ประกอบด้วยรายการคำถามที่เป็นแบบเฉพาะเจาะจง ผู้ตรวจสอบต้องตอบคำถามไปพร้อมกับที่กำลังอ่านเอกสารที่ตรวจสอบ โดยสถานการณ์ (Scenario) จะมุ่งเน้นไปที่การค้นหาข้อบกพร่องว่าเป็นข้อบกพร่องประเภทใดตามที่กำหนดไว้

ข. เทคนิคการอ่านบนพื้นฐานของสถานการณ์ที่อยู่บนพื้นของฟังก์ชันพอยต์ (Scenario-based Reading Technique Based on Function Points) โดย Cheng และ Jeffery (1996) ได้วางรากฐานการพัฒนาสถานการณ์บนพื้นฐานของการวิเคราะห์ด้วยวิธีฟังก์ชันพอยต์ (Function Point Analysis) สถานการณ์ของฟังก์ชันพอยต์ (Function Point Scenario) จะประกอบด้วยรายการคำถามที่เป็นองค์ประกอบแบบฟังก์ชันพอยต์ (Function Point Element)

ค. เทคนิคการอ่านบนพื้นฐานของการใช้ (Usage-Based Reading) หรือเรียกโดยย่อว่าเทคนิคยูบีอาร์ (UBR) Thelin และคณะ (2003) กล่าวไว้ว่าเป็นเทคนิคที่สถานการณ์

มุ่งประเด็นไปที่แรงงานที่ใช้ในการค้นหาข้อบกพร่อง และคำนึงถึงเหตุผลที่ว่าข้อบกพร่องแต่ละประเภทมีความสำคัญที่แตกต่างกัน จึงจัดลำดับความสำคัญในการค้นหาข้อบกพร่องว่าข้อบกพร่องใดที่มีความวิกฤต โดยดูจากมุมมองของผู้ใช้

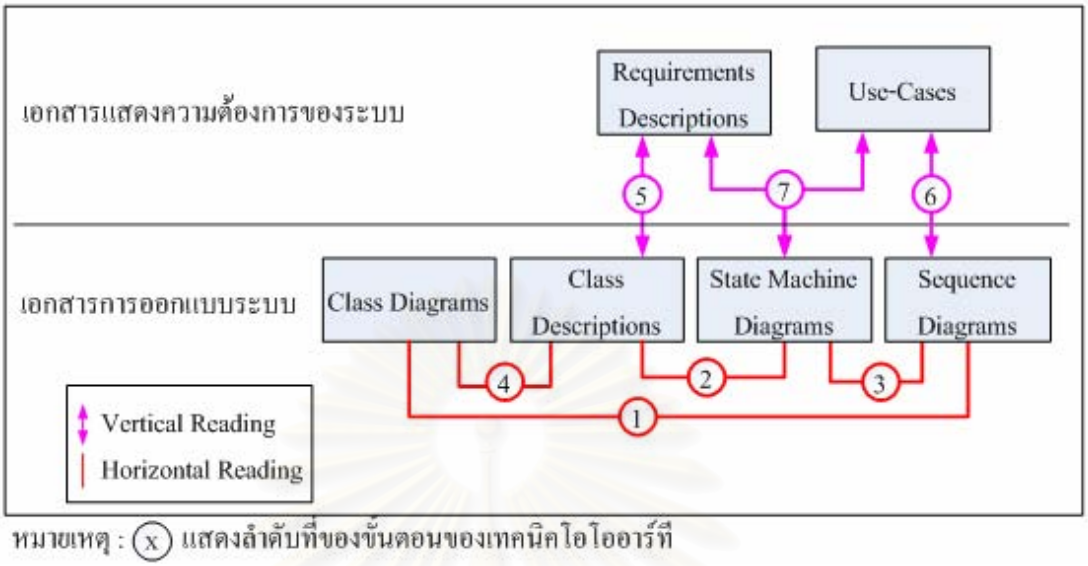
ง. เทคนิคการอ่านบนพื้นฐานของมุมมอง (Perspective Based Reading) หรือเรียกโดยย่อว่าเทคนิคพีบีอาร์ (PBR) Basili และคณะ (1996) ได้กำหนดว่าขั้นตอนการดำเนินการเพื่อรองรับการตรวจสอบเอกสารตามมุมมองของผู้เกี่ยวข้องแต่ละหน้าที่ โดยมีวัตถุประสงค์เพื่อค้นหาข้อบกพร่องที่ต่างมุมมองกัน โดยสถานการณ์ของเทคนิคนี้รองรับการตรวจสอบเอกสารจากมุมมองของผู้ถือผลประโยชน์ร่วม (Stakeholders) แต่ละคน โดย Fowler (1986) ระบุว่ามุมมองของผู้ถือผลประโยชน์ร่วมที่ให้ความสำคัญในการตรวจสอบประกอบด้วย ผู้ใช้งานซอฟต์แวร์ ผู้ออกแบบซอฟต์แวร์ ผู้ทดสอบซอฟต์แวร์ ผู้พัฒนาซอฟต์แวร์ เทคนิคนี้ได้แนวความคิดจากบทความเกี่ยวกับการตรวจสอบซอฟต์แวร์ในอดีต เช่น Fagan (1976) รายงานว่าส่วนหนึ่งของโปรแกรมควรถูกตรวจสอบโดยผู้ทดสอบระบบ ขณะที่ Fowler (1986) แนะนำว่าคณะตรวจสอบแต่ละคนควรมีมุมมองเฉพาะตัวสำหรับแต่ละสิ่งที่นำมาตรวจสอบ นอกจากนี้ Basili และคณะ (1996) Laitenberger และคณะ (2000) Biffl และ Halling (2002) กล่าวว่าผลิตภัณฑ์ที่นำมาตรวจสอบควรได้รับการตรวจสอบจากมุมมองของผู้ถือผลประโยชน์ร่วมที่แตกต่างกัน โดยมุมมองขึ้นกับบทบาทของแต่ละคนในการพัฒนาซอฟต์แวร์

มีงานวิจัยที่เกี่ยวข้องกับการวัดประสิทธิผลของการนำเทคนิคพีบีอาร์มาใช้ในการตรวจสอบซอฟต์แวร์ โดยการเปรียบเทียบกับเทคนิคซีบีอาร์ มีหลายงานวิจัยที่แสดงให้เห็นว่าเทคนิคพีบีอาร์มีประสิทธิผลมากกว่าเทคนิคซีบีอาร์ เช่น จากงานวิจัยของ Basili และคณะ (1996) กับ Porter และ Votta (1998) ที่ได้ทดลองโดยการตรวจสอบเอกสารแสดงความต้องการซอฟต์แวร์ งานวิจัยของ Laitenberger และคณะ (2000) ทดลองโดยการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่พัฒนาด้วยวิธีเชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล และงานวิจัยของ Laitenberger และคณะ (2000) ทดลองโดยการตรวจสอบโปรแกรม ผลของการทดลองเหล่านี้แสดงให้เห็นว่าเทคนิคพีบีอาร์มีประสิทธิผลในการตรวจสอบซอฟต์แวร์มากกว่าเทคนิคซีบีอาร์

จ. เทคนิคการอ่านบนพื้นฐานของความสามารถติดตาม (Traceability-Based Reading) หรือเรียกโดยย่อว่าเทคนิคทีบีอาร์ (TBR) สามารถเรียกอีกชื่อหนึ่งว่า เทคนิคการอ่านเชิงวัตถุ (Object-Oriented Reading Technique) หรือเรียกโดยย่อว่า เทคนิคโอโออาร์ที (OORT) โดย Basili (1996) กล่าวว่าเทคนิคในการอ่านซอฟต์แวร์ส่วนใหญ่จะถูกพัฒนาขึ้นเพื่อรองรับการพัฒนาซอฟต์แวร์ด้วยวิธีเชิงโครงสร้าง (Structure Development Process) เป็นการพัฒนาซอฟต์แวร์ในรูปแบบที่ใช้กันมานานแล้ว มีลักษณะเป็นเอกสารที่ใช้ตัวอักษรในการบรรยาย Booch (1999)

กล่าวว่าถ้าเป็นการพัฒนาซอฟต์แวร์ด้วยวิธีเชิงวัตถุ เอกสารการออกแบบจะอยู่ในรูปแบบของแผนภาพเพื่อให้มีความเข้าใจง่ายขึ้น ภาษาที่นิยมใช้ในการออกแบบคือภาษายูเอ็มแอล และการพัฒนาซอฟต์แวร์ด้วยวิธีเชิงวัตถุได้เริ่มมาแทนที่แบบโครงสร้างแล้ว ทำให้เทคนิคการอ่านซอฟต์แวร์ที่ใช้ในการค้นหาข้อบกพร่องในการตรวจสอบซอฟต์แวร์ต้องปรับเปลี่ยนตามไปด้วย ด้วยเหตุผลนี้ Travassos และคณะ (1999) ได้เสนอเทคนิคการอ่านซอฟต์แวร์ที่เหมาะกับการออกแบบในการพัฒนาด้วยวิธีเชิงวัตถุโดยเฉพาะ โดยกล่าวว่า เทคนิคนี้มีแนวคิดหลักเพื่อให้มั่นใจในเรื่องความถูกต้องสอดคล้องกันของเอกสารทั้งในเอกสารเดียวกัน และระหว่างเอกสารที่เสนอระบบเดียวกันเพื่อให้มั่นใจในความถูกต้องและความสมบูรณ์ (Completeness) ปัจจุบันเทคนิคโอโออาร์ทีเป็นรุ่น 3.0

Shull และคณะ (1999) กล่าวว่าปัญหาหลักที่ทำให้เกิดเทคนิคโอโออาร์ทีมีอยู่ 2 ประการ คือ (1) เอกสารที่ใช้ในการออกแบบในแผนภาพที่ต่างกันสามารถอธิบายระบบเดียวกันให้ออกมาในทิศทางเดียวกันหรือไม่ (2) เอกสารที่ออกแบบถูกต้องตรงตามความต้องการที่กำหนดไว้หรือไม่ จากปัญหาเหล่านี้จึงนิยามวิธีการจับคู่แผนภาพ เพื่อนำมาเปรียบเทียบในการค้นหาข้อบกพร่องโดยแบ่งออกเป็น 2 ส่วนหลักคือ (1) การอ่านตามแนวนอน (Horizontal Reading) ช่วยในการตรวจสอบว่าแต่ละแผนภาพอธิบายระบบเดียวกัน กล่าวคือแต่ละแผนภาพที่ออกแบบมีความสอดคล้องตรงกันหรือไม่ (2) การอ่านตามแนวตั้ง (Vertical Reading) ช่วยในการตรวจสอบว่าแผนภาพที่ออกแบบนำเสนอระบบที่ถูกต้อง กล่าวคือออกแบบได้ตรงตามความต้องการของซอฟต์แวร์กำหนดไว้ ในเอกสารอธิบายระบบหรือไม่ โดยดูได้จากเอกสารความต้องการของระบบ (Requirement Description) ที่เก็บข้อมูลจากความต้องการของลูกค้า แผนภาพยูสเคส (Use Case Diagram) และเอกสารอธิบายยูสเคส (Use Case Description) แสดงดังรูปที่ 2-8



รูปที่ 2-8 แสดงรูปแบบการอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ของเทคนิคโอโออาร์ที (Travassos และคณะ, 1999)

เทคนิคโอโออาร์ทีเป็นเทคนิคในการตรวจสอบเพื่อค้นหาข้อบกพร่อง โดยนำเอกสารที่ใช้ในการอธิบายระบบ ได้แก่ เอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) และแผนภาพยูสเคส (Use Case Diagram) และเอกสารอธิบายยูสเคส (Use Case Description) และเอกสารการออกแบบซอฟต์แวร์ ได้แก่ แผนภาพคลาส (Class Diagram) คำอธิบายคลาส (Class Description) แผนภาพซีเควนซ์ (Sequence Diagram) และแผนภาพสถานะ (State Machine Diagram) มาตรวจสอบหาข้อบกพร่อง โดยการเปรียบเทียบทีละ 2 แผนภาพหรือเอกสาร โดยเปรียบเทียบทั้งหมด 7 คู่ โดยเทคนิคโอโออาร์ทีได้แบ่งการเปรียบเทียบออกเป็น 7 ขั้นตอนในการอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ดังรูปที่ 2-8 โดยมีรายละเอียดดังต่อไปนี้

(1) ขั้นตอนการเปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพคลาส

(2) ขั้นตอนการเปรียบเทียบระหว่างแผนภาพสถานะและเอกสารอธิบายคลาส

(3) ขั้นตอนการเปรียบเทียบระหว่างแผนภาพสถานะ และแผนภาพซีเควนซ์

(4) ขั้นตอนการเปรียบเทียบระหว่างแผนภาพคลาส และเอกสารอธิบายคลาส

⑤ ขั้นตอนการเปรียบเทียบระหว่างเอกสารอธิบายคลาส และเอกสารอธิบายความต้องการซอฟต์แวร์

⑥ ขั้นตอนการเปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพยูสเคส พร้อมทั้งเอกสารอธิบายยูสเคส

⑦ ขั้นตอนการเปรียบเทียบระหว่างแผนภาพสถานะ และเอกสารอธิบายความต้องการซอฟต์แวร์ / แผนภาพยูสเคส พร้อมทั้งเอกสารอธิบายยูสเคส

มีงานวิจัยที่เกี่ยวข้องกับการเปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคโอโออาร์ที โดย Travassos และคณะ (1999) ทดลองเพื่อวัดความสามารถในการค้นหาข้อบกพร่องในเอกสารการออกแบบ โดยใช้เทคนิคการอ่านซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเพื่อเพิ่มคุณภาพของซอฟต์แวร์ ผลการทดลองสรุปว่าเทคนิคโอโออาร์ทีสามารถทำให้ผู้ทดลองพึงพอใจผลการค้นหาข้อบกพร่องจากเทคนิคการอ่านนี้ โดยสามารถหาข้อบกพร่องได้โดยเฉลี่ย 56% ของข้อบกพร่อง อาจจะต้องใช้เวลาตรวจสอบมากแต่ว่ามีเอกสารแนะนำเพื่อให้ผู้ตรวจสอบเข้าใจขั้นตอนการตรวจสอบ ทำให้สามารถค้นหาข้อบกพร่องได้มาก เป็นที่พอใจของผู้ตรวจสอบ

อุมาพร นิลเอวะ (2549) ได้เปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคโอโออาร์ทีกับเทคนิคซีบีอาร์ โดยนำมาใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล ผลการทดลองแสดงว่าเทคนิโอโออาร์ทีไม่ได้มีประสิทธิภาพและประสิทธิผลมากกว่าเทคนิคซีบีอาร์ อาจจะเกิดจากระยะเวลาในการตรวจสอบที่มีจำกัด ทำให้เทคนิคโอโออาร์ทีที่มีขั้นตอนการตรวจสอบที่มากกว่าสามารถตรวจสอบข้อบกพร่องได้น้อยกว่าเทคนิคซีบีอาร์ โดยได้กำหนดระยะเวลาของขั้นตอนการจัดเตรียมในการตรวจสอบซอฟต์แวร์ เพื่อให้ผู้ตรวจสอบแต่ละคนค้นหาข้อบกพร่องไว้ 90 นาที อาจจะเพียงพอสำหรับการค้นหาข้อบกพร่องตามขั้นตอนของเทคนิคซีบีอาร์ที่มีเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์เพียง 1 หน้ากระดาษ แต่ไม่เพียงพอสำหรับเทคนิคโอโออาร์ทีที่มีเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่ค่อนข้างมาก

2.7 ประเภทของข้อบกพร่องที่พบในเอกสารการออกแบบซอฟต์แวร์

การออกแบบซอฟต์แวร์เริ่มทำหลังจากที่เก็บข้อมูลในส่วนของความต้องการระบบแล้ว การออกแบบซอฟต์แวร์ด้วยวิธีเชิงวัตถุเหมือนเป็นการอธิบายความต้องการของระบบในอีกรูปแบบหนึ่งโดยจะอยู่ในรูปแบบของแผนภาพ (Travassos และคณะ, 1999) เพื่อแสดงให้ผู้ที่จะนำเอกสารการออกแบบไปใช้ในการพัฒนาซอฟต์แวร์เข้าใจถึงความต้องการได้ง่ายขึ้น นอกเหนือจากการอ่านจากเอกสารความต้องการโดยตรง ดังนั้นการออกแบบจึงเป็นส่วนที่สำคัญมากในการพัฒนา

ซอฟต์แวร์ การตรวจสอบซอฟต์แวร์ในส่วนของการออกแบบเป็นสิ่งที่สำคัญ เพราะการค้นหาข้อบกพร่องได้ในขั้นตอนการออกแบบสามารถลดค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ได้ถึง 44% (Laitenberger และคณะ, 2000)

ข้อบกพร่องจากขั้นตอนการออกแบบสามารถแบ่งได้หลายประเภท โดยจะขึ้นอยู่กับหลักการที่ใช้ในการกำหนดกลุ่มของประเภทของข้อบกพร่อง หลักการในการแบ่งส่วนใหญ่จะได้จากผลการวิจัยเกี่ยวกับการตรวจสอบซอฟต์แวร์ เป็นการค้นหาข้อบกพร่องต่างๆที่มีในเอกสารการออกแบบ สามารถแบ่งได้ดังนี้

1. ประเภทข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส ได้จากงานวิจัยของ Lange และ Chaundron (2006) โดยมีรายละเอียดดังนี้

- ข้อความไม่มีชื่อ (Message without Name :EnN) : ในแผนภาพซีเควนซ์อ็อบเจ็กต์จะแลกเปลี่ยนข้อความต่อกัน โดยลูกศรที่ส่งต้องมีคำอธิบายเป็นข้อความแสดงถึงชื่อของเมท็อดของอ็อบเจ็กต์ที่ลูกศรชี้ไปแสดงอยู่ ถ้าไม่มีแสดงว่าเป็นข้อบกพร่อง
- ข้อความไม่มีวิธีดำเนินการ (Message without Mothod :EcM) : ข้อความที่ส่งจากอ็อบเจ็กต์หนึ่งไปอีกอ็อบเจ็กต์หนึ่งจะแสดงว่าเป็นการเรียกใช้เมท็อดของอ็อบเจ็กต์นั้น โดยชื่อข้อความต้องเป็นชื่อเมท็อดของอ็อบเจ็กต์นั้น ถ้าไม่ตรงก็ถือว่าเป็นข้อบกพร่อง
- ข้อความส่งผิดทิศทาง (Message in the wrong direction : ED) : ถ้าข้อความเรียกจากคลาส A ไปคลาส B แต่ว่าชื่อเมท็อดเป็นของคลาส A แทนที่จะเป็นของคลาส B แสดงว่าเป็นการไม่สอดคล้องกันระหว่างชื่อข้อความและชื่อเมท็อด
- คลาสไม่มีในแผนภาพซีเควนซ์ (Class not instantiated in SD : CnSD) : อ็อบเจ็กต์ในแผนภาพซีเควนซ์ควรแสดงถึงคลาสในแผนภาพคลาส ถ้าไม่มีคลาสในแผนภาพซีเควนซ์แต่มีในแผนภาพคลาส ถือว่าเป็นข้อบกพร่องประเภทนี้
- อ็อบเจ็กต์ที่ไม่มีคลาสในแผนภาพคลาส (Object has no Class in CD : CnCD) : มีอ็อบเจ็กต์ในแผนภาพซีเควนซ์ แต่ว่าไม่มีในแผนภาพคลาส
- ยูสเคสไม่มีแผนภาพซีเควนซ์ (Use Case without SD : UCnSD) : แผนภาพซีเควนซ์เขียนตามคำแนะนำของแผนภาพยูสเคส แต่อาจจะเกิดข้อบกพร่องขึ้นได้ถ้าคำแนะนำของแผนภาพยูสเคสไม่ได้เขียนในแผนภาพซีเควนซ์ทั้งหมด

- การเรียกใช้คลาสเดียวกันในแผนภาพเดียวกันมากกว่า 1 ครั้ง (Multiple definitions of class with equal names : Cm) : ถ้าใน 1 แผนภาพ มีคลาสที่มีชื่อเดียวกันมากกว่า 1 คลาสถือว่าเป็นข้อบกพร่อง
- เมท็อดไม่ถูกเรียกในแผนภาพซีเควนซ์ (Method not called in SD : MnSD) : มีเมท็อดในแผนภาพคลาสที่ไม่มีในแผนภาพซีเวนซ์

งานวิจัยของ Lange และ Chaundron (2006) เป็นการวิจัยเพื่อวัดผลกระทบของข้อบกพร่องในแผนภาพยูเอ็มแอล ระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส ที่เป็นเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุ และได้สรุปผลของการค้นหาข้อบกพร่องไว้ว่า ประเภทข้อบกพร่องที่ตรวจพบมากที่สุดเป็น ประเภทคลาสไม่มีในแผนภาพซีเควนซ์ รองลงมาเป็นประเภทข้อความไม่มีชื่อ และประเภทข้อบกพร่องที่สามารถตรวจพบน้อยที่สุดเป็น ประเภทการเรียกใช้คลาสเดียวกันในแผนภาพเดียวกันมากกว่า 1 ครั้ง การแบ่งประเภทข้อบกพร่องในกลุ่มนี้มีลักษณะการแบ่งโดยพิจารณาถึงหลักในการเขียนแผนภาพยูเอ็มแอล เพื่อให้ถูกต้องตามหลักการของภาษายูเอ็มแอล

2. ประเภทข้อบกพร่องที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง กล่าวไว้ในงานวิจัยของ Travassos และคณะ(1999) โดยมีรายละเอียดดังนี้

- การละเลย (Omission) : เอกสารการออกแบบที่ตรวจสอบไม่ได้แสดงข้อมูลที่จำเป็นที่แสดงในเอกสารอธิบายความต้องการซอฟต์แวร์
- ข้อมูลไม่เป็นความจริง (Incorrect Fact) : เอกสารการออกแบบแสดงข้อมูลไม่ตรงกับเอกสารอธิบายความต้องการซอฟต์แวร์
- ข้อมูลไม่สอดคล้องกัน (Inconsistency) : เอกสารการออกแบบต่างประเภทกันแสดงข้อมูลไม่สอดคล้องกัน
- ข้อมูลไม่ชัดเจน (Ambiguity) : เอกสารการออกแบบแสดงข้อมูลไม่ชัดเจน ทำให้ผู้ที่จะนำไปใช้อาจจะตีความผิด หรือเข้าใจผิดได้
- แสดงข้อมูลที่ไม่จำเป็น (Extraneous Information) : เอกสารการออกแบบแสดงข้อมูลที่ไม่จำเป็นหรือไม่เกี่ยวข้องกับการออกแบบ

งานวิจัยนี้ใช้เทคนิคโอโออาร์ทีในการค้นหาข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ ประกอบด้วย แผนภาพคลาสที่มีจำนวน 7 คลาส แผนภาพสถานะที่มีจำนวน 3 แผนภาพ และแผนภาพซีเควนซ์ที่มีจำนวน 5 แผนภาพ ประเภทข้อบกพร่องที่พบสามารถตรวจพบได้ทั้งในส่วนของการอ่านแนวนอน ที่เปรียบเทียบเอกสารการออกแบบซอฟต์แวร์ต่างประเภทกัน และการอ่านแนวตั้ง ที่เปรียบเทียบเอกสารการอ่านซอฟต์แวร์กับเอกสารความต้องการ เพื่อพิจารณา

ความถูกต้องและความต้องกันของการออกแบบ โดยข้อบกพร่องที่ค้นหาได้จากการอ่านแนวนอนและการอ่านแนวตั้งได้ปริมาณใกล้เคียงกัน โดยการอ่านแนวนอนสามารถตรวจพบข้อบกพร่องเฉลี่ยประมาณ 11.7 ข้อบกพร่อง และการอ่านแนวตั้งสามารถตรวจพบข้อบกพร่องเฉลี่ย 10.4 ข้อบกพร่อง โดยการอ่านแนวนอนสามารถตรวจพบประเภทข้อบกพร่องในส่วนของข้อมูลไม่ชัดเจนและข้อมูลไม่สอดคล้องกันมากที่สุด และการอ่านแนวตั้งสามารถตรวจพบประเภทข้อบกพร่องในส่วนของ การละเลยและข้อมูลไม่เป็นจริงมากที่สุด

3. ประเภทข้อบกพร่องที่พิจารณาจากความผิดพลาดในการนำเสนอหน้าที่การทำงานของข้อมูล เผยแพร่โดย Basili และคณะ (1996)

- การขาดหน้าที่ (Missing Functionally or Missing Feature : MF) คือ เอกสารการออกแบบแสดงข้อมูลในส่วนของหน้าที่การทำงานหรือคุณสมบัติของซอฟต์แวร์ที่ระบุในเอกสารอธิยานความต้องการของระบบไม่ครบถ้วน
- การติดต่อขาดหายไป (Missing Interface :MI) เอกสารการออกแบบแสดงข้อมูลที่ระบบจะติดต่อสื่อสารกับสิ่งที่อยู่นอกขอบเขตของระบบถูกละเลย
- การขาดคุณสมบัติเฉพาะ (Missing Performance : MP) เอกสารการออกแบบแสดงข้อมูลในส่วนของคุณสมบัติเฉพาะถูกละเลย อาจจะทำให้ไม่สามารถผ่านการทดสอบระบบได้
- สิ่งแวดล้อมขาดหายไป (Missing Environment : ME) เอกสารการออกแบบแสดงข้อมูลที่อธิบายความต้องการในส่วนของ ฮาร์ดแวร์ (hardware) ซอฟต์แวร์ (software) ฐานข้อมูล หรือบุคคลที่เกี่ยวข้องในระบบถูกละเลย
- ข้อมูลไม่ชัดเจน (Ambiguous Information : WA) เอกสารการออกแบบแสดงการอธิบายการทำงานของระบบไม่ชัดเจนทำให้เกิดการสับสนหรือเข้าใจผิดได้ ในที่นี้ไม่ได้หมายถึงภาษาที่ใช้อธิบายเท่านั้น แต่รวมถึงการทำงานของระบบที่ออกแบบด้วย
- ข้อมูลไม่สอดคล้องกัน (Inconsistent Information : WI) เอกสารการออกแบบที่ต่างปะเภทกัน มีความไม่สอดคล้องกันของข้อมูลที่ตรวจสอบ

งานวิจัยของ Basili และคณะ (1996) ที่เป็นการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคพีบีอาร์ พบว่า ประเภทของข้อบกพร่องที่พบมากที่สุด คือ ข้อมูลไม่สอดคล้องกัน รองลงมาคือ การขาดหน้าที่ และประเภทที่พบน้อยที่สุดคือ การขาดคุณสมบัติเฉพาะ

4. ประเภทข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง เปิดเผยโดย Strauss และ Ebenau (1994) โดยสามารถแบ่งออกได้ดังต่อไปนี้

- สิ่งสูญหาย (Missing) คือ เอกสารการออกแบบที่นำมาตรวจสอบมีบางสิ่งที่จำเป็นขาดหายไป
- สิ่งผิดพลาด (Wrong) คือ เอกสารการออกแบบที่นำมาตรวจสอบมีบางสิ่งที่ผิดพลาด หรือไม่มีความชัดเจน
- สิ่งเพิ่มเติม (Extra) คือเอกสารการออกแบบที่นำมาตรวจสอบมีบางสิ่งเพิ่มเข้ามาเป็นสิ่งที่ไม่จำเป็นต้องนำมาใช้ในการออกแบบจึงควรกำจัดออกไป

จากการวิจัยของ Strauss และ Ebenau (1994) พบว่าประเภทข้อบกพร่องที่สามารถตรวจพบมากที่สุด คือ สิ่งผิดพลาด รองลงมาคือ สิ่งสูญหาย และที่พบน้อยที่สุดคือ สิ่งเพิ่มเติม แต่สิ่งสูญหายก็เป็นประเภทข้อบกพร่องที่สามารถตรวจพบง่ายที่สุดด้วย

จากประเภทข้อบกพร่องที่แบ่งเป็น 4 กลุ่มดังกล่าว ประเภทข้อบกพร่องตามกลุ่มที่ 1 ที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส (Lange และ Chaundron, 2006) กลุ่มที่ 2 ที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง (Travassos และคณะ, 1999) และ กลุ่มที่ 4 ที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง (Strauss และ Ebenau, 1994) เป็นผลการวิจัยที่เกี่ยวข้องกับเอกสารการออกแบบซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอลโดยตรง แต่ประเภทข้อบกพร่องตามกลุ่มที่ 3 ที่พิจารณาจากความผิดพลาดในการนำเสนอหน้าที่การทำงานของข้อมูล (Basili และคณะ, 1996) เกี่ยวข้องกับเอกสารการออกแบบซอฟต์แวร์ ที่ไม่ได้ระบุชัดเจนว่าเป็นการออกแบบโดยใช้แผนภาพยูเอ็มแอล

ถ้าพิจารณาประเภทข้อบกพร่องทั้งหมดพบว่า ประเภทข้อบกพร่องตามกลุ่มที่ 3 ที่พิจารณาจากความผิดพลาดในการนำเสนอหน้าที่การทำงานของข้อมูล (Basili และคณะ, 1996) มีประเภทข้อบกพร่องบางส่วนที่เหมือนประเภทข้อบกพร่องตามกลุ่มที่ 2 ที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง (Travassos และคณะ, 1999) โดยประเภทข้อบกพร่องในส่วนที่ไม่เหมือนกับประเภทข้อบกพร่องในกลุ่มที่ 2 เป็นประเภทข้อบกพร่องที่ไม่สามารถตรวจพบได้ในเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอล

ดังนั้นในส่วนของการวิเคราะห์การกระจายตัวของข้อบกพร่องตามประเภทข้อบกพร่องที่ได้จากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ เพื่อใช้ยืนยันความเหมาะสมของเอกสารการออกแบบซอฟต์แวร์ว่าสามารถเป็นตัวแทนที่ดีของเอกสารการออกแบบซอฟต์แวร์ทั้งหมด จะวิเคราะห์ประเภทข้อบกพร่อง 3 กลุ่ม ดังนี้ (1) ประเภทข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส (Lange และ Chaundron, 2006) (2) ประเภทข้อบกพร่องที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของ

ข้อบกพร่อง (Travassos และคณะ, 1999) และ (3) ประเภทข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง (Strauss และ Ebenau, 1994)

จากประเภทข้อบกพร่องตามกลุ่มที่ 1 ที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส (Lange และ Chaundron, 2006) จะเห็นได้ว่าชื่อของประเภทข้อบกพร่องในกลุ่มนี้ แสดงลักษณะการเกิดของข้อบกพร่องที่ตำแหน่งใดและแผนภาพใดอย่างชัดเจน อย่างไรก็ดีจากการวิจัยของ Lange และ Chaundron (2006) ไม่ครอบคลุมทุกแผนภาพของยูเอ็มแอล แสดงว่าประเภทข้อบกพร่องตามกลุ่มนี้ที่ได้จากงานวิจัยของ Lange และ Chaundron (2006) ไม่ครอบคลุมประเภทข้อบกพร่องที่เกิดในทุกแผนภาพของยูเอ็มแอลเช่นกัน

ถ้าพิจารณาประเภทข้อบกพร่องตามกลุ่มที่ 4 ที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง (Strauss และ Ebenau, 1994) ดูเหมือนจะเป็นการจัดกลุ่มของประเภทข้อบกพร่องที่กว้างที่สุด หากจัดประเภทข้อบกพร่องจากกลุ่มอื่น โดยใช้ประเภทข้อบกพร่องตามกลุ่มที่ 4 เป็นหลัก จะได้ผลดังตารางที่ 2-1

ตารางที่ 2-1 แสดงการจัดกลุ่มของประเภทข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ จากงานวิจัยในอดีต

กลุ่มของประเภทข้อบกพร่อง			
4. พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง (Strauss และ Ebenau, 1994)	**1. พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส** (Lange และ Chaundron, 2006)	**2. พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง** (Travassos และคณะ, 1999)	**3. พิจารณาจากการความผิดพลาดในการนำเสนอหน้าที่การทำงานของข้อมูล** (Basili และคณะ, 1996)
- สิ่งสูญหาย	- ข้อความไม่มีชื่อ - คลาสไม่มีในแผนภาพซีเควนซ์ - อ็อบเจ็กต์ที่ไม่มีคลาสในแผนภาพคลาส - ยูสเคสไม่มีแผนภาพซีเควนซ์ - เมท็อดไม่ถูกเรียกในแผนภาพซีเควนซ์	- การละเลย	- การขาดหน้าที่ - การขาดคุณสมบัติเฉพาะ
- สิ่งผิดพลาด	- ข้อความไม่มีวิธีดำเนินการ - ข้อความส่งผิดทิศทาง - การเรียกใช้คลาสเดียวกันในแผนภาพเดียวกันมากกว่า 1 ครั้ง	- ข้อมูลไม่สอดคล้องกัน - ข้อมูลไม่ชัดเจน	- ข้อมูลไม่ชัดเจน - ข้อมูลไม่สอดคล้องกัน
- สิ่งเพิ่มเติม	-	- แสดงข้อมูลที่ไม่จำเป็น	-

อย่างไรก็ดี ยังมีประเภทข้อบกพร่องบางประเภทที่ไม่สามารถจัดให้อยู่ในประเภทข้อบกพร่องตามกลุ่มที่ 4 ที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง (Strauss และ Ebenau, 1994) ได้ เนื่องจากความหมายของประเภทข้อบกพร่องเหล่านั้นไม่ตรงกับประเภทข้อบกพร่องตามกลุ่มที่

4 โดยตารางที่ 2-2 จะแสดงประเภทข้อบกพร่องจากกลุ่มอื่นที่ไม่สามารถจัดให้เป็นประเภทเดียวกับประเภทข้อบกพร่องตามกลุ่มที่ 4 ได้

ตารางที่ 2-2 แสดงประเภทข้อบกพร่องในกลุ่มที่ 2 และกลุ่มที่ 3 ที่ไม่สามารถจัดรวมกับกลุ่มที่ 4 ได้

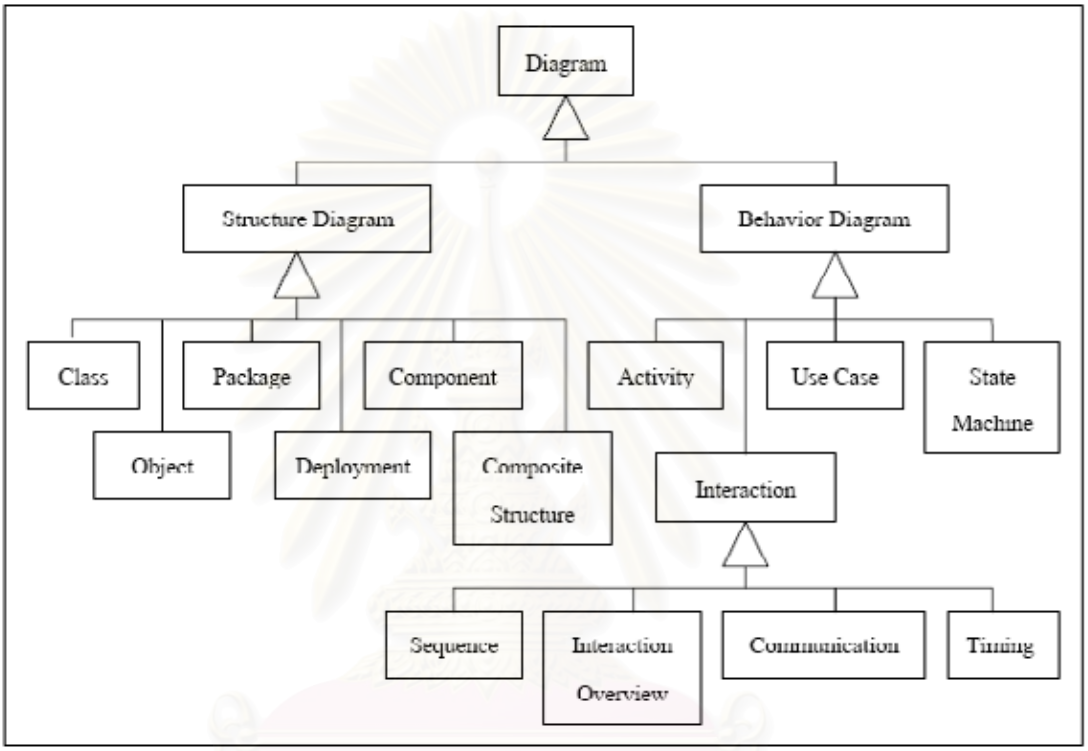
กลุ่มของประเภทข้อบกพร่อง	
2. พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง (Travassos และคณะ, 1999)	**3. พิจารณาจากความผิดพลาดในการนำเสนอหน้าที่การทำงานของข้อมูล** (Basili และคณะ, 1996)
- ข้อมูลไม่เป็นความจริง	- การติดต่อขาดหายไป - สิ่งแวดล้อมขาดหายไป

2.8 ภาษายูเอ็มแอล (UML: Unified Modeling Language)

แผนภาพยูเอ็มแอลเป็นภาษาสัญลักษณ์รูปภาพมาตรฐานที่ใช้สำหรับการออกแบบแบบจำลองเชิงวัตถุ (Object-Oriented Modeling) (Dennis และคณะ, 2005) โดยมีพื้นฐานอยู่บนหลักการของวิชวลโมเดลลิง (Visual Modeling) นั่นคือการสร้างแบบจำลองโดยการใช้สัญลักษณ์รูปภาพเพื่อทำความเข้าใจกับความต้องการของลูกค้า ทำให้ระบบที่ออกแบบมีความชัดเจน และสามารถบำรุงรักษาระบบได้ง่ายขึ้น และจากการที่ยูเอ็มแอลเป็นภาษารูปภาพที่มีมาตรฐานและเป็นภาษาสากลที่ใช้ในการออกแบบซอฟต์แวร์ที่พัฒนาเชิงวัตถุ ดังนั้นเอกสารแสดงการออกแบบซอฟต์แวร์ที่ใช้แผนภาพยูเอ็มแอล จึงสามารถใช้สื่อสารระหว่างผู้ร่วมงานให้เข้าใจระบบได้ตรงกันภายในระยะเวลาอันรวดเร็ว

นอกจากนี้การนำเสนอด้วยแผนภาพยูเอ็มแอล ยังสนับสนุนหลักการเชิงวัตถุได้อย่างครบถ้วน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544) แนวคิดเชิงวัตถุได้รับความนิยมจากผู้ออกแบบและผู้พัฒนาซอฟต์แวร์ในปัจจุบัน ทำให้มีผู้คิดค้นวิธีการพัฒนาซอฟต์แวร์เชิงวัตถุขึ้นหลายวิธี ถ้าเลือกวิธีในการวิเคราะห์ออกแบบของใครก็จำเป็นต้องใช้วิธีการนำเสนอผลลัพธ์ของการวิเคราะห์ออกแบบตามที่ผู้สร้างกำหนดด้วย แต่ละวิธีต่างกำหนดสัญลักษณ์รูปภาพ (Notation) ที่ต่างกันไป ดังนั้นจึงได้รวมวิธีการต่างๆเข้าไว้ด้วยกัน โดยในปี 1994 ได้นำวิธี Booch ของ Booch (1994) และวิธีโอเอ็มที (OMT) ของ Rumbaugh (1991) มารวมกัน เป็นการพัฒนาซอฟต์แวร์เชิงวัตถุที่เป็นหนึ่งเดียวกัน (Unified Method) ต่อมาในปี 1995 ได้นำโอโอเอสอี (OOSE) เป็นกระบวนการของ Jacobson (1992) เข้ามาร่วมกับโครงการนี้ เป็นการสร้างภาษาแบบจำลองขึ้นใหม่เรียกว่าภาษายูเอ็มแอล (Unified Modeling Language: UML) เนื่องจากวิธีการพัฒนาซอฟต์แวร์

เชิงวัตถุของทั้ง 3 คนมีชื่อเสียงอยู่แล้วในช่วงเวลานั้น ดังนั้นภาษายูเอ็มแอลที่ถูกพัฒนาขึ้นนี้จึงกลายเป็นที่นิยมใช้กันอย่างแพร่หลาย (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544) โดยในปัจจุบันแผนภาพยูเอ็มแอลรุ่นที่ถูกเผยแพร่คือรุ่นที่ 2.0 ถูกพัฒนาโดยองค์กรโอเอ็มจี (OMG) ในปี 2003 ในรุ่นนี้ประกอบด้วยแผนภาพทั้งหมด 13 แผนภาพ (Dennis และคณะ, 2005)



รูปที่ 2-9 แสดงแผนภาพทั้งหมดของยูเอ็มแอลรุ่น 2.0 (Dennis และคณะ, 2005)

2.8.1 แผนภาพยูเอ็มแอลรุ่น 2.0 สาเหตุที่ยูเอ็มแอลต้องมีแผนภาพที่หลากหลาย เนื่องจากระบบที่มีความซับซ้อนมาก ยังไม่มีวิธีการออกแบบระบบที่สามารถใช้เพียงแผนภาพเดียว แล้วสามารถให้มุมมองได้ทั้งหมด จึงต้องเลือกแผนภาพหลายๆแผนภาพเพื่อให้สามารถเสนอมุมมองได้ครบถ้วน ดังนั้นนอกจากผู้ออกแบบและผู้พัฒนาจะต้องทำความเข้าใจกับองค์ประกอบของยูเอ็มแอลแล้ว ยังต้องทำความเข้าใจกับวัตถุประสงค์ของแต่ละแผนภาพ เพื่อให้สามารถเลือกใช้งานได้อย่างถูกต้อง และเหมาะสมกับมุมมองที่ต้องการนำเสนอ แผนภาพของยูเอ็มแอลรุ่น 2.0 มีทั้งหมด 13 แผนภาพ (Dennis และคณะ, 2005) ดังต่อไปนี้

1. แผนภาพคลาส (Class Diagram) วัตถุประสงค์สำคัญของแผนภาพคลาสคือ เพื่อสร้างคำศัพท์ที่ถูกใช้โดยนักวิเคราะห์และผู้ใช้ และเพื่อใช้ค้นหาคลาสที่จำเป็นจากคำบรรยายยูสเคส

(ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544) นอกจากนี้ยังนำเสนอสิ่งของและแนวความคิดที่อยู่ในโปรแกรมประยุกต์ (Application)

2. แผนภาพอ็อบเจ็กต์ (Object Diagram) มีความคล้ายคลึงกับแผนภาพคลาส แต่มีความแตกต่างที่สำคัญตรงที่แผนภาพอ็อบเจ็กต์แสดงอ็อบเจ็กต์ (Object) และความสัมพันธ์ระหว่างอ็อบเจ็กต์ วัตถุประสงค์สำคัญของแผนภาพอ็อบเจ็กต์คือให้นักวิเคราะห์เปิดเผยรายละเอียดของคลาสได้เพิ่มขึ้น

3. แผนภาพแพ็กเกจ (Package Diagram) ใช้เพื่อรวมกลุ่มส่วนย่อย (Element) ของแผนภาพอื่นๆเข้าไว้ด้วยกันเป็น 1 แพ็กเกจ โดยแผนภาพแพ็กเกจมีพื้นฐานเดียวกับแผนภาพคลาส นั่นคือเพื่อแสดงแพ็กเกจและความสัมพันธ์เท่านั้น

4. แผนภาพดีพลอยเมนต์ (Deployment Diagram) ใช้ในการนำเสนอความสัมพันธ์ระหว่างองค์ประกอบของฮาร์ดแวร์ (Hardware) ที่ใช้ในโครงสร้างพื้นฐานทางกายภาพ (Physical Infrastructure) ของระบบข้อมูล นอกจากนี้ยังถูกใช้เพื่อนำเสนอองค์ประกอบของซอฟต์แวร์อีกด้วย

5. แผนภาพคอมโพเนนต์ (Component Diagram) ให้ผู้ออกแบบจำลองแบบของความสัมพันธ์ทางกายภาพระหว่างมอดูลทางกายภาพของโปรแกรม โดยเมื่อแผนภาพนี้นำไปรวมกับแผนภาพดีพลอยเมนต์แล้ว สามารถใช้ในการแสดงการกระจายทางกายภาพของมอดูลซอฟต์แวร์ (Software Module) นอกจากนี้สามารถใช้ในการออกแบบและพัฒนาระบบเชิงส่วนประกอบ (Component-Based System)

6. แผนภาพคอมโพซีทสตรักเจอร์ (Composite Structure Diagram) เป็นแผนภาพใหม่ที่เพิ่มเข้ามาในแผนภาพยูเอ็มแอลรุ่น 2.0 ใช้เมื่อโครงสร้างภายในของคลาสมีความซับซ้อนโดยใช้เพื่อจำลองความสัมพันธ์ระหว่างส่วนของคลาส

7. แผนภาพกิจกรรม (Activity Diagram) เพื่อจำลองกระแสงานของแต่ละยูสเคสทั้งหมดในแผนภาพยูสเคส แสดงการทำงานของอ็อบเจ็กต์และกิจกรรม (Activity) ที่เกิดขึ้นในกลุ่มของอ็อบเจ็กต์

8. แผนภาพซีเควนซ์ (Sequence Diagram) แสดงการโต้ตอบแบบพลวัต (Dynamic) ระหว่างอ็อบเจ็กต์ของแต่ละยูสเคสทั้งหมดในแผนภาพยูสเคส โดยให้ความสำคัญกับลำดับเวลาของกิจกรรม

9. แผนภาพคอมมิวนิเคชัน (Communication Diagram) มุ่งประเด็นไปที่กลุ่มของข่าวสาร (Message) ถูกส่งผ่านระหว่างอ็อบเจกต์ที่เกี่ยวข้องกัน โดยแผนภาพซีเควนซ์และแผนภาพคอมมิวนิเคชันเตรียมข้อมูลเหมือนกัน

10. แผนภาพแสดงการโต้ตอบ (Interaction Overview Diagram) ช่วยให้นักวิเคราะห์ทำความเข้าใจยูสเคสที่มีความซับซ้อน โดยการให้คำอธิบายสายงานของกระบวนการ ประโยชน์ที่สำคัญของแผนภาพแสดงการโต้ตอบ คือสามารถจำลองลำดับของสายงานได้ง่าย อย่างไรก็ตาม สามารถใช้แผนภาพกิจกรรม และยูสเคสแทนได้

11. แผนภาพไทม์มิง (Timing Diagram) แสดงการโต้ตอบระหว่างอ็อบเจ็กต์คู่กันกับแกนของเวลา วัตถุประสงค์หลักคือแสดงการเปลี่ยนแปลงสถานะของอ็อบเจ็กต์ เหมาะในการใช้เพื่อออกแบบระบบฝังตัว (Embedded System) และระบบทำงานแบบทันที (Real-Time System)

12. แผนภาพสถานะ (State Machine Diagram) อธิบายการเปลี่ยนแปลงสถานะของอ็อบเจ็กต์ระหว่างระยะเวลาที่มีอยู่ (Lifetime) และแสดงลำดับของเหตุการณ์ที่อ็อบเจ็กต์ตอบสนอง

13. แผนภาพยูสเคส (Use Case Diagram) ใช้จำลองการโต้ตอบของข้อมูลและสิ่งแวดล้อมของข้อมูล โดยสิ่งแวดล้อมของข้อมูลรวมถึงผู้ใช้ขั้นปลาย (End User) และระบบภายนอกที่โต้ตอบกับข้อมูล วัตถุประสงค์หลักของแผนภาพยูสเคส คือเพื่อทำความเข้าใจความต้องการของลูกค้าและแสดงหน้าที่ที่ระบบต้องทำได้ ถือว่าแผนภาพนี้เป็นสิ่งสำคัญสำหรับการวิเคราะห์และออกแบบระบบเชิงวัตถุ

ในการใช้งานแต่ละแผนภาพนอกจากต้องทำความเข้าใจกับวัตถุประสงค์ของการใช้งานแผนภาพแล้ว ยังต้องเข้าใจส่วนประกอบของแต่ละแผนภาพด้วย งานวิจัยนี้เป็นการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล ในการทดลองจะตรวจสอบ 4 แผนภาพเท่านั้น เนื่องจากเทคนิคโอโออาร์ทีถูกจัดทำขึ้นตั้งแต่ปี 1998 จึงถูกออกแบบเพื่อนำมาใช้ในการตรวจสอบแผนภาพยูเอ็มแอลรุ่น 1.1 ดังนั้นงานวิจัยนี้จึงมุ่งประเด็นไปที่แผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะตามที่เทคนิคโอโออาร์ทีกำหนดให้ตรวจสอบเท่านั้น

2.8.2 แผนภาพยูสเคส ใช้เพื่อช่วยในการจำลองความต้องการของผู้ใช้ สิ่งสำคัญในการสร้างแผนภาพยูสเคส คือการค้นหาว่าระบบสามารถทำอะไรได้บ้าง โดยไม่สนใจว่ามีกลไกการทำงานอย่างไร แผนภาพยูสเคสมีประโยชน์คือ เพื่อให้ผู้พัฒนาทราบถึงความสามารถของระบบว่าต้องทำอะไรได้บ้าง ทราบถึงผู้ใช้งานในแต่ละส่วนของระบบ (แต่ละยูสเคส) (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544) แผนภาพยูสเคสมีส่วนประกอบสำคัญดังต่อไปนี้

1. แอกเตอร์ (Actor) คือบุคคลหรือระบบภายนอกที่มีส่วนเกี่ยวข้องกับระบบที่พัฒนา (Dennis และคณะ, 2005) โดยในการค้นหาแอกเตอร์ของระบบคือ ดูว่ามีใครเป็นผู้ใช้หรือดูแลระบบ อุปกรณ์ฮาร์ดแวร์ที่เชื่อมต่อระบบ และระบบภายนอกที่เชื่อมต่อกับระบบที่เราพัฒนา

ชื่อของแอกเตอร์จะแสดงบทบาทและหน้าที่ที่มีต่อระบบ ในการเขียนแอกเตอร์จะแทนด้วยสัญลักษณ์รูปคน (Stick Man) ดังรูปที่ 2-10 โดยวางไว้นอกขอบเขตของระบบ

รูปที่ 2-10 แสดงแอกเตอร์ (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

2. ยูสเคส เป็นฟังก์ชันการทำงานต่างๆที่ซอฟต์แวร์ต้องทำได้ทั้งหมดจึงจะถือว่าซอฟต์แวร์ไม่มีข้อผิดพลาด โดยยูสเคสต้องถูกกระทำโดยแอกเตอร์ คือ การส่งและรับข้อมูลระหว่างแอกเตอร์ (Dennis และคณะ, 2005) และในบางกรณีอาจมียูสเคสที่ไม่ได้ถูกกระทำโดยแอกเตอร์ แต่อาจมีความสัมพันธ์ร่วมกับยูสเคสอื่นๆที่มีความสัมพันธ์โดยตรงกับแอกเตอร์แทน ในการเขียนยูสเคสจะแสดงด้วยสัญลักษณ์รูปวงรี ดังรูปที่ 2-11 โดยวางไว้ในขอบเขตของระบบ

รูปที่ 2-11 แสดงยูสเคส (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

3. ขอบเขตระบบ เป็นการแสดงขอบเขตการทำงานของระบบ (Dennis และคณะ, 2005) โดยสามารถแสดงได้ด้วยสัญลักษณ์รูปสี่เหลี่ยม ข้างในขอบเขตจะประกอบด้วยยูสเคสต่างๆของระบบ และมีชื่อระบบเขียนอยู่ด้านบนหรือข้างในรูปสี่เหลี่ยม ดังรูปที่ 2-12

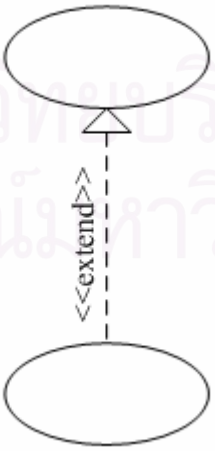
รูปที่ 2-12 แสดงขอบเขตของระบบ (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

4. ความสัมพันธ์ในแผนภาพยูสเคส สามารถแบ่งออกได้ 4 ประเภทคือ

ก. ความสัมพันธ์แบบเจเนอรัลไลเซชัน (Generalization Relationship) เป็นความสัมพันธ์ระหว่างแอกเตอร์ด้วยกันหรือยูสเคสด้วยกัน กล่าวคือ ถ้ามีหลายแอกเตอร์ในระบบมีคุณสมบัติคล้ายกัน สามารถแยกออกมาเป็นอีกแอกเตอร์หนึ่ง ที่รวมบทบาทที่เหมือนกันของแต่ละแอกเตอร์ถือว่าเป็นคุณสมบัติพื้นฐาน โดยแอกเตอร์พื้นฐานจะถูกสืบทอด (Inherit) จากแอกเตอร์อื่นๆ โดยสัญลักษณ์ที่ใช้คือลูกศรหัวโปร่งชี้จากแอกเตอร์หนึ่งไปยังแอกเตอร์ที่ถูกสืบทอด ดังรูปที่ 2-13

รูปที่ 2-13 แสดงความสัมพันธ์แบบเจเนอรัลไลเซชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

ข. ความสัมพันธ์แบบขยาย (Extend Relationship) เป็นความสัมพันธ์ระหว่างยูสเคส กล่าวคือทำยูสเคส A แล้วอาจทำยูสเคส B ต่อหรือไม่ทำก็ได้ เช่นทำยูสเคสอ่านอีเมลแล้ว อาจทำยูสเคสตอบอีเมลหรือยูสเคสส่งอีเมล (E-Mail) ด้วยก็ได้เป็นต้น โดยสัญลักษณ์ที่ใช้คือ ลูกศรเส้นประที่ชี้จากยูสเคสแรก ไปยังยูสเคสที่ถูกช่วยเหลือหรือถูกขยาย โดยมีคำว่าเอ็กซ์เทนด์ (Extend) อยู่ที่กึ่งกลางลูกศร ดังรูปที่ 2-14



รูปที่ 2-14 แสดงความสัมพันธ์แบบขยาย (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

ค. ความสัมพันธ์แบบรวม (Include Relationship) เป็นความสัมพันธ์ระหว่างยูสเคส กล่าวคือยูสเคสหนึ่งจำเป็นต้องอาศัยการทำงานจากอีกยูสเคสหนึ่งเสมอ (Dennis และคณะ, 2005) เช่น ยูสเคสลงทะเบียนรายวิชาจำเป็นต้องใช้ยูสเคสตรวจสอบรายวิชาที่ต้องเรียนก่อนหน้า เนื่องจากการที่จะลงทะเบียนแต่รายวิชาจำเป็นต้องตรวจสอบก่อนว่ารายวิชาก่อนหน้าที่จำเป็นต้องเรียนนั้นได้เรียนผ่านมาแล้ว นั่นคือถ้าจะทำยูสเคส A ต้องทำยูสเคส B ด้วยเสมอ โดยการใช้สัญลักษณ์ลูกศรเส้นประที่มีคำว่าอินคลูด (Include) อยู่ที่กลางลูกศรชี้ไปยังยูสเคสที่ถูกเรียกใช้หรือถูกรวมไว้ด้วยกัน ดังรูปที่ 2-15

รูปที่ 2-15 แสดงความสัมพันธ์แบบรวม (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

ง. ความสัมพันธ์แบบแอสโซซิเอชัน (Association Relationship) เป็นความสัมพันธ์ระหว่างยูสเคสกับแอกเตอร์ กล่าวคือแอกเตอร์โต้ตอบกับยูสเคส (Dennis และคณะ, 2005) เช่น แอกเตอร์แลกเปลี่ยนข้อมูลข่าวสารกับระบบที่พัฒนาเป็นต้น ใช้สัญลักษณ์เส้นตรงดังรูปที่ 2-16

รูปที่ 2-16 แสดงความสัมพันธ์แบบแอสโซซิเอชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

2.8.3 แผนภาพคลาส วัตถุประสงค์ของแผนภาพคลาสคือ แสดงโครงสร้างของระบบประกอบไปด้วยคลาส และความสัมพันธ์ระหว่างคลาสเหล่านั้น (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544; Dennis และคณะ, 2005) โดยแผนภาพคลาสมีส่วนประกอบสำคัญดังนี้

1. คลาส คือกลุ่มของอ็อบเจ็กต์ ดังนั้นการที่จะหาคลาสของอ็อบเจ็กต์ได้ จะต้องจัดหมวดหมู่ของอ็อบเจ็กต์หลายๆอ็อบเจ็กต์ได้ โดยคลาสในระบบอาจนำมาจากคลาสไลบรารีที่ผู้พัฒนาอื่นสร้างไว้ อุปกรณ์ต่างๆที่ระบบต้องควบคุม เช่น คลาสเครื่องพิมพ์ เป็นต้น หรือแอกเตอร์ในแผนภาพยูสเคส โดยรายละเอียดของคลาสมีดังต่อไปนี้ (Dennis และคณะ, 2005)

ก. ชื่อคลาส (Class Name) ในการตั้งชื่อคลาสควรตั้งชื่อให้สื่อความหมาย

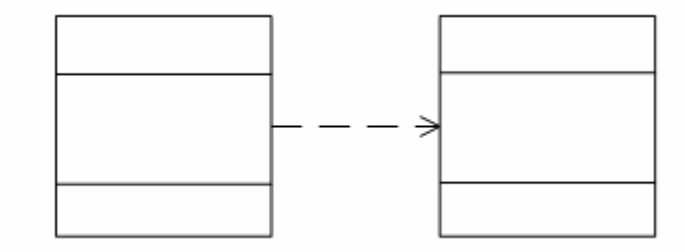
ข. แอตทริบิวต์ (Attribute) เป็นการนำเสนอคุณสมบัติที่อธิบายสถานะของอ็อบเจ็กต์ โดยประกอบด้วยชนิดการเข้าถึง (Visibility) ของแอตทริบิวต์ ชื่อของแอตทริบิวต์ประเภทของแอตทริบิวต์ อยู่ต่อจากเครื่องหมายมหัพภาคคู่ (:) และค่าเริ่มต้นของแอตทริบิวต์ อาจมีหรือไม่มีก็ได้แต่ถ้ามีจะอยู่ต่อจากเครื่องหมายเท่ากับ

ค. โอเปอร์เรชัน (Operation) เป็นการแสดงการกระทำของคลาส โดยประกอบด้วยชนิดการเข้าถึงโอเปอร์เรชัน ชื่อของโอเปอร์เรชัน พารามิเตอร์ (Parameter) ที่จำเป็นต่อการทำงานของโอเปอร์เรชัน พารามิเตอร์จะประกอบด้วยชื่อพารามิเตอร์ ตามด้วยเครื่องหมายมหัพภาคคู่ และประเภทของพารามิเตอร์ตามลำดับ ประเภทของค่าที่ส่งคืน (Return Type) จะเขียนตามเครื่องหมายมหัพภาคคู่

รูปที่ 2-17 แสดงการกำหนดแอตทริบิวต์ และโอเปอเรชันภายในคลาส (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

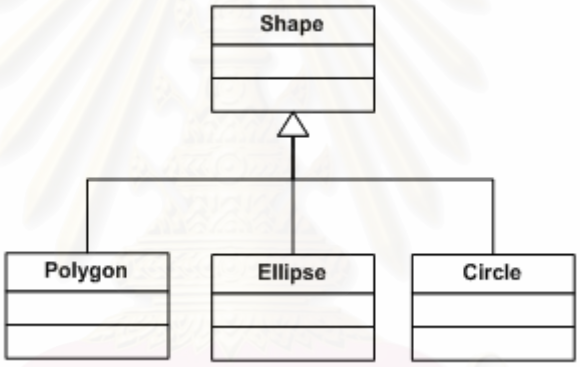
2. ความสัมพันธ์ คือความสัมพันธ์ระหว่างคลาส สามารถแบ่งออกได้เป็น 3 ประเภทดังต่อไปนี้

ก. ความสัมพันธ์แบบพึ่งพิง (Dependency Relationship) ความสัมพันธ์แบบนี้เกิดขึ้นเมื่อการเปลี่ยนแปลงที่เกิดขึ้นกับคลาสที่ถูกพึ่งพิง (Independent Class) จะส่งผลต่อคลาสที่พึ่งพิง (Dependent Class) คลาสดังกล่าว (Dennis และคณะ, 2005) สัญลักษณ์ที่ใช้คือลูกศรหัวโปร่งแบบเส้นประชี้จากคลาสที่พึ่งพิงไปยังคลาสที่ถูกพึ่งพิง ดังรูปที่ 2-18



รูปที่ 2-18 แสดงความสัมพันธ์แบบพึ่งพิง (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

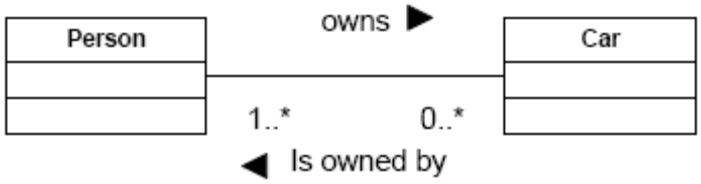
ข. ความสัมพันธ์แบบเจเนอรัลไลเซชัน (Generalization Relationship) คือ ความสัมพันธ์ระหว่างซูเปอร์คลาสและซับคลาส เป็นการสืบทอดคุณสมบัติจากซูเปอร์คลาสนั่นเอง (Dennis และคณะ, 2005) สัญลักษณ์ที่ใช้คือเส้นตรงทึบที่มีหัวลูกศรเป็นสามเหลี่ยมโปร่ง โดยชี้จากซับคลาสไปยังซูเปอร์คลาส ดังรูปที่ 2-19



รูปที่ 2-19 แสดงความสัมพันธ์แบบเจเนอรัลไลเซชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

ค. ความสัมพันธ์แบบแอสโซซิเอชัน เป็นความสัมพันธ์ระหว่างคลาสอีกชนิดหนึ่ง โดยสามารถแบ่งออกได้เป็น

- นอร์มอลแอสโซซิเอชัน (Normal Association) โดยปกติความสัมพันธ์แบบนี้จะเป็นความสัมพันธ์แบบ 2 ทิศทาง ใช้สัญลักษณ์เส้นตรงทึบเชื่อมระหว่าง 2 คลาส และมีชื่อของความสัมพันธ์กำกับอยู่ โดยชื่อที่ใช้มักเป็นคำกริยาโดยส่วนใหญ่ นอกจากนี้ยังสามารถกำหนดทิศทางของชื่อความสัมพันธ์ได้ โดยการวาดสามเหลี่ยมทึบไว้ด้านซ้ายหรือด้านขวาของชื่อ ช่วยให้อ่านความสัมพันธ์ได้อย่างถูกต้อง และที่แต่ละเส้นความสัมพันธ์อาจมี 2 ชื่อความสัมพันธ์มีทิศทางตรงข้ามกัน และสามารถกำหนดปริมาณของคลาสหรืออ็อบเจ็กต์ที่สัมพันธ์กันอยู่เรียกว่ามัลติไพลซิตี (Multiplicity) สามารถกำหนดได้หลายรูปแบบโดยการใส่ตัวเลขไว้ที่ปลายด้านใดด้านหนึ่งของความสัมพันธ์ดังรูปที่ 2-20



รูปที่ 2-20 แสดงความสัมพันธ์แบบแอสโซซิเอชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

นอกจากนี้ยังกำหนดบทบาท (Role) ให้กับแต่ละคลาสที่เชื่อมต่ออยู่กับเส้นความสัมพันธ์ได้เช่นกันดังรูปที่ 2-21

รูปที่ 2-21 แสดงความสัมพันธ์แบบแอสโซซิเอชัน เมื่อกำหนดบทบาทของแต่ละคลาส (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

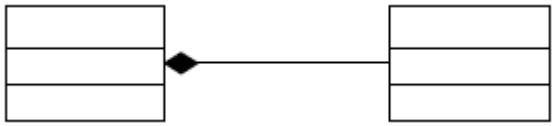
3. แอกกริเกชัน (Aggregation) เป็นความสัมพันธ์ระหว่างคลาสหรืออ็อบเจ็กต์ในแง่ของการรวมกันหรือการประกอบกัน สามารถแบ่งได้ 2 แบบดังนี้

ก. นอร์มอลแอกกริเกชัน (Normal Aggregation) เป็นการกำหนดส่วนประกอบแบบไม่จำเป็น (Dennis และคณะ, 2005) ใช้สัญลักษณ์เส้นตรงทึบโยงระหว่างคลาส โดยมีข้าวหลามตัดติดอยู่ระหว่างปลายเส้นความสัมพันธ์กับคลาสที่ใหญ่กว่า เช่น ประตูเป็นส่วนหนึ่งของรถ ในขณะเดียวกันสามารถกำหนดชื่อความสัมพันธ์ ทิศทางของชื่อความสัมพันธ์ ทิศทางของความสัมพันธ์ และปริมาณที่สัมพันธ์กันได้ดังรูปที่ 2-22

รูปที่ 2-22 แสดงนอร์มอลแอกกริเกชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

ข. คอมโพซิชัน (Composition) มีความคล้ายคลึงกับนอร์มอลแอกกริเกชัน แต่เป็นการกำหนดส่วนประกอบแบบจำเป็น เช่น คลาสรถ (Car) ประกอบด้วยคลาสเครื่องยนต์ (Engine)

เมื่อคลาสที่เป็นเจ้าของ (คลาสรถ) ถูกทำลาย คลาสที่เป็นองค์ประกอบ (คลาสเครื่องยนต์) จะถูกทำลายไปด้วยพร้อมๆกัน ดังรูปที่ 2-23

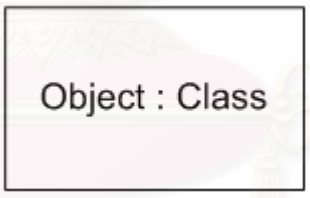


รูปที่ 2-23 แสดงคอมโพซิชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

2.8.4 แผนภาพซีเควนซ์ แผนภาพนี้จะบอกว่าในยูสเคส อ็อบเจ็กต์แต่ละตัวติดต่อสื่อสารกันอย่างไร มีขั้นตอนการทำงานอย่างไร โดยจะเน้นไปที่แกนเวลาเป็นสำคัญ (Dennis และคณะ, 2005) ถ้าเวลาเปลี่ยนขั้นตอนการทำงานจะเปลี่ยนโดยมีแอกเตอร์เป็นผู้เริ่มต้นขั้นตอนโดยแผนภาพซีเควนซ์มีส่วนประกอบดังต่อไปนี้

1. แอกเตอร์ คือบุคคลหรือระบบภายนอกที่โต้ตอบกับระบบ โดยการรับส่งเมสเซจ (Message) (Dennis และคณะ, 2005) สัญลักษณ์ที่ใช้คือรูปคน ดังรูปที่ 2-10

2. อ็อบเจ็กต์แทนด้วยรูปสี่เหลี่ยมเรียงกันตามแนวนอน ภายในบรรจุชื่ออ็อบเจ็กต์ตามด้วยเครื่องหมายมหัพภาคคู่และชื่อคลาส ดังรูปที่ 2-24



รูปที่ 2-24 แสดงอ็อบเจ็กต์ (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

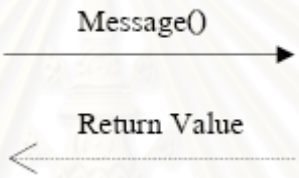
3. ไลฟ์ไลน์ (Lifeline) แสดงถึงชีวิตของอ็อบเจ็กต์ (Dennis และคณะ, 2005) แทนด้วยเส้นประแนวตั้งดังรูปที่ 2-25

รูปที่ 2-25 แสดงไลฟ์ไลน์ (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

4. แอกติเวชัน (Activation) ใช้แสดงช่วงเวลาที่อ็อบเจ็กต์กำลังส่งและรับเมสเซจ (Dennis และคณะ, 2005) แทนด้วยสี่เหลี่ยมแนวตั้ง ดังรูปที่ 2-26

รูปที่ 2-26 แสดงแอกติเวชัน(ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

5. เมสเซจ เป็นการส่งและคืนข้อมูล (Dennis และคณะ, 2005) สัญลักษณ์ที่ใช้ในการส่งข้อมูลคือลูกศรเส้นทึบ และสัญลักษณ์ที่ใช้ในการคืนข้อมูลคือลูกศรเส้นประดังรูปที่ 2-27



รูปที่ 2-27 แสดงเมสเซจ (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

6. อ็อบเจกต์ดีสตรักชัน (Object Destruction) เป็นการใช้กากบาทวางไว้ที่ปลายไลฟ์ไลน์ของอ็อบเจ็กต์ เพื่อแสดงว่าการทำงานของอ็อบเจ็กต์สิ้นสุดแล้ว (Dennis และคณะ, 2005) ดังรูปที่ 2-28

X

รูปที่ 2-28 แสดงอ็อบเจกต์ดีสตรักชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

2.8.5 แผนภาพสถานะ เป็นการแสดงพฤติกรรมของคลาสต่างๆในระบบว่ามีสถานะอะไรบ้าง จะเปลี่ยนสถานะเมื่อเกิดเหตุการณ์อะไร แผนภาพสถานะของแต่ละคลาสประกอบไปด้วยสถานะต่างๆที่สามารถเกิดขึ้นได้ (Dennis และคณะ, 2005) เช่น รถอยู่ในสถานะกำลังวิ่ง ลิฟต์อยู่ในสถานะขึ้นเป็นต้น และเมื่อเวลาผ่านไปหรือมีเหตุการณ์บางอย่างเกิดขึ้น ย่อมทำให้เกิดการเปลี่ยนสถานะหรือเปลี่ยนพฤติกรรมได้ เช่น หน้าจอคอมพิวเตอร์อยู่ในสถานะเปิด แต่เมื่อถูกกดสวิตช์ปิดจะเปลี่ยนสถานะเป็นปิด หรือถ้าปล่อยหน้าจอทิ้งไว้ 5 นาทีจะเปลี่ยนสถานะเป็นหน้าจอดับเป็นต้น พฤติกรรมหรือสถานะต่างๆเหล่านี้ย่อมเปลี่ยนไปได้เสมอ แผนภาพสถานะประกอบด้วยส่วนต่างๆดังนี้

1. สถานะ (State) คือสถานะของอ็อบเจ็กต์ (Dennis และคณะ, 2005) โดยแสดงด้วยสัญลักษณ์รูปสี่เหลี่ยมหัวมน ดังรูปที่ 2-29

รูปที่ 2-29 แสดงสถานะ (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

2. สถานะเริ่มต้น (Initial State) เป็นจุดเริ่มต้นสถานะของอ็อบเจ็กต์ (Dennis และคณะ, 2005) แสดงด้วยสัญลักษณ์รูปวงกลมทึบ ดังรูปที่ 2-30

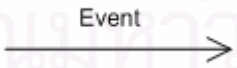
รูปที่ 2-30 แสดงสถานะเริ่มต้น (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

3. สถานะสิ้นสุด (Final State) เป็นจุดสิ้นสุดการกระทำของอ็อบเจ็กต์ (Dennis และคณะ, 2005) แสดงด้วยสัญลักษณ์รูปวงกลมโปร่งล้อมรอบวงกลมทึบข้างในเรียกว่า ตาวัว (Bull's Eye) ดังรูปที่ 2-31

รูปที่ 2-31 แสดงสถานะสิ้นสุด (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

4. อีเวนต์ (Event) คือเหตุการณ์ที่ทำให้สถานะเปลี่ยนแปลงไป (Dennis และคณะ, 2005) ถูกทำเป็นเครื่องหมายบนทรานซิชัน (Transition)

5. ทรานซิชัน คือเส้นลูกศรเส้นทึบที่ชี้จากสถานะหนึ่งไปยังอีกสถานะหนึ่ง โดยมีอีเวนต์แสดงบนเส้นเพื่อบอกว่าเมื่อเกิดเหตุการณ์หนึ่งจะทำให้สถานะหนึ่งเปลี่ยนไปอีกสถานะหนึ่ง (Dennis และคณะ, 2005) ดังรูปที่ 2-32



รูปที่ 2-32 แสดงทรานซิชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

นอกจากแผนภาพทั้ง 4 แผนภาพนี้แล้ว เทคนิคโอโออาร์ทียังนำเอกสารอธิบายความต้องการซอฟต์แวร์ และคำอธิบายคลาสมาตรวจสอบด้วย โดยทั้ง 2 เอกสารมีลักษณะดังต่อไปนี้

2.8.6 เอกสารอธิบายความต้องการซอฟต์แวร์ เป็นเอกสารที่ใช้อธิบายความต้องการทางซอฟต์แวร์ ในเอกสารจะแบ่งออกเป็น 2 ส่วนคือ (1) ความต้องการทางฟังก์ชัน (Functional

Requirement) เป็นความต้องการทางด้านฟังก์ชันการทำงานของซอฟต์แวร์ที่ต้องทำได้ (2) ความต้องการที่ไม่เป็นฟังก์ชัน (Nonfunctional Requirement) เป็นความต้องการที่กำหนดคุณภาพในการทำงานของซอฟต์แวร์ เช่น ประสิทธิภาพ ความน่าเชื่อถือได้เป็นต้น (Dennis และคณะ, 2005)

2.8.7 เอกสารอธิบายคลาส เป็นเอกสารที่อธิบายรายละเอียดของแต่ละคลาส สามารถแบ่งออกเป็น 3 ส่วนคือ (1) รายละเอียดของคลาส ประกอบด้วยชื่อคลาส คำอธิบายคลาส ซูเปอร์คลาส ซับคลาส ชื่อความสัมพันธ์ รูปแบบความสัมพันธ์ ประเภทความสัมพันธ์ มัลติไพลซิตี (2) รายละเอียดของแอตทริบิวต์ ประกอบด้วยชื่อแอตทริบิวต์ คำอธิบายแอตทริบิวต์ การเข้าถึงแอตทริบิวต์ ประเภทของแอตทริบิวต์ (อาจมีค่าเริ่มต้น) (3) เมท็อด ประกอบด้วย ชื่อเมท็อด คำอธิบายเมท็อด การเข้าถึงเมท็อด ชื่อพารามิเตอร์ ประเภทพารามิเตอร์ ประเภทของค่าที่ส่งคืน (Stone, 2001)

บทที่ 3
ระเบียบวิธีวิจัย

3.1 ความนำ

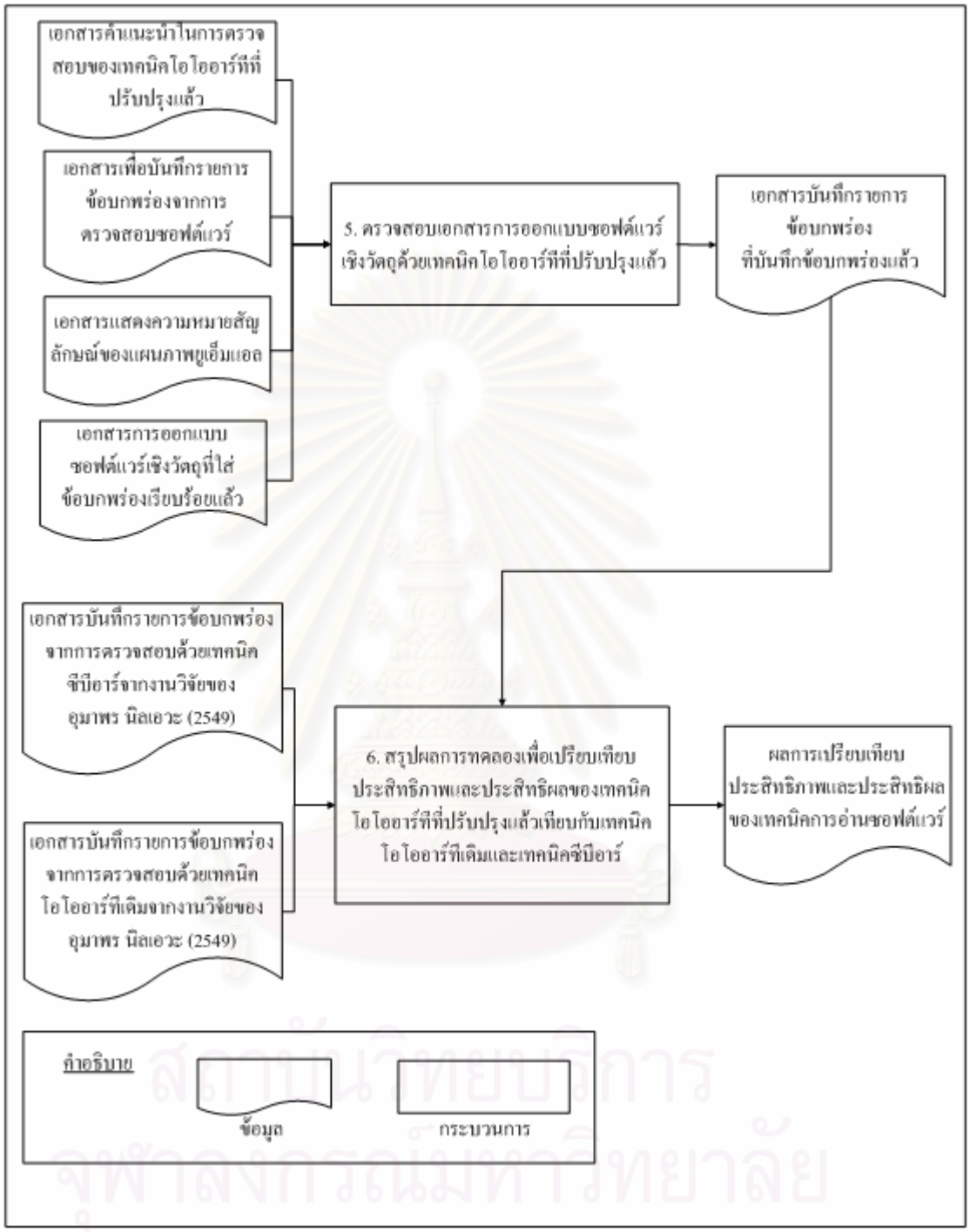
บทนี้นำเสนอแนวทางในการทำวิจัย ประกอบด้วย แผนแบบการทดลอง (Experimental Design) เพื่ออธิบายถึงขั้นตอนการดำเนินการของงานวิจัย ตัวแปรที่เกี่ยวข้องกับการทดลอง การทดสอบสมมติฐาน ประชากรและหน่วยทดลอง แนวทางการทดลอง เครื่องมือที่ใช้ในการทดลอง การเก็บรวบรวมข้อมูล ความถูกต้องและความน่าเชื่อถือของข้อมูลที่เก็บ และกรอบการวิเคราะห์ข้อมูล

3.2 แผนแบบการทดลอง

งานวิจัยนี้จัดทำขึ้นโดยมีวัตถุประสงค์ 3 ประการ คือ (1) เพื่อวิเคราะห์การกระจายตัวของข้อบกพร่องที่พบจากการใช้เทคนิคโอโออาร์ทีในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยวิธีเชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล (2) เพื่อเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ และ (3) เพื่อเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ขั้นตอนการดำเนินการของงานวิจัยนี้ แบ่งออกเป็น 2 ส่วน เพื่อรองรับวัตถุประสงค์ของงานวิจัย โดยแบ่งเป็น (1) การศึกษาประเภทข้อบกพร่องและปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง และ (2) การเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และ เทคนิคซีบีอาร์ ดังรูปที่ 3-1 และ รูปที่ 3-2 โดยรูปที่ 3-1 แสดงขั้นตอนการดำเนินการในส่วนที่ 1 ที่เป็นการศึกษาประเภทข้อบกพร่องและปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง และรูปที่ 3-2 แสดงขั้นตอนการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟตแวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และ เทคนิคซีบีอาร์

รูปที่ 3-1 แสดงขั้นตอนการดำเนินการของงานวิจัยในส่วนของการศึกษาประเภทข้อบกพร่องและปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง



รูปที่ 3-2 แสดงขั้นตอนการดำเนินการของงานวิจัย ในส่วนของการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และ เทคนิคซีบีอาร์

3.3 ขั้นตอนการศึกษาประเภทข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์และการปรับปรุงเทคนิคโอโออาร์ที

จากวัตถุประสงค์ข้อที่ 1 ที่ต้องการวิเคราะห์การกระจายตัวของข้อบกพร่องที่พบจากการใช้เทคนิคโอโออาร์ทีในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยวิธีเชิงวัตถุที่ออกแบบโดยใช้แผนภาพยูเอ็มแอล เริ่มต้นจากการศึกษาประเภทข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์โดยกระทำเพื่อนำประเภทข้อบกพร่องที่ศึกษาได้ มาเปรียบเทียบกับข้อบกพร่องที่ได้จากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที เพื่อพิจารณาว่าเอกสารการออกแบบซอฟต์แวร์ที่เลือกมาตรวจสอบ เพื่อปรับปรุงเทคนิคโอโออาร์ที เป็นตัวแทนที่ดีของเอกสารการออกแบบซอฟต์แวร์ทั้งหมด และพิจารณาจำนวนข้อบกพร่องที่สามารถตรวจพบในแต่ละขั้นตอนของเทคนิคโอโออาร์ที และพิจารณาจำนวนข้อบกพร่องที่มีในเอกสารการออกแบบซอฟต์แวร์แต่ละประเภทที่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ที เพื่อเป็นแนวทางในการปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง เพื่อลดระยะเวลาในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีลง โดยมีขั้นตอนในการดำเนินการเพื่อตอบสนองวัตถุประสงค์นี้ ดังนี้

1. การศึกษาประเภทของข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์จากงานวิจัยในอดีต เพื่อให้ทราบถึงประเภทของข้อบกพร่องที่เกิดขึ้นในเอกสารการออกแบบซอฟต์แวร์
2. ผู้วิจัยกำหนดให้มีผู้ที่ทำหน้าที่เป็นผู้ตรวจสอบ เพื่อดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุด้วยเทคนิคโอโออาร์ที ในส่วนนี้จะทำเฉพาะขั้นตอนการประชุมแนะนำ (Overview) ที่เป็นขั้นตอนที่จะอธิบายเอกสารการออกแบบซอฟต์แวร์ เทคนิคการอ่านซอฟต์แวร์ และเอกสารต่างๆที่ใช้ในการตรวจสอบให้ผู้ตรวจสอบทราบ เพื่อไม่ให้เกิดความผิดพลาดในการตรวจสอบ ขั้นตอนการจัดเตรียม (Preparation) เป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนตรวจสอบเอกสารการออกแบบซอฟต์แวร์ เพื่อค้นหาข้อบกพร่อง และขั้นตอนการประชุมตรวจสอบ (Inspection Meeting) เป็นขั้นตอนที่ทุกคนในคณะตรวจสอบต้องเข้าประชุมร่วมกัน เพื่อพิจารณาข้อบกพร่องที่ได้จากขั้นตอนการจัดเตรียมของผู้ตรวจสอบแต่ละคน ของขั้นตอนการตรวจสอบซอฟต์แวร์ (Software Inspection)

ขั้นตอนการจัดเตรียมจะไม่จำกัดเวลาในการตรวจสอบ เพื่อให้ผู้ตรวจสอบแต่ละคนสามารถค้นหาข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุให้ได้ครบถ้วนที่สุด ส่วนขั้นตอนการประชุมตรวจสอบจะต้องกำหนดระยะเวลาในการดำเนินการเพื่อไม่ให้ใช้ระยะเวลามากจนอาจจะทำให้ผู้ตรวจสอบเครียดเกินไป ทำให้ผลการประชุมตรวจสอบออกมาไม่ดี (อุมาพร นิลเอวะ, 2549)

การกำหนดระยะเวลาที่ใช้ ผู้วิจัยพิจารณาจากงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่เป็นการทดลองเพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคการอ่านซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ พบว่ากำหนดระยะเวลาที่ใช้ในขั้นตอนการประชุมตรวจสอบไม่เกิน 60 นาที ดังนั้นในงานวิจัยในส่วนนี้จึงกำหนดระยะเวลาในการดำเนินการในขั้นตอนการประชุมตรวจสอบไม่เกิน 60 นาที ต่อ 1 โครงงานเช่นกัน

ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ กำหนดให้มีผู้ตรวจสอบจำนวน 3 คน ต่อ 1 โครงงาน เพื่อให้ทำหน้าที่ได้สอดคล้องกับที่ Strauss และ Ebenau (1994) กล่าวไว้ว่าขนาดของคณะผู้ตรวจสอบอย่างน้อยที่สุดควรประกอบด้วย 3 คน คือ ผู้ดำเนินรายการ (Moderator) ผู้อ่าน (Reader) และผู้บันทึก (Recorder) ขั้นตอนการประชุมตรวจสอบเป็นขั้นตอนสำหรับให้ผู้ตรวจสอบประชุมร่วมกัน เพื่อค้นหาข้อบกพร่องและสรุปว่าเป็นข้อบกพร่องจริงหรือไม่ และทางผู้วิจัยต้องการให้สามารถมีผู้ชี้ขาดในการตัดสินใจได้ กรณีที่ผู้ตรวจสอบมีความเห็นไม่ตรงกัน ดังนั้นผู้วิจัยจึงกำหนดผู้ตรวจสอบไว้จำนวน 3 คน ต่อ 1 โครงงาน

3. การวิเคราะห์ข้อบกพร่องที่ค้นหาได้จากการตรวจสอบด้วยเทคนิคโอโออาร์ที โดยวิเคราะห์การกระจายตัวของประเภทข้อบกพร่องที่ศึกษาพบในงานวิจัยในอดีต เพื่อพิจารณาว่าเอกสารการออกแบบซอฟต์แวร์ที่เลือกมาตรวจสอบ เพื่อปรับปรุงเทคนิคโอโออาร์ที เป็นตัวแทนที่ดีของเอกสารการออกแบบซอฟต์แวร์ทั้งหมด และพิจารณาจำนวนข้อบกพร่องที่สามารถตรวจพบในแต่ละขั้นตอนของเทคนิคโอโออาร์ที และพิจารณาจำนวนข้อบกพร่องที่มีในเอกสารการออกแบบซอฟต์แวร์แต่ละประเภทที่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ที เพื่อใช้เป็นข้อมูลในการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีลง

4. การปรับปรุงเทคนิคโอโออาร์ที โดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง เพื่อให้ผู้ตรวจสอบใช้เวลาในการตรวจสอบน้อยลง แต่ว่ามีประสิทธิภาพและประสิทธิผลไม่น้อยกว่าเทคนิคโอโออาร์ทีเดิม เนื่องจากเทคนิคโอโออาร์ทีมีเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่มีขั้นตอนการตรวจสอบที่ใช้ในการค้นหาข้อบกพร่องมาก จึงใช้เวลาในการตรวจสอบมากกว่าเทคนิคอื่น (Conradi และคณะ, 2003)

ดังนั้นจึงควรลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีลง โดยสิ่งที่เป็นหลักเกณฑ์ในการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ที พิจารณาจากจำนวนข้อบกพร่องที่มีในเอกสารการออกแบบซอฟต์แวร์แต่ละประเภทที่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ที และจำนวนข้อบกพร่องที่สามารถตรวจพบในแต่ละขั้นตอน (Reading) ของเทคนิคโอโออาร์ที โดยต้องให้ครอบคลุมทั้งการอ่านแนวตั้ง (Vertical Reading) ของเทคนิคโอโออาร์ที ที่เป็นขั้นตอน (Reading) ที่ถูกพัฒนาขึ้น

เพื่อค้นหาข้อบกพร่องของขั้นตอนการออกแบบเทียบกับขั้นตอนการวิเคราะห์ความต้องการ เป็นสิ่งที่สำคัญมากเพื่อแสดงถึงความถูกต้องของการออกแบบ และทำให้เทคนิคโอโออาร์ทีแตกต่างจากเทคนิคการตรวจสอบซอฟต์แวร์เทคนิคอื่น เนื่องจากเทคนิคอื่นไม่มีขั้นตอนในการเปรียบเทียบความถูกต้องของการออกแบบเทียบกับการวิเคราะห์ความต้องการ และการอ่านแนวนอน (Horizontal Reading) ที่เป็นการตรวจสอบความสอดคล้องกันของเอกสารการออกแบบซอฟต์แวร์ต่างประเภทกันว่าเป็นไปในทิศทางเดียวกันหรือไม่

3.3.1 การเลือกเอกสารการออกแบบซอฟต์แวร์ เนื่องจากผู้วิจัยไม่สามารถนำเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยใช้แผนภาพยูเอ็มแอลทั้งหมด มาใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีในส่วนนี้ของงานวิจัยได้ ดังนั้นจึงต้องเลือกเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ ที่ออกแบบโดยใช้แผนภาพยูเอ็มแอล ที่เป็นตัวแทนที่ดี โดยเอกสารการออกแบบซอฟต์แวร์ที่เลือก ได้แก่ โครงงานการออกแบบระบบทางด้านธุรกิจที่ใช้แผนภาพยูเอ็มแอล ที่ออกแบบโดยนิสิตที่ศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ในรายวิชาการออกแบบระบบงานด้านธุรกิจ ที่เป็นผลงานกลุ่มของนิสิตตั้งแต่ปีการศึกษา 2543 ถึงปีการศึกษา 2550 จำนวนมากกว่า 120 โครงงาน ผู้วิจัยได้กำหนดจำนวนเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่เป็นตัวแทนของเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุทั้งหมดไว้ 15 โครงงาน

โดยการกำหนดจำนวนเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่เป็นตัวแทนของเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุทั้งหมด ผู้วิจัยพิจารณาจากงานวิจัยอื่นที่มีลักษณะการวิจัยคล้ายกับงานนี้ คือ งานวิจัยของ Travassos และคณะ (1999) เป็นการทดลองเพื่อค้นหาข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ ด้วยเทคนิคโอโออาร์ที เพื่อพิจารณาว่าการใช้เทคนิคโอโออาร์ทีช่วยในการค้นหาข้อบกพร่อง จะช่วยเพิ่มคุณภาพของซอฟต์แวร์ได้หรือไม่ โดยกำหนดเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ เพื่อใช้ในการตรวจสอบจำนวน 15 โครงงาน ดังนั้นงานวิจัยนี้ที่เป็นการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ โดยใช้เทคนิคโอโออาร์ทีในการค้นหาข้อบกพร่องเช่นกัน จึงกำหนดจำนวนเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ใช้ในการตรวจสอบไว้ 15 โครงงาน

ผลงานที่คัดเลือกมาเป็นผลงานกลุ่มของนิสิต ที่มีจำนวนยูสเคสไม่น้อยกว่า 5 ยูสเคส ที่แสดงการทำงานหลักของระบบ และเป็นระบบที่นำไปประยุกต์ใช้ทางธุรกิจด้วย โดยเอกสารการออกแบบซอฟต์แวร์ ต้องประกอบด้วยแผนภาพยูเอ็มแอลต่างๆ ได้แก่ แผนภาพยูสเคส (Use Case Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) แผนภาพสถานะ (State Machine Diagram) และ

แผนภาพคลาส (Class Diagram) นอกจากนี้ยังมีเอกสารอธิบายคลาส (Class Description) เอกสารอธิบายยูสเคส (Use Case Description) และเอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) ตามข้อกำหนดของเทคนิคโอโออาร์ทีที่กำหนดแนวทางการตรวจสอบว่าต้องตรวจสอบแผนภาพและเอกสารเหล่านี้ จากโครงงานทั้งหมด 120 โครงงาน มีเพียง 13 โครงงานเท่านั้นที่ถูกคัดเลือก เนื่องจากเป็นโครงงานที่มีแผนภาพและเอกสารที่ตรวจสอบด้วยเทคนิคโอโออาร์ทีครบตามที่กำหนดไว้ ซึ่งโครงงานส่วนใหญ่จะมีแผนภาพที่ขาดหายไปจากเอกสารการออกแบบระบบงานด้านธุรกิจที่ต้องตรวจสอบด้วยเทคนิคโอโออาร์ที คือ แผนภาพสถานะที่เป็นเอกสารการออกแบบซอฟต์แวร์ที่มีความสำคัญที่ต้องนำไปใช้ในส่วนของการดำเนินงานขั้นต่อไปของกระบวนการสร้างซอฟต์แวร์ (Conradi และคณะ, 2003) โครงงานที่ขาดแผนภาพสถานะจะไม่ได้รับคัดเลือก

อย่างไรก็ดีในจำนวน 13 โครงงานที่ถูกคัดเลือกมามี 6 โครงงานที่ขาดความสมบูรณ์ของเอกสารอธิบายคลาส เพราะไม่อธิบายการทำงานของเมท็อดในแต่ละคลาส แต่เนื่องจากเอกสารอธิบายคลาสไม่ได้เป็นเอกสารสำคัญที่แสดงถึงการออกแบบซอฟต์แวร์ที่จะถูกนำไปใช้ในขั้นตอนต่อไปของการพัฒนาซอฟต์แวร์ (Conradi และคณะ, 2003) ผู้วิจัยจึงยอมรับโครงงานที่ขาดความสมบูรณ์ในจุดนี้ได้ โครงงานที่เลือกมาเป็นหน่วยทดลองในงานวิจัยในส่วนนี้ เป็นโครงงานการออกแบบระบบงานด้านธุรกิจที่มีความหลากหลาย ได้แก่ ระบบการขาย ระบบคลังสินค้า ระบบการเช่า-คืน ระบบจัดการบริการลูกค้า ระบบรับคำสั่งซื้อและส่งมอบสินค้า ธุรกิจรับเหมาก่อสร้าง บริษัททัวร์ ธุรกิจดูแลรักษารถยนต์ ธุรกิจสปา เป็นต้น

3.3.2 การเลือกผู้ตรวจสอบเอกสารการออกแบบซอฟต์แวร์ เนื่องจากงานวิจัยในส่วนนี้เป็นการศึกษาข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์ เพื่อวิเคราะห์การกระจายตัวของข้อบกพร่องที่พบจากการใช้เทคนิคโอโออาร์ทีในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยวิธีเชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล ดังนั้นจึงต้องมีการเลือกผู้ตรวจสอบที่ทำหน้าที่ตรวจสอบเอกสารการออกแบบซอฟต์แวร์ โดยต้องเป็นตัวแทนที่ดีของผู้ตรวจสอบจริงในองค์กรรับจ้างพัฒนาซอฟต์แวร์ โดยผู้วิจัยต้องการจะเลือกพนักงานจากองค์กรรับจ้างพัฒนาซอฟต์แวร์จริงจำนวน 3 คน เนื่องจากในการตรวจสอบกำหนดให้มีผู้ตรวจสอบจำนวน 3 คน ต่อ 1 โครงงาน เพื่อให้ทำหน้าที่ได้สอดคล้องกับที่ Strauss และ Ebenau (1994) กล่าวไว้ว่า ขนาดของคณะผู้ตรวจสอบอย่างน้อยที่สุดควรประกอบด้วย 3 คน คือผู้ดำเนินรายการ (Moderator) ผู้อ่าน (Reader) และผู้บันทึก (Recorder)

ในขั้นตอนการจัดเตรียมของขั้นตอนการตรวจสอบซอฟต์แวร์ ที่ให้ผู้ตรวจสอบแต่ละคนดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ พนักงานจากองค์กรรับจ้างพัฒนา

ซอฟต์แวร์ไม่สามารถตรวจสอบเอกสารการออกแบบซอฟต์แวร์ได้ครบทั้ง 13 โครงงาน เนื่องจากไม่มีเวลาในการตรวจสอบที่เพียงพอ เพราะแต่ละโครงงานเป็นการออกแบบระบบงานด้านธุรกิจจริง ดังนั้นเอกสารการออกแบบจึงมีปริมาณมาก ทำให้ต้องใช้ระยะเวลาในการตรวจสอบมาก แต่พนักงานจากองค์กรรับจ้างพัฒนาซอฟต์แวร์ไม่มีเวลาที่เพียงพอในการดำเนินการตรวจสอบถึงคนละ 13 โครงงาน หรือถ้าจะดำเนินการตรวจสอบทั้งหมดผลที่ได้อาจจะไม่สมบูรณ์ ดังนั้นผู้วิจัยจึงเลือกผู้ตรวจสอบกลุ่มใหม่ที่สามารถเป็นตัวแทนที่ดีของผู้ตรวจสอบจริงในองค์กรรับจ้างพัฒนาซอฟต์แวร์ คือ นิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่เรียนครบทุกรายวิชาในหลักสูตรแล้ว เนื่องจากมีคุณสมบัติที่พร้อมจะเป็นผู้ตรวจสอบจริงในองค์กรรับจ้างพัฒนาซอฟต์แวร์

ดังนั้นผู้ตรวจสอบที่ดำเนินการตรวจสอบในงานวิจัยในส่วนนี้ แบ่งออกเป็น 2 กลุ่ม คือ (1) นิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่เรียนครบทุกรายวิชาในหลักสูตรแล้ว และ (2) พนักงานองค์กรรับจ้างพัฒนาซอฟต์แวร์ที่ทำหน้าที่เป็นนักวิเคราะห์ระบบ (System Analyst)และ ผู้ประกันคุณภาพซอฟต์แวร์ (Software Quality Assurance) โดยผู้วิจัยได้กำหนดให้การตรวจสอบแต่ละโครงงานต้องมีพนักงานองค์กรรับจ้างพัฒนาซอฟต์แวร์ตรวจสอบอย่างน้อยโครงงานละ 1 คน ซึ่งสามารถเป็นผู้ชี้ขาดในขั้นตอนการประชุมตรวจสอบได้ เนื่องจากเป็นผู้ที่มีประสบการณ์ในการตรวจสอบเอกสารการออกแบบระบบธุรกิจจริงแล้ว เพื่อให้สามารถค้นหาข้อบกพร่องที่แท้จริงของแต่ละโครงงานให้ได้มากที่สุด

3.3.3 เครื่องมือที่ใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ เครื่องมือที่ใช้ประกอบด้วย (1) เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ (2) เอกสารบันทึกรายการข้อบกพร่อง และ (3) เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล โดยมีรายละเอียดดังนี้

3.3.3.1 เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ เป็นเอกสารที่ใช้เป็นแนวทางในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที ผู้วิจัยใช้เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ ที่เคยใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) มาแล้ว เนื่องจากผ่านการทดสอบและใช้งานจริงในงานวิจัยดังกล่าวแล้ว จึงมีความถูกต้องและน่าเชื่อถือ สำหรับรูปแบบของเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ จะแบ่งการตรวจสอบออกเป็น 7 ขั้นตอน (Reading) หลักเพื่อตรวจสอบความถูกต้องของแต่ละแผนภาพในเอกสารแสดงการออกแบบซอฟต์แวร์เทียบกับความต้องการของระบบ และความสอดคล้องกันของเอกสารการออกแบบซอฟต์แวร์ต่างประเภท

กัน แต่ละขั้นตอน (Reading) ของการตรวจสอบจะแสดงแผนภาพที่เปรียบเทียบ วัตถุประสงค์ของขั้นตอน (Reading) นั้น เอกสารที่เข้าสู่กระบวนการ และขั้นตอนย่อยๆในการตรวจสอบ โดยรูปที่ 3-3 แสดงเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ อุมาพร นิลเอวะ (2549) ได้แปลและปรับปรุงรูปแบบเอกสารคำแนะนำให้มีความกระชับและสามารถเข้าใจได้ง่าย

Reading 1 : Sequence Diagram x Class Diagram			
วัตถุประสงค์	เพื่อตรวจสอบว่า Class และความสัมพันธ์ใน Class Diagram ถูกนำเสนอได้ตรงตามที่ถูกระบุใน Sequence Diagram		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	Sequence Diagram	- *ขีดเส้นใต้*ที่ Object หรือ Class ด้วยสีน้ำเงิน และที่ Message ด้วยสีเขียว - *วงกลม* Constraint และ Condition ของ Message ด้วยสีเหลือง	
2	Sequence Diagram → Class Diagram	*ตรวจสอบ*ทุกๆ Object หรือ Class (น้ำเงิน) ใน Sequence Diagram ว่าแสดงเป็น Class ใน Class Diagram หรือไม่	มี Object หรือ Class (น้ำเงิน) ที่แสดงใน Sequence Diagram แต่ไม่ได้แสดงเป็น Class ใน Class Diagram

รูปที่ 3-3 แสดงตัวอย่างเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที (อุมาพร นิลเอวะ, 2549)

3.3.3.2 เอกสารบันทึกรายการข้อบกพร่อง เป็นเอกสารบันทึกข้อบกพร่องที่ใช้ในการตรวจสอบซอฟต์แวร์ในงานวิจัยของ อุมาพร นิลเอวะ (2549) มาแล้ว เนื่องจากผ่านการทดสอบและใช้งานจริงในงานวิจัยดังกล่าวแล้ว จึงมีความถูกต้องและน่าเชื่อถือ เพื่อบันทึกผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ในส่วนนี้ของการวิจัย โดยสิ่งที่ต้องระบุในเอกสารบันทึกรายการข้อบกพร่อง ประกอบด้วย รายละเอียดของข้อบกพร่อง ตำแหน่งที่พบว่าพบที่แผนภาพใดและตำแหน่งใดของแผนภาพ และสามารถตรวจพบได้จากขั้นตอน (Reading) ใดของเทคนิคโอโออาร์ทีแสดงได้ดังรูปที่ 3-4

เอกสารบันทึกรายการข้อบกพร่อง

ข้อ	Reading / ข้อ	ชื่อเอกสาร	ตำแหน่งที่พบ	คำอธิบาย
1.				
2.				
3.				
4.				
5.				
6.				

รูปที่ 3-4 แสดงตัวอย่างเอกสารบันทึกรายการข้อบกพร่อง (อุมาพร นิลเอวะ, 2549)

3.3.3.3 เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล เป็นเอกสารแสดงความหมายของสัญลักษณ์ต่างๆที่แสดงในแผนภาพยูเอ็มแอล ที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) มาแล้ว ซึ่งจะประกอบด้วยสัญลักษณ์ของแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ โดยเอกสารนี้จะช่วยให้ผู้ตรวจสอบอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ได้เข้าใจมากยิ่งขึ้น (ดูรายละเอียดในภาคผนวก จ)

3.3.4 การดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ ผู้วิจัยกำหนดให้ผู้ตรวจสอบตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุ โดยกำหนดให้ดำเนินการเฉพาะขั้นตอนการประชุมแนะนำ (Overview) ขั้นตอนการจัดเตรียม (Preparation) และขั้นตอนการประชุมตรวจสอบ (Inspection Meeting) ของการตรวจสอบซอฟต์แวร์ (Software Inspection) เท่านั้น โดยมีรายละเอียดดังนี้

1. ผู้วิจัยเริ่มขั้นตอนการประชุมแนะนำของการตรวจสอบซอฟต์แวร์ โดยการแจกเอกสารต่างๆที่ใช้ในการตรวจสอบซอฟต์แวร์ ได้แก่ เอกสารแสดงการอธิบายความต้องการและเอกสารแสดงการออกแบบซอฟต์แวร์ เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่ใช้เป็นแนวทางในการตรวจสอบด้วยเทคนิคโอออาร์ที เอกสารบันทึกรายการข้อบกพร่อง สำหรับให้ผู้ตรวจสอบใช้บันทึกข้อบกพร่องที่สามารถตรวจพบ และเอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล จากนั้นอธิบายระบบและเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้ตรวจสอบให้ผู้ตรวจสอบเข้าใจ เพื่อให้ผู้ตรวจสอบสามารถค้นหาข้อบกพร่องได้อย่างถูกต้อง

2. หลังจากอธิบายระบบและเทคนิคการอ่านซอฟต์แวร์ที่ต้องใช้ในการตรวจสอบให้ผู้ตรวจสอบเข้าใจเรียบร้อยแล้ว จึงเริ่มเข้าสู่ขั้นตอนการจัดเตรียม โดยในขั้นตอนการจัดเตรียมผู้วิจัยกำหนดให้ผู้ตรวจสอบตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุโดยใช้เทคนิคโอออาร์ทีตามเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่กำหนดให้ เมื่อผู้ตรวจสอบสามารถตรวจพบ

ข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ ผู้วิจัยกำหนดให้ผู้ตรวจสอบบันทึกข้อบกพร่องที่ตรวจพบลงในเอกสารบันทึกรายการข้อบกพร่อง ขั้นตอนการจัดเตรียมเป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนค้นหาข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ให้ได้มากที่สุด ดังนั้นจึงไม่จำกัดระยะเวลาในการตรวจสอบของผู้ตรวจสอบ

3. หลังจากผู้ตรวจสอบแต่ละคนตรวจสอบเอกสารการออกแบบซอฟต์แวร์เสร็จเรียบร้อยแล้ว จึงดำเนินการในขั้นตอนการประชุมตรวจสอบ เป็นขั้นตอนที่ผู้ตรวจสอบทุกคนต้องเข้าประชุมร่วมกัน เพื่อชี้แจงเกี่ยวกับข้อบกพร่องที่ผู้ตรวจสอบแต่ละคนสามารถตรวจพบได้จากขั้นตอนการจัดเตรียม และให้ผู้ตรวจสอบคนอื่นพิจารณาข้อบกพร่องนั้นว่าเป็นข้อบกพร่องที่แท้จริงของเอกสารการออกแบบซอฟต์แวร์นั้นหรือไม่ ถ้าผู้ตรวจสอบยอมรับข้อบกพร่องนั้นแล้ว ข้อบกพร่องนั้นจะถูกบันทึกลงในเอกสารบันทึกรายการข้อบกพร่องของขั้นตอนการประชุมตรวจสอบ เพื่อแสดงข้อบกพร่องทั้งหมดที่สามารถตรวจพบได้ของเอกสารการออกแบบซอฟต์แวร์นั้น

ขั้นตอนการประชุมตรวจสอบจำเป็นต้องจำกัดระยะเวลาในการดำเนินการเพื่อไม่ให้ใช้เวลานานจนทำให้ผู้ตรวจสอบเครียดเกินไป โดยกำหนดให้ใช้เวลาไม่เกิน 60 นาที ต่อ 1 โครงงาน โดยผู้วิจัยจะนัดหมายให้ผู้ตรวจสอบทั้ง 3 คนของแต่ละโครงงานมาประชุมกัน สถานที่ที่ใช้ดำเนินการในขั้นตอนการประชุมตรวจสอบ คือ บริเวณใต้อาคารอนุสรณ์ 50 ปี คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย สาเหตุที่ผู้วิจัยเลือกสถานที่นี้เนื่องจากต้องการให้ผู้ตรวจสอบอยู่ในสถานที่ปลอดโปร่ง สามารถดื่มน้ำและรับประทานขนมได้ เพื่อไม่ให้ผู้ตรวจสอบเกิดความเครียดและอึดอัดในการประชุมมากเกินไป จะได้ตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่กำหนดได้อย่างถูกต้องแม่นยำ จากนั้นผู้วิจัยจะนำผลของข้อบกพร่องทั้งหมดจากเอกสารบันทึกรายการข้อบกพร่องของขั้นตอนการประชุมตรวจสอบไปวิเคราะห์ และดำเนินการปรับปรุงเทคนิคโอโออาร์ทีต่อไป

3.3.5 กรอบการวิเคราะห์ข้อมูล แหล่งเก็บข้อมูลของการตรวจสอบซอฟต์แวร์ ได้มาจากเอกสารบันทึกรายการข้อบกพร่องจากขั้นตอนการประชุมตรวจสอบ โดยข้อมูลจากเอกสารนี้จะนำไปวิเคราะห์การกระจายตัวของประเภทข้อบกพร่องที่ศึกษาได้จากงานวิจัยในอดีต เพื่อพิจารณาว่าเอกสารการออกแบบซอฟต์แวร์ที่เลือกมาตรวจสอบ เพื่อปรับปรุงเทคนิคโอโออาร์ที เป็นตัวแทนที่ดีของเอกสารการออกแบบซอฟต์แวร์ทั้งหมด และพิจารณาจำนวนข้อบกพร่องที่มีในเอกสารการออกแบบซอฟต์แวร์แต่ละประเภทที่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ที และพิจารณาจำนวนข้อบกพร่องที่สามารถตรวจพบในแต่ละขั้นตอนของเทคนิคโอโออาร์ที เพื่อใช้เป็นข้อมูลในการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีลง เพื่อให้ได้

เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ ที่คาดว่าจะทำให้การตรวจสอบมีประสิทธิภาพและประสิทธิผลดีกว่าเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีเดิม

3.4 การเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

จากวัตถุประสงค์ข้อที่ 2 ที่ต้องการเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ และวัตถุประสงค์ข้อที่ 3 ที่ต้องการเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ เพื่อเป็นการวัดประสิทธิภาพและประสิทธิผลของการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่กระทำในส่วนแรกของงานวิจัย

การวิจัยในส่วนนี้จะดำเนินการตรวจสอบซอฟต์แวร์ ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเท่านั้น และนำผลการตรวจสอบที่ได้ไปเปรียบเทียบกับผลการทดลองจากงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่ดำเนินการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ โดยงานวิจัยนี้จะกำหนดหน่วยทดลองที่จะมาเป็นผู้ที่ทำหน้าที่ตรวจสอบซอฟต์แวร์ ให้มีคุณสมบัติเหมือนหน่วยดลองของงานวิจัยของ อุมาพร นิลเอวะ (2549) ดังนั้นในการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์จะเป็นการนำผลการทดลองจากหน่วยทดลองที่มีคุณสมบัติเหมือนกัน ที่ดำเนินการวิจัยในช่วงเวลาที่ต่างกันมาเปรียบเทียบกัน เนื่องจากเป็นผลการทดลองจาก 2 งานวิจัย

โดยในการทดลองของงานวิจัยในส่วนนี้ จะทำเฉพาะขั้นตอนการประชุมแนะนำ (Overview) และขั้นตอนการจัดเตรียม (Preparation) ของขั้นตอนการตรวจสอบซอฟต์แวร์ (Software Inspection) เท่านั้น จะไม่ทำขั้นตอนการประชุมตรวจสอบ (Inspection Meeting) เนื่องจาก จากงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่ทดลองเกี่ยวกับการเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ออกแบบโดยวิธีการเชิงวัตถุ โดยเปรียบเทียบระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ สรุปไว้ว่า ในขั้นตอนการประชุมตรวจสอบไม่พบข้อบกพร่องเพิ่มขึ้นจากขั้นตอนการจัดเตรียม ที่ผู้ตรวจสอบแต่ละคนดำเนินการค้นหาข้อบกพร่องเลย ซึ่งสอดคล้องกับงานวิจัยของนักวิจัยบางกลุ่ม (Votta, 1993; Porter และคณะ, 1995; Johnson และ Tjahjono, 1998) ที่กล่าวว่าขั้นตอนการประชุมตรวจสอบไม่ได้ช่วยให้พบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมเลย ดังนั้นงานวิจัยนี้จึงตรวจสอบซอฟต์แวร์เฉพาะ

ขั้นตอนการประชุมแนะนำและขั้นตอนการจัดเตรียมเท่านั้น ไม่ทำขั้นตอนการประชุมตรวจสอบ โดยมีขั้นตอนในการดำเนินการเพื่อตอบสนองวัตถุประสงค์นี้ ดังนี้

1. ผู้วิจัยกำหนดให้ผู้ตรวจสอบดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว โดยจะนำผลการตรวจสอบที่ได้ไปเปรียบเทียบกับผลการทดลองของอุมาพร นิลเอวะ (2549) ที่ดำเนินการวิจัยโดยการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ เพื่อใช้เป็นปัจจัยในการพิจารณาประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ลงของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

2. สรุปผลการทดลองเพื่อเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ และเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ เพื่อพิจารณาถึงประสิทธิภาพและประสิทธิผลของการลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ดำเนินการในส่วนแรกของงานวิจัย

3.4.1 ตัวแปรที่เกี่ยวข้องกับการทดลอง เนื่องจากงานวิจัยนี้มีตัวแปรต้น 1 ตัว นั่นคือ เทคนิคการอ่านซอฟต์แวร์ โดยมีค่าที่เป็นไปได้ 3 ค่า คือ (1) เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว (2) เทคนิคโอโออาร์ทีเดิม และ (3) เทคนิคซีบีอาร์ ดังนั้นในงานวิจัยนี้จึงมีกลุ่มทดลอง 3 กลุ่มที่ได้รับทรีทเมนต์แตกต่างกัน คือกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีเดิม และกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว โดยงานวิจัยนี้ประกอบด้วย ตัวแปรต้น ตัวแปรตาม และตัวแปรควบคุมดังต่อไปนี้

3.4.1.1 ตัวแปรต้น งานวิจัยในส่วนนี้ให้ความสนใจว่าเทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกันส่งผลกับประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์หรือไม่ ดังนั้นตัวแปรต้นของงานวิจัยนี้มี 1 ตัวแปร คือเทคนิคการอ่านซอฟต์แวร์ ที่มี 3 ค่าคือ (1) เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว (2) เทคนิคโอโออาร์ทีเดิม และ (3) เทคนิคซีบีอาร์

3.4.1.2 ตัวแปรตาม เนื่องจากขั้นตอนนี้ของงานวิจัยเป็นการเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ จึงประกอบด้วยตัวแปรตาม 2 ตัว ดังต่อไปนี้

1. ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ สามารถวัดค่าตัวแปรนี้ได้จากจำนวนข้อบกพร่องเฉลี่ยที่ผู้ตรวจสอบพบในระยะเวลา 1 นาที (Sabalaiuskaite และคณะ, 2002)

$$\text{ประสิทธิภาพ} = \frac{\text{จำนวนข้อบกพร่องที่พบในการตรวจสอบ}}{\text{ระยะเวลาที่ใช้ในการตรวจสอบ (นาที)}}$$

2. ประสิทธิผลของการตรวจสอบซอฟต์แวร์ สามารถวัดค่าตัวแปรนี้ได้จากร้อยละของจำนวนข้อบกพร่องที่ผู้ตรวจสอบพบเทียบกับจำนวนข้อบกพร่องทั้งหมดที่มี (Sabalaiuskaite และคณะ, 2002)

$$\text{ประสิทธิผล} = \left(\frac{\text{จำนวนข้อบกพร่องที่พบในการตรวจสอบ}}{\text{จำนวนข้อบกพร่องทั้งหมดที่มี}} \right) \times 100$$

3.4.1.3 ตัวแปรควบคุม เพื่อให้ผลการทดลองเกิดจากเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้อย่างแท้จริง ผู้วิจัยจึงต้องควบคุมปัจจัยอื่นๆที่อาจส่งผลกับการทดลองให้มีความคงที่หรือเหมือนกันมากที่สุด โดยต้องควบคุมให้เหมือนกับงานวิจัยของ อุมาพร นิลเอวะ (2549) ทุกประการ เนื่องจากต้องการนำผลการทดลองของงานวิจัยนี้ ได้แก่ ผลการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ไปเปรียบเทียบกับผลการทดลองในงานวิจัยของ อุมาพร นิลเอวะ (2549) ได้แก่ ผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคซีบีอาร์ ดังนั้นงานวิจัยนี้จึงต้องควบคุมปัจจัยต่างๆให้เหมือนกับงานวิจัยของ อุมาพร นิลเอวะ (2549) โดยประกอบด้วยปัจจัยต่างๆดังนี้

1. เอกสารแสดงการอธิบายความต้องการและเอกสารแสดงการออกแบบซอฟต์แวร์ เป็นเอกสารแสดงการอธิบายความต้องการและเอกสารแสดงการออกแบบซอฟต์แวร์ที่เป็นระบบธุรกิจที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่ได้กำหนดข้อบกพร่องลงในแผนภาพและเอกสารการออกแบบซอฟต์แวร์เหมือนงานวิจัยของ อุมาพร นิลเอวะ (2549) ทุกประการ

โดยเอกสารแสดงคำอธิบายความต้องการและเอกสารแสดงการออกแบบซอฟต์แวร์ที่ใช้เป็นเอกสารการออกแบบระบบการเช่า- คืนรถ ที่กำหนดข้อบกพร่องในกลุ่มของข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง จำนวน 20 ข้อ ประกอบด้วยข้อบกพร่อง 3 ประเภท ดังนี้ (1) สิ่งสูญหาย จำนวน 8 ข้อ (2) สิ่งผิดพลาด จำนวน 8 ข้อ (3) สิ่งเพิ่มเติม จำนวน 8 ข้อ โดยเอกสารแสดงการออกแบบซอฟต์แวร์จะประกอบด้วยแผนภาพยูเอ็มแอลต่างๆ ได้แก่ แผนภาพยูสเคส (Use Case Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) แผนภาพสถานะ

(State Machine Diagram) และแผนภาพคลาส (Class Diagram) นอกจากนี้ยังมีเอกสารอธิบายคลาส (Class Description) เอกสารอธิบายยูสเคส (Use Case Description) และเอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) สาเหตุที่เลือกแผนภาพเหล่านี้มาตรวจสอบเนื่องจากเทคนิคโอโออาร์ทีได้กำหนดแนวทางการตรวจสอบไว้ว่าต้องตรวจสอบแผนภาพเหล่านี้ โดยทุกหน่วยทดลองจะได้รับเอกสารแสดงความต้องการของระบบและเอกสารแสดงการออกแบบซอฟต์แวร์ที่มีรายละเอียดเหมือนกันทุกประการ

2. เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ เป็นเอกสารที่ใช้เป็นแนวทางในการตรวจสอบซอฟต์แวร์สำหรับการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที ที่ใช้ในงานวิจัยของ อุมพร นิลเอวะ (2549) โดยเอกสารแนะนำในการตรวจสอบซอฟต์แวร์ที่ใช้ในงานวิจัยนี้จะต้องลดจำนวนขั้นตอน (Reading) ในการตรวจสอบของเอกสารคำแนะนำการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีลง ตามการปรับปรุงที่เป็นผลการศึกษาในขั้นตอนก่อนหน้า โดยในงานวิจัยนี้จะใช้เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ดังกล่าว มาดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ โดยหน่วยทดลองจะได้รับเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่มีรายละเอียดเหมือนกันทุกประการ

3. เอกสารบันทึกรายการข้อบกพร่อง ใช้ในการบันทึกผลการทดลองในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ โดยใช้บันทึกข้อบกพร่องที่ผู้ตรวจสอบตรวจพบด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เอกสารบันทึกรายการข้อบกพร่องเป็นเอกสารที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) โดยรายละเอียดที่ต้องบันทึกประกอบด้วย ชื่อผู้ตรวจสอบ รายละเอียดของข้อบกพร่อง ตำแหน่งที่พบว่าพบที่แผนภาพใดและตำแหน่งใดของแผนภาพ พร้อมทั้งต้องบันทึกระยะเวลาที่ผู้ตรวจสอบใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เพื่อค้นหาข้อบกพร่องทั้งหมดด้วย (ผู้วิจัยกำหนดระยะเวลาที่ใช้ในค้นหาข้อบกพร่องไม่เกิน 90 นาที เพื่อให้เท่ากับงานวิจัยของ อุมาพร นิลเอวะ (2549)) โดยทุกหน่วยทดลองจะได้รับเอกสารบันทึกรายการข้อบกพร่องที่มีลักษณะเหมือนกันทุกประการ

4. เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล เป็นเอกสารที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) โดยเอกสารดังกล่าวเป็นเอกสารที่แสดงความหมายสัญลักษณ์ของแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ เพื่อช่วยให้หน่วยทดลองอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ได้เข้าใจมากขึ้น โดยที่ทุกหน่วยทดลองจะได้รับเอกสารเอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอลที่มีลักษณะเหมือนกันทุกประการ

5. ระยะเวลาที่ใช้ในการทดลอง การทดลองในส่วนนี้ของงานวิจัยเป็นการตรวจสอบซอฟต์แวร์ในขั้นตอนการประชุมแนะนำ (Overview) และขั้นตอนการจัดเตรียม (Preparation) โดยขั้นตอนการประชุมแนะนำเป็นขั้นตอนการอธิบายให้หน่วยทดลองเข้าใจถึงเอกสารต่างๆที่ใช้ในการทดลอง โดยจะควบคุมให้ใช้เวลาไม่เกิน 30 นาที เพื่อให้เท่ากับงานวิจัยของ อุมาพร นิลเอวะ (2549) ซึ่งมีความเหมาะสมกับปริมาณเอกสารต่างๆที่ต้องอธิบายเพื่อให้ผู้ตรวจสอบเข้าใจ และขั้นตอนการจัดเตรียมเป็นขั้นตอนที่แต่ละหน่วยทดลองค้นหาข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์ที่กำหนด โดยควบคุมให้ทุกหน่วยทดลองใช้ระยะเวลาไม่เกิน 90 นาที เพื่อให้เท่ากับที่ระบุไว้ในงานวิจัยของ อุมาพร นิลเอวะ (2549)

3.4.2 การทดสอบสมมติฐาน งานวิจัยนี้ต้องการเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ ระหว่าง เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ โดยจะตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล ดังนั้น ผู้วิจัยจึงตั้งสมมติฐานในงานวิจัยดังต่อไปนี้

1. การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล

H_0 : ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยแผนภาพยูเอ็มแอลมีค่าไม่ต่างกัน

H_1 : ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยแผนภาพยูเอ็มแอลมีอย่างน้อย 1 คู่ที่มีค่าแตกต่างกัน

2. การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล

H_0 : ประสิทธิผลในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอลมีค่าไม่ต่างกัน

H_1 : ประสิทธิผลในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอลมีอย่างน้อย 1 คู่ที่มีค่าแตกต่างกัน

สาเหตุที่ทางผู้วิจัยตั้งสมมติฐานดังที่ได้กล่าวมาข้างต้น เนื่องจาก อุมาพร นิลเอวะ (2549) วิจัยและสรุปผลการวิจัยว่า เทคนิคโอโออาร์ทีไม่ได้มีประสิทธิภาพและประสิทธิผลมากกว่าเทคนิคซีบีอาร์ เนื่องจากเทคนิคโอโออาร์ทีมีขั้นตอนในการตรวจสอบซอฟต์แวร์มาก ดังนั้นจึงควรปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง โดยคำนึงถึงความสามารถในการค้นหาข้อบกพร่องที่มีในการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล ดังนั้นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วจึงน่าจะมีประสิทธิภาพและประสิทธิผลมากกว่าเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

3.4.3 ประชากรและหน่วยทดลอง สำหรับงานวิจัยนี้ประชากรคือ ผู้ตรวจสอบทั้งหมดจากองค์กรรับจ้างพัฒนาซอฟต์แวร์ เพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้ในการตรวจสอบซอฟต์แวร์ แต่ในทางปฏิบัติไม่สามารถนำผู้ตรวจสอบทั้งหมดมาทดลองได้ เนื่องจากผู้ตรวจสอบซอฟต์แวร์ในองค์กรรับจ้างพัฒนาซอฟต์แวร์มีงานในความรับผิดชอบที่ต้องทำในองค์กร จึงเป็นการยากที่จะนำผู้ตรวจสอบซอฟต์แวร์ในองค์กรรับจ้างพัฒนาซอฟต์แวร์มาใช้ในการทดลอง ผู้วิจัยจึงต้องเลือกหน่วยทดลองขึ้นมาเป็นตัวแทนประชากรดังกล่าว

หน่วยทดลองที่ดีจะต้องเป็นตัวแทนที่ดีของประชากร โดยสามารถให้ข้อมูลได้เช่นเดียวกับประชากร และลดค่าใช้จ่ายในการดำเนินการวิจัย (ศิริวรรณ เสรีรัตน์ และคณะ, 2541) โดยหน่วยทดลองที่นำมาใช้เพื่อเป็นตัวแทนของประชากรในการตอบวัตถุประสงค์ของงานวิจัยนี้ คือ นิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจและสาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่จบปริญญาตรีสาขาคอมพิวเตอร์ หรือสาขาที่เกี่ยวข้อง ที่มีจำนวนหน่วยกิตในวิชาที่เกี่ยวข้องกับคอมพิวเตอร์จำนวน 35 หน่วยกิต ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุแล้ว การกำหนดหน่วยทดลองเป็นนิสิตกลุ่มนี้ เพราะนิสิตที่เรียนสาขาวิชาดังกล่าวมีความรู้ความสามารถที่จะทำหน้าที่เป็นผู้ตรวจสอบซอฟต์แวร์ในอนาคตได้ ซึ่งน่าจะเป็นตัวแทนที่ดีของผู้ตรวจสอบซอฟต์แวร์ จึงเหมาะสมกับงานวิจัยนี้ที่ต้องการเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ โดยการนำเทคนิคการอ่านซอฟต์แวร์มาช่วยในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล

นอกจากนี้ยังสะดวกต่อผู้วิจัยในการติดต่อเพื่อให้มาร่วมการทดลอง และเป็นหน่วยทดลองที่มีคุณสมบัติเหมือนงานวิจัยของ อุมาพร นิลเอวะ (2549) เนื่องจากผู้วิจัยต้องการนำผลการทดลองที่ได้จากงานวิจัยนี้ คือ ผลการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วไป

เปรียบเทียบกับผลการทดลองที่ได้จากงานวิจัยของ อุมาพร นิลเอวะ (2549) คือ ผลการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

หน่วยทดลองที่ใช้ในงานวิจัยนี้ ได้แก่ นิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ และสาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุ จำนวน 24 คน เพื่อให้เท่ากับจำนวนของหน่วยทดลองที่ตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ในงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่กำหนดให้แต่ละเทคนิคมีจำนวนหน่วยทดลอง 24 คน เนื่องจากงานวิจัยนี้ต้องการนำผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เปรียบเทียบกับผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ จากงานวิจัยของ อุมาพร นิลเอวะ (2549) เพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของ เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

3.4.4 แนวทางการทดลอง งานวิจัยในส่วนนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) เนื่องจากกำหนดให้ตัวแปรที่สนใจเปลี่ยนค่า เพื่อวัดผลกระทบที่มีต่ออีกตัวแปร (ศิริวรรณ เสรีรัตน์ และคณะ, 2541; Babbie, 2001) โดยมีตัวแปรที่สนใจ คือ เทคนิคการอ่านซอฟต์แวร์ (Software Reading Techniques) ที่นำมาช่วยในการตรวจสอบซอฟต์แวร์ (Software Inspection) กล่าวคือ งานวิจัยในส่วนนี้ต้องการเปรียบเทียบประสิทธิภาพ (Efficiency) และประสิทธิผล (Effectiveness) ของการตรวจสอบซอฟต์แวร์โดยการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน เพื่อให้ผลการทดลองเกิดจากเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้อย่างแท้จริง

ในงานวิจัยนี้จึงต้องควบคุมตัวแปรอื่นๆได้แก่ (1) เอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล (UML) (2) เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ ที่นำมาใช้เป็นแนวทางในการตรวจสอบซอฟต์แวร์ โดยเป็นเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่ใช้ในการทดสอบ เป็นเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว (3) เอกสารบันทึกรายการข้อบกพร่อง (4) เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล และ (5) ระยะเวลาที่ใช้ในการตรวจสอบ โดยงานวิจัยในส่วนนี้จะดำเนินการตรวจสอบซอฟต์แวร์ในขั้นตอนการประชุมแนะนำและขั้นตอนการจัดเตรียม โดยขั้นตอนการประชุมแนะนำกำหนดให้ใช้ระยะเวลาไม่เกิน 30 นาที และขั้นตอนการจัดเตรียมกำหนดให้ใช้ระยะเวลาไม่เกิน 90 นาที

จากลักษณะการวิจัยดังกล่าวเห็นได้ว่างานวิจัยนี้ไม่สามารถนำประชากรจริง อันได้แก่ ผู้ตรวจสอบจากองค์กรรับจ้างพัฒนาซอฟต์แวร์มาใช้ในการทดลอง และทางผู้วิจัยไม่สามารถนำเอกสารการออกแบบซอฟต์แวร์ที่กลุ่มผู้ผลิตซอฟต์แวร์จัดทำขึ้นจริงมาใช้ในการทดลอง (การตรวจสอบซอฟต์แวร์) แต่จะใช้เอกสารแสดงการออกแบบซอฟต์แวร์ที่มีความถูกต้องสมบูรณ์และใส่ข้อบกพร่องตามที่งานวิจัยของ อุมาพร นิลเอวะ (2549) ใช้ เนื่องจากต้องการนำผลการทดลองของงานวิจัยนี้ไปเปรียบเทียบกับผลการทดลองของ อุมาพร นิลเอวะ (2549)

3.4.5 เครื่องมือที่ใช้ในการทดลองเครื่องมือที่ใช้ในการทดลองนี้ประกอบด้วย 4 ส่วนคือ (1) เอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์ (2) เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ (3) เอกสารบันทึกรายการบกพร่อง และ (4) เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล โดยมีรายละเอียดดังนี้

3.4.5.1 เอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์ ผู้วิจัยกำหนดให้ใช้เอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์เรื่องระบบเช่า-คืนรถ ที่เป็นเอกสารที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) โดยเป็นการวิจัยเพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคโอโออาร์ทีเดิมเทียบกับเทคนิคซีบีอาร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยวิธีการเชิงวัตถุ ประกอบด้วย เอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) แผนภาพยูสเคส (Use Case Diagram) เอกสารอธิบายยูสเคส (Use Case Description) แผนภาพซีเควนซ์ (Sequence Diagram) แผนภาพสถานะ (State Machine Diagram) แผนภาพคลาส (Class Diagram) และเอกสารอธิบายคลาส (Class Description) ผู้วิจัยเลือกเอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์จากงานวิจัยของ อุมาพร นิลเอวะ (2549) เนื่องจากต้องการนำผลการทดลองของงานวิจัยนี้ คือ ข้อบกพร่องที่ได้จากเอกสารบันทึกรายการข้อบกพร่อง ที่ตรวจสอบด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เปรียบเทียบกับผลการทดลองของงานวิจัยของ อุมาพร นิลเอวะ (2549) คือ ข้อบกพร่องที่ได้จากเอกสารบันทึกรายการข้อบกพร่อง ที่ตรวจสอบด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

3.4.5.2 เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ เอกสารคำแนะนำที่ใช้เป็นแนวทางในการตรวจสอบซอฟต์แวร์ เป็นเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ สำหรับการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที ที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) โดยได้ลดจำนวนขั้นตอน (Reading) ในการตรวจสอบซอฟต์แวร์ลงตามผลการศึกษาในส่วนแรกของงานวิจัยนี้ เพื่อใช้ตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

3.4.5.3 เอกสารบันทึกรายการข้อบกพร่อง ที่ใช้บันทึกผลการทดลองในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ โดยจะใช้เอกสารเดียวกับที่ใช้ในงานวิจัยของอุมาพร นิลเอวะ (2549) ประกอบด้วย ชื่อผู้ตรวจสอบ ข้อบกพร่องที่พบ รายละเอียดของข้อบกพร่อง ตำแหน่งที่พบว่าพบที่แผนภาพใดและตำแหน่งใดของแผนภาพ และต้องบันทึกระยะเวลาที่หน่วยทดลองใช้ในการค้นหาข้อบกพร่องทั้งหมดด้วย

3.4.5.4 เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล เป็นเอกสารแสดงความหมายของสัญลักษณ์ต่างๆที่แสดงในแผนภาพยูเอ็มแอล ที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) ประกอบด้วย สัญลักษณ์ของแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ โดยเอกสารนี้จะช่วยให้หน่วยทดลองอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ได้เข้าใจมากยิ่งขึ้น

3.4.6 การเก็บรวบรวมข้อมูล การทดลองในส่วนนี้ผู้วิจัยกำหนดให้หน่วยทดลองตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบโดยแผนภาพยูเอ็มแอล โดยใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว โดยในการทดลองจะทำเฉพาะส่วนของขั้นตอนการประชุมแนะนำและขั้นตอนการจัดเตรียม โดยมีขั้นตอนการเก็บข้อมูล ดังนี้

1. ผู้วิจัยเริ่มขั้นตอนการประชุมแนะนำของการตรวจสอบซอฟต์แวร์ โดยการแจกเอกสารต่างๆที่ใช้ในการทดลองได้แก่ เอกสารแสดงการอธิบายความต้องการและเอกสารแสดงการออกแบบซอฟต์แวร์ เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่ใช้เป็นแนวทางในการตรวจสอบด้วยเทคนิคโออาร์ทีที่ปรับปรุงแล้ว เอกสารบันทึกรายการข้อบกพร่อง สำหรับให้หน่วยทดลองใช้บันทึกข้อบกพร่องที่สามารถตรวจพบ และเอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล จากนั้นอธิบายระบบและเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้ตรวจสอบให้หน่วยทดลองเข้าใจ เพื่อให้หน่วยทดลองสามารถค้นหาข้อบกพร่องได้อย่างถูกต้อง ซึ่งผู้วิจัยกำหนดเวลาที่ใช้ในการทำขั้นตอนประชุมแนะนำไม่เกิน 30 นาที โดยการกำหนดเวลาที่ใช้ในการทำขั้นตอนการประชุมแนะนำ ผู้วิจัยกำหนดให้เท่ากับระยะเวลาที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) เนื่องจากต้องการควบคุมปัจจัยต่างๆให้เท่ากัน เพื่อให้ผลการทดลองที่ได้เกิดจากเทคนิคการตรวจสอบซอฟต์แวร์ที่กำหนดให้เท่านั้น

2. หลังจากอธิบายระบบและเทคนิคการอ่านซอฟต์แวร์ที่ต้องใช้ในการตรวจสอบให้หน่วยทดลองเข้าใจเรียบร้อยแล้ว จึงเริ่มเข้าสู่ขั้นตอนการจัดเตรียม โดยเริ่มจากหน่วยทดลองแต่ละคนค้นหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ โดยใช้เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่กำหนดให้เป็นแนวทางเพื่อช่วยในการค้นหาข้อบกพร่อง เมื่อหน่วยทดลองพบข้อบกพร่องจะบันทึกข้อบกพร่องเหล่านั้นในเอกสารบันทึกรายการข้อบกพร่อง โดยบันทึก

ขั้นตอน (Reading) ของการตรวจสอบที่สามารถตรวจพบข้อบกพร่องนั้น รายละเอียดข้อบกพร่อง ชื่อแผนภาพที่พบข้อบกพร่อง ตำแหน่งของข้อบกพร่องในแผนภาพ และเมื่อหน่วยทดลองตรวจสอบเอกสารการออกแบบซอฟต์แวร์เสร็จเรียบร้อยแล้ว หรือหมดเวลาตามที่ผู้วิจัยกำหนดให้ใช้ในขั้นตอนการจัดเตรียมเรียบร้อยแล้ว หน่วยทดลองต้องบันทึกระยะเวลาที่ใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วย เพื่อใช้ในการวิเคราะห์ข้อมูลต่อไป

ผู้วิจัยกำหนดให้หน่วยทดลองใช้เวลาในการค้นหาข้อกพร่องในขั้นตอนการจัดเตรียมไม่เกิน 90 นาที โดยการกำหนดเวลาที่ใช้ในการค้นหาข้อบกพร่อง ผู้วิจัยกำหนดให้มีระยะเวลาเท่ากันกับงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่ใช้ในขั้นตอนการจัดเตรียม เนื่องจากต้องการนำผลการทดลองของงานวิจัยนี้ไปเปรียบเทียบกับผลการทดลองของงานวิจัยของ อุมาพร นิลเอวะ (2549)

3.4.7 ความถูกต้อง (Validity) และความน่าเชื่อถือ (Reliability) ของข้อมูลที่เก็บ การทดลองในส่วนที่สองนี้ เป็นการทดลองเพื่อตอบวัตถุประสงค์ในการเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างทคนิคโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ และการเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ถ้าจะให้มีความถูกต้องและน่าเชื่อถือ จำเป็นต้องควบคุมปัจจัยอื่นที่อาจส่งผลกับตัวแปรตาม ได้แก่ การเลือกหน่วยทดลอง เครื่องมือที่ใช้ในการทดลอง และวิธีการเก็บข้อมูล เพื่อให้ผลการทดลองที่ได้เกิดจากเทคนิคการอ่านซอฟต์แวร์เท่านั้นโดยมีวิธีการดังต่อไปนี้

3.4.7.1 การเลือกหน่วยทดลอง งานวิจัยนี้เป็นการเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุ ที่ออกแบบโดยแผนภาพยูเอ็มแอล ดังนั้นหน่วยทดลองที่ผู้วิจัยต้องการนำมาใช้ในการทดลอง คือ ผู้ตรวจสอบทั้งหมดที่มาจากองค์กรรับจ้างพัฒนาซอฟต์แวร์ แต่ในทางปฏิบัติแล้วไม่สามารถทำได้ และเพื่อให้คุณสมบัติของหน่วยทดลองที่จะมาเป็นตัวแทนของประชากรในการทดลอง มีความใกล้เคียงกับประชากรที่ผู้วิจัยต้องการมากที่สุด ผู้วิจัยจึงเลือกตัวแทนของประชากรเป็นนิสิตในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ และสาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุ เพื่อให้มีคุณสมบัติเหมือนหน่วยทดลองของอุมาพร นิลเอวะ (2549) โดยกำหนดให้มีหน่วยทดลองจำนวน 24 คน เพื่อให้เท่ากับจำนวนหน่วยทดลองที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) เนื่องจากต้องการเปรียบเทียบผลการทดลอง

ของงานวิจัยนี้กับงานวิจัยของ อุมาพร นิลเอวะ (2549) จึงเป็นการกำหนดหน่วยทดลองที่มีคุณสมบัติเหมือนกัน ที่ดำเนินการทดลองคนละช่วงเวลา

3.4.7.2 เครื่องมือที่ใช้ในการทดลอง ประกอบด้วย 4 ส่วนคือ (1) เอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์ (2) เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ (3) เอกสารบันทึกรายการบกพร่อง และ (4) เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล ที่เป็นเครื่องมือที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่เป็นการวิจัยเพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยวิธีการเชิงวัตถุ เนื่องจากต้องการนำผลการทดลองของงานวิจัยนี้ คือ ข้อบกพร่องที่ได้จากเอกสารบันทึกรายการข้อบกพร่อง ที่ตรวจสอบด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เปรียบเทียบกับผลการทดลองของงานวิจัยของ อุมาพร นิลเอวะ (2549) คือ ข้อบกพร่องที่ได้จากเอกสารบันทึกรายการข้อบกพร่อง ที่ตรวจสอบด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ จึงต้องควบคุมให้ใช้เครื่องมือในการทดลองที่เหมือนกัน เพื่อให้ได้ผลการทดลองที่มีความน่าเชื่อถือ

3.4.7.3 การเก็บรวบรวมข้อมูล การเก็บข้อมูลจากหน่วยทดลอง ผู้วิจัยจะอธิบายเอกสารแสดงการออกแบบซอฟต์แวร์และเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ เพื่อควบคุมให้ทุกหน่วยทดลองมีความเข้าใจในเอกสารแสดงการออกแบบซอฟต์แวร์ และเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ตรงกัน และควบคุมระยะเวลาในการทดลองเท่ากัน สำหรับสถานที่ที่ใช้ในการทดลองเป็นห้อง IT Studio ที่ชั้น 9 ของตึกมหิตลาธิเบศร สาเหตุที่ผู้วิจัยเลือกสถานที่นี้ เนื่องจากเป็นสถานที่ที่มีความเงียบ ไม่มีเสียงรบกวน ที่จะทำลายสมาธิในการตรวจสอบซอฟต์แวร์ของหน่วยทดลอง เพื่อที่จะได้ตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่กำหนดได้อย่างถูกต้องแม่นยำ และเป็นสถานที่ที่สามารถรองรับจำนวนหน่วยทดลองทั้งหมด ที่มีจำนวน 24 คนได้พร้อมกัน เมื่อหน่วยทดลองดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เรียบร้อยแล้ว ผู้วิจัยจะนับจำนวนข้อบกพร่องที่แท้จริงที่หน่วยทดลองสามารถค้นหาได้จากเอกสารบันทึกรายการข้อบกพร่อง และนำข้อบกพร่องนั้นไปวิเคราะห์เพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ โดยเปรียบเทียบกับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ต่อไป

3.4.8 กรอบการวิเคราะห์ข้อมูล แหล่งเก็บข้อมูลของงานวิจัยนี้ได้มาจากเอกสารบันทึกรายการข้อบกพร่อง โดยข้อมูลจากเอกสารนี้จะนำไปเปรียบเทียบกับผลการทดลองจากงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่ตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ และวิเคราะห์ผลการทดลองเพื่อ

ตอบวัตถุประสงค์ที่ต้องการเปรียบเทียบประสิทธิภาพของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ และต้องการเปรียบเทียบประสิทธิผลของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ และทดสอบสมมติฐานของงานวิจัย โดยสามารถแบ่งกรอบการวิเคราะห์ข้อมูลได้ดังนี้

3.4.8.1 การเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยแผนภาพยูเอ็มแอล

การทดสอบสมมติฐานนี้ได้ข้อมูลจากเอกสารบันทึกรายงานข้อบกพร่อง โดยแบ่งการคำนวณออกเป็น 3 กลุ่มทดลอง คือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วในการตรวจสอบซอฟต์แวร์ (2) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีเดิมในการตรวจสอบซอฟต์แวร์ และ (3) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบซอฟต์แวร์

<u>สมมติฐาน</u>

$H_0: \mu_{EFF1} = \mu_{EFF2} = \mu_{EFF3}$

$H_1: \mu_{EFFi} \neq \mu_{EFFj}$ อย่างน้อย 1 คู่

โดย i ไม่เท่ากับ j และ i และ j = 1, 2, 3

<u>โดยที่</u>

μ_{EFF1} = ค่าเฉลี่ยของประสิทธิภาพหรือค่าเฉลี่ยของประสิทธิผลที่ได้จากการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วในการตรวจสอบซอฟต์แวร์

μ_{EFF2} = ค่าเฉลี่ยของประสิทธิภาพหรือค่าเฉลี่ยของประสิทธิผลที่ได้จากการใช้เทคนิคโอโออาร์ทีเดิมในการตรวจสอบซอฟต์แวร์

μ_{EFF3} = ค่าเฉลี่ยของประสิทธิภาพหรือค่าเฉลี่ยของประสิทธิผลที่ได้จากการใช้เทคนิคซีบีอาร์ในการตรวจสอบซอฟต์แวร์

หลังจากคำนวณค่าเฉลี่ยของประสิทธิภาพหรือค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของแต่ละหน่วยทดลองได้แล้ว จะเห็นว่าจากสมมติฐานที่ตั้งเป็นการ

เปรียบเทียบหน่วยทดลองที่แบ่งออกเป็น 3 กลุ่มที่ได้รับทรีทเมนต์ที่ต่างกัน ดังนั้นในการวิเคราะห์ข้อมูลจึงต้องใช้วิธีการวิเคราะห์ความแปรปรวน (Analysis of Variance) หรือเรียกโดยย่อว่า ANOVA

โดยถ้าจะใช้วิธีการวิเคราะห์ความแปรปรวนได้ ต้องพิจารณาเงื่อนไขของความแปรปรวนก่อน ประกอบด้วย (1) ค่าตัวแปรตามที่ได้จากหน่วยทดลองต้องเป็นการแจกแจงแบบปกติ (Normal Distribution) (2) ค่าความแปรปรวนของหน่วยทดลองคนละกลุ่มต้องเท่ากัน และ (3) การสุ่มหน่วยทดลองแต่ละกลุ่มต้องเป็นอิสระกัน (กัลยา วาณิชย์บัญชา, 2544) ถ้าไม่ผ่านเงื่อนไขควรใช้สถิติไม่ใช้พารามิเตอร์ในการวิเคราะห์ผลการทดลอง แต่ถ้าผ่านเงื่อนไขทั้งหมดจะสามารถใช้การวิเคราะห์ความแปรปรวนในการวิเคราะห์ผลการทดลองได้ (กัลยา วาณิชย์บัญชา, 2544) โดยการทดลองในส่วนนี้ของงานวิจัยเป็นการทดสอบตัวแปรต้นเพียงตัวเดียว คือ เทคนิคการอ่านซอฟต์แวร์ ที่มีค่าที่ใช้ทดสอบ 3 ค่า คือ เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และ เทคนิคซีบีอาร์ ดังนั้นในการวิเคราะห์ความแปรปรวน จึงเลือก การวิเคราะห์ความแปรปรวนแบบทางเดียว (One-way Analysis of Variance) ที่ใช้สำหรับการทดสอบความแตกต่างระหว่างค่าเฉลี่ยจากหน่วยทดลองตั้งแต่ 3 กลุ่มขึ้นไป เพื่อตรวจสอบว่าตัวแปรต้น 1 ตัว ที่มีค่าที่สามารถแบ่งกลุ่มได้ 3 กลุ่มขึ้นไปส่งผลแตกต่างกันหรือไม่ (กัลยา วาณิชย์บัญชา, 2544)

โดยการวิเคราะห์ความแปรปรวน เป็นวิธีการทางสถิติที่ใช้แยกความแปรปรวนหรือความผันแปรของข้อมูลทั้งหมดออกเป็นส่วนๆ สำหรับการวิเคราะห์ความแปรปรวนแบบทางเดียว ความแปรปรวนหรือความผันแปรทั้งหมด จะประกอบด้วย ความแปรปรวนอันเกิดจากความแตกต่างระหว่างกลุ่ม และความแปรปรวนภายในกลุ่ม โดยค่าสถิติที่ใช้ในการทดสอบสมมติฐาน คือ สถิติเอฟ (F) โดยเป็นสถิติที่ใช้เปรียบเทียบความแปรปรวนระหว่างกลุ่มกับความแปรปรวนภายในกลุ่ม หรือตรวจสอบจากค่า Sig. (Significance) ที่ได้จากตารางการวิเคราะห์ความแปรปรวน ในการทดสอบสมมติฐานก็ได้ (กัลยา วาณิชย์บัญชา, 2544)

ในการยอมรับหรือปฏิเสธสมมติฐาน กระทำได้โดยการเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนดไว้ โดยจะยอมรับ H_0 หรือค่าเฉลี่ยของประสิทธิภาพหรือค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ที่นำมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอลมีค่าไม่ต่างกัน แต่ถ้าค่า Sig. ที่ได้น้อยกว่า ระดับนัยสำคัญที่กำหนด จะปฏิเสธ H_0 และยอมรับ H_1 หรือค่าเฉลี่ยของประสิทธิภาพหรือค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ที่นำมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบ

ซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอลมีอย่างน้อย 1 คู่ที่มีค่าแตกต่างกัน (ศิริวรรณ เสรีรัตน์ และคณะ, 2541)

ถ้าต้องการทราบว่าประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์คู่ใดบ้างที่มีค่าไม่เท่ากัน และประสิทธิภาพหรือประสิทธิผลของการตรวจสอบด้วยเทคนิคใดมีค่ามากกว่ากัน ต้องเปรียบเทียบประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยวิธีการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) หรือการทดสอบความแตกต่างของค่าเฉลี่ยทีละคู่ (กัลยา วานิชย์บัญชา, 2550) งานวิจัยนี้ต้องการทราบค่าประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ดังนั้นจึงดำเนินการเปรียบเทียบประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม และเปรียบเทียบประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคซีบีอาร์ โดยใช้หลักการในการทดสอบดังนี้

1. การเปรียบเทียบประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม พิจารณาจากค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แสดงว่า ประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าเท่ากัน แต่ถ้าค่า Sig. มีค่าน้อยกว่าระดับนัยสำคัญจะปฏิเสธ H_0 และยอมรับ H_1 แสดงว่า ประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าไม่เท่ากัน ถ้าปฏิเสธ H_0 แล้วต้องการทราบว่าประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วหรือเทคนิคโอโออาร์ทีเดิมมีค่ามากกว่ากัน ต้องเปรียบเทียบที่ค่าเฉลี่ยของประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ ของทั้งสองเทคนิค

2. การเปรียบเทียบประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคซีบีอาร์ พิจารณาจากค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แสดงว่าประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วและเทคนิคซีบีอาร์มีค่าเท่ากัน แต่ถ้าค่า Sig. มีค่าน้อยกว่าระดับนัยสำคัญจะปฏิเสธ H_0 และยอมรับ H_1 แสดงว่า ประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคซีบีอาร์มีค่าไม่เท่ากัน ถ้าปฏิเสธ H_0 แล้วต้องการทราบว่าประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วหรือเทคนิค

ซีบีอาร์มีค่ามากกว่ากัน ต้องเปรียบเทียบที่ค่าเฉลี่ยของประสิทธิภาพหรือประสิทธิผลของการตรวจสอบซอฟต์แวร์ ของทั้งสองเทคนิค

บทที่ 4
ผลการวิเคราะห์ข้อมูล

4.1 บทนำ

บทนี้กล่าวถึงผลการศึกษาเพื่อปรับปรุงเทคนิคการอ่านซอฟต์แวร์ และผลการวิเคราะห์ข้อมูลที่ได้จากการทดลอง เพื่อนำมาตอบวัตถุประสงค์ของงานวิจัยที่กล่าวมาข้างต้น ได้แก่ การเปรียบเทียบประสิทธิภาพ (Efficiency) ประสิทธิผล (Effective) ในการตรวจสอบซอฟต์แวร์ (Software Inspection) ระหว่างการนำเทคนิคโอโออาร์ทีที่ได้จากการปรับปรุงแล้ว เทียบกับเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีเดิม ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอล ประกอบด้วยการวิเคราะห์ข้อมูลในลักษณะสถิติเชิงพรรณนา (Descriptive statistic) การตรวจสอบการแจกแจงของข้อมูล การทดสอบความแปรปรวน ผลการทดสอบสมมติฐานในลักษณะของสถิติเชิงอนุมาน (Inferential statistics) และผลการวิเคราะห์ข้อมูลเพิ่มเติม (Exploration)

4.2 การปรับปรุงเทคนิคโอโออาร์ที เพื่อลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง

หลังจากผู้วิจัยศึกษาประเภทของข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุจากงานวิจัยในอดีต และกำหนดแนวทางการศึกษาข้อมูล และเครื่องมือที่ต้องใช้ในการตรวจสอบซอฟต์แวร์เรียบร้อยแล้ว ได้แก่ เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที เอกสารบันทึกข้อบกพร่อง และเอกสารแสดงสัญลักษณ์ของแผนภาพยูเอ็มแอล โดยผู้วิจัยเลือกใช้เครื่องมือที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) มาแล้ว เนื่องจากผ่านการทดสอบจนกระทั่งเป็นเครื่องมือที่มีความน่าเชื่อถือ หลังจากนั้นผู้วิจัยจึงดำเนินการตามขั้นตอนการปรับปรุงเทคนิคโอโออาร์ทีที่กำหนดไว้ ดังนี้

4.2.1 การตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุด้วยเทคนิคโอโออาร์ที

การตรวจสอบซอฟต์แวร์ (Software Inspection) มีขั้นตอนการตรวจสอบซอฟต์แวร์มี 6 ขั้นตอน ดังนี้ (Frankovich, 1995) (1) การวางแผน (Planning) (2) การประชุมแนะนำ (Overview) (3) การจัดเตรียม (Preparation) (4) การประชุมตรวจสอบ (Inspection Meeting) (5) การแก้ไขใหม่ (Re-work) และ (6) การติดตาม (Follow Up)

งานวิจัยในส่วนนี้ต้องการปรับปรุงเทคนิคการอ่านซอฟต์แวร์ ที่เรียกว่า เทคนิคการอ่านเชิงวัตถุ (Object-Oriented Reading Technique) หรือเรียกโดยย่อว่า เทคนิคโอโออาร์ที (OORT) (Basili และคณะ, 1996) ดังนั้นจากขั้นตอนการตรวจสอบซอฟต์แวร์ทั้ง 6 ขั้นตอน งานวิจัยนี้ได้

เลือกดำเนินการเฉพาะขั้นตอนการประชุมแนะนำ (Overview) ขั้นตอนการจัดเตรียม (Preparation) และขั้นตอนการประชุมตรวจสอบ (Inspection Meeting) เท่านั้น เนื่องจากขั้นตอนการประชุมแนะนำ (Overview) เป็นขั้นตอนที่จะอธิบายเอกสารการออกแบบซอฟต์แวร์ เทคนิคการอ่านซอฟต์แวร์ และเอกสารต่างๆที่ใช้ในการตรวจสอบให้ผู้ตรวจสอบทราบ เพื่อไม่ให้เกิดความผิดพลาดในการตรวจสอบ ขั้นตอนการจัดเตรียมเป็นขั้นตอนที่สามารถนำเทคนิคการอ่านซอฟต์แวร์มาช่วยผู้ตรวจสอบแต่ละคนค้นหาและระบุข้อบกพร่อง และขั้นตอนการประชุมตรวจสอบเป็นขั้นตอนการพิจารณาข้อบกพร่องร่วมกันของผู้ที่เกี่ยวข้อง ว่าข้อบกพร่องที่ผู้ตรวจสอบแต่ละคนตรวจพบเป็นข้อบกพร่องจริงหรือไม่ และสรุปข้อบกพร่องทั้งหมดที่สามารถตรวจพบได้

งานวิจัยนี้กำหนดให้ใช้เทคนิคโอโออาร์ทีตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล ที่ออกแบบระบบงานด้านธุรกิจจำนวน 13 โครงงานที่ได้คัดเลือกไว้ กำหนดให้แต่ละโครงงานมีผู้ตรวจสอบ 3 คน ที่ดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เป็นรายบุคคลแยกกันในขั้นตอนการจัดเตรียมและนำผลการตรวจสอบมาวิเคราะห์ร่วมกันในขั้นตอนการประชุมตรวจสอบ และสรุปว่าเป็นข้อบกพร่องจริงหรือไม่ ดังนั้นทางผู้วิจัยจึงกำหนดจำนวนผู้ตรวจสอบเป็นจำนวนคี่ เพื่อให้สามารถมีผู้ชี้ขาดในการตัดสินใจได้กรณีที่ผู้ตรวจสอบมีความเห็นไม่ตรงกันของแต่ละโครงงาน และสามารถทำหน้าที่ได้สอดคล้องกับที่ Strauss และ Ebenau (1994) กล่าว คือ ขนาดของคณะผู้ตรวจสอบอย่างน้อยที่สุดควรประกอบด้วย 3 คน คือ ผู้ดำเนินรายการ (Moderator) ผู้อ่าน (Reader) และผู้บันทึก (Recorder)

ผู้ตรวจสอบที่ดำเนินการตรวจสอบในงานวิจัยในส่วนนี้ แบ่งออกเป็น 2 กลุ่ม คือ (1) นิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่เรียนครบทุกรายวิชาในหลักสูตรแล้ว และ (2) พนักงานองค์กรรับจ้างพัฒนาซอฟต์แวร์ที่ทำหน้าที่เป็นนักวิเคราะห์ระบบ (System Analyst) และ ผู้ประกันคุณภาพซอฟต์แวร์ (Software Quality Assurance) โดยผู้วิจัยได้กำหนดให้การตรวจสอบแต่ละโครงงานต้องมีพนักงานองค์กรรับจ้างพัฒนาซอฟต์แวร์ตรวจสอบอย่างน้อยโครงงานละ 1 คน เพื่อสามารถเป็นผู้ชี้ขาดในขั้นตอนการประชุมตรวจสอบได้ เนื่องจากเป็นผู้ที่มีประสบการณ์ในการตรวจสอบเอกสารการออกแบบระบบธุรกิจจริงแล้ว เพื่อให้สามารถค้นหาข้อบกพร่องที่แท้จริงของแต่ละโครงงานให้ได้มากที่สุด

ในการดำเนินการวิจัยเริ่มจากขั้นตอนการประชุมแนะนำ โดยผู้วิจัยได้ให้เอกสารการออกแบบระบบงานด้านธุรกิจและเครื่องมือที่ต้องใช้ในการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมแก่ผู้ตรวจสอบ โดยเครื่องมือที่ต้องใช้ในการตรวจสอบซอฟต์แวร์ ประกอบด้วย เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที เอกสารบันทึกรายการข้อบกพร่องเพื่อ

บันทึกข้อบกพร่องจากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ และเอกสารแสดงสัญลักษณ์ของแผนภาพยูเอ็มแอล พร้อมทั้งอธิบายวิธีการและขั้นตอนการตรวจสอบ และอธิบายวิธีใช้เครื่องมือต่างๆที่ต้องใช้ในการตรวจสอบ เพื่อให้ผู้ตรวจสอบเข้าใจถึงขั้นตอนและวิธีการดำเนินการตรวจสอบ และสามารถใช้เครื่องมือได้อย่างถูกต้องตรงกัน หลังจากนั้นจะดำเนินการในขั้นตอนการจัดเตรียม โดยผู้วิจัยได้ให้ผู้ตรวจสอบแต่ละคนดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ที่ได้รับ ตามเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ผู้ตรวจสอบได้รับ สำหรับขั้นตอนการจัดเตรียมผู้วิจัยไม่ได้กำหนดระยะเวลาที่ใช้ในการตรวจสอบ เพื่อให้ผู้ตรวจสอบสามารถค้นหาข้อบกพร่องที่มีในโครงงานได้มากที่สุด โดยผู้ตรวจสอบแต่ละคนจะได้จำนวนโครงงานที่ต้องตรวจสอบไม่เท่ากัน

โดยพยายามให้พนักงานองค์กรรับจ้างพัฒนาซอฟต์แวร์ได้ตรวจสอบเอกสารการออกแบบซอฟต์แวร์ในปริมาณที่มากที่สุด เนื่องจากเป็นผู้ที่มีประสบการณ์ในการตรวจสอบซอฟต์แวร์ของระบบจริงมาแล้ว แต่เนื่องจากพนักงานองค์กรรับจ้างพัฒนาซอฟต์แวร์ไม่มีเวลาตรวจสอบได้ครบ 13 โครงงาน ดังนั้นบางโครงงานจึงต้องให้นิสิตปัจจุบันตรวจสอบแทน โดยผู้วิจัยจะติดตามผลการตรวจสอบอย่างสม่ำเสมอ เพื่อไม่ให้ผู้ตรวจสอบละเลยการตรวจสอบ และหลังจากผู้ตรวจสอบดำเนินการในขั้นตอนการจัดเตรียมเสร็จเรียบร้อยแล้ว ผู้วิจัยจึงดำเนินการในขั้นตอนการประชุมตรวจสอบของแต่ละโครงงาน ในขั้นตอนนี้ผู้วิจัยได้จำกัดระยะเวลาในการดำเนินการ โดยกำหนดไว้ไม่เกิน 60 นาที ต่อ 1 โครงงาน เพื่อไม่ให้ใช้เวลาในการประชุมมากเกินไป อาจจะทำให้ผู้ตรวจสอบที่เข้าประชุมเครียดเกินไป ทำให้ผลการประชุมตรวจสอบออกมาไม่ดีเท่าที่ควรเมื่อเทียบกับระยะเวลาที่ใช้ (อุมาพร นิลเอวะ, 2549)

4.2.2 การวิเคราะห์ข้อบกพร่องที่ค้นหาได้จากการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที จากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ ที่เป็นโครงงานการออกแบบระบบงานด้านธุรกิจด้วยแผนภาพยูเอ็มแอล ที่จัดทำโดยนิสิต ที่เป็นส่วนหนึ่งของวิชาการวิเคราะห์และออกแบบระบบงานด้านธุรกิจ ในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่เป็นเอกสารการออกแบบที่ได้รับการตรวจสอบด้วยเทคนิคโอโออาร์ทีมีจำนวน 13 โครงงาน

โดยในการวิเคราะห์ข้อบกพร่องที่ค้นหาได้จากการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีจะดำเนินการเปรียบเทียบจำนวนข้อบกพร่องของประเภทข้อบกพร่องจากเอกสารการออกแบบซอฟต์แวร์ เพื่อใช้ยืนยันความเหมาะสมของเอกสารการออกแบบซอฟต์แวร์ว่าสามารถเป็นตัวแทนที่ดีของเอกสารการออกแบบซอฟต์แวร์ทั้งหมด ในการเป็นเอกสารการออกแบบซอฟต์แวร์ที่ใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ เพื่อใช้ปรับปรุงเทคนิคโอโออาร์ที

และเปรียบเทียบจำนวนข้อบกพร่องของแต่ละเอกสารการออกแบบซอฟต์แวร์ และเปรียบเทียบจำนวนข้อบกพร่องของขั้นตอน (Reading) การตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ที เพื่อใช้เป็นแนวทางในการปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง

4.2.2.1 การเปรียบเทียบจำนวนข้อบกพร่องของประเภทข้อบกพร่องจากเอกสารการออกแบบซอฟต์แวร์ ผู้วิจัยนำข้อบกพร่องจากเอกสารบันทึกรายการข้อบกพร่องของแต่ละโครงงาน ที่เป็นผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ของงานวิจัยในส่วนนี้ มาวิเคราะห์การกระจายตัวตามประเภทข้อบกพร่องในเอกสารการออกแบบจากงานวิจัยในอดีต โดยแบ่งการเปรียบเทียบออกเป็น 3 กลุ่ม ตามเงื่อนไขที่ใช้กำหนดกลุ่มของประเภทข้อบกพร่อง ประกอบด้วย (1) ประเภทข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส (2) ประเภทข้อบกพร่องที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง และ (3) ประเภทข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง ได้ผลดังนี้

1. การวิเคราะห์การกระจายตัวของข้อบกพร่องจากเอกสารบันทึกรายการข้อบกพร่องตามประเภทข้อบกพร่องที่ได้จากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ในส่วนนี้ตามประเภทของข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส ที่ได้จากงานวิจัยของ Lange และ Chaundron (2006) ประกอบด้วยข้อบกพร่อง 8 ประเภท ดังนี้ (1) ข้อความไม่มีชื่อ (2) ข้อความไม่มีวิธีดำเนินการ (3) ข้อความส่งผิดทิศทาง (4) คลาสไม่มีในแผนภาพซีเควนซ์ (5) อ็อบเจ็กต์ที่ไม่มีคลาสในแผนภาพคลาส (6) ยูสเคสไม่มีแผนภาพซีเควนซ์ (7) การเรียกใช้คลาสเดียวกันในแผนภาพเดียวกันมากกว่า 1 ครั้ง และ (8) เมท็อดไม่ถูกเรียกในแผนภาพซีเควนซ์ แต่มีประเภทของข้อบกพร่องบางประเภทไม่สามารถตรวจสอบได้จากเทคนิคโอโออาร์ที โดยเทคนิคโอโออาร์ทีสามารถตรวจสอบข้อบกพร่องในกลุ่มนี้ได้ 3 ประเภท ดังนี้ (1) ข้อความไม่มีวิธีดำเนินการ (2) อ็อบเจ็กต์ที่ไม่มีคลาสในแผนภาพคลาส (3) ยูสเคสไม่มีแผนภาพซีเควนซ์ โดยได้ผลการวิเคราะห์การกระจายตัวของประเภทข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส เฉพาะประเภทที่สามารถตรวจพบไปด้วยเทคนิคโอโออาร์ที ดังตารางที่ 4-1

ตารางที่ 4-1 ตารางแสดงการกระจายตัวของข้อบกพร่องที่ได้จากการตรวจสอบซอฟต์แวร์ ตามประเภทของข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส เฉพาะประเภทที่สามารถตรวจพบไปด้วยเทคนิคโอโออาร์ที

โครงงาน	ประเภทของข้อบกพร่อง		
	ข้อความไม่มีวิธีดำเนินการ	อ็อบเจ็กต์ที่ไม่มีคลาสในแผนภาพคลาส	ยูสเคสไม่มีในแผนภาพซีเควนซ์
1	15	-	4
2	9	2	10
3	10	-	21
4	26	-	15
5	5	2	3
6	42	5	8
7	123	16	16
8	2	-	15
9	16	-	5
10	43	2	21
11	1	-	14
12	16	-	10
13	12	1	11
รวม	**320**	**28**	**153**

ตารางที่ 4-1 แสดงจำนวนข้อบกพร่องที่สามารถตรวจสอบได้จากเอกสารการออกแบบซอฟต์แวร์ โดยแบ่งตามประเภทของข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส เฉพาะประเภทที่สามารถตรวจพบไปด้วยเทคนิคโอโออาร์ที โดยสามารถเรียงลำดับของปริมาณข้อบกพร่องที่เกิดในแต่ละประเภทของข้อบกพร่องที่สามารถตรวจสอบได้จากมากไปน้อย ดังนี้ (1) ข้อความไม่มีวิธีดำเนินการ (2) ยูสเคสไม่มีแผนภาพซีเควนซ์ (3) อ็อบเจ็กต์ที่ไม่มีคลาสในแผนภาพคลาส

2. การวิเคราะห์การกระจายตัวของข้อบกพร่องจากเอกสารบันทึกรายการข้อบกพร่องตามประเภทข้อบกพร่องที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง กล่าวไว้ในงานวิจัยของ Travassos และคณะ (1999) ประกอบด้วย (1) การละเลย (2) ข้อมูลไม่เป็นความจริง (3) ข้อมูลไม่สอดคล้องกัน (4) ข้อมูลไม่ชัดเจน และ (5) แสดงข้อมูลไม่จำเป็น ได้ผลดังตารางที่ 4-2

ตารางที่ 4-2 ตารางแสดงการกระจายตัวของข้อบกพร่องที่ได้จากการตรวจสอบซอฟต์แวร์ ตามประเภทของข้อบกพร่องที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง

โครงงาน	ประเภทข้อบกพร่อง				
	การละเลย	ข้อมูลไม่เป็นความจริง	ข้อมูลไม่สอดคล้องกัน	ข้อมูลไม่ชัดเจน	แสดงข้อมูลไม่จำเป็น
1	16	-	30	19	1
2	6	-	37	22	-
3	9	-	37	17	-
4	16	1	38	-	-
5	6	-	26	16	-
6	14	-	79	7	-
7	35	6	222	20	-
8	30	2	12	10	-
9	11	-	21	18	-
10	22	1	70	50	-
11	14	-	71	15	-
12	7	1	66	2	-
13	11	3	7	33	3
รวม	**197**	**14**	**716**	**229**	**4**

ตารางที่ 4-2 แสดงจำนวนข้อบกพร่องของประเภทของข้อบกพร่องที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง โดยประเภทข้อบกพร่องที่ได้จำนวนข้อบกพร่องเรียงลำดับจากมากไปน้อยของงานวิจัยนี้ ได้แก่ (1) ข้อมูลไม่สอดคล้องกัน (2) ข้อมูลไม่ชัดเจน (3) การละเลย (4) ข้อมูลไม่เป็นความจริง และ (5) แสดงข้อความไม่จำเป็น

จากงานวิจัยของ Travassos และคณะ (1999) ที่ใช้เทคนิคโอโออาร์ทีตรวจสอบเอกสารการออกแบบซอฟต์แวร์ ได้ผลสรุปว่าการอ่านแนวนอนสามารถตรวจพบข้อบกพร่องประเภทข้อมูลไม่สอดคล้องกันและข้อมูลไม่ชัดเจนมากที่สุด และการอ่านแนวตั้งสามารถตรวจพบข้อบกพร่องประเภทการละเลยและข้อมูลไม่เป็นจริงมากที่สุด ซึ่งผลที่ได้มีความสอดคล้องกับงานวิจัยนี้

3. การวิเคราะห์การกระจายตัวของข้อบกพร่องจากเอกสารบันทึกรายการข้อบกพร่องตามประเภทข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง เปิดเผยโดย Strauss และ Ebenau (1994) ประกอบด้วยประเภทของข้อบกพร่องดังนี้ (1) สิ่งสูญหาย (2) สิ่งผิดพลาด และ (3) สิ่งเพิ่มเติม ได้ผลดังตารางที่ 4-3

ตารางที่ 4-3 ตารางแสดงการกระจายตัวของข้อบกพร่องที่ได้จากการตรวจสอบซอฟต์แวร์ ตามประเภทของข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง

โครงงาน	ประเภทข้อบกพร่อง		
	สิ่งสูญหาย	สิ่งผิดพลาด	สิ่งเพิ่มเติม
1	8	58	-
2	13	52	-
3	39	20	4
4	14	41	-
5	13	35	-
6	30	70	-
7	262	24	1
8	45	9	-
9	37	13	-
10	64	69	-
11	5	93	2
12	12	64	-
13	32	42	3
รวม	**574**	**590**	**10**

ตารางที่ 4-3 แสดงจำนวนข้อบกพร่องของแต่ละประเภทของข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง โดยสามารถเรียงลำดับของปริมาณข้อบกพร่องที่เกิดในแต่ละประเภทของข้อบกพร่องที่สามารถตรวจสอบได้โดยแบ่งตามประเภทของข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่องจากมากไปน้อย ดังนี้ (1) สิ่งผิดพลาด (2) สิ่งสูญหาย (3) สิ่งเพิ่มเติม

จากงานวิจัยของ Strauss และ Ebenau (1994) พบว่าประเภทข้อบกพร่องที่สามารถตรวจพบมากที่สุด คือ สิ่งผิดพลาด รองลงมาคือ สิ่งสูญหาย และที่พบน้อยที่สุดคือ สิ่งเพิ่มเติม เมื่อเปรียบเทียบผลการทดลองระหว่างของ Strauss และ Ebenau (1994) กับผลการทดลองของงานวิจัยนี้พบว่าเป็นไปในทิศทางเดียวกัน

4.2.2.2 การเปรียบเทียบจำนวนข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์

จากผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ สามารถแจกแจงจำนวนข้อบกพร่องตามแผนภาพหรือเอกสารการออกแบบ ได้ดังตารางที่ 4-4

ตารางที่ 4-4 ตารางแสดงการแจกแจงจำนวนข้อบกพร่องตามแผนภาพหรือเอกสารการออกแบบซอฟต์แวร์

โครงงาน	เอกสารความต้องการ	เอกสารอธิบายยูสเคส	แผนภาพคลาส	เอกสารอธิบายคลาส	แผนภาพซีเควนซ์	แผนภาพสถานะ
1	2	2	31	1	8	5
2	1	-	37	10	11	6
3	1	-	16	18	25	3
4	1	-	27	5	21	1
5	1	-	24	7	11	5
6	8	-	61	8	18	5
7	1	-	210	26	37	13
8	2	-	5	23	23	1
9	1	-	31	10	6	2
10	1	-	55	45	27	5
11	1	-	39	13	15	9
12	1	-	43	22	25	9
13	2	-	32	13	17	14
รวม	**23**	**2**	**611**	**202**	**244**	**78**

ตารางที่ 4-4 แสดงจำนวนข้อบกพร่องจากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที โดยแสดงจำนวนข้อบกพร่องที่สามารถตรวจพบได้ในแต่ละแผนภาพหรือเอกสารในการออกแบบซอฟต์แวร์ สามารถเรียงลำดับของปริมาณข้อบกพร่องที่เกิดในแต่ละเอกสารการออกแบบซอฟต์แวร์จากมากไปน้อย ได้ดังนี้ (1) แผนภาพคลาส (2) แผนภาพซีเควนซ์ (3) เอกสารอธิบายคลาส (4) แผนภาพสถานะ (5) เอกสารความต้องการ และ (6) เอกสารอธิบายยูสเคส

จากงานวิจัยของ Mrdalj และคณะ (2004) (อ้างถึงใน อุมาพร นิลเอวะ, 2549) กล่าวไว้ว่าจำนวนของข้อบกพร่องที่พบในแต่ละแผนภาพ เรียงจากมากที่สุดไปน้อยที่สุด 3 ลำดับแรก เป็นดังนี้คือ (1) แผนภาพคลาส (2) แผนภาพซีเควนซ์ และ (3) แผนภาพสถานะ แต่ใน

งานวิจัยของ Mrdalj และคณะ (2004) ไม่ได้กล่าวถึงการตรวจสอบเอกสารอธิบายคลาส เนื่องจากเป็นการตรวจสอบเฉพาะแผนภาพของแผนภาพยูเอ็มแอลเท่านั้น เมื่อพิจารณาผลการทดลองของงานวิจัยนี้พบว่ามีความใกล้เคียงกับงานวิจัยของ Mrdalj และคณะ (2004) แต่งานวิจัยนี้ตรวจสอบเอกสารอธิบายคลาสด้วย

4.2.2.3 การเปรียบเทียบจำนวนข้อบกพร่องของขั้นตอน (Reading) การตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ที จากผลการตรวจสอบเอกสารการออกแบบซอฟต์แวร์จากเอกสารบันทึกรายการข้อบกพร่องของขั้นตอนการประชุมตรวจสอบของแต่ละโครงงาน สามารถแจกแจงจำนวนข้อบกพร่องของขั้นตอน (Reading) การตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ที ได้ดังตารางที่ 4-5

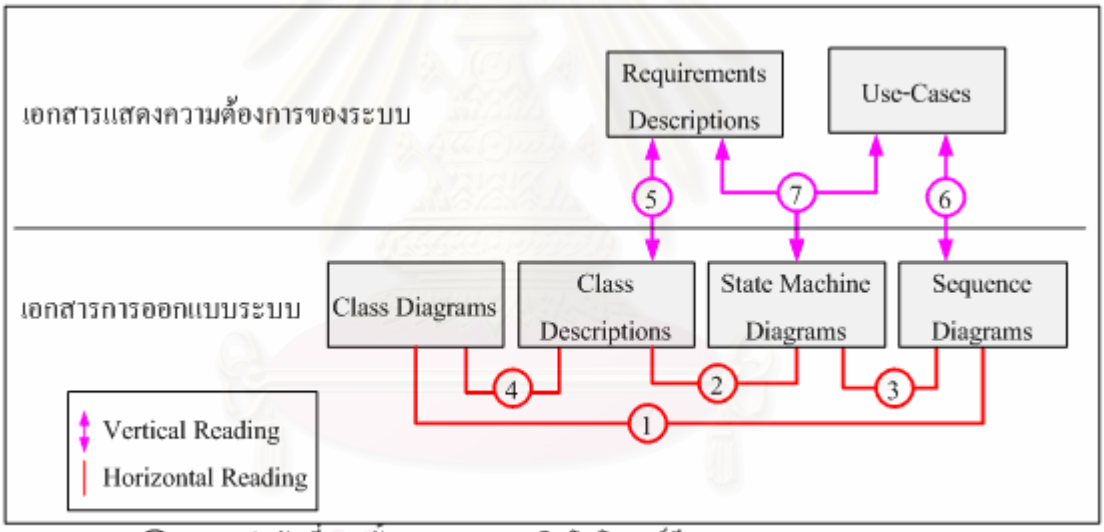
ตารางที่ 4-5 ตารางแสดงจำนวนข้อบกพร่องที่พบในแต่ละขั้นตอน (Reading) ของเทคนิคโอโออาร์ที จากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์

โครงงาน	จำนวนข้อบกพร่องในแต่ละขั้นตอน (Reading) ของเทคนิคโอโออาร์ที						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	19	5	4	15	17	4	2
2	21	1	6	16	10	10	1
3	16	3	4	3	15	21	1
4	28	1	6	2	2	15	1
5	16	1	1	10	9	10	1
6	58	5	6	5	14	12	1
7	200	7	11	25	22	21	1
8	2	1	8	9	17	15	2
9	18	0	5	13	7	6	1
10	51	4	6	9	41	21	1
11	28	1	11	16	9	10	1
12	39	2	14	21	9	14	1
13	21	4	13	9	13	11	6
รวม	**517**	**35**	**95**	**153**	**185**	**170**	**20**

ตารางที่ 4-5 แสดงจำนวนข้อบกพร่องจากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที โดยแสดงจำนวนข้อบกพร่องที่สามารถตรวจพบได้ในแต่ละขั้นตอน (Reading) ของเทคนิคโอโออาร์ที จากผลรวมข้อบกพร่องของโครงงานที่ผู้ตรวจสอบค้นหาได้ พบว่าขั้นตอนที่ 1 เป็นขั้นตอนที่ผู้ตรวจสอบสามารถค้นหาข้อบกพร่องได้ปริมาณมากที่สุด และ ขั้นตอนที่ 7 เป็นขั้นตอนที่ผู้ตรวจสอบสามารถค้นหาข้อบกพร่องได้น้อยที่สุด จากงานวิจัยของ Conradi และคณะ (2003) ที่เป็นการวิจัยที่ใช้เทคนิคโอโออาร์ทีตรวจสอบเอกสารการออกแบบซอฟต์แวร์ กล่าวว่าจากการตรวจสอบซอฟ์แวร์ด้วยเทคนิคโอโออาร์ที สรุปได้ว่าขั้นตอนที่ 1 เป็นขั้นตอนที่สามารถตรวจบข้อบกพร่องได้มากที่สุด และขั้นตอนที่ 7 เป็นขั้นตอนที่สามารถ

ตรวจพบข้อบกพร่องได้น้อยที่สุด ซึ่งผลการทดลองของงานวิจัยนี้มีความสอดคล้องกับงานวิจัยของ Conradi และคณะ (2003) เช่นกัน

4.2.3 การปรับปรุงเทคนิคโอโออาร์ที เพื่อลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง เทคนิคโอโออาร์ทีแบ่งออกเป็น 2 ส่วนหลัก คือ (1) การอ่านตามแนวนอน (Horizontal Reading) ช่วยในการตรวจสอบว่าแต่ละแผนภาพอธิบายการทำงานของระบบเดียวกันกล่าวคือแต่ละแผนภาพที่ออกแบบมีความสอดคล้องตรงกันหรือไม่ (2) การอ่านตามแนวตั้ง (Vertical Reading) ช่วยในการตรวจสอบว่าแผนภาพที่ออกแบบนำเสนอระบบที่ถูกต้อง กล่าวคือออกแบบได้ตรงตามความต้องการของซอฟต์แวร์ที่กำหนดไว้ โดยการอธิบายความต้องการของซอฟต์แวร์ของระบบดูได้จากเอกสารความต้องการของลูกค้า แผนภาพยูสเคส (Use Case Diagram) และเอกสารอธิบายยูสเคส (Use Case Description) แสดงดังรูปที่ 4-1



รูปที่ 4-1 แสดงรูปแบบการอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ของเทคนิคโอโออาร์ที (Travassos และคณะ, 1999)

จากรูปที่ 4-1 แสดงว่าเทคนิคโอโออาร์ทีมี 7 ขั้นตอนในการอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ โดยแต่ละขั้นตอนเป็นการเปรียบเทียบทีละ 2 แผนภาพหรือเอกสาร โดยเปรียบเทียบทั้งหมด 7 คู่ โดยเทคนิคโอโออาร์ทีได้แบ่งการเปรียบเทียบออกเป็น 7 ขั้นตอนในการอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ดังรูปที่ 4-1 โดยมีรายละเอียดดังต่อไปนี้

① ขั้นตอนการเปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพคลาส

② ขั้นตอนการเปรียบเทียบระหว่างแผนภาพสถานะและเอกสารอธิบายคลาส

③ ขั้นตอนการเปรียบเทียบระหว่างแผนภาพสถานะ และแผนภาพซีเควนซ์

④ ขั้นตอนการเปรียบเทียบระหว่างแผนภาพคลาส และเอกสารอธิบายคลาส

⑤ ขั้นตอนการเปรียบเทียบระหว่างเอกสารอธิบายคลาส และเอกสารอธิบายความต้องการซอฟต์แวร์

⑥ ขั้นตอนการเปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพยูสเคส พร้อมทั้งเอกสารอธิบายยูสเคส

⑦ ขั้นตอนการเปรียบเทียบระหว่างแผนภาพสถานะ และเอกสารอธิบายความต้องการซอฟต์แวร์ / แผนภาพยูสเคส พร้อมทั้งเอกสารอธิบายยูสเคส

ขั้นตอน (Reading) ของเทคนิคโอโออาร์ทีสามารถแบ่งเป็น 2 ส่วน คือ (1) การอ่านตามแนวนอน ในขั้นตอนที่ 1 ถึง ขั้นตอนที่ 4 และ (2) การอ่านตามแนวตั้ง ในขั้นตอนที่ 5 ถึง ขั้นตอนที่ 7 และเมื่อพิจารณาขั้นตอน (Reading) ของเทคนิคโอโออาร์ทีและผลการตรวจสอบซอฟต์แวร์จากเอกสารการออกแบบซอฟต์แวร์ ได้ผลดังตารางที่ 4-5 โดยสามารถเรียงลำดับขั้นตอน (Reading) ที่มีลำดับของปริมาณข้อบกพร่องจากมากไปน้อยได้ ดังนี้ (1) ขั้นตอนที่ 1 เปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพคลาส (2) ขั้นตอนที่ 5 เปรียบเทียบระหว่างเอกสารอธิบายคลาส และเอกสารอธิบายความต้องการซอฟต์แวร์ (3) ขั้นตอนที่ 6 เปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพยูสเคสพร้อมทั้งเอกสารอธิบายยูสเคส (4) ขั้นตอนที่ 4 เปรียบเทียบระหว่างแผนภาพคลาส และเอกสารอธิบายคลาส (5) ขั้นตอนที่ 3 เปรียบเทียบระหว่างแผนภาพสถานะ และแผนภาพซีเควนซ์ (6) ขั้นตอนที่ 2 เปรียบเทียบระหว่างแผนภาพสถานะและเอกสารอธิบายคลาส (7) ขั้นตอนที่ 7 เปรียบเทียบระหว่างแผนภาพสถานะ และเอกสารอธิบายความต้องการซอฟต์แวร์ / แผนภาพยูสเคส และเอกสารอธิบายยูสเคส และมีผลรวมของจำนวนข้อบกพร่องที่เกิดในแต่ละขั้นตอน (Reading) ของเทคนิคโอโออาร์ที ดังตารางที่ 4-6

ตารางที่ 4-6 ตารางแสดงผลสรุปการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ที

	Object Oriented Reading Technique (OORT)						
	การอ่านตามแนวนอน				**การอ่านตามแนวตั้ง**		
	1	**2**	**3**	**4**	**5**	**6**	**7**
จำนวนข้อบกพร่องรวม	517	35	95	153	185	170	20

มีงานวิจัยที่เกี่ยวข้องกับการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ที โดย Travassos และคณะ (1999) ทดลองเพื่อวัดความสามารถในการค้นหาข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ โดยใช้เทคนิคการอ่านซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเพื่อเพิ่มคุณภาพของซอฟต์แวร์ ผลการทดลองสรุปว่าเทคนิคโอโออาร์ทีสามารถทำให้ผู้ทดลองพึงพอใจผลการค้นหาข้อบกพร่องจากเทคนิคการอ่านนี้ โดยสามารถหาข้อบกพร่องได้โดยเฉลี่ย 56% ของข้อบกพร่องทั้งหมด แต่เทคนิคโอโออาร์ทีอาจจะต้องใช้เวลาตรวจสอบมากเนื่องจากมีเอกสารคำแนะนำเพื่อให้ผู้ตรวจสอบเข้าใจขั้นตอนการตรวจสอบโดยละเอียด ทำให้สามารถค้นหาข้อบกพร่องได้มาก เป็นที่พอใจของผู้ตรวจสอบ

อุมาพร นิลเอวะ (2549) ได้วิจัยเพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคโอโออาร์ทีกับเทคนิคซีบีอาร์ โดยนำมาใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล ผลการวิจัยแสดงว่าเทคนิคโอโออาร์ทีไม่ได้มีประสิทธิภาพและประสิทธิผลมากกว่าเทคนิคซีบีอาร์ ซึ่งอาจจะเกิดจากระยะเวลาในการตรวจสอบที่มีจำกัด ทำให้เทคนิคโอโออาร์ทีที่มีขั้นตอนการตรวจสอบที่มากกว่าสามารถตรวจสอบข้อบกพร่องได้น้อยกว่าเทคนิคซีบีอาร์ โดยได้กำหนดระยะเวลาของขั้นตอนการจัดเตรียมในการตรวจสอบซอฟต์แวร์ เพื่อให้ผู้ตรวจสอบแต่ละคนค้นหาข้อบกพร่องไว้ 90 นาที อาจจะเพียงพอสำหรับการค้นหาข้อบกพร่องตามขั้นตอนของเทคนิคซีบีอาร์ที่มีจำนวน เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์เพียง 1 หน้ากระดาษ แต่ไม่เพียงพอสำหรับเทคนิคโอโออาร์ทีที่มีจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่ค่อนข้างมากที่มีจำนวน 8 หน้ากระดาษ

จากเหตุผลดังกล่าวทำให้ผู้วิจัยเห็นถึงความสำคัญในการปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง อีกทั้งเอกสารคำแนะนำในการตรวจอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที สามารถตรวจพบข้อบกพร่องข้อเดียวกันได้จากหลายขั้นตอนของเอกสาร (Conradi และคณะ, 2003) ดังนั้นถ้าลดจำนวนเอกสารคำแนะนำในการ

ตรวจสอบซอฟต์แวร์ลง ข้อบกพร่องบางส่วนที่สามารถตรวจพบในขั้นตอนที่ถูกลดลงไป อาจจะสามารถตรวจพบในขั้นตอนที่ไม่ถูกตัดออกไปได้

จากการวิเคราะห์ผลการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีในการตรวจสอบเอกสารแสดงความต้องการและเอกสารการออกแบบซอฟต์แวร์ ผู้วิจัยจึงดำเนินการปรับปรุงเทคนิคโอโออาร์ที โดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง ถ้าพิจารณาจากจำนวนข้อบกพร่องที่สามารถตรวจพบได้จากเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ในแต่ละขั้นตอนของเทคนิคโอโออาร์ทีจากมากไปน้อย เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วจะประกอบด้วย ขั้นตอนที่ 1 ที่เปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพคลาส ขั้นตอนที่ 4 ที่เปรียบเทียบระหว่างแผนภาพคลาส และเอกสารอธิบายคลาส ขั้นตอนที่ 5 ที่เปรียบเทียบระหว่างเอกสารอธิบายคลาส และเอกสารอธิบายความต้องการซอฟต์แวร์ และขั้นตอนที่ 6 ที่เปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพยูสเคสพร้อมทั้งเอกสารอธิบายยูสเคส

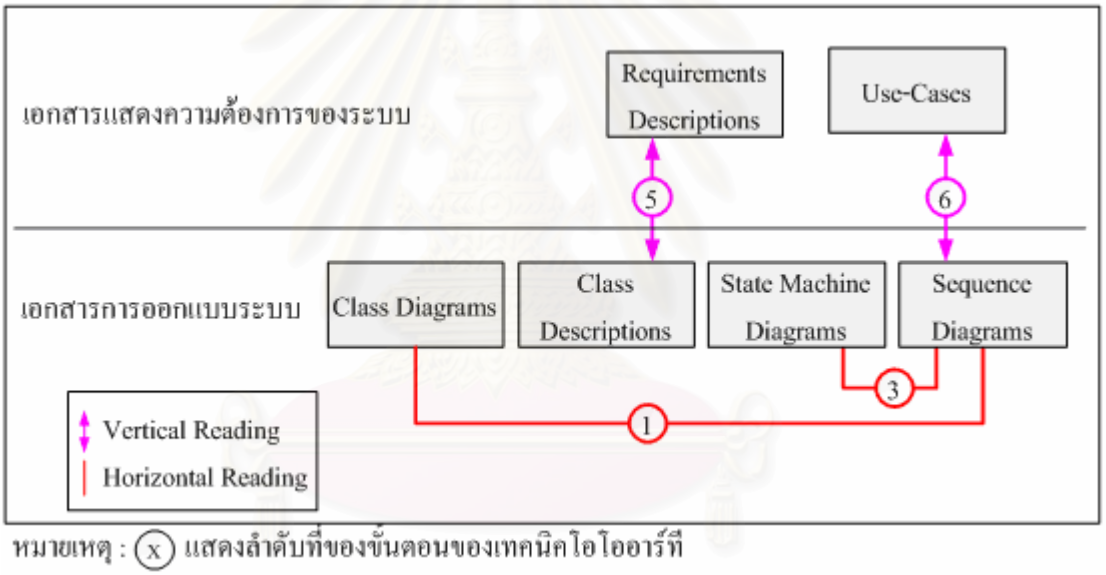
จากการพิจารณาเพียงจำนวนข้อบกพร่องที่สามารถตรวจพบได้จากเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ในแต่ละขั้นตอน ผู้วิจัยพบว่ายังไม่สามารถครอบคลุมเอกสารการออกแบบซอฟต์แวร์ที่ตรวจสอบได้ทั้งหมด ซึ่งแผนภาพสถานะเป็นเอกสารการออกแบบซอฟต์แวร์ที่ไม่ได้รับการตรวจสอบ แต่ Conradi และคณะ (2003) กล่าวว่าแผนภาพสถานะเป็นเอกสารสำคัญในส่วนของการออกแบบซอฟต์แวร์ ที่ต้องถูกนำไปใช้ในขั้นตอนต่อไปของการพัฒนาซอฟต์แวร์ ดังนั้นในการปรับปรุงเทคนิคโอโออาร์ทีผู้วิจัยจึงต้องการให้แผนภาพสถานะได้รับการตรวจสอบด้วย และยังต้องคำนึงถึงความครอบคลุมของเอกสารทั้งหมดที่ได้รับการตรวจสอบด้วยเทคนิคโอโออาร์ทีด้วย

จากการพิจารณาขั้นตอน (Reading) ของเทคนิคโอโออาร์ทีที่ได้ตรวจสอบแผนภาพสถานะ ผู้วิจัยพบว่าขั้นตอนที่ 3 ที่เปรียบเทียบระหว่างแผนภาพสถานะ และแผนภาพซีเควนซ์เป็นขั้นตอนที่สามารถตรวจพบข้อบกพร่องได้มากที่สุดจากขั้นตอนที่เหลือ ดังนั้นผู้วิจัยจึงเลือกขั้นตอน (Reading) นี้เป็นอีกหนึ่งขั้นตอนของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเมื่อพิจารณาเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ทีมีจำนวนขั้นตอน (Reading) ทั้งหมด 5 ขั้นตอน พบว่าเทคนิคโอโออาร์ทีที่ได้ยังคงมีเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่ยังมากอยู่

จากเหตุผลดังกล่าว ผู้วิจัยจึงต้องการที่จะลดจำนวนเอกสารคำแนะนำของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วลงอีก โดยถ้าพิจารณาจากขั้นตอน (Reading) ที่ได้รับคัดเลือกมาพบว่าขั้นตอนที่ 4 ที่เปรียบเทียบระหว่างแผนภาพคลาสและเอกสารอธิบายคลาส ซึ่ง Conradi และคณะ (2003) กล่าวว่าเป็นขั้นตอน (Reading) ที่ต้องการตรวจสอบความสอดคล้องกันของเอกสารทั้งสองเพื่อพิจารณาว่าเอกสารอธิบายคลาสเขียนสอดคล้องกับแผนภาพคลาสหรือไม่ ซึ่งเป็นขั้นตอนที่ไม่มี

ความจำเป็นที่ต้องทำ เนื่องจากเอกสารอธิบายคลาสไม่ได้เป็นเอกสารสำคัญที่แสดงถึงการออกแบบซอฟต์แวร์ที่ถูกนำไปใช้ในขั้นตอนต่อไปของการพัฒนาซอฟต์แวร์ ผู้วิจัยจึงตัดขั้นตอนนี้ออก เพื่อให้เป็นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

ดังนั้นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ที่ได้จากการพิจารณาถึงจำนวนข้อบกพร่องที่สามารถตรวจพบได้ในแต่ละขั้นตอนของเทคนิคโอโออาร์ที และการพิจารณาถึงความครอบคลุมการตรวจสอบเอกสารแสดงความต้องการและเอกสารการออกแบบซอฟต์แวร์ทั้งหมด จึงทำให้ผู้วิจัยเสนอเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ที่ประกอบด้วย ขั้นตอนที่ 1 ขั้นตอนที่ 3 ขั้นตอนที่ 5 และขั้นตอนที่ 6 ที่เปรียบเทียบทั้งหมด 4 คู่ ของเอกสารแสดงความต้องการและเอกสารการออกแบบซอฟต์แวร์ ดังรูปที่ 4-2



รูปที่ 4-2 แสดงรูปแบบการอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

จากรูปที่ 4-2 แสดงขั้นตอน (Reading) การตรวจสอบของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว โดยมี 4 ขั้นตอน (Reading) ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยแต่ละขั้นตอน (Reading) เป็นการเปรียบเทียบทีละ 2 แผนภาพหรือเอกสาร โดยเปรียบเทียบทั้งหมด 4 คู่ ดังรูปที่ 4-2 โดยมีรายละเอียดดังต่อไปนี้

(1) ขั้นตอนการเปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพคลาส

(3) ขั้นตอนการเปรียบเทียบระหว่างแผนภาพสถานะ และแผนภาพ

ซีเควนซ์

⑤ ขั้นตอนการเปรียบเทียบระหว่างเอกสารอธิบายคลาส และเอกสารอธิบายความต้องการซอฟต์แวร์

⑥ ขั้นตอนการเปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพยูสเคส พร้อมทั้งเอกสารอธิบายยูสเคส

4.3 การเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และ เทคนิคซีบีอาร์

หลังจากปรับปรุงเทคนิคโอโออาร์ที เพื่อลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง และกำหนดแผนแบบการทดลอง (Experimental Design) พร้อมทั้งจัดเตรียมเครื่องมือที่ต้องใช้ในการตรวจสอบซอฟต์แวร์ ได้แก่ เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เอกสารบันทึกรายการข้อบกพร่อง เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล และเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยแผนภาพยูเอ็มแอล โดยผู้วิจัยเลือกใช้เครื่องมือในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ ที่เคยใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) มาแล้ว เนื่องจากต้องการเปรียบเทียบผลการวิจัยของงานวิจัยนี้กับงานวิจัยของ อุมาพร นิลเอวะ (2549) งานวิจัยในส่วนนี้เป็นการใช้เทคนิคการอ่านซอฟต์แวร์ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ และนำผลการทดลองที่ได้ ซึ่งก็คือผลจากการตรวจสอบซอฟต์แวร์จากเอกสารบันทึกรายการข้อบกพร่องของผู้ตรวจสอบ ไปเปรียบเทียบกับเทคนิคการอ่านซอฟต์แวร์อื่น ได้แก่ เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ จากงานวิจัยของอุมาพร นิลเอวะ (2549) เพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลของเทคนิคการอ่านซอฟต์แวร์ทั้ง 3 เทคนิค

ผู้วิจัยดำเนินการทดลองตามแผนแบบการทดลองที่กำหนดไว้ กล่าวคือ เก็บข้อมูลจากหน่วยทดลองที่เป็นนิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ และ สาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการพัฒนาซอฟต์แวร์เชิงวัตถุแล้ว เพื่อให้มีคุณสมบัติเหมือนหน่วยทดลองของอุมาพร นิลเอวะ (2549) โดยกำหนดให้มีจำนวนหน่วยทดลอง 24 คน เพื่อให้เท่ากับจำนวนหน่วยทดลองที่ใช้ในงานวิจัยของอุมาพร นิลเอวะ (2549) เพราะต้องการเปรียบเทียบผลการทดลองของงานวิจัยนี้กับงานวิจัยของอุมาพร นิลเอวะ (2549)

ในการทดลองหน่วยทดลองจะตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิค

โอโออาร์ทีที่ปรับปรุงแล้ว เพื่อค้นหาข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ที่ทางผู้วิจัยกำหนดไว้ โดยผู้วิจัยเก็บรวบรวมข้อมูลและนำข้อมูลที่ได้มาวิเคราะห์ โดยเปรียบเทียบกับผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์จากงานวิจัยของ อุมาพร นิลเอวะ (2549) เพื่อตอบวัตถุประสงค์ของงานวิจัยที่ต้องการเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์โดยการเปรียบเทียบ (1) ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ (2) ประสิทธิผลของการตรวจสอบซอฟต์แวร์

4.3.1 การวิเคราะห์ข้อมูลในลักษณะสถิติเชิงพรรณนา (Descriptive Statistics)

งานวิจัยในส่วนนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) เพื่อเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ดังที่กล่าวมาแล้วข้างต้น โดยมีหน่วยทดลองทั้งหมด 24 คน สำหรับทดลองในขั้นตอนการประชุมแนะนำ (Overview) และขั้นตอนการจัดเตรียม (Preparation) ของการตรวจสอบซอฟต์แวร์ (Software Inspection) ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เพื่อนำผลการทดลองไปเปรียบเทียบกับผลการทดลองของอุมาพร นิลเอวะ (2549) โดยกำหนดระยะเวลาที่ใช้ในการดำเนินการในขั้นตอนการประชุมแนะนำไม่เกิน 30 นาที และกำหนดระยะเวลาในการดำเนินการของขั้นตอนการจัดเตรียมไม่เกิน 90 นาที เพื่อให้เท่ากับระยะเวลาที่กำหนดในงานวิจัยของ อุมาพร นิลเอวะ (2549) ผลการทดลองสามารถแสดงค่าสถิติของจำนวนข้อบกพร่อง และระยะเวลาที่ใช้ในการตรวจสอบ ดังตารางที่ 4-7

ตารางที่ 4-7 ตารางแสดงค่าสถิติของจำนวนข้อบกพร่อง และระยะเวลาที่ใช้ในการตรวจสอบด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

ตัวแปร	ค่าต่ำสุด/ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
จำนวนข้อบกพร่อง	3 / 11	6.625	2.242
ระยะเวลาที่ใช้ (นาที)	75 / 90	85.833	7.173

ตารางที่ 4-7 แสดงค่าที่ได้จากการตรวจสอบซอฟต์แวร์ของหน่วยทดลอง 2 ค่า คือ (1) ค่าเฉลี่ยของข้อบกพร่อง และ (2) ค่าเฉลี่ยของระยะเวลาที่ใช้ เมื่อนำผลการทดลองของงานวิจัยนี้ไปเปรียบเทียบกับผลการทดลองของอุมาพร นิลเอวะ (2549) ที่ตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ สามารถแสดงค่าสถิติของจำนวนข้อบกพร่อง และระยะเวลาที่ใช้ในการตรวจสอบ ได้ดังตารางที่ 4-8

ตารางที่ 4-8 ตารางแสดงค่าสถิติของจำนวนข้อบกพร่อง และระยะเวลาที่ใช้ในการตรวจสอบด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด/ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
จำนวนข้อบกพร่อง	โอโออาร์ทีที่ปรับปรุงแล้ว	24	3 / 11	6.625	2.242
	โอโออาร์ทีเดิม	24	1 / 10	5.500	2.303
	ซีบีอาร์	24	3 / 15	8.000	2.670
ระยะเวลาที่ใช้ (นาที)	โอโออาร์ทีที่ปรับปรุงแล้ว	24	75 / 90	85.833	7.173
	โอโออาร์ทีเดิม	24	85 / 90	89.583	1.412
	ซีบีอาร์	24	65 / 90	81.625	9.016

ตารางที่ 4-8 แสดงค่าสถิติของจำนวนข้อบกพร่อง และระยะเวลาที่ใช้ในการตรวจสอบด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ค่าที่ได้มี 2 ค่า คือ (1) ค่าเฉลี่ยของข้อบกพร่อง ซึ่งเทคนิคซีบีอาร์มีค่าสูงสุด รองลงมาเป็นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าน้อยที่สุด และ (2) ค่าเฉลี่ยของระยะเวลาที่ใช้ ซึ่งเทคนิคซีบีอาร์มีค่าต่ำสุด รองลงมาเป็นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าสูงสุด

เมื่อนำผลการทดลองของงานวิจัยนี้ที่เป็นผลการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และผลการทดลองของ อุมาพร นิลเอวะ (2549) ที่เป็นผลการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ไปหาค่าประสิทธิภาพและประสิทธิผลด้วยวิธีการด้านล่าง จะได้ผลดังตารางที่ 4-9

ตารางที่ 4-9 ตารางแสดงค่าประสิทธิภาพและประสิทธิผลของผลการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด/ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	โอโออาร์ทีที่ปรับปรุงแล้ว	24	0.033 / 0.122	0.078	0.0256
	โอโออาร์ทีเดิม	24	0.22 / 0.111	0.061	0.0259
	ซีบีอาร์	24	0.036 / 0.172	0.098	0.0315
ประสิทธิผล (%)	โอโออาร์ทีที่ปรับปรุงแล้ว	24	15 / 55	33.13	11.211
	โอโออาร์ทีเดิม	24	5 / 50	27.5	11.516
	ซีบีอาร์	24	15 / 75	40	13.351

ตารางที่ 4-9 แสดงค่าสถิติของค่าประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ค่าที่ได้มี 2 ค่า คือ (1) ค่าเฉลี่ยของประสิทธิภาพ ซึ่งเทคนิคซีบีอาร์มีค่าสูงสุด รองลงมาเป็นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าต่ำที่สุด และ (2) ค่าเฉลี่ยของประสิทธิผล ซึ่งเทคนิคซีบีอาร์มีค่าสูงสุด รองลงมาเป็นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าต่ำสุด

4.3.2 การทดสอบสมมติฐาน

ผู้วิจัยเลือกการวิเคราะห์ความแปรปรวน (Analysis of Variance (ANOVA)) เป็นเทคนิคการวิเคราะห์เพื่อทดสอบสมมติฐาน เนื่องจากสมมติฐานที่ตั้งเป็นการเปรียบเทียบหน่วยทดลองที่แบ่งออกเป็น 3 กลุ่มที่ได้รับทรีทเมนต์ที่ต่างกัน โดยการทดลองในส่วนนี้ของงานวิจัยเป็นการทดสอบตัวแปรต้นเพียงตัวเดียว คือ เทคนิคการอ่านซอฟต์แวร์ (Software Reading Technique) ที่มีค่าที่ใช้ทดสอบ 3 ค่า คือ เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และ เทคนิคซีบีอาร์ ดังนั้นในการวิเคราะห์ความแปรปรวน จึงเลือก การวิเคราะห์ความแปรปรวนแบบทางเดียว (One-way Analysis of Variance) ที่ใช้สำหรับการทดสอบความแตกต่างระหว่างค่าเฉลี่ยจากหน่วยทดลองตั้งแต่ 3 กลุ่มขึ้นไป เพื่อตรวจสอบว่าตัวแปรต้น 1 ตัว ที่มีค่าที่สามารถแบ่งกลุ่มได้ 3 กลุ่มขึ้นไปส่งผลแตกต่างกันหรือไม่

สำหรับการวิเคราะห์ความแปรปรวนแบบทางเดียว ความแปรปรวนหรือความผันแปรทั้งหมด ประกอบด้วย ความแปรปรวนอันเกิดจากความแตกต่างระหว่างกลุ่ม และความ

แปรปรวนภายในกลุ่ม โดยค่าสถิติที่ใช้ในการทดสอบสมมติฐาน คือ สถิติเอฟ (F) โดยเป็นสถิติที่ใช้เปรียบเทียบความแปรปรวนระหว่างกลุ่มกับความแปรปรวนภายในกลุ่ม (กัลยา วาณิชย์บัญชา, 2544)

ในการใช้การวิเคราะห์ความแปรปรวนจะต้องพิจารณาเงื่อนไขของการวิเคราะห์ความแปรปรวน 3 ประการ ดังนี้ (1) ค่าตัวแปรตามที่ได้จากหน่วยทดลองต้องเป็นการแจกแจงแบบปกติ (Normal Distribution) (2) ค่าความแปรปรวนของหน่วยทดลองคนละกลุ่มต้องเท่ากัน และ (3) การสุ่มหน่วยทดลองแต่ละกลุ่มต้องเป็นอิสระกัน (กัลยา วาณิชย์บัญชา, 2544)

สำหรับการสุ่มหน่วยทดลองทั้งของงานวิจัยนี้และงานวิจัยของ อุมาพร นิลเอวะ (2549) ได้เลือกมาจากหน่วยทดลองที่คุณสมบัติเหมือนกัน คือ นิสิตในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ และสาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุ โดยในการเลือกหน่วยทดลองนั้น ไม่มีการตั้งข้อจำกัด หรือกำหนดคุณสมบัติให้หน่วยทดลองของแต่ละกลุ่ม แต่เป็นการสุ่มหน่วยทดลองเป็นไปอย่างอิสระ โดยไม่มีข้อกำหนดใดๆในการสุ่ม ดังนั้นการสุ่มหน่วยทดลองของแต่ละกลุ่มจะเป็นอิสระต่อกัน

4.3.2.1 การตรวจสอบการแจกแจงของข้อมูล

1. ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ โดยมีสมมติฐานดังนี้

   H_0: ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ มีการแจกแจงแบบปกติ

   H_1: ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ ไม่ได้มีการแจกแจงแบบปกติ

2. ประสิทธิผลของการตรวจสอบซอฟต์แวร์ โดยมีสมมติฐานดังนี้

   H_0: ประสิทธิผลของการตรวจสอบซอฟต์แวร์ มีการแจกแจงแบบปกติ

   H_1: ประสิทธิผลของการตรวจสอบซอฟต์แวร์ ไม่ได้มีการแจกแจงแบบปกติ

การตรวจสอบการแจกแจงของข้อมูลเชิงปริมาณโดยใช้สถิติทดสอบเพื่อทดสอบว่า ข้อมูลมีการแจกแจงแบบปกติหรือไม่ สถิติที่ใช้ในการทดสอบที่ใช้มี Kolmogorov-Smirnov Test กับ การทดสอบชาพิโร-วิลค์ (Shapiro-Wilk Test) โดย Kolmogorov-Smirnov Test ใช้สำหรับทดสอบข้อมูลที่มีขนาดตัวอย่างมากกว่า 50 หน่วย และ การทดสอบชาพิโร-วิลค์

(Shapiro-Wilk Test) ใช้สำหรับทดสอบข้อมูลที่มีขนาดตัวอย่างไม่เกิน 50 หน่วย (กัลยา วานิชย์บัญชา, 2550) สำหรับงานวิจัยนี้มีขนาดหน่วยตัวอย่างในแต่ละกลุ่มไม่เกิน 50 หน่วย ดังนั้นจึงใช้เทคนิค การทดสอบชาพิโร-วิลค์ (Shapiro-Wilk Test) ในการตรวจสอบการแจกแจงของข้อมูล โดยจะยอมรับ H_0 ถ้าค่า Sig. (Significance) ของการทดสอบมากกว่าระดับนัยสำคัญที่กำหนด และจะปฏิเสธ H_0 และยอมรับ H_1 ถ้าค่า Sig. (Significance) ของการทดสอบน้อยกว่าระดับนัยสำคัญที่กำหนด (กัลยา วานิชย์บัญชา, 2550) โดยงานวิจัยนี้กำหนดระดับนัยสำคัญเท่ากับ 0.05

ตารางที่ 4-10 ตารางแสดงค่าสถิติทดสอบการแจกแจงปกติ (Test of Normality) ของข้อมูลทั้งสองตัวแปร ที่ได้จากการตรวจสอบซอฟต์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

ตัวแปร	Shapiro-Wilk		
	Statistic	df	Sig.
ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ (ข้อบกพร่อง / 1 นาที)	0.960	24	0.440
ประสิทธิผลของการตรวจสอบซอฟต์แวร์ (%)	0.945	24	0.214

จากตารางที่ 4-10 พบว่าค่า Sig. ของตัวแปรทุกตัวมีค่ามากกว่าระดับนัยสำคัญที่กำหนดไว้ แสดงว่ายอมรับ H_0 หมายความว่าตัวแปรทุกตัวมีการแจกแจงแบบปกติ และเมื่อเปรียบเทียบกับการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ที่ได้จากงานวิจัยของอุมาพร นิลเอวะ (2549) สามารถแสดงได้ดังตารางที่ 4-11

ตารางที่ 4-11 ตารางแสดงค่าสถิติทดสอบการแจกแจงปกติของข้อมูลทั้งสองตัวแปร ที่ได้จากการตรวจสอบซอฟต์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	Shapiro-Wilk		
		Statistic	df	Sig.
ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ (ข้อบกพร่อง / 1 นาที)	โอโออาร์ทีที่ปรับปรุงแล้ว	0.96	24	0.440
	โอโออาร์ทีเดิม	0.963	24	0.506
	ซีบีอาร์	0.921	24	0.062
ประสิทธิผลของการตรวจสอบซอฟต์แวร์ (%)	โอโออาร์ทีที่ปรับปรุงแล้ว	0.945	24	0.214
	โอโออาร์ทีเดิม	0.961	24	0.468
	ซีบีอาร์	0.936	24	0.131

จากตารางที่ 4-11 พบว่าค่า Sig. ของตัวแปรทุกตัวมีค่ามากกว่าระดับนัยสำคัญที่กำหนดไว้ แสดงว่ายอมรับ H_0 ซึ่งหมายความว่าตัวแปรทุกตัวมีการแจกแจงแบบปกติ แสดงให้เห็นว่าค่าประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีการแจกแจงแบบปกติ

4.3.2.2 การทดสอบความแปรปรวนของหน่วยทดลอง

1. การทดสอบค่าความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์ว่ามีค่าเท่ากันหรือไม่ สามารถกำหนดสมมติฐานได้ดังนี้

$$H_0: \sigma^2_{OORTNew} = \sigma^2_{OORTOld} = \sigma^2_{CBR}$$
$$H_1: \sigma^2_i \neq \sigma^2_j \text{ อย่างน้อย 1 คู่}$$

โดยที่ $\sigma^2_{OORTNew}$ = ความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

$\sigma^2_{OORTOld}$ = ความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีเดิม

σ^2_{CBR} = ความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคซีบีอาร์

i , j = ประเภทของเทคนิคการอ่านซอฟต์แวร์

การตรวจสอบสมมติฐานนี้เป็นการทดสอบความแปรปรวนของหน่วยทดลองที่แบ่งออกเป็น 3 กลุ่มที่ได้รับทรีทเมนต์ที่ต่างกัน ดังนั้นจึงต้องพิจารณาจากค่า Sig. จากตาราง การทดสอบความเป็นเอกพันธ์ของความแปรปรวน (Test of Homogeneity of Variance) ดังตารางที่ 4-12

ตารางที่ 4-12 ตารางแสดงค่าการทดสอบความเป็นเอกพันธ์ของความแปรปรวน (Test of Homogeneity of Variance) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์

Test of Homogeneity of Variances

ตัวแปร	Levene Statistic	df1	df2	Sig.
ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ (ข้อบกพร่อง / 1 นาที)	0.026	2	69	0.974

เนื่องจากในที่นี้เป็นการทดสอบ 2 ด้าน จึงเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig.มีค่าน้อยกว่าระดับนัยสำคัญจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05 และค่า Sig. คือ 0.974 จะเห็นว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงยอมรับ H_0 หมายความว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าความแปรปรวนเท่ากัน

2. การทดสอบค่าความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์ว่ามีค่าเท่ากันหรือไม่ สามารถกำหนดสมมติฐานได้ดังนี้

$$H_0: \sigma^2_{OORTNew} = \sigma^2_{OORTOld} = \sigma^2_{CBR}$$

$$H_1: \sigma^2_i \neq \sigma^2_j \text{ อย่างน้อย 1 คู่}$$

โดยที่ $\sigma^2_{OORTNew}$ = ความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

$\sigma^2_{OORTOld}$ = ความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีเดิม

σ^2_{CBR} = ความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคซีบีอาร์

i, j = ประเภทของเทคนิคการอ่านซอฟต์แวร์

การตรวจสอบสมมติฐานนี้เป็นการทดสอบความแปรปรวนของหน่วยทดลองที่แบ่งออกเป็น 3 กลุ่มที่ได้รับทรีทเมนต์ที่ต่างกัน ดังนี้จึงต้องพิจารณาจากค่า Sig. จากตารางการทดสอบความเป็นเอกพันธ์ของความแปรปรวน (Test of Homogeneity of Variance) ดังตารางที่ 4-13

ตารางที่ 4-13 ตารางแสดงค่า การทดสอบความเป็นเอกพันธ์ของความแปรปรวน (Test of Homogeneity of Variance) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์

Test of Homogeneity of Variances

ตัวแปร	Levene Statistic	df1	df2	Sig.
ประสิทธิผลของการตรวจสอบซอฟต์แวร์ (%)	0.182	2	69	0.834

เนื่องจากในที่นี้เป็นการทดสอบ 2 ด้าน จึงเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig.มีค่าน้อยกว่าระดับนัยสำคัญจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05 และค่า Sig. คือ 0.834 จะเห็นว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงยอมรับ H_0 หมายความว่า ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าความแปรปรวนเท่ากัน

4.3.2.3 การเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ผู้วิจัยต้องการเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอล โดยแบ่งการคำนวณออกเป็น 3 กลุ่มทดลอง คือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วในการตรวจสอบซอฟต์แวร์ (2) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีเดิมในการตรวจสอบซอฟต์แวร์ และ (3) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบซอฟต์แวร์ ดังนั้นผู้วิจัยจึงเลือกการวิเคราะห์ความแปรปรวน (Analysis of Variance (ANOVA)) เป็นเทคนิคในการวิเคราะห์เพื่อทดสอบสมมติฐาน ในการใช้การวิเคราะห์ความแปรปรวนต้องตรวจสอบเงื่อนไขของการวิเคราะห์ความแปรปรวนทั้ง 3 ประการ ดังนี้ (1) ค่าตัวแปรตามที่ได้จากหน่วยทดลองต้อง

เป็นการแจกแจงแบบปกติ (Normal Distribution) (2) ค่าความแปรปรวนของหน่วยทดลองคนละกลุ่มต้องเท่ากัน และ (3) การสุ่มหน่วยทดลองแต่ละกลุ่มต้องเป็นอิสระกัน (กัลยา วาณิชย์บัญชา, 2544)

จากการตรวจสอบในหัวข้อก่อนหน้านี้ สรุปได้ว่า ตัวแปรตามของหัวข้อนี้คือ ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ จากตารางที่ 4-11 แสดงว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีการแจกแจงแบบปกติ และจากตารางที่ 4-12 แสดงว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าความแปรปรวนเท่ากัน และการสุ่มหน่วยทดลองเป็นการสุ่มจากหน่วยทดลองที่คุณสมบัติเหมือนกันและไม่มีเงื่อนไขในการสุ่ม ดังนั้นการสุ่มหน่วยทดลองจึงเป็นอิสระต่อกัน แสดงว่าสามารถใช้การวิเคราะห์ความแปรปรวนในการทดสอบสมมติฐานได้

4.3.2.3.1 การทดสอบสมมติฐาน

การเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ กำหนดสมมติฐานดังนี้

$$H_0 : \mu_{OORTNew} = \mu_{OORTOld} = \mu_{CBR}$$
$$H_1 : \mu_i \neq \mu_j \text{ อย่างน้อย 1 คู่}$$

โดยที่ $\mu_{OORTNew}$ = ค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

$\mu_{OORTOld}$ = ค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีเดิม

μ_{CBR} = ค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคซีบีอาร์

i , j = ประเภทของเทคนิคการอ่านซอฟต์แวร์

เนื่องจากผลทดสอบความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์แล้ว โดยสรุปว่ามีค่าความแปรปรวนเท่ากัน ดังนั้นในการทดสอบสมมติฐานจะใช้ค่า Sig. ของสถิติ

ทดสอบเอฟ (F - Test) ในตารางการวิเคราะห์ความแปรปรวน (ANOVA) ดังตารางที่ 4-14 ในการทดสอบสมมติฐาน

ตารางที่ 4-14 ตารางแสดงค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ โดยใช้การวิเคราะห์ความแปรปรวน (ANOVA)

ANOVA

	Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Between Groups	0.016	2	0.008	10.608	0.000
Within Groups	0.053	69	0.001		
Total	0.070	71			

เนื่องจากในที่นี้เป็นการทดสอบ 2 ด้าน จึงเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig.มีค่าน้อยกว่าจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05 และค่า Sig. คือ 0.000 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 แสดงว่าค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ระหว่างการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์มีค่าแตกต่างกันอย่างน้อย 1 คู่

ถ้าต้องการทราบว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์คู่ใดบ้างที่มีค่าไม่เท่ากัน และประสิทธิภาพของการตรวจสอบด้วยเทคนิคใดมีค่ามากกว่ากันนั้น ต้องเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ด้วยวิธีการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) หรือการทดสอบความแตกต่างของค่าเฉลี่ยทีละคู่ งานวิจัยนี้ต้องการทราบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ดังนั้นจึงดำเนินการเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิม และเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคซีบีอาร์ โดยผลการเปรียบเทียบแสดงดังตารางที่ 4-15 ซึ่งเป็นตารางแสดงการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) ของประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตารางที่ 4-15 ตารางแสดงผลการเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

Multiple Comparisons

เทคนิคการตรวจสอบซอฟต์แวร์ตั้งต้น	เทคนิคการตรวจสอบซอฟต์แวร์เปรียบเทียบ	Mean Difference	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
โอโออาร์ทีที่ปรับปรุงแล้ว	โอโออาร์ทีเดิม	.016208	0.008	0.047	0.000	0.032
	ซีบีอาร์	-.020667	0.008	0.012	-0.037	-0.005

จากตารางที่ 4-15 ต้องเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig. มีค่าน้อยกว่าจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05

1. การเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม มีค่า Sig. คือ 0.047 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 หมายความว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมไม่เท่ากัน ดังนั้นจึงต้องพิจารณาที่ความแตกต่างของค่าเฉลี่ย (Mean Difference) ซึ่งแสดงการเปรียบเทียบค่าเฉลี่ยของทั้งสองตัวแปร คือ ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม มีค่าเท่ากับ .016208 ดูเหมือนว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าดีกว่าประสิทธิภาพของเทคนิคโอโออาร์ทีเดิม และเมื่อพิจารณาค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์จากตารางที่ 4-9 พบว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าดีกว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีเดิม

2. การเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคซีบีอาร์ มีค่า Sig. คือ 0.012 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 หมายความว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคซีบีอาร์มีค่าไม่เท่ากัน ดังนั้นจึงต้องพิจารณาที่ความแตกต่างของค่าเฉลี่ย (Mean Difference) ซึ่งแสดงการเปรียบเทียบค่าเฉลี่ย

ของทั้งสองตัวแปร คือ ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคซีบีอาร์ มีค่าเท่ากับ -.020667 ดูเหมือนว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าต่ำกว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคซีบีอาร์ และเมื่อพิจารณาค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์จากตารางที่ 4-9 พบว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ไม่ได้มีค่าดีกว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคซีบีอาร์

4.3.2.4 การเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ผู้วิจัยต้องการเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอล โดยแบ่งการคำนวณออกเป็น 3 กลุ่มทดลอง คือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วในการตรวจสอบซอฟต์แวร์ (2) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีเดิมในการตรวจสอบซอฟต์แวร์ และ (3) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบซอฟต์แวร์ ดังนั้นผู้วิจัยจึงเลือกการวิเคราะห์ความแปรปรวน (Analysis of Variance (ANOVA)) เป็นเทคนิคในการวิเคราะห์เพื่อทดสอบสมมติฐาน ในการใช้การวิเคราะห์ความแปรปรวนต้องตรวจสอบเงื่อนไขของการวิเคราะห์ความแปรปรวนทั้ง 3 ประการ ดังนี้ (1) ค่าตัวแปรตามที่ได้จากหน่วยทดลองต้องเป็นการแจกแจงแบบปกติ (Normal Distribution) (2) ค่าความแปรปรวนของหน่วยทดลองคนละกลุ่มต้องเท่ากัน และ (3) การสุ่มหน่วยทดลองแต่ละกลุ่มต้องเป็นอิสระกัน (กัลยา วาณิชย์บัญชา, 2544)

จากตรวจสอบในหัวข้อก่อนหน้านี้ สรุปได้ว่า ตัวแปรตามของหัวข้อนี้คือประสิทธิผลของการตรวจสอบซอฟต์แวร์ จากตารางที่ 4-11 แสดงว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีการแจกแจงแบบปกติ และจากตารางที่ 4-13 แสดงว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าความแปรปรวนเท่ากัน และการสุ่มหน่วยทดลองเป็นการสุ่มจากหน่วยทดลองที่คุณสมบัติเหมือนกันและไม่มีเงื่อนไขในการสุ่ม ดังนั้นการสุ่มหน่วยทดลองจึงเป็นอิสระต่อกัน แสดงว่าสามารถใช้การวิเคราะห์ความแปรปรวนในการทดสอบสมมติฐานได้

4.3.2.4.1 การทดสอบสมมติฐาน

การเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ กำหนดสมมติฐานดังนี้

$$H_0 : \mu_{OORTNew} = \mu_{OORTOld} = \mu_{CBR}$$
$$H_1 : \mu_i \neq \mu_j \text{ อย่างน้อย 1 คู่}$$

โดยที่ $\mu_{OORTNew}$ = ค่าเฉลี่ยของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

$\mu_{OORTOld}$ = ค่าเฉลี่ยของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีเดิม

μ_{CBR} = ค่าเฉลี่ยของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคซีบีอาร์

i , j = ประเภทของเทคนิคการอ่านซอฟต์แวร์

เนื่องจากทดสอบความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์แล้ว โดยสรุปว่ามีค่าความแปรปรวนเท่ากัน ดังนั้นในการทดสอบสมมติฐานจะใช้ค่า Sig. ของสถิติทดสอบเอฟ (F - Test) ในตารางการวิเคราะห์ความแปรปรวน (ANOVA) ดังตารางที่ 4-16 ในการทดสอบสมมติฐาน

ตารางที่ 4-16 ตารางแสดงค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ โดยใช้การวิเคราะห์ความแปรปรวน (ANOVA)

ANOVA

	Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Between Groups	1,881.250	2	940.625	6.464	0.003
Within Groups	10,040.625	69	145.516		
Total	11,921.875	71			

เนื่องจากในที่นี้เป็นการทดสอบ 2 ด้าน จึงเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig. มีค่าน้อยกว่าระดับนัยสำคัญจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05 และค่า Sig. คือ 0.003 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 แสดงว่าค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ระหว่างการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์มีค่าแตกต่างกันอย่างน้อย 1 คู่

ถ้าต้องการทราบว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์คู่ใดบ้างที่มีค่าไม่เท่ากัน และประสิทธิผลของการตรวจสอบด้วยเทคนิคใดมีค่ามากกว่ากันนั้น ต้องเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยวิธีการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) หรือการทดสอบความแตกต่างของค่าเฉลี่ยทีละคู่ งานวิจัยนี้ต้องการทราบว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ดังนั้นจึงดำเนินการเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิม และเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคซีบีอาร์ โดยผลการเปรียบเทียบ แสดงดังตารางที่ 4-17 ซึ่งเป็นตารางแสดงการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) ของประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตารางที่ 4-17 ตารางแสดงผลการเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

Multiple Comparisons

เทคนิคการตรวจสอบซอฟต์แวร์ตั้งต้น	เทคนิคการตรวจสอบซอฟต์แวร์เปรียบเทียบ	Mean Difference	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
โอโออาร์ทีที่ปรับปรุงแล้ว	โอโออาร์ทีเดิม	5.62500	3.48229	0.111	-1.3220	12.5720
	ซีบีอาร์	-6.87500	3.48229	0.052	-13.8220	0.0720
โอโออาร์ทีเดิม	ซีบีอาร์	-12.50000	3.48229	0.001	-19.4470	-5.5530
	โอโออาร์ทีที่ปรับปรุงแล้ว	-5.62500	3.48229	0.111	-12.5720	1.3220

จากตารางที่ 4-17 ต้องเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig. มีค่าน้อยกว่าระดับนัยสำคัญจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05

1. การเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม มีค่า Sig. คือ 0.111 จะเห็นว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงไม่สามารถปฏิเสธ H_0 หมายความว่า ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าไม่แตกต่างกัน

2. การเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคซีบีอาร์ มีค่า Sig. คือ 0.052 จะเห็นว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงไม่สามารถปฏิเสธ H_0 หมายความว่า ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคซีบีอาร์มีค่าไม่แตกต่างกัน

จากการทดสอบสมมติฐานพบว่า ค่าเฉลี่ยของประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่าง เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีอย่างน้อย 1 คู่ที่แตกต่างกัน โดยจากตารางที่ 4-17 ถ้าเปรียบเทียบระหว่างเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ พบว่าค่า Sig. ที่ได้ คือ 0.001 จะเห็น

ว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 หมายความว่า ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าแตกต่างกัน

4.4 การวิเคราะห์ข้อมูลเพิ่มเติม (Exploration)

หลังจากผู้วิจัยทดสอบสมมติฐานเรียบร้อยแล้ว จะเห็นว่าผลที่ได้ไม่ตรงตามที่ผู้วิจัยคาดไว้ กล่าวคือประสิทธิภาพของการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีค่าสูงกว่าการใช้เทคนิคโอโออาร์ทีเดิม แต่มีค่าประสิทธิภาพของการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ยังคงต่ำว่าการใช้เทคนิคซีบีอาร์ และการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิผลไม่แตกต่างจากการใช้เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์

เมื่อพิจารณาข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ของงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่งานวิจัยนี้ใช้ในการตรวจสอบ พบว่ามีข้อบกพร่องจำนวน 20 ข้อ และในจำนวนนี้มีข้อบกพร่อง 4 ข้อที่เกิดในเอกสารอธิบายคลาสและสามารถตรวจพบได้เฉพาะจากขั้นตอน (Reading) ที่ 4 ของเทคนิคโอโออาร์ทีที่เป็นการเปรียบเทียบระหว่างแผนภาพคลาสกับเอกสารอธิบายคลาส แต่งานวิจัยของ Conradi (2003) ที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ ได้กล่าวว่า เอกสารอธิบายคลาสไม่ได้เป็นเอกสารสำคัญที่แสดงถึงการออกแบบซอฟต์แวร์ที่ถูกนำไปใช้ในขั้นตอนต่อไปของการพัฒนาซอฟต์แวร์ แต่เป็นเอกสารที่ใช้อธิบายแผนภาพคลาสเท่านั้น ดังนั้นข้อบกพร่องที่เกิดในส่วนของเอกสารอธิบายคลาสจึงไม่ใช่ข้อบกพร่องที่มีความสำคัญที่ต้องได้รับการแก้ไขโดยด่วน และยังทำให้เพิ่มต้นทุนในการแก้ไข โดยที่ไม่ได้ใช้ประโยชน์จากเอกสารเท่าที่ควร

ดังนั้นข้อบกพร่องทั้ง 4 ข้อที่เกิดในเอกสารอธิบายคลาสและสามารถตรวจพบได้จากเฉพาะการเปรียบเทียบระหว่างแผนภาพคลาสและเอกสารอธิบายคลาสเท่านั้น จึงไม่ควรเป็นข้อบกพร่องที่มีความสำคัญมากที่ต้องได้รับการแก้ไขโดยด่วน เพราะฉะนั้นถ้าลดการพิจารณาจำนวนข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ลง โดยตัดข้อบกพร่องทั้ง 4 ข้อที่เกิดในเอกสารอธิบายคลาสและสามารถค้นหาได้เฉพาะจากขั้นตอน (Reading) ที่ 4 ของเทคนิคโอโออาร์ทีผลการทดลองที่ได้อาจจะแตกต่างจากเดิมที่พิจารณาข้อบกพร่องครบทั้ง 20 ข้อ

ดังนั้นในส่วนของการวิเคราะห์ข้อมูลเพิ่มเติมในส่วนนี้ของงานวิจัย จึงลดการพิจารณาจำนวนข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์ลง เหลือ 16 ข้อ โดยไม่คำนึงถึงข้อบกพร่องของเอกสารอธิบายคลาสที่ได้จากการเปรียบเทียบเอกสารอธิบายคลาสกับแผนภาพคลาส แล้วจึงเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ดังนี้

4.4.1 การวิเคราะห์ข้อมูลในลักษณะสถิติเชิงพรรณนา (Descriptive Statistics)

จากผลการทดลองของ อุมาพร นิลเอวะ (2549) ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ และผลการทดลองของงานวิจัยนี้ที่ตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว จะพิจารณาจำนวนข้อบกพร่องที่หน่วยทดลองตรวจสอบได้และบันทึกผลในเอกสารบันทึกรายการข้อบกพร่องในขั้นตอนการจัดเตรียม ที่กำหนดระยะเวลาที่ใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ไว้ไม่เกิน 90 นาที โดยจำนวนข้อบกพร่องที่พิจารณามีจำนวนทั้งหมด 16 ข้อ เนื่องจากไม่พิจารณาข้อบกพร่องในเอกสารอธิบายคลาสที่สามารถตรวจพบได้จากการเปรียบเทียบระหว่างแผนภาพคลาสและเอกสารอธิบายคลาสเท่านั้น จะได้ค่าสถิติของจำนวนข้อบกพร่องดังตารางที่ 4-18 โดยระยะเวลาที่ใช้ในการตรวจสอบจะเท่าเดิม

ตารางที่ 4-18 ตารางแสดงค่าสถิติของจำนวนข้อบกพร่อง จากการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ที และเทคนิคซีบีอาร์

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด/ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
จำนวนข้อบกพร่อง	โอโออาร์ทีที่ปรับปรุงแล้ว	24	3 / 11	6.625	2.242
	โอโออาร์ทีเดิม	24	1 / 7	4.080	1.558
	ซีบีอาร์	24	2 / 11	5.210	1.933

ตารางที่ 4-18 แสดงค่าสถิติของจำนวนข้อบกพร่องจากการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ที และเทคนิคซีบีอาร์ โดยจำนวนข้อบกพร่องที่พิจารณามีจำนวนทั้งหมด 16 ข้อ เนื่องจากไม่พิจารณาข้อบกพร่องในเอกสารอธิบายคลาสที่สามารถตรวจพบได้จากการเปรียบเทียบระหว่างแผนภาพคลาสและเอกสารอธิบายคลาสเท่านั้น โดยค่าที่ได้ คือ (1) ค่าเฉลี่ยของข้อบกพร่อง ซึ่งเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีค่าสูงที่สุด รองลงมาเป็นเทคนิคซีบีอาร์ และเทคนิคโอโออาร์ทีเดิมมีค่าต่ำที่สุด ซึ่งค่าที่ได้แตกต่างเดิม ที่จากการพิจารณาข้อบกพร่องจำนวน 20 ข้อ ที่รวมข้อบกพร่องของเอกสารอธิบายคลาสที่สามารถค้นหาได้เฉพาะจากการเปรียบเทียบแผนภาพคลาสและเอกสารอธิบายคลาสด้วย โดยถ้าพิจารณาข้อบกพร่องจำนวน 20 ข้อบกพร่อง ค่าเฉลี่ยของข้อบกพร่องในเอกสารการออกแบบซอฟต์แวร์ที่ตรวจสอบด้วยเทคนิคซีบีอาร์มีค่าสูงที่สุด รองลงมาเป็นเทคนิคโอโออาร์ทีที่

ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าต่ำที่สุด แล้วนำผลการวิเคราะห์ข้างต้นไปหาค่าประสิทธิภาพและประสิทธิผล จะได้ผลดังตารางที่ 4-19

ตารางที่ 4-19 ตารางแสดงค่าประสิทธิภาพและประสิทธิผลของผลการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด/ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	โอโออาร์ทีที่ปรับปรุงแล้ว	24	0.033 / 0.122	0.078	0.0256
	โอโออาร์ทีเดิม	24	0.01 / 0.08	0.046	0.017
	ซีบีอาร์	24	0.02 / 0.13	0.064	0.023
ประสิทธิผล (%)	โอโออาร์ทีที่ปรับปรุงแล้ว	24	15 / 55	33.13	11.211
	โอโออาร์ทีเดิม	24	6.25 / 43.75	25.52	9.738
	ซีบีอาร์	24	12.50 / 68.75	32.55	12.082

ตารางที่ 4-19 แสดงค่าสถิติของค่าประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ โดยจำนวนข้อบกพร่องที่พิจารณามีจำนวนทั้งหมด 16 ข้อ เนื่องจากไม่พิจารณาข้อบกพร่องในเอกสารอธิบายคลาสที่สามารถตรวจพบได้จากการเปรียบเทียบระหว่างแผนภาพคลาสและเอกสารอธิบายคลาสเท่านั้น โดยค่าที่ได้มี 2 ค่า คือ (1) ค่าเฉลี่ยของประสิทธิภาพ ซึ่งเทคนิคเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีค่าสูงที่สุด รองลงมาเป็นเทคนิคซีบีอาร์ และเทคนิคโอโออาร์ทีเดิมมีค่าต่ำที่สุด ซึ่งแตกต่างจากเดิมที่พิจารณาข้อบกพร่องจำนวน 20 ข้อ โดยถ้าพิจารณาข้อบกพร่องทั้ง 20 ข้อ ผลของค่าเฉลี่ยประสิทธิภาพที่ได้ คือ เทคนิคซีบีอาร์มีค่าสูงที่สุด รองลงมาเป็นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าต่ำที่สุด และ (2) ค่าเฉลี่ยของประสิทธิผล ซึ่งเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีค่าสูงที่สุด รองลงมาเป็นเทคนิคซีบีอาร์ และเทคนิคโอโออาร์ทีเดิมมีค่าต่ำสุด ซึ่งแตกต่างจากเดิมที่พิจารณาข้อบกพร่องจำนวน 20 ข้อ โดยถ้าพิจารณาข้อบกพร่องทั้ง 20 ข้อ ผลของค่าเฉลี่ยของประสิทธิผลที่ตรวจสอบได้ คือ เทคนิคซีบีอาร์มีค่าสูงสุด รองลงมาเป็นเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าต่ำที่สุด

4.4.2 การทดสอบสมมติฐาน

ผู้วิจัยเลือกการวิเคราะห์ความแปรปรวน (Analysis of Variance (ANOVA)) เป็นเทคนิคการวิเคราะห์เพื่อทดสอบสมมติฐาน เนื่องจากสมมติฐานที่ตั้งเป็นการเปรียบเทียบหน่วยทดลองที่แบ่งออกเป็น 3 กลุ่มที่ได้รับทรีทเมนต์ที่ต่างกัน โดยการทดลองในส่วนนี้ของงานวิจัยเป็นการทดสอบตัวแปรต้นเพียงตัวเดียว คือ เทคนิคการอ่านซอฟต์แวร์ ที่มีค่าที่ใช้ทดสอบ 3 ค่า คือ เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และ เทคนิคซีบีอาร์ ดังนั้นในการวิเคราะห์ความแปรปรวน จึงเลือก การวิเคราะห์ความแปรปรวนแบบทางเดียว (One-way Analysis of Variance) ที่ใช้สำหรับการทดสอบความแตกต่างระหว่างค่าเฉลี่ยจากหน่วยทดลองตั้งแต่ 3 กลุ่มขึ้นไป เพื่อตรวจสอบว่าตัวแปรต้น 1 ตัว ที่มีค่าที่สามารถแบ่งกลุ่มได้ 3 กลุ่มขึ้นไปส่งผลแตกต่างกันหรือไม่ (กัลยา วาณิชย์บัญชา, 2544)

4.4.2.1 การตรวจสอบการแจกแจงของข้อมูล

1. ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ โดยมีสมมติฐานดังนี้

H_0: ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ มีการแจกแจงแบบปกติ

H_1: ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ ไม่ได้มีการแจกแจงแบบปกติ

2. ประสิทธิผลของการตรวจสอบซอฟต์แวร์ โดยมีสมมติฐานดังนี้

H_0: ประสิทธิผลของการตรวจสอบซอฟต์แวร์ มีการแจกแจงแบบปกติ

H_1: ประสิทธิผลของการตรวจสอบซอฟต์แวร์ ไม่ได้มีการแจกแจงแบบปกติ

การตรวจสอบการแจกแจงของข้อมูลเชิงปริมาณโดยใช้สถิติทดสอบเพื่อทดสอบว่า ข้อมูลมีการแจกแจงแบบปกติหรือไม่ สถิติที่ใช้ในการทดสอบสำหรับงานวิจัยนี้ที่มีขนาดหน่วยตัวอย่างในแต่ละกลุ่มไม่เกิน 50 หน่วย ดังนั้นจึงใช้เทคนิค การทดสอบชาพิโร-วิลค์ (Shapiro-Wilk Test) ในการตรวจสอบการแจกแจงของข้อมูล โดยจะยอมรับ H_0 ถ้าค่า Sig. (Significance) ของการทดสอบมากกว่าระดับนัยสำคัญที่กำหนด ปฏิเสธ H_0 ถ้าค่า Sig. (Significance) ของการทดสอบน้อยกว่าระดับนัยสำคัญที่กำหนด โดยงานวิจัยนี้กำหนดระดับนัยสำคัญเท่ากับ 0.05

ตารางที่ 4-20 ตารางแสดงค่าสถิติทดสอบการแจกแจงปกติของข้อมูลทั้งสองตัวแปร ที่ได้จากการตรวจสอบซอฟต์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	Shapiro-Wilk		
		Statistic	df	Sig.
ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ (ข้อบกพร่อง / 1 นาที)	โอโออาร์ทีที่ปรับปรุงแล้ว	0.960	24	0.440
	โอโออาร์ทีเดิม	0.953	24	0.315
	ซีบีอาร์	0.955	24	0.341
ประสิทธิผลของการตรวจสอบซอฟต์แวร์ (%)	โอโออาร์ทีที่ปรับปรุงแล้ว	0.945	24	0.214
	โอโออาร์ทีเดิม	0.950	24	0.268
	ซีบีอาร์	0.925	24	0.074

จากตารางที่ 4-20 พบว่าค่า Sig. ของตัวแปรทุกตัวมีค่ามากกว่าระดับนัยสำคัญที่กำหนดไว้ แสดงว่ายอมรับ H_0 หมายความว่าตัวแปรทุกตัวมีการแจกแจงแบบปกติ แสดงให้เห็นว่าค่าประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีการแจกแจงแบบปกติ

4.4.2.2 การทดสอบความแปรปรวนของหน่วยทดลอง

1. การทดสอบค่าความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์ว่ามีค่าเท่ากันหรือไม่ สามารถกำหนดสมมติฐานได้ดังนี้

$$H_0: \sigma^2_{OORTNew} = \sigma^2_{OORTOld} = \sigma^2_{CBR}$$

$$H_1: \sigma^2_i \neq \sigma^2_j \text{ อย่างน้อย 1 คู่}$$

โดยที่ $\sigma^2_{OORTNew}$ = ความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

$\sigma^2_{OORTOld}$ = ความแปรปรวนของประะสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีเดิม

σ^2_{CBR} = ความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคซีบีอาร์

i, j = ประเภทของเทคนิคการอ่านซอฟต์แวร์

การตรวจสอบสมมติฐานนี้เป็นการทดสอบความแปรปรวนของหน่วยทดลองที่แบ่งออกเป็น 3 กลุ่มที่ได้รับทรีทเมนต์ที่ต่างกัน ดังนั้นจึงต้องพิจารณาจากค่า Sig. จากตาราง การทดสอบความเป็นเอกพันธ์ของความแปรปรวน (Test of Homogeneity of Variance) ดังตารางที่ 4-21

ตารางที่ 4-21 ตารางแสดงค่า การทดสอบความเป็นเอกพันธ์ของความแปรปรวน (Test of Homogeneity of Variance) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์

Test of Homogeneity of Variances

ตัวแปร	Levene Statistic	df1	df2	Sig.
ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ (ข้อบกพร่อง / 1 นาที)	1.588	2	69	0.212

เนื่องจากในที่นี้เป็นการทดสอบ 2 ด้าน จึงเปรียบเทียบ Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig.มีค่าน้อยกว่าจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05 และค่า Sig. คือ 0.212 จะเห็นว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงยอมรับ H_0 หมายความว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าความแปรปรวนเท่ากัน

2. การทดสอบค่าความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์ว่ามีค่าเท่ากันหรือไม่ สามารถกำหนดสมมติฐานได้ดังนี้

$$H_0: \sigma^2_{OORTNew} = \sigma^2_{OORTOld} = \sigma^2_{CBR}$$
$$H_1: \sigma^2_i \neq \sigma^2_j \text{ อย่างน้อย 1 คู่}$$

โดยที่ $\sigma^2_{OORTNew}$ = ความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

$\sigma^2_{OORTOld}$ = ความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีเดิม

σ^2_{CBR} = ความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคซีบีอาร์

i, j = ประเภทของเทคนิคการอ่านซอฟต์แวร์

การตรวจสอบสมมติฐานนี้เป็นการทดสอบความแปรปรวนของหน่วยทดลองที่แบ่งออกเป็น 3 กลุ่มที่ได้รับทรีทเมนต์ที่ต่างกัน ดังนี้จึงต้องพิจารณาจากค่า Sig. จากตารางการทดสอบความเป็นเอกพันธ์ของความแปรปรวน (Test of Homogeneity of Variance) ดังตารางที่ 4-22

ตารางที่ 4-22 ตารางแสดงค่า การทดสอบความเป็นเอกพันธ์ของความแปรปรวน (Test of Homogeneity of Variance) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์

Test of Homogeneity of Variances

ตัวแปร	Levene Statistic	df1	df2	Sig.
ประสิทธิผลของการตรวจสอบซอฟต์แวร์ (%)	0.325	2	69	0.724

เนื่องจากในที่นี้เป็นการทดสอบ 2 ด้าน จึงเปรียบเทียบ Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig.มีค่าน้อยกว่าจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05 และค่า Sig. คือ 0.724 จะเห็นว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงยอมรับ H_0 หมายความว่า ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าความแปรปรวนเท่ากัน

4.4.2.3 การเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ผู้วิจัยต้องการเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอล โดยแบ่งการคำนวณออกเป็น 3 กลุ่มทดลอง คือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วในการตรวจสอบซอฟต์แวร์ (2) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีเดิมในการตรวจสอบซอฟต์แวร์ และ (3) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบซอฟต์แวร์

ดังนั้นผู้วิจัยจึงเลือกการวิเคราะห์ความแปรปรวน (Analysis of Variance (ANOVA)) เป็นเทคนิคในการวิเคราะห์เพื่อทดสอบสมมติฐาน ซึ่งจากตรวจสอบในหัวข้อก่อนหน้านี้ สรุปได้ว่า ตัวแปรตามของหัวข้อนี้คือประสิทธิภาพของการตรวจสอบซอฟต์แวร์ จากตารางที่ 4-20 แสดงว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีการแจกแจงแบบปกติ และจากตารางที่ 4-21 แสดงว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าความแปรปรวนเท่ากัน และการสุ่มหน่วยทดลองเป็นการสุ่มจากหน่วยทดลองที่คุณสมบัติเหมือนกันและไม่มีเงื่อนไขในการสุ่ม ดังนั้นการสุ่มหน่วยทดลองจึงเป็นอิสระต่อกัน แสดงว่าสามารถใช้การวิเคราะห์ความแปรปรวนในการทดสอบสมมติฐานได้

4.4.2.3.1 การทดสอบสมมติฐาน

การเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ กำหนดสมมติฐานดังนี้

$$H_0 : \mu_{OORTNew} = \mu_{OORTOld} = \mu_{CBR}$$
$$H_1 : \mu_i \neq \mu_j \text{ อย่างน้อย 1 คู่}$$

โดยที่ $\mu_{OORTNew}$ = ค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

$\mu_{OORTOld}$ = ค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีเดิม

μ_{CBR} = ค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์โดยเทคนิคซีบีอาร์

i, j = ประเภทของเทคนิคการอ่านซอฟต์แวร์

เนื่องจากผลทดสอบความแปรปรวนของประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์แล้ว โดยสรุปว่ามีค่าความแปรปรวนเท่ากัน ดังนั้นในการทดสอบสมมติฐานจะใช้ค่า Sig. ของสถิติทดสอบเอฟ (F - Test) ในตารางการวิเคราะห์ความแปรปรวน (ANOVA) ดังตารางที่ 4-23 ในการทดสอบสมมติฐาน

ตารางที่ 4-23 ตารางแสดงค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ โดยใช้การวิเคราะห์ความแปรปรวน (ANOVA)

ANOVA

	Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Between Groups	0.013	2	0.006	12.702	0.000
Within Groups	0.034	69	0.000		
Total	0.047	71			

เนื่องจากในที่นี้เป็นการทดสอบ 2 ด้าน จึงเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig.มีค่าน้อยกว่าจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05 และค่า Sig. คือ 0.000 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 แสดงว่าค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์มีค่าแตกต่างกัน อย่างน้อย 1 คู่

ถ้าต้องการทราบว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์คู่ใดบ้างที่มีค่าไม่เท่ากัน และประสิทธิภาพของการตรวจสอบด้วยเทคนิคใดมีค่ามากกว่ากันนั้น ต้องเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ด้วยวิธีการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) หรือการทดสอบความแตกต่างของค่าเฉลี่ยทีละคู่ งานวิจัยนี้ต้องการทราบว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเมื่อเทียบกับเทคนิคโอโออ์ทีเดิมและเทคนิคซีบีอาร์ ดังนั้นจึงดำเนินการเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิม และเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคซีบีอาร์ โดยผลการเปรียบเทียบแสดงดังตารางที่ 4-24 ซึ่งเป็นตารางแสดงการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) ของประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตารางที่ 4-24 ตารางแสดงผลการเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

Multiple Comparisons

เทคนิคการตรวจสอบซอฟต์แวร์ตั้งต้น	เทคนิคการตรวจสอบซอฟต์แวร์เปรียบเทียบ	Mean Difference	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
โอโออาร์ทีที่ปรับปรุงแล้ว	โอโออาร์ทีเดิม	.03221	0.00641	0.000	0.0194	0.0450
	ซีบีอาร์	.01371	0.00641	0.036	0.0009	0.0265

จากตารางที่ 4-24 ต้องเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig. มีค่าน้อยกว่าจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05

1. การเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม มีค่า Sig. คือ 0.000 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 หมายความว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมไม่เท่ากัน ดังนั้นจึงต้องพิจารณาที่ความแตกต่างของค่าเฉลี่ย (Mean Difference) ซึ่งแสดงการเปรียบเทียบค่าเฉลี่ยของทั้งสองตัวแปร คือ ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม มีค่าเท่ากับ 0.03221ดูเหมือนว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าสูงกว่าประสิทธิภาพของเทคนิคโอโออาร์ทีเดิม และเมื่อพิจารณาค่าเฉลี่ยของประสิทธิภาพของการตรวจสอบซอฟต์แวร์จากตารางที่ 4-19 พบว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าดีกว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีเดิม

2. การเปรียบเทียบประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคซีบีอาร์ มีค่า Sig. คือ 0.036 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 หมายความว่า ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคซีบีอาร์มีค่าไม่เท่ากัน ดังนั้นจึงต้องพิจารณาที่ความแตกต่างของค่าเฉลี่ย (Mean Difference) ซึ่งแสดงการเปรียบเทียบค่าเฉลี่ยของทั้งสองตัวแปร คือ ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่

ปรับปรุงแล้ว เทียบกับเทคนิคซีบีอาร์ มีค่าเท่ากับ 0.01371 ดูเหมือนว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าสูงกว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคซีบีอาร์ และเมื่อพิจารณาค่าเฉลี่ยจากตารางที่ 4-19 พบว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าดีกว่าประสิทธิภาพของการตรวจสอบซอฟต์แวร์ของเทคนิคซีบีอาร์

4.4.2.4 การเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ผู้วิจัยต้องการเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการอธิบายความต้องการและการออกแบบซอฟต์แวร์ ที่ออกแบบโดยแผนภาพยูเอ็มแอล โดยแบ่งการคำนวณออกเป็น 3 กลุ่มทดลอง คือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วในการตรวจสอบซอฟต์แวร์ (2) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีเดิมในการตรวจสอบซอฟต์แวร์ และ (3) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบซอฟต์แวร์ ดังนั้นผู้วิจัยจึงเลือกการวิเคราะห์ความแปรปรวน (Analysis of Variance (ANOVA)) เป็นเทคนิคในการวิเคราะห์เพื่อทดสอบสมมติฐาน ซึ่งจากตรวจสอบในหัวข้อก่อนหน้านี้ สรุปได้ว่า ตัวแปรตามของหัวข้อนี้คือประสิทธิผลของการตรวจสอบซอฟต์แวร์ จากตารางที่ 4-20 แสดงว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีการแจกแจงแบบปกติ และจากตารางที่ 4-22 แสดงว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์มีค่าความแปรปรวนเท่ากัน และการสุ่มหน่วยทดลองเป็นการสุ่มจากหน่วยทดลองที่คุณสมบัติเหมือนกันและไม่มีเงื่อนไขในการสุ่ม ดังนั้นการสุ่มหน่วยทดลองจึงเป็นอิสระต่อกัน แสดงว่าสามารถใช้การวิเคราะห์ความแปรปรวนในการทดสอบสมมติฐานได้

4.4.2.4.1 การทดสอบสมมติฐาน

การเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ กำหนดสมมติฐานดังนี้

$$H_0 : \mu_{OORTNew} = \mu_{OORTOld} = \mu_{CBR}$$
$$H_1 : \mu_i \neq \mu_j \text{ อย่างน้อย 1 คู่}$$

โดยที่ $\mu_{OORTNew}$ = ค่าเฉลี่ยของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

$\mu_{OORTOld}$ = ค่าเฉลี่ยของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคโอโออาร์ทีเดิม

μ_{CBR} = ค่าเฉลี่ยของประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยเทคนิคซีบีอาร์

i, j = ประเภทของเทคนิคการอ่านซอฟต์แวร์

เนื่องจากผลทดสอบความแปรปรวนของประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์แล้ว โดยสรุปว่ามีค่าความแปรปรวนเท่ากัน ดังนั้นในการทดสอบสมมติฐานจะใช้ค่า Sig. ของสถิติทดสอบเอฟ (F - Test) ในตารางการวิเคราะห์ความแปรปรวน (ANOVA) ดังตารางที่ 4-25 ในการทดสอบสมมติฐาน

ตารางที่ 4-25 ตารางแสดงค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ โดยใช้การวิเคราะห์ความแปรปรวน (ANOVA)

ANOVA

	Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Between Groups	860.720	2	430.360	3.523	0.035
Within Groups	8,429.362	69	122.165		
Total	9,290.082	71			

เนื่องจากในที่นี้เป็นการทดสอบ 2 ด้าน จึงเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig.มีค่าน้อยกว่าจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05 และค่า Sig. คือ 0.035 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 แสดงว่าค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ระหว่างการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์มีค่าแตกต่างกัน อย่างน้อย 1 คู่

ถ้าต้องการทราบว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์คู่ใดบ้างที่มีค่าไม่เท่ากัน และประสิทธิผลของการตรวจสอบด้วยเทคนิคใดมีค่ามากกว่ากันนั้น ต้องเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยวิธีการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) หรือการทดสอบความแตกต่างของค่าเฉลี่ยทีละคู่ งานวิจัยนี้ต้องการทราบว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ดังนั้นจึงดำเนินการเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิม และเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคซีบีอาร์ โดยผลการเปรียบเทียบ แสดงดังตารางที่ 4-26 ซึ่งเป็นตารางแสดงการเปรียบเทียบเชิงซ้อน (Multiple Comparisons) ของประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

ตารางที่ 4-26 ตารางแสดงผลการเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์

Multiple Comparisons

เทคนิคการตรวจสอบซอฟต์แวร์ตั้งต้น	เทคนิคการตรวจสอบซอฟต์แวร์เปรียบเทียบ	Mean Difference	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
โอโออาร์ทีที่ปรับปรุงแล้ว	โอโออาร์ทีเดิม	7.60417	3.19067	0.020	1.2389	13.9694
	ซีบีอาร์	0.57292	3.19067	0.858	-5.7923	6.9381

จากตารางที่ 4-26 ต้องเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญที่กำหนด ถ้าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญจะยอมรับ H_0 แต่ถ้าค่า Sig. มีค่าน้อยกว่าจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2550) โดยในที่นี้กำหนดค่าระดับนัยสำคัญเท่ากับ 0.05

1. การเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ทีเดิม มีค่า Sig. คือ 0.020 จะเห็นว่าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงปฏิเสธ H_0 หมายความว่า ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคโอโออาร์ทีเดิมมีค่าไม่

เท่ากัน ดังนั้นจึงต้องพิจารณาที่ความแตกต่างของค่าเฉลี่ย (Mean Difference) ซึ่งแสดงการเปรียบเทียบค่าเฉลี่ยของทั้งสองตัวแปร คือ ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคโอโออาร์ที มีค่าเท่ากับ 7.60417 ดูเหมือนว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าสูงกว่าประสิทธิผลของเทคนิคโอโออาร์ทีเดิม และเมื่อพิจารณาค่าเฉลี่ยจากตารางที่ 4-19 พบว่า ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว มีค่าดีกว่าประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีเดิม

2. การเปรียบเทียบประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทียบกับเทคนิคซีบีอาร์ มีค่า Sig. คือ 0.858 จะเห็นว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงไม่สามารถปฏิเสธ H_0 หมายความว่า ประสิทธิผลของการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว และเทคนิคซีบีอาร์มีค่าไม่แตกต่างกัน

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 บทนำ

บทนี้นำเสนอสรุปผลการวิเคราะห์เพื่อตอบวัตถุประสงค์ของงานวิจัย การอภิปรายประเด็นต่างๆ ที่เกิดขึ้นในงานวิจัย การนำงานวิจัยนี้ไปใช้ประโยชน์ในเชิงทฤษฎีและเชิงประยุกต์ ข้อจำกัดของงานวิจัยและข้อเสนอแนะเพื่อเป็นโอกาสในการศึกษาต่อไปในภายภาคหน้า

5.2 การศึกษาประเภทของข้อบกพร่องและปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง

5.2.1 ขั้นตอนการดำเนินงาน

งานวิจัยในส่วนนี้เริ่มจากการศึกษาประเภทของข้อบกพร่องจากเอกสารการออกแบบซอฟต์แวร์ และดำเนินการตรวจสอบซอฟต์แวร์โดยใช้เทคนิคโอโออาร์ที โดยเอกสารการออกแบบซอฟต์แวร์ที่ใช้เป็นโครงงานการออกแบบระบบทางด้านธุรกิจที่ออกแบบโดยแผนภาพยูเอ็มแอล ที่ออกแบบโดยนิสิตที่ศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ในรายวิชาการออกแบบระบบงานด้านธุรกิจ จำนวน 13 โครงงาน อย่างไรก็ดีในจำนวน 13 โครงงานที่ถูกคัดเลือกมามี 6 โครงงานที่ขาดความสมบูรณ์ในเอกสารอธิบายคลาส เนื่องจากไม่อธิบายการทำงานของเมท็อดในแต่ละคลาส โดยโครงงานที่เลือกมาเป็นโครงงานการออกแบบระบบงานด้านธุรกิจที่มีความหลากหลาย ได้แก่ ระบบการขาย ระบบคลังสินค้า ระบบการเช่า ระบบจัดการบริการลูกค้า ระบบรับคำสั่งซื้อและส่งมอบสินค้า ธุรกิจรับเหมาก่อสร้าง บริษัททัวร์ ธุรกิจดูแลรักษารถยนต์ ธุรกิจสปา เป็นต้น

หลังจากนั้นจะดำเนินการวิเคราะห์ข้อบกพร่องที่ค้นหาได้จากการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที โดยจะเปรียบเทียบจำนวนข้อบกพร่องของประเภทข้อบกพร่องจากเอกสารการออกแบบซอฟต์แวร์ เพื่อใช้ยืนยันความเหมาะสมของเอกสารการออกแบบซอฟต์แวร์ว่าสามารถเป็นตัวแทนที่ดีของเอกสารการออกแบบซอฟต์แวร์ทั้งหมด และเปรียบเทียบจำนวนข้อบกพร่องของแต่ละเอกสารการออกแบบซอฟต์แวร์ และเปรียบเทียบจำนวนข้อบกพร่องของขั้นตอน (Reading) การตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ที เพื่อใช้เป็นแนวทางในการปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์

5.2.2 บทสรุป

งานวิจัยในส่วนนี้ต้องการที่จะปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง โดยเริ่มจากการศึกษาประเภทข้อบกพร่องของเอกสารการออกแบบซอฟต์แวร์ และดำเนินการตรวจสอบเอกสารการออกแบบระบบงานด้วยธุรกิจใน ขั้นตอนการประชุมแนะนำ (overview) ขั้นตอนการจัดเตรียม (Preparation) และขั้นตอนการประชุมตรวจสอบ (Inspection Meeting) และนำผลการตรวจสอบซอฟต์แวร์จากเอกสารบันทึกรายการข้อบกพร่องในขั้นตอนการประชุมตรวจสอบของทุกโครงงานมาวิเคราะห์เพื่อปรับปรุงเทคนิคโอโออาร์ที

เมื่อตรวจสอบเอกสารการออกแบบซอฟต์แวร์โดยการใช้เทคนิคโอโออาร์ทีเรียบร้อยแล้ว ผู้วิจัยจะนำข้อบกพร่องจากเอกสารรายการข้อบกพร่องของแต่ละโครงงาน มาวิเคราะห์การกระจายตัวตามประเภทข้อบกพร่องในเอกสารการออกแบบจากงานวิจัยในอดีต โดยแบ่งประเภทข้อบกพร่องที่จะเปรียบเทียบออกเป็น 3 กลุ่ม ประกอบด้วย (1) ข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส (2) ข้อบกพร่องที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง และ (3) ข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง

ผลการเปรียบเทียบสรุปว่าจำนวนข้อบกพร่องของแต่ละประเภทที่สามารถตรวจสอบได้จากงานวิจัยในส่วนนี้ มีส่วนคล้ายคลึงกับจำนวนข้อบกพร่องของประเภทของข้อบกพร่องในอดีต (ดูตารางที่ 4-1, ตารางที่ 4-2 และตารางที่ 4-3 ประกอบ) กล่าวคือ (1) ข้อบกพร่องที่พิจารณาจากการตรวจสอบความสอดคล้องกันระหว่างแผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพยูสเคส ข้อบกพร่องประเภทข้อความไม่มีวิธีดำเนินการสามารถตรวจพบได้มากที่สุด (2) ข้อบกพร่องที่พิจารณาจากความสัมพันธ์ของข้อมูลและลักษณะของข้อบกพร่อง ข้อบกพร่องประเภทข้อมูลไม่สอดคล้องกันสามารถตรวจพบได้มากที่สุด ซึ่งเหมือนกับงานวิจัยในอดีต และ (3) ข้อบกพร่องที่พิจารณาจากสาเหตุที่ทำให้เกิดข้อบกพร่อง ข้อบกพร่องประเภทสิ่งผิดพลาดสามารตรวจพบได้มากที่สุด ซึ่งเหมือนกับงานวิจัยในอดีต รวมทั้งที่กล่าวไว้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) ด้วย

เมื่อนำข้อบกพร่องในเอกสารรายการข้อบกพร่องจากการตรวจสอบซอฟต์แวร์ มาวิเคราะห์การกระจายตัวของข้อบกพร่อง โดยพิจารณาข้อบกพร่องที่พบในแต่ละแผนภาพหรือเอกสารการออกแบบซอฟต์แวร์พบว่า แผนภาพคลาสสามารถตรวจพบข้อบกพร่องมากที่สุด รองลงมาคือ แผนภาพซีเควนซ์ และเอกสารอธิบายยูสเคสสามารถตรวจพบข้อบกพร่องได้น้อยที่สุด (ดูตารางที่ 4-4 ประกอบ)

ถ้าวิเคราะห์การกระจายตัวของข้อบกพร่อง โดยพิจารณาจำนวนข้อบกพร่องที่สามารถตรวจพบได้ในแต่ละขั้นตอนของเทคนิคโอโออาร์ที พบว่า ขั้นตอนที่ 1 ที่เปรียบเทียบระหว่างแผนภาพคลาสและแผนภาพซีเควนซ์ สามารถตรวจพบข้อบกพร่องได้มากที่สุด รองมาคือขั้นตอนที่ 5 ที่เปรียบเทียบระหว่างเอกสารอธิบายคลาส และเอกสารอธิบายความต้องการซอฟต์แวร์ และขั้นตอนที่สามารถตรวจพบข้อบกพร่องได้น้อยที่สุดคือ ขั้นตอนที่ 7 ที่เปรียบเทียบระหว่างแผนภาพสถานะ และเอกสารอธิบายความต้องการซอฟต์แวร์ / แผนภาพยูสเคส พร้อมทั้งเอกสารอธิบายยูสเคส (ดูตารางที่ 4-5 ประกอบ)

หลังจากนั้นจึงปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง โดยพิจารณาจากจำนวนข้อบกพร่องที่สามารถตรวจสอบได้ในแต่ละขั้นตอน (Reading) ของเทคนิคโอโออาร์ที และความครอบคลุมถึงทุกแผนภาพหรือเอกสารการออกแบบซอฟต์แวร์ที่นำมาตรวจสอบ ดังนั้นในการปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง จึงตัดขั้นตอนที่ 2 ขั้นตอนที่ 4 และขั้นตอนที่ 7 ออก เหลือเฉพาะขั้นตอนที่ 1 ขั้นตอนที่ 3 ขั้นตอนที่ 5 และขั้นตอนที่ 6 เท่านั้น (ดูรูปที่ 4-2 ประกอบ)

5.3 การเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และ เทคนิคซีบีอาร์

5.3.1 การทดลองและลักษณะของหน่วยทดลอง

งานวิจัยในส่วนนี้เป็นการวิจัยเชิงทดลอง โดยนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ที่ได้จากขั้นตอนแรกของงานวิจัยมาเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์กับเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ โดยหน่วยทดลองที่ใช้ในการทดลองในส่วนนี้ คือ นิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ และสาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่จบปริญญาตรีสาขาวิชาด้านคอมพิวเตอร์ หรือสาขาที่เกี่ยวข้อง ที่มีจำนวนหน่วยกิตในวิชาคอมพิวเตอร์อย่างน้อย 35 หน่วยกิต และผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุในระดับมหาบัณฑิตแล้ว จำนวน 24 คน

การตรวจสอบซอฟต์แวร์ของงานวิจัยในส่วนนี้ จะดำเนินการเฉพาะขั้นตอนการประชุมแนะนำ (Overview) และขั้นตอนการจัดเตรียม (Preparation) ของการตรวจสอบซอฟต์แวร์เพื่อนำผลการตรวจสอบที่ได้มาเปรียบเทียบกับผลการตรวจสอบด้วยเทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ที่เหมือนกัน ที่เป็นผลการวิจัยของ

อุมาพร นิลเอวะ (2549) โดยหน่วยทดลองของทั้งสามเทคนิคการอ่านซอฟต์แวร์ ไม่ได้ดำเนินการตรวจสอบเอกสารการออกแบบซอฟต์แวร์พร้อมกัน โดยเป็นการใช้หน่วยทดลองคนละกลุ่มที่มีคุณสมบัติเหมือนกัน โดยกำหนดให้มีจำนวนหน่วยทดลองในการตรวจสอบซอฟต์แวร์แต่ละเทคนิคจำนวน 24 คนเท่ากัน

5.3.2 บทสรุป

งานวิจัยในส่วนนี้เป็นการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยแผนภาพยูเอ็มแอล สามารถสรุปผลการเปรียบเทียบได้ดังนี้

5.3.2.1 การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ผลการเปรียบเทียบพบว่าการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์สูงกว่าการใช้เทคนิคโอโออาร์ทีเดิม แต่ยังคงมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์ไม่สูงกว่าการใช้เทคนิคซีบีอาร์ ซึ่งต่างจากที่ผู้วิจัยคาดไว้ก่อนที่จะทดลองว่า การใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วน่าจะมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์สูงกว่าการใช้เทคนิคโอโออาร์ทีเดิมและมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์อย่างน้อยเท่ากันกับการใช้เทคนิคซีบีอาร์

การที่ผลการวิจัยที่ได้มีลักษณะดังกล่าว อาจจะเกิดจากการที่เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วได้ลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลงแล้วทำให้มีขั้นตอนในการตรวจสอบซอฟต์แวร์น้อยลง ทำให้ผู้ตรวจสอบสามารถตรวจพบข้อบกพร่องที่มีในเอกสารการออกแบบซอฟต์แวร์ได้มากขึ้น เนื่องจากการลดขั้นตอนการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว คำนึงถึงปริมาณข้อบกพร่องที่สามารถตรวจพบได้ในแต่ละขั้นตอนและการครอบคลุมถึงเอกสารการออกแบบซอฟต์แวร์ทั้งหมดที่ตรวจสอบ ทำให้มีประสิทธิภาพในการตรวจสอบซอฟต์แวร์ดีกว่าเทคนิคโอโออาร์ทีเดิม แต่เนื่องจากเทคนิคซีบีอาร์มีขั้นตอนของเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ที่น้อยกว่ามาก โดยเอกสารคำแนะนำในการตวจสอบซอฟต์แวร์ของเทคนิคซีบีอาร์มีปริมาณไม่เกินหนึ่งหน้ากระดาษ ทำให้ผู้ตรวจสอบด้วยเทคนิคนี้อาจจะไม่ต้องใช้เวลามากในการค้นหาข้อบกพร่อง ทำให้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วไม่ได้มีประสิทธิภาพในการตรวจสอบซอฟต์แวร์สูงกว่าเทคนิคซีบีอาร์

5.3.2.2 การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ ผลการเปรียบเทียบพบว่าการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิผลในการตรวจสอบซอฟต์แวร์ไม่ต่างจากการใช้เทคนิคโอโออาร์ทีเดิมและการใช้เทคนิคซีบีอาร์ ซึ่งต่างจากที่ผู้วิจัยคาดไว้ก่อนที่จะทดลองว่า การใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วน่าจะมีค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์สูงกว่าการใช้เทคนิคโอโออาร์ทีเดิมและมีประสิทธิผลในการตรวจสอบซอฟต์แวร์อย่างน้อยเท่ากันกับการใช้เทคนิคซีบีอาร์

การที่ผลการวิจัยที่ได้มีลักษณะดังกล่าว อาจจะเกิดจากการที่ประสิทธิผลในการตรวจสอบซอฟต์แวร์ พิจารณาจากจำนวนข้อบกพร่องที่สามารถตรวจพบเทียบกับจำนวนข้อบกพร่องทั้งหมด แต่เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วไม่ได้คำนึงถึงข้อบกพร่องในเอกสารอธิบายคลาส ที่สามารถตรวจพบเมื่อเปรียบเทียบเอกสารอธิบายคลาสกับแผนภาพคลาส เนื่องจากเอกสารอธิบายคลาสไม่ได้เป็นเอกสารสำคัญที่แสดงถึงการออกแบบซอฟต์แวร์ที่ถูกนำไปใช้ในขั้นตอนต่อไปของการพัฒนาซอฟต์แวร์ (Conradi และคณะ, 2003) ดังนั้นข้อบกพร่องในส่วนนี้จึงไม่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว แต่สามารถตรวจพบได้ด้วยเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ดังนั้นเมื่อเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของการใช้เทคนิคการอ่านซอฟต์แวร์ทั้งสามเทคนิค จึงมีค่าไม่ต่างกัน

5.3.2.3 สรุปผลการวิเคราะห์ข้อมูลเพิ่มเติม (Exploration) การวิเคราะห์ข้อมูลเพิ่มเติม เป็นการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์ด้วยแผนภาพยูเอ็มแอล

การวิเคราะห์ข้อมูลเพิ่มเติมจะไม่คำนึงถึงข้อบกพร่องในเอกสารอธิบายคลาส ที่สามารถตรวจพบได้จากการเปรียบเทียบแผนภาพคลาสและเอกสารอธิบายคลาส เนื่องจากเอกสารอธิบายคลาสไม่ได้เป็นเอกสารสำคัญที่แสดงถึงการออกแบบซอฟต์แวร์ที่ถูกนำไปใช้ในขั้นตอนต่อไปของการพัฒนาซอฟต์แวร์ (Conradi และคณะ, 2003) โดยสามารถสรุปผลการวิเคราะห์ข้อมูลเพิ่มเติมได้ดังนี้

ผลการเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ พบว่าการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์สูงกว่าการใช้เทคนิคโอโออาร์ทีเดิม และการใช้เทคนิคซีบีอาร์

ผลการเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ พบว่าการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิผลในการตรวจสอบ

ซอฟต์แวร์ สูงกว่าการใช้เทคนิคโอโออาร์ทีเดิม แต่มีประสิทธิผลในการตรวจสอบซอฟต์แวร์ไม่แตกต่างกับการใช้เทคนิคซีบีอาร์

การที่ผลการวิจัยที่ได้มีลักษณะดังกล่าว น่าจะเกิดจากข้อบกพร่องที่ไม่ได้คำนึงถึงในการวิเคราะห์ข้อมูลเพิ่มเติม ซึ่งก็คือข้อบกพร่องในเอกสารอธิบายคลาส ที่สามารถตรวจพบได้จากการเปรียบเทียบแผนภาพคลาสและเอกสารอธิบายคลาส เป็นข้อบกพร่องที่ถูกตรวจสอบในการใช้เทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ตรวจสอบซอฟต์แวร์ ที่ดำเนินการในงานวิจัยของ อุมาพร นิลเอวะ (2549) แต่ไม่มีขั้นตอนการตรวจสอบข้อบกพร่องดังกล่าว จากการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วในการตรวจสอบซอฟต์แวร์ในงานวิจัยนี้

จากการไม่คำนึงถึงข้อบกพร่องดังกล่าว อาจเป็นไปได้ว่าหน่วยทดลองในงานวิจัยของ อุมาพร นิลเอวะ (2549) ที่ดำเนินการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ ได้ใช้เวลาในการตรวจสอบข้อบกพร่องดังกล่าวด้วย แต่สำหรับการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วในการตรวจสอบซอฟต์แวร์ของงานวิจัยนี้ ไม่ได้เสียเวลาในการตรวจสอบข้อบกพร่องดังกล่าว ทำให้ประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ที่ได้ไม่สมบูรณ์เมื่อเทียบกับเวลาที่ใช้ในการตรวจสอบ

5.4 การนำผลงานวิจัยไปประยุกต์ใช้ (Contribution)

งานวิจัยนี้นอกจากนำไปใช้ในเชิงทฤษฎีแล้ว ยังสามารถนำไปประยุกต์ใช้ในทางปฏิบัติได้ดังต่อไปนี้

5.4.1 การนำงานวิจัยไปใช้ในเชิงทฤษฎี (Theoretical Contribution) งานวิจัยนี้เป็นการต่อยอดองค์ความรู้ด้านการตรวจสอบซอฟต์แวร์ (Software Inspection) เนื่องจากผู้วิจัยทบทวนวรรณกรรมในอดีต (Literature Review) แล้วพบว่า มีการนำเทคนิคโอโออาร์ทีมาศึกษาน้อยมาก ทั้งที่เทคนิคโอโออาร์ทีเป็นเทคนิคสำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุโดยเฉพาะ

โดยมีงานวิจัยที่เกี่ยวข้องกับเทคนิคโอโออาร์ที คือ งานวิจัยของ Conradi และคณะ (2003) ที่ศึกษาเทียบเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีกับเทคนิคอาร์แอนด์ไอ (R&I Techniques) ซึ่งเป็นเทคนิคที่ใช้ในการตรวจสอบซอฟต์แวร์ของบริษัทอีริคสันประเทศนอร์เวย์ และงานวิจัยของ Travassos และคณะ (1999) ทดลองเพื่อวัดความสามารถในการค้นหาข้อบกพร่องในเอกสารการออกแบบ โดยใช้เทคนิคการอ่านซอฟต์แวร์ด้วยเทคนิคโอโออาร์ทีเพื่อเพิ่มคุณภาพของซอฟต์แวร์ ผลการทดลองสรุปว่าเทคนิคโอโอ

อาร์ทีมีขั้นตอนการตรวจสอบที่ละเอียด สามารถทำให้ผู้ตรวจสอบค้นหาข้อบกพร่องได้มากและเป็นที่พอใจของผู้ตรวจสอบ และอุมาพร นิลเอวะ (2549) เปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีกับเทคนิคซีบีอาร์ โดยนำมาใช้ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์เชิงวัตถุที่ออกแบบโดยแผนภาพยูเอ็มแอล ผลการทดลองแสดงว่าเทคนิโอโออาร์ทีไม่ได้มีประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์มากกว่าเทคนิคซีบีอาร์ อาจจะเกิดจากระยะเวลาในการตรวจสอบที่มีจำกัด ทำให้เทคนิคโอโออาร์ทีที่มีขั้นตอนการตรวจสอบที่มากกว่าสามารถตรวจสอบข้อบกพร่องได้น้อยกว่าเทคนิคซีบีอาร์

ดังนั้นผู้วิจัยจึงต้องการปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง และดำเนินการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคการอ่านซอฟต์แวร์ ระหว่าง เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว เทคนิคโอโออาร์ทีเดิม และเทคนิคซีบีอาร์ เนื่องจากงานวิจัยของ อุมาพร นิลเอวะ (2549) ได้เปรียบเทียบเทคนิคโอโออาร์ทีกับเทคนิคซีบีอาร์แล้ว ดังนั้นผู้วิจัยจึงต้องการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วกับเทคนิคซีบีอาร์อีกครั้ง ว่าเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วจะมีประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์สูงกว่าเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีเดิมหรือไม่ เพื่อเป็นแนวทางให้กับผู้ที่สนใจในด้านของการตรวจสอบซอฟต์แวร์ได้นำข้อมูลเหล่านี้ไปใช้ในการศึกษาต่อไป

5.4.2 การนำงานวิจัยไปใช้ในเชิงประยุกต์ (Practical Contribution) จากผลการทดลองของงานวิจัยนี้แสดงให้เห็นว่า การใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์สูงกว่าการใช้เทคนิคโอโออาร์ทีเดิม แต่การใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์ไม่สูงกว่าการใช้เทคนิคซีบีอาร์ และการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิผลในการตรวจสอบซอฟต์แวร์ไม่แตกต่างจากการใช้เทคนิคโอโออาร์ทีเดิมและการใช้เทคนิคซีบีอาร์

เมื่อสำรวจผลการวิเคราะห์ข้อมูลเพิ่มเติม โดยลดจำนวนข้อบกพร่องทั้งหมดที่จะพิจารณาลง โดยไม่คำนึงถึงข้อบกพร่องในเอกสารอธิบายคลาสที่สามารถตรวจพบเมื่อเทียบกับแผนภาพคลาสพบว่า การใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์สูงกว่าการใช้เทคนิคโอโออาร์ทีเดิมและการใช้เทคนิคซีบีอาร์ และการใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิผลในการตรวจสอบซอฟต์แวร์สูงกว่าการใช้เทคนิคโอโออาร์ทีเดิม แต่การใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิผลในการตรวจสอบซอฟต์แวร์ไม่แตกต่างจากการใช้เทคนิคซีบีอาร์

การเลือกเทคนิคการอ่านซอฟต์แวร์ไปใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยวิธีการเชิงวัตถุ ถ้าจำกัดระยะเวลาที่ใช้ในการตรวจสอบซอฟต์แวร์และผู้ตรวจสอบมีประสบการณ์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุมาแล้ว องค์กรรับจ้างพัฒนาซอฟต์แวร์ควรจะนำเทคนิคซีบีอาร์ไปใช้ในการตรวจสอบซอฟต์แวร์เนื่องจากเป็นเทคนิคการอ่านซอฟต์แวร์ที่ไม่ใช้เวลาในการตรวจสอบมาก แต่ก็สามารถพบข้อบกพร่องได้ เนื่องจากเอกสารเช็กลิสต์ที่เป็นเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์สำหรับเทคนิคซีบีอาร์ ไม่ได้มีคำอธิบายในการตรวจสอบที่ละเอียดจึงไม่เสียเวลาในการอ่านและปฏิบัติตามมากนัก โดยจำเป็นต้องอาศัยประสบการณ์ของผู้ตรวจสอบที่จะนำเทคนิคซีบีอาร์ไปใช้

ถ้าองค์กรรับจ้างพัฒนาซอฟต์แวร์ทีเลือกเทคนิคการอ่านซอฟต์แวร์ โดยคำนึงถึงการจำกัดระยะเวลาที่ใช้ในการตรวจสอบซอฟต์แวร์ และผู้ตรวจสอบซอฟต์แวร์ไม่มีประสบการณ์ในการตรวจสอบซอฟต์แวร์มาก องค์กรรับจ้างพัฒนาซอฟต์แวร์ควรนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วไปใช้ในการตรวจสอบซอฟต์แวร์ เนื่องจากเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วมีประสิทธิภาพในการตรวจสอบซอฟต์แวร์สูงกว่าเทคนิคโอโออาร์ทีเดิมและเทคนิคซีบีอาร์ และมีประสิทธิผลในการตรวจสอบซอฟต์แวร์สูงกว่าเทคนิคโอโออาร์ทีเดิม แต่มีประสิทธิผลเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วเท่ากันกับเทคนิคซีบีอาร์

ถ้าไม่ได้คำนึงถึงข้อบกพร่องของเอกสารอธิบายคลาสที่เกิดจากการค้นหาโดยเปรียบเทียบเอกสารอธิบายคลาสและแผนภาพคลาส โดยคำอธิบายคลาสไม่ได้เป็นเอกสารสำคัญที่แสดงถึงการออกแบบซอฟต์แวร์ที่ถูกนำไปใช้ในขั้นตอนต่อไปของการพัฒนาซอฟต์แวร์ แต่เป็นเอกสารที่ใช้อธิบายแผนภาพคลาสเท่านั้น ดังนั้นข้อบกพร่องที่เกิดในส่วนของเอกสารอธิบายคลาสจึงไม่ใช่ข้อบกพร่องที่มีความสำคัญที่ต้องได้รับการแก้ไขโดยด่วน และยังทำให้เพิ่มต้นทุนในการแก้ไข โดยที่ไม่ได้ใช้ประโยชน์จากเอกสารเท่าที่ควร (Conradi และคณะ, 2003)

ถ้าต้องการนำเทคนิคโอโออาร์ทีที่ปรับปรุงแล้วไปใช้ในการตรวจสอบซอฟต์แวร์ต้องยอมรับการมีข้อบกพร่องบางระการของเทคนิคนี้ เนื่องจากเทคนิคนี้ได้ลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง อาจจะทำให้ข้อบกพร่องที่สามารถค้นหาได้จากขั้นตอนที่ 2 ที่เปรียบเทียบระหว่างแผนภาพสถานะ และเอกสารอธิบายคลาส และขั้นตอนที่ 7 ที่เปรียบเทียบระหว่างแผนภาพสถานะ และ เอกสารอธิบายความต้องการซอฟต์แวร์ / แผนภาพยูสเคสพร้อมทั้งเอกสารอธิบายยูสเคส ของเทคนิคโอโออาร์ทีไม่ได้รับการตรวจสอบ

5.5 ข้อจำกัดและข้อเสนอแนะของงานวิจัย

งานวิจัยนี้มีข้อจำกัดบางประการ ผู้วิจัยขอสรุปและเสนอแนะแนวทาง ดังต่อไปนี้

1. การนำผลการทดลองไปใช้ให้คำนึงด้วยว่า ในส่วนแรกของงานวิจัยที่ต้องการปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนนำในการตรวจสอบซอฟต์แวร์ลง ได้มีการตรวจสอบซอฟต์แวร์ด้วยเทคนิคโอโออาร์ที โดยผู้ตรวจสอบที่ใช้เป็นพนักงานบริษัทรับจ้างพัฒนาซอฟต์แวร์ ตรวจสอบร่วมกับนิสิตในระดับมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ไม่ใช่ผู้ตรวจสอบมืออาชีพจากองค์กรรับจ้างพัฒนาซอฟต์แวร์ทุกคน ดังนั้นในการนำไปขยายผลกับผู้ตรวจสอบซอฟต์แวร์ในองค์กรรับจ้างพัฒนาซอฟต์แวร์อาจได้ผลที่แตกต่างไป

2. ในการปรับปรุงเทคนิคโอโออาร์ทีโดยลดจำนวนเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ลง เป็นการตรวจสอบโครงงานการออกแบบระบบงานด้านธุรกิจของนิสิตสาขาเทคโนโลยีสารสนเทศทางธุรกิจ จำนวน 13 โครงงาน แต่ถ้านำไปขยายผลโดยเพิ่มจำนวนโครงงานมากกว่านี้อาจจะได้ผลที่แตกต่างไป

3. ในการทดลองผู้วิจัยได้กำหนดระยะเวลาที่ใช้ในการทดลองขั้นตอนการจัดเตรียมไว้ แต่ปกติแล้วการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม เป็นขั้นตอนที่ให้ผู้ตรวจสอบนำเอกสารกลับไปตรวจสอบแล้วนำมาส่ง จึงไม่มีการกำหนดระยะเวลาที่แน่นอน ดังนั้นเมื่อนำผลที่ได้จากการทดลองไปขยายผลกับการตรวจสอบซอฟต์แวร์ในองค์กรรับจ้างพัฒนาซอฟต์แวร์ อาจให้ผลที่แตกต่างออกไป

4. เอกสารแสดงการออกแบบซอฟต์แวร์ที่นำมาใช้ในการทดลองเป็นระบบที่มีขนาดเล็กและมีความซับซ้อนไม่มากนัก เพื่อให้หน่วยทดลองสามารถค้นหาข้อบกพร่องได้ในระยะเวลาที่กำหนด โดยเมื่อนำไปขยายผลกับระบบจริงที่องค์กรรับจ้างพัฒนาซอฟต์แวร์จัดทำขึ้นอาจให้ผลที่แตกต่างไป

5. ข้อบกพร่องที่ใส่ในเอกสารแสดงการออกแบบซอฟต์แวร์ที่นำมาใช้ในการทดลองมีจำนวน 20 ข้อ รวมข้อบกพร่องที่มีในเอกสารอธิบายคลาสที่สามารถค้นหาได้เฉพาะจากการเปรียบเทียบแผนภาพคลาสและเอกสารอธิบายคลาสด้วย ทำให้ระยะเวลาที่ใช้เป็นระยะเวลาของการค้นหาข้อบกพร่อง 20 ข้อ ถ้าทดลองใหม่โดยไม่ใส่ข้อบกพร่องในส่วนนี้ในเอกสารการออกแบบซอฟต์แวร์เลย อาจจะให้ผลที่แตกต่างไป

รายการอ้างอิง

ภาษาไทย

กัลยา วาณิชย์บัญชา. 2550. การใช้ SPSS for Windows ในการวิเคราะห์ข้อมูล. พิมพ์ครั้งที่ 10. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

กัลยา วาณิชย์บัญชา. 2550. การวิเคราะห์สถิติ : สถิติสำหรับการบริหารและวิจัย. พิมพ์ครั้งที่ 9. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

กัลยา วาณิชย์บัญชา. 2544. หลักสถิติ. พิมพ์ครั้งที่ 6. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

ชาลี วรกุลพิพัฒน์ และ เทพฤทธิ์ บัณฑิตวัฒนาวงศ์. 2544. UML ภาษามาตรฐานเพื่อผู้พัฒนาซอฟต์แวร์. กรุงเทพมหานคร: สำนักพิมพ์ซีเอ็ดยูเคชั่น.

บงกชธร เพ็ชรนิจินดา. 2548. Software quality assurance. PC magazine 11 (มี.ค. 48) : 121 – 139

ศิริวรรณ เสรีรัตน์, สมชาย หิรัญกิตติ, จิรศักดิ์ จิยะจันทน์, ชวลิต ประภวานนท์, ณดา จันทร์สม และ วลัยลักษณ์ อัตธีรวงศ์. 2541. การวิจัยธุรกิจ. กรุงเทพมหานคร: สำนักพิมพ์เพชรจรัสแสงแห่งโลกธุรกิจ.

อุมาพร นิลเอวะ. 2549. การเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยวิธีการเชิงวัตถุ. วิทยานิพนธ์ปริญญามหาบัณฑิต. ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย.

ภาษาอังกฤษ

Babbie, E. 2001. The Practice of Social Research. Ninth Edition. USA: Wadsworth/Thomson Learning.

Basili, V., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sorumgard, S. and Zelkowitz, M. 1996. The Empirical Investigation of Perspective-based Reading. Empirical Software Engineering: An International Journal 1: 133-164.

Biffl, S. and Halling, M. 2002. Investigating the Influence of Inspector Capability Factors with Four Inspection Techniques on Inspection Performance. Eighth IEEE International Symposium on Software Metrics (METRICS'02) : 107-117.

Booch 1994. Object-Oriented Analysis and Design with Application. New York: Benjamin-Cummings Publishing Company.

Booch, G., Rumbaugh, J. and Jacobson, I. 1999. The Unified Modeling Language User Guide. USA: Addison-Wesley.

Cheng, B., and Jeffery, R. 1996. Comparing Inspection Strategies for Software Requirements Specifications. Proceedings of the 1996 Australian Software Engineering Conference : 203-211.

Chernak, Y. 1996. A Statistical Approach to the Inspection Checklist Formal Synthesis and Improvement. IEEE Transactions on Software Engineering 12: 866-874.

Conradi, R., Mohagheghi, P., Arif, T., Hegde L. C., Bunde G. A. and Pedersen A. 2003. Object-Oriented Reading Techniques for Inspection of UML Models – An Industrial Experiment. Lecture Notes in Computer Science, Publisher: Springer Berlin / Heidelberg 2743: 483-500.

Dennis, A., Wixom B. H. and Tegarden D. 2005. Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach. Second Edition. USA: John Wiley & Sons.

Fagan, M. E. 1976. Design and Code Inspection to Reduce Errors in Program Development. IBM Systems Journal 3: 182-211.

Fowler, P., 1986. In-process Inspections of Work products at AT&T. AT&T Technical Journal :102–112.

Frankovich, J. 1995. The Software Inspection Process [On-Line]. Available from: http://sern.ucalgary.ca/courses/seng/621/W97/johnf/inspections.htm#1.0

Fusaro, P., Lanubile, F. and Visaggio G. 1997. A replicated experiment to assess requirements inspections techniques. Empirical Software Engineering Journal 2, 1: 39-57.

Gilb, T. and Graham, D. 1993. Software Inspection. USA: Addison-Wesley.

He, L. and Carver, J. 2006. PBR vs. Checklist : A Replication in the N-old Inspection Context. Proceedings of the 2006 International Symposium on Empirical Software Engineering : 95-104.

Horch, J. W. 1996. Practical Guide to Software Quality Management. New York: Artech House.

IEEE Standard. 1997. IEEE Standard for Software Reviews [On-Line]. Available from : http://en.wikipedia.org/wiki/Software_walkthrough

IEEE Standard. 1997. IEEE Standard for Software Reviews [On-Line]. Available from : http://en.wikipedia.org/wiki/Software_inspection

Jacobson 1992. Object-Oriented Software Engineering: A Use Case Driven Approach. New York: Addison-Wesley Professional.

Johnson, P. M. and Tjahjono D. 1998. Does Every Inspection Really Need a Meeting?, Empirical Software Engineering. An International Journal 3, 1: 9-35.

Laitenberger, O., Atkinson, C., Schlich, M. El-Emam, K. 1999. An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents. Canada: National Research Council of Canada.

Laitenberger, O., Atkinson, C., Schlich, M. and Emam K. E. 2000. An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents. Journal of Systems and Software 53, 2: 183-204.

Lange, C and Chaundron, M. 2006. Effect of Defects in UML Models – An Experimental Investigation. Proceeding of the 28th international conference on Software engineering **:** 401 - 411

Mrdalj, S., Scazzero, J. and Jovanovic V. 2004. Effectiveness of the User Interface Driven System Design Using UML [On-Line]. Available from: http://www.comsis.fon.bg.ac.yu/ComSIS/Volume02/Papers/Mrdalj.htm

O'Regan, G. 2003. A Practical Approach to Software Quality. New York: Springer-Verlag.

Parnas, D. L. and Lawford M. 2003. The Role of Inspection in Software Quality Assurance. IEEE Transactions on Software Engineering 29, 8: 674-676.

Porter, A. A. and Votta, L. G. 1994. An Experiment to Assess Different Detection Methods for Software Requirements Inspection. Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy : 103-112.

Porter, A., Votta, L. G. and Basili, V. 1995. Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment. IEEE Transactions on Software Engineering 21, 6: 563-575.

Porter, A. and Votta, L. G. 1998. Comparing Detection Methods for Software Requirements Specification: A Replication Using Professional Subjects. Empirical Software Engineering 3: 355-379.

Pressman, R. S. 1997. Software Engineering: A Practitioner's Approach. Fourth Edition. USA: McGraw-Hill.

Rumbaugh, R., Blaha, R., Lorensen, Eddy, Premerlani 1991. Object-Oriented Modeling and Design. New York: Prentice-Hall.

Sabaliauskaite, G., Matsukawa, F., Kusumoto, S. and Inoue K. 2002. An Experimental Comparison of Checklist-Based Reading and Perspective-Based Reading for UML Design Document Inspection. Proceedings of The 2002 International Symposium on Empirical Software Engineering : 148-157.

Schulmeyer, G. G. and Mcmanus J. I. 1992. Handbook of Software Quality Assurance. Second Edition. New York: Van Nostrand Reinhold.

Shull, F. 1998. Developing Techniques for Using Software Documents: A Series of Empirical Studies. Ph.D. Thesis. University of Maryland.

Shull, F., Travassos, G. and Basili, V. 1999. Towards Techniques for improved OO Design Inspections. ECOOP Workshop : 334.

Strauss, S. H. and Ebenau R. G. 1994. Software Inspection Process. USA: McGraw-Hill.

Thelin, T., Runeson, P. and Wohlin, C. 2003. An Experimental Comparison of Usage-Based Reading and Checklist-Based Reading. IEEE Transactions on Software Engineering 29, 8: 687-704.

Travassos, G., Shull,F., Fredericks, M. and Basili, 1999. Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality. Proceedings of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications : 47-56.

Travassos, G., Shull,F., Carver, J., and Basili, 2002. Reading Techniques for OO Design Inspections[On-Line]. Available from: www.cs.umd.edu/Library/TRs/CS-TR-4353/CS-TR-4353.pdf

United States Department of Defense. 1988. Defense System Software Quality Program. Washington DC.: NAVMAT 09Y.

Votta, L. 1993. Does Every Inspection Need a Meeting?. ACM Software Engineering Notes 18, 5: 107–114.

Zhang, Z., Basili, V. and Shneiderman, B. 1998. An empirical study of Perspective -based usability inspection. Human Factors and Ergonomics Society Annual Meeting, Chicago.

ภาคผนวก

ภาคผนวก ก
รายละเอียดของระบบที่ได้รับการตรวจสอบเพื่อปรับปรุงเทคนิคโอโออาร์ที

1. รายชื่อระบบที่ตรวจสอบเพื่อปรับปรุงเทคนิคโอโออาร์ที

1. ระบบจัดจำหน่ายสื่อการสอน
2. ระบบสนามเทนนิส
3. ระบบรับคำสั่งซื้อและส่งมอบสินค้า ในธุรกิจผลิตพลาสติก
4. ระบบจัดการการบริการลูกค้าของคลินิกความงาม
5. ระบบคลังสินค้าของธุรกิจขายวีดีโอ
6. ระบบบริหารจัดการสปาของธุรกิจให้บริการสปา
7. ระบบร้านดูแลรักษารถยนต์
8. ระบบธุรกิจการขายบ้านจัดสรร
9. ระบบธุรกิจด้านท่องเที่ยว
10. ระบบรับเหมาก่อสร้าง
11. ระบบจัดการบริหารห้องพัก
12. ระบบจัดการการขายและคลังสินค้าร้านดอกไม้
13. ระบบจัดการการขายอาคารชุด

2. รายละเอียดของเอกสารการออกแบบระบบงานด้านธุรกิจ

โครงงาน	จำนวนรายละเอียดของเอกสารการออกแบบระบบงานด้านธุรกิจ			
	จำนวนยูสเคส	จำนวนคลาส	จำนวนแผนภาพซีเควนซ์	จำนวนแผนภาพสถานะ
1	13	22	13	2
2	15	16	19	3
3	7	19	24	7
4	8	11	9	1
5	11	43	29	1
6	41	36	43	3
7	6	14	12	1
8	6	19	6	3
9	15	26	39	1
10	8	12	21	1
11	9	12	16	2
12	10	20	25	3
13	6	19	6	3

ภาคผนวก ข
เอกสารแสดงการออกแบบซอฟต์แวร์

1. เอกสารแสดงการออกแบบซอฟต์แวร์

เอกสารแสดงการออกแบบซอฟต์แวร์ที่ใช้ในงานวิจัยนี้เป็นเอกสารที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) ประกอบด้วย แผนภาพยูสเคส (Use Case Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) แผนภาพสถานะ (State Machine Diagram) และแผนภาพคลาส (Class Diagram) นอกจากนี้ยังมีเอกสารอธิบายคลาสของระบบ (Class Description) คำอธิบายยูสเคส (Use Case Description) และเอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) โดยมีรายละเอียดดังนี้

1.1 เอกสารแสดงความต้องการซอฟต์แวร์

Requirement Description

ระบบการเช่ารถนี้จัดทำขึ้นเพื่อให้ Officer สามารถจัดการการให้บริการเช่ารถได้อย่างสะดวก โดยระบบประกอบด้วยฟังก์ชันการทำงานต่างๆดังต่อไปนี้

1. เมื่อลูกค้าต้องการเช่ารถ Officer จะสร้างข้อมูลของลูกค้าที่ต้องการเช่ารถ ซึ่งประกอบด้วยชื่อ-สกุล เลขที่ใบขับขี่ ที่อยู่ และเบอร์โทรศัพท์

2. Officer สามารถค้นหารถที่ยังไม่ได้ถูกเช่าในวันที่ลูกค้าต้องการเช่าได้ โดยในการค้นหาจะค้นหาตามประเภทรถและวันที่ลูกค้าต้องการเช่า กรณีที่ค้นหาแล้วไม่พบรถว่างลูกค้าจะไม่สามารถเช่ารถตามที่ต้องการได้

3. เมื่อสามารถค้นหารถตามที่ลูกค้าต้องการได้แล้ว ลูกค้าจึงจะสามารถเช่ารถได้ โดย Officer จะสร้างรายการการเช่ารถซึ่งประกอบด้วย ข้อมูลส่วนบุคคลของลูกค้าที่ต้องการเช่ารถ วันที่ต้องการเช่า รถที่ต้องการเช่า และสถานะรายการการเช่ารถ

4. รายการการเช่ารถแต่ละรายการสามารถเช่ารถได้เพียง 1 คันเท่านั้น ถ้าลูกค้าต้องการเช่ามากกว่า 1 คัน จะต้องสร้างรายการการเช่าอีกรายการ

5. เมื่อถึงวันแรกของการเช่ารถ ลูกค้าจึงจะสามารถมารับรถได้ โดย Officer สามารถสอบถามรายการการเช่ารถที่ลูกค้าต้องการมารับรถ (ลูกค้าสามารถมารับรถได้ภายในกำหนดระยะเวลาที่เช่า)

6. เมื่อลูกค้าต้องการยกเลิก Officer สามารถสอบถามรายการการเช่าที่ลูกค้าต้องการยกเลิกได้

7. เมื่อลูกค้านำรถมาคืน Officer สามารถสอบถามรายการการเช่าที่ลูกค้าต้องการนำรถมาคืนได้ โดยลูกค้าจะต้องนำรถมาคืนภายในกำหนดเวลาที่เช่า (ลูกค้านำรถมาคืนได้ไม่เกินวันสุดท้ายของการเช่ารถ)

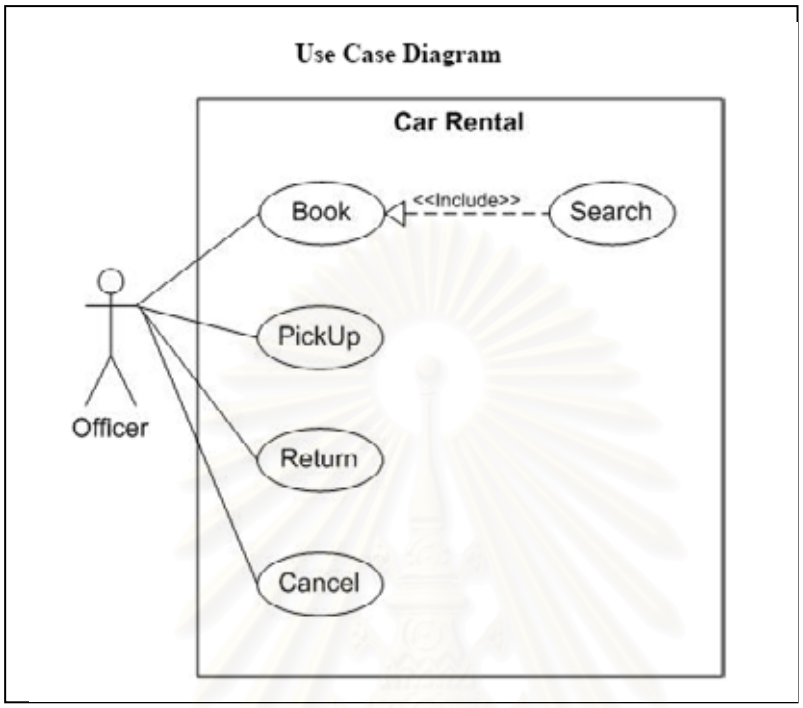
8. เมื่อลูกค้ายกเลิกรายการการเช่ารถ หรือนำรถที่เช่ามาคืนเรียบร้อยแล้ว Officer จะทำการลบข้อมูลของลูกค้านั้นทิ้ง

9. Officer สามารถเปลี่ยนสถานะของรายการการเช่ารถแต่ละรายการได้ โดยเมื่อจัดทำรายการการเช่ารถ สถานะของรายการการเช่าจะเป็น Open เมื่อลูกค้ามารับรถสถานะของรายการการเช่าจะถูกเปลี่ยนเป็น Pick Up และเมื่อลูกค้านำรถที่เช่ามาคืนหรือเมื่อลูกค้ายกเลิกรายการการเช่า Officer จะลบรายการการเช่านั้นทิ้ง

10. Officer สามารถเปลี่ยนสถานะของรถแต่ละคัน โดยสถานะของรถจะเป็น Available เมื่อลูกค้ามารับรถสถานะของรถจะถูกเปลี่ยนเป็น Pick Up และเมื่อลูกค้านำรถมาคืนสถานะของรถจะ ถูกเปลี่ยนเป็น Available ใหม่

<u>หมายเหตุ</u> ระบบนี้ไม่คำนึงถึงการคำนวณค่าใช้จ่ายในการเช่ารถ ไม่คำนึงถึงการเพิ่มหรือยกเลิกรถออกจากระบบ และกำหนดว่าลูกค้าไม่สามารถทำรายการการเช่าหลายรายการในช่วงเวลาเดียวกันได้

1.2 แผนภาพยูสเคสและคำอธิบายยูสเคส



Use Case Description

1. Book	
Goal	จัดทำรายการการเช่ารถ
Actor	Officer
Precondition	ลูกค้าแจ้งความต้องการในการเช่ารถเรียบร้อยแล้ว
Postcondition	จัดทำรายการการเช่ารถเรียบร้อยแล้ว (กรณีที่มีรถว่าง) หรือลบข้อมูลลูกค้าทิ้ง (กรณีที่ไม่มีรถว่าง)
Main Success Scenario	1. Officer สร้างข้อมูลของลูกค้าที่ต้องการเช่ารถ ซึ่งประกอบด้วยชื่อ-สกุล เลขที่ใบขับขี่ ที่อยู่และเบอร์โทรศัพท์ 2. ได้รับข้อมูลลูกค้าที่ต้องการเช่ารถ 3. ค้นหารถที่มีสถานะว่างในวันที่ลูกค้าต้องการเช่า 4. Officer สร้างรายการการเช่าซึ่งประกอบด้วยข้อมูลของลูกค้าที่ต้องการเช่ารถ วันที่ต้องการเช่า รถที่ต้องการเช่า สถานะของรายการการเช่ารถ 5. ได้รายการการเช่าตามที่สร้างไว้
Extensions	3a กรณีที่ไม่สามารถค้นหารถตามที่ลูกค้าต้องการ Officer ไม่สามารถสร้างรายการการเช่ารถได้ (ลบข้อมูลลูกค้าทิ้ง)

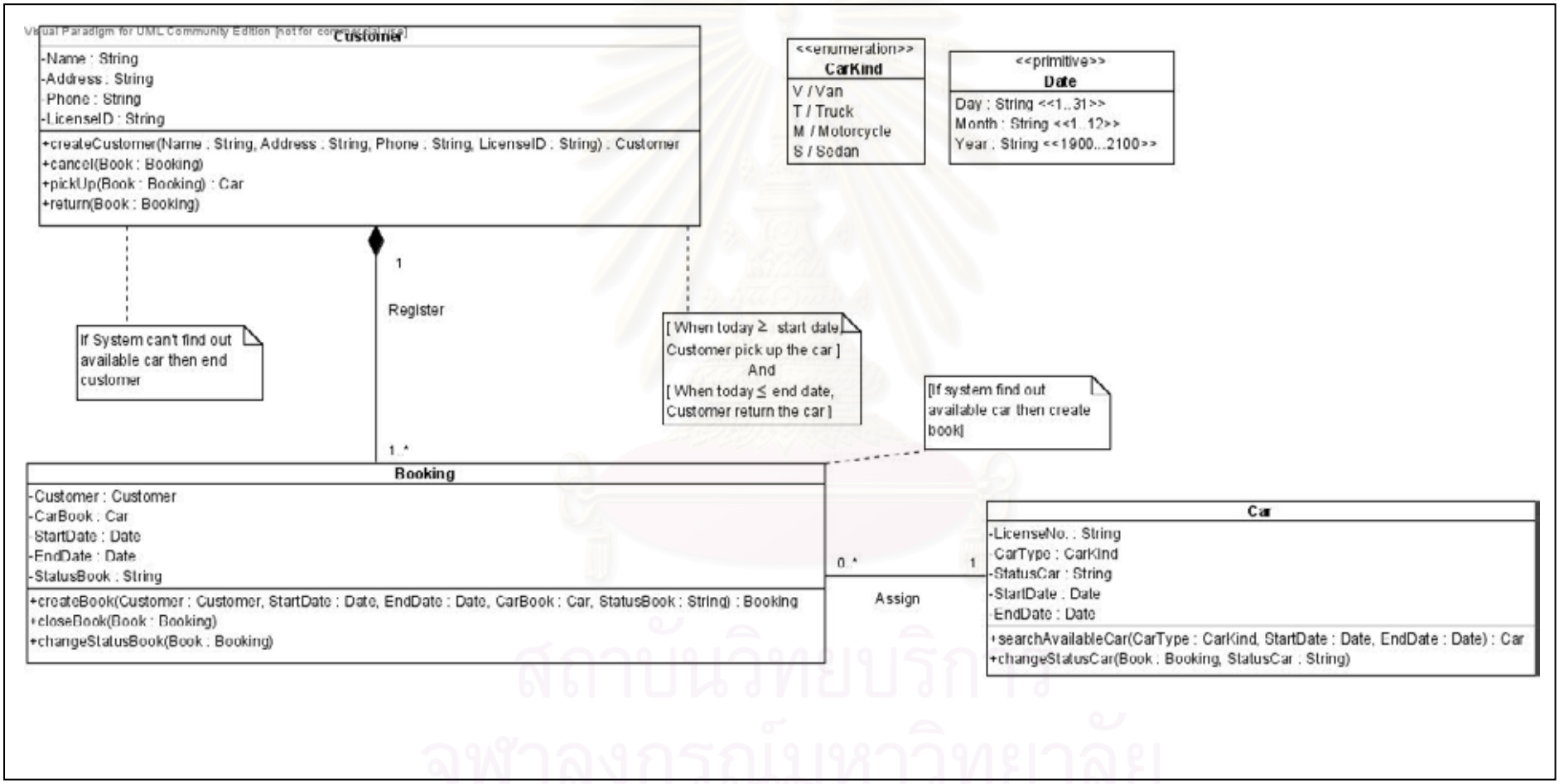
2. Search	
Goal	ค้นหารถที่มีสถานะว่างในวันที่ลูกค้าต้องการเช่า โดยค้นหาตามประเภทรถที่ลูกค้าต้องการ
Actor	Officer
Precondition	ลูกค้าแจ้งข้อมูลต่างๆที่ใช้ในการเช่ารถเรียบร้อยแล้ว
Postcondition	ได้รับผลการค้นหารถเรียบร้อยแล้ว
Main Success Scenario	1. Officer ค้นหารถที่ว่าง โดยค้นหาตามประเภทรถที่ลูกค้าต้องการ และวันที่ลูกค้าต้องการเช่ารถ 2. Officer ได้รับข้อมูลการค้นหารถ

3. PickUp	
Goal	จัดการการรับรถของลูกค้า
Actor	Officer
Precondition	ลูกค้าต้องการรับรถและอยู่ในช่วงของระยะเวลาที่ลูกค้าต้องการเช่า
Postcondition	ลูกค้าได้รับรถตามรายการการเช่าเรียบร้อยแล้ว
Main Success Scenario	1. Officer สอบถามรายการการเช่าที่ลูกค้าต้องการมารับรถ 2. ได้ข้อมูลรถที่ลูกค้าต้องการมารับ 3. Officer เปลี่ยนสถานะรถของรายการการเช่านี้เป็น Pick Up 4. Officer เปลี่ยนสถานะรายการการเช่านี้เป็น Pick Up

4. Return	
Goal	จัดการการคืนรถของลูกค้า
Actor	Officer
Precondition	ลูกค้าต้องการคืนรถและอยู่ในช่วงเวลาไม่เกินวันสุดท้ายของการเช่ารถ
Postcondition	ลูกค้าคืนรถเรียบร้อยแล้ว
Main Success Scenario	1. Officer สอบถามรายการการเช่าที่ลูกค้าต้องการนำรถมาคืน 2. Officer เปลี่ยนสถานะรถของรายการการเช่านี้เป็น Available 3. Officer ลบรายการการเช่านั้น 4. Officer ลบลูกค้าทิ้ง

5. Cancel	
Goal	ยกเลิกรายการการเช่า
Actor	Officer
Precondition	ลูกค้าต้องการยกเลิกรายการการเช่ารถ
Postcondition	ลบรายการการเช่าเรียบร้อยแล้ว
Main Success Scenario	1. Officer สอบถามรายการการเช่าที่ลูกค้าต้องการยกเลิก 2. Officer ลบรายการการเช่านั้น 3. Officer ลบลูกค้าทิ้ง

1.3 แผนภาพคลาสและคำอธิบายคลาส



Class Description

1. Customer				
Description	ลูกค้าที่ต้องการเช่ารถ			
Superclass	-			
Subclass	-			
Attributes				
Name	**Description**	**Data Type**	**Initial Value**	**Visibility**
Name	ชื่อลูกค้า	String	-	Private
LicenseID	เลขที่ใบขับขี่ลูกค้า	String	-	Private
Methods				
1. createCustomer				
Description	ฟังก์ชันที่ใช้สร้างข้อมูลลูกค้าที่ต้องการเช่ารถ			
Visibility	Public			
Parameters	**Name**	**Description**	**Data Type**	
	Name	ชื่อลูกค้า	String	
	Address	ที่อยู่ลูกค้า	String	
	Phone	เบอร์โทรศัพท์ลูกค้า	String	
	LicenseID	เลขที่ใบขับขี่ลูกค้า	String	
Return Type	Customer			
2. pickUp				
Description	ฟังก์ชันที่ใช้สอบถามรายการการเช่าที่ลูกค้าต้องการมารับรถ			
Visibility	Public			
Parameters	**Name**	**Description**	**Data Type**	
	Book	รายการการเช่าที่ต้องการมารับรถ	Booking	
Return Type	Car			
3. return				
Description	ฟังก์ชันที่ใช้สอบถามรายการการเช่าที่ลูกค้าต้องการนำรถมาคืน			
Visibility	Public			
Parameters	**Name**	**Description**	**Data Type**	
	Book	รายการการเช่าที่ต้องการนำรถมาคืน	Booking	
Return Type	-			

4. cancel			
Description	ฟังก์ชันที่ใช้สอบถามรายการการเช่าที่ลูกค้าต้องการยกเลิก		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Book	รายการการเช่าที่ต้องการยกเลิก	Booking
Return Type	-		
5. endCustomer			
Description	ฟังก์ชันที่ใช้ลบข้อมูลลูกค้า		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Customer	ลูกค้าที่ต้องการลบ	Customer
Return Type	-		

2. Officer

Description	พนักงานของบริษัทให้บริการเช่ารถ
Superclass	-
Subclass	-

Attributes

Name	Description	Data Type	Initial Value	Visibility
Name	ชื่อพนักงาน	String	-	Private
OfficerID	รหัสพนักงาน	String	-	Private

Methods

1. createOfficer			
Description	ฟังก์ชันที่ใช้สร้างข้อมูลพนักงาน		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Name	ชื่อพนักงาน	String
	OfficerID	รหัสพนักงาน	String
Return Type	Officer		
2. endOfficer			
Description	ฟังก์ชันที่ใช้ลบข้อมูลพนักงาน		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Officer	ข้อมูลพนักงานที่ต้องการลบ	Officer
Return Type	-		

3. Booking					
Description	รายการการเช่ารถ				
Superclass	-				
Subclass	Car				
Attributes					
Name	**Description**		**Data Type**	**Initial Value**	**Visibility**
Customer	ลูกค้าที่ต้องการเช่ารถ		Customer	-	Private
StartDate	วันเริ่มต้นการเช่ารถ		Date	-	Private
EndDate	วันสิ้นสุดการเช่ารถ		Date	-	Private
CarBook	รถที่ต้องการเช่า		Car	-	Private
StatusBook	สถานะของรายการการเช่า		String	Open	Private
Methods					
1. createBook					
Description	ฟังก์ชันที่ใช้สร้างรายการการเช่ารถ				
Visibility	Public				
Parameters	**Name**	**Description**		**Data Type**	
	Customer	ข้อมูลลูกค้าของรายการการเช่านี้		Customer	
	StartDate	วันเริ่มต้นการเช่ารถ		Date	
	EndDate	วันสิ้นสุดการเช่ารถ		Date	
	CarBook	รถคันที่ต้องการเช่าสำหรับรายการการเช่านี้		Car	
	StatusBook	สถานะของรายการการเช่า		String	
Return Type	Booking				
2. closeBook					
Description	ฟังก์ชันที่ใช้ลบรายการการเช่ารถ				
Visibility	Public				
Parameters	**Name**	**Description**		**Data Type**	
	Book	รายการการเช่าที่ต้องการลบ		Booking	
Return Type	-				

4. Car					
Description	รถที่ให้เช่า				
Superclass	-				
Subclass	-				
Attributes					
Name	**Description**		**Data Type**	**Initial Value**	**Visibility**
LicenseNo.	ทะเบียนรถ		String	-	Private
CarType	ประเภทรถ		CarKind	-	Private
StatusCar	สถานะของรถ		String	-	Private
StartDate	วันที่เริ่มต้น		Date	-	Private
EndDate	วันที่สิ้นสุด		Date	-	Private
Methods					
1. searchAvailableCar					
Description	ฟังก์ชันที่ใช้ค้นหารถที่มีสถานะ Available โดยค้นหาตามประเภทรถที่ลูกค้าต้องการเช่า				
Visibility	Public				
Parameters	**Name**	**Description**		**Data Type**	
	CarType	ประเภทรถที่ลูกค้าต้องการเช่า		CarKind	
Return Type	Car				
2. changeStatusCar					
Description	ฟังก์ชันที่ใช้เปลี่ยนสถานะของรถ ซึ่งแบ่งออกเป็น Available และ PickUp				
Visibility	Public				
Parameters	**Name**	**Description**		**Data Type**	
	Book	รายการการเช่าที่ต้องการเปลี่ยนสถานะรถ		Booking	
	StatusCar	สถานะรถ		String	
Return Type	-				

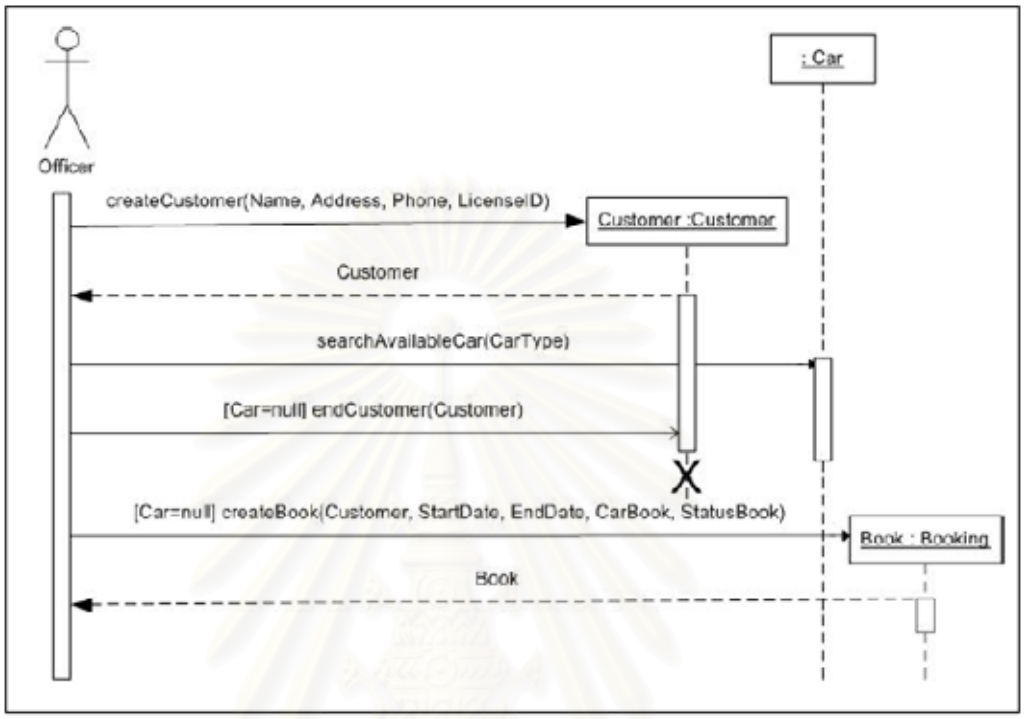
Association	
1. Register	
Description	ลูกค้าต้องการลงทะเบียนการเช่ารถ
Association Type	Composition
From - To	Customer – Booking
Cardinality	1 : 1..*
2. Assign	
Description	มอบหมายรถให้กับรายการการเช่า
Association Type	Association
From - To	Car – Booking
Cardinality	1 : 0..*
3. Search	
Description	ค้นหารถที่ลูกค้าต้องการ
Association Type	Association
From - To	Customer - Car
Cardinality	1 : 0..*

ข้อจำกัด :

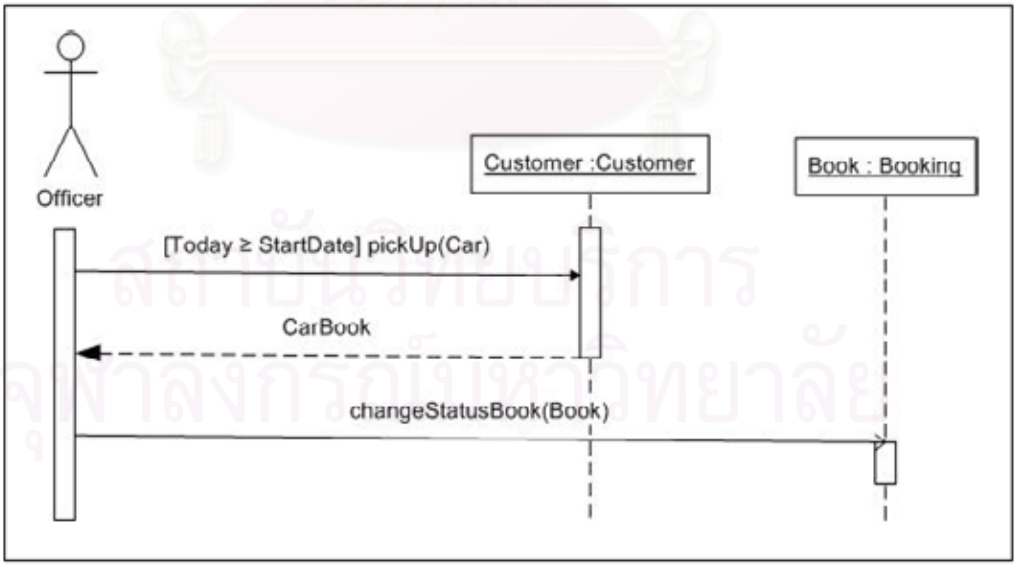
1. สามารถทำรายการการเช่ารถได้ ก็ต่อเมื่อระบบสามารถค้นหารถตามที่ลูกค้าต้องการเช่าได้
2. ลูกค้าสามารถมารับรถได้ภายในระยะเวลาที่เช่า

1.4 แผนภาพซีเควนซ์

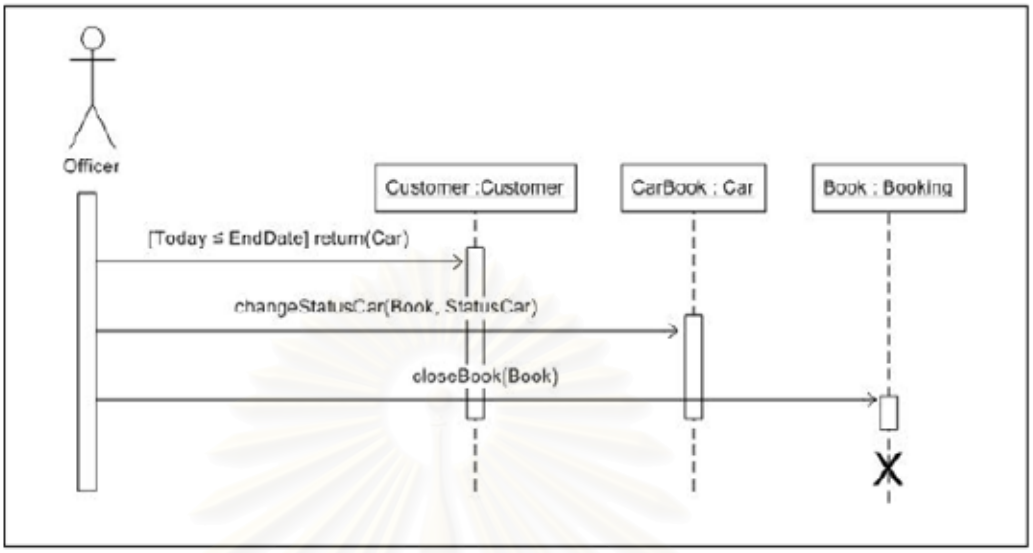
Sequence Diagram (Book)



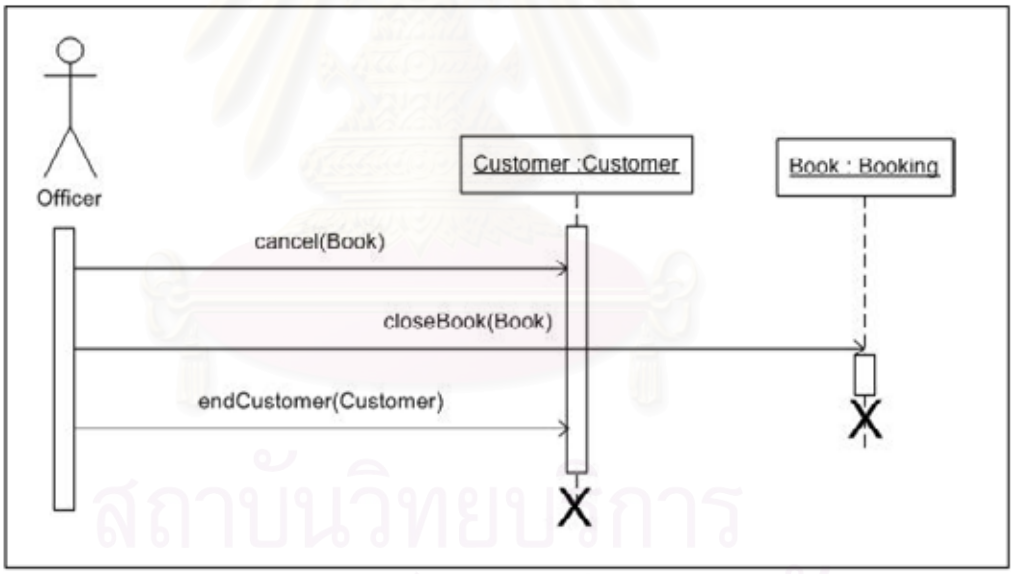
Sequence Diagram (Pick Up)



Sequence Diagram (Return)

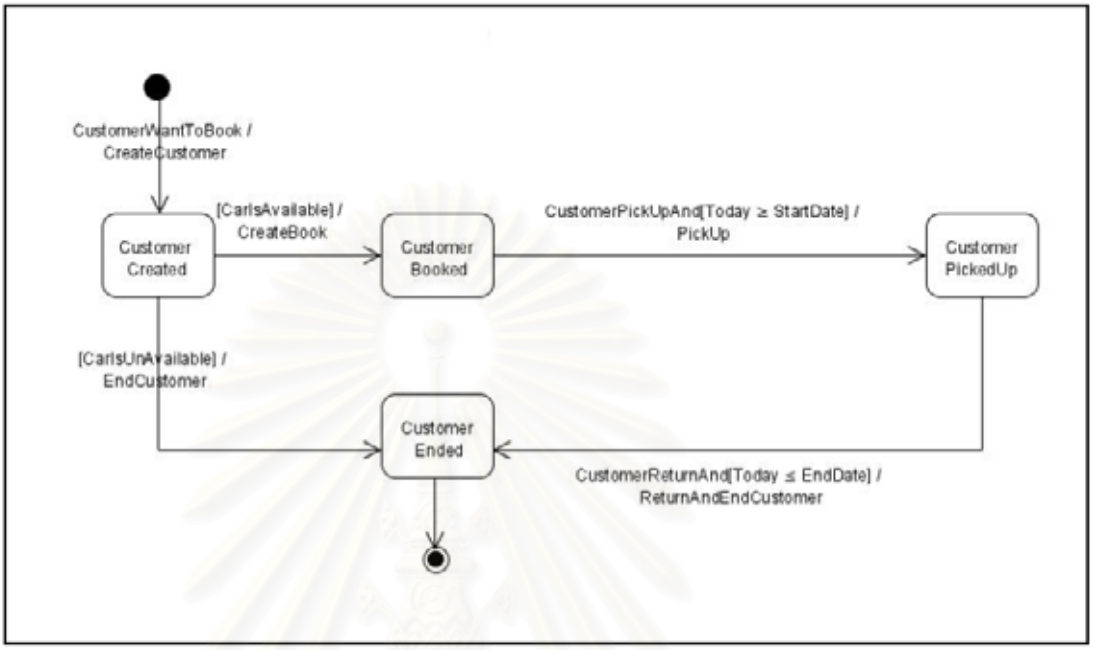


Sequence Diagram (Cancel)

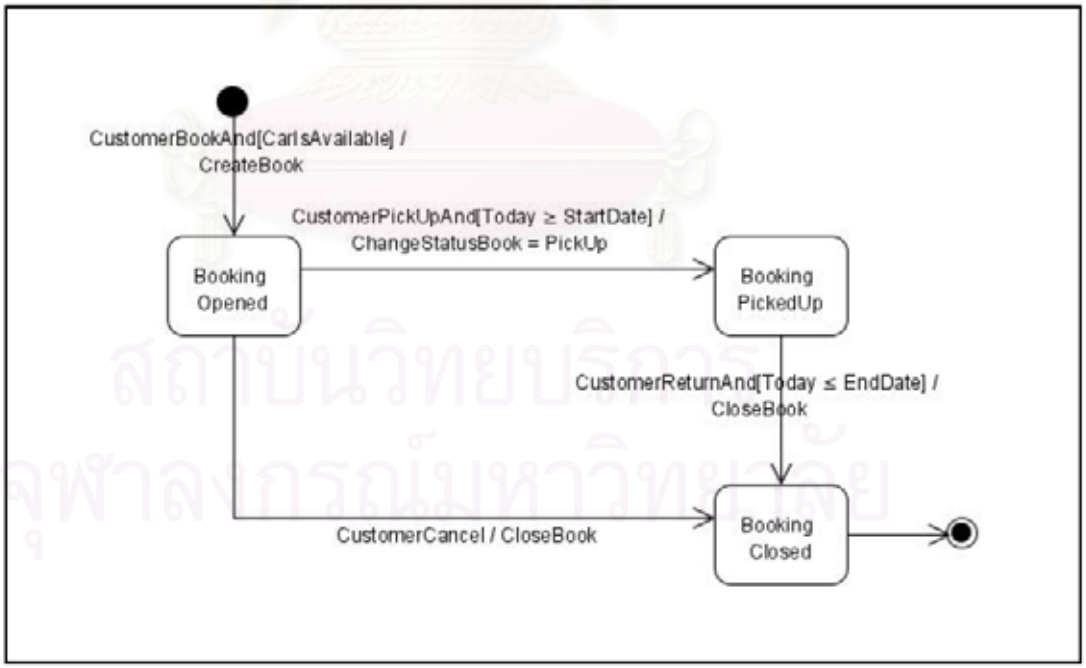


1.5 แผนภาพสถานะ

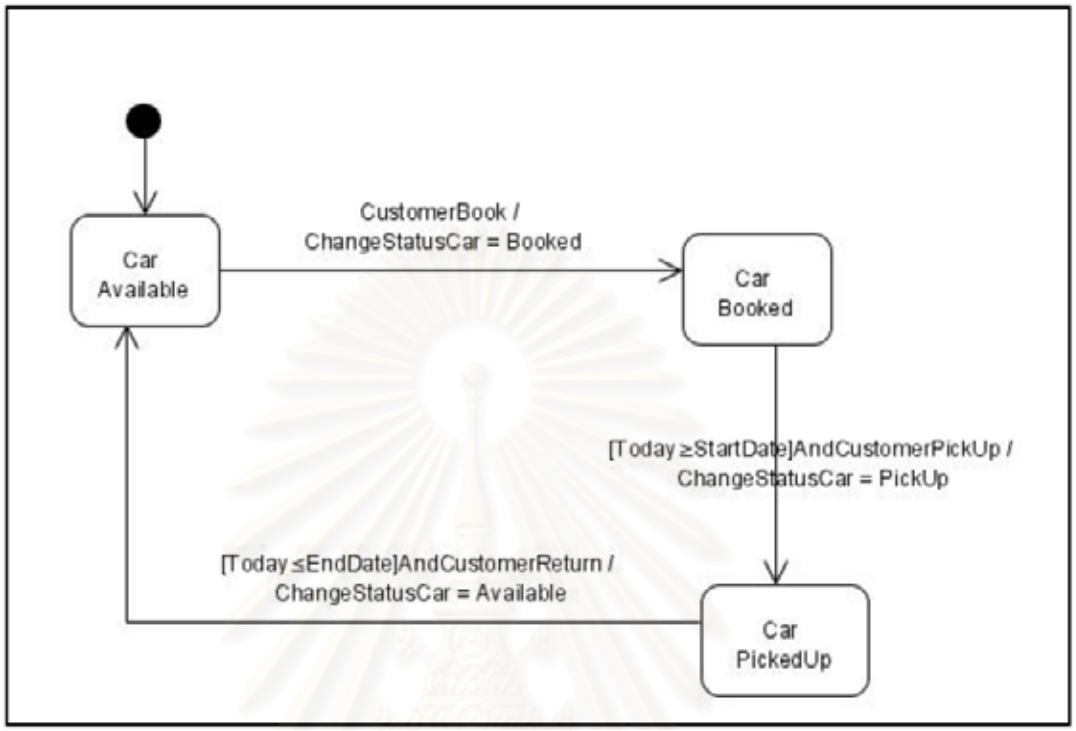
State Machine Diagram (Customer)



State Machine Diagram (Booking)



State Machine Diagram (Car)



ภาคผนวก ค
เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์

1. เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว

Reading 1 : Sequence Diagram x Class Diagram			
วัตถุประสงค์	เพื่อตรวจสอบว่า Class และความสัมพันธ์ใน Class Diagram ถูกนำเสนอได้ตรงตามที่ถูกระบุใน Sequence Diagram		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	Sequence Diagram	- *ขีดเส้นใต้*ที่ Object หรือ Class ด้วยสีน้ำเงิน และที่ Message ด้วยสีเขียว - *วงกลม* Constraint และ Condition ของ Message ด้วยสีเหลือง	
2	Sequence Diagram → Class Diagram	*ตรวจสอบ*ทุกๆ Object หรือ Class (น้ำเงิน) ใน Sequence Diagram ว่าแสดงเป็น Class ใน Class Diagram หรือไม่	มี Object หรือ Class (น้ำเงิน) ที่แสดงใน Sequence Diagram แต่ไม่ได้แสดงเป็น Class ใน Class Diagram
3	Sequence Diagram → Class Diagram	*ตรวจสอบ*ทุกๆ Message ใน Sequence Diagram (เขียว) ว่าแสดงเป็น Method ใน Class Diagram หรือไม่	มี Message ที่แสดงใน Sequence Diagram แต่ไม่ได้แสดงเป็น Method ใน Class Diagram
4	Sequence Diagram → Class Diagram	ใน Sequence Diagram สำหรับ Class ที่มีการส่ง Message ถึงกัน หรือ Message มีการเรียกใช้ Attribute ของอีก Class ให้ถือว่า Class คู่นั้นมีความสัมพันธ์กัน ดังนั้นให้*ตรวจสอบ*ว่าใน Class Diagram มีความสัมพันธ์ (Association) เชื่อมระหว่าง Class คู่นั้นหรือไม่	Class ที่มีความสัมพันธ์กันใน Sequence Diagram แต่ใน Class Diagram ไม่แสดงความสัมพันธ์ (Association) เชื่อมระหว่าง Class คู่นั้น
5	Sequence Diagram → Class Diagram	*ถ้า*ใน Sequence Diagram มีการแสดง Constraint หรือ Condition (เหลือง) ให้*ตรวจสอบ*ว่า Class Diagram แสดงไว้ด้วยหรือไม่	มี Constraint หรือ Condition (เหลือง) ใน Sequence Diagram แต่ไม่ได้แสดงไว้ใน Class Diagram
6	Class Diagram	*ตรวจสอบ*ชื่อ Class, Attribute และ Method ว่าสื่อความหมายได้ชัดเจนหรือไม่	ชื่อ Class, Attribute และ Method สื่อความหมายไม่ชัดเจน
7	Class Diagram	*ตรวจสอบ*ประเภทของความสัมพันธ์ระหว่าง Class ว่ามีความเหมาะสมหรือไม่	ใช้ประเภทของความสัมพันธ์ระหว่าง Class ไม่เหมาะสม

Reading 3 : Sequence Diagram x State Machine Diagram			
วัตถุประสงค์	ตรวจสอบการเปลี่ยน State ของ Object ใน State Machine Diagram เทียบกับการส่งและรับ Message ใน Sequence Diagram ว่าสอดคล้องกันหรือไม่ **(ตรวจสอบทีละ State Machine Diagram)**		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	State Machine Diagram	*เขียน*อักษรกำกับแต่ละ Transition Action (เขียว) โดยเขียนเรียงลำดับตามการเปลี่ยนสถานะ (A1, A2, A3…)	
2	State Machine Diagram → Sequence Diagram	*หา* Sequence Diagram ที่มีการใช้ Object ที่ State Machine Diagram นี้นำเสนอ	ไม่พบ Sequence Diagram ใดเลยที่ใช้ Object นี้
3	State Machine Diagram → Sequence Diagram	- *ค้นหา* Message ใน Sequence Diagram (เขียว) ที่สอดคล้องกับ Transition Action ใน State Machine Diagram (เขียว) เมื่อพบแล้วให้ *ทำดอกจัน* ที่ Message และที่ Transition Action คู่นั้น - *ตรวจสอบว่า* Message และ Transition Action ที่จับคู่กันนั้นแสดง Constraint และ Condition ได้ตรงกันหรือไม่	- ไม่สามารถหา Message ที่สอดคล้องกับ Transition Action ได้ - มี Message และ Transition Action ที่จับคู่กันแล้ว แสดง Constraint และ Condition ไม่ตรงกัน
4	State Machine Diagram → Sequence Diagram	*ถ้า* Message ที่ทำดอกจันไว้ในขั้นตอนก่อนหน้าปรากฏใน Sequence Diagram เดียวกันแล้ว ให้*ตรวจสอบว่า* Message ที่คู่กับ Transition Action ที่มีอักษรกำกับเป็น A1 ต้องพบก่อน Message ที่จับคู่กับ A2	Message ที่แสดงใน Sequence Diagram ไม่เรียงลำดับตามที่ Transition Action แสดงไว้

Reading 5 : Class Description x Requirement Description			
วัตถุประสงค์	เพื่อตรวจสอบว่า Class Description นำเสนอ Method ได้เหมาะสมและถูกต้องตรงตาม Requirement Description		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	Requirement Description	- *ขีดเส้นใต้ด้วยสีน้ำเงินที่คำนาม (คำนามเดียวกันไม่ต้องขีดซ้ำ)* - *ขีดเส้นใต้ด้วยสีเขียวที่คำกริยา (คำกริยาเดียวกันไม่ต้องขีดซ้ำ)* - *วงกลม*ด้วยสีเหลืองที่ Constraint และ Condition	
2	Requirement Description → Class Description	*ค้นหา* Method ใน Class Description ที่สอดคล้องกับคำกริยา (เขียว) ใน Requirement Description เมื่อพบแล้วให้*ทำดอก*จันสีเขียวที่ชื่อของ Method และที่คำกริยา	ไม่พบ Method ที่สอดคล้องกับคำกริยา
3	Requirement Description → Class Description	*ค้นหา* Class หรือ Attribute ใน Class Description ที่สอดคล้องกับคำนาม (น้ำเงิน) ใน Requirement Description เมื่อพบแล้วให้*ทำดอก*จันสีน้ำเงินที่คำนาม และที่ Class หรือ Attribute	ไม่พบ Class หรือ Attribute ที่สอดคล้องกับคำนาม
4	Requirement Description → Class Description	*ถ้า*คำนามใน Requirement Description สอดคล้องกับ Class ใน Class Description ให้*ตรวจสอบว่า* - Class (ดอกจันน้ำเงิน) มีข้อมูลที่ครบถ้วนสำหรับฟังก์ชันการทำงาน -Attribute (ดอกจันน้ำเงิน) ของ Class เหล่านี้เป็นคำนาม - Method (ดอกจันเขียว) ของ Class เป็นคำกริยา - Constraint สำหรับ Class เหล่านี้ตรงกับใน Requirement Description	- Class (ดอกจันน้ำเงิน) มีข้อมูลสำหรับฟังก์ชันการทำงานไม่ครบถ้วน - Attribute (ดอกจันน้ำเงิน) ของ Class ไม่ใช่คำนาม - Method (ดอกจันเขียว) ของ Class ไม่ใช่คำกริยา - Constraint ใน Class Description ไม่ตรงตามใน Requirement Description
5	Requirement Description → Class Description	*ถ้า*คำนามใน Requirement Description สอดคล้องกับ Attribute ใน Class Description ให้*ตรวจสอบว่า* - ประเภทของ Attribute (ดอกจันน้ำเงิน) กำหนดไว้อย่างเหมาะสม	- ประเภทของ Attribute (ดอกจันน้ำเงิน) ใน Class Description ไม่เหมาะสม
6	Requirement Description	*ตรวจสอบว่า*คำนามหรือคำกริยาทุกคำใน Requirement Description ถูกทำดอกจัน	มีคำนามหรือคำกริยาใน Requirement Description ที่ไม่ได้ถูกทำดอกจัน

Reading 6 : Sequence Diagram x Use Cases (Use Case Diagram + Use Case Description)			
วัตถุประสงค์	ตรวจสอบว่า Sequence Diagram นำเสนอ Object และ Message ได้อย่างถูกต้องสอดคล้องกับ Use Cases		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	Use Cases	- *ขีดเส้นใต้และใส่ตัวเลขกำกับ*ด้วยสีน้ำเงินไว้ที่คำนาม *(คำนามเดียวกันไม่ต้องขีดซ้ำ)* - *ขีดเส้นใต้และใส่ตัวเลขกำกับ*ด้วยสีเขียวที่คำกริยา *(คำกริยาเดียวกันไม่ต้องขีดซ้ำ)* - *วงกลม*ด้วยสีเหลืองที่ Constraint และ Condition	
2	Use Cases	*อ่านดู*ว่าในการทำแต่ละฟังก์ชันต้องใช้ข้อมูลใดบ้าง จากนั้น*เขียน* Di,j ด้วยสีแดงกำกับที่ข้อมูล โดยที่ i และ j คือตัวเลขที่กำกับอยู่บนคำนามที่ข้อมูลส่งหากัน	
3	Sequence Diagram	- *ขีดเส้นใต้*ที่ Object ด้วยสีน้ำเงิน และ*ใส่ตัวเลขกำกับ*โดยให้เลขสอดคล้องกับ Use Cases - *ขีดเส้นใต้*ด้วยสีเขียวและ*ใส่ตัวเลขกำกับ*ตามลำดับการส่ง Message - *วงกลม*ด้วยสีเหลืองที่ Constraint และ Condition	ไม่พบ Sequence Diagram ใดนำเสนอคำนามที่สอดคล้องกับคำนามที่แสดงใน Use Cases
4	Sequence Diagram	*เขียน* Di,j ด้วยสีแดงที่ข้อมูลที่ถูกส่งระหว่าง Object โดยที่ i และ j เป็นตัวเลขที่กำกับอยู่บน Object ที่ข้อมูลส่งหากัน	
5	Use Cases → Sequence Diagram	- *ตรวจสอบ*ลำดับการส่ง Message (เขียว) ใน Sequence Diagram ว่าตรงกับใน Use Cases หรือไม่ - *ตรวจสอบ*ข้อมูลของแต่ละ Message ที่ถูกส่งใน Sequence Diagram (แดง) ว่าตรงกับใน Use Cases (แดง) หรือไม่	- ลำดับของการส่ง Message (เขียว) ใน Sequence Diagram ไม่ตรงกับใน Use Cases - ข้อมูลของแต่ละ Message ที่ถูกส่งใน Sequence Diagram (แดง) ไม่ตรงกับใน Use Cases (แดง)
6	Use Cases → Sequence Diagram	*ตรวจสอบ*ว่า Constraint และ Condition (เหลือง) ใน Sequence Diagram ตรงกับใน Use Cases (เหลือง) หรือไม่	Constraint หรือ Condition (เหลือง) ใน Sequence Diagram ไม่ตรงกับใน Use Cases (เหลือง)

155

2. เอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีเดิม ที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549)

Reading 1 : Sequence Diagram x Class Diagram			
วัตถุประสงค์	เพื่อตรวจสอบว่า Class และความสัมพันธ์ใน Class Diagram ถูกนำเสนอได้ครบถ้วนตามที่ถูกระบุใน Sequence Diagram		
ขั้นตอน	**แผนภาพ**	**สิ่งที่ต้องทำ**	**ข้อบกพร่อง**
1	Sequence Diagram	- *ขีดเส้นใต้*ที่ Object หรือ Class ด้วยสีน้ำเงิน และที่ Message ด้วยสีเขียว - *วงกลม* Constraint และ Condition ของ Message ด้วยสีเหลือง	
2	Sequence Diagram → Class Diagram	*ตรวจสอบ*ทุกๆ Object หรือ Class (น้ำเงิน) ใน Sequence Diagram ว่าแสดงเป็น Class ใน Class Diagram หรือไม่	มี Object หรือ Class (น้ำเงิน) ที่แสดงใน Sequence Diagram แต่ไม่ได้แสดงเป็น Class ใน Class Diagram
3	Sequence Diagram → Class Diagram	*ตรวจสอบ*ทุกๆ Message ใน Sequence Diagram (เขียว) ว่าแสดงเป็น Method ใน Class Diagram หรือไม่	มี Message ที่แสดงใน Sequence Diagram แต่ไม่ได้แสดงเป็น Method ใน Class Diagram
4	Sequence Diagram → Class Diagram	ใน Sequence Diagram สำหรับ Class ที่มีการส่ง Message ถึงกัน หรือ Message มีการเรียกใช้ Attribute ของอีก Class ให้ถือว่า Class คู่นั้นมีความสัมพันธ์กัน ดังนั้นให้*ตรวจสอบ*ว่าใน Class Diagram มีความสัมพันธ์ (Association) เชื่อมระหว่าง Class คู่นั้นหรือไม่	Class ที่มีความสัมพันธ์กันใน Sequence Diagram แต่ใน Class Diagram ไม่แสดงความสัมพันธ์ (Association) เชื่อมระหว่าง Class คู่นั้น
5	Sequence Diagram → Class Diagram	*ถ้า*ใน Sequence Diagram มีการแสดง Constraint หรือ Condition (เหลือง) ให้*ตรวจสอบ*ว่า Class Diagram แสดงไว้ด้วยหรือไม่	มี Constraint หรือ Condition (เหลือง) ใน Sequence Diagram แต่ไม่ได้แสดงไว้ใน Class Diagram
6	Class Diagram	*ตรวจสอบ*ชื่อ Class, Attribute และ Method ว่าสื่อความหมายได้ชัดเจนหรือไม่	ชื่อ Class, Attribute และ Method สื่อความหมายไม่ชัดเจน
7	Class Diagram	*ตรวจสอบ*ประเภทของความสัมพันธ์ระหว่าง Class ว่ามีความเหมาะสมหรือไม่	ใช้ประเภทของความสัมพันธ์ระหว่าง Class ไม่เหมาะสม

Reading 2 : State Machine Diagram x Class Description			
วัตถุประสงค์	ตรวจสอบ Class ที่แสดงใน Class Description ว่าตรงกับฟังก์ชันการทำงานที่ถูกระบุไว้ใน State Machine Diagram **(ตรวจสอบทีละ State Machine Diagram)**		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	State Machine Diagram	- *ขีดเส้นใต้*ที่ชื่อ State ด้วยสีน้ำเงิน - *วงกลม*ที่ Transition Action (แสดงด้วยสัญลักษณ์ลูกศร) ด้วยสีเขียว	
2	State Machine Diagram	*ตรวจสอบ* State (น้ำเงิน) และ Transition Action (เขียว) ว่าอ่านแล้วเข้าใจหรือไม่	มี State (น้ำเงิน) หรือ Transition Action (เขียว) ที่อ่านแล้วไม่เข้าใจความหมาย
3	State Machine Diagram →Class Description	*ตรวจสอบ*ว่าทุกๆ Object ที่นำมาแสดงใน State Machine Diagram ถูกแสดงเป็น Class ใน Class Description หรือไม่	มี Object ที่นำมาแสดงใน State Machine Diagram แต่ไม่ได้แสดงเป็น Class ใน Class Description
4	State Machine Diagram →Class Description	*ถ้ามี*การกำหนดค่าที่เป็นไปได้ของ Attribute ใน Class Description ให้*ตรวจสอบ*ว่าตรงกับที่แสดงใน State Machine Diagram หรือไม่	ค่าที่เป็นไปได้ของ Attribute ใน Class Description ไม่ตรงกับที่ State Machine Diagram แสดงไว้
5	State Machine Diagram →Class Description	*ตรวจสอบว่ามี* Method ใน Class Description ที่สามารถทำให้เกิด Transition Action (เขียว) ใน State Machine Diagram หรือไม่	ไม่สามารถหา Method ใน Class Description ที่ทำให้เกิด Transition Action (เขียว)

Reading 3 : Sequence Diagram x State Machine Diagram			
วัตถุประสงค์	ตรวจสอบการเปลี่ยน State ของ Object ใน State Machine Diagram เทียบกับการส่งและรับ Message ใน Sequence Diagram ว่าสอดคล้องกันหรือไม่ **(ตรวจสอบทีละ State Machine Diagram)**		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	State Machine Diagram	*เขียน*อักษรกำกับแต่ละ Transition Action (เขียว) โดยเขียนเรียงลำดับตามการเปลี่ยนสถานะ (A1, A2, A3…)	
2	State Machine Diagram → Sequence Diagram	*หา* Sequence Diagram ที่มีการใช้ Object ที่ State Machine Diagram นี้นำเสนอ	ไม่พบ Sequence Diagram ใดเลยที่ใช้ Object นี้
3	State Machine Diagram → Sequence Diagram	- *ค้นหา* Message ใน Sequence Diagram (เขียว) ที่สอดคล้องกับ Transition Action ใน State Machine Diagram (เขียว) เมื่อพบแล้วให้ *ทำดอกจัน* ที่ Message และที่ Transition Action คู่นั้น - *ตรวจสอบว่า* Message และ Transition Action ที่จับคู่กันนั้นแสดง Constraint และ Condition ได้ตรงกันหรือไม่	- ไม่สามารถหา Message ที่สอดคล้องกับ Transition Action ได้ - มี Message และ Transition Action ที่จับคู่กันแล้ว แสดง Constraint และ Condition ไม่ตรงกัน
4	State Machine Diagram → Sequence Diagram	*ถ้า* Message ที่ทำดอกจันไว้ในขั้นตอนก่อนหน้าปรากฏใน Sequence Diagram เดียวกันแล้ว ให้*ตรวจสอบว่า* Message ที่คู่กับ Transition Action ที่มีอักษรกำกับเป็น A1 ต้องพบก่อน Message ที่จับคู่กับ A2	Message ที่แสดงใน Sequence Diagram ไม่เรียงลำดับตามที่ Transition Action แสดงไว้

Reading 4 : Class Diagram x Class Description			
วัตถุประสงค์	เพื่อตรวจสอบว่า Class Description นำเสนอข้อมูลที่จำเป็นครบถ้วนตามที่ Class Diagram แสดงไว้ และสื่อความหมายได้เข้าใจชัดเจน		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	Class Diagram → Class Description	*ตรวจสอบว่า* Class ที่แสดงใน Class Diagram ได้ถูกอธิบายไว้ใน Class Description หรือไม่	มี Class ที่ปรากฏใน Class Diagram แต่ไม่ได้อธิบายไว้ใน Class Description
2	Class Description	*ตรวจสอบชื่อ* Class และคำอธิบาย Class ว่าอ่านแล้วเข้าใจหรือไม่	ชื่อ Class หรือคำอธิบาย Class ไม่ชัดเจน
3	Class Diagram	- *ตรวจสอบประเภทของ* Attribute ที่ใช้ว่ามีความเหมาะสมหรือไม่ - *ตรวจสอบว่า* Attribute ที่แสดงไว้เพียงพอต่อการทำงานของ Method หรือไม่ - *ตรวจสอบว่า* Constraint สื่อความหมายได้เข้าใจหรือไม่	- ใช้ประเภทของ Attribute ไม่เหมาะสม - Attribute ที่แสดงไว้ไม่เพียงพอต่อการทำงานของ Method - Constraint สื่อความหมายไม่เข้าใจ
4	Class Diagram → Class Description	*ตรวจสอบว่า* Attribute, Method และ Constraint ที่แสดงใน Class Description ตรงกับใน Class Diagram หรือไม่	มี Attribute, Method หรือ Constraint ที่แสดงใน Class Description แล้วไม่ตรงกับใน Class Diagram
5	Class Diagram → Class Description	*ถ้า* Class Diagram มีการ Inherit ให้*ตรวจสอบว่า*ใน Class Description ได้อธิบายไว้อย่างถูกต้อง และ*ตรวจสอบชื่อของ* Class ที่มีการ Inherit ว่าสื่อความหมายได้เข้าใจ	Class Diagram มีการ Inherit แต่ใน Class Description ไม่ได้อธิบายไว้ หรือชื่อของ Class ที่มีการ Inherit สื่อความหมายไม่ชัดเจน
6	Class Diagram	- *ตรวจสอบประเภทของความสัมพันธ์ว่าเหมาะสมหรือไม่* - *ตรวจสอบชื่อของความสัมพันธ์ว่าสื่อออกมาได้เข้าใจหรือไม่* - *ตรวจสอบ* Cardinality ว่ากำหนดไว้อย่างเหมาะสมหรือไม่	- ใช้ประเภทของความสัมพันธ์ไม่เหมาะสม - ชื่อความสัมพันธ์ไม่สื่อความหมาย - กำหนด Cardinality ไม่เหมาะสม
7	Class Diagram → Class Description	*ตรวจสอบประเภทของความสัมพันธ์* และ Cardinality ใน Class Description ว่าตรงกับใน Class Diagram หรือไม่	ประเภทของความสัมพันธ์ หรือ Cardinality ใน Class Description ไม่ตรงกับใน Class Diagram
8	Class Description → Class Diagram	*ตรวจสอบว่า*ทุก Class ที่อธิบายไว้ใน Class Description นั้นปรากฏอยู่ใน Class Diagram	มี Class ที่อธิบายไว้ใน Class Description แต่ไม่ได้แสดงไว้ใน Class Diagram

Reading 5 : Class Description x Requirement Description			
วัตถุประสงค์	เพื่อตรวจสอบว่า Class Description นำเสนอ Method ได้เหมาะสมและถูกต้องตรงตาม Requirement Description		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	Requirement Description	- *ขีดเส้นใต้ด้วยสีน้ำเงินที่คำนาม (คำนามเดียวกันไม่ต้องขีดซ้ำ)* - *ขีดเส้นใต้ด้วยสีเขียวที่คำกริยา (คำกริยาเดียวกันไม่ต้องขีดซ้ำ)* - *วงกลม*ด้วยสีเหลืองที่ Constraint และ Condition	
2	Requirement Description → Class Description	*ค้นหา* Method ใน Class Description ที่สอดคล้องกับคำกริยา (เขียว) ใน Requirement Description เมื่อพบแล้วให้*ทำดอก*จันสีเขียวที่ชื่อของ Method และที่คำกริยา	ไม่พบ Method ที่สอดคล้องกับคำกริยา
3	Requirement Description → Class Description	*ค้นหา* Class หรือ Attribute ใน Class Description ที่สอดคล้องกับคำนาม (น้ำเงิน) ใน Requirement Description เมื่อพบแล้วให้*ทำดอก*จันสีน้ำเงินที่คำนาม และที่ Class หรือ Attribute	ไม่พบ Class หรือ Attribute ที่สอดคล้องกับคำนาม
4	Requirement Description → Class Description	*ถ้า*คำนามใน Requirement Description สอดคล้องกับ Class ใน Class Description ให้*ตรวจสอบว่า* - Class (ดอกจันน้ำเงิน) มีข้อมูลที่ครบถ้วนสำหรับฟังก์ชันการทำงาน -Attribute (ดอกจันน้ำเงิน) ของ Class เหล่านี้เป็นคำนาม - Method (ดอกจันเขียว) ของ Class เป็นคำกริยา - Constraint สำหรับ Class เหล่านี้ตรงกับใน Requirement Description	- Class (ดอกจันน้ำเงิน) มีข้อมูลสำหรับฟังก์ชันการทำงานไม่ครบถ้วน - Attribute (ดอกจันน้ำเงิน) ของ Class ไม่ใช่คำนาม - Method (ดอกจันเขียว) ของ Class ไม่ใช่คำกริยา - Constraint ใน Class Description ไม่ตรงตามใน Requirement Description
5	Requirement Description → Class Description	*ถ้า*คำนามใน Requirement Description สอดคล้องกับ Attribute ใน Class Description ให้*ตรวจสอบว่า* - ประเภทของ Attribute (ดอกจันน้ำเงิน) กำหนดไว้อย่างเหมาะสม	- ประเภทของ Attribute (ดอกจันน้ำเงิน) ใน Class Description ไม่เหมาะสม
6	Requirement Description	*ตรวจสอบว่า*คำนามหรือคำกริยาทุกคำใน Requirement Description ถูกทำดอกจัน	มีคำนามหรือคำกริยาใน Requirement Description ที่ไม่ได้ถูกทำดอกจัน

Reading 6 : Sequence Diagram x Use Cases (Use Case Diagram + Use Case Description)			
วัตถุประสงค์	ตรวจสอบว่า Sequence Diagram นำเสนอ Object และ Message ได้อย่างถูกต้องสอดคล้องกับ Use Cases		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	Use Cases	- *ขีดเส้นใต้และใส่ตัวเลขกำกับ*ด้วยสีน้ำเงินไว้ที่คำนาม *(คำนามเดียวกันไม่ต้องขีดซ้ำ)* - *ขีดเส้นใต้และใส่ตัวเลขกำกับ*ด้วยสีเขียวที่คำกริยา *(คำกริยาเดียวกันไม่ต้องขีดซ้ำ)* - *วงกลม*ด้วยสีเหลืองที่ Constraint และ Condition	
2	Use Cases	*อ่านดู*ว่าในการทำแต่ละฟังก์ชันต้องใช้ข้อมูลใดบ้าง จากนั้น*เขียน* Di,j ด้วยสีแดงกำกับที่ข้อมูล โดยที่ i และ j คือตัวเลขที่กำกับอยู่บนคำนามที่ข้อมูลส่งหากัน	
3	Sequence Diagram	- *ขีดเส้นใต้*ที่ Object ด้วยสีน้ำเงิน และ*ใส่ตัวเลขกำกับ*โดยให้เลขสอดคล้องกับ Use Cases - *ขีดเส้นใต้*ด้วยสีเขียวและ*ใส่ตัวเลขกำกับ*ตามลำดับการส่ง Message - *วงกลม*ด้วยสีเหลืองที่ Constraint และ Condition	ไม่พบ Sequence Diagram ใดนำเสนอคำนามที่สอดคล้องกับคำนามที่แสดงใน Use Cases
4	Sequence Diagram	*เขียน* Di,j ด้วยสีแดงที่ข้อมูลที่ถูกส่งระหว่าง Object โดยที่ i และ j เป็นตัวเลขที่กำกับอยู่บน Object ที่ข้อมูลส่งหากัน	
5	Use Cases → Sequence Diagram	- *ตรวจสอบ*ลำดับการส่ง Message (เขียว) ใน Sequence Diagram ว่าตรงกับใน Use Cases หรือไม่ - *ตรวจสอบ*ข้อมูลของแต่ละ Message ที่ถูกส่งใน Sequence Diagram (แดง) ว่าตรงกับใน Use Cases (แดง) หรือไม่	- ลำดับของการส่ง Message (เขียว) ใน Sequence Diagram ไม่ตรงกับใน Use Cases - ข้อมูลของแต่ละ Message ที่ถูกส่งใน Sequence Diagram (แดง) ไม่ตรงกับใน Use Cases (แดง)
6	Use Cases → Sequence Diagram	*ตรวจสอบ*ว่า Constraint และ Condition (เหลือง) ใน Sequence Diagram ตรงกับใน Use Cases (เหลือง) หรือไม่	Constraint หรือ Condition (เหลือง) ใน Sequence Diagram ไม่ตรงกับใน Use Cases (เหลือง)

161

Reading 7 : State Machine Diagram x Requirement Description and Use Cases			
วัตถุประสงค์	ตรวจสอบ State Machine Diagram ว่าอธิบาย State และ Event ที่ทำให้เกิดการเปลี่ยน State ได้สอดคล้องกับใน Requirement Description และ Use Cases **(ตรวจสอบทีละ State Machine Diagram)**		
ขั้นตอน	**แผนภาพ**	**สิ่งที่ต้องทำ**	**ข้อบกพร่อง**
1.	Requirement Description	*วงกลม*ด้วยสีแดงที่ฟังก์ชันการทำงานของระบบ	
2.	Requirement Description	*ขีดเส้นใต้*ด้วยดินสอและใส่หมายเลขกำกับที่คำอธิบาย State ทั้งหมดของแต่ละ Object	
3.	Requirement Description	*สร้าง Matrix* ขนาด NxN (Adjacency Matrix) โดยที่ N คือจำนวน State โดยที่กำหนดให้แนวนอนเป็น State ต้นทาง และแนวตั้งเป็น State ปลายทาง	
4.	Requirement Description	- *ถ้า*สามารถกำหนด Event ที่เป็นสาเหตุให้เปลี่ยน State ระหว่างคู่ใดได้ให้*ใส่ Event* ไว้ที่ช่องนั้น (*ถ้า* Event นั้นมี Constraint หรือ Condition อธิบายไว้ใน Requirement Description ให้*เขียน*ลงใน Adjacency Matrix ด้วย) - *ถ้า*คิดว่าเป็นไปไม่ได้ที่จะเปลี่ยน State ระหว่าง State คู่นั้นให้*ใส่เครื่องหมายกากบาท* - *ถ้า*ไม่แน่ใจให้ปล่อยช่องนั้นว่างไว้	
5.	State Machine Diagram → Use Cases	*ค้นหา* Object ใน Use Cases ที่เกี่ยวข้องกับ State Machine Diagram นี้	ไม่พบ Object ที่เกี่ยวข้องกับ State Machine Diagram นี้
6.	Use Cases	- *สำหรับช่องที่กากบาท และช่องว่าง*ใน Adjacency Matrix ให้ดูจาก Use Cases ว่า Event ใดที่เป็นสาเหตุให้ State เปลี่ยน *ถ้าพบให้เขียน* Event นั้นลงไปแทน แต่*ถ้าไม่พบให้กากบาท*ลงไปในช่องว่าง	

7.	Requirement Description → State Machine Diagram	*ค้นหา* State ใน State Machine Diagram ที่สอดคล้องกับ State ที่ใส่ตัวเลขกำกับไว้ใน Requirement Description *เมื่อพบแล้วให้เขียนตัวเลขด้วยดินสอกำกับโดยใช้ตัวเลขเดียวกับใน* Requirement Description	State ที่แสดงใน State Machine Diagram ไม่สอดคล้องกับ State ที่แสดงใน Requirement Description
8.	Adjacency Matrix → State Machine Diagram	- *ตรวจสอบว่า* Event ทั้งหมดใน Adjacency Matrix ได้ถูกนำเสนออย่างครบถ้วนใน State Machine Diagram - *ตรวจสอบ* Constraint ใน State Machine Diagram ตรงกับใน Adjacency Matrix หรือไม่	- มี Event ใน Adjacency Matrix ที่ไม่ได้ถูกนำเสนอใน State Machine Diagram - Constraint ใน State Machine Diagram ไม่ตรงกับใน Adjacency Matrix
9.	State Machine Diagram → Adjacency Matrix	- *ตรวจสอบว่า* Event ใน State Machine Diagram ทั้งหมดได้ถูกนำเสนออย่างครบถ้วนใน Adjacency Matrix	- มี Event ใน State Machine Diagram ที่ไม่ได้แสดงใน Adjacency Matrix

3. เอกสารเช็กลิสต์ (Checklist) ที่ใช้ในกาตรวจสอบด้วยเทคนิคซีบีอาร์ ที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549)

Checklist ใช้สำหรับตรวจสอบเพื่อหาข้อบกพร่องใน Use Case, Class Diagram, Sequence Diagram และ State Machine Diagram โดยตอบคำถามที่เกี่ยวข้องกับแผนภาพเหล่านี้ เมื่อพบข้อบกพร่องให้บันทึกลงในเอกสารบันทึกข้อบกพร่อง		
Use Case (Use Case Diagram + Use Case Description)		
1.	ตรวจสอบ Use Case Diagram ว่าสัญลักษณ์ที่ใช้ในการนำเสนอการทำงานของระบบถูกต้องหรือไม่	☐ ใช่ ☐ ไม่ใช่
2.	ตรวจสอบ Use Case Diagram ว่านำเสนอการทำงานของระบบได้ครบถ้วน และถูกต้องตรงตาม Requirement Description หรือไม่	☐ ใช่ ☐ ไม่ใช่
3.	ตรวจสอบ Use Case Description ว่ามีการอธิบายวัตถุประสงค์และ Actor ของแต่ละ Use Case หรือไม่ และอธิบายได้อย่างถูกต้องหรือไม่	☐ ใช่ ☐ ไม่ใช่
4.	ตรวจสอบ Use Case Description ว่ามีการนำเสนอ Precondition และ Postcondition ของแต่ละ Use Case หรือไม่ และนำเสนอได้อย่างถูกต้องหรือไม่	☐ ใช่ ☐ ไม่ใช่
5.	ตรวจสอบ Use Case Description ว่ามีการอธิบายขั้นตอนการทำงานของแต่ละ Use Case ได้ครบถ้วนและถูกต้องหรือไม่	☐ ใช่ ☐ ไม่ใช่
Class Diagram + Class Description		
1.	ตรวจสอบ Class Diagram ว่าแสดง Class ครบถ้วน ชื่อ Class และชื่อความสัมพันธ์ (Association) สื่อความหมายชัดเจน จากนั้นตรวจสอบประเภทความสัมพันธ์ (Association) ที่ใช้ และ Cardinality ที่กำหนดว่ามีความเหมาะสมหรือไม่	☐ ใช่ ☐ ไม่ใช่
2.	ตรวจสอบ Class Diagram ว่าแสดง Attribute ไว้ครบถ้วน ชื่อ Attribute สื่อความหมายชัดเจน และกำหนดประเภทของ Attribute กับการเข้าถึง (Visibility) Attribute ไว้อย่างเหมาะสมหรือไม่	☐ ใช่ ☐ ไม่ใช่
3.	ตรวจสอบ Class Diagram ว่าแสดง Method ครบถ้วนตรงตาม Requirement Description หรือไม่ จากนั้นตรวจสอบชื่อ Method ว่าสื่อความหมาย และกำหนด Signature กับ Return Type ของ Method ได้อย่างถูกต้อง	☐ ใช่ ☐ ไม่ใช่
4.	ถ้าใน Requirement Description มีการแสดงเงื่อนไขหรือข้อจำกัดของระบบ ให้ตรวจสอบว่าใน Class Diagram มีการแสดง Constraint หรือ Condition เหล่านั้นไว้อย่างครบถ้วน และถูกต้อง	☐ ใช่ ☐ ไม่ใช่
5.	ตรวจสอบว่า Class Description มีการอธิบายครบถ้วนตามที่ Class Diagram แสดงไว้หรือไม่	☐ ใช่ ☐ ไม่ใช่
Sequence Diagram		
1.	ตรวจสอบสัญลักษณ์ที่ใช้ในการโต้ตอบระหว่าง Object ของแต่ละแผนภาพว่ามีความถูกต้องหรือไม่	☐ ใช่ ☐ ไม่ใช่
2.	ถ้าใน Requirement Description มีการแสดงเงื่อนไขหรือข้อจำกัดของระบบ ให้ตรวจสอบว่าใน Sequence Diagram มีการแสดง Constraint หรือ Condition เหล่านั้นไว้อย่างครบถ้วน และถูกต้องหรือไม่	☐ ใช่ ☐ ไม่ใช่
3.	ตรวจสอบว่า Message ถูกแลกเปลี่ยนระหว่าง Class ที่ถูกต้อง จากนั้นตรวจสอบข้อมูลที่ใช้ในการแลกเปลี่ยน Message และการคืนค่า (Return) ว่าถูกต้องหรือไม่	☐ ใช่ ☐ ไม่ใช่
4.	ตรวจสอบว่าทุกๆ Method ที่แสดงใน Class Diagram ถูกแสดงเป็น Message ใน Sequence Diagram หรือไม่	☐ ใช่ ☐ ไม่ใช่
5.	ตรวจสอบลำดับการแลกเปลี่ยน Message ว่าสอดคล้องกับขั้นตอนการทำงานของแต่ละ Use Case ที่แสดงใน Use Case Description หรือไม่	☐ ใช่ ☐ ไม่ใช่
State Machine Diagram		
1.	ตรวจสอบสัญลักษณ์ที่ใช้ในการอธิบายการเปลี่ยนแปลงสถานะของ Object ว่าถูกต้องหรือไม่	☐ ใช่ ☐ ไม่ใช่
2.	แต่ละ State Machine Diagram นำเสนอได้อย่างชัดเจนว่าเป็นการอธิบายสถานะของ Object ใด	☐ ใช่ ☐ ไม่ใช่
3.	ตรวจสอบ State, Event และ Transition Action ว่าครบถ้วนหรือไม่ และอ่านแล้วเข้าใจหรือไม่	☐ ใช่ ☐ ไม่ใช่
4.	ตรวจสอบว่าทุก Object ที่นำมาแสดงใน State Machine Diagram ถูกแสดงเป็น Class และ Transition Action ได้ถูกแสดงเป็น Method ใน Class Diagram	☐ ใช่ ☐ ไม่ใช่
5.	ตรวจสอบว่าการเปลี่ยนสถานะของแต่ละ Object นั้นสอดคล้องกับลำดับการแลกเปลี่ยน Message ใน Sequence Diagram	☐ ใช่ ☐ ไม่ใช่

ภาคผนวก ง

เอกสารบันทึกรายการข้อบกพร่อง

1. เอกสารบันทึกรายการข้อบกพร่อง ที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549)

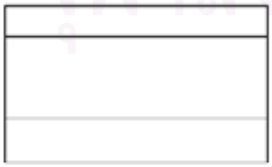
เอกสารบันทึกรายการข้อบกพร่อง

ข้อ	Reading / ข้อ	ชื่อเอกสาร	ตำแหน่งที่พบ	คำอธิบาย
1.				
2.				
3.				
4.				
5.				
6.				
7.				
8.				
9.				
10.				

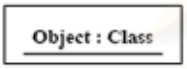
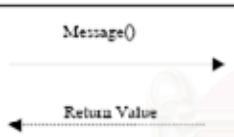
ภาคผนวก จ
เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล

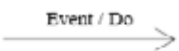
เอกสารนี้เป็นเอกสารที่ใช้ในงานวิจัยของ อุมาพร นิลเอวะ (2549) ประกอบด้วยสัญลักษณ์ของแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ จัดทำขึ้นเพื่อให้หน่วยทดลองสามารถอ่านเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุให้เข้าใจมากยิ่งขึ้น

Use Case Diagram	
ใช้เพื่อช่วยในการจำลองความต้องการของผู้ใช้ เพื่อให้ผู้พัฒนาทราบถึงความสามารถของระบบว่าต้องทำอะไรได้บ้าง ทราบถึงผู้ใช้งานในแต่ละส่วนของระบบ	
	**Actor** คือบุคคลหรือระบบภายนอกที่มีส่วนเกี่ยวข้องกับระบบที่พัฒนา
	**Use Case** เป็นฟังก์ชันการทำงานต่างๆที่ซอฟต์แวร์ทำได้
	ขอบเขตระบบ เป็นการแสดงขอบเขตการทำงานของระบบ ซึ่งข้างในขอบเขตนั้นจะประกอบด้วย Use Case ต่างๆของระบบ
	**Generalization Relationship** เป็นความสัมพันธ์ที่แสดงการสืบทอด (Inherit) ซึ่งอาจเกิดระหว่าง Actor หรือระหว่าง Use Case
	**Extend Relationship** เป็นความสัมพันธ์ระหว่าง Use Case กล่าวคือทำ Use Case A แล้วอาจทำ Use Case B ต่อหรือไม่ทำก็ได้
	**Include Relationship** เป็นความสัมพันธ์ระหว่าง Use Case กล่าวคือ ถ้าจะทำ Use Case A ต้องทำ Use Case B ด้วยเสมอ
	Association Relationship เป็นความสัมพันธ์ระหว่าง Use Case กับActor

Class Diagram	
แสดงโครงสร้างของระบบซึ่งประกอบไปด้วย Class และความสัมพันธ์ระหว่าง Class เหล่านั้น	
	**Class** คือกลุ่มของ Object โดยจะแสดง 1. ชื่อ Class 2. Attribute (แสดง Visibility, ชื่อ, ประเภท) 3. Method (แสดง Visibility, ชื่อ, Parameter, ประเภทของ Parameter, Return Type) หมายเหตุ : Visibility คือการเข้าถึงแบ่งเป็น Private (-) , Public (+) , Protected (#)
	Dependency Relationship ความสัมพันธ์แบบนี้เกิดขึ้นเมื่อการเปลี่ยนแปลงที่เกิดขึ้นกับ Class ที่ถูกพึ่งพิง (Independent Class) จะส่งผลต่อ Class ที่พึ่งพิง (Dependent Class) Class ดังกล่าว

	Generalization Relationship คือความสัมพันธ์ระหว่าง Superclass และ Subclass เป็นการ Inherit คุณสมบัติจาก Superclass นั่นเอง
	Association เป็นความสัมพันธ์แบบทั่วไป โดยปกติความสัมพันธ์แบบนี้จะเป็นความสัมพันธ์แบบ 2 ทิศทาง
	Aggregation เป็นการกำหนดส่วนประกอบแบบไม่จำเป็น
	Composition เป็นการกำหนดส่วนประกอบแบบที่มีความจำเป็น

Sequence Diagram	
แสดงให้เห็นว่าในแต่ละ Use Case นั้น แต่ละ Object ติดต่อสื่อสารกันอย่างไร มีขั้นตอนการทำงานอย่างไร โดยเน้นที่แกนเวลาเป็นสำคัญ	
	Actor คือบุคคลหรือระบบภายนอกที่มีการโต้ตอบกับระบบ โดยการรับส่ง Message
	Object ภายในบรรจุชื่อ Object ตามด้วยเครื่องหมายมหัพภาคคู่และชื่อ Class
	Lifeline แสดงถึงชีวิตของ Object
	Activation ใช้แสดงช่วงเวลาที่ Object กำลังส่งและรับ Message
	Message เป็นการส่งและคืนข้อมูล
X	Object Destruction เป็นการใช้กากบาทวางไว้ที่ปลาย Lifeline ของ Object เพื่อแสดงว่า Object นั้นสิ้นสุดแล้ว

State Machine Diagram	
เป็นการแสดงพฤติกรรมของ Class ต่างๆในระบบว่ามีสถานะอะไรบ้าง จะเปลี่ยนสถานะเมื่อเกิดเหตุการณ์อะไร	
	สถานะ (State) คือสถานะของ Object
	สถานะเริ่มต้น (Initial State) เป็นจุดเริ่มต้นสถานะของ Object
	สถานะสิ้นสุด (Final State) เป็นจุดสิ้นสุดการกระทำของ Object
	Transition Action คือเส้นลูกศรเส้นทึบที่ชี้จากสถานะหนึ่งไปยังอีกสถานะหนึ่ง โดยมี Event แสดงบนเส้น เพื่อบอกว่าเมื่อเกิดเหตุการณ์หนึ่งจะทำให้สถานะหนึ่งเปลี่ยนไปอีกสถานะหนึ่ง

ภาคผนวก ฉ
ผลการตรวจสอบซอฟต์แวร์ของผู้ตรวจสอบที่ใช้เทคนิคโอโออาร์ทีที่ปรับปรุงแล้ว ในการตรวจสอบเอกสารการออกแบบซอฟต์แวร์

ลำดับที่ของผู้ตรวจสอบ	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
1	10	85	0.118	50
2	4	90	0.044	20
3	6	90	0.067	30
4	6	90	0.067	30
5	7	90	0.078	35
6	3	90	0.033	15
7	4	90	0.044	20
8	3	90	0.033	15
9	4	90	0.044	20
10	11	90	0.122	55
11	7	75	0.093	35
12	6	80	0.075	30
13	6	70	0.086	30
14	6	70	0.086	30
15	6	70	0.086	30
16	8	85	0.094	40
17	7	85	0.082	35
18	6	90	0.067	30
19	8	90	0.089	40
20	5	90	0.056	25
21	7	90	0.078	35

ลำดับที่ของ ผู้ตรวจสอบ	จำนวน ข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
22	10	90	0.111	50
23	9	90	0.1	45
24	10	90	0.111	50

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวปรียาภรณ์ บุญพยนต์ เกิดวันที่ 16 กุมภาพันธ์ พ.ศ. 2526 สำเร็จการศึกษาวิทยาศาสตบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ จากภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีพ.ศ. 2548 จากนั้นได้เข้าศึกษาต่อในระดับปริญญาโท ในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย