การเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยวิธีการเชิงวัตถุ

นางสาวอุมาพร นิลเอวะ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549
ISBN 974-14-2057-9

AN EXPERIMENTAL COMPARISON OF SOFTWARE READING TECHNIQUES FOR OBJECT-ORIENTED DESIGN DOCUMENT INSPECTION

Miss Umaporn Ninava

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 2006

ISBN 974-14-2057-9

หัวข้อวิทยานิพนธ์	การเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ออกแบบโดยวิธีการเชิงวัตถุ
โดย	นางสาวอุมาพร นิลเอวะ
สาขาวิชา	การพัฒนาซอฟต์แวร์ด้านธุรกิจ
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. อัษฎาพร ทรัพย์สมบูรณ์

คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะพาณิชยศาสตร์และการบัญชี
(ผู้ช่วยศาสตราจารย์ ดร. ดนุชา คุณพนิชกิจ)

คณะกรรมการสอบวิทยานิพนธ์

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

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

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. ถาวร อานุภาพไตรรงค์)

อุมาพร นิลเอวะ : การเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ออกแบบโดยวิธีการเชิงวัตถุ. (An Experimental Comparison of Software Reading Techniques for Object-Oriented Design Document Inspection) อ. ที่ปรึกษา : ผศ.ดร. อัษฎาพร ทรัพย์สมบูรณ์, 213 หน้า.
ISBN 974-14-2057-9.

วัตถุประสงค์ของการวิจัยนี้คือ เพื่อเปรียบเทียบ (1) ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ (2) ประสิทธิผลในการตรวจสอบซอฟต์แวร์ และ (3) ประสิทธิผลในการกำจัดผลบวกปลอม ของเทคนิคการอ่านซอฟต์แวร์ที่ต่างกันในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ออกแบบโดยวิธีการเชิงวัตถุ 2 เทคนิค คือ เทคนิคซีบีอาร์และเทคนิคโอโออาร์ที เอกสารแสดงการออกแบบซอฟต์แวร์ที่ใช้ในการวิจัยนี้เป็นแผนภาพยูเอ็มแอล ซึ่งประกอบด้วย แผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ การตรวจสอบซอฟต์แวร์จะกระทำเฉพาะขั้นตอนการประชุมแนะนำ การจัดเตรียม และการประชุมตรวจสอบเท่านั้น

งานวิจัยนี้เป็นงานวิจัยเชิงทดลองในห้องปฏิบัติการ ซึ่งหน่วยทดลองที่นำมาใช้เพื่อตอบวัตถุประสงค์ของงานวิจัยนี้คือ นิสิตปริญญามหาบัณฑิต ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการออกแบบและพัฒนาซอฟต์แวร์เชิงวัตถุ และจบการศึกษาด้านคอมพิวเตอร์ในระดับปริญญาบัณฑิต โดยมีหน่วยกิตในรายวิชาด้านคอมพิวเตอร์ไม่ต่ำกว่า 35 หน่วยกิต สำหรับขั้นตอนการจัดเตรียมจะประกอบด้วยหน่วยทดลองทั้งหมด 48 คน และในขั้นตอนการประชุมตรวจสอบผู้วิจัยจัดกลุ่มทดลองเป็นกลุ่มละ 3 คน ทำให้ได้กลุ่มทดลองในขั้นตอนการประชุมตรวจสอบทั้งหมด 16 กลุ่ม

ผลการวิจัยพบว่า การนำเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยวิธีการเชิงวัตถุ ไม่ได้มีประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์สูงกว่าการนำเทคนิคซีบีอาร์มาใช้ แต่ทั้งสองเทคนิคให้ค่าประสิทธิผลในการกำจัดผลบวกปลอมไม่แตกต่างกัน

ภาควิชา.....สถิติ..... ลายมือชื่อนิสิต.....อุมาพร นิลเอวะ.....
สาขาวิชา...การพัฒนาซอฟต์แวร์ด้านธุรกิจ... ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา.....2549.....

4782478626 : MAJOR BUSINESS SOFTWARE DEVELOPMENT
KEY WORD: SOFTWARE READING TECHNIQUES / SOFTWARE INSPECTION / UML
UMAPORN NINAVA : AN EXPERIMENTAL COMPARISON OF SOFTWARE READING TECHNIQUES FOR OBJECT-ORIENTED DESIGN DOCUMENT INSPECTION. THESIS ADVISOR : ASST. PROF. ASSADAPORN SAPSOMBOON, 213 pp. ISBN 974-14-2057-9.

The purpose of this research is to compare the efficiency and the effectiveness of Object-Oriented design document inspection and the effectiveness of false positive removal between 2 software reading techniques for Object-Oriented design document inspection: Object-Oriented Reading Techniques (OORT) and Checklist-Based Reading (CBR). A design document in this research is a set of UML diagrams consisting of Use Case Diagram, Class Diagram, Sequence Diagram and State Machine Diagram. Inspection processes studied in this research include only three stages: Overview, Preparation and Inspection Meeting.

This research is an experimental research in a laboratory. Subjects are students in Master Degree already taken a class in Object-Oriented design and holding a Bachelor Degree related to computer science. There are 48 subjects in the Preparation process, and are divided into groups of 3 in the Inspection Meeting process. Therefore, there are 16 groups in the Inspection meeting process.

The results indicate that the Object-Oriented Reading Techniques (OORT) is not more effective or efficient than the Checklist-Based Reading (CBR) in software inspection of Object-Oriented design document. However, there is no difference in the effectiveness of false positive removal between Object-Oriented Reading Techniques and Checklist-Based Reading.

Department.....Statistics..... Student's Signature.....Umaporn Ninava.....
Field of Study ...Business Software Development... Advisor's Signature.....Assadaporn Sapsomboon.....
Academic Year .....2006.....

กิตติกรรมประกาศ

ผู้วิจัยขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร. อัษฎาพร ทรัพย์สมบูรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาสละเวลาในการชี้แนะแนวทางต่างๆให้กับผู้วิจัยจนสำเร็จเป็นวิทยานิพนธ์ฉบับนี้ และขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร. ชัชพงศ์ ตั้งมณี ประธานกรรมการวิทยานิพนธ์ อาจารย์ ดร. จันทร์เจ้า มงคลนาวิน กรรมการวิทยานิพนธ์ ที่ช่วยชี้แนะสิ่งต่างๆ อาจารย์อังคณา อัมพรสิทธิกุลที่ช่วยตรวจสอบเอกสารต่างๆที่ใช้ในการทดลอง และอาจารย์ทุกท่านที่ให้ความรู้และอบรมสิ่งต่างๆให้กับผู้วิจัย และขอขอบคุณหน่วยทดลองทุกท่านที่สละเวลามาช่วยในการทดลองที่แสนยาวนานของผู้วิจัย

ที่สำคัญที่สุดขอขอบพระคุณคุณพ่อ คุณแม่ที่สละทรัพย์ครั้งแล้วครั้งเล่าให้กับลูกสาวได้เรียนจนจบ และครอบครัวที่คอยให้กำลังใจและรับฟังคำบ่นต่างๆนานา สุดท้ายขอขอบคุณเพื่อนๆทุกคนที่ช่วยเหลือ และเพื่อนที่ดีที่สุดที่มาเป็นเพื่อนทุกครั้งที่ผู้วิจัยต้องมาทำการทดลองอันแสนยาวนาน

สารบัญ

หน้า

หน้า

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

หน้า

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการนำซอฟต์แวร์มาใช้มีเพิ่มมากขึ้น ทำให้บริษัทผู้ผลิตซอฟต์แวร์จำเป็นต้องผลิตซอฟต์แวร์ให้ตรงตามความต้องการของลูกค้า (Requirement) เพื่อให้สามารถอยู่รอดในตลาดที่มีการแข่งขันสูงได้ ดังนั้นทางบริษัทผู้ผลิตจึงมุ่งประเด็นไปที่ความพึงพอใจของลูกค้า ทั้งนี้ความต้องการหลักของลูกค้าคือต้องการซอฟต์แวร์ที่มีคุณภาพ (Quality) (O'Regan, 2002) โดย Juran (1951) นิยามคำว่าคุณภาพไว้ว่า คุณภาพคือความเหมาะสมสำหรับการใช้ (อ้างถึงใน O'Regan, 2002) และ Crosby (1980) ให้นิยามว่าคุณภาพคือความสอดคล้องกับความต้องการของลูกค้า (อ้างถึงใน O'Regan, 2002) ดังนั้นในการผลิตซอฟต์แวร์จึงจำเป็นต้องมีการควบคุมคุณภาพของซอฟต์แวร์ (Software Quality Control) เพื่อให้ได้ซอฟต์แวร์ที่สอดคล้องกับความต้องการ ของลูกค้าและเหมาะสมในการใช้งาน

วิธีการหนึ่งที่มีประสิทธิภาพและประสิทธิผลในการควบคุมคุณภาพของซอฟต์แวร์คือ การตรวจสอบซอฟต์แวร์ (Software Inspection) (Sabaliauskaite และคณะ, 2004) การตรวจสอบซอฟต์แวร์ถือเป็นกระบวนการที่สำคัญ เห็นได้จากการที่ตัวแบบวุฒิภาวะความสามารถ (Capability Maturity Model Integration: CMMI) กำหนดให้เป็นกระบวนการหนึ่งตั้งแต่ระดับ 3 ของตัวแบบ โดยการตรวจสอบว่าการออกแบบนั้นมีการออกแบบได้ตรงตามความต้องการทางซอฟต์แวร์หรือไม่ และการทำงานของซอฟต์แวร์มีความถูกต้องตรงตามที่ได้ออกแบบไว้หรือไม่ (Carnegie Mellon Software Engineering Institute, 2002)

กระบวนการหลักของการตรวจสอบซอฟต์แวร์คือ การค้นหาข้อบกพร่อง (Defect Detection) จากการตรวจสอบเอกสารต่างๆ เช่น เอกสารแสดงความต้องการซอฟต์แวร์ (Requirement Document) เอกสารแสดงการออกแบบซอฟต์แวร์ (Design Document) เอกสารที่ใช้ในการทดสอบซอฟต์แวร์ (Test Document) รวมถึงการตรวจสอบโปรแกรม (Code) เพื่อบันทึกและกำจัดข้อบกพร่องเหล่านั้น (Broy และ Denert, 2000) การตรวจสอบซอฟต์แวร์สามารถเป็นตัวชี้วัดคุณภาพของซอฟต์แวร์ แต่ไม่ได้เป็นการชี้วัดคุณภาพของผู้พัฒนาซอฟต์แวร์ การตรวจสอบซอฟต์แวร์มีข้อดีคือ สามารถตรวจสอบเพื่อค้นหาข้อบกพร่องได้ตั้งแต่ขั้นตอนต้นๆของการพัฒนาซอฟต์แวร์ ซึ่งการตรวจสอบซอฟต์แวร์มักทำตั้งแต่ขั้นตอนการออกแบบซอฟต์แวร์ (Software Design) นั่นคือการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ เนื่องจากในการค้นหาข้อบกพร่องนั้นยิ่งทำในขั้นตอนหลังๆของกระบวนการพัฒนาซอฟต์แวร์ค่าใช้จ่ายในการค้นหา

ข้อบกพร่องยิ่งสูงขึ้น (Fagan, 1976; Laitenberger และคณะ, 2000; Broy และ Denert, 2000; Biffl และ Halling, 2002; Sabaliauskaite และคณะ, 2004) ดังนั้นเมื่อออกแบบซอฟต์แวร์เสร็จแล้ว ผู้พัฒนาจึงควรตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ เพื่อตรวจสอบความถูกต้องตรงกันระหว่างการออกแบบแต่ละส่วนในเอกสารแสดงการออกแบบซอฟต์แวร์ และดูว่าเอกสารแสดงการออกแบบซอฟต์แวร์เหล่านั้นถูกต้องตรงตามความต้องการของลูกค้าหรือไม่ (Travassos และคณะ, 1999) เพื่อกำจัดข้อบกพร่องที่อาจเกิดขึ้นออกไป เนื่องจากข้อบกพร่องเหล่านี้จะกลายเป็นความยุ่งยากถ้าถูกปล่อยมาถึงขั้นตอนการพัฒนาโปรแกรม ซึ่งการแก้ไขในขั้นตอนนี้จะทำได้ยากขึ้นและอาจทำไม่ได้ถ้าโปรแกรมได้ถูกนำไปใช้งานแล้ว (Pressman,1997; Pfleeger,1998) นอกจากนี้การตรวจสอบซอฟต์แวร์ยังช่วยหลีกเลี่ยงการแก้ไขซอฟต์แวร์ใหม่ (Rework) ซึ่งทำให้ช่วยลดค่าใช้จ่ายในการผลิตซอฟต์แวร์ได้ (Sabaliauskaite, 2004; Laitenberger และคณะ, 2000)

การตรวจสอบซอฟต์แวร์สามารถช่วยในการค้นหาและแก้ไขข้อบกพร่องได้ถึงร้อยละ 50 - 90 ของข้อบกพร่องทั้งหมดที่มีในเอกสาร (Fagan, 1986; Gilb และ Graham, 1993) และยังเพิ่มผลผลิตสุทธิ (Net Productivity) ร้อยละ 30 - 50 และลดเวลาในการผลิตลงร้อยละ 10 - 30 (Gilb และ Graham, 1993) การทดลองของบริษัทอีริคสัน (Ericsson) ประเทศนอร์เวย์ (Norway) พบว่าการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ช่วยลดแรงงาน (Effort) ในการพัฒนาซอฟต์แวร์ได้ร้อยละ 20 (Conradi และคณะ, 1999) และงานวิจัยของ Briand และคณะ (1998) รายงานว่าการตรวจสอบซอฟต์แวร์ในขั้นตอนของการออกแบบซอฟต์แวร์ ช่วยประหยัดค่าใช้จ่ายในการทดสอบซอฟต์แวร์โดยเฉลี่ยร้อยละ 44 ขณะที่การตรวจสอบซอฟต์แวร์โดยการตรวจสอบโปรแกรม ช่วยประหยัดค่าใช้จ่ายในการทดสอบซอฟต์แวร์ร้อยละ 39 ผลการวิจัยต่างๆข้างต้นแสดงให้เห็นว่าการตรวจสอบซอฟต์แวร์นอกจากจะช่วยปรับปรุงคุณภาพของซอฟต์แวร์แล้ว ยังช่วยลดค่าใช้จ่ายและระยะเวลาในการพัฒนาซอฟต์แวร์อีกด้วย ชี้ได้ว่าการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ช่วยประหยัดค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ได้มากกว่าการตรวจสอบโปรแกรม ดังนั้นการตรวจสอบซอฟต์แวร์ควรเริ่มกระทำตั้งแต่ขั้นตอนของการออกแบบซอฟต์แวร์

การตรวจสอบซอฟต์แวร์ถูกเสนอเป็นครั้งแรกโดย Fagan ในปีค.ศ. 1976 ซึ่งกล่าวไว้ว่าการตรวจสอบซอฟต์แวร์นั้นทำเพื่อควบคุมและปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ (Software Development Process) (อ้างถึงใน Sabaliauskaite, 2004) และกำหนดขั้นตอนการตรวจสอบซอฟต์แวร์ไว้ 6 ขั้นตอนดังนี้

1. การวางแผน (Planning) เป็นขั้นตอนสำหรับวางแผนการตรวจสอบซอฟต์แวร์โดยการกำหนดหน้าที่ความรับผิดชอบให้กับผู้ตรวจสอบ (Inspector) กำหนดระยะเวลาและช่วงเวลาในการตรวจสอบซอฟต์แวร์

2. การประชุมแนะนำ (Overview) เป็นขั้นตอนในการสรุปข้อมูลของเอกสารต่างๆที่นำมาใช้ในการตรวจสอบซอฟต์แวร์ เพื่อให้ผู้ตรวจสอบทุกคนเข้าใจเอกสารได้ตรงกัน

3. การจัดเตรียม (Preparation) เป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนนำเอกสารที่ได้รับไปตรวจสอบเพื่อค้นหาข้อบกพร่อง กล่าวคือเป็นการตรวจสอบเฉพาะบุคคล (Individual Inspection) ไม่มีการปรึกษากัน จากนั้นบันทึกข้อบกพร่องไว้เพื่อนำมาสรุปข้อบกพร่องในที่ประชุมอีกครั้ง

4. การประชุมตรวจสอบ (Inspection Meeting) เป็นขั้นตอนสำหรับให้ผู้ตรวจสอบและผู้เขียน (Author) ซึ่งเป็นเจ้าของเอกสารที่นำมาตรวจสอบ ประชุมร่วมกันเพื่อสรุปว่าข้อบกพร่องที่พบเป็นข้อบกพร่องจริงหรือไม่และค้นหาข้อบกพร่องเพิ่มเติม จากนั้นจัดกลุ่มข้อบกพร่องและบันทึกข้อบกพร่องเพื่อส่งให้กับผู้เขียนนำไปแก้ไขต่อไป

5. การแก้ไขใหม่ (Rework) เป็นขั้นตอนที่ให้ผู้เขียนแก้ไขข้อบกพร่องตามที่ผู้ตรวจสอบได้แจ้งให้แก้ไข

6. การติดตาม (Follow-Up) เป็นขั้นตอนสำหรับตรวจสอบว่าผู้เขียนแก้ไขเอกสารตามที่ผู้ตรวจสอบแจ้งไว้เรียบร้อยหรือไม่

แม้ว่าทุกขั้นตอนของการตรวจสอบซอฟต์แวร์จะสำคัญ แต่หัวใจของการตรวจสอบซอฟต์แวร์อยู่ที่ขั้นตอนการจัดเตรียม (Laitenberger และคณะ, 2000) อันเป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนค้นหาข้อบกพร่องในเอกสารที่ต้องการตรวจสอบ ซึ่งเป็นวัตถุประสงค์หลักในการตรวจสอบซอฟต์แวร์ เพื่อให้เอกสารนั้นมีความถูกต้อง (Correctness) ความต้องกัน (Consistency) กับเอกสารอื่นที่นำเสนอระบบเดียวกัน และความสามารถการบำรุงรักษา (Maintainability) คือมีความยืดหยุ่นในการรองรับการเปลี่ยนแปลงแก้ไข (Laitenberger และคณะ, 2000) แม้ว่า Fagan (1976) รายงานว่าจำนวนข้อบกพร่องที่ค้นพบในขั้นตอนการประชุมตรวจสอบนั้นมากกว่าในขั้นตอนการจัดเตรียม แต่จากการทดลองของนักวิจัยบางกลุ่ม (Votta, 1993; Porter และคณะ, 1995; Johnson และ Tjahjono, 1998) ได้ศึกษาประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการประชุมตรวจสอบ และพบว่าขั้นตอนการประชุมตรวจสอบไม่ได้ช่วยให้พบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมเลย

การวัดประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการประชุมตรวจสอบ จะเปรียบเทียบจาก (1) อัตราการขยายในการประชุม (Meeting Gain) อันหมายถึงข้อบกพร่องที่พบเป็นครั้งแรกในขั้นตอนการประชุมตรวจสอบ โดยการเปรียบเทียบรายการข้อบกพร่องในขั้นตอนการจัดเตรียม เทียบกับรายการข้อบกพร่องในขั้นตอนการประชุมตรวจสอบว่ามีข้อบกพร่องใหม่เพิ่มขึ้นมาหรือไม่ และ (2) อัตราการสูญหายในการประชุม (Meeting Loss) อันหมายถึงข้อบกพร่องที่พบตั้งแต่ขั้นตอนการจัดเตรียม แต่ไม่ได้ถูกบันทึกว่าเป็นข้อบกพร่องในขั้นตอนการ

ประชุมตรวจสอบ โดยการเปรียบเทียบรายการข้อบกพร่องในขั้นตอนการจัดเตรียม กับรายการข้อบกพร่องในขั้นตอนการประชุมตรวจสอบว่ามีข้อบกพร่องใดหายไปหรือไม่

ดังนั้นงานวิจัยในปัจจุบันจึงมุ่งประเด็นไปที่การปรับปรุงประสิทธิภาพการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม โดยการศึกษาตัวแปรที่ส่งผลต่อประสิทธิภาพของกระบวนการค้นหาข้อบกพร่องในขั้นตอนการจัดเตรียม วิธีการหนึ่งที่นำมาใช้เพื่อช่วยค้นหาข้อบกพร่องในขั้นตอนการจัดเตรียมคือการใช้เทคนิคการอ่านซอฟต์แวร์ (Software Reading Techniques) (Porter และ Votta, 1994; Basili และคณะ, 1996; Miller และคณะ, 1998; Regnell และคณะ, 2000; Biffl และ Halling, 2000; Halling และ Biffl, 2001;) เทคนิคการอ่านซอฟต์แวร์เป็นเทคนิคที่ช่วยเพิ่มประสิทธิผลของการตรวจสอบซอฟต์แวร์ เนื่องจากในการตรวจสอบผู้ดำเนินรายการ (Moderator) ซึ่งเป็นผู้ควบคุมการตรวจสอบจะจัดเตรียมคำแนะนำ (Guidelines) ให้กับผู้ตรวจสอบได้ใช้เป็นแนวทางในการค้นหาข้อบกพร่องในขั้นตอนการจัดเตรียม โดยในคำแนะนำมีคำอธิบายเกี่ยวกับขั้นตอนการค้นหาข้อบกพร่อง และวิธีในการพิจารณาเอกสารถึงจุดที่ควรตรวจสอบในเอกสาร ซึ่งคำแนะนำเหล่านี้จะช่วยระบุข้อบกพร่อง (Basili และคณะ, 1996) โดย Basili และคณะ (1996) กล่าวว่าเทคนิคการอ่านซอฟต์แวร์เป็นเทคนิคสำหรับขั้นตอนการจัดเตรียม ที่ช่วยเพิ่มประสิทธิผล (Effectiveness) ในการตรวจสอบซอฟต์แวร์โดยไม่จำกัดเฉพาะโปรแกรมเท่านั้น ยังสามารถนำเทคนิคการอ่านซอฟต์แวร์มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ และเอกสารอื่นๆได้อีกด้วย (Porter และคณะ,1995; Basili และคณะ, 1996; Fusaro และคณะ, 1997; Shull, 1998; Zhang,1998)

เทคนิคการอ่านซอฟต์แวร์สามารถแบ่งออกได้หลายวิธีตามรูปแบบของคำแนะนำที่แตกต่างกัน เทคนิคที่นิยมนำมาใช้ในการตรวจสอบซอฟต์แวร์คือเทคนิคการอ่านบนพื้นฐานของเช็กลิสต์ (Checklist-Based Reading) หรือเรียกโดยย่อว่าเทคนิคซีบีอาร์ (CBR) (Porter และคณะ, 1995; Laitenberger และ Debaud, 2000; Biffl และ Halling, 2002; Sabaliauskaite และคณะ, 2002; Ciolkowski และคณะ, 2003; Sabaliauskaite, 2004) โดยเทคนิคนี้ได้จัดเตรียมคำแนะนำที่อยู่ในรูปแบบของเช็กลิสต์ (Checklist) ซึ่งเป็นรายการคำถามแบบใช่/ไม่ใช่ (Yes/No Questions) รายการคำถามเหล่านี้ช่วยให้ผู้ตรวจสอบทราบว่าจะต้องมองหาข้อบกพร่องที่ส่วนใดในเอกสารที่นำมาตรวจสอบ (Chernak, 1996) โดยในการจัดทำเช็กลิสต์นั้นสามารถจัดทำได้หลายรูปแบบขึ้นกับว่าต้องการนำเช็กลิสต์ไปตรวจสอบเอกสารอะไร และเนื่องจากในปัจจุบันการออกแบบและพัฒนาซอฟต์แวร์เชิงวัตถุ (Object-Oriented) ได้เข้ามาแทนที่วิธีพัฒนาซอฟต์แวร์แบบสัญนิยม (Conventional) กล่าวคือการพัฒนาซอฟต์แวร์แบบสัญนิยมจะแยกข้อมูล (data) และฟังก์ชัน (function) ออกจากกัน โดยในการออกแบบจะมองการทำงานให้มีการดำเนินไปตามฟังก์ชันต่างๆ

และแบ่งกลุ่มของฟังก์ชันออกเป็นมอดูล (Module) คือแบ่งตามหน้าที่การทำงาน ซึ่งฟังก์ชันเหล่านั้นจะประมวลผลไปตามข้อมูล เมื่อมีการเปลี่ยนแปลงโครงสร้างข้อมูลใดๆจะต้องตามแก้ฟังก์ชันต่างๆที่เกี่ยวข้อง ซึ่งอาจจะอยู่ในมอดูลเดียวกันหรือต่างมอดูลกันก็แล้วแต่ ซึ่งก่อให้เกิดความยุ่งยาก (Laitenberger และคณะ, 2000) แนวคิดเชิงวัตถุเป็นการพัฒนาซอฟต์แวร์ที่รวมข้อมูลและฟังก์ชันเป็นโครงสร้างเดียวกันซึ่งเรียกว่าคลาส (Class) สำหรับแนวคิดเชิงวัตถุจะเรียกฟังก์ชันว่าเมท็อด (Method)

การรวมข้อมูลและฟังก์ชันเป็นโครงสร้างเดียวกันช่วยลดความซับซ้อนในการใช้ข้อมูลข้ามเมท็อดได้ เนื่องจากถูกออกแบบให้มองกลุ่มข้อมูลและฟังก์ชันรวมอยู่ด้วยกัน แนวคิดเชิงวัตถุจึงถูกนำมาใช้เพิ่มขึ้นในกระบวนการพัฒนาซอฟต์แวร์ (Laitenberger และคณะ, 2000) นอกจากนี้แนวคิดเชิงวัตถุยังมีลักษณะที่โดดเด่นในเรื่องของ (1) ความสามารถการนำกลับมาใช้ใหม่ (Reusability) คือซอฟต์แวร์ที่พัฒนาขึ้นนั้นสามารถนำไปติดตั้งกับระบบอื่นหรือสิ่งแวดล้อมอื่นได้ง่าย โดยมีการแก้ไขเพียงเล็กน้อยเท่านั้น (2) ความสามารถการปรับขนาดได้ (Scalability) คือความสามารถรองรับการเพิ่มหรือการขยายระบบได้โดยง่าย และ (3) ความสามารถการบำรุงรักษา คือความสามารถในการรองรับการเปลี่ยนแปลงที่จะเกิดขึ้น มีความยืดหยุ่นในการเปลี่ยนแปลงแก้ไข (Ahrendt และคณะ, 2002; Lang และคณะ, 2005) โดยในการออกแบบซอฟต์แวร์ที่พัฒนาตามรูปแบบเชิงวัตถุนั้นนิยมใช้ภาษายูเอ็มแอล (UML: Unified Modeling Language) ซึ่งเป็นภาษามาตรฐานที่ใช้สำหรับการออกแบบแบบจำลองเชิงวัตถุ (Object-Oriented Modeling) (Booch และคณะ, 1999; Larman, 2002) แต่จากเทคนิคการอ่านซอฟต์แวร์ที่มีอยู่มีเพียงเทคนิคเดียวที่จัดทำขึ้นเพื่อใช้ตรวจสอบซอฟต์แวร์ที่พัฒนาตามรูปแบบเชิงวัตถุโดยเฉพาะ นั่นคือเทคนิคการอ่านเชิงวัตถุ (Object-Oriented Reading Techniques) หรือเรียกโดยย่อว่าเทคนิคโอโออาร์ที (OORT)

เทคนิคโอโออาร์ทีถูกจัดทำขึ้นโดยเฉพาะเพื่อช่วยในการค้นหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่พัฒนาตามรูปแบบเชิงวัตถุโดยเฉพาะ ซึ่ง Travassos และคณะ (1999) กล่าวว่าเทคนิคนี้มีแนวคิดหลักเพื่อให้ผู้ออกแบบซอฟต์แวร์มั่นใจในเรื่องความถูกต้องสอดคล้องกันของเอกสารแสดงการออกแบบซอฟต์แวร์ ทั้งในเอกสารเดียวกันและระหว่างเอกสารที่เสนอระบบเดียวกัน เพื่อให้มั่นใจในความถูกต้องและความสมบูรณ์(Completeness) ของเอกสารแสดงการออกแบบซอฟต์แวร์ เป็นเทคนิคที่จัดเตรียมคำแนะนำในรูปแบบที่เรียกว่าสถานการณ์ (Scenario) ให้กับผู้ตรวจสอบ เพื่อบอกว่าต้องตรวจสอบอะไรและตรวจสอบอย่างไร โดยสถานการณ์ (Scenario) จะประกอบด้วย 3 ส่วน (Porter และ Votta, 1994) คือ (1) คำแนะนำเกี่ยวกับหน้าที่ความรับผิดชอบ และสิ่งที่ต้องเกี่ยวข้องในแต่ละหน้าที่ (2) วิธีในการคัดเลือกข้อมูลที่สำคัญออกมาจากเอกสารที่ตรวจสอบ (3) คำถามที่สามารถตอบได้ด้วยข้อมูลที่คัดออกมา

ในการตรวจสอบซอฟต์แวร์นั้นยังขาดข้อมูลที่ใช้อธิบายให้ผู้ตรวจสอบเข้าใจขั้นตอนการค้นหาข้อบกพร่องในเอกสารที่ถูกพัฒนาตามรูปแบบเชิงวัตถุ เช่น โปรแกรมหรือแผนภาพ (Diagram) เนื่องจากการตรวจสอบซอฟต์แวร์ได้ถูกพัฒนาขึ้นตั้งแต่การพัฒนาซอฟต์แวร์โครงสร้างแบบสัญนิยม จึงจำเป็นที่จะต้องปรับปรุงเทคนิคการอ่านซอฟต์แวร์ที่มีอยู่ และหาว่าเทคนิคใดเหมาะสมสำหรับการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ หรือโปรแกรมที่พัฒนาตามรูปแบบเชิงวัตถุ (Travassos และคณะ, 1999; Laitenberger และคณะ, 2000; Dunsmore และคณะ, 2001; Dunsmore, 2002; Sabaliauskaite และคณะ, 2002; Sabaliauskaite และคณะ, 2003)

จากปัญหาข้างต้นทำให้มีการพยายามวิจัยเพื่อค้นหาเทคนิคการอ่านซอฟต์แวร์ที่เหมาะสมในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่พัฒนาตามรูปแบบเชิงวัตถุ เนื่องจากในการพบข้อบกพร่องตั้งแต่ขั้นตอนการออกแบบจะช่วยลดค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ได้มากกว่าการพบข้อบกพร่องในขั้นตอนการพัฒนาโปรแกรมดังที่ได้กล่าวมาแล้วข้างต้น โดยในการนำเทคนิคการอ่านซอฟต์แวร์ไปใช้ในบริษัทพัฒนาซอฟต์แวร์นั้นผู้ใช้มักเลือกเทคนิคการอ่านซอฟต์แวร์ โดยพิจารณาจากประสิทธิผล (Effectiveness) ของการตรวจสอบซอฟต์แวร์ซึ่งสามารถวัดได้จากจำนวนข้อบกพร่องที่พบ และประสิทธิภาพ (Efficiency) ซึ่งวัดได้จากแรงงานและระยะเวลาที่ใช้ในการตรวจสอบซอฟต์แวร์ (Laitenberger และ Debaud, 2000) นอกจากนี้ยังคำนึงถึงการกำจัดผลบวกปลอม (False Positive) อีกด้วย โดยผลบวกปลอมหมายถึงข้อบกพร่องที่ผู้ตรวจสอบบันทึกไว้แต่ในความเป็นจริงแล้วไม่ได้เป็นข้อบกพร่อง ดังนั้นถ้าปล่อยให้เกิดผลบวกปลอมจะทำให้การแก้ไขข้อบกพร่องผิดพลาด ซึ่งจะส่งผลต่อการพัฒนาซอฟต์แวร์ ทำให้ซอฟต์แวร์ที่ได้นั้นไม่มีคุณภาพ ดังนั้นงานวิจัยส่วนใหญ่ที่ผ่านมาจึงทดลองเพื่อเปรียบเทียบประสิทธิภาพ และประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยการนำเทคนิคการอ่านซอฟต์แวร์มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

มีงานวิจัยที่เกี่ยวข้องกับการวัดประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์โดยการเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ ซึ่งแสดงให้เห็นว่าการใช้เทคนิคซีบีอาร์ช่วยในขั้นตอนการจัดเตรียม เพิ่มประสิทธิภาพในการตรวจสอบซอฟต์แวร์สูงกว่าเทคนิคอื่น แต่ใช้เวลาในการตรวจสอบนานกว่า (Sabaliauskaite, 2000) แต่บางงานวิจัยไม่สามารถสรุปผลได้ว่าเทคนิคซีบีอาร์นั้นช่วยให้การตรวจสอบซอฟต์แวร์มีประสิทธิภาพมากขึ้นเห็นได้จากงานวิจัยของ Fusaro และคณะ (1997) และงานวิจัยของ Dunsmore และคณะ (2003) สำหรับงานวิจัยที่วัดประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคโอโออาร์ทีคืองานวิจัยของ Melo และคณะ (2001) และ Conradi และคณะ (2003) ได้ทดลองและสรุปผลได้ว่าเทคนิคโอโออาร์ทีสามารถค้นหาข้อบกพร่องได้มาก และช่วยในการกำจัดผลบวกปลอม (False Positive)

จากที่ได้กล่าวมาข้างต้นจะเห็นได้ว่างานวิจัยที่ผ่านมาได้ให้ความสำคัญในการศึกษาเพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ โดยการนำเทคนิคการอ่านซอฟต์แวร์มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล เนื่องจากมีงานวิจัยได้กล่าวไว้ว่าจำเป็นที่จะต้องปรับปรุงเทคนิคการอ่านซอฟต์แวร์ที่มีอยู่ และหาว่าเทคนิคใดที่เหมาะสมสำหรับการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบตามรูปแบบเชิงวัตถุ (Travassos และคณะ, 1999; Laitenberger และคณะ, 2000; Dunsmore และคณะ, 2001; Dunsmore, 2002; Sabaliauskaite และคณะ, 2002; Sabaliauskaite และคณะ, 2003)

งานวิจัยนี้จึงสนใจ (1) การเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม โดยการนำเทคนิคการอ่านที่แตกต่างกันมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล (2) การเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ โดยการนำเทคนิคการอ่านที่แตกต่างกันมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล (3) การเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ โดยนำเทคนิคการอ่านที่แตกต่างกันมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล เนื่องจากผลบวกปลอมจะส่งผลทำให้การพัฒนาซอฟต์แวร์มีปัญหาในอนาคต

ในงานวิจัยนี้แบ่งการเก็บข้อมูลออกเป็น 2 ขั้นตอนคือเก็บข้อมูลจาก (1) ขั้นตอนการจัดเตรียมซึ่งเป็นการตรวจสอบเฉพาะบุคคล (Individual Inspection) (2) ขั้นตอนการประชุมตรวจสอบซึ่งเป็นการตรวจสอบร่วมกันระหว่างคณะผู้ตรวจสอบ โดยในการทดลองขั้นตอนการประชุมตรวจสอบนั้นกำหนดให้หนึ่งกลุ่มประกอบด้วยผู้ตรวจสอบสามคน โดยการวิจัยเป็นการเปรียบเทียบระหว่างเทคนิคซีบีอาร์ซึ่งเป็นเทคนิคที่นิยมนำมาใช้ในการตรวจสอบซอฟต์แวร์ โดยเทคนิคซีบีอาร์ที่นำมาใช้ในงานวิจัยจะนำเช็กลิสต์สำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลโดยเฉพาะมาใช้ เทียบกับเทคนิคโอโออาร์ทีซึ่งเป็นเทคนิคที่จัดทำขึ้นสำหรับตรวจสอบซอฟต์แวร์ที่ถูกพัฒนาขึ้นตามรูปแบบเชิงวัตถุโดยเฉพาะ เนื่องจากหลังจากผู้วิจัยได้ทบทวนวรรณกรรม (Literature Review) ที่เกี่ยวข้องกับเทคนิคการอ่านซอฟต์แวร์ ยังไม่มีงานวิจัยที่เปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล โดยการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบ ว่าเทคนิคใดมีประสิทธิภาพ

และประสิทธิผลมากกว่ากัน เพื่อเป็นแนวทางให้กับผู้ที่ต้องการนำเทคนิคเหล่านี้ไปใช้ในการตรวจสอบซอฟต์แวร์ โดยการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลต่อไป

1.2 วัตถุประสงค์งานวิจัย

1.2.1 เพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม ระหว่างการใช้เทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

1.2.2 เพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

1.2.3 เพื่อเปรียบเทียบประสิทธิผลของการกำจัดผลบวกปลอมในขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

1.3 ขอบเขตงานวิจัย

1.3.1 เทคนิคการอ่านซอฟต์แวร์ที่นำมาเปรียบเทียบเพื่อหาประสิทธิภาพ และประสิทธิผลในการตรวจสอบซอฟต์แวร์นั้นใช้ 2 เทคนิค คือเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที

1.3.2 ในงานวิจัยนี้เป็นการตรวจสอบซอฟต์แวร์โดยทำเฉพาะขั้นตอนการประชุมแนะนำการจัดเตรียม และการประชุมตรวจสอบเท่านั้น

1.3.3 งานวิจัยนี้ตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ทางด้านธุรกิจ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล ซึ่งประกอบด้วยแผนภาพยูสเคส (Use Case Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) แผนภาพสถานะ (State Machine Diagram) และแผนภาพคลาส (Class Diagram) เอกสารอธิบายคลาสของระบบ (Class Description) และเอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) เท่านั้น

1.3.4 เอกสารแสดงการออกแบบซอฟต์แวร์ที่นำมาใช้ในการทดลองเป็นแผนภาพที่มีความถูกต้องและน่าเชื่อถือ โดยผู้วิจัยกำหนดข้อบกพร่องแต่ละประเภทไว้ในทุกแผนภาพของเอกสารแสดงการออกแบบซอฟต์แวร์

1.3.5 เอกสารแสดงการออกแบบซอฟต์แวร์ถูกสร้างด้วยโปรแกรมวิชวล พาราดาม ฟอร์ ยูเอ็มแอล คอมมิวนิตี เอดิชัน เวอร์ชัน 5.1 (Visual Paradigm for UML Community Edition Version 5.1)

1.4 ขั้นตอนการดำเนินงานวิจัย

1.4.1 ศึกษาการใช้แผนภาพยูเอ็มแอลในการออกแบบระบบเชิงวัตถุ

1.4.2 ศึกษากระบวนการตรวจสอบซอฟต์แวร์ เพื่อใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยใช้แผนภาพยูเอ็มแอล

1.4.3 ศึกษากระบวนการดำเนินการของเทคนิคการอ่านซอฟต์แวร์แต่ละเทคนิคที่นำมาใช้ในการตรวจสอบซอฟต์แวร์

1.4.4 เลือกเอกสารแสดงการออกแบบซอฟต์แวร์ที่ต้องการนำมาตรวจสอบ อันเป็นเอกสารแสดงการออกแบบซอฟต์แวร์ทางด้านธุรกิจ ที่มีความถูกต้องสมบูรณ์และมีความน่าเชื่อถือ โดยผู้วิจัยกำหนดข้อบกพร่องแต่ละประเภทไว้ในทุกๆแผนภาพของเอกสารแสดงการออกแบบซอฟต์แวร์เพื่อนำไปใช้ในการทดลอง

1.4.5 ออกแบบการทดลองเพื่อให้สามารถเก็บข้อมูลที่ต้องการนำไปใช้ในการตอบวัตถุประสงค์ของงานวิจัยได้อย่างครบถ้วน

1.4.6 แบ่งกลุ่มตัวอย่างเป็น 2 กลุ่มคือ (1) กลุ่มที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ (2) กลุ่มที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์

1.4.7 ดำเนินการทดลองตามที่ได้ออกแบบไว้ โดยในการทดลองแบ่งออกเป็น 3 ขั้นตอน คือ ขั้นตอนการประชุมแนะนำ ขั้นตอนการจัดเตรียม และขั้นตอนการประชุมตรวจสอบ

1.4.8 เก็บรวบรวมข้อมูลที่ได้จากการทดลอง โดยในการเก็บข้อมูลนั้นแบ่งออกเป็น 2 ขั้นตอน คือขั้นตอนการจัดเตรียม และขั้นตอนการประชุมตรวจสอบ

1.4.9 นำข้อมูลที่ได้จากการทดลองมาวิเคราะห์เพื่อตอบวัตถุประสงค์การวิจัย

1.4.10 จัดทำเอกสารรายงานผลการทดลองที่ได้ และรวบรวมเอกสารงานวิจัยต่างๆ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ผู้ที่ต้องการควบคุมคุณภาพซอฟต์แวร์ที่จัดทำขึ้น สามารถนำงานวิจัยนี้ไปใช้เป็นแนวทางสำหรับควบคุมคุณภาพซอฟต์แวร์ในขั้นตอนของการออกแบบซอฟต์แวร์ เพื่อปรับปรุงคุณภาพของซอฟต์แวร์ที่จัดทำให้ดียิ่งขึ้น

1.5.2 ผลการทดลองที่ได้สามารถเป็นแนวทางให้กับผู้ที่ต้องการนำเทคนิคซีบีอาร์ หรือเทคนิคโอโออาร์ที่ไปใช้ในการตรวจสอบซอฟต์แวร์ เพื่อให้สามารถเลือกเทคนิคที่นำมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่จัดทำขึ้นโดยการใช้แผนภาพยูเอ็มแอลได้อย่างเหมาะสม

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 บทนำ

งานวิจัยนี้เป็นการเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ (Software Inspection) โดยการนำเทคนิคการอ่านซอฟต์แวร์ (Software Reading Techniques) มาตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ใช้แผนภาพยูเอ็มแอล (UML) ซึ่งมีทฤษฎีที่เกี่ยวข้องกับงานวิจัยดังต่อไปนี้ คุณภาพซอฟต์แวร์ (Software Quality) การควบคุมคุณภาพซอฟต์แวร์ (Software Quality Control) การทบทวนซอฟต์แวร์ (Software Review) การตรวจสอบซอฟต์แวร์ เทคนิคการอ่านซอฟต์แวร์ และภาษายูเอ็มแอล

2.2 คุณภาพซอฟต์แวร์ (Software Quality)

มีผู้ให้คำนิยามของคุณภาพซอฟต์แวร์ไว้มากมายได้แก่ คุณภาพของซอฟต์แวร์คือระดับความสามารถของซอฟต์แวร์ โดยการรวมคุณลักษณะ (Attribute) ที่เป็นที่ต้องการ หรือคุณลักษณะที่ยอดเยี่ยมของระบบคอมพิวเตอร์ (Fisher และ Light, 1979; Software Engineering Technical Committee of the IEEE Computer Society, 1983) ความสามารถของซอฟต์แวร์ในการตอบสนองความต้องการที่ได้กำหนดไว้ (United States Department of Defense, 1988) ความเหมาะสมในการใช้งานของซอฟต์แวร์ (Schulmeyer และ Mcmanus, 1992) สำหรับงานวิจัยนี้คุณภาพซอฟต์แวร์คือการที่ซอฟต์แวร์มีฟังก์ชันการทำงานที่ถูกต้องและสอดคล้องกับความต้องการที่ได้กำหนดไว้

การประเมินคุณภาพซอฟต์แวร์ เป็นการวัดคุณลักษณะต่างๆของซอฟต์แวร์ว่าครบตามที่ลูกค้าต้องการหรือไม่ โดยจะใช้ตัวชี้บอกคุณภาพซอฟต์แวร์ (Software Quality Indicator) เป็นตัวกำหนดลักษณะคุณภาพซอฟต์แวร์ (Software Quality Characteristics) เช่น การวัดความสามารถการใช้งานของซอฟต์แวร์ จะใช้ความหนาแน่นของข้อบกพร่อง ความหนาแน่นของความผิดพร่องและการเตรียมเอกสารเป็นตัวชี้บอกเป็นต้น (Schulmeyer และ Mcmanus, 1992) ซึ่งจะอธิบายโดยละเอียดในหัวข้อ 2.2.1 และ 2.2.2 ต่อไป

2.2.1 ลักษณะคุณภาพซอฟต์แวร์ (Software Quality Characteristics) จากนิยามของคุณภาพซอฟต์แวร์ข้างต้นที่กล่าวไว้ว่า คุณภาพซอฟต์แวร์คือการรวมคุณลักษณะที่เป็นที่ต้องการ ดังนั้นในการประเมินว่าซอฟต์แวร์มีคุณภาพหรือไม่ จึงต้องวัดว่าซอฟต์แวร์นั้นมีลักษณะคุณภาพซอฟต์แวร์ครบตามความต้องการของลูกค้าหรือไม่ โดย McCall (1979) กำหนดลักษณะคุณภาพของซอฟต์แวร์ไว้ 11 ข้อดังนี้

1. ความถูกต้อง (Correctness) หมายถึงขอบเขตของซอฟต์แวร์ในการตอบสนองวัตถุประสงค์ของผู้ใช้ หรือการแสดงให้เห็นว่าซอฟต์แวร์สามารถตอบสนองความต้องการของลูกค้าได้อย่างถูกต้องครบถ้วน

2. ความเชื่อถือได้ (Reliability) หมายถึงความถูกต้องและแม่นยำในการทำงานแต่ละฟังก์ชันของซอฟต์แวร์

3. ประสิทธิภาพ (Efficiency) หมายถึงจำนวนรวมของทรัพยากรที่ใช้ในการทำงานฟังก์ชันหนึ่งของซอฟต์แวร์ เพื่อแสดงให้เห็นว่าการทำงานแต่ละฟังก์ชันนั้นมีการใช้ทรัพยากรได้อย่างเหมาะสม

4. บูรณภาพ (Integrity) คือขอบเขตการควบคุมการเข้าถึงซอฟต์แวร์ โดยผู้ที่ไม่ได้รับอนุญาต (Unauthorized Person) กล่าวคือซอฟต์แวร์มีการจำกัดขอบเขตการเข้าใช้งานได้อย่างเหมาะสม

5. ความสามารถการใช้งาน (Usability) คือความสามารถของผู้ใช้ในการเรียนรู้ การปฏิบัติ การเตรียมข้อมูลเข้า (Inputs) และการแปลข้อมูลออก (Outputs) ของโปรแกรม

6. ความสามารถการบำรุงรักษา (Maintainability) คือความสามารถในการติดตั้งและซ่อมแซมข้อบกพร่องในการดำเนินการของซอฟต์แวร์ ความสามารถในการรองรับการเปลี่ยนแปลงที่จะเกิดขึ้น

7. ความสามารถการทดสอบ (Testability) คือความสามารถในการทดสอบซอฟต์แวร์เพื่อรับประกันการทำงานของฟังก์ชัน

8. ความสามารถการปรับตัว (Flexibility) คือความพยายามในการเปลี่ยนแปลงการดำเนินการของซอฟต์แวร์

9. ความสามารถของการใช้ได้บนหลายแพลตฟอร์ม (Portability) คือความพยายามในการย้ายโปรแกรมจากโครงแบบฮาร์ดแวร์ (Hardware Configuration) หนึ่ง ไปยังอีกโครงแบบฮาร์ดแวร์หนึ่ง หรือสิ่งแวดล้อมของระบบหนึ่งไปอีกสิ่งแวดล้อมของระบบหนึ่ง

10. ความสามารถการนำกลับมาใช้ใหม่ (Reusability) คือขอบเขตของซอฟต์แวร์ที่สามารถนำไปใช้ในโปรแกรมประยุกต์ (Application) อื่น หรือสิ่งแวดล้อมอื่นได้ง่ายโดยการแก้ไขเพียงเล็กน้อยเท่านั้น

11. ความสามารถการเชื่อมต่อ (Interoperability) คือความพยายามในการเชื่อมต่อระบบหนึ่งไปยังอีกระบบหนึ่ง

2.2.2 ตัวชี้บอกคุณภาพซอฟต์แวร์ (Software Quality Indicator) ในการกำหนดคุณภาพซอฟต์แวร์ จะใช้ตัวชี้บอกคุณภาพซอฟต์แวร์เป็นตัวกำหนด โดย United State Air Force (1987) ได้เสนอตัวชี้บอกคุณภาพซอฟต์แวร์ เพื่อบอกถึงปัจจัยที่ใช้กำหนดคุณภาพซอฟต์แวร์ไว้ 7 ข้อดังนี้

1. ความสมบูรณ์ (Completeness) เป็นตัวชี้บอกเกี่ยวกับความสมบูรณ์ของข้อกำหนดซอฟต์แวร์ (Software Specification) ในระหว่างขั้นตอนวิเคราะห์ความต้องการในกระบวนการพัฒนาซอฟต์แวร์

2. โครงสร้างการออกแบบ (Design Structure) เป็นตัวชี้บอกที่ถูกใช้ในการกำหนดความง่ายของการออกแบบที่ลงรายละเอียด (Detailed Design) การออกแบบไม่ซับซ้อน และมีความชัดเจน

3. ความหนาแน่นของข้อบกพร่อง (Defect Density) เป็นตัวชี้บอกไปยังคุณภาพของการออกแบบซอฟต์แวร์ และการสร้างโปรแกรมซึ่งสามารถหาค่าบกพร่องเหล่านี้ได้ โดยการตรวจสอบการออกแบบ (Design Inspection) และการตรวจสอบโปรแกรม (Code Inspection)

4. ความหนาแน่นของความผิดพร่อง (Fault Density) เป็นตัวชี้บอกเหมือนกับความหนาแน่นของข้อบกพร่อง แต่มีข้อแตกต่างคือสามารถหาค่าบกพร่องเหล่านี้ได้โดยการทดสอบ (Test) ซึ่งข้อบกพร่องเหล่านี้สามารถตรวจพบในขั้นตอนการประยุกต์ (Application Phase) และถูกเรียกว่าความผิดพร่อง (Fault)

5. ขอบเขตการทดสอบ (Test Coverage) ตัวชี้บอกนี้เป็นการเสนอการวัดค่าความสมบูรณ์ในความก้าวหน้าของการทดสอบ จากทั้งมุมมองของผู้พัฒนาและผู้ใช้

6. ความพอเพียงในการทดสอบ (Test Sufficiency) เป็นเครื่องมือที่มีประโยชน์ในการประเมินความพอเพียงในการทดสอบเบ็ดเสร็จ (Integration Test) และการทดสอบระบบ (System Test) ซึ่งอยู่บนพื้นฐานของการทำนายความบกพร่องซอฟต์แวร์ที่คงเหลืออยู่ ตัวชี้บอกนี้มักถูกใช้กับขอบเขตการทดสอบ

7. การเตรียมเอกสาร (Documentation) วัตถุประสงค์ของตัวชี้บอกนี้คือ เพื่อเพิ่มความพอเพียงและความสมบูรณ์ของการเตรียมเอกสารซอฟต์แวร์

ตารางที่ 2-1 ตารางแสดงความสัมพันธ์ระหว่างตัวชี้บอกคุณภาพซอฟต์แวร์ และลักษณะคุณภาพซอฟต์แวร์ (Schulmeyer และ Mcmanus, 1992)

คุณลักษณะ / ตัวชี้บอก	ความถูกต้อง	ประสิทธิภาพ	ความสามารถการบำรุงรักษา	ความเชื่อถือได้	ความสามารถการทดสอบระบบ	ความสามารถในการใช้งานร่วมกับระบบอื่นได้	ความสามารถในการใช้งานใหม่	ความสามารถในการใช้งาน	ความสมบูรณ์	ความสามารถในการใช้งาน
ความสมบูรณ์	X					X		X		X	
โครงสร้างการออกแบบ		X				X		X		X	
ความหนาแน่นของข้อบกพร่อง	X					X		X		X	X
ความหนาแน่นของความผิดพร่อง	X					X		X		X	X
ขอบเขตการทดสอบ	X					X		X		X	
ความพอเพียงในการทดสอบ	X					X		X		X	
การเตรียมเอกสาร	X					X		X		X	X

ตารางที่ 2-1 ข้างต้นแสดงให้เห็นว่าความหนาแน่นของข้อบกพร่อง และความหนาแน่นของความผิดพร่อง เป็นตัวชี้บอกถึงลักษณะของคุณภาพหลายข้อได้แก่ ความถูกต้อง ประสิทธิภาพ ความสามารถการบำรุงรักษา ความเชื่อถือได้ ความสามารถการทดสอบระบบ และความสามารถการใช้งาน และจากที่ Yeh (1993) กล่าวว่าข้อบกพร่องเป็นสิ่งสำคัญที่ส่งผลต่อคุณภาพของซอฟต์แวร์ ดังนั้นจึงเกิดการควบคุมคุณภาพซอฟต์แวร์ โดยมุ่งประเด็นไปที่การค้นหาข้อบกพร่องและกำจัดข้อบกพร่องเหล่านั้นเพื่อให้ได้ซอฟต์แวร์ที่มีคุณภาพ

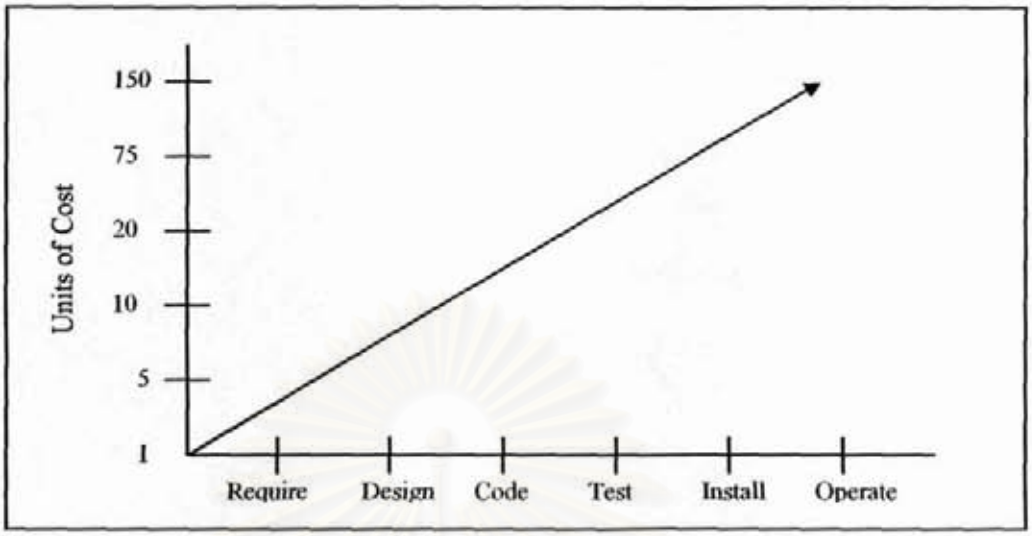
2.3 การควบคุมคุณภาพซอฟต์แวร์ (Software Quality Control)

การควบคุมคุณภาพซอฟต์แวร์ หมายถึงการตรวจสอบกระบวนการพัฒนาซอฟต์แวร์ โดยประเมินว่าซอฟต์แวร์ที่พัฒนาสอดคล้องกับความต้องการ (Requirements) ที่ได้ระบุไว้ในขั้นตอน (Phase) ก่อนหน้าหรือไม่ เพื่อให้ซอฟต์แวร์ที่ผลิตขึ้นมานั้นมีคุณภาพตรงตามที่ได้กำหนดไว้ โดยในการควบคุมคุณภาพนั้นเกี่ยวข้องกับ 2 กิจกรรมหลักคือ (O'Regan, 2002)

1. การทบทวนซอฟต์แวร์ (Software Review) เป็นขั้นแรกและขั้นสำคัญของการควบคุมคุณภาพ โดยการตรวจสอบเพื่อค้นหาและระบุข้อบกพร่อง การทบทวนซอฟต์แวร์สามารถกระทำได้ตั้งแต่ขั้นตอนต้นๆของการพัฒนาซอฟต์แวร์ โดยการตรวจสอบเอกสารต่างๆ เช่น เอกสารแสดงความต้องการซอฟต์แวร์ เอกสารแสดงการออกแบบซอฟต์แวร์ เอกสารที่ใช้ในการทดสอบซอฟต์แวร์ รวมถึงการตรวจสอบโปรแกรม โดยการตรวจสอบว่าเอกสารต่างๆที่จัดทำขึ้นนั้นมีความต้องกันหรือไม่ (Consistency) คือดูว่าแต่ละเอกสารนั้นนำเสนอระบบออกมาได้ตรงกันหรือไม่ และยังดูว่าสอดคล้องกับความต้องการของลูกค้าหรือไม่ เพื่อให้สามารถพัฒนาซอฟต์แวร์ได้อย่างมีคุณภาพ การทบทวนซอฟต์แวร์สามารถแบ่งได้เป็น (1) การทบทวนระดับเดียวกัน (Peer Reviews) (2) การตรวจตลอด (Walkthroughs) และ (3) การตรวจสอบ (Inspection)

2. การทดสอบซอฟต์แวร์ (Software Testing) เป็นการทวนสอบ (Verify) ความถูกต้องของระบบซอฟต์แวร์ (Software System) และระบุข้อบกพร่องที่เกิดขึ้น เพื่อเพิ่มความเชื่อมั่นให้กับลูกค้าในผลิตภัณฑ์ที่จัดทำขึ้น ดังนั้นในการทดสอบซอฟต์แวร์ต้องทำหลังจากที่ซอฟต์แวร์ได้ถูกพัฒนาขึ้นแล้ว

การศึกษาของ Boehm (1981) ได้แสดงค่าใช้จ่ายสำหรับการระบุข้อบกพร่องในแต่ละขั้นตอนของการพัฒนาซอฟต์แวร์ ดังรูปที่ 2-1 ซึ่งชี้ให้เห็นว่าในการค้นหาข้อบกพร่องนั้น ยิ่งทำในขั้นตอนหลังๆของการพัฒนาซอฟต์แวร์ค่าใช้จ่ายในการค้นหาข้อบกพร่องยิ่งสูงขึ้น ดังนั้นในการตรวจสอบข้อบกพร่องของซอฟต์แวร์จึงควรกระทำตั้งแต่ขั้นตอนต้นๆของการพัฒนาซอฟต์แวร์ ซึ่งสามารถทำได้โดยการทบทวนซอฟต์แวร์



รูปที่ 2-1 แสดงค่าใช้จ่ายสำหรับการระบุข้อบกพร่องในแต่ละขั้นตอนของการพัฒนาซอฟต์แวร์ (Boehm, 1981)

2.4 การทบทวนซอฟต์แวร์ (Software Review)

การทบทวนซอฟต์แวร์เป็นกระบวนการที่สำคัญในการพัฒนาซอฟต์แวร์ เห็นได้จากการที่แบบจำลองวุฒิภาวะความสามารถ (Capability Maturity Model Integration: CMMI) กำหนดไว้เป็นกระบวนการหนึ่งตั้งแต่ระดับที่ 3 ของแบบจำลอง (Carnegie Mellon Software Engineering Institute, 2002) โดยเป็นการระบุว่าควรมีการทบทวนซอฟต์แวร์โดยอาจอยู่ในรูปของการทบทวนระดับเดียวกัน (Peer Review) การตรวจตลอด (Walkthroughs) หรือ การตรวจสอบ (Inspection) เพื่อตรวจสอบว่าการออกแบบซอฟต์แวร์นั้น ออกแบบได้ตรงตามความต้องการทางซอฟต์แวร์ของลูกค้า และเพื่อตรวจสอบว่าการทำงานของซอฟต์แวร์มีความถูกต้องตรงตามที่ได้ออกแบบไว้ โดยในการทบทวนซอฟต์แวร์สามารถแบ่งออกได้ 3 ประเภทตามระดับความเป็นทางการ (Formal) ในการทบทวนซอฟต์แวร์ โดยสามารถเรียงลำดับจากน้อยไปมากได้ดังนี้ (Horch, 1996)

1. การทบทวนระดับเดียวกัน (Peer Reviews) เป็นการทบทวนซอฟต์แวร์ที่มีความเป็นทางการน้อยที่สุด โดยการตั้งคำถามเพื่อตรวจสอบให้มั่นใจในความถูกต้องของซอฟต์แวร์ เช่น ข้อมูลที่ได้รับมาถูกต้องหรือไม่ หรือสูตรที่ใช้นั้นมีความถูกต้องหรือไม่ ส่วนมากมักถูกใช้ระหว่างการร่างซอฟต์แวร์ โดยผลของการทบทวนระดับเดียวกันนั้นมักเป็นคำพูดไม่มีการบันทึกเป็นเอกสาร (Horch, 1996)

2. การตรวจตลอด (Walkthroughs) เป็นวิธีที่ผู้ออกแบบซอฟต์แวร์ (Designer) หรือนักเขียนโปรแกรม (Programmer) นำสมาชิกของทีมพัฒนาตรวจสอบความต้องการของลูกค้า การเตรียมเอกสาร การออกแบบ หรือโปรแกรม โดยการถามคำถามและแสดงความคิดเห็นเกี่ยวกับ

ข้อบกพร่องที่เป็นไปได้ การฝ่าฝืนมาตรฐานของการพัฒนาและปัญหาอื่นๆ ลักษณะพิเศษของการตรวจตลอดคือสมาชิกทั้งหมดที่ตรวจสอบ ยกเว้นผู้ออกแบบซอฟต์แวร์และนักเขียนโปรแกรม จำเป็นต้องเข้าใจรายละเอียด (โครงสร้าง ขั้นตอนวิธี โครงสร้างข้อมูลและอื่นๆ) ของซอฟต์แวร์ก่อนที่จะตรวจสอบ (Kenett และ Baker, 1999) การตรวจตลอดไม่มีการแบ่งขั้นตอนการทำอย่างชัดเจน มีกระบวนการหลักๆคือการตรวจหาและบันทึกข้อบกพร่องซึ่งทำไปพร้อมๆกัน ดังนั้นจึงไม่ควรใช้เวลาเกิน 2 ชั่วโมง (Schach, 1999)

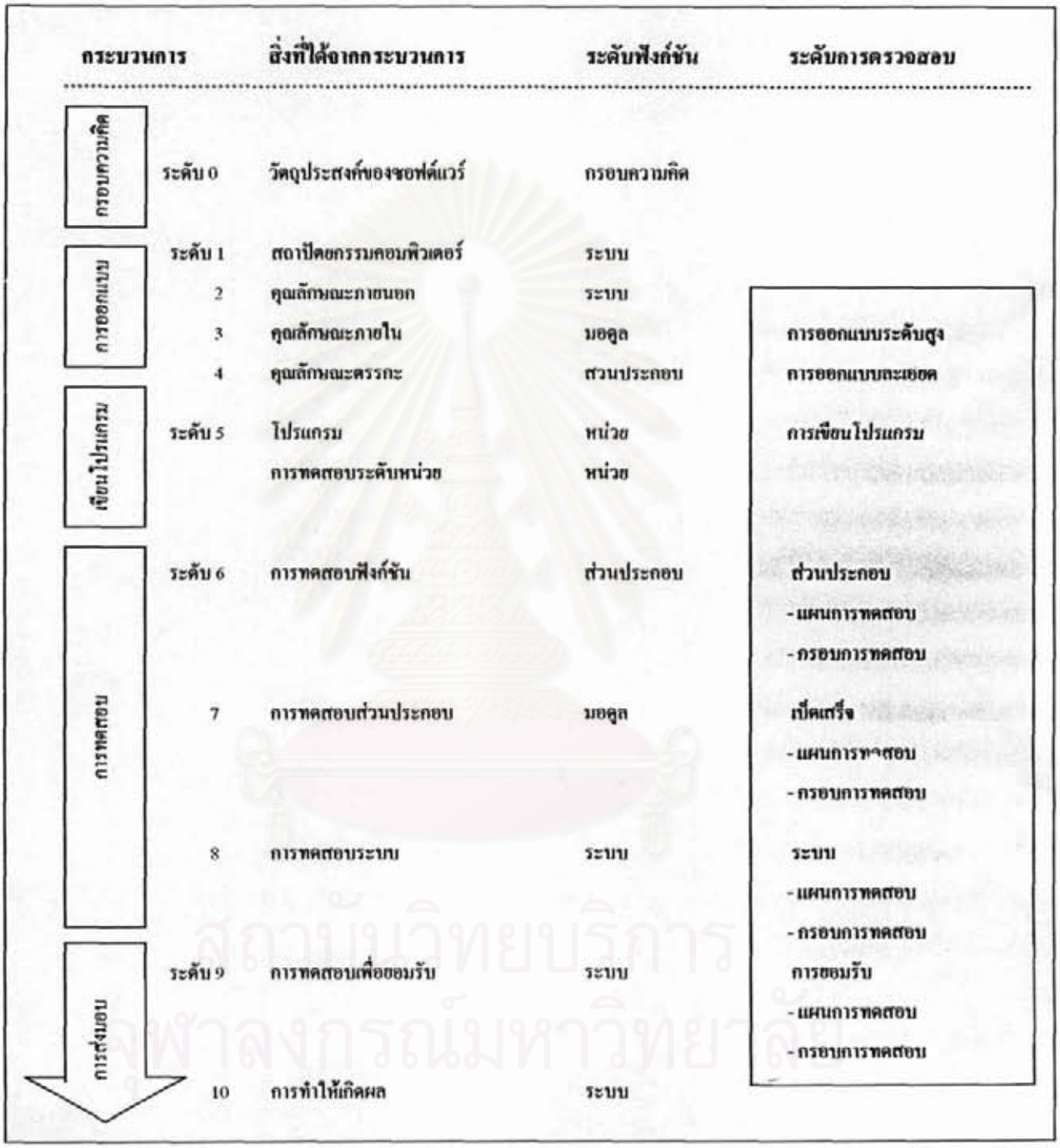
3. การตรวจสอบ (Inspection) เป็นวิธีที่มีความเป็นทางการในการทบทวนซอฟต์แวร์มากที่สุด เพื่อค้นหาข้อบกพร่อง การฝ่าฝืนมาตรฐานของการพัฒนา และปัญหาอื่นๆ (Horch, 1996; Kenett และ Baker, 1999) ซึ่งประกอบด้วยขั้นตอนในการทำที่มีการแบ่งแยกกันอย่างชัดเจน โดยแต่ละขั้นตอนมีวัตถุประสงค์ของมันเอง (Kenett และ Baker, 1999) โดยงานวิจัยนี้ให้ความสนใจในเรื่องของการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์

2.5 การตรวจสอบซอฟต์แวร์ (Software Inspection)

การตรวจสอบซอฟต์แวร์ได้ถูกแนะนำ และใช้เป็นครั้งแรกตั้งแต่ต้นทศวรรษ 1970 ซึ่งเป็นแนวคิดของ Fagan (1976) ซึ่งมีความต้องการที่จะกำจัดข้อบกพร่อง โดยคำนึงถึงโครงสร้าง การควบคุม และมุ่งประเด็นไปที่วิธีในการทวนสอบโปรแกรม (Code Verification) ซึ่งวิธีของ Fagan คือการตรวจสอบโปรแกรมและการออกแบบ (Code and Design Inspection) ซึ่งในปัจจุบันเรียกว่าการตรวจสอบซอฟต์แวร์ โดยสร้างวิธีนี้ขึ้นมาเพื่อควบคุมขั้นตอนภายในกระบวนการพัฒนาซอฟต์แวร์ และเพื่อให้แน่ใจในความสามารถวัดได้ (Measurability) และสามารถทำซ้ำได้ (Repeatability) ของการตรวจสอบซอฟต์แวร์ ดังนั้น Fagan (1976) จึงได้นิยามขั้นตอนการตรวจสอบซอฟต์แวร์ และกำหนดบทบาทของผู้ที่มีส่วนร่วมและหน้าที่ไว้อย่างชัดเจน Fagan (1976) ให้ความสำคัญกับการบันทึกการตรวจสอบซอฟต์แวร์ และการวิเคราะห์ผลของการตรวจสอบซอฟต์แวร์ โดยข้อมูลในการตรวจสอบซอฟต์แวร์จะถูกนำไปใช้เพื่อระบุมอดูลที่ผิด พลาดได้ง่าย และวิเคราะห์ความถี่ของประเภทข้อบกพร่องที่เกิดขึ้น

การตรวจสอบซอฟต์แวร์นั้นสามารถทำได้ในหลายๆขั้นตอนของกระบวนการพัฒนาซอฟต์แวร์ โดยการตรวจสอบเอกสารต่างๆในแต่ละขั้นตอนของกระบวนการพัฒนาซอฟต์แวร์หรือโปรแกรม จากรูปที่ 2-2 เห็นได้ว่าเริ่มมีการตรวจสอบซอฟต์แวร์ตั้งแต่ขั้นตอนออกแบบของกระบวนการพัฒนาซอฟต์แวร์ โดยการตรวจสอบเอกสารที่บันทึกการออกแบบ ขั้นตอนการเขียนโปรแกรมโดยการตรวจสอบโปรแกรม ขั้นตอนการทดสอบ และขั้นตอนของการส่งมอบซอฟต์แวร์

(Delivery) ซึ่งต้องมีการทดสอบซอฟต์แวร์โดยลูกค้า จึงต้องมีการตรวจสอบเอกสารบันทึกการวางแผนการทดสอบ (Test Plan) และกรอบการทดสอบ (Test Case)



รูปที่ 2-2 แสดงความสัมพันธ์ของกระบวนการพัฒนาซอฟต์แวร์และการตรวจสอบซอฟต์แวร์ (Strauss และ Ebenau, 1994)

2.5.1 คณะตรวจสอบซอฟต์แวร์ (Software Inspection Team) การตรวจสอบซอฟต์แวร์ประกอบด้วยกลุ่มคนที่ทำงานด้วยกัน เพื่อค้นหาและกำจัดข้อบกพร่อง และเพื่อให้การตรวจสอบซอฟต์แวร์เสร็จสมบูรณ์ได้มีการกำหนดบทบาทไว้ทั้งหมด 5 บทบาท ซึ่งแต่ละบทบาทมีการแบ่งความรับผิดชอบอย่างชัดเจนดังต่อไปนี้ (Strauss และ Ebenau, 1994)

1. ผู้เขียนเอกสาร (Author) คือผู้ที่สร้างเอกสารที่ถูกตรวจสอบ หน้าที่ของผู้เขียนคือรับผิดชอบในการเปลี่ยนแปลงแก้ไขเอกสาร (ปกติแล้วมักเป็นคนเดียวกับผู้ที่เขียนเอกสารที่นำมาตรวจสอบ แต่ไม่จำเป็นเสมอไป) ตอบคำถามทั้งหมดที่ผู้ตรวจสอบถามในขั้นตอนการประชุมตรวจสอบ และค้นหาข้อบกพร่องตามพื้นฐานความเข้าใจในตัวเอกสารที่ตรวจสอบนั้น

2. ผู้ดำเนินรายการ (Moderator) เป็นผู้ที่รับผิดชอบการตรวจสอบซอฟต์แวร์ให้สามารถปฏิบัติไปได้จนเสร็จเรียบร้อย โดยรับผิดชอบทุกขั้นตอนในตรวจสอบซอฟต์แวร์ ซึ่งผู้ดำเนินรายการเป็นผู้ที่ตัดสินว่าเอกสารที่จะนำมาตรวจสอบนั้นพร้อมสำหรับการนำมาตรวจสอบแล้วหรือยัง และเป็นผู้กำหนดหน้าที่ของคณะตรวจสอบ โดยผู้ดำเนินรายการนั้นควรเป็นผู้ที่มีประสบการณ์ และมีความรู้ในหลายๆด้าน

3. ผู้อ่าน (Reader) ไม่ใช่ผู้เขียน โดยผู้อ่านจะเตรียมการเพื่ออธิบายฟังก์ชันของงานมีการถอดความโดยละเอียดให้เหมาะสมสำหรับการตรวจสอบ นั่นคือผู้อ่านพยายามที่จะอธิบายและแปลข้อมูลของงานให้สามารถเข้าใจได้ภายในระยะเวลาอันสั้นซึ่งเป็นสิ่งที่สำคัญมาก และไม่ใช้ผู้ดำเนินรายการกับผู้บันทึกมาทำหน้าที่เป็นผู้อ่าน

4. ผู้บันทึก (Recorder) หน้าที่ของผู้บันทึกคือ บันทึกและจัดกลุ่มข้อบกพร่องในขั้นตอนการประชุมตรวจสอบทั้งหมดลงในรายการข้อบกพร่อง (Defect List) และช่วยผู้ดำเนินรายการในการเตรียมรายงานของขั้นตอนการประชุมตรวจสอบ

5. ผู้ตรวจสอบ (Inspector) สมาชิกทั้งหมดที่เข้ามามีส่วนร่วมในการตรวจสอบซอฟต์แวร์ ถือว่าเป็นผู้ตรวจสอบรวมทั้งผู้เขียนด้วย ซึ่งความรับผิดชอบของผู้ตรวจสอบคือ ค้นหาข้อบกพร่องไม่ได้เสนอวิธีแก้ปัญหา

ขนาดของคณะผู้ตรวจสอบอย่างน้อยที่สุดควรประกอบด้วย 3 คนคือ ผู้ดำเนินรายการ (ซึ่งอาจทำหน้าที่เป็นผู้บันทึกด้วย) ผู้อ่าน และผู้เขียน โดยส่วนมากแล้วแนะนำว่าไม่ควรเกิน 7 คน (Strauss และ Ebenau, 1994)

2.5.2 ขั้นตอนการตรวจสอบซอฟต์แวร์ Fagan (1976) กำหนดขั้นตอนในการตรวจสอบซอฟต์แวร์ไว้ 6 ขั้นตอนดังนี้ (อ้างถึงใน Strauss และ Ebenau, 1994)

1. การวางแผน (Planning) การตรวจสอบซอฟต์แวร์ควรถูกวางแผนขั้นตอนไว้ในวัฏจักรการพัฒนาซอฟต์แวร์ (Development Cycle) โดยขั้นตอนนี้เป็นหน้าที่รับผิดชอบร่วมกันของผู้ดำเนินรายการ และผู้เขียน

ผู้ดำเนินรายการมีหน้าที่รับผิดชอบดังนี้

- ตรวจสอบผลิตภัณฑ์ (Work Product) ที่นำมาตรวจสอบโดยประเมินจากเกณฑ์การรับเข้า (Entry Criteria) ของการตรวจสอบซอฟต์แวร์ที่ได้ตั้งไว้ ถ้ายังไม่พร้อมที่จะทำ ผู้ดำเนินรายการจะแจ้งให้ผู้เขียนทราบว่าต้องทำอย่างไรให้เอกสารนั้นพร้อมที่จะนำมาตรวจสอบ
- กำหนดตารางเวลาในการตรวจสอบซอฟต์แวร์ โดยควรกำหนดเวลาในการทำขั้นตอนการจัดเตรียมให้เพียงพอในการค้นหาข้อบกพร่อง และเวลาที่ใช้สำหรับขั้นตอนการประชุมตรวจสอบไม่ควรเกิน 2-3 ชั่วโมง
- เลือกผู้ตรวจสอบ และมอบหมายหน้าที่ให้กับผู้อ่านและผู้บันทึก โดยคณะผู้ตรวจสอบควรมีขนาดเล็ก ซึ่งบางทีอาจมาจากแผนกอื่นเพื่อความเหมาะสม และมั่นใจได้ในการตรวจสอบ
- จัดหาห้องที่ใช้ในการทำขั้นตอนการประชุมตรวจสอบให้เหมาะสม และจัดหาอุปกรณ์ช่วยสำหรับผู้อ่านและผู้บันทึก
- จัดเตรียมเอกสารแจ้งการประชุมตรวจสอบ (Inspection Meeting Notice) ซึ่งต้องประกอบด้วยข้อมูลเกี่ยวกับประเภทของการตรวจสอบซอฟต์แวร์ วัน เวลา สถานที่ และระยะเวลาในการทำ

ผู้เขียนมีหน้าที่รับผิดชอบดังนี้

- ตรวจสอบผลิตภัณฑ์ที่นำมาตรวจสอบกับผู้ดำเนินรายการ เพื่อดูว่าผลิตภัณฑ์นั้นพร้อมที่จะนำไปตรวจสอบหรือยัง
- รวบรวมและแจกจ่ายผลิตภัณฑ์ที่นำมาตรวจสอบไปยังคณะผู้ตรวจสอบ

ตารางที่ 2-2 ตารางแสดงข้อมูลเข้า กระบวนการ และข้อมูลออกของขั้นตอนการวางแผน (Strauss และ Ebenau, 1994)

ข้อมูลเข้า	กระบวนการ	ข้อมูลออก
1. ผลิตภัณฑ์ที่จะนำมาตรวจสอบเสร็จสมบูรณ์แล้ว	1. ผู้ดำเนินรายการตรวจสอบผลิตภัณฑ์ที่นำมาตรวจสอบกับเกณฑ์การรับเข้า	1. เกณฑ์การรับเข้าถูกตรวจสอบแล้ว
2. เกณฑ์การรับเข้าในการตรวจสอบ	2. ผู้ดำเนินรายการเลือกผู้ตรวจสอบ	2. เอกสารประกาศการประชุมการตรวจสอบ
3. อัตราเวลาในการตรวจสอบ (Inspection Rate)	3. ผู้ดำเนินรายการกำหนดเวลาในการตรวจสอบซอฟต์แวร์	3. สิ่งที่นำมาตรวจสอบ

2. การประชุมแนะนำ (Overview) จัดทำขึ้นเพื่อเพิ่มความเข้าใจให้กับผู้ตรวจสอบเกี่ยวกับผลิตภัณฑ์ที่นำมาตรวจสอบ และเทคนิคที่นำมาใช้ในการตรวจสอบซอฟต์แวร์ โดยใช้ระยะเวลาในการทำประมาณ 2-3 ชั่วโมง ซึ่งขั้นตอนนี้เป็นหน้าที่ของผู้ดำเนินรายการ ผู้เขียน และผู้ตรวจสอบ

ผู้ดำเนินรายการมีหน้าที่รับผิดชอบดังนี้

- จัดเตรียม ประกาศ และจัดการประชุมแนะนำ
- แนะนำและจัดเตรียมข้อมูลที่จำเป็นเพื่อให้การตรวจสอบซอฟต์แวร์เสร็จสมบูรณ์

ผู้เขียนมีหน้าที่รับผิดชอบดังนี้

- อธิบายเทคนิคที่ใช้ในการตรวจสอบซอฟต์แวร์
- อธิบายข้อมูลที่จำเป็น เช่น ฟังก์ชันการทำงานของผลิตภัณฑ์ที่นำมาตรวจสอบเพื่อให้ผู้ตรวจสอบเข้าใจในงานที่กำลังจะทำ

ผู้ตรวจสอบมีหน้าที่รับผิดชอบดังนี้

- ตั้งคำถามเกี่ยวกับผลิตภัณฑ์ที่นำมาตรวจสอบ สมมติฐานของผลิตภัณฑ์ที่นำมาตรวจสอบให้เข้าใจความหมาย และมีความเข้าใจในผลิตภัณฑ์เพื่อนำไปทำขั้นตอนการจัดเตรียมต่อไป

ตารางที่ 2-3 ตารางแสดงข้อมูลเข้า กระบวนการ และข้อมูลออกของขั้นตอนการประชุมแนะนำ (Strauss และ Ebenau, 1994)

ข้อมูลเข้า	กระบวนการ	ข้อมูลออก
1. ผลิตภัณฑ์ที่จะนำมาตรวจสอบ	1. ผู้ดำเนินรายการจัดการทำขั้นตอนการประชุมแนะนำ	1. เพิ่มความเข้าใจในผลิตภัณฑ์ที่นำมาตรวจสอบมากขึ้น
2. เอกสารที่เกี่ยวข้อง	2. ผู้เขียนอธิบายผลิตภัณฑ์ที่นำมาตรวจสอบให้กับผู้เข้าร่วม	2. คำตอบเกี่ยวกับผลิตภัณฑ์ที่จะนำมาตรวจสอบ
	3. ผู้ดำเนินรายการกำหนดความรับผิดชอบในการอ่าน	3. หน้าที่ความรับผิดชอบได้ถูกกำหนดแล้ว

3. การจัดเตรียม (Preparation) ขั้นตอนนี้ผู้ตรวจสอบทุกคนจะได้รับสำเนาของผลิตภัณฑ์เพื่อนำไปตรวจสอบ นอกจากนี้ยังได้รับเอกสารที่เกี่ยวข้อง เช่น เช็กลิสต์เพื่อนำมาใช้เป็นแนวทางในการตรวจสอบ สำหรับระยะเวลาที่ใช้ในขั้นตอนนี้ขึ้นกับความเข้าใจของผู้ตรวจสอบในผลิตภัณฑ์ที่นำมาตรวจสอบ ขั้นตอนนี้เป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนค้นหาข้อบกพร่อง ซึ่งอาจมีเครื่องมือและเทคนิคที่นำมาช่วยในการค้นหาข้อบกพร่อง โดยขั้นตอนนี้เป็นหน้าที่ของผู้ตรวจสอบ

ผู้ตรวจสอบมีหน้าที่รับผิดชอบดังนี้

- ใช้เช็กลิสต์เพื่อช่วยในการค้นหาข้อบกพร่อง
- บันทึกเวลาที่ใช้ในการทำขั้นตอนการจัดเตรียม (Preparation Time) และรายงานเวลาที่ใช้ในการทำขั้นตอนการจัดเตรียม ให้กับผู้ดำเนินรายการทราบในขั้นตอนการประชุมตรวจ สอบ

เวลาที่ใช้ในการทำขั้นตอนการจัดเตรียมของผู้ตรวจสอบทุกคน ถูกรวบรวมในขั้นตอนการประชุมตรวจสอบ เพื่อนำไปประเมินประสิทธิผลของกระบวนการตรวจสอบซอฟต์แวร์ต่อไป โดยข้อบกพร่องนั้นถูกบันทึกระหว่างการทำขั้นตอนการจัดเตรียม

ตารางที่ 2-4 ตารางแสดงข้อมูลเข้า กระบวนการ และข้อมูลออกของขั้นตอนการจัดเตรียม (Strauss และ Ebcnau, 1994)

ข้อมูลเข้า	กระบวนการ	ข้อมูลออก
1. ผลิตภัณฑ์ที่นำมาตรวจสอบ	1. ผู้ตรวจสอบทุกคนศึกษาผลิตภัณฑ์ที่นำมาตรวจสอบ	1. ความรู้และเข้าใจเกี่ยวกับผลิตภัณฑ์ที่นำมาตรวจสอบ
2. เช็กลิสต์	2. ผู้ตรวจสอบบันทึกข้อบกพร่องที่พบและบันทึกเวลาที่ใช้ในการทำขั้นตอนการจัดเตรียม	2. ข้อบกพร่อง และเวลาที่ใช้ขั้นตอนการจัดเตรียมถูกบันทึก

4. การประชุมตรวจสอบ (Inspection Meeting) ขั้นตอนนี้เป็นความร่วมมือกันระหว่างผู้เขียน ผู้ดำเนินรายการ และผู้ตรวจสอบ โดยจะตรวจสอบผลิตภัณฑ์ที่นำมาตรวจสอบทีละแผนภาพในกรณีที่เป็นเอกสารแสดงการออกแบบซอฟต์แวร์ และทีละบรรทัดในกรณีที่เป็นโปรแกรม หรือเอกสารอื่นเพื่อค้นหาข้อบกพร่อง และผู้ตรวจสอบตั้งคำถามโดยนำข้อบกพร่องที่พบในขั้นตอนการจัดเตรียมมาเป็นพื้นฐาน เพื่อวิเคราะห์ว่าเป็นข้อบกพร่องจริงหรือไม่ และให้ผู้เขียนเป็นผู้อธิบาย ถ้าจริงจะบันทึกลงเอกสารบันทึกรายการข้อบกพร่อง (Inspection Defect List) ซึ่งจะส่งให้กับผู้เขียนนำไปแก้ไขต่อไป นอกจากนี้มีการจัดกลุ่มของข้อบกพร่อง เช่น ประเภทของข้อบกพร่อง กลุ่มของข้อบกพร่อง และระดับความรุนแรง

การทำขั้นตอนการประชุมตรวจสอบมีกำหนดการดังนี้

1. การแนะนำ คือขั้นตอนที่ผู้ดำเนินรายการอธิบายบทบาทให้กับผู้ตรวจสอบ และบอกวัตถุประสงค์ในการตรวจสอบซอฟต์แวร์ เพื่อให้ผู้ตรวจสอบมุ่งประเด็นไปที่การค้นหาข้อบกพร่อง

2. การเตรียมความพร้อม คือขั้นตอนที่ผู้ดำเนินรายการตรวจสอบความพร้อม โดยการถามเวลาที่ใช้ในการทำขั้นตอนการจัดเตรียมของผู้ตรวจสอบแต่ละคน โดยผู้ดำเนินรายการจะบันทึกข้อมูลนี้ไว้ในรายงานการจัดการการตรวจสอบ (Inspection Management Report) ถ้าผู้ดำเนินรายการเชื่อว่าผู้ตรวจสอบทุกคนพร้อมจะดำเนินการขั้นตอนประชุมตรวจสอบต่อไป แต่ถ้าไม่พร้อมจะหยุดและปรับตารางเวลา เพื่อนัดมาทำขั้นตอนประชุมตรวจสอบใหม่อีกครั้ง

3. อ่านและบันทึกข้อบกพร่อง คือขั้นตอนของการนำข้อบกพร่องที่ได้จากขั้นตอนการจัดเตรียมมาปรึกษากันว่าเป็นข้อบกพร่องจริงหรือไม่ จากนั้นจัดกลุ่มข้อบกพร่องและบันทึกลงในเอกสารบันทึกรายการข้อบกพร่อง โดยเอกสารบันทึกรายการข้อบกพร่องจะบันทึกสิ่งที่ผู้เขียนต้องแก้ไข จึงเป็นสิ่งสำคัญว่าผู้ตรวจสอบทุกคนต้องเห็นพ้องกัน

4. ตรวจสอบข้อบกพร่อง คือขั้นตอนที่ผู้บันทึกตรวจสอบเอกสารบันทึกรายการข้อบกพร่องร่วมกับผู้ดำเนินรายการ เพื่อตรวจสอบความสมบูรณ์และความถูกต้องของเอกสารบันทึกรายการข้อบกพร่อง

5. สรุปวิธีการควบคุมผลิตภัณฑ์ (Product Disposition) เมื่อเอกสารบันทึกรายการข้อบกพร่องถูกทำอย่างสมบูรณ์และได้รับการยอมรับแล้ว ผู้ดำเนินรายการจะพิจารณาว่าผลิตภัณฑ์ที่นำมาตรวจสอบนั้นจำเป็นต้องนำกลับมาตรวจสอบอีกหรือไม่ โดยการกำหนดวิธีการควบคุมผลิตภัณฑ์ ซึ่งสามารถแบ่งออกได้ 3 ประเภทคือ (1) ยอมรับ (Accept) แสดงว่าผลิตภัณฑ์ที่นำมาตรวจสอบนั้นไม่จำเป็นต้องนำไปแก้ไขใหม่ (Rework) และไม่ต้องนำมาตรวจสอบอีก (2) เงื่อนไข (Conditionally) มีข้อบกพร่องที่มีระดับรุนแรงจำนวนไม่มากนัก และสิ่งที่จะต้องปรับปรุงนั้นไม่กระทบกับการออกแบบของผลิตภัณฑ์ที่นำมาตรวจสอบ ดังนั้นผู้เขียนต้องแก้ไขใหม่แต่ไม่ต้องนำมาตรวจสอบอีก (3) การตรวจสอบอีกครั้ง (Reinspect) มีข้อบกพร่องที่มีระดับรุนแรงจำนวนมาก หรือในการแก้ไขใหม่นั้นมีการเปลี่ยนแปลงการออกแบบเดิมที่ได้ทำไว้ ดังนั้นผู้เขียนต้องแก้ไขใหม่และนำกลับมาตรวจสอบอีกครั้ง

ผู้ดำเนินรายการมีหน้าที่รับผิดชอบดังนี้

- ดำเนินการขั้นตอนการประชุมตรวจสอบเพื่อให้เสร็จสิ้นตามกำหนดเวลา
- ปฏิบัติตามกำหนดการของการทำขั้นตอนการประชุมตรวจสอบ
- นำเอกสารอ้างอิงที่เป็นประโยชน์ในการทำขั้นตอนการประชุมการตรวจสอบ เช่น เอกสารอ้างอิงภาษาโปรแกรม เอกสารที่แสดงมาตรฐานเป็นต้น ให้กับผู้ตรวจสอบใช้เป็นแนวทางในการตรวจสอบ
- ทำสำเนาเอกสารบันทึกรายการข้อบกพร่องให้กับผู้เขียน และทำเอกสารสรุปข้อบกพร่อง (Inspection Defect Summary) และรายงานการจัดการการตรวจสอบ (Inspection Management Report)
- สรุปวิธีการควบคุมผลิตภัณฑ์ที่นำมาตรวจสอบ โดยการกำหนดวิธีการควบคุมผลิตภัณฑ์

ผู้เขียนมีหน้าที่รับผิดชอบดังนี้

- มุ่งประเด็นไปที่การทำความเข้าใจเกี่ยวกับข้อคิดเห็นทั้งหมด และช่วยในการค้นหาข้อบกพร่อง
- ทำความเข้าใจและบันทึกข้อคิดเห็นซึ่งช่วยอธิบายข้อบกพร่องเพื่อนำไปแก้ไขต่อไป

- ตรวจสอบผลิตภัณฑ์ที่นำมาตรวจสอบ ซึ่งอยู่บนพื้นฐานของความสามารถในการทำความเข้าใจอย่างแท้จริง
- ตอบคำถามเพื่อช่วยผู้ตรวจสอบในการวิเคราะห์ว่าข้อบกพร่องที่พบนั้นเป็นข้อบกพร่องจริงหรือไม่

ผู้อ่านมีหน้าที่รับผิดชอบดังนี้

- อธิบายและแปลฟังก์ชันการทำงานของผลิตภัณฑ์ที่นำมาตรวจสอบให้ผู้ตรวจสอบเข้าใจชัดเจน
- ตอบคำถามที่ถูกถามโดยผู้ตรวจสอบ หรือถามผู้เขียนเพื่อความชัดเจน

ผู้บันทึกมีหน้าที่รับผิดชอบดังนี้

- ทำความคุ้นเคยกับการจัดกลุ่มข้อบกพร่อง เพื่อจัดประเภทข้อบกพร่องที่พบในผลิตภัณฑ์ที่นำมาตรวจสอบ
- บันทึกข้อบกพร่องทั้งหมดลงในเอกสารบันทึกรายการข้อบกพร่อง
- สรุปและอธิบายข้อบกพร่องอย่างชัดเจน (ไม่ใช่วิธีในการแก้ไขข้อบกพร่อง)
- ส่งเอกสารบันทึกรายการข้อบกพร่องให้กับผู้ดำเนินรายการตรวจสอบข้อบกพร่อง

ผู้ตรวจสอบมีหน้าที่รับผิดชอบดังนี้

- ค้นหาข้อบกพร่องซึ่งเป็นงานที่สำคัญที่สุด
- ให้ความเห็นและมีส่วนร่วมในขั้นตอนการประชุมตรวจสอบ

ตารางที่ 2-5 ตารางแสดงข้อมูลเข้า กระบวนการ และข้อมูลออกของขั้นตอนการประชุมตรวจสอบ (Strauss และ Ebenau, 1994)

ข้อมูลเข้า	กระบวนการ	ข้อมูลออก
1. ผลิตภัณฑ์ที่นำมาตรวจสอบถูกตรวจสอบแล้ว	1. ผู้ดำเนินรายการเริ่มขั้นตอนการประชุมตรวจสอบ และตรวจสอบเวลาที่ใช้ในการทำขั้นตอนการจัดเตรียม	1. เอกสารบันทึกรายการข้อบกพร่อง
2. เอกสารอ้างอิงที่จำเป็น	2. อ่านผลิตภัณฑ์ที่นำมาตรวจสอบ	2. รายงานการตรวจสอบ (Meeting Report)

ตารางที่ 2-5 (ต่อ) ตารางแสดงข้อมูลเข้า กระบวนการ และข้อมูลออกของขั้นตอนการประชุมตรวจสอบ (Strauss และ Ebenau, 1994)

ข้อมูลเข้า	กระบวนการ	ข้อมูลออก
3. เวลาที่ใช้ในการทำขั้นตอนการจัดเตรียมของผู้ตรวจสอบแต่ละคน	3. ระบุและบันทึกข้อบกพร่อง	3. กำหนดวิธีการควบคุมผลิตภัณฑ์
4. ข้อบกพร่องที่ผู้ตรวจสอบแต่ละคนพบในขั้นตอนการจัดเตรียม	4. เอกสารบันทึกรายการข้อบกพร่องถูกตรวจสอบ และกำหนดวิธีการควบคุมผลิตภัณฑ์	
5. เช็กลิสต์	5. จัดทำรายงานการตรวจสอบและเอกสารสรุปข้อบกพร่อง	
6. รูปแบบหรืออุปกรณ์ในการบันทึกขั้นตอนการประชุมตรวจสอบ		

5. การแก้ไขใหม่ (Rework) เป็นขั้นตอนที่ผู้เขียนกลับไปจัดการแก้ไขข้อบกพร่องตามที่พบ โดยคำนึงถึงระดับความรุนแรงของข้อบกพร่อง และส่งผลิตภัณฑ์ที่แก้ไขเรียบร้อยแล้วคืนให้กับผู้ดำเนินรายการเพื่อเข้าสู่ขั้นตอนการติดตาม (Follow-Up)

ผู้เขียนมีหน้าที่รับผิดชอบดังนี้

- แก้ไขข้อบกพร่องทั้งหมด
- แจ้งกับผู้ดำเนินรายการเมื่อแก้ไขเสร็จเรียบร้อยเพื่อเข้าสู่ขั้นตอนการติดตาม

ตารางที่ 2-6 ตารางแสดงข้อมูลเข้า กระบวนการและข้อมูลออกของขั้นตอนการแก้ไขใหม่ (Strauss และ Ebenau, 1994)

ข้อมูลเข้า	กระบวนการ	ข้อมูลออก
1. เอกสารบันทึกรายการข้อบกพร่อง	1. ผู้เขียนแก้ปัญหาทั้งหมดที่บันทึกในเอกสารบันทึกรายการข้อบกพร่อง	1. ผลิตภัณฑ์ที่นำมาตรวจสอบถูกแก้ไขแล้ว

ตารางที่ 2-6 (ต่อ) ตารางแสดงข้อมูลเข้า กระบวนการและข้อมูลออกของขั้นตอนการแก้ไขใหม่ (Strauss และ Ebenau, 1994)

ข้อมูลเข้า	กระบวนการ	ข้อมูลออก
2. วิธีการควบคุมผลิตภัณฑ์		2. ผู้ดำเนินรายการได้รับแจ้งว่าผลิตภัณฑ์ที่นำมาตรวจสอบถูกแก้ไขเรียบร้อยแล้ว

6. การติดตาม (Follow-Up) เป็นขั้นตอนสำหรับผู้ดำเนินรายการในการติดตามการแก้ไขตามวิธีการควบคุมผลิตภัณฑ์หลังการตรวจสอบ (Product Disposition) หากกำหนดวิธีการเป็นเงื่อนไข (Conditional) หมายความว่าผู้ดำเนินรายการเป็นผู้ตรวจสอบการแก้ไขผลิตภัณฑ์ที่นำมาตรวจสอบ แต่หากกำหนดวิธีการเป็นการตรวจสอบอีกครั้ง (Reinspect) หมายความว่าผู้ดำเนินรายการจะนำผลิตภัณฑ์ที่นำมาตรวจสอบเข้าสู่กระบวนการอีกครั้ง

ผู้ดำเนินรายการมีหน้าที่รับผิดชอบดังนี้

- ตรวจสอบผลิตภัณฑ์ที่นำมาตรวจสอบที่ถูกแก้ไขในกรณีที่วิธีการควบคุมผลิตภัณฑ์มีค่าเป็นเงื่อนไข
- จัดทำการตรวจสอบผลิตภัณฑ์ที่ถูกแก้ไขแล้วอีกครั้ง ในกรณีที่วิธีการควบคุมผลิตภัณฑ์มีค่าเป็นตรวจสอบอีกครั้ง
- ถ้าวิธีการควบคุมผลิตภัณฑ์มีค่าเป็นยอมรับ ให้บันทึกรับรองในรายงานการจัดการว่าผลิตภัณฑ์ที่นำมาตรวจสอบนี้สมบูรณ์

ตารางที่ 2-7 ตารางแสดงข้อมูลเข้า กระบวนการ และข้อมูลออกของขั้นตอนการติดตาม (Strauss และ Ebenau, 1994)

ข้อมูลเข้า	กระบวนการ	ข้อมูลออก
1. ผลิตภัณฑ์ที่แก้ไขแล้ว	1. ผู้ดำเนินรายการตรวจสอบผลิตภัณฑ์ที่แก้ไขแล้ว	1. ผลิตภัณฑ์ที่แก้ไขได้ถูกตรวจสอบแล้ว
2. (ถ้ามีการตรวจสอบอีกครั้งต้องใช้ข้อมูลเข้าตามขั้นที่ 1-4)	2. (ถ้ามีการตรวจสอบอีกครั้งต้องใช้กระบวนการตามขั้นที่ 1-4)	2. (ถ้าวิธีการควบคุมผลิตภัณฑ์เป็นเงื่อนไข หรือตรวจสอบอีกครั้งให้ดูข้อมูลออกของขั้นตอน 4)
	3. ผู้ดำเนินรายการจัดทำรายงานการตรวจสอบ	3. รายงานการตรวจสอบเสร็จสมบูรณ์

ตารางที่ 2-8 ตารางแสดงความสัมพันธ์ระหว่างขั้นตอนและบทบาท (Strauss และ Ebenau, 1994)

ขั้นตอน	ผู้ดำเนินรายการ	ผู้เขียน	ผู้อ่าน	ผู้บันทึก	ผู้ตรวจสอบ
การวางแผน	X	X			
การประชุมแนะนำ	X	X	-	-	-
การจัดเตรียม	X	X	X	X	X
การประชุมตรวจสอบ	X	X	X	X	X
การแก้ไขใหม่		X			
การติดตาม	X	X	(X)	(X)	(X)

หมายเหตุ

1. เครื่องหมายขีดสั้น (Dash) หมายถึงคณะผู้ตรวจสอบเหล่านี้ไม่ได้มีบทบาทเกี่ยวกับขั้นตอน แต่เป็นผู้ถูกเสนอโดยรับฟังข้อมูลต่างๆเกี่ยวกับผลิตภัณฑ์ที่นำมาตรวจสอบ ที่ผู้ดำเนินรายการและผู้เขียนนำเสนอ

2. เครื่องหมายวงเล็บ (Parentheses) หมายถึงบทบาทของคณะผู้ตรวจสอบจะขึ้นกับวิธีการควบคุมผลิตภัณฑ์ ซึ่งจะเป็นตัวชี้ว่าคณะผู้ตรวจสอบไหนบ้างที่เกี่ยวข้องกับขั้นตอน

3. ไม่มีเครื่องหมาย หมายถึงบทบาทนั้นไม่มีหน้าที่เกี่ยวข้องกับขั้นตอนการตรวจสอบซอฟต์แวร์

2.5.3 ประเภทของข้อบกพร่อง (Defect Type) Fagan (1976) กำหนดประเภทของข้อบกพร่องในการออกแบบซอฟต์แวร์ไว้ 7 ประเภทดังนี้ (อ้างถึงใน O'Regan, 2002)

ตารางที่ 2-9 ตารางแสดงกลุ่มของข้อบกพร่องของการออกแบบซอฟต์แวร์ที่ Fagan กำหนด (อ้างถึงใน O'Regan, 2002)

การออกแบบ	ประเภท
ความสามารถในการใช้งาน (Usability)	UY
ความต้องการซอฟต์แวร์ (Requirements)	RQ
ตรรกะ (Logic)	LO
ตัวประสานระบบ (System Interface)	IS
ความน่าเชื่อถือ (Reliability)	RY
ความสามารถการบำรุงรักษา (Maintainability)	MN

ตารางที่ 2-9 (ต่อ) ตารางแสดงกลุ่มของข้อบกพร่องของการออกแบบซอฟต์แวร์ที่ Fagan กำหนด (อ้างถึงใน O'Regan, 2002)

การออกแบบ	ประเภท
การจัดการข้อผิดพลาด (Error Handling)	EH
อื่นๆ (Other)	OT

2.5.4 กลุ่มของข้อบกพร่อง (Defect Class) เป็นการแบ่งกลุ่มตามสาเหตุที่ทำให้เกิดข้อบกพร่อง โดยสามารถแบ่งออกได้ดังต่อไปนี้ (Strauss และ Ebenau, 1994)

1. สิ่งสูญหาย (Missing) คือผลิตภัณฑ์ที่นำมาตรวจสอบมีบางสิ่งที่จำเป็นขาดหายไป
2. สิ่งผิดพลาด (Wrong) คือผลิตภัณฑ์ที่นำมาตรวจสอบมีบางสิ่งที่ผิดพลาด หรือไม่มีความชัดเจน
3. สิ่งเพิ่มเติม (Extra) คือผลิตภัณฑ์ที่นำมาตรวจสอบมีบางสิ่งเพิ่มเข้ามา ซึ่งเป็นสิ่งที่ไม่จำเป็นต้องนำมาใช้ในการออกแบบจึงควรกำจัดออกไป

2.5.5 ระดับความรุนแรงของข้อบกพร่อง (Defect Severity) สามารถแบ่งระดับความรุนแรงได้ทั้งหมด 4 ระดับ (O'Regan, 2002; Strauss และ Ebenau, 1994)

1. สำคัญ (Major: M) เป็นข้อบกพร่องที่คาดว่าเป็นสาเหตุที่ทำให้ผลิตภัณฑ์ที่นำมาตรวจสอบเกิดความล้มเหลว หรือเบี่ยงเบนจากที่ได้กำหนดไว้
2. รุนแรงน้อย (Minor: m) เป็นข้อบกพร่องที่มีระดับความรุนแรงต่ำ ส่งผลในการลดระดับความมีประสิทธิผลของผลิตภัณฑ์ที่นำมาตรวจสอบ หรือทำให้เกิดความสับสนในรูปแบบของผลิตภัณฑ์ที่นำมาตรวจสอบ และส่งผลกระทบกับการดำเนินการ หรือการพัฒนาของผลิตภัณฑ์ที่นำมาตรวจสอบ
3. การปรับปรุงกระบวนการ (Process Improvement: PI) เป็นคำแนะนำเกี่ยวกับการปรับปรุงกระบวนการ เพื่อให้ผลิตภัณฑ์ที่จัดทำขึ้นนั้นมีคุณภาพมากยิ่งขึ้น
4. ตรวจสอบหาความจริง (Investigate: INV) เป็นสิ่งที่ไม่ชัดเจนว่าเป็นข้อบกพร่องหรือไม่

2.5.6 รายงานที่เกี่ยวข้องกับการตรวจสอบซอฟต์แวร์ (Inspection Report) ในการตรวจสอบซอฟต์แวร์นั้นมีการบันทึกข้อมูล และจัดทำรายงานต่างๆดังนี้ (Strauss และ Ebenau, 1994)

1. เอกสารแจ้งการประชุมตรวจสอบ (Inspection Meeting Notice) เป็นเอกสารแบบทางการที่จัดทำขึ้นเพื่อเชิญผู้เข้าร่วมในการตรวจสอบซอฟต์แวร์ ซึ่งเอกสารนี้จะแสดงวัน เวลา สถานที่ที่ใช้ในขั้นตอนประชุมตรวจสอบ ประเภทการทำประชุมตรวจสอบ เช่น การประชุมแนะนำการตรวจสอบ การตรวจสอบอีกครั้ง (Reinspect) ประเภทของการตรวจสอบซอฟต์แวร์ เช่น โปรแกรม กรอบการทดสอบ แผนการทดสอบ เอกสารแสดงการออกแบบซอฟต์แวร์เป็นต้น

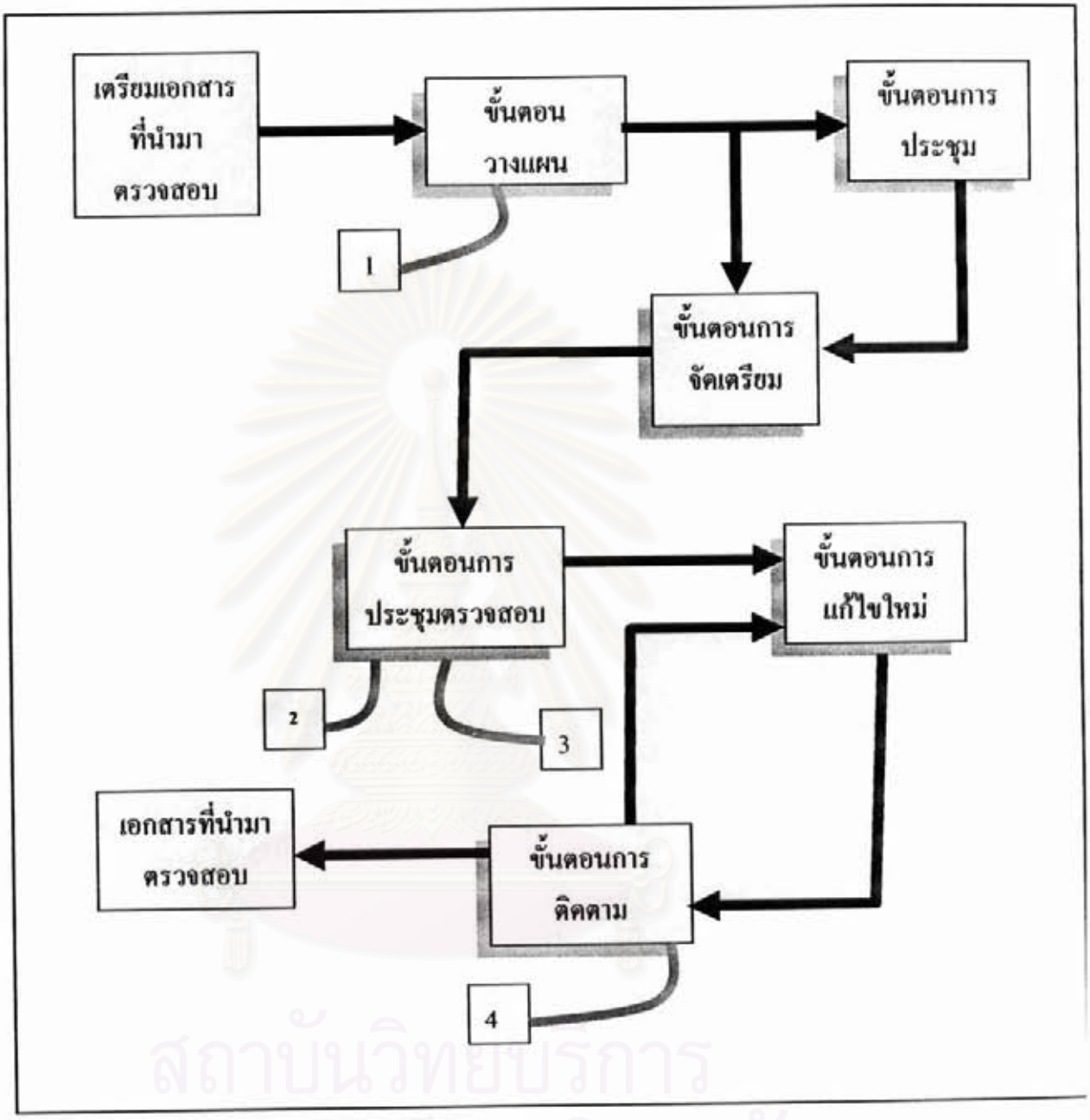
2. เอกสารบันทึกรายการข้อบกพร่อง (Inspection Defect List) เป็นเอกสารที่ผู้บันทึกใช้บันทึกรายการข้อบกพร่องโดยบันทึกตำแหน่งที่พบข้อบกพร่อง รายละเอียดของข้อบกพร่อง ประเภทของข้อบกพร่อง กลุ่มของข้อบกพร่อง และระดับความรุนแรงของข้อบกพร่อง เอกสารนี้จัดทำขึ้นเพื่อส่งให้กับผู้เขียนนำข้อบกพร่องเหล่านี้ไปแก้ไขต่อไป

3. เอกสารสรุปข้อบกพร่อง (Inspection Defect Summary) จัดทำขึ้นเพื่อรายงานจำนวนข้อบกพร่องที่พบ และจัดกลุ่มข้อบกพร่องนั้น โดยอาจมีการจัดกลุ่มข้อบกพร่องตามระดับความรุนแรงของข้อบกพร่อง ซึ่งเอกสารนี้ได้หลังจากทำขั้นตอนการประชุมตรวจสอบ

4. รายงานการจัดการการตรวจสอบ (Inspection Management Report) เป็นรายงานสถิติที่ได้จากการการตรวจสอบซอฟต์แวร์ เพื่อส่งให้กับผู้บริหารนำไปใช้ในการจัดการปรับปรุงการการตรวจสอบซอฟต์แวร์ต่อไป โดยเอกสารนี้ได้หลังจากขั้นตอนการประชุมตรวจสอบ

5. รายงานการจัดการการตรวจสอบที่มีการรับรอง (Inspection Management Report: Certified) จัดทำขึ้นเพื่อแจ้งผู้บริหารว่าได้แก้ไขเสร็จเรียบร้อยแล้ว และเป็นการรับรองว่าการตรวจสอบซอฟต์แวร์นั้นเสร็จเรียบร้อยแล้ว รายงานนี้ถูกทำหลังจากขั้นตอนการติดตาม

รูปที่ 2-3 เป็นการแสดงลำดับของขั้นตอนการตรวจสอบซอฟต์แวร์ ซึ่งบางกรณีที่คณะผู้ตรวจสอบมีความรู้ในเรื่องผลิตภัณฑ์ที่นำมาตรวจสอบแล้ว ก็ไม่จำเป็นที่จะต้องมีการอธิบายข้อมูลของผลิตภัณฑ์ที่นำมาตรวจสอบอีก จึงสามารถที่จะข้ามขั้นตอนการประชุมแนะนำไปที่ขั้นตอนการจัดเตรียมได้เลย และในขั้นตอนการติดตามนั้นถ้าตรวจสอบแล้วพบว่าผู้เขียนยังแก้ไขข้อบกพร่องไม่เรียบร้อย จะต้องกลับไปที่ขั้นตอนการแก้ไขใหม่เพื่อแก้ไขข้อบกพร่องอีกครั้ง และยังแสดงให้เห็นว่าในแต่ละขั้นตอนนั้นทำให้ได้เอกสารอะไรออกมาบ้าง



รูปที่ 2-3 แสดงสายงานกระบวนการตรวจสอบซอฟต์แวร์ (Strauss และ Ebenau, 1994)

หมายเหตุ

1 : เอกสารแจ้งการประชุมตรวจสอบ

2 : เอกสารบันทึกรายการข้อบกพร่อง

3 : เอกสารสรุปข้อบกพร่อง

4 : รายงานการจัดการการตรวจสอบ

2.6 เทคนิคการอ่านซอฟต์แวร์ (Software Reading Techniques)

หัวใจที่สำคัญที่สุดประการหนึ่งของการตรวจสอบซอฟต์แวร์อยู่ที่ขั้นตอนการจัดเตรียม (Laitenberger และคณะ, 2000) ซึ่งเป็นขั้นตอนที่ผู้ตรวจสอบแต่ละคนค้นหาข้อบกพร่องในเอกสารที่ต้องการตรวจสอบ ซึ่งเป็นวัตถุประสงค์หลักในการตรวจสอบซอฟต์แวร์เพื่อให้เอกสารนั้นมีความถูกต้อง ความต้องกัน (Consistency) กับเอกสารอื่นที่นำเสนอระบบเดียวกัน และความสามารถการบำรุงรักษา (Laitenberger และคณะ, 2000) ซึ่งวิธีการหนึ่งที่นำมาใช้เพื่อช่วยในการค้นหาข้อบกพร่องในขั้นตอนการจัดเตรียม คือการใช้เทคนิคการอ่านซอฟต์แวร์ (Porter และ Votta , 1994; Basili และคณะ, 1996; Miller และคณะ, 1998; Regnell และคณะ, 2000; Biffl และ Halling, 2000; Halling และ Biffl, 2001)โดย Basili และคณะ (1996) กล่าวว่าเทคนิคการอ่านซอฟต์แวร์เป็นเทคนิคสำหรับขั้นตอนการจัดเตรียม ที่ช่วยเพิ่มประสิทธิผลในการตรวจสอบซอฟต์แวร์โดยไม่จำกัดเฉพาะโปรแกรมเท่านั้น ยังสามารถนำเทคนิคการอ่านซอฟต์แวร์มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ และเอกสารอื่นๆได้อีกด้วย (Porter และคณะ,1995; Basili และคณะ, 1996; Fusaro และคณะ, 1997; Shull, 1998; Zhang และคณะ,1998) เทคนิคการอ่านซอฟต์แวร์ช่วยเพิ่มประสิทธิผลของการตรวจสอบซอฟต์แวร์ เนื่องจากได้จัดเตรียมคำแนะนำ (Guidelines) ให้กับผู้ตรวจสอบได้ใช้เป็นแนวทางในการค้นหาข้อบกพร่องในขั้นตอนการจัดเตรียม โดยในคำแนะนำมีคำอธิบายเกี่ยวกับขั้นตอนการค้นหาข้อบกพร่อง และวิธีในการพิจารณาเอกสารว่าควรสนใจตรงไหนของเอกสารเป็นพิเศษ ซึ่งคำแนะนำเหล่านี้จะช่วยในการระบุข้อบกพร่อง (Basili และคณะ, 1996)

2.6.1 ประเภทของเทคนิคการอ่านซอฟต์แวร์ สามารถแบ่งออกได้หลายประเภทตามรูปแบบของคำแนะนำที่แตกต่างกันดังนี้

1. เทคนิคเฉพาะกิจ (Ad-hoc) เป็นเทคนิคที่ง่ายที่สุดเนื่องจากไม่มีการเตรียมวิธีในการตรวจสอบซอฟต์แวร์ให้กับผู้ตรวจสอบซอฟต์แวร์ว่าต้องดำเนินการอย่างไร หรือต้องมองหาข้อบกพร่องตรงส่วนใดของผลิตภัณฑ์ที่นำมาตรวจสอบบ้างในขณะที่กำลังอ่านเอกสาร Laitenberger และคณะ (2000) กล่าวไว้ว่าวิธีนี้ผู้ตรวจสอบต้องเป็นผู้มีประสบการณ์เพื่อที่จะทราบได้ว่าต้องทำอย่างไรจึงจะสามารถค้นหาข้อบกพร่องที่มีอยู่ให้พบ

2. เทคนิคการอ่านบนพื้นฐานของเช็กลิสต์ (Checklist-Based Reading) หรือเรียกโดยย่อว่าเทคนิคซีบีอาร์ (CBR) เป็นเทคนิคที่นิยมนำมาใช้ในการตรวจสอบซอฟต์แวร์ (Porter และคณะ, 1995; Laitenberger และ Debaud, 2000; Biffl และ Halling, 2002; Sabaliauskaite และคณะ, 2002; Ciolkowski และคณะ, 2003; Sabaliauskaite, 2004) โดยเทคนิคนี้ได้จัดเตรียมคำแนะนำที่อยู่ในรูปแบบของเช็กลิสต์ (Checklist) ซึ่งเป็นรายการคำถามแบบใช่/ไม่ใช่ (Yes/No Questions) เพื่อ

ช่วยให้ผู้ตรวจสอบทราบว่าจะต้องมองหาข้อบกพร่องที่ส่วนใดในเอกสาร (Chernak, 1996) ซึ่งการตั้ง จะต้องตั้งคำถามให้เหมาะสมกับเอกสารที่นำไปตรวจสอบ เพื่อให้ผู้ตรวจสอบสามารถค้นหาข้อบกพร่องได้ เช่น การตรวจสอบโปรแกรมเชิงวัตถุ และการตรวจสอบโปรแกรมแบบสัญนิยมจะมีโครงสร้างของโปรแกรมที่แตกต่างกัน ดังนั้นคำถามในเช็กลิสต์ที่นำมาใช้ในการตรวจสอบจะต้องแตกต่างกันด้วย โดยในการตอบคำถามนั้นถ้าคำตอบเป็นปฏิเสธแสดงว่ามีข้อบกพร่อง และผู้ตรวจสอบจะต้องใส่รายละเอียดเกี่ยวกับข้อบกพร่องที่พบลงในเอกสารบันทึกข้อบกพร่อง (Inspection Defect List) สำหรับลักษณะของคำถามในเช็กลิสต์ เป็นคำถามแบบทั่วไปไม่มีความเป็นเฉพาะเจาะจงสำหรับแต่ละสภาพแวดล้อม

แม้ว่าเทคนิคชีบีอาร์เป็นวิธีที่รองรับการตรวจสอบซอฟต์แวร์ได้มากกว่าเทคนิคเฉพาะกิจแต่เทคนิคนี้ยังมีข้อด้อยอยู่โดย Laitenberger, Atkinson, Schlich และคณะ (2000) กล่าวไว้ว่า (1) เช็กลิสต์มักขึ้นกับข้อมูลความผิดพลาดที่เกิดขึ้นในอดีต เป็นการนำประสบการณ์ข้อบกพร่องขององค์กรมาสร้างเป็นเช็กลิสต์ ถ้าไม่มีประสบการณ์ในอดีตมาก่อน คำถามของเช็กลิสต์มักได้มาจากงานเขียนที่ได้รับการตีพิมพ์ออกมา ซึ่งทั้ง 2 กรณีนี้ผู้ตรวจสอบจะไม่ให้ความสนใจประเภทของข้อบกพร่องที่ไม่เคยถูกค้นพบในอดีต ทำให้พลาดข้อบกพร่องบางอย่างหรือกลุ่มของข้อบกพร่องทั้งกลุ่มไป (2) เช็กลิสต์มักประกอบด้วยรายการคำถามที่มากเกินไป และไม่ได้บอกวิธีในการตอบคำถามดังนั้นจึงไม่ชัดเจนสำหรับผู้ตรวจสอบในการนำไปใช้ (3) ผู้ตรวจสอบต้องตรวจสอบข้อมูลทุกอย่างในเอกสารเพื่อหาข้อบกพร่องที่อาจเกิดขึ้นได้ ซึ่งเป็นสาเหตุให้ผู้ตรวจสอบรับภาระจากรายละเอียดที่ไม่จำเป็นจำนวนมาก แต่จากปัญหาในเรื่องของการที่เช็กลิสต์ประกอบด้วยรายการคำถามที่มากเกินไป Gilb และ Graham (1993) จึงได้แนะนำว่าเช็กลิสต์นั้นไม่ควรมีความยาวเกิน 1 หน้ากระดาษ นั่นหมายถึงจำนวนคำถามไม่ควรมีเกิน 25 ข้อ

Porter และคณะ (1995) ได้วิจัยเพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคชีบีอาร์และเทคนิคเฉพาะกิจ พบว่าเทคนิคเฉพาะกิจมีประสิทธิภาพ และประสิทธิผลสูงกว่าเทคนิคชีบีอาร์

3. เทคนิคการอ่านบนพื้นฐานของสถานการณ์ (Scenario-Based Reading) หรือเรียกโดยย่อว่าเทคนิคเอสบีอาร์ (SCR) เป็นเทคนิคที่จัดเตรียมคำแนะนำในรูปแบบที่เรียกว่าสถานการณ์ (Scenario) ให้กับผู้ตรวจสอบ เพื่อบอกว่าต้องตรวจสอบอะไรและตรวจสอบอย่างไร โดยสถานการณ์ (Scenario) จะประกอบด้วย 3 ส่วน (Porter และ Votta, 1994) คือ (1) คำแนะนำเกี่ยวกับหน้าที่ความรับผิดชอบ และสิ่งที่ต้องเกี่ยวข้องในแต่ละหน้าที่ (2) วิธีในการคัดเลือกข้อมูลที่สำคัญออกมาจากเอกสารที่ตรวจสอบ (3) คำถามที่สามารถตอบได้ด้วยข้อมูลที่คัดออกมา

เทคนิคเอสบีอาร์สามารถแบ่งออกได้ 5 ประเภท โดยมีความแตกต่างกันที่รูปแบบของสถานการณ์ (Scenario) ซึ่งได้ถูกเสนอขึ้นมาอย่างหลากหลาย เพื่อความเหมาะสมกับเอกสารที่นำมาตรวจสอบและวัตถุประสงค์ในการตรวจสอบ

ก. เทคนิคการอ่านบนพื้นฐานของข้อบกพร่อง (Defect-Based Reading) หรือเรียกโดยย่อว่าเทคนิคดีบีอาร์ (DBR) Porter และคณะ (1995) อธิบายสถานการณ์ (Scenario) บนพื้นฐานของการจัดกลุ่มข้อบกพร่องที่กำหนดขึ้นสำหรับใช้ในการตรวจสอบซอฟต์แวร์ โดยสถานการณ์ (Scenario) เหล่านี้ได้มาจากประเภทของข้อบกพร่อง ซึ่งประกอบด้วยรายการคำถามที่เป็นแบบเฉพาะเจาะจง ผู้ตรวจสอบต้องตอบคำถามไปพร้อมกับที่กำลังอ่านเอกสารที่ตรวจสอบ โดยสถานการณ์ (Scenario) จะมุ่งเน้นไปที่การค้นหาข้อบกพร่องว่าเป็นข้อบกพร่องประเภทใด

ข. เทคนิคการอ่านบนพื้นฐานของมุมมอง (Perspective Based Reading) หรือเรียกโดยย่อว่าเทคนิคพีบีอาร์ (PBR) Basili และคณะ (1996) ได้เสนอเทคนิคนี้ โดยสถานการณ์ (Scenario) ของเทคนิคนี้รองรับการตรวจสอบเอกสารจากมุมมองของผู้ถือผลประโยชน์ร่วม (Stakeholders) แต่ละคน วัตถุประสงค์คือค้นหาข้อบกพร่องที่ต่างมุมมองกันเพื่อให้ได้ข้อบกพร่องที่ครบถ้วนจากหลายๆมุมมอง โดยมุ่งเน้นที่มุมมองของผู้ใช้ซอฟต์แวร์ ผู้ออกแบบซอฟต์แวร์ และผู้ทดสอบซอฟต์แวร์ (Basili และคณะ, 1996; Laitenberger, 2000; Regnell และคณะ, 2000; Sabaliauskaite และคณะ, 2003; Thelin, 2002) เทคนิคนี้ได้แนวความคิดจากบทความเกี่ยวกับการตรวจสอบซอฟต์แวร์ในอดีต เช่น Fagan (1976) รายงานว่าส่วนหนึ่งของโปรแกรมควรถูกตรวจสอบโดยผู้ทดสอบระบบ ขณะที่ Fowler (1986) แนะนำว่าคณะผู้ตรวจสอบแต่ละคนควรมีมุมมองเฉพาะตัวสำหรับแต่ละผลิตภัณฑ์ที่นำมาตรวจสอบ นอกจากนี้ Basili และคณะ (1996), Laitenberger และคณะ (2000), Biffl และ Halling (2002) กล่าวว่าผลิตภัณฑ์ที่นำมาตรวจสอบควรได้รับการตรวจสอบจากมุมมองของผู้ถือผลประโยชน์ร่วมที่แตกต่างกัน ซึ่งมุมมองนั้นขึ้นกับบทบาทของแต่ละคนในการพัฒนาซอฟต์แวร์

มีงานวิจัยที่เกี่ยวข้องกับการวัดประสิทธิผลของการนำเทคนิคพีบีอาร์มาใช้ในการตรวจสอบซอฟต์แวร์ โดยการเปรียบเทียบกับเทคนิคซีบีอาร์ แต่จากงานวิจัยเหล่านี้ยังไม่สามารถระบุได้แน่ชัดว่าเทคนิคพีบีอาร์มีประสิทธิผลมากกว่าเทคนิคซีบีอาร์ เนื่องจากมีหลายงานวิจัยที่แสดงให้เห็นว่าเทคนิคพีบีอาร์มีประสิทธิผลมากกว่าเทคนิคซีบีอาร์ เช่น จากงานวิจัยของ Basili และคณะ (1996) กับ Porter และ Votta (1998) ที่ได้ทดลองโดยการตรวจสอบเอกสารแสดงความต้องการซอฟต์แวร์ งานวิจัยของ Laitenberger และคณะ (2000) ทดลองโดยการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล และงานวิจัยของ

Laitenberger และคณะ (2001) ทดลองโดยการตรวจสอบโปรแกรม ซึ่งผลของการทดลองเหล่านี้แสดงให้เห็นว่าเทคนิคพีบีอาร์มีประสิทธิผลในการตรวจสอบซอฟต์แวร์มากกว่าเทคนิคซีบีอาร์ แต่มีหลายงานวิจัยที่ไม่สามารถแสดงความแตกต่างอย่างมีนัยสำคัญของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ระหว่างการนำ 2 เทคนิคนี้ไปใช้ เห็นได้จากงานวิจัยของ Fusaro และคณะ (1997) ประเมินเทคนิคในการตรวจสอบความต้องการของลูกค้า และงานวิจัยของ Dunsmore และคณะ (2003) ประเมินเทคนิคโดยการตรวจสอบโปรแกรม นอกจากนี้งานวิจัยของ Sabaliauskaite (2004) ทดลองโดยการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

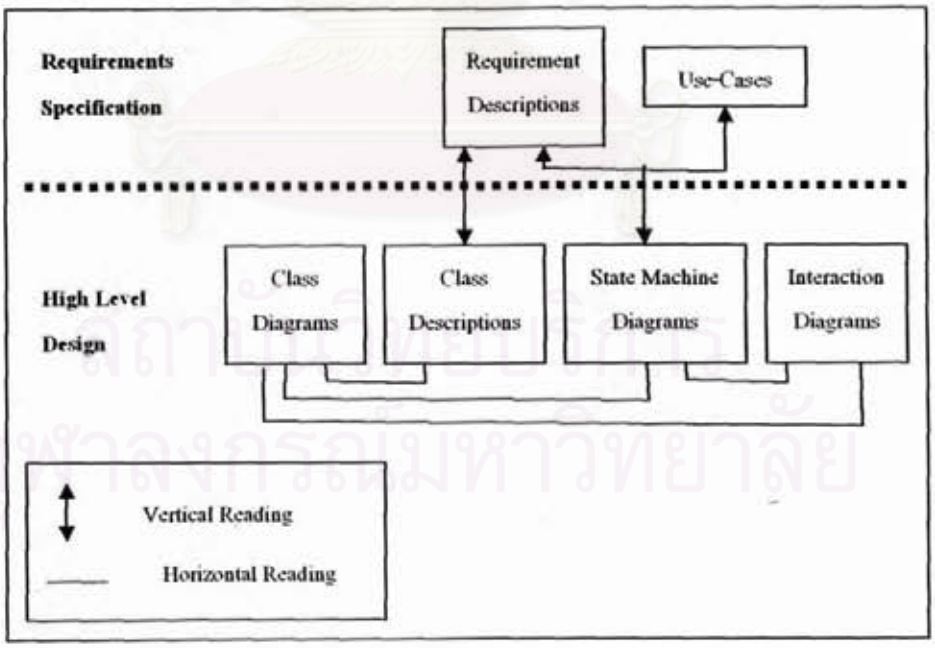
ค. เทคนิคการอ่านบนพื้นฐานของสถานการณ์ที่อยู่บนพื้นของฟังก์ชันพอยต์ (Scenario-Based Reading Techniques Based on Function Points) โดยเทคนิคนี้ Cheng และ Jeffery (1996) ได้วางรากฐานการพัฒนาสถานการณ์ (Scenario) บนพื้นฐานของการวิเคราะห์แบบฟังก์ชันพอยต์ (Function Point Analysis) โดยการวิเคราะห์แบบฟังก์ชันพอยต์ได้นิยามระบบซอฟต์แวร์ (Software System) ในรูปขององค์ประกอบแบบฟังก์ชันพอยต์ (Function Point Element) เช่น ข้อมูลเข้า (Inputs), แฟ้มข้อมูล (Files), อินไควรี (Inquiries) และข้อมูลออก (Outputs) ซึ่งสถานการณ์ของฟังก์ชันพอยต์ (Function Point Scenario) จะประกอบด้วยรายการคำถามที่เป็นองค์ประกอบแบบฟังก์ชันพอยต์ (Function Point Element)

ง. เทคนิคการอ่านบนพื้นฐานของการใช้ (Usage-Based Reading) หรือเรียกโดยย่อว่าเทคนิคยูบีอาร์ (UBR) เป็นเทคนิคที่สถานการณ์ (Scenario) มุ่งประเด็นไปที่แรงงาน (Effort) ที่ใช้ในการค้นหาข้อบกพร่อง ด้วยเหตุผลที่ว่าข้อบกพร่องแต่ละประเภทมีความสำคัญที่แตกต่างกัน จึงจัดลำดับความสำคัญในการค้นหาข้อบกพร่องว่าข้อบกพร่องใดที่มีความวิกฤต โดยดูจากมุมมองของผู้ใช้ (Thelin, 2002) จากงานวิจัยของ Thelin และคณะ (2003) เปรียบเทียบเพื่อดูประสิทธิภาพและประสิทธิผลของเทคนิคยูบีอาร์กับเทคนิคซีบีอาร์ ซึ่งผลการทดลองแสดงให้เห็นว่าเทคนิคยูบีอาร์มีประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์มากกว่าเทคนิคซีบีอาร์

จ. เทคนิคการอ่านบนพื้นฐานของความสามารถการติดตาม (Traceability-Based Reading) หรือเรียกโดยย่อว่าเทคนิคทีบีอาร์ (TBR) เป็นเทคนิคที่สถานการณ์ (Scenario) ถูกจัดทำขึ้นเพื่อใช้ในการติดตามว่าการออกแบบนั้นตรงตามความต้องการทางซอฟต์แวร์ที่กำหนดไว้หรือไม่ และยังติดตามว่าในการออกแบบนั้นแต่ละแผนภาพสื่อถึงระบบเดียวกันหรือไม่ โดยเป็นการค้นหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ ที่พัฒนาตามลักษณะของซอฟต์แวร์เชิงวัตถุ (Object-Oriented) โดยเฉพาะ (Travassos และคณะ, 1999) ซึ่งเทคนิคนี้เรียกอีกชื่อหนึ่งว่าเทคนิคการอ่านเชิงวัตถุ (Object-Oriented Reading Techniques) หรือเรียกโดยย่อว่าเทคนิค

โอโออาร์ที (OORT) โดย Travassos และคณะ (1999) ได้เสนอเทคนิคนี้ตั้งแต่ปี 1998 โดยกล่าวว่าเทคนิคนี้มีแนวคิดหลักเพื่อให้มั่นใจในเรื่องความถูกต้องสอดคล้องกันของเอกสารทั้งในเอกสารเดียวกัน และระหว่างเอกสารที่เสนอระบบเดียวกันเพื่อให้มั่นใจในความถูกต้องและความสมบูรณ์ (Completeness) ปัจจุบันเทคนิคโอโออาร์ทีเป็นรุ่น 3.0

โดย Shull และคณะ (1999) กล่าวไว้ว่าปัญหาหลัก 2 ข้อที่เป็นตัวผลักดันให้เกิดแนวคิดในการทำเทคนิคโอโออาร์ทีคือ (1) เพื่อตรวจสอบว่าการออกแบบแต่ละส่วนในเอกสารแสดงการออกแบบซอฟต์แวร์อธิบายข้อมูลออกมาได้ตรงกันหรือไม่ (2) เพื่อตรวจสอบว่าเอกสารแสดงการออกแบบซอฟต์แวร์ที่จัดทำขึ้นนั้นถูกต้องตรงตามความต้องการของลูกค้าหรือไม่ จากคำถามเหล่านี้จึงได้มีการนิยามวิธีการจับคู่แผนภาพ เพื่อนำมาเปรียบเทียบโดยแบ่งออกเป็น 2 ส่วนหลักคือ (1) การอ่านตามแนวนอน (Horizontal Reading) ช่วยในการตรวจสอบว่าแต่ละแผนภาพมีการอธิบายระบบเดียวกัน กล่าวคือแต่ละแผนภาพที่ออกแบบนั้นมีความสอดคล้องตรงกันหรือไม่ (2) การอ่านตามแนวตรง (Vertical Reading) ช่วยในการตรวจสอบว่าที่ออกแบบนั้นเป็นการนำเสนอระบบที่ถูกต้อง กล่าวคือในออกแบบได้ตรงตามที่ความต้องการทางซอฟต์แวร์กำหนดไว้ ซึ่งในการอธิบายระบบดูได้จากความต้องการของลูกค้า และแผนภาพยูสเคส (Use Case Diagram) แสดงดังรูปที่ 2-4



รูปที่ 2-4 แสดงรูปแบบการอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ของเทคนิคโอโออาร์ที (Travassos และคณะ, 1999)

เทคนิคโอโออาร์ทีเป็นการนำ (1) แผนภาพยูสเคส (Use Case Diagram) (2) แผนภาพคลาส (Class Diagram) (3) แผนภาพซีเควนซ์ (Sequence Diagram) (4) แผนภาพสถานะ (State Machine Diagram) (5) เอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) และ (6) คำอธิบายคลาส (Class Description) มาตรวจสอบหาข้อบกพร่องโดยการเปรียบเทียบระหว่าง 2 แผนภาพหรือเอกสาร ซึ่งเทคนิคโอโออาร์ทีประกอบด้วย 7 ขั้นตอน ในการอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ดังต่อไปนี้

(1) เทคนิคโอโออาร์ที-1 ขั้นตอนของการเปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพคลาส

(2) เทคนิคโอโออาร์ที-2 ขั้นตอนของการเปรียบเทียบระหว่างแผนภาพสถานะและคำอธิบายคลาส

(3) เทคนิคโอโออาร์ที-3 ขั้นตอนของการเปรียบเทียบระหว่างแผนภาพสถานะ และแผนภาพซีเควนซ์

(4) เทคนิคโอโออาร์ที-4 ขั้นตอนของการเปรียบเทียบระหว่างแผนภาพคลาส และคำอธิบายคลาส

(5) เทคนิคโอโออาร์ที-5 ขั้นตอนของการเปรียบเทียบระหว่างคำอธิบายคลาส และเอกสารอธิบายความต้องการซอฟต์แวร์

(6) เทคนิคโอโออาร์ที-6 ขั้นตอนของการเปรียบเทียบระหว่างแผนภาพซีเควนซ์ และแผนภาพยูสเคส

(7) เทคนิคโอโออาร์ที-7 ขั้นตอนของการเปรียบเทียบระหว่างแผนภาพสถานะ และเอกสารอธิบายความต้องการซอฟต์แวร์ / แผนภาพยูสเคส

เทคนิคโอโออาร์ทีมีการแบ่งกลุ่มของข้อบกพร่องตามสาเหตุที่ทำให้เกิดข้อบกพร่อง ซึ่งเพิ่มเติมจากที่ Fagan (1976) กำหนดไว้ 2 กลุ่มคือ (1) ความไม่สอดคล้องกัน (Inconsistency) คือแผนภาพแต่ละอันแสดงข้อมูลอย่างเดียวกันออกมาได้ไม่ตรงกัน (2) ความกำกวม (Ambiguity) คือการออกแบบไม่ชัดเจน เนื่องจากไม่เข้าใจความหมายของข้อมูลหรือความต้องการของลูกค้าที่ได้รับ งานวิจัยที่วัดประสิทธิภาพและประสิทธิผลของเทคนิคโอโออาร์ทีคือ งานวิจัยของ Melo และคณะ (2001) ซึ่งจากการทดลองสามารถสรุปผลได้ว่า เทคนิคโอโออาร์ทีสามารถค้นหาข้อบกพร่องได้เป็นจำนวนมาก และช่วยในการกำจัดผลบวกปลอม (False Positive) กล่าวคือเป็นการกำจัดข้อบกพร่องที่ผู้ตรวจสอบบันทึกไว้แต่แท้จริงแล้วไม่ได้เป็นข้อบกพร่อง ทั้งนี้ Conradi และคณะ (2003) ได้ทดลองเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีกับเทคนิคอาร์แอนด์ไอ (R&I Techniques) ซึ่งเป็นเทคนิคที่ใช้ในการตรวจสอบซอฟต์แวร์

ของบริษัทอีริคสันประเทศนอร์เวย์ โดยผลการทดลองแสดงให้เห็นว่าเทคนิคโอโออาร์ที่สามารถค้นหาข้อบกพร่องได้มากกว่าเทคนิคอาร์แอนด์ไอ

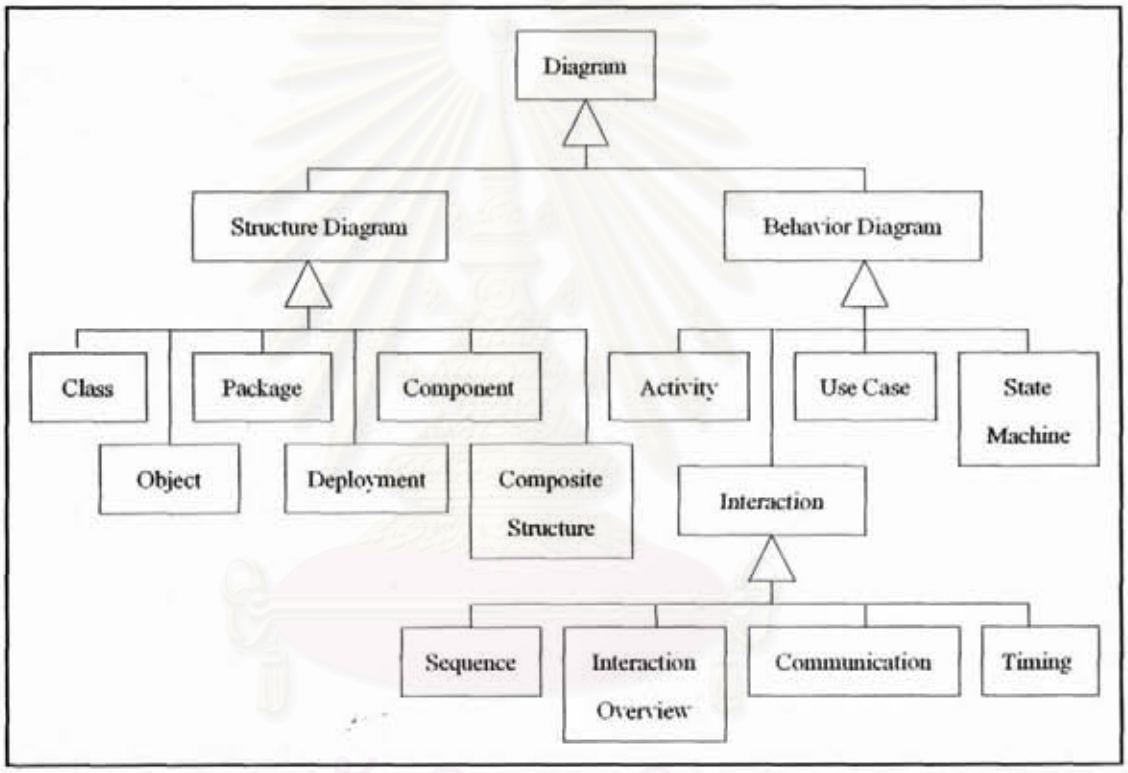
จากงานวิจัยต่างๆข้างต้นมีการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล โดยมีการนำเทคนิคซีบีอาร์ซึ่งเป็นเทคนิคมาตรฐานที่ใช้ในอุตสาหกรรม (Porter และคณะ (1995) เปรียบเทียบกับเทคนิคเฉพาะกิจ เทคนิคพีบีอาร์ และเทคนิคยูบีอาร์ และมีการเปรียบเทียบระหว่างเทคนิคโอโออาร์ที่ซึ่งเป็นเทคนิคที่ใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลโดยเฉพาะ กับเทคนิคอาร์แอนด์ไอ (R&I Techniques) ซึ่งเป็นเทคนิคที่ใช้ในการตรวจสอบซอฟต์แวร์ของบริษัทอีริคสัน ประเทศนอร์เวย์ ดังนั้นผู้วิจัยจึงสนใจที่จะเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล ระหว่างเทคนิคซีบีอาร์กับเทคนิคโอโออาร์ที เพื่อเพิ่มเติมข้อมูลงานวิจัยที่เกี่ยวกับการเปรียบเทียบประสิทธิภาพและประสิทธิผลของการนำเทคนิคการอ่านซอฟต์แวร์มาใช้ในการตรวจสอบซอฟต์แวร์เชิงวัตถุ

2.7 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 ซึ่งถ้าเลือกวิธีในการวิเคราะห์ออกแบบของใครก็จำเป็นต้องใช้วิธีการนำเสนอผลลัพธ์ของการวิเคราะห์ออกแบบตามที่ผู้สร้างกำหนดด้วย ซึ่งแต่ละวิธีต่างมีการกำหนดสัญลักษณ์รูปภาพ (Notation) ที่ต่างกันไป ดังนั้นจึงได้มีการรวมวิธีการต่างๆเข้าไว้ด้วยกัน โดยในปี 1994 ได้นำวิธี Booch ของ Grady Booch และวิธีโอเอ็มที (OMT) ของ James Rumbaugh มารวมกัน ซึ่งเป็นการพัฒนาซอฟต์แวร์เชิงวัตถุที่เป็นหนึ่งเดียวกัน (Unified Method)

ต่อมาในปี 1995 ได้นำโอโอเอสอี (OOSE) ซึ่งเป็นกระบวนการของ Ivar Jacobson เข้ามาร่วมกับโครงการนี้ ซึ่งเป็นการสร้างภาษาแบบจำลองขึ้นใหม่เรียกว่าภาษายูเอ็มแอล (Unified Modeling Language: UML) เนื่องจากวิธีการพัฒนาซอฟต์แวร์เชิงวัตถุของทั้ง 3 คนมีชื่อเสียงอยู่แล้วในช่วงเวลานั้น ดังนั้นภาษายูเอ็มแอลที่ถูกพัฒนาขึ้นนี้จึงกลายเป็นที่นิยมใช้กันอย่างแพร่หลาย (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544) โดยในปัจจุบันแผนภาพยูเอ็มแอลรุ่นที่ถูกเผยแพร่ออกสู่สาธารณชนคือรุ่นที่ 2.0 ซึ่งถูกพัฒนาโดยองค์กรโอเอ็มจี (OMG) ในปี 2003 ซึ่งในรุ่นนี้ประกอบด้วยแผนภาพทั้งหมด 13 แผนภาพ (Dennis และคณะ, 2005)



รูปที่ 2-5 แสดงแผนภาพทั้งหมดของยูเอ็มแอลรุ่น 2.0 (Dennis และคณะ, 2005)

2.7.1 แผนภาพยูเอ็มแอลรุ่น 2.0 สาเหตุที่ยูเอ็มแอลต้องมีแผนภาพที่หลากหลาย เนื่องจากระบบที่มีความซับซ้อนมากๆนั้น ยังไม่มีวิธีการออกแบบระบบที่สามารถใช้เพียงแผนภาพเดียว แล้วสามารถให้มุมมองได้ทั้งหมด จึงต้องมีการเลือกแผนภาพหลายๆแผนภาพเพื่อให้สามารถเสนอมุมมองได้ครบถ้วน ดังนั้นนอกจากผู้ออกแบบและผู้พัฒนาจะต้องทำความเข้าใจกับองค์ประกอบของยูเอ็มแอลแล้ว ยังต้องทำความเข้าใจกับวัตถุประสงค์ของแต่ละแผนภาพ เพื่อให้สามารถเลือกใช้งานได้อย่างถูกต้อง และเหมาะสมกับมุมมองที่ต้องการนำเสนอ ซึ่งแผนภาพของยูเอ็มแอลรุ่น 2.0 มีทั้งหมด 13 แผนภาพ (Dennis และคณะ, 2005) ดังต่อไปนี้

1. แผนภาพคลาส (Class Diagrma) 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. แผนภาพคอมโพซิทสตรักเจอร์ (Composite Structure Diagram) เป็นแผนภาพใหม่ที่เพิ่มเข้ามาในแผนภาพยูเอ็มแอลรุ่น 2.0 ใช้เมื่อโครงสร้างภายในของคลาสมีความซับซ้อน โดยใช้เพื่อจำลองความสัมพันธ์ระหว่างส่วนของคลาส

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

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

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

10. แผนภาพแสดงการโต้ตอบ (Interaction Overview Diagram) ช่วยให้นักวิเคราะห์ทำความเข้าใจยูสเคสที่มีความซับซ้อน โดยการให้คำอธิบายสายงานของกระบวนการ ประโยชน์ที่สำคัญของแผนภาพแสดงการโต้ตอบ คือสามารถจำลองลำดับของสายงานได้ง่าย อย่างไรก็ตามสามารถใช้แผนภาพกิจกรรม และยูสเคสแทนที่ได้

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

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

13. แผนภาพยูสเคส (Use Case Diagram) ใช้จำลองการโต้ตอบของข้อมูลและสิ่งแวดล้อมของข้อมูล โดยสิ่งแวดล้อมของข้อมูลนั้นรวมถึงผู้ใช้ขั้นปลาย (End User) และระบบภายนอกที่โต้ตอบกับข้อมูล วัตถุประสงค์หลักของแผนภาพยูสเคส คือเพื่อทำความเข้าใจความต้องการของลูกค้าและแสดงหน้าที่ที่ระบบต้องทำได้ ซึ่งถือว่าแผนภาพนี้เป็นสิ่งสำคัญสำหรับการวิเคราะห์และออกแบบระบบเชิงวัตถุ

ในการใช้งานแต่ละแผนภาพนอกจากต้องทำความเข้าใจกับวัตถุประสงค์ของการใช้งานแผนภาพแล้ว ยังต้องเข้าใจส่วนประกอบของแต่ละแผนภาพด้วย ซึ่งงานวิจัยนี้เป็นการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล ซึ่งในการทดลองจะตรวจสอบ 4 แผนภาพเท่านั้น เนื่องจากเทคนิคโอโออาร์ทีถูกจัดทำขึ้นตั้งแต่ปี 1998 จึงถูกออกแบบเพื่อนำมาใช้ในการตรวจสอบแผนภาพยูเอ็มแอลรุ่น 1.1 ดังนั้นงานวิจัยนี้จึงมุ่งประเด็นไปที่แผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะตามที่เทคนิคโอโออาร์ทีกำหนดให้ตรวจสอบเท่านั้น

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

1. แอกเตอร์ (Actor) คือบุคคลหรือระบบภายนอกที่มีส่วนเกี่ยวข้องกับระบบที่พัฒนา (Dennis และคณะ, 2005) โดยในการค้นหาแอกเตอร์ของระบบคือ ดูว่ามีใครเป็นผู้ใช้หรือดูแลระบบ อุปกรณ์ฮาร์ดแวร์ที่เชื่อมต่อระบบ และระบบภายนอกที่เชื่อมต่อกับระบบที่เราพัฒนา ซึ่งชื่อของแอกเตอร์จะแสดงบทบาทและหน้าที่ที่มีต่อระบบ ในการเขียนแอกเตอร์นั้นจะแทนด้วยสัญลักษณ์รูปคน (Stick Man) ดังรูปที่ 2-6 โดยวางไว้นอกขอบเขตของระบบ

รูปที่ 2-6 แสดงแอกเตอร์ (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

2. ยูสเคส เป็นฟังก์ชันการทำงานต่างๆที่ซอฟต์แวร์ต้องทำได้ทั้งหมดจึงจะถือว่าซอฟต์แวร์ไม่มีข้อผิดพลาด โดยยูสเคสต้องถูกกระทำโดยแอกเตอร์ คือมีการส่งและรับข้อมูลระหว่างแอกเตอร์ (Dennis และคณะ, 2005) และในบางกรณีอาจมียูสเคสที่ไม่ได้ถูกกระทำโดยแอกเตอร์ แต่อาจมีความสัมพันธ์ร่วมกับยูสเคสอื่นๆที่มีความสัมพันธ์โดยตรงกับแอกเตอร์แทน ซึ่งในการเขียนยูสเคสจะแสดงด้วยสัญลักษณ์รูปวงรี ดังรูปที่ 2-7 โดยวางไว้ในขอบเขตของระบบ

รูปที่ 2-7 แสดงยูสเคส (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

3. ขอบเขตระบบ เป็นการแสดงขอบเขตการทำงานของระบบ (Dennis และคณะ, 2005) โดยสามารถแสดงได้ด้วยสัญลักษณ์รูปสี่เหลี่ยม ซึ่งข้างในขอบเขตนั้นจะประกอบด้วยยูสเคสต่างๆของระบบ และมีชื่อระบบเขียนอยู่ด้านบนหรือข้างในรูปสี่เหลี่ยม ดังรูปที่ 2-8

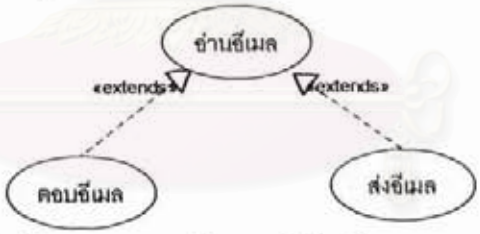
รูปที่ 2-8 แสดงขอบเขตของระบบ (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

4. ความสัมพันธ์ในแผนภาพยูสเคส สามารถแบ่งออกได้ 4 ประเภทคือ

ก. ความสัมพันธ์แบบเจเนอรัลไลเซชัน (Generalization Relationship) เป็นความสัมพันธ์ระหว่างแอกเตอร์ด้วยกันหรือยูสเคสด้วยกัน กล่าวคือ ถ้ามีหลายแอกเตอร์ในระบบซึ่งมีคุณสมบัติคล้ายกัน สามารถแยกออกมาเป็นอีกแอกเตอร์หนึ่ง ที่รวมบทบาทที่เหมือนกันของแต่ละแอกเตอร์ซึ่งถือว่าเป็นคุณสมบัติพื้นฐาน โดยแอกเตอร์พื้นฐานจะถูกสืบทอด (Inherit) จากแอกเตอร์อื่นๆ โดยสัญลักษณ์ที่ใช้คือลูกศรหัวโปร่งชี้จากแอกเตอร์หนึ่งไปยังแอกเตอร์ที่ถูกสืบทอด ดังรูปที่ 2-9

รูปที่ 2-9 แสดงความสัมพันธ์แบบเจเนอรัลไลเซชัน (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

ข. ความสัมพันธ์แบบขยาย (Extend Relationship) เป็นความสัมพันธ์ระหว่างยูสเคส กล่าวคือทำยูสเคส A แล้วอาจทำยูสเคส B ต่อหรือไม่ทำก็ได้ เช่นทำยูสเคสอ่านอีเมลแล้ว อาจทำยูสเคสตอบอีเมลหรือยูสเคสส่งอีเมล (E-Mail) ด้วยก็ได้เป็นต้น โดยสัญลักษณ์ที่ใช้คือลูกศรเส้นประที่ชี้จากยูสเคสแรก ไปยังยูสเคสที่ถูกช่วยเหลือหรือถูกขยาย โดยมีคำว่าเอ็กซ์เทนด์ (Extend) อยู่ที่กึ่งกลางลูกศร ดังรูปที่ 2-10



รูปที่ 2-10 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 2-11

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

ง. ความสัมพันธ์แบบแอสโซซิเอชัน (Association Relationship) เป็นความสัมพันธ์ระหว่างยูสเคสกับแอกเตอร์ กล่าวคือแอกเตอร์มีการโต้ตอบกับยูสเคส (Dennis และคณะ, 2005) เช่น แอกเตอร์แลกเปลี่ยนข้อมูลข่าวสารกับระบบที่พัฒนาเป็นต้น ใช้สัญลักษณ์เส้นตรงดังรูปที่ 2-12

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

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

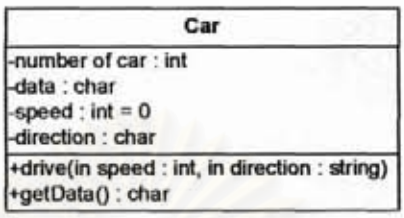
1. คลาส คือกลุ่มของอ็อบเจ็กต์ ดังนั้นการที่จะหาคลาสของอ็อบเจ็กต์ได้ จะต้องจัดหมวดหมู่ของอ็อบเจ็กต์หลายๆอ็อบเจ็กต์ได้ โดยคลาสในระบบอาจนำมาจากคลาสไลบรารีที่ผู้พัฒนาอื่นสร้างไว้ อุปกรณ์ต่างๆที่ระบบต้องควบคุม เช่น คลาสเครื่องพิมพ์ เป็นต้น หรือแอกเตอร์ในแผนภาพยูสเคส โดยรายละเอียดของคลาสมีดังต่อไปนี้ (Dennis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

ก. ชื่อคลาส (Class Name) ในการตั้งชื่อคลาสควรตั้งชื่อให้สื่อความหมาย

ข. แอตทริบิวต์ (Attribute) เป็นการนำเสนอคุณสมบัติที่อธิบายสถานะของอ็อบเจ็กต์ โดยประกอบด้วยชนิดการเข้าถึง (Visibility) ของแอตทริบิวต์ ชื่อของแอตทริบิวต์ ประเภทของแอตทริบิวต์ ซึ่งอยู่ต่อจากเครื่องหมายมหัพภาคคู่ (:) และค่าเริ่มต้นของแอตทริบิวต์ ซึ่งอาจมีหรือไม่มีก็ได้แต่ถ้ามีจะอยู่ต่อจากเครื่องหมายเท่ากับ

ค. โอเปอร์เรชัน (Operation) เป็นการแสดงการกระทำของคลาส โดยประกอบด้วยชนิดการเข้าถึงโอเปอร์เรชัน ชื่อของโอเปอร์เรชัน พารามิเตอร์ (Parameter) ที่จำเป็น

ต่อการทำงานของโอเปอร์เรชัน ซึ่งพารามิเตอร์จะประกอบด้วยชื่อพารามิเตอร์ ตามด้วยเครื่องหมายมหัพภาคคู่ และประเภทของพารามิเตอร์ตามลำดับ ประเภทของค่าที่ส่งคืน (Return Type) ซึ่งจะเขียนตามเครื่องหมายมหัพภาคคู่



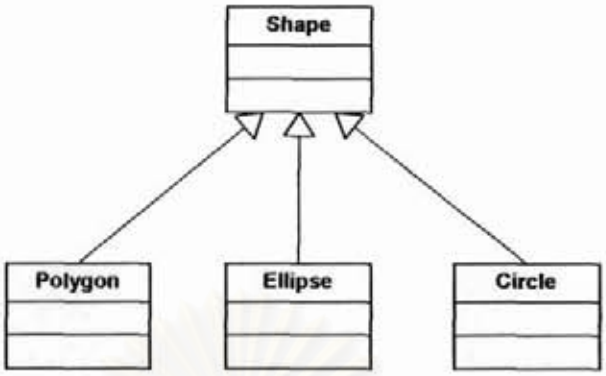
รูปที่ 2-13 แสดงการกำหนดแอตทริบิวต์ และโอเปอเรชันภายในคลาส (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

2. ความสัมพันธ์ คือความสัมพันธ์ระหว่างคลาสซึ่งสามารถแบ่งออกได้เป็น 3 ประเภทดังต่อไปนี้

ก. ความสัมพันธ์แบบพึ่งพิง (Dependency Relationship) ความสัมพันธ์แบบนี้เกิดขึ้นเมื่อการเปลี่ยนแปลงที่เกิดขึ้นกับคลาสที่ถูกพึ่งพิง (Independent Class) จะส่งผลต่อคลาสที่พึ่งพิง (Dependent Class) คลาสดังกล่าว (Dennis และคณะ, 2005) สัญลักษณ์ที่ใช้คือลูกศรหัวโปร่งแบบเส้นประชี้จากคลาสที่พึ่งพิงไปยังคลาสที่ถูกพึ่งพิง ดังรูปที่ 2-14

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

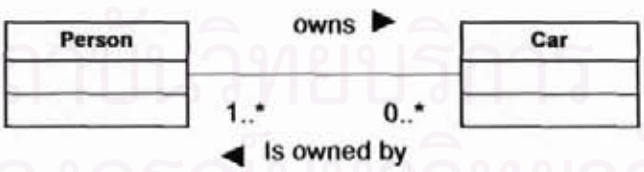
ข. ความสัมพันธ์แบบเจเนอรัลไลเซชัน (Generalization Relationship) คือความสัมพันธ์ระหว่างซูเปอร์คลาสและซับคลาส เป็นการสืบทอดคุณสมบัติจากซูเปอร์คลาสนั่นเอง (Dennis และคณะ, 2005) สัญลักษณ์ที่ใช้คือเส้นตรงทึบที่มีหัวลูกศรเป็นสามเหลี่ยมโปร่งโดยชี้จากซับคลาสไปยังซูเปอร์คลาส ดังรูปที่ 2-15



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

ค. ความสัมพันธ์แบบแอสโซซิเอชัน เป็นความสัมพันธ์ระหว่างคลาสอีกชนิดหนึ่งโดยสามารถแบ่งออกได้เป็น

- นอร์มอลแอสโซซิเอชัน (Normal Association) โดยปกติความสัมพันธ์แบบนี้จะเป็นความสัมพันธ์แบบ 2 ทิศทาง ซึ่งใช้สัญลักษณ์เส้นตรงทึบเชื่อมระหว่าง 2 คลาส และมีชื่อของความสัมพันธ์กำกับอยู่ โดยชื่อที่ใช้มักเป็นคำกริยาโดยส่วนใหญ่ นอกจากนี้ยังสามารถกำหนดทิศทางของชื่อความสัมพันธ์ได้ โดยการวาดสามเหลี่ยมทึบไว้ด้านซ้ายหรือด้านขวาของชื่อ ซึ่งช่วยให้อ่านความสัมพันธ์ได้อย่างถูกต้อง และที่แต่ละเส้นความสัมพันธ์อาจมี 2 ชื่อ ความสัมพันธ์ซึ่งมีทิศทางตรงข้ามกัน และสามารถกำหนดปริมาณของคลาสหรืออ็อบเจ็กต์ที่สัมพันธ์กันอยู่เรียกว่ามัลติไพลซิตี (Multiplicity) ซึ่งสามารถกำหนดได้หลายรูปแบบโดยการใส่ตัวเลขไว้ที่ปลายด้านใดด้านหนึ่งของความสัมพันธ์ดังรูปที่ 2-16



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

นอกจากนี้ยังมีการกำหนดบทบาท (Role) ให้กับแต่ละคลาสที่เชื่อมต่ออยู่กับเส้นความสัมพันธ์ได้เช่นกันดังรูปที่ 2-17

รูปที่ 2-17 แสดงความสัมพันธ์แบบแอสโซซิเอชัน เมื่อมีการกำหนดบทบาทของแต่ละคลาส (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

2. แอกกริเกชัน (Aggregation) เป็นความสัมพันธ์ระหว่างคลาสหรืออ็อบเจ็กต์ในแง่ของการรวมกันหรือการประกอบกัน ซึ่งสามารถแบ่งได้ 2 แบบดังนี้

ก. นอร์มอลแอกกริเกชัน (Normal Aggregation) เป็นการกำหนดส่วนประกอบแบบไม่จำเป็น (Dennis และคณะ, 2005) ใช้สัญลักษณ์เส้นตรงทึบโยงระหว่างคลาส โดยมีข้าวหลามตัดติดอยู่ระหว่างปลายเส้นความสัมพันธ์กับคลาสที่ใหญ่กว่า เช่น ประตูเป็นส่วนหนึ่งของรถ ในขณะเดียวกันสามารถกำหนดชื่อความสัมพันธ์ ทิศทางของชื่อความสัมพันธ์ ทิศทางของความสัมพันธ์ และปริมาณที่สัมพันธ์กันได้ดังรูปที่ 2-18

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

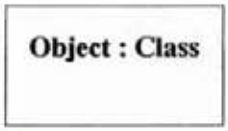
ข. คอมโพซิชัน (Composition) มีความคล้ายคลึงกับนอร์มอลแอกกริเกชัน แต่เป็นการกำหนดส่วนประกอบแบบจำเป็น เช่น คลาสรถ (Car) เป็นเจ้าของคลาสเครื่องยนต์ (Engine) เมื่อคลาสที่เป็นเจ้าของ (คลาสรถ) ถูกทำลาย คลาสที่เป็นองค์ประกอบ (คลาสเครื่องยนต์) จะถูกทำลายไปด้วยพร้อมๆกัน ดังรูปที่ 2-19

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

2.7.4 แผนภาพซีเควนซ์ แผนภาพนี้จะบอกว่าในยูสเคสนั้น อ็อบเจ็กต์แต่ละตัวติดต่อสื่อสารกันอย่างไร มีขั้นตอนการทำงานอย่างไร โดยจะเน้นไปที่แกนเวลาเป็นสำคัญ (Dennis และคณะ, 2005) ถ้าเวลาเปลี่ยนขั้นตอนการทำงานจะเปลี่ยนโดยมีแอกเตอร์เป็นผู้เริ่มต้นขั้นตอน โดยแผนภาพซีเควนซ์มีส่วนประกอบดังต่อไปนี้

1. แอกเตอร์ คือบุคคลหรือระบบภายนอกที่มีการโต้ตอบกับระบบ โดยการรับส่งเมสเซจ (Message) (Dennis และคณะ, 2005) สัญลักษณ์ที่ใช้คือรูปคน ดังรูปที่ 2-6

2. อ็อบเจ็กต์แทนด้วยรูปสี่เหลี่ยมเรียงกันตามแนวนอน ภายในบรรจุชื่ออ็อบเจ็กต์ตามด้วยเครื่องหมายมหัพภาคคู่และชื่อคลาส ดังรูปที่ 2-20



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

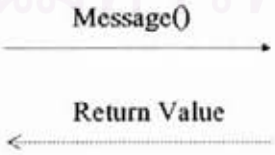
3. ไลฟ์ไลน์ (Lifeline) แสดงถึงชีวิตของอ็อบเจ็กต์ (Dennis และคณะ, 2005) แทนด้วยเส้นประแนวตั้งดังรูปที่ 2-21

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

4. แอกติเวชัน (Activation) ใช้แสดงช่วงเวลาที่อ็อบเจ็กต์กำลังส่งและรับเมสเซจ (Dennis และคณะ, 2005) แทนด้วยสี่เหลี่ยมแนวตั้ง ดังรูปที่ 2-22

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

5. เมสเซจ เป็นการส่งและคืนข้อมูล (Dennis และคณะ, 2005) สัญลักษณ์ที่ใช้ในการส่งข้อมูลคือลูกศรเส้นทึบ และสัญลักษณ์ที่ใช้ในการคืนข้อมูลคือลูกศรเส้นประดังรูปที่ 2-23



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

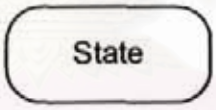
6. อ็อบเจกต์ดีสตรักชัน (Object Destruction) เป็นการใช้กากบาทวางไว้ที่ปลายไลฟ์ไลน์ของอ็อบเจ็กต์ เพื่อแสดงว่าอ็อบเจ็กต์นั้นสิ้นสุดแล้ว (Dennis และคณะ, 2005) ดังรูปที่ 2-24

X

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

2.7.5 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 2-25



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

2. สถานะเริ่มต้น (Initial State) เป็นจุดเริ่มต้นสถานะของอ็อบเจ็กต์ (Dennis และคณะ, 2005) แสดงด้วยสัญลักษณ์รูปวงกลมทึบ ดังรูปที่ 2-26

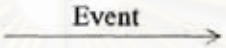
รูปที่ 2-26 แสดงสถานะเริ่มต้น (ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์, 2544)

3. สถานะสิ้นสุด (Final State) เป็นจุดสิ้นสุดการกระทำของอ็อบเจ็กต์ (Dennis และคณะ, 2005) แสดงด้วยสัญลักษณ์รูปวงกลมโปร่งล้อมรอบวงกลมทึบข้างในเรียกว่า ตาวัว (Bull's Eye) ดังรูปที่ 2-27

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

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

5. ทรานซิชัน คือเส้นลูกศรเส้นทึบที่ชี้จากสถานะหนึ่งไปยังอีกสถานะหนึ่ง โดยมีอีเวนต์แสดงบนเส้นเพื่อบอกว่าเมื่อเกิดเหตุการณ์หนึ่งจะทำให้สถานะหนึ่งเปลี่ยนไปอีกสถานะหนึ่ง (Dennis และคณะ, 2005) ดังรูปที่ 2-28



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

นอกจากแผนภาพทั้ง 4 แผนภาพนี้แล้ว เทคนิคโอโออาร์ที่ยังนำเอกสารอธิบายความต้องการซอฟต์แวร์ และคำอธิบายคลาสมาตรวจสอบด้วย โดยทั้ง 2 เอกสารมีลักษณะดังต่อไปนี้

2.7.6 เอกสารอธิบายความต้องการซอฟต์แวร์ เป็นเอกสารที่ใช้อธิบายความต้องการทางซอฟต์แวร์ ซึ่งในเอกสารนั้นจะแบ่งออกเป็น 2 ส่วนคือ (1) ความต้องการทางฟังก์ชัน (Functional Requirement) เป็นความต้องการทางด้านฟังก์ชันการทำงานของซอฟต์แวร์ที่ต้องทำได้ (2) ความต้องการที่ไม่เป็นฟังก์ชัน (Nonfunctional Requirement) เป็นความต้องการที่กำหนดคุณภาพในการทำงานของซอฟต์แวร์ เช่น ประสิทธิภาพ ความน่าเชื่อถือได้เป็นต้น (Dennis และคณะ, 2005)

2.7.7 คำอธิบายคลาส เป็นเอกสารที่อธิบายรายละเอียดของแต่ละคลาสซึ่งสามารถแบ่งออกเป็น 3 ส่วนคือ (1) รายละเอียดของคลาส ประกอบด้วยชื่อคลาส คำอธิบายคลาส ซูเปอร์คลาส ซับคลาส ชื่อความสัมพันธ์ รูปแบบความสัมพันธ์ ประเภทความสัมพันธ์ มัลติไพลซิตี (2) รายละเอียดของแอตทริบิวต์ ประกอบด้วยชื่อแอตทริบิวต์ คำอธิบายแอตทริบิวต์ การเข้าถึงแอตทริบิวต์ ประเภทของแอตทริบิวต์ (อาจมีค่าเริ่มต้น) (3) เมท็อด ประกอบด้วยชื่อเมท็อด คำอธิบายเมท็อด การเข้าถึงเมท็อด ชื่อพารามิเตอร์ ประเภทพารามิเตอร์ ประเภทของค่าที่ส่งคืน (Stone, 2001)

บทที่ 3
ระเบียบวิธีวิจัย

3.1 บทนำ

แนวทางในการทำวิจัยได้นำเสนอไว้ในบทนี้ อันประกอบด้วยแผนแบบการทดลอง (Experimental Design) การทดสอบสมมติฐาน แนวทางการทำวิจัย ประชากรและหน่วยทดลอง การเตรียมเครื่องมือที่ใช้ในการทดลอง ขั้นตอนการเก็บข้อมูล (Data Gathering Execution) ประเด็นของความเชื่อถือได้ (Reliability) ความถูกต้อง (Validity) และกรอบการวิเคราะห์ข้อมูล (Data Analysis Framework) ดังรายละเอียดต่อไปนี้

3.2 แผนแบบการทดลอง

วัตถุประสงค์ของงานวิจัยนี้จัดทำขึ้นเพื่อเปรียบเทียบว่า ระหว่างการนำเทคนิคซีบีอาร์ (CBR) และเทคนิคโอโออาร์ที (OORT) ไปใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบด้วยวิธีการเชิงวัตถุ จะให้ประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม (Preparation) กับขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ (Inspection Meeting) และประสิทธิผลในการกำจัดผลบวกปลอม (False Positive) ของขั้นตอนการประชุมตรวจสอบแตกต่างกันหรือไม่อย่างไร

จากวัตถุประสงค์งานวิจัยข้างต้น จึงใช้แผนแบบการทดลองแบบสองกลุ่มหนึ่งปัจจัย (Single Factor Two Group Design) ซึ่งเป็นแผนแบบการทดลองสำหรับเปรียบเทียบค่าเฉลี่ยของกลุ่มทดลองสองกลุ่ม (Wohlin และคณะ, 2000) เนื่องจากงานวิจัยนี้มีตัวแปรต้น 1 ตัวนั่นคือเทคนิคการอ่านซอฟต์แวร์ โดยมีค่าที่เป็นไปได้ 2 ค่าคือเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที ดังนั้นในงานวิจัยจึงมีกลุ่มทดลอง 2 กลุ่มซึ่งได้รับทรีทเมนต์ที่แตกต่างกัน คือกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ และกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที โดยงานวิจัยนี้ประกอบด้วยตัวแปรต้น ตัวแปรตาม และตัวแปรควบคุมดังต่อไปนี้

3.2.1 ตัวแปรต้น งานวิจัยนี้ให้ความสนใจว่าเทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกันส่งผลกับประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์หรือไม่ ดังนั้นตัวแปรต้นของงานวิจัยนี้มี 1 ตัวแปร คือเทคนิคการอ่านซอฟต์แวร์ ซึ่งตัวแปรนี้มี 2 ค่าคือ (1) เทคนิคโอโออาร์ที และ (2) เทคนิคซีบีอาร์

3.2.2 ตัวแปรตาม เนื่องจากงานวิจัยนี้เป็นการเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม และขั้นตอนการจัดเตรียมรวมกับขั้นตอนการ

ประชุมตรวจสอบ และเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ จึงประกอบด้วยตัวแปรตาม 5 ตัวดังต่อไปนี้

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

$$\text{ประสิทธิภาพ} = \frac{\text{จำนวนข้อบกพร่องที่พบในการตรวจสอบขั้นตอนการจัดเตรียม}}{\text{ระยะเวลาที่ใช้ในการตรวจสอบขั้นตอนการจัดเตรียม}}$$

2. ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม สามารถวัดค่าตัวแปรนี้ได้จากร้อยละของจำนวนข้อบกพร่องที่ผู้ตรวจสอบพบเทียบกับจำนวนข้อบกพร่องทั้งหมดที่มี (Sabalaiuskaite และคณะ, 2002)

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

3. ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ สามารถวัดค่าตัวแปรนี้ได้จากจำนวนข้อบกพร่องเฉลี่ยที่คณะผู้ตรวจสอบพบในระยะเวลา 1 นาที (Sabalaiuskaite และคณะ, 2002)

$$\text{ประสิทธิภาพ} = \frac{\text{จำนวนข้อบกพร่องที่พบในการตรวจสอบขั้นตอนการประชุมตรวจสอบ}}{\text{ระยะเวลาที่ใช้ในการตรวจสอบขั้นตอนการประชุมตรวจสอบ}}$$

4. ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ สามารถวัดค่าตัวแปรนี้ได้จากร้อยละของจำนวนข้อบกพร่องที่คณะผู้ตรวจสอบพบเทียบกับจำนวนข้อบกพร่องทั้งหมดที่มี (Sabalaiuskaite และคณะ, 2002)

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

5. ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ สามารถวัดค่าตัวแปรนี้ได้จากร้อยละของผลบวกปลอมที่ถูกกำจัดในขั้นตอนการประชุมตรวจสอบจากจำนวนผลบวกปลอมทั้งหมดที่พบในขั้นตอนการจัดเตรียม (Sabaliauskaite, 2004)

$$\text{ประสิทธิผลในการกำจัดผลบวกปลอม} = \left(\frac{FP_{Preparation} - FP_{Meeting}}{FP_{Preparation}} \right) \times 100$$

โดยที่ $FP_{Preparation}$ = จำนวนผลบวกปลอมที่พบในขั้นตอนการจัดเตรียม

$FP_{Meeting}$ = จำนวนผลบวกปลอมที่พบในขั้นตอนการประชุมตรวจสอบ

3.2.3 ตัวแปรควบคุม เพื่อให้ผลการทดลองนั้นเกิดจากเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้อย่างแท้จริง ผู้วิจัยจึงต้องควบคุมปัจจัยอื่นๆที่อาจส่งผลกับการทดลองให้มีความคงที่หรือเหมือนกันมากที่สุด ดังต่อไปนี้

1. เอกสารแสดงการออกแบบซอฟต์แวร์ เป็นเอกสารแสดงการออกแบบซอฟต์แวร์ที่เป็นระบบธุรกิจที่ผู้วิจัยนำมาใช้ในการทดลอง โดยให้หน่วยทดลองค้นหาข้อบกพร่องที่ทางผู้วิจัยใส่ไว้ ซึ่งเอกสารแสดงการออกแบบซอฟต์แวร์จะประกอบด้วยแผนภาพยูเอ็มแอลต่างๆ ได้แก่แผนภาพยูสเคส (Use Case Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) แผนภาพสถานะ (State Machine Diagram) และแผนภาพคลาส (Class Diagram) นอกจากนี้ยังมีเอกสารอธิบายคลาสของระบบ (Class Description) และเอกสารอธิบายความต้องการซอฟต์แวร์ (Requirement Description) สาเหตุที่เลือกแผนภาพเหล่านี้มาตรวจสอบนั้น เนื่องจากเทคนิคโอโออาร์ทีได้กำหนดแนวทางการตรวจสอบไว้ว่าต้องตรวจสอบแผนภาพเหล่านี้ ร่วมกับเอกสารอธิบายความต้องการซอฟต์แวร์และเอกสารอธิบายคลาสของระบบซึ่งทุกๆแผนภาพจะมีการกำหนดข้อบกพร่องแต่ละประเภทไว้ โดยที่ทุกหน่วยทดลองจะได้รับเอกสารแสดงการออกแบบซอฟต์แวร์ที่มีรายละเอียดเหมือนกันทุกประการ

2. เอกสารคำแนะนำ เป็นเอกสารที่ใช้เป็นแนวทางในการตรวจสอบซอฟต์แวร์ ซึ่งแบ่งออกเป็น 2 ประเภทตามเทคนิคการอ่านซอฟต์แวร์ดังนี้ (1) เช็กลิสต์เป็นคำแนะนำสำหรับกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบเท่านั้น โดยมีลักษณะเป็นรายการคำถามแบบใช่ / ไม่ใช่ โดยทุกหน่วยทดลองที่ใช้เทคนิคซีบีอาร์ในการทดลองจะได้รับเช็กลิสต์ที่มีรายละเอียดเหมือนกันทุกประการ และ (2) สถานการณ์ (Scenario) เป็นคำแนะนำสำหรับกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบเท่านั้น ซึ่งกลุ่มทดลองที่ทดลองโดยการใช้เทคนิคโอโออาร์ทีจะได้รับสถานการณ์ (Scenario) นี้ โดยทุกหน่วยทดลองที่ใช้เทคนิคโอโออาร์ทีจะได้รับสถานการณ์ (Scenario) ที่มีรายละเอียดเหมือนกันทุกประการ

3. เอกสารที่ใช้บันทึกข้อบกพร่องที่พบ ใช้ในการบันทึกผลการทดลองโดยแบ่งออกเป็น 2 เอกสาร (1) เอกสารที่ใช้บันทึกข้อบกพร่องที่พบในขั้นตอนการจัดเตรียมซึ่งจะประกอบด้วย ชื่อผู้ตรวจสอบ เทคนิคที่ใช้ในการตรวจสอบ ข้อบกพร่องที่พบ รายละเอียดของ

ข้อบกพร่อง ตำแหน่งที่พบว่าพบที่แผนภาพใดและตำแหน่งใดของแผนภาพ ระยะเวลาที่ใช้ในขั้นตอนการจัดเตรียม (ผู้วิจัยกำหนดเวลาที่ใช้ในขั้นตอนการจัดเตรียมไม่เกิน 90 นาที) โดยทุกหน่วยทดลองจะได้รับเอกสารที่มีลักษณะเหมือนกันทุกประการ และ (2) เอกสารที่ใช้บันทึกข้อบกพร่องที่พบในขั้นตอนการประชุมตรวจสอบ ซึ่งจะประกอบด้วย รหัสคณะผู้ตรวจสอบ เทคนิคที่ใช้ในการตรวจสอบ ข้อบกพร่องที่พบ รายละเอียดของข้อบกพร่อง ตำแหน่งที่พบว่าพบที่แผนภาพใดและตำแหน่งใดของแผนภาพ ระยะเวลาที่ใช้ในการทำขั้นตอนการประชุมตรวจสอบ (ผู้วิจัยกำหนดเวลาที่ใช้ในการทำขั้นตอนการประชุมตรวจสอบไม่เกิน 60 นาที) โดยทุกคณะผู้ตรวจสอบจะได้รับเอกสารนี้ซึ่งมีลักษณะเหมือนกันทุกประการ

4. เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล เป็นเอกสารที่แสดงความหมายสัญลักษณ์ของแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ เพื่อช่วยให้หน่วยทดลองอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ได้เข้าใจมากขึ้น โดยที่ทุกหน่วยทดลองจะได้รับเอกสารนี้เหมือนกันทุกประการ

5. ระยะเวลาที่ใช้ในการทดลอง ในการทดลองแบ่งออกเป็น 3 ขั้นตอนคือ (1) การประชุมแนะนำ (Overview) เป็นขั้นตอนที่ทางผู้วิจัยอธิบายระบบ และเทคนิคการอ่านซอฟต์แวร์ให้หน่วยทดลองเข้าใจ โดยใช้เวลาในการอธิบายไม่เกิน 30 นาที (2) ขั้นตอนการจัดเตรียม (Preparation) เป็นขั้นตอนที่แต่ละหน่วยทดลองค้นหาข้อบกพร่อง โดยควบคุมให้ทุกหน่วยทดลองใช้ระยะเวลาไม่เกิน 90 นาที (3) ขั้นตอนการประชุมตรวจสอบ (Inspection Meeting) เป็นขั้นตอนที่แต่ละกลุ่มทดลองค้นหาข้อบกพร่อง และแลกเปลี่ยนความเห็นกันเพื่อบันทึกข้อบกพร่องที่พบ โดยควบคุมให้ทุกกลุ่มทดลองใช้เวลาไม่เกิน 60 นาที

3.3 การทดสอบสมมติฐาน

งานวิจัยนี้ต้องการเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม และขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที และเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล ดังนั้นผู้วิจัยจึงตั้งสมมติฐานในงานวิจัยดังต่อไปนี้

1. การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมระหว่างการนำเทคนิคซีบีอาร์ และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

H_0: ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลนั้นมีค่าไม่ต่างกัน

H_1: ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม โดยการนำเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล มีค่าสูงกว่าการนำเทคนิคซีบีอาร์มาใช้

2. การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

H_0: ประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลนั้นมีค่าไม่ต่างกัน

H_1: ประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม โดยการนำเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล มีค่าสูงกว่าการนำเทคนิคซีบีอาร์มาใช้

3. การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

H_0: ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลนั้นมีค่าไม่ต่างกัน

H_1: ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบโดยการนำเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล มีค่าสูงกว่าการนำเทคนิคซีบีอาร์มาใช้

4. การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

H_0: ประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการ ตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลนั้นมีค่าไม่ต่างกัน

H_1: ประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบ โดยการนำเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล มีค่าสูงกว่าการนำเทคนิคซีบีอาร์มาใช้

5. การเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล

H_0: ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลนั้นมีค่าไม่ต่างกัน

H_1: ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ โดยการนำเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล มีค่าสูงกว่าการนำเทคนิคซีบีอาร์มาใช้

สาเหตุที่ทางผู้วิจัยตั้งสมมติฐานดังที่ได้กล่าวมาข้างต้นนั้น เนื่องจากผู้วิจัยคาดว่าเทคนิคโอโออาร์ทีเป็นเทคนิคที่ถูกจัดทำขึ้นสำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ใช้แผนภาพยูเอ็มแอลในการออกแบบโดยเฉพาะ จึงน่าจะมีประสิทธิภาพและประสิทธิผลในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ใช้แผนภาพยูเอ็มแอลในการออกแบบมากกว่าเทคนิคซีบีอาร์ และจากงานวิจัยของ Conradi และคณะ (2003) สรุปผลได้ว่าเทคนิคโอโออาร์ทีสามารถค้นหาข้อบกพร่องได้เป็นจำนวนมาก และยังช่วยในการกำจัดผลบวกปลอม (False Positive) อีกด้วย

3.4 ประชากรและหน่วยทดลอง

ประชากร (Population หรือ Universe) หมายถึง หน่วยข้อมูลทั้งหมดตามที่ได้กำหนดไว้ก่อนการเลือกหน่วยทดลอง โดยจะต้องครอบคลุมในประเด็นที่สนใจจะศึกษา (ศิริวรรณ เสรีรัตน์ และคณะ, 2541) สำหรับงานวิจัยนี้ประชากรคือ ผู้ตรวจสอบทั้งหมดจากองค์กรที่รับจ้างพัฒนาซอฟต์แวร์ เพื่อเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้ในการตรวจสอบซอฟต์แวร์ แต่

ในทางปฏิบัตินั้นไม่สามารถนำผู้ตรวจสอบมาทดลองได้ เนื่องจากผู้ตรวจสอบซอฟต์แวร์ในองค์กรที่รับจ้างพัฒนามีงานในความรับผิดชอบที่ต้องทำในองค์กร จึงเป็นการยากที่จะนำผู้ตรวจสอบซอฟต์แวร์ในองค์กรที่รับจ้างพัฒนาซอฟต์แวร์มาใช้ในการทดลอง ผู้วิจัยจึงต้องเลือกหน่วยทดลองขึ้นมาเป็นตัวแทนประชากรดังกล่าว

หน่วยทดลองหมายถึง บางหน่วยของประชากรทั้งหมดที่ถูกเลือกขึ้นมาเป็นตัวแทนในการวิจัย ตามวิธีการและหลักเกณฑ์ที่กำหนดไว้หน่วยทดลองที่ดีจะให้ข้อมูลของประชากร และลดค่าใช้จ่ายในการดำเนินการวิจัย (ศิริวรรณ เสรีรัตน์ และคณะ, 2541) โดยหน่วยทดลองที่นำมาใช้เพื่อเป็นตัวแทนของประชากรในการตอบวัตถุประสงค์ของงานวิจัยนี้คือ นิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการพัฒนาซอฟต์แวร์เชิงวัตถุ การกำหนดหน่วยทดลองเป็นนิสิตกลุ่มนี้เพราะ นิสิตที่จบสาขาวิชานี้มีแนวโน้มที่จะทำงานในองค์กรที่รับจ้างพัฒนาซอฟต์แวร์ ซึ่งตรงตามลักษณะประชากรที่ผู้วิจัยต้องการ และนิสิตสาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ มีการกำหนดให้ศึกษารายวิชาที่มีเนื้อหาด้านการพัฒนาซอฟต์แวร์เชิงวัตถุ จึงเหมาะสมกับงานวิจัยนี้ที่ต้องการเปรียบเทียบประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ โดยการนำเทคนิคการอ่านซอฟต์แวร์มาช่วยในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล นอกจากนี้ยังสะดวกต่อผู้วิจัยในการติดต่อเพื่อให้มาร่วมการทดลอง

ในการทดลองผู้วิจัยแบ่งกลุ่มทดลองออกเป็น 2 กลุ่มคือ (1) กลุ่มที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบ (2) กลุ่มที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ โดยใช้วิธีสุ่มหน่วยทดลองลงในกลุ่มที่แบ่งไว้ ซึ่งทั้งสองกลุ่มจะมีจำนวนหน่วยทดลองเท่ากัน สาเหตุที่ผู้วิจัยใช้วิธีการสุ่มเนื่องจากผู้วิจัยถือว่าทุกหน่วยทดลองมีลักษณะที่เหมือนกันตามที่ผู้วิจัยต้องการ และหลังจากที่ทุกหน่วยทดลองได้ทดลองในขั้นตอนการจัดเตรียมเรียบร้อยแล้ว ผู้วิจัยจะจัดกลุ่มหน่วยทดลองเป็นกลุ่มละ 3 คนเพื่อตรวจสอบในขั้นตอนการประชุมตรวจสอบต่อไป กล่าวคือในขั้นตอนการประชุมตรวจสอบนั้นมีการจัดกลุ่มกลุ่มละ 3 คน โดยหน่วยทดลองในกลุ่มจะต้องใช้เทคนิคการอ่านซอฟต์แวร์แบบเดียวกัน

สาเหตุที่ทางผู้วิจัยกำหนดจำนวนกลุ่มทดลองเป็นกลุ่มละ 3 คนเพื่อใช้ในขั้นตอนการประชุมตรวจสอบ เนื่องจากขั้นตอนการประชุมตรวจสอบเป็นขั้นตอนสำหรับให้หน่วยทดลองประชุมร่วมกัน เพื่อค้นหาข้อบกพร่องและสรุปว่าเป็นข้อบกพร่องจริงหรือไม่ ดังนั้นทางผู้วิจัยจึงกำหนดจำนวนหน่วยทดลองในกลุ่มย่อยเป็นจำนวนคี่ เพื่อให้สามารถมีผู้ชี้ขาดในการตัดสินใจได้กรณีที่หน่วยทดลองมีความเห็นไม่ตรงกัน และ Strauss และ Ebenau (1994) ได้กล่าวไว้ว่า ขนาดของคณะผู้ตรวจสอบอย่างน้อยที่สุดควรประกอบด้วย 3 คน คือผู้ดำเนินรายการ (Moderator) ผู้อ่าน

(Reader) และผู้บันทึก (Recorder) ดังนั้นทางผู้วิจัยจึงกำหนดจำนวนของหน่วยทดลองต่ำที่สุด เพื่อให้ได้จำนวนกลุ่มย่อยมากที่สุด

หน่วยทดลองที่กำหนดไว้ข้างต้นซึ่งได้แก่ นิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการพัฒนาซอฟต์แวร์เชิงวัตถุ มีจำนวนทั้งหมด 44 คน โดยในการทำขั้นตอนการประชุมตรวจสอบนั้น ผู้วิจัยจะจัดกลุ่มทดลองเป็นกลุ่มย่อยกลุ่มละ 3 คน ดังนั้นผู้วิจัยจึงหาหน่วยทดลองเพิ่มจากเดิมที่มีอยู่ 44 คนให้เป็น 48 คน โดยในการเพิ่มหน่วยทดลองนั้นจะใช้คุณสมบัติของผู้สมัครเข้าศึกษาหลักสูตรการพัฒนาซอฟต์แวร์ด้านธุรกิจเป็นเกณฑ์ในการคัดเลือก เพื่อให้สามารถจัดกลุ่มทดลองในขั้นตอนการประชุมตรวจสอบตามลักษณะที่กล่าวมาแล้วข้างต้นได้ ซึ่งจะได้กลุ่มทดลองในขั้นตอนการประชุมตรวจสอบทั้งหมด 16 กลุ่ม และในขั้นตอนการจัดเตรียมประกอบด้วยหน่วยทดลองทั้งหมด 48 คน

3.5 แนวทางการทำวิจัย

งานวิจัยนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) เนื่องจากงานวิจัยนี้มีการกำหนดให้ตัวแปรที่สนใจเปลี่ยนค่า เพื่อวัดผลกระทบที่มีต่ออีกตัวแปร (ศิริวรรณ เสรีรัตน์ และคณะ, 2541; Babbie, 2001) โดยที่งานวิจัยนี้มีตัวแปรที่สนใจคือเทคนิคการอ่านซอฟต์แวร์ (Software Reading Techniques) ที่นำมาช่วยในการตรวจสอบซอฟต์แวร์ (Software Inspection) กล่าวคืองานวิจัยนี้ต้องการเปรียบเทียบประสิทธิภาพ (Efficiency) และประสิทธิผล (Effectiveness) ของการตรวจสอบซอฟต์แวร์โดยการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน เพื่อให้ผลการทดลองนั้นเกิดจากเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้อย่างแท้จริง ในงานวิจัยนี้จึงควบคุมตัวแปรอื่นๆได้แก่ (1) เอกสารแสดงการออกแบบซอฟต์แวร์ซึ่งประกอบด้วยแผนภาพยูเอ็มแอล (UML) (2) เอกสารคำแนะนำ (Guideline) ที่มานำใช้เป็นแนวทางในการตรวจสอบซอฟต์แวร์ ซึ่งแบ่งออกเป็น 2 ประเภทคือ เช็กลิสต์ซึ่งเป็นเอกสารคำแนะนำสำหรับกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบเท่านั้น และสถานการณ์ (Scenario) ซึ่งเป็นเอกสารคำแนะนำสำหรับกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที (OORT) ในการตรวจสอบเท่านั้น (3) เอกสารที่ใช้บันทึกผลการตรวจสอบ (4) เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล และ (5) ระยะเวลาที่ใช้ในการตรวจสอบ

จากลักษณะการวิจัยดังกล่าวเห็นได้ว่างานวิจัยนี้ไม่สามารถนำประชากรจริง ซึ่งได้แก่ผู้ตรวจสอบจากองค์กรรับจ้างพัฒนาซอฟต์แวร์มาใช้ในการทดลอง และทางผู้วิจัยไม่สามารถนำเอกสารการออกแบบซอฟต์แวร์ที่กลุ่มผู้ผลิตซอฟต์แวร์จัดทำขึ้นจริงมาใช้ในการทดลอง (การตรวจสอบซอฟต์แวร์) แต่จะใช้เอกสารแสดงการออกแบบซอฟต์แวร์ที่มีความถูกต้องสมบูรณ์มา

กำหนดใส่ข้อบกพร่องเพิ่มเติมเอง กล่าวคืองานวิจัยมีการควบคุมตัวแปรต่างๆ ทำให้ไม่สามารถใช้การวิจัยแบบกึ่งทดลอง (Quasi Experimental) ผู้วิจัยจึงเลือกใช้งานวิจัยเชิงทดลองในห้องปฏิบัติการ (Laboratory Experiment) เพื่อควบคุมตัวแปรอื่นๆที่ไม่ได้สนใจให้ส่งผลกระทบต่อตัวแปรตามน้อยที่สุด (ศิริวรรณ เสรีรัตน์ และคณะ, 2541; สำนักงานคณะกรรมการวิจัยแห่งชาติ, 2547)

งานวิจัยนี้ไม่เป็นการวิจัยเชิงทดลองโดยสมบูรณ์ เนื่องจากในการวิจัยเชิงทดลองในห้องปฏิบัติการควรประกอบด้วย 2 กลุ่มคือ (1) กลุ่มทดลอง (Experimental Group) ซึ่งเป็นกลุ่มที่ได้รับการกระตุ้นในการทดลอง สำหรับงานวิจัยนี้คือเทคนิคการอ่านซอฟต์แวร์ ซึ่งจะได้รับเอกสารคำแนะนำที่แตกต่างกัน ขึ้นกับเทคนิคการอ่านซอฟต์แวร์ที่ใช้ในการทดลอง (2) กลุ่มควบคุม (Control Group) เป็นกลุ่มที่ไม่ได้รับการกระตุ้นในการทดลอง เพื่อใช้ยืนยันว่าการที่สิ่งที่นักวิจัยสนใจเปลี่ยนค่านั้น เป็นผลมาจากการได้รับการกระตุ้นจากการทดลอง (Babbie, 2001) โดยกลุ่มควบคุมสำหรับงานวิจัยนี้คือกลุ่มทดลองที่ไม่ได้รับเอกสารคำแนะนำในการตรวจสอบซอฟต์แวร์ ซึ่งกลุ่มควบคุมในลักษณะนี้ถือว่าเป็นการนำรูปแบบของเทคนิคการอ่านซอฟต์แวร์แบบเทคนิคเฉพาะกิจ (Ad-hoc) มาใช้ ซึ่งเป็นเทคนิคที่ผู้ตรวจสอบต้องเป็นผู้มีประสบการณ์ในการตรวจสอบซอฟต์แวร์ เพื่อที่จะทราบได้ว่าต้องทำอย่างไรจึงจะสามารถค้นหาข้อบกพร่องที่มีอยู่ให้พบ (Laitenberger และคณะ, 2000) แต่กลุ่มทดลองที่นำมาใช้ในงานวิจัยนี้เป็นผู้ที่ไม่มีประสบการณ์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์จริงมาก่อน ดังนั้นการที่มีกลุ่มควบคุมก็ถือว่าเป็นกลุ่มควบคุมที่ไม่มีประสิทธิภาพสำหรับการทดลอง ทางผู้วิจัยจึงไม่กำหนดให้มีกลุ่มควบคุมในการทดลอง

3.6 การเตรียมเครื่องมือที่ใช้ในการทดลอง

เครื่องมือที่ใช้ในการทดลองนี้ประกอบด้วย 4 ส่วนคือ (1) เอกสารแสดงการออกแบบซอฟต์แวร์ (2) เอกสารคำแนะนำ (3) เอกสารบันทึกข้อบกพร่อง และ (4) เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล โดยผู้วิจัยมีขั้นตอนการเตรียมเครื่องมือที่ใช้ในการทดลองดังต่อไปนี้

3.6.1 การเตรียมเอกสารแสดงการออกแบบซอฟต์แวร์ เอกสารแสดงการออกแบบซอฟต์แวร์เป็นเอกสารที่ผู้วิจัยนำมาให้หน่วยทดลองตรวจสอบเพื่อหาข้อบกพร่อง ซึ่งจะประกอบด้วยแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ

3.6.1.1 การคัดเลือกเอกสารแสดงการออกแบบซอฟต์แวร์ ผู้วิจัยเลือกเอกสารแสดงการออกแบบซอฟต์แวร์เรื่องระบบเช่ารถ ซึ่งเป็นกรณีศึกษาจากเอกสารประกอบการสอนของมหาวิทยาลัยเบรเมน (University of Bremen) คณะวิทยาศาสตร์ ภาควิชาคณิตศาสตร์และวิทยาการ

คอมพิวเตอร์ (Department for Mathematics and Computer Science) จากเว็บไซต์ http://www.db.informatik.uni-bremen.de/teaching/ courses/ss2002_oose/lecture05.pdf ซึ่งประกอบด้วยแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ (ดูรายละเอียดในภาคผนวก ก) ผู้วิจัยเลือกกรณีศึกษานี้ เนื่องจากระบบนี้มีการตรวจสอบความถูกต้องตรงกันของแต่ละแผนภาพเอาไว้ ทำให้เอกสารแสดงการออกแบบซอฟต์แวร์เรื่องระบบการเช่ารถนี้มีความน่าเชื่อถือ

ผู้วิจัยศึกษาเอกสารแสดงการออกแบบซอฟต์แวร์ของระบบเช่ารถนี้ และปรับขอบเขตระบบให้มีขนาดเล็กลง เพื่อให้เหมาะสมกับระยะเวลาที่กำหนดไว้สำหรับการทดลอง และแก้ไขให้ระบบมีความถูกต้องมากที่สุด จากนั้นผู้วิจัยจัดให้มีการทดสอบเครื่องมือที่ใช้ในการทดลอง (Pretest) ขึ้น โดยให้หน่วยทดลองตรวจสอบหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงแล้ว เพื่อนำผลการตรวจสอบที่ได้มาปรับปรุงแก้ไขเอกสารแสดงการออกแบบซอฟต์แวร์ของระบบการเช่ารถให้มีความถูกต้องมากยิ่งขึ้น (ดูรายละเอียดการทดสอบได้ที่หัวข้อ 3.7.1 และ 3.7.2) และผู้วิจัยนำเอกสารสถานการณ์ (Scenario) ซึ่งเป็นเอกสารคำแนะนำของเทคนิคโอโออาร์ทีที่เป็นต้นฉบับ มาใช้ในการตรวจสอบหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ เพื่อให้เอกสารแสดงการออกแบบซอฟต์แวร์ของระบบการเช่ารถที่จะนำไปใช้ในการทดลองนั้นมีความถูกต้องมากที่สุด (ดูรายละเอียดในภาคผนวก ก)

3.6.1.2 การกำหนดข้อบกพร่องลงในเอกสารแสดงการออกแบบซอฟต์แวร์ เมื่อเอกสารแสดงการออกแบบซอฟต์แวร์ที่นำมาใช้ในการทดลองมีความถูกต้องสมบูรณ์แล้ว ผู้วิจัยจึงกำหนดข้อบกพร่องลงไปในแต่ละแผนภาพของเอกสารแสดงการออกแบบซอฟต์แวร์ โดยแบ่งประเภทของข้อบกพร่องออกเป็น 3 แบบคือ

1. สิ่งสูญหาย (Missing) กล่าวคือมีบางสิ่งที่จำเป็นขาดหายไปจากเอกสารแสดงการออกแบบซอฟต์แวร์

2. สิ่งผิดพลาด (Wrong) กล่าวคือมีบางสิ่งที่ผิดพลาด หรือไม่มีความชัดเจนในเอกสารแสดงการออกแบบซอฟต์แวร์

3. สิ่งเพิ่มเติม (Extra) คือมีบางสิ่งเพิ่มเข้ามาเอกสารแสดงการออกแบบซอฟต์แวร์ซึ่งเป็นสิ่งที่ไม่จำเป็นในการออกแบบ

งานวิจัยของ Mrdalj และคณะ (2004) พบว่าประเภทของข้อบกพร่องที่พบมากที่สุดในแผนภาพยูเอ็มแอลคือสิ่งผิดพลาด รองลงมาคือสิ่งเพิ่มเติม และที่พบน้อยที่สุดคือสิ่งสูญหาย และจากงานวิจัยของ Sabaliauskaite (2004) กล่าวว่าข้อบกพร่องประเภทสิ่งสูญหายนั้น สามารถถูกตรวจพบได้ง่ายที่สุด ดังนั้นผู้วิจัยจึงกำหนดข้อบกพร่องประเภทสิ่งผิดพลาดซึ่งเกิดขึ้นบ่อยที่สุดใน

แผนภาพยูเอ็มแอล และสิ่งสูญหายซึ่งเป็นข้อบกพร่องที่ค้นหาได้ง่ายที่สุดไว้ในจำนวนที่เท่ากัน และใส่ข้อบกพร่องประเภทสิ่งเพิ่มเติมไว้น้อยที่สุด เนื่องจากเป็นข้อบกพร่องที่มีโอกาสเกิดขึ้นปานกลางและสามารถตรวจพบได้ยาก

การกำหนดจำนวนข้อบกพร่องที่ใส่ในเอกสารแสดงการออกแบบซอฟต์แวร์นั้น ผู้วิจัยได้ศึกษาจากงานวิจัยของ Sabaliauskaite (2004) เนื่องจากลักษณะงานวิจัยมีความคล้ายคลึงกับงานวิจัยนี้ กล่าวคือเป็นการเปรียบเทียบประสิทธิภาพของเทคนิคการอ่านซอฟต์แวร์ ซึ่งงานวิจัยของ Sabaliauskaite (2004) ได้กำหนดเวลาที่ใช้ในการทำขั้นตอนการจัดเตรียม (Preparation) 60 นาที และขั้นตอนการประชุมตรวจสอบ (Inspection Meeting) 30 นาที และกำหนดข้อบกพร่องไว้จำนวน 15 ข้อ สำหรับการทดลองนี้ผู้วิจัยกำหนดเวลาที่ใช้ในการทำขั้นตอนการจัดเตรียม 90 นาที และขั้นตอนการประชุมตรวจสอบ 60 นาที ดังนั้นผู้วิจัยจึงกำหนดข้อบกพร่องจำนวนมากกว่างานวิจัยของ Sabaliauskaite (2004) โดยกำหนดจำนวนข้อบกพร่องไว้ที่ 20 ข้อ โดยแบ่งตามประเภทของข้อบกพร่องได้ดังตารางที่ 3-1

ตารางที่ 3-1 ตารางแสดงจำนวนข้อบกพร่องโดยแบ่งตามประเภทของข้อบกพร่องในการวิจัย

ประเภทของข้อบกพร่อง	จำนวนข้อบกพร่อง
สิ่งสูญหาย	8
สิ่งผิดพลาด	8
สิ่งเพิ่มเติม	4

สำหรับการกำหนดข้อบกพร่องลงในแต่ละแผนภาพนั้น ผู้วิจัยคำนึงถึงจำนวนหน้าของแต่ละแผนภาพ โดยเอกสารแสดงการออกแบบซอฟต์แวร์ประกอบด้วย 4 แผนภาพได้แก่ แผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ ซึ่งแต่ละแผนภาพมีจำนวนหน้าดังตารางที่ 3-2

ตารางที่ 3-2 ตารางแสดงจำนวนหน้าของแต่ละแผนภาพในเอกสารแสดงการออกแบบซอฟต์แวร์

ชื่อแผนภาพ	จำนวนหน้า
แผนภาพคลาสและคำอธิบายคลาส	7
แผนภาพซีเควนซ์	4
แผนภาพสถานะ	3
แผนภาพยูสเคส และคำอธิบายยูสเคส	3

การกำหนดข้อบกพร่องลงในแต่ละแผนภาพนั้น นอกจากผู้วิจัยจะคำนึงจากจำนวนหน้าของแต่ละแผนภาพแล้ว ยังพิจารณาจากงานวิจัยของ Mrdalj และคณะ (2004) ซึ่งกล่าวไว้ว่าจำนวนของข้อบกพร่องที่พบในแต่ละแผนภาพ เรียงจากมากที่สุดไปน้อยที่สุดเป็นดังนี้คือ (1) แผนภาพคลาส (2) แผนภาพซีเควนซ์ และ (3) แผนภาพสถานะ ดังนั้นผู้วิจัยจึงกำหนดข้อบกพร่องในแผนภาพคลาส และแผนภาพซีเควนซ์มากที่สุด ส่วนแผนภาพสถานะกำหนดข้อบกพร่องไว้ในจำนวนที่ไม่มากนัก และกำหนดข้อบกพร่องในแผนภาพยูสเคสไว้น้อยที่สุด เนื่องจากแผนภาพ ยูสเคสเป็นการแสดงฟังก์ชันการทำงานของระบบ ซึ่งเขียนขึ้นจากเอกสารแสดงความต้องการทางซอฟต์แวร์ ดังนั้นในการวาดแผนภาพยูสเคสจึงมีโอกาสผิดพลาดน้อยที่สุด จากเหตุผลดังที่กล่าวมาข้างต้นผู้วิจัยจึงใส่ข้อบกพร่องโดยแบ่งตามประเภทของแผนภาพดังตารางที่ 3-3

ตารางที่ 3-3 ตารางแสดงจำนวนข้อบกพร่องโดยแบ่งตามแผนภาพในเอกสารแสดงการออกแบบซอฟต์แวร์

ชื่อแผนภาพ	จำนวนข้อบกพร่อง
แผนภาพคลาสและคำอธิบายคลาส	9
แผนภาพซีเควนซ์	7
แผนภาพสถานะ	3
แผนภาพยูสเคสและคำอธิบายยูสเคส	1

การกำหนดข้อบกพร่องนั้นผู้วิจัยได้ปรับแก้ให้เหมาะสม โดยการให้หน่วยทดลองทดสอบทั้งหมด 2 รอบ (ดูรายละเอียดการทดสอบที่หัวข้อ 3.7.3) เพื่อปรับปรุงและแก้ไขตำแหน่งของการใส่ข้อบกพร่องให้เหมาะสม ซึ่งจะช่วยให้หน่วยทดลองสามารถค้นหาข้อบกพร่องได้ภายในระยะเวลาที่กำหนด (ดูรายละเอียดในภาคผนวก ก) ซึ่งรายละเอียดของข้อบกพร่องที่ใส่ในเอกสารแสดงการออกแบบซอฟต์แวร์แสดงได้ดังตารางที่ 3-4

ตารางที่ 3-4 ตารางแสดงรายละเอียดของข้อบกพร่องที่กำหนดลงในเอกสารแสดงการออกแบบซอฟต์แวร์

ข้อที่	แผนภาพ	รายละเอียดข้อบกพร่อง	ประเภทของข้อบกพร่อง	สามารถตรวจพบที่ Reading
1	แผนภาพยูสเคส	สัญลักษณ์แสดงการใช้ความสัมพันธ์แบบรวม (Include Relationship) ผิด (ที่ถูกจะต้องหันหัวลูกศรไปทางยูสเคส Search)	สิ่งผิดพลาด	-
2	แผนภาพคลาส	ไม่มีเมท็อด endCustomer แสดงในคลาส Customer	สิ่งสูญหาย	1,4
3	แผนภาพคลาส	ความสัมพันธ์ที่ชื่อ Register แสดงประเภทของความสัมพันธ์ผิด (ที่ถูกต้องใช้ความสัมพันธ์แบบแอสโซซิเอชัน)	สิ่งผิดพลาด	1,4
4	คำอธิบายคลาส	คลาส Customer แสดงแอตทริบิวต์ไม่ครบ	สิ่งสูญหาย	4,5
5	คำอธิบายคลาส	ไม่มีเมท็อด changeStatusBook ในคลาส Booking	สิ่งสูญหาย	2,4,5
6	คำอธิบายคลาส	มีการแสดงซับคลาสของคลาส Booking (ที่ถูกต้องไม่มีซับคลาส)	สิ่งผิดพลาด	4
7	คำอธิบายคลาส	พารามิเตอร์ของเมท็อด searchAvailableCar ผิด (ที่ถูกพารามิเตอร์ต้องประกอบด้วย CarType, StartDate และ EndDate)	สิ่งผิดพลาด	4
8	คำอธิบายคลาส	แสดงเงื่อนไขของระบบไม่ครบ	สิ่งสูญหาย	4,5
9	คำอธิบายคลาส	มีความสัมพันธ์ที่ชื่อ Search เกินมา	สิ่งเพิ่มเติม	4

ตารางที่ 3-4 (ต่อ) ตารางแสดงรายละเอียดของข้อบกพร่องที่กำหนดลงในเอกสารแสดงการออกแบบซอฟต์แวร์

ข้อที่	แผนภาพ	รายละเอียดข้อบกพร่อง	ประเภทของข้อบกพร่อง	สามารถตรวจพบที่ Reading
10	คำอธิบายคลาส	มีคลาส Officer เกินมา	สิ่งเพิ่มเติม	4
11	แผนภาพซีเควนซ์	ไม่มีเมสเซจที่ใช้ในการคืนข้อมูลการค้นหารถ	สิ่งสูญหาย	6
12	แผนภาพซีเควนซ์	เงื่อนไขการส่งเมสเซจชื่อ createBook ที่แสดงในแผนภาพ Book ผิด (ที่ถูกต้องเป็น Car ≠ null)	สิ่งผิดพลาด	1,3,6
13	แผนภาพซีเควนซ์	ข้อมูลที่ใช้ในการส่งเมสเซจ searchAvailableCar ที่แสดงในแผนภาพ Book ผิด (ที่ถูกต้องเป็น CarType, StartDate และ EndDate)	สิ่งผิดพลาด	1,6
14	แผนภาพซีเควนซ์	ไม่มีเมสเซจชื่อ changeStatusCar	สิ่งสูญหาย	3,6
15	แผนภาพซีเควนซ์	ข้อมูลที่ใช้ในการส่งเมสเซจ PickUp ที่แสดงในแผนภาพ PickUp ผิด (ที่ถูกต้องเป็น Book)	สิ่งผิดพลาด	1,6
16	แผนภาพซีเควนซ์	ข้อมูลที่ใช้ในการส่งเมสเซจ Return ที่แสดงในแผนภาพ Return ผิด (ที่ถูกต้องเป็น Book)	สิ่งผิดพลาด	1,6
17	แผนภาพซีเควนซ์	ไม่มีเมสเซจชื่อ endCustomer	สิ่งสูญหาย	1,6
18	แผนภาพสถานะ	ไม่มีทรานซิชันของการยกเลิกการเช่า	สิ่งสูญหาย	1,2,7
19	แผนภาพสถานะ	แผนภาพ Car มีสถานะชื่อ CarBooked เกินมา	สิ่งเพิ่มเติม	3,7
20	แผนภาพสถานะ	แผนภาพ Car มีทรานซิชันเกินมา	สิ่งเพิ่มเติม	3,7

3.6.2 การเตรียมเอกสารคำแนะนำ เอกสารคำแนะนำที่ใช้เป็นแนวทางในการตรวจสอบซอฟต์แวร์แบ่งออกเป็น 2 ประเภท ตามเทคนิคการอ่านซอฟต์แวร์คือ (1) สถานการณ์ (Scenario) เป็นคำแนะนำสำหรับกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบ (2) เช็กลิสต์ (Checklist) เป็นคำแนะนำสำหรับกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ

3.6.2.1 การเตรียมเอกสารสถานการณ์ (Scenario) ผู้วิจัยจัดทำเอกสารสถานการณ์ (Scenario)โดยการแปลจากงานวิจัยของ Travassos และคณะ (2002) ซึ่งในการแปลเอกสารสถานการณ์ (Scenario) นั้น อาจารย์ที่ปรึกษาได้ตรวจสอบความถูกต้องในการแปลของผู้วิจัยด้วย สำหรับรูปแบบของเอกสารสถานการณ์ (Scenario) จะแบ่งการตรวจสอบออกเป็น 7 ขั้นตอนหลัก เพื่อตรวจสอบความถูกต้องของแต่ละแผนภาพในเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการเปรียบเทียบแผนภาพต่างๆว่านำเสนอได้ถูกต้องตรงกันหรือไม่ ซึ่งแต่ละขั้นตอนของการตรวจสอบจะแสดงแผนภาพที่เปรียบเทียบ วัตถุประสงค์ เอกสารที่เข้าสู่กระบวนการ และขั้นตอนย่อยๆในการตรวจสอบ ดังรูปที่ 3-1 (ดูรายละเอียดในภาคผนวก ข)

Reading 1 – Sequence x Class Diagrams

Goal: To verify that the class diagram for the system describes classes and their relationships in such a way that the behaviors specified in the sequence diagrams are correctly captured. To do this, you will first check that the classes and objects specified in the sequence diagram appear in the class diagram. Then you will check that the class diagram describes relationships, behaviors, and conditions that capture the dynamic services as described on the sequence diagram.

Inputs to process:

1. A class diagram (possibly divided into packages) that describes the classes of a system and how they are associated.
2. Sequence diagrams that describe the classes, objects, and possibly actors of a system and how they collaborate to capture services of the system.

I. **Take a sequence diagram and read it to understand the system services described and how the system should implement those services.**

INPUTS: Sequence diagram (SD).

OUTPUTS: System objects (marked in blue on SD);
Services of the system (marked in green on SD);
Conditions on the services (marked in yellow on SD).

A. For each sequence diagram, underline the system objects and classes, and any actors, with a blue pen.

B. Underline the information exchanged between objects (the horizontal arrows) with a green pen. Consider whether this information represents *messages* or *services* of the system. If the information exchanged is very detailed, at the level of messages, you should abstract several messages together to understand the services they work to provide. Example 2 provides an illustration of messages being abstracted into services. Annotate the sequence diagram by writing down these services, and underline them in green also.

C. Circle any of the following constraints on the messages and services with a yellow pen: restrictions on the number of classes/objects to which a message can be sent, restrictions on the global values of an attribute, dependencies between data, or time constraints that can affect the state of the object. Also circle any conditions that determine under what circumstances a message will be sent. The sequence diagram in Example 2 contains several examples of constraints and conditions on messages. The conditions concerning payment type and payment time determine when messages authorize_payment and new_payment_type_request will be sent, while the restrictions on response_time for message authorize_payment represent time constraints.

รูปที่ 3-1 แสดงตัวอย่างเอกสารสถานการณ์ (Travassos และคณะ, 2002)

หลังจากผู้วิจัยแปลเอกสารสถานการณ์ (Scenario) เรียบร้อยแล้ว ผู้วิจัยได้ปรับปรุงรูปแบบเอกสารสถานการณ์ (Scenario)ให้มีความกระชับและสามารถเข้าใจได้ง่าย เพื่อนำมาใช้ในการทดลองที่มีการกำหนดระยะเวลา โดยการปรับเอกสารสถานการณ์ (Scenario) ให้อยู่ในรูปแบบของตาราง แสดงได้ดังรูปที่ 3-2 (ดูรายละเอียดในภาคผนวก ข)

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 3-2 แสดงตัวอย่างเอกสารสถานการณ์ (Scenario) ที่ใช้ในการทดลอง

ผู้วิจัยทดสอบเอกสารสถานการณ์ (Scenario) ก่อนนำไปใช้ในการทดลองจริง โดยการให้หน่วยทดลองนำเอกสารสถานการณ์ (Scenario) ไปใช้เป็นแนวทางในการตรวจสอบหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงแล้ว จากนั้นสอบถามปัญหาที่เกิดจากการใช้เอกสารสถานการณ์ (Scenario) เพื่อนำข้อแนะนำต่างๆมาปรับปรุงเอกสารสถานการณ์ (Scenario)ให้มีความเหมาะสมก่อนนำไปใช้ในการทดลองจริง (ดูรายละเอียดการทดสอบที่หัวข้อ 3.7.1 และ 3.7.2)

3.6.2.2 การเตรียมเอกสารเช็กลิสต์ (Checklist) ผู้วิจัยจัดทำเอกสารเช็กลิสต์โดยอ้างอิงจากโครงสร้างการทำเช็กลิสต์ของ Unhelkar (2005) และจากเช็กลิสต์ในงานวิจัยของ Sabalaiuskaite และคณะ (2002) ที่ได้จัดทำเช็กลิสต์สำหรับใช้ในการทดลองเพื่อเปรียบเทียบการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์กับเทคนิคอื่น และเช็กลิสต์ที่ Wang และ Lu (2001) จัดทำขึ้น (ดูรายละเอียดในภาคผนวก ข) โดยผู้วิจัยจัดทำเอกสารเช็กลิสต์ซึ่งประกอบด้วยคำถามใช่/ไม่ใช่ (Yes/No Question) จำนวน 20 คำถาม โดยเอกสารเช็กลิสต์ที่ผู้วิจัยจัดทำขึ้นมีรูปแบบดังรูปที่ 3-3 (ดูรายละเอียดในภาคผนวก ข)

Checklist ใช้สำหรับตรวจสอบเพื่อหาข้อบกพร่องใน Use Case, Class Diagram, Sequence Diagram และ State Machine Diagram โดยตอบคำถามที่เกี่ยวข้องกับแผนภาพเหล่านี้ เมื่อพบข้อบกพร่องให้บันทึกลงในเอกสารบันทึกข้อบกพร่อง		
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

รูปที่ 3-3 แสดงตัวอย่างเช็กลิสต์ที่ใช้ในการทดลอง

ผู้วิจัยทดสอบเอกสารเช็กลิสต์ที่จัดทำขึ้นก่อนนำไปใช้ในการทดลอง โดยการให้หน่วยทดลองใช้เอกสารเช็กลิสต์นี้เป็นแนวทางในการตรวจสอบหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงแล้ว จากนั้นสอบถามปัญหาที่เกิดจากการใช้เอกสารเช็กลิสต์ และให้ผู้เชี่ยวชาญด้านการตรวจสอบซอฟต์แวร์ตรวจสอบเอกสารเช็กลิสต์อีกครั้ง เพื่อนำข้อแนะนำต่างๆมาปรับปรุงเอกสารเช็กลิสต์ให้มีความเหมาะสมก่อนนำไปใช้ในการทดลอง (ดูรายละเอียดการทดสอบที่หัวข้อ 3.7.2)

3.6.3 การเตรียมเอกสารบันทึกข้อบกพร่อง การทดลองนี้มีการใช้เอกสารบันทึกข้อบกพร่อง 2 แบบคือ (1) เอกสารบันทึกข้อบกพร่องในขั้นตอนการจัดเตรียม และ (2) เอกสารบันทึกข้อบกพร่องในขั้นตอนการประชุมตรวจสอบ โดยผู้วิจัยจัดทำให้เอกสารบันทึกข้อบกพร่องสามารถเก็บข้อมูลที่ต้องการนำไปวิเคราะห์ได้อย่างครบถ้วน

3.6.3.1 การเตรียมเอกสารบันทึกข้อบกพร่องขั้นตอนการจัดเตรียม ผู้วิจัยจัดทำเอกสารบันทึกข้อบกพร่องขั้นตอนการจัดเตรียม โดยให้หน่วยทดลองสามารถบันทึกชื่อ เวลาที่ใช้ในการทำขั้นตอนการจัดเตรียม และรายละเอียดของข้อบกพร่อง แสดงได้ดังรูปที่ 3-4 (ดูรายละเอียดในภาคผนวก ค)

เอกสารบันทึกข้อบกพร่อง (Preparation)

รหัสผู้ตรวจสอบ.....

เวลาที่ใช้.....

ข้อ	Reading / ข้อ	ชื่อเอกสาร	ตำแหน่งที่พบ	คำอธิบาย
1.				

รูปที่ 3-4 แสดงตัวอย่างเอกสารบันทึกข้อบกพร่องสำหรับขั้นตอนการจัดเตรียม

3.6.3.2 การเตรียมเอกสารบันทึกข้อบกพร่องขั้นตอนการประชุมตรวจสอบ ผู้วิจัยจัดทำเอกสารบันทึกข้อบกพร่องขั้นตอนการประชุมตรวจสอบ โดยให้กลุ่มทดลองสามารถบันทึกรายชื่อของหน่วยทดลองในกลุ่ม เวลาที่ใช้ในการทำขั้นตอนการประชุมตรวจสอบ และรายละเอียดของข้อบกพร่องที่พบ แสดงได้ดังรูปที่ 3-5 (ดูรายละเอียดในภาคผนวก ค)

เอกสารบันทึกข้อบกพร่อง (Meeting)

รหัสผู้ตรวจสอบ 1).....

2).....

3).....

เวลาที่ใช้.....

ข้อ	Reading / ข้อ	ชื่อเอกสาร	ตำแหน่งที่พบ	คำอธิบาย
1.				

รูปที่ 3-5 แสดงตัวอย่างเอกสารบันทึกข้อบกพร่องสำหรับขั้นตอนการประชุมตรวจสอบ

เมื่อผู้วิจัยจัดทำเอกสารบันทึกข้อบกพร่องทั้ง 2 แบบเรียบร้อยแล้ว ผู้วิจัยจัดให้ทดสอบเอกสารบันทึกข้อบกพร่อง โดยการให้หน่วยทดลองนำเอกสารบันทึกข้อบกพร่องไปใช้บันทึกข้อบกพร่องในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ เพื่อดูว่าเอกสารบันทึกข้อบกพร่องที่จัดทำขึ้นนั้นสามารถบันทึกข้อมูลที่นำไปใช้ในการตอบวัตถุประสงค์ได้อย่างครบถ้วน (ดูรายละเอียดการทดสอบที่หัวข้อ 3.7.1)

3.6.4 การเตรียมเอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล เป็นเอกสารแสดงความหมายของสัญลักษณ์ต่างๆที่แสดงในแผนภาพยูเอ็มแอล ซึ่งจะประกอบด้วยสัญลักษณ์

ของแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ โดยเอกสารนี้จะช่วยให้หน่วยทดลองอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ได้เข้าใจมากยิ่งขึ้น (ดูรายละเอียดในภาคผนวก ง)

3.7 การทดสอบเครื่องมือที่ใช้ในการทดลอง (Pretest)

ผู้วิจัยจัดให้มีการทดสอบเครื่องมือที่ใช้ในการทดลอง เพื่อศึกษาปัญหาของเอกสารต่างๆที่จะนำไปใช้ในการทดลองซึ่งได้แก่ เอกสารแสดงการออกแบบซอฟต์แวร์ เอกสารคำแนะนำ และเอกสารบันทึกข้อบกพร่อง ว่าควรปรับปรุงอย่างไรให้มีความถูกต้องและเหมาะสมก่อนที่จะนำเอกสารเหล่านั้นไปใช้ในการทดลอง สำหรับหน่วยทดลองที่มาทดสอบเครื่องมือที่ใช้ในการทดลองนั้น เป็นหน่วยทดลองที่มีคุณสมบัติตามที่ผู้วิจัยได้กำหนดไว้ข้างต้น ซึ่งผู้วิจัยได้ทดสอบเครื่องมือที่ใช้ในการทดลอง (Pretest) ทั้งหมด 4 ครั้ง แต่ละครั้งของการทดสอบจะใช้หน่วยทดลองไม่ซ้ำกัน โดยกระทำการทดสอบในช่วงเดือนกรกฎาคม พ.ศ. 2549 ซึ่งมีรายละเอียดการทดสอบเครื่องมือที่ใช้ในการทดลองดังต่อไปนี้

3.7.1 การทดสอบครั้งที่ 1 จัดทำขึ้นเพื่อตรวจสอบว่าเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงจากต้นฉบับแล้วนั้นมีความถูกต้องหรือไม่ และสถานการณ์ (Scenario) ที่ผู้วิจัยปรับปรุงจากต้นฉบับนั้น หน่วยทดลองอ่านแล้วเข้าใจหรือไม่ นอกจากนี้ยังเป็นการตรวจสอบเอกสารบันทึกข้อบกพร่องอีกด้วย ว่าสามารถบันทึกข้อมูลที่ผู้วิจัยต้องการนำไปใช้ในการตอบวัตถุประสงค์ของงานวิจัยได้อย่างครบถ้วนหรือไม่

3.7.1.1 วิธีการทดสอบ หน่วยทดลองจำนวน 1 คนตรวจสอบหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงแล้ว (ไม่ได้ใส่ข้อบกพร่องลงในเอกสารแสดงการออกแบบซอฟต์แวร์) โดยการใช้สถานการณ์ (Scenario) ที่ผู้วิจัยปรับปรุงจากต้นฉบับเป็นเอกสารคำแนะนำสำหรับตรวจสอบ สำหรับการทดสอบครั้งที่ 1 นี้ ผู้วิจัยไม่ได้มีการกำหนดระยะเวลาในการตรวจสอบ เนื่องจากผู้วิจัยต้องการให้หน่วยทดลองค้นหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงแล้ว เพื่อนำผลการตรวจสอบที่ได้มาแก้ไขให้เอกสารแสดงการออกแบบซอฟต์แวร์มีความถูกต้องมากที่สุด

3.7.1.2 ผลที่ได้จากการทดสอบ ผู้วิจัยแก้ไขเอกสารแสดงการออกแบบซอฟต์แวร์ให้มีความถูกต้อง และปรับปรุงสถานการณ์ (Scenario) ให้มีความกระชับและสามารถเข้าใจได้ง่าย

3.7.2 การทดสอบครั้งที่ 2 จัดทำขึ้นเพื่อตรวจสอบว่าเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงจากการทดสอบครั้งที่ 1 แล้วนั้นมีความถูกต้องหรือไม่ และเช็กลิสต์ที่ผู้วิจัยจัดทำขึ้นนั้นหน่วยทดลองอ่านแล้วเข้าใจหรือไม่ การทดสอบครั้งนี้ใช้หน่วยทดลองจำนวน 1 คน

3.7.2.1 วิธีการทดสอบ หน่วยทดลองจำนวน 1 คนตรวจสอบหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงจากการทดสอบครั้งที่ 1 (ไม่ได้ใส่ข้อบกพร่องลงในเอกสารแสดงการออกแบบซอฟต์แวร์) โดยการใช้เช็กลิสต์เป็นเอกสารคำแนะนำสำหรับตรวจสอบ สำหรับการทดสอบครั้งที่ 2 นี้ ผู้วิจัยไม่ได้มีการกำหนดระยะเวลาในการตรวจสอบ เนื่องจากผู้วิจัยต้องการให้หน่วยทดลองค้นหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงแล้ว เพื่อนำผลการตรวจสอบที่ได้มาแก้ไขให้เอกสารแสดงการออกแบบซอฟต์แวร์มีความถูกต้องมากที่สุด

3.7.2.2 ผลที่ได้จากการทดสอบ ผู้วิจัยแก้ไขเอกสารแสดงการออกแบบซอฟต์แวร์ให้มีความถูกต้อง และปรับปรุงเช็กลิสต์ให้มีความชัดเจนและสามารถเข้าใจได้ง่าย

3.7.3 การทดสอบครั้งที่ 3 จัดทำขึ้นเพื่อตรวจสอบว่าการใส่ข้อบกพร่องลงในเอกสารแสดงการออกแบบซอฟต์แวร์นั้น มีความเหมาะสมกับระยะเวลาที่ผู้วิจัยกำหนดไว้หรือไม่ การทดสอบครั้งนี้ใช้หน่วยทดลองจำนวน 1 คน

3.7.3.1 วิธีการทดสอบ หน่วยทดลองจำนวน 1 คนตรวจสอบหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยใส่ข้อบกพร่องไว้เรียบร้อยแล้ว โดยการใช้สถานการณ์ (Scenario) ที่ผู้วิจัยปรับปรุงจากการทดสอบครั้งที่ 1 เป็นเอกสารคำแนะนำสำหรับการตรวจสอบ โดยการทดสอบครั้งที่ 3 นี้ ผู้วิจัยกำหนดระยะเวลาในการตรวจสอบตามที่ได้กำหนดไว้ในระเบียบวิธีวิจัย เนื่องจากผู้วิจัยต้องการทราบว่าการใส่ข้อบกพร่องในลักษณะนี้มีความเหมาะสมกับระยะเวลาที่ผู้วิจัยกำหนดไว้หรือไม่ กล่าวคือหน่วยทดลองสามารถค้นหาข้อบกพร่องได้ตามเวลาที่ทางผู้วิจัยได้กำหนดไว้หรือไม่ เพื่อให้การใส่ข้อบกพร่องที่กำหนดลงในเอกสารนั้นไม่ยากเกินไปจนไม่สามารถค้นหาข้อบกพร่องได้ตามเวลาที่กำหนด หรือง่ายเกินไปจนไม่สามารถนำไปใช้ในการเปรียบเทียบเพื่อตอบวัตถุประสงค์ได้ โดยการทดสอบครั้งนี้ทำ 2 รอบ เนื่องจากพบว่าผลจากการทำรอบแรกนั้น หน่วยทดลองไม่สามารถค้นหาข้อบกพร่องได้ จึงมีการปรับการใส่ข้อบกพร่องใหม่อีกครั้งเพื่อให้หน่วยทดลองสามารถค้นหาข้อบกพร่องได้ทันตามระยะเวลาที่กำหนด ซึ่งในรอบที่สองหน่วยทดลองสามารถหาข้อบกพร่องได้ในระยะเวลาที่กำหนด

3.7.3.2 ผลที่ได้จากการทดสอบ ผู้วิจัยปรับปรุงการใส่ข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ เพื่อให้หน่วยทดลองสามารถค้นหาข้อบกพร่องได้ในระยะเวลาที่กำหนด และปรับปรุงสถานการณ์ (Scenario) ให้เข้าใจและมีความกระชับมากยิ่งขึ้น

3.7.4 การทดสอบครั้งที่ 4 จัดทำขึ้นเพื่อตรวจสอบความเที่ยงตรงของเครื่องมือเชิงประจักษ์ (Face Validity) เพื่อให้เอกสารต่างๆที่นำไปใช้ในการทดลองนั้นมีความถูกต้องมากที่สุด การทดสอบครั้งนี้ใช้หน่วยทดลองจำนวน 4 คน

3.7.4.1 วิธีการทดสอบ ผู้วิจัยแบ่งหน่วยทดลอง 2 คนตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ผู้วิจัยปรับปรุงการใส่ข้อบกพร่องตามผลที่ได้จากการทดสอบครั้งที่ 3 โดยการใช้สถานการณ์ (Scenario) ที่ผู้วิจัยปรับปรุงจากผลการทดสอบครั้งที่ 3 เป็นเอกสารคำแนะนำ และหน่วยทดลองอีก 2 คนใช้เช็กลิสต์ที่ผู้วิจัยปรับปรุงจากผลการทดสอบครั้งที่ 2 เป็นเอกสารคำแนะนำ ซึ่งในการทดลองมีการกำหนดระยะเวลาตามที่ได้กำหนดไว้ในระเบียบวิธีวิจัย เพื่อเป็นการตรวจสอบความเที่ยงตรงของเครื่องมือเป็นครั้งสุดท้ายก่อนการทดลองจริง

3.7.4.2 ผลที่ได้จากการทดสอบ ผู้วิจัยเพิ่มเอกสารคู่มือแสดงสัญลักษณ์ของแผนภาพยูเอ็มแอล เพื่อให้หน่วยทดลองสามารถค้นหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ใช้แผนภาพยูเอ็มแอลได้ง่ายขึ้น

3.8 การเก็บรวบรวมข้อมูล

งานวิจัยนี้แบ่งหน่วยทดลองด้วยวิธีการสุ่มหน่วยทดลองออกเป็น 2 กลุ่มคือ (1) กลุ่มที่ใช้เทคนิคโอโออาร์ที (2) กลุ่มที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล โดยในการเก็บข้อมูลแต่ละครั้งจะเก็บอย่างน้อย 1 กลุ่มทดลอง (1 กลุ่มทดลองคือหน่วยทดลอง 3 คน) เนื่องจากในการทำขั้นตอนการประชุมตรวจสอบเป็นขั้นตอนที่ต้องทำในลักษณะเป็นกลุ่มทดลอง โดยในการทดลองแต่ละครั้งจะทำครบทุกขั้นตอนต่อเนื่องกันไปคือ ขั้นตอนการประชุมแนะนำ ขั้นตอนการจัดเตรียม ขั้นตอนการประชุมตรวจสอบ โดยมีขั้นตอนการเก็บข้อมูลดังต่อไปนี้

1. ผู้วิจัยเริ่มขั้นตอนการประชุมแนะนำของการตรวจสอบซอฟต์แวร์ โดยการแจกเอกสารต่างๆที่ใช้ในการทดลองได้แก่ เอกสารแสดงการออกแบบซอฟต์แวร์ เอกสารคำแนะนำที่ใช้เป็นแนวทางในการตรวจสอบ (เช็กลิสต์สำหรับกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ และสถานการณ์ (Scenario) สำหรับกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที) เอกสารที่ใช้บันทึกข้อบกพร่อง และเอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล จากนั้นอธิบายระบบและเทคนิคการอ่านซอฟต์แวร์ที่นำมาใช้ตรวจสอบให้หน่วยทดลองเข้าใจ เพื่อให้หน่วยทดลองสามารถค้นหาข้อบกพร่องได้อย่างถูกต้อง ซึ่งผู้วิจัยกำหนดเวลาที่ใช้ในการทำขั้นตอนประชุมแนะนำไม่เกิน 30 นาที โดยในการกำหนดเวลาที่ใช้ในการทำขั้นตอนการประชุมแนะนำ ผู้วิจัยพิจารณาจากความซับซ้อนของระบบ และจำนวนหน้าของเอกสารแสดงการออกแบบซอฟต์แวร์ที่นำมาตรวจสอบ

2. หลังจากอธิบายระบบและเทคนิคการอ่านซอฟต์แวร์ให้หน่วยทดลองเข้าใจเรียบร้อยแล้วจะเข้าสู่ขั้นตอนการจัดเตรียม นั่นคือหน่วยทดลองแต่ละหน่วยจะค้นหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการใช้เอกสารคำแนะนำที่ได้รับมาเป็นแนวทางเพื่อช่วยในการ

ค้นหาข้อบกพร่อง เมื่อหน่วยทดลองพบข้อบกพร่องจะบันทึกข้อบกพร่องเหล่านั้นในเอกสารที่ใช้บันทึกข้อบกพร่องสำหรับขั้นตอนการจัดเตรียม โดยบันทึกรายละเอียดข้อบกพร่อง ชื่อแผนภาพที่พบข้อบกพร่องนั้น ตำแหน่งของข้อบกพร่องในแผนภาพ และบันทึกระยะเวลาที่ใช้ในการตรวจสอบขั้นตอนการจัดเตรียม

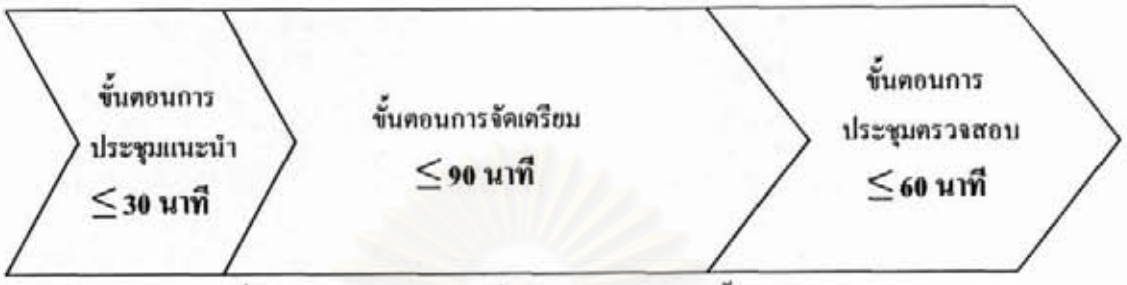
ผู้วิจัยกำหนดให้หน่วยทดลองใช้เวลาในขั้นตอนการจัดเตรียมไม่เกิน 90 นาที โดยการกำหนดเวลาที่ใช้ในการทำขั้นตอนการจัดเตรียมนั้น ผู้วิจัยพิจารณาจากงานวิจัยอื่นๆที่มีลักษณะการวิจัยคล้ายกับงานวิจัยนี้ ซึ่งจะกำหนดอยู่ที่ 60 – 120 นาที (Porter และคณะ, 1995; Sabaliauskaite และคณะ, 2002; Sabaliauskaite และคณะ, 2004) ซึ่งผู้วิจัยพิจารณาแล้วว่าระยะเวลา 90 นาทีเป็นระยะเวลาที่เหมาะสมกับขั้นตอนนี้ เนื่องจาก 60 นาทีนั้นเป็นระยะเวลาที่น้อยเกินไป หน่วยทดลองไม่สามารถที่จะตรวจสอบหาข้อบกพร่องได้ทัน และสำหรับ 120 นาทีนั้นเป็นระยะเวลาที่นานเกินไป เนื่องจากหน่วยทดลองยังต้องตรวจสอบในขั้นตอนการประชุมตรวจสอบต่อจากขั้นตอนนี้ ถ้าในขั้นตอนนี้ใช้ระยะเวลานานอาจส่งผลให้หน่วยทดลองมีความเหนื่อยล้าเกินไปในการที่จะตรวจสอขั้นตอนการประชุมตรวจสอบให้ถูกต้องได้

3. หลังจากทำขั้นตอนการจัดเตรียมเสร็จเรียบร้อยแล้ว ผู้วิจัยจะกำหนดกลุ่มทดลองกลุ่มละ 3 คน เพื่อเข้าสู่ขั้นตอนการประชุมตรวจสอบ

4. เมื่อกำหนดกลุ่มทดลองเรียบร้อยแล้วจะเข้าสู่ขั้นตอนการประชุมตรวจสอบ โดยแต่ละหน่วยทดลองภายในกลุ่มทดลองจะอ่านเอกสารแสดงการออกแบบซอฟต์แวร์ร่วมกันเพื่อหาข้อบกพร่อง และนำเอกสารบันทึกข้อบกพร่องที่ได้จากขั้นตอนการจัดเตรียมของแต่ละหน่วย มาปรึกษากันว่าข้อบกพร่องที่แต่ละหน่วยทดลองพบนั้นเป็นข้อบกพร่องหรือไม่ จากนั้นบันทึกข้อบกพร่องที่ทั้งกลุ่มเห็นชอบว่าเป็นข้อบกพร่องลงเอกสารที่ใช้บันทึกข้อบกพร่องสำหรับขั้นตอนการประชุมตรวจสอบ โดยการบันทึกรายละเอียดข้อบกพร่อง ชื่อแผนภาพที่พบข้อบกพร่องนั้น ตำแหน่งของข้อบกพร่องในแผนภาพ และบันทึกระยะเวลาที่ใช้ในการทำขั้นตอนการประชุมตรวจสอบ

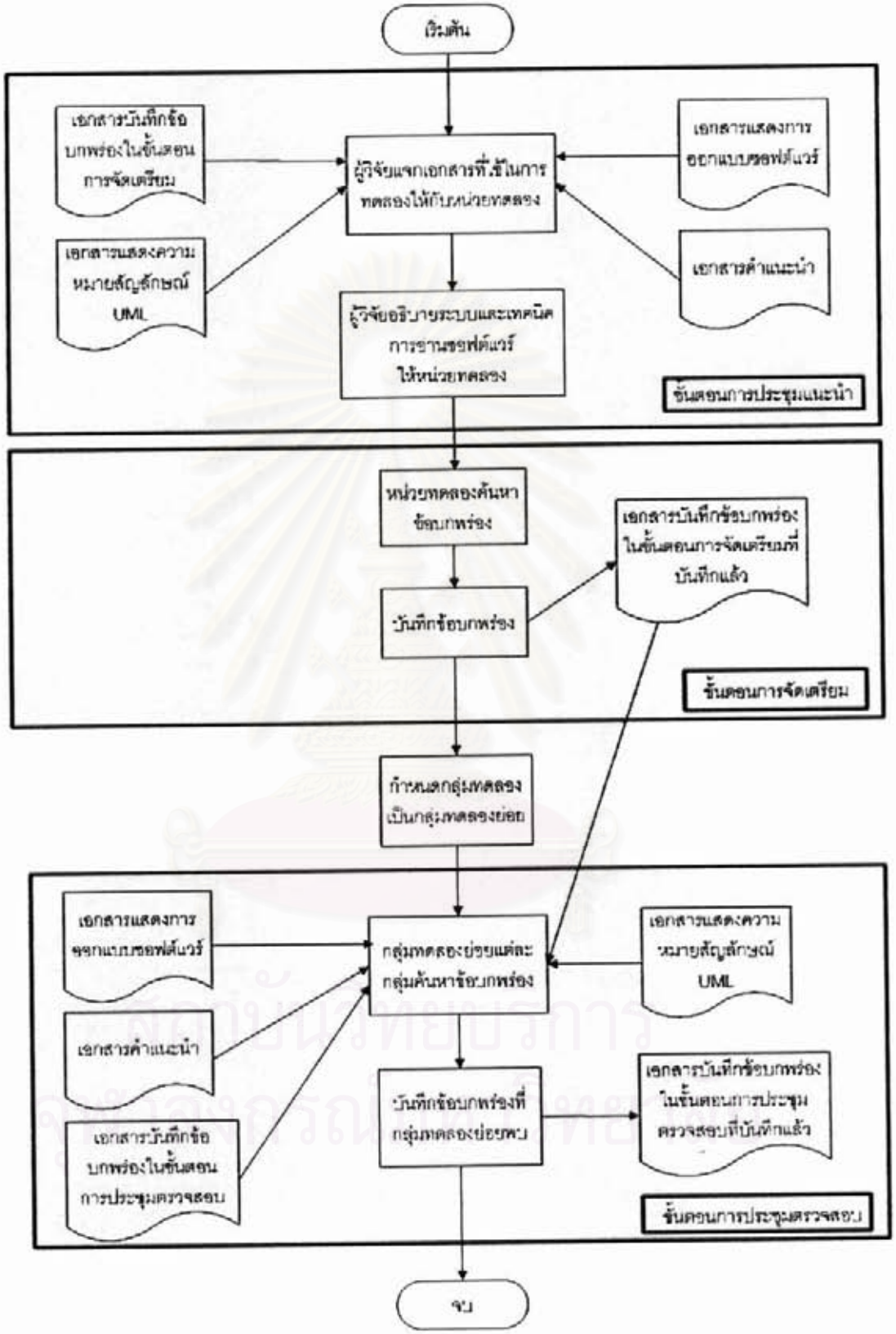
ผู้วิจัยกำหนดให้หน่วยทดลองใช้เวลาในการทำขั้นตอนการประชุมตรวจสอบไม่เกิน 60 นาที ซึ่งการที่ผู้วิจัยกำหนดเวลาที่ใช้ในขั้นตอนการประชุมตรวจสอบน้อยกว่าขั้นตอนการจัดเตรียม เนื่องจากหน่วยทดลองได้ทำความเข้าใจกับระบบและเทคนิคการตรวจสอบซอฟต์แวร์ตั้งแต่ขั้นตอนการจัดเตรียมแล้ว ดังนั้นเวลาที่ใช้ในขั้นตอนการประชุมตรวจสอบจึงน้อยกว่าที่ใช้ในขั้นตอนการจัดเตรียม นอกจากนี้ผู้วิจัยยังพิจารณาจากงานวิจัยอื่นๆที่มีลักษณะการวิจัยคล้ายกับงานวิจัยนี้ ซึ่งมีการลดระยะเวลาในการทำขั้นตอนการประชุมตรวจสอบให้น้อยกว่าการทำขั้นตอนการจัดเตรียม (Porter และคณะ, 1995; Sabaliauskaite และคณะ, 2002; Sabaliauskaite และคณะ,

2004) จากขั้นตอนต่างๆข้างต้นใช้เวลาทั้งหมดไม่เกิน 3 ชั่วโมงต่อเนื่องกัน สามารถแสดงระยะเวลาที่ใช้ในการทดลองแต่ละขั้นตอนได้ดังรูปที่ 3-6



รูปที่ 3-6 แสดงระยะเวลาที่ใช้ในการทำแต่ละขั้นตอนของการทดลอง

รูปที่ 3-7 แสดงขั้นตอนที่ผู้วิจัยออกแบบเพื่อใช้ในการเก็บข้อมูลจากหน่วยทดลอง โดยในการเก็บข้อมูลกระทำในช่วงเดือนสิงหาคม พ.ศ. 2549 สถานที่ในการทดลองคือใต้อาคารมหิดลาธิเบศร์ และใต้อาคารอนุสรณ์ 50 ปี คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ 3-7 แสดงขั้นตอนที่ผู้วิจัยออกแบบเพื่อใช้ในการเก็บข้อมูลจากหน่วยทดลอง

3.9 ความถูกต้อง (Validity) และความน่าเชื่อถือ (Reliability) ของข้อมูลที่เก็บ

ในการตอบวัตถุประสงค์ให้มีความถูกต้องและน่าเชื่อถือนั้น จำเป็นต้องควบคุมปัจจัยอื่นที่อาจส่งผลกับตัวแปรตามได้แก่ การเลือกหน่วยทดลอง เครื่องมือที่ใช้ในการทดลอง และสภาพแวดล้อมของการทดลอง เพื่อให้ผลการทดลองที่ได้เกิดจากเทคนิคการอ่านซอฟต์แวร์เท่านั้นโดยมีวิธีการดังต่อไปนี้

3.9.1 การเลือกหน่วยทดลอง งานวิจัยนี้เป็นการเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยการใช้แผนภาพยูเอ็มแอล ดังนั้นหน่วยทดลองที่ผู้วิจัยต้องการนำมาใช้ในการทดลองคือ ผู้ตรวจสอบทั้งหมดที่มาจากองค์กรรับจ้างพัฒนาซอฟต์แวร์ ซึ่งในทางปฏิบัติแล้วไม่สามารถทำได้ และเพื่อให้คุณสมบัติของหน่วยทดลองที่จะมาเป็นตัวแทนของประชากรในการทดลอง มีความใกล้เคียงกับประชากรที่ผู้วิจัยต้องการมากที่สุด ผู้วิจัยจึงเลือกตัวแทนของประชากรเป็นนิสิตปัจจุบัน (ปีการศึกษา 2549) ในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการพัฒนาซอฟต์แวร์เชิงวัตถุ

เนื่องจากนิสิตปริญญามหาบัณฑิตในสาขาวิชานี้มีคุณสมบัติพร้อมที่จะทำงานในบริษัท ซึ่งสามารถเทียบได้กับผู้ตรวจสอบที่ทำงานในองค์กรรับจ้างพัฒนาซอฟต์แวร์ได้ และหน่วยทดลองที่ศึกษาในสาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการออกแบบและพัฒนาซอฟต์แวร์เชิงวัตถุ เป็นผู้ที่มีความรู้ความเข้าใจในการออกแบบซอฟต์แวร์โดยการใช้แผนภาพยูเอ็มแอล จึงสามารถที่จะเป็นผู้ตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอลได้ การที่ผู้วิจัยควบคุมการเก็บข้อมูลตามที่ได้กล่าวมานี้ ช่วยให้งานวิจัยมีความเที่ยงตรงภายใน (Internal Validity) สูง กล่าวคือสามารถสรุปผลอ้างอิงไปหากลุ่มประชากรเป้าหมายได้อย่างถูกต้อง แต่จะมีค่าความเที่ยงตรงภายนอก (External Validity) ต่ำ กล่าวคือผลที่ได้จากงานวิจัยไม่สามารถใช้อธิบายได้โดยทั่วไป (Generalization)

3.9.2 เครื่องมือที่ใช้ในการทดลอง ประกอบด้วย 4 ส่วนคือ

1. เอกสารแสดงการออกแบบซอฟต์แวร์ ผู้วิจัยเลือกเอกสารแสดงการออกแบบซอฟต์แวร์ระบบด้านธุรกิจ ที่มีความถูกต้องสมบูรณ์และมีความน่าเชื่อถือ เนื่องจากเอกสารแสดงการออกแบบซอฟต์แวร์นี้มีการตรวจสอบความถูกต้องตรงกันของแต่ละแผนภาพ โดยในการกำหนดข้อบกพร่องแต่ละประเภทลงในเอกสารแสดงการออกแบบซอฟต์แวร์นั้น ทางผู้วิจัยจะสุ่มหน่วยทดลองขึ้นมาจำนวนหนึ่งเพื่อทำการทดสอบเครื่องมือที่ใช้ในการทดลอง ว่าหน่วยทดลอง

สามารถค้นหาข้อบกพร่องได้ตามเวลาที่ทางผู้วิจัยได้กำหนดไว้หรือไม่ เพื่อให้ข้อบกพร่องที่กำหนดลงในเอกสารนั้นไม่ยากเกินไปจนไม่สามารถค้นหาข้อบกพร่องได้ตามเวลาที่กำหนด หรือง่ายเกินไปจนไม่สามารถนำไปใช้ในการเปรียบเทียบเพื่อตอบวัตถุประสงค์ได้

2. เอกสารคำแนะนำ เป็นเอกสารที่ใช้เป็นแนวทางในการตรวจสอบซอฟต์แวร์ ซึ่งแบ่งออกเป็น 2 ประเภทตามเทคนิคการอ่านซอฟต์แวร์ดังนี้ (1) เช็กลิสต์ซึ่งผู้วิจัยได้จัดทำขึ้นโดยการอ้างอิงกับงานวิจัย 3 งานคืองานวิจัยของ Sabalaiuskaite และคณะ (2002) ที่ได้จัดทำเช็กลิสต์สำหรับใช้ในการทดลองเพื่อเปรียบเทียบกับเทคนิคอื่นแล้ว พบว่าเช็กลิสต์ที่จัดทำขึ้นนั้นช่วยเพิ่มประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ งานวิจัยของ Wang และ Lu (2001) ที่พบว่าโดยส่วนมากแล้วแต่ละแผนภาพนั้นมีข้อบกพร่องตรงส่วนไหน ซึ่งทำให้เช็กลิสต์นี้มีความน่าเชื่อถือ และเช็กลิสต์ของ Unhelkar (2005) ซึ่งเป็นเช็กลิสต์ในหนังสือที่เกี่ยวกับการตรวจสอบแผนภาพยูเอ็มแอลเพื่อให้แผนภาพมีคุณภาพ ซึ่งผู้วิจัยจะจัดทำเช็กลิสต์ขึ้นมาโดยประกอบด้วยคำถามใช่/ไม่ใช่ (Yes/No Question) จำนวน 20 คำถาม เนื่องจากงานวิจัยของ Gilb และ Graham (1993) กล่าวว่าเช็กลิสต์ที่ดีไม่ควรมีความยาวเกิน 1 หน้ากระดาษ หรือประมาณ 25 ข้อ จากการเลือกเช็กลิสต์ที่มีความน่าเชื่อถือมาปรับใช้ให้เหมาะสมกับงานวิจัยนี้ ทำให้การเก็บข้อมูลมีความถูกต้องและน่าเชื่อถือ นอกจากนี้ผู้วิจัยได้ทดสอบเครื่องมือที่ใช้ในการทดลอง (Pretest) เพื่อตรวจสอบว่าเอกสารเช็กลิสต์ที่จัดทำขึ้นนั้น หน่วยทดลองอ่านแล้วเข้าใจ (2) สถานการณ์ (Scenario) เป็นเอกสารคำแนะนำที่เทคนิคโอโออาร์ทีกำหนดเป็นรูปแบบที่แน่นอน สำหรับนำไปใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ถูกพัฒนาโดยการใช้แผนภาพยูเอ็มแอล ดังนั้นจึงมีความน่าเชื่อถือในการนำไปใช้ในการทดลอง นอกจากนี้ผู้วิจัยได้ทดสอบเครื่องมือที่ใช้ในการทดลอง (Pretest) เพื่อตรวจสอบว่าเอกสารสถานการณ์ (Scenario) ที่จัดทำขึ้นนั้น หน่วยทดลองอ่านแล้วเข้าใจ

3. เอกสารที่ใช้บันทึกข้อบกพร่องที่พบ ใช้ในการบันทึกผลการทดลองโดยแบ่งออกเป็น 2 เอกสาร (1) เอกสารที่ใช้บันทึกข้อบกพร่องที่พบในขั้นตอนการจัดเตรียม (2) เอกสารที่ใช้บันทึกข้อบกพร่องที่พบในขั้นตอนการประชุมตรวจสอบ ซึ่งเอกสารทั้งสองนี้ถูกออกแบบให้สามารถเก็บข้อมูลที่ต้องการนำไปวิเคราะห์ได้อย่างครบถ้วน ทำให้ผู้วิจัยสามารถเก็บข้อมูลจากการทดลองได้ครบถ้วนสมบูรณ์

4. เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล เป็นเอกสารที่แสดงความหมายสัญลักษณ์ต่างๆของแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ ซึ่งเอกสารนี้จะช่วยให้หน่วยทดลองสามารถเข้าใจเอกสารแสดงการออกแบบซอฟต์แวร์ได้ง่ายขึ้น โดยเอกสารนี้อ้างอิงตามสัญลักษณ์แผนภาพยูเอ็มแอล 2.0 (Dennis และคณะ, 2005)

ก่อนการทดลองผู้วิจัยจะให้ผู้ที่มีประสบการณ์ และมีความเชี่ยวชาญทางด้านการตรวจสอบซอฟต์แวร์ ตรวจสอบเอกสารเพื่อดูความถูกต้อง และความเหมาะสมในการนำเอกสารเหล่านี้ไปใช้ในการทดลอง

3.9.3 วิธีการเก็บข้อมูล ผู้วิจัยกำหนดให้การทดลองแต่ละครั้งเป็นการทดลองอย่างน้อย 1 กลุ่มทดลอง (1 กลุ่มทดลองคือหน่วยทดลอง 3 คน) เนื่องจากการทดลองขั้นตอนการประชุมตรวจสอบต้องทำเป็นกลุ่มทดลองกลุ่มละ 3 คน การที่ผู้วิจัยจัดการทดลองแยกเป็นหลายครั้งครั้งละอย่างน้อย 1 กลุ่มทดลอง เนื่องจากโอกาสที่หน่วยทดลองจะสามารถมาทดลองในระยะเวลาการทดลองทั้งหมด 3 ชั่วโมงพร้อมกันเป็นไปได้ยาก ผู้วิจัยจึงต้องแบ่งการทดลองออกเป็นหลายครั้งตามความสะดวกของหน่วยทดลอง โดยทุกครั้งที่ทดลองนั้นผู้วิจัยจะอธิบายเอกสารแสดงการออกแบบซอฟต์แวร์และเอกสารคำแนะนำเหมือนกันทุกครั้ง เพื่อควบคุมให้ทุกหน่วยทดลองมีความเข้าใจในเอกสารแสดงการออกแบบซอฟต์แวร์ และเอกสารคำแนะนำตรงกัน และควบคุมระยะเวลาแต่ละขั้นตอนที่ใช้ในการทดลองเหมือนกันทุกครั้ง สำหรับสถานที่ที่ใช้ในการทดลองเป็นได้อาคารมหิดลาธิเบศรและได้อาคารอนุสรณ์ 50 ปี สาเหตุที่ผู้วิจัยเลือกสถานที่นี้เนื่องจากต้องการให้หน่วยทดลองอยู่ในสถานที่ปลอดโปร่ง สามารถดื่มน้ำและรับประทานขนมได้ เพื่อไม่ให้หน่วยทดลองเกิดความเครียดและอึดอัดในการทดลองมากเกินไป และในขั้นตอนการประชุมตรวจสอบเป็นขั้นตอนที่หน่วยทดลองต้องพูดจาแลกเปลี่ยนความเห็นกัน ดังนั้นสถานที่ที่ใช้ในการทดลองจึงต้องเป็นที่ที่สามารถใช้เสียงได้ในระดับหนึ่ง

3.10 กรอบการวิเคราะห์ข้อมูล

แหล่งเก็บข้อมูลของงานวิจัยนี้ได้มาจากเอกสารบันทึกข้อบกพร่องในขั้นตอนการจัดเตรียมและเอกสารบันทึกข้อบกพร่องในขั้นตอนการประชุมตรวจสอบ ซึ่งข้อมูลจากทั้ง 2 เอกสารนี้จะนำไปวิเคราะห์เพื่อตอบวัตถุประสงค์ และทดสอบสมมติฐานของงานวิจัย ซึ่งสามารถแบ่งกรอบการวิเคราะห์ข้อมูลได้ดังนี้

1. การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมระหว่างการนำเทคนิคซีบีอาร์ และเทคนิคไอไออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล การทดสอบสมมติฐานนี้ได้ข้อมูลจากเอกสารบันทึกข้อบกพร่องในขั้นตอนการจัดเตรียม โดยแบ่งการคำนวณออกเป็น 2 กลุ่มทดลอง คือ (1) กลุ่มทดลองที่ใช้เทคนิคไอไออาร์ทีในการตรวจสอบ (2) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ

<u>สมมติฐาน</u>

$$H_0 : \mu_{(EFF1,Pre)} = \mu_{(EFF2,Pre)}$$

$$H_1 : \mu_{(EFF1,Pre)} > \mu_{(EFF2,Pre)}$$

โดยที่ $\mu_{(EFF1,Pre)}$ = ค่าเฉลี่ยของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากการใช้เทคนิคโอโออาร์ที

$\mu_{(EFF2,Pre)}$ = ค่าเฉลี่ยของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากการใช้เทคนิคซีบีอาร์

หลังจากคำนวณประสิทธิภาพในการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมของแต่ละหน่วยทดลองได้แล้ว ผู้วิจัยจะทดสอบว่าเป็นการแจกแจงปกติ (Normal Distribution) หรือไม่ ถ้าหากเป็นจะทดสอบสมมติฐานแบบที (t-test) เนื่องจากการทดสอบสมมติฐานนี้เป็นการทดสอบความแตกต่างระหว่างค่าเฉลี่ยของค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของ 2 ประชากร (กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ และเทคนิคโอโออาร์ที) โดยที่ขนาดตัวอย่างมีขนาดเล็ก และมีค่าความแปรปรวนของประชากรเท่ากัน เนื่องจากกลุ่มทดลอง 2 กลุ่มที่นำมาใช้ถูกสุ่มมาจากประชากรเดียวกัน ดังนั้นผู้วิจัยจึงเลือกใช้การทดสอบสมมติฐานแบบที ที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $n_1 + n_2 - 2$ (กัลยา วาณิชย์บัญชา, 2544) ดังนี้

$$t = \frac{\left(\overline{X}_1 - \overline{X}_2\right)}{S\sqrt{(1/n_1) + (1/n_2)}}$$

โดยที่ $\overline{X}_1$ = ค่าเฉลี่ยของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที

$\overline{X}_2$ = ค่าเฉลี่ยของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์

S = ส่วนเบี่ยงเบนมาตรฐานของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลอง

n_1 = จำนวนหน่วยทดลองที่ใช้เทคนิคโอโออาร์ที

n_2 = จำนวนหน่วยทดลองที่ใช้เทคนิคซีบีอาร์

ถ้าค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมที่ได้จากหน่วยทดลองไม่แจกแจงแบบปกติ จะต้องมีการใช้การทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์ (Nonparametric Test)

2. การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมระหว่างการนำเทคนิคซีบีอาร์ และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล การทดสอบสมมติฐานนี้ได้ข้อมูลจากเอกสารบันทึกข้อบกพร่องในขั้นตอนการจัดเตรียม โดยแบ่งการคำนวณออกเป็น 2 กลุ่มทดลอง คือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที ในการตรวจสอบ (2) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ

<u>สมมติฐาน</u>

$$H_0 : \mu_{(eff1,Pre)} = \mu_{(eff2,Pre)}$$

$$H_1 : \mu_{(eff1,Pre)} > \mu_{(eff2,Pre)}$$

โดยที่ $\mu_{(eff1,Pre)}$ = ค่าเฉลี่ยของประสิทธิผลในขั้นตอนการจัดเตรียมที่ได้จากการใช้เทคนิคโอโออาร์ที

$\mu_{(eff2,Pre)}$ = ค่าเฉลี่ยของประสิทธิผลในขั้นตอนการจัดเตรียมที่ได้จากการใช้เทคนิคซีบีอาร์

หลังจากคำนวณประสิทธิผลการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมของแต่ละหน่วยทดลองได้แล้ว ผู้วิจัยจะทดสอบว่าเป็นการแจกแจงปกติหรือไม่ ถ้าหากเป็นจะทดสอบสมมติฐานแบบที (t-test) เนื่องจากการทดสอบสมมติฐานนี้เป็นการทดสอบความแตกต่างระหว่างค่าเฉลี่ยของค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ของ 2 ประชากร (กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ และเทคนิคโอโออาร์ที) โดยที่ขนาดตัวอย่างมีขนาดเล็ก และมีค่าความแปรปรวนของประชากรเท่ากัน เนื่องจากกลุ่มทดลอง 2 กลุ่มที่นำมาใช้ถูกสุ่มมาจากประชากรเดียวกัน ดังนั้นผู้วิจัยจึงเลือกใช้การทดสอบสมมติฐานแบบที ที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $n_1 + n_2 - 2$ (กัลยา วาณิชย์บัญชา, 2544) ดังนี้

$$t = \frac{\left(\overline{X}_1 - \overline{X}_2\right)}{S\sqrt{(1/n_1)+(1/n_2)}}$$

โดยที่ $\overline{X}_1$ = ค่าเฉลี่ยของประสิทธิผลในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที

$\overline{X}_2$ = ค่าเฉลี่ยของประสิทธิผลในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์

S = ส่วนเบี่ยงเบนมาตรฐานของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลอง

n_1 = จำนวนหน่วยทดลองที่ใช้เทคนิคโอโออาร์ที

n_2 = จำนวนหน่วยทดลองที่ใช้เทคนิคซีบีอาร์

ถ้าหากประสิทธิผลการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมที่ได้จากหน่วยทดลองไม่แจกแจงแบบปกติ จะต้องมีการใช้การทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์

3. การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล การทดสอบสมมติฐานนี้ได้ข้อมูลจากเอกสารบันทึกข้อบกพร่องในขั้นตอนการประชุมตรวจสอบ โดยแบ่งการคำนวณออกเป็น 2 กลุ่มทดลองคือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบ (2) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ

<u>สมมติฐาน</u>

$$H_0 : \mu_{(EFF1,Pre+Meet)} = \mu_{(EFF2,Pre+Meet)}$$

$$H_1 : \mu_{(EFF1,Pre+Meet)} > \mu_{(EFF2,Pre+Meet)}$$

โดยที่ $\mu_{(EFF1,Pre+Meet)}$ = ค่าเฉลี่ยของประสิทธิภาพในขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากการใช้เทคนิคโอโออาร์ที

$\mu_{(EFF2,Pre+Meet)}$ = ค่าเฉลี่ยของประสิทธิภาพในขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากการใช้เทคนิคซีบีอาร์

หลังจากคำนวณประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบของกลุ่มที่ถูกจัดทำขึ้นสำหรับการทำขั้นตอนการประชุมตรวจสอบในแต่ละกลุ่มทดลองได้แล้ว ผู้วิจัยจะทดสอบว่าเป็นการแจกแจงปกติหรือไม่ ถ้าหากเป็นจะทดสอบสมมติฐานแบบที (t-test) เนื่องจากการทดสอบสมมติฐานนี้เป็นการทดสอบความแตกต่างระหว่างค่าเฉลี่ยของค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของ 2 ประชากร (กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ และเทคนิคโอโออาร์ที) โดยที่ขนาดตัวอย่างมีขนาดเล็ก และมีค่าความแปรปรวนของประชากรเท่ากัน เนื่องจากกลุ่มทดลอง 2 กลุ่มที่นำมาใช้ถูกสุ่มมาจากประชากรเดียวกัน ดังนั้นผู้วิจัยจึงเลือกใช้การทดสอบสมมติฐานแบบที ที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $n_1 + n_2 - 2$ (กัลยา วาณิชย์บัญชา, 2544) ดังนี้

$$t = \frac{\left(\overline{X}_1 - \overline{X}_2\right)}{S\sqrt{(1/n_1) + (1/n_2)}}$$

โดยที่ $\overline{X}_1$ = ค่าเฉลี่ยของประสิทธิภาพในขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที

$\overline{X}_2$ = ค่าเฉลี่ยของประสิทธิภาพในขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์

S = ส่วนเบี่ยงเบนมาตรฐานของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลอง

n_1 = จำนวนหน่วยทดลองที่ใช้เทคนิคโอโออาร์ที

n_2 = จำนวนหน่วยทดลองที่ใช้เทคนิคซีบีอาร์

ถ้าหากประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากหน่วยทดลองไม่แจกแจงแบบปกติ จะใช้การทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์

4. การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบของกลุ่มที่ถูกจัดทำขึ้นสำหรับการทำขั้นตอนการประชุมตรวจสอบในแต่ละกลุ่มทดลองได้แล้ว ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล การทดสอบสมมติฐานนี้ได้ข้อมูลจากเอกสารบันทึกข้อบกพร่องในขั้นตอนการประชุมตรวจสอบ โดยแบ่งการคำนวณออกเป็น 2 กลุ่มทดลองคือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบ (2) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ

<u>สมมติฐาน</u>

$$H_0 : \mu_{(eff1,Pre+Meet)} = \mu_{(eff2,Pre+Meet)}$$

$$H_1 : \mu_{(eff1,Pre+Meet)} > \mu_{(eff2,Pre+Meet)}$$

โดยที่ $\mu_{(eff1,Pre+Meet)}$ = ค่าเฉลี่ยของประสิทธิผลในขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากการใช้เทคนิคโอโออาร์ที

$\mu_{(eff2,Pre+Meet)}$ = ค่าเฉลี่ยของประสิทธิผลในขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากการใช้เทคนิคซีบีอาร์

หลังจากคำนวณประสิทธิผลการตรวจสอบซอฟต์แวร์ ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบในแต่ละกลุ่มทดลองได้แล้ว ผู้วิจัยจะทดสอบว่าเป็นการแจกแจงปกติหรือไม่ ถ้าหากเป็นจะทดสอบสมมติฐานแบบที (t-test) เนื่องจากการทดสอบสมมติฐานนี้เป็นการทดสอบความแตกต่างระหว่างค่าเฉลี่ยของค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ของ 2 ประชากร (กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ และเทคนิคโอโออาร์ที) โดยที่ขนาดตัวอย่างมีขนาดเล็ก

และมีค่าความแปรปรวนของประชากรเท่ากัน เนื่องจากกลุ่มทดลอง 2 กลุ่มที่นำมาใช้ถูกสุ่มมาจากประชากรเดียวกัน ดังนั้นผู้วิจัยจึงเลือกใช้การทดสอบสมมติฐานแบบที ที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $n_1 + n_2 - 2$ (กัลยา วาณิชย์บัญชา, 2544) ดังนี้

$$t = \frac{\left(\overline{X}_1 - \overline{X}_2\right)}{S\sqrt{(1/n_1)+(1/n_2)}}$$

โดยที่ $\overline{X}_1$ = ค่าเฉลี่ยของประสิทธิผลในขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที

$\overline{X}_2$ = ค่าเฉลี่ยของประสิทธิผลในขั้นตอนการจัดเตรียม รวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์

S = ส่วนเบี่ยงเบนมาตรฐานของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลอง

n_1 = จำนวนหน่วยทดลองที่ใช้เทคนิคโอโออาร์ที

n_2 = จำนวนหน่วยทดลองที่ใช้เทคนิคซีบีอาร์

ถ้าหากประสิทธิผลการตรวจสอบซอฟต์แวร์ ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากหน่วยทดลองไม่แจกแจงแบบปกติ จะต้องมีการใช้การทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์

5. การเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล การทดสอบสมมติฐานนี้ได้ข้อมูลจากเอกสารบันทึกข้อบกพร่องในขั้นตอนการจัดเตรียม และเอกสารบันทึกข้อบกพร่องในขั้นตอน การประชุมตรวจสอบ โดยแบ่งการคำนวณออกเป็น 2 กลุ่มทดลองคือ (1) กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบ (2) กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ

<u>สมมติฐาน</u>

$H_0 : \mu_{FR1} = \mu_{FR2}$

$H_1 : \mu_{FR1} > \mu_{FR2}$

โดยที่ μ_{FR1} = ค่าเฉลี่ยของประสิทธิผลในการกำจัดผลบวกปลอมที่ได้จากการใช้เทคนิคโอโออาร์ที

μ_{FR2} = ค่าเฉลี่ยของประสิทธิผลในการกำจัดผลบวกปลอมที่ได้จากการใช้เทคนิคซีบีอาร์

หลังจากคำนวณประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบแต่ละหน่วยทดลองได้แล้ว ผู้วิจัยจะทดสอบว่าเป็นการแจกแจงปกติหรือไม่ ถ้าหากเป็นจะทดสอบสมมติฐานแบบที (t-test) เนื่องจากการทดสอบสมมติฐานนี้เป็นการทดสอบความแตกต่างระหว่างค่าเฉลี่ยของค่าประสิทธิผลในการกำจัดผลบวกปลอมของ 2 ประชากร (กลุ่มทดลองที่ใช้เทคนิคซีบีอาร์ และเทคนิคโอโออาร์ที) โดยที่ขนาดตัวอย่างมีขนาดเล็ก และมีค่าความแปรปรวนของประชากรเท่ากัน เนื่องจากกลุ่มทดลอง 2 กลุ่มที่นำมาใช้ถูกสุ่มมาจากประชากรเดียวกัน ดังนั้นผู้วิจัยจึงเลือกใช้การทดสอบสมมติฐานแบบที ที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $n_1 + n_2 - 2$ (กัลยา วาณิชย์บัญชา, 2544) ดังนี้

$$t = \frac{\left(\overline{X}_1 - \overline{X}_2\right)}{S\sqrt{(1/n_1)+(1/n_2)}}$$

โดยที่ $\overline{X}_1$ = ค่าเฉลี่ยของประสิทธิผลในการกำจัดผลบวกปลอมที่ได้จากกลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที

$\overline{X}_2$ = ค่าเฉลี่ยของประสิทธิผลในการกำจัดผลบวกปลอมที่ได้จากกลุ่มทดลองที่ใช้เทคนิคซีบีอาร์

S = ส่วนเบี่ยงเบนมาตรฐานของประสิทธิภาพในขั้นตอนการจัดเตรียมที่ได้จากกลุ่มทดลอง

n_1 = จำนวนหน่วยทดลองที่ใช้เทคนิคโอโออาร์ที

n_2 = จำนวนหน่วยทดลองที่ใช้เทคนิคซีบีอาร์

ถ้าหากประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบที่ได้จากหน่วยทดลองไม่แจกแจงแบบปกติ จะต้องมีการใช้การทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์

บทที่ 4
ผลการวิเคราะห์ข้อมูล

4.1 บทนำ

บทนี้จะกล่าวถึงผลการวิเคราะห์ข้อมูลที่ได้จากการทดลอง เพื่อนำมาตอบวัตถุประสงค์ของงานวิจัยที่กล่าวมาข้างต้น ซึ่งได้แก่การเปรียบเทียบประสิทธิภาพ (Efficiency) ประสิทธิผล (Effective) ในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม และขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ และเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอม (False Positive) ของขั้นตอนการประชุมตรวจสอบ ระหว่างการนำเทคนิคซีบีอาร์และเทคนิคไอโออาร์ทีมาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ ที่ถูกออกแบบโดยการใช้แผนภาพยูเอ็มแอล ซึ่งจะประกอบด้วยการวิเคราะห์ข้อมูลในลักษณะสถิติเชิงพรรณนา (Descriptive statistic) การตรวจสอบการแจกแจงของข้อมูล ผลการทดสอบสมมติฐานในลักษณะของสถิติเชิงอนุมาน (Inferential statistics) และผลการวิเคราะห์ข้อมูลเพิ่มเติม

4.2 ผลการวิเคราะห์ข้อมูล

หลังจากผู้วิจัยได้กำหนดแผนแบบการทดลอง (Experimental Design) และจัดทำเครื่องมือที่ใช้ในการทดลองซึ่งได้แก่เอกสารแสดงการออกแบบซอฟต์แวร์ เอกสารคำแนะนำ เอกสารบันทึกข้อบกพร่อง และเอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอลเรียบร้อยแล้ว ผู้วิจัยได้ดำเนินการทดลองตามแผนแบบการทดลอง กล่าวคือเก็บข้อมูลจากหน่วยทดลองที่เป็นนิสิตปัจจุบันในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการพัฒนาซอฟต์แวร์เชิงวัตถุ แต่เนื่องจากผู้วิจัยได้ใช้หน่วยทดลองที่กำหนดไว้ข้างต้นมาใช้ในการทดสอบเครื่องมือที่ใช้ในการทดลอง (Pretest) แล้ว และมีบางหน่วยทดลองที่ไม่สามารถมาเข้าร่วมการทดลองได้ ดังนั้นผู้วิจัยจึงสุ่มหน่วยทดลองที่มีคุณสมบัติใกล้เคียงกับหน่วยทดลองที่ผู้วิจัยกำหนดไว้ นั่นคือเป็นนิสิตปริญญามหาบัณฑิตจากสาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ วิทยาการคอมพิวเตอร์ วิศวกรรมซอฟต์แวร์ และวิทยาการสารสนเทศที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการออกแบบและพัฒนาซอฟต์แวร์เชิงวัตถุ และจบการศึกษาด้านคอมพิวเตอร์ในระดับปริญญาบัณฑิต โดยมีหน่วยกิตในรายวิชาด้านคอมพิวเตอร์ไม่ต่ำกว่า 35 หน่วยกิต ซึ่งเป็นคุณสมบัติของผู้สมัครเข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย เพื่อให้จำนวนหน่วย

ทดลองครบตามที่ได้กำหนดไว้ในแผนแบบการทดลองนั่นคือ 48 คน โดยแบ่งหน่วยทดลองออกเป็น 2 กลุ่ม คือ

กลุ่มที่ 1 กลุ่มที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบ กล่าวคือตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการใช้สถานการณ์ (Scenario) เป็นเอกสารคำแนะนำสำหรับตรวจสอบ

กลุ่มที่ 2 กลุ่มที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบ กล่าวคือตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการใช้เช็กลิสต์ (Checklist) เป็นเอกสารคำแนะนำสำหรับตรวจสอบ

ในการทดลองนั้นหน่วยทดลองจะตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ เพื่อค้นหาข้อบกพร่องในเอกสารแสดงการออกแบบซอฟต์แวร์ที่ทางผู้วิจัยกำหนดไว้ โดยผู้วิจัยเก็บรวบรวมข้อมูลและนำข้อมูลที่ได้มาวิเคราะห์ เพื่อตอบวัตถุประสงค์ของงานวิจัยที่ต้องการเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ โดยการเปรียบเทียบ (1) ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม (2) ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม (3)ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ (4) ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ (5) ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ซึ่งการวิเคราะห์ผลจะประกอบด้วยการวิเคราะห์ข้อมูลขั้นต้น (Descriptive Statistics) และการวิเคราะห์ข้อมูลเชิงอนุมาน (Inferential Statistics)

4.2.1 การวิเคราะห์ข้อมูลในลักษณะสถิติเชิงพรรณนา (Descriptive Statistics) งานวิจัยนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) เพื่อเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ดังที่กล่าวมาแล้วข้างต้น โดยมีหน่วยทดลองทั้งหมด 48 คนสำหรับขั้นตอนการจัดเตรียม และสำหรับขั้นตอนการประชุมตรวจสอบนั้น จะจับกลุ่มทดลองกลุ่มละ 3 คน นั่นคือมีกลุ่มทดลองทั้งหมด 16 กลุ่ม โดยหน่วยทดลองทั้งหมดถูกแบ่งออกเป็น 2 กลุ่มที่ได้รับตัวแปรอิสระที่ต่างกัน แสดงได้ดังตารางที่ 4-1

ตารางที่ 4-1 ตารางแจกแจงจำนวนหน่วยทดลองในขั้นตอนการจัดเตรียมและขั้นตอนการประชุมตรวจสอบ จำแนกตามเทคนิคการอ่านซอฟต์แวร์

ขั้นตอน	เทคนิคการอ่านซอฟต์แวร์		รวม
	โอโออาร์ที	ซีบีอาร์	
การจัดเตรียม	24 คน	24 คน	48 คน
การประชุมตรวจสอบ	8 กลุ่ม	8 กลุ่ม	16 กลุ่ม

จากที่กล่าวไว้ข้างต้นแล้วว่าหน่วยทดลองประกอบด้วย นิสิตปริญญามหาบัณฑิตที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการออกแบบและพัฒนาซอฟต์แวร์เชิงวัตถุ และจบการศึกษาด้านคอมพิวเตอร์ในระดับปริญญาบัณฑิต โดยสามารถจำแนกหน่วยทดลองตามสาขาวิชาที่ศึกษาในระดับปริญญามหาบัณฑิตได้ ดังตารางที่ 4-2

ตารางที่ 4-2 ตารางแจกแจงจำนวนหน่วยทดลองจำแนกตามสาขาวิชาที่ศึกษาในระดับปริญญามหาบัณฑิต และเทคนิคการอ่านซอฟต์แวร์

สาขาวิชาที่ศึกษาในระดับปริญญามหาบัณฑิต	เทคนิคซีบีอาร์		เทคนิคโอโออาร์ที	
	ความถี่ (คน)	ร้อยละ	ความถี่ (คน)	ร้อยละ
การพัฒนาซอฟต์แวร์ด้านธุรกิจ	13	27.08	15	31.25
เทคโนโลยีสารสนเทศทางธุรกิจ	4	8.33	6	12.50
วิทยาการคอมพิวเตอร์	1	2.08	-	-
วิศวกรรมซอฟต์แวร์	4	8.33	-	-
วิทยาการสารสนเทศ	2	4.17	3	6.25
รวม	24	49.99	24	50.00

จากตารางที่ 4-2 แสดงจำนวนหน่วยทดลองโดยแบ่งตามสาขาวิชาที่ศึกษาในระดับปริญญามหาบัณฑิต ซึ่งสาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจมีจำนวน 28 คน คิดเป็นร้อยละ 58.33 ของหน่วยทดลองทั้งหมด โดยที่สาขาวิชานี้เป็นหน่วยทดลองที่ผู้วิจัยกำหนดไว้ตั้งแต่แรก แต่จากการที่ผู้วิจัยนำหน่วยทดลองไปใช้ในการตรวจสอบเครื่องมือที่ใช้ในการทดลองบ้างแล้ว จึงได้นำนิสิตจากสาขาวิชาเทคโนโลยีสารสนเทศทางธุรกิจ สาขาวิชาวิทยาการคอมพิวเตอร์ สาขาวิชาวิศวกรรมซอฟต์แวร์ และสาขาวิชาวิทยาการสารสนเทศมาเป็นหน่วยทดลองแทน

ในการทดลองนี้ผู้วิจัยกำหนดข้อบกพร่องลงในเอกสารแสดงการออกแบบซอฟต์แวร์จำนวน 20 ข้อ และกำหนดระยะเวลาที่ใช้ในขั้นตอนการจัดเตรียมไม่เกิน 90 นาที ซึ่งผลที่ได้จากการทดลองในขั้นตอนการจัดเตรียมของหน่วยทดลองทั้งหมด 48 คน สามารถแสดงค่าสถิติของจำนวนข้อบกพร่องที่พบในขั้นตอนการจัดเตรียม และระยะเวลาที่ใช้ในการตรวจสอบขั้นตอนการจัดเตรียมของเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ได้ ดังตารางที่ 4-3

ตารางที่ 4-3 ตารางแสดงค่าสถิติจำนวนข้อบกพร่องที่เป็นข้อบกพร่องจริง และระยะเวลาที่ใช้ในการตรวจสอบขั้นตอนการจัดเตรียมของเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด / ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
จำนวนข้อบกพร่องที่เป็นข้อบกพร่องจริง	โอโออาร์ที	48	1 / 10	5.50	2.303
	ซีบีอาร์	48	3 / 15	8.00	2.670
ระยะเวลาที่ใช้	โอโออาร์ที	48	85 / 90	89.5833	1.41165
	ซีบีอาร์	48	65 / 90	81.6250	9.01599

จากตารางที่ 4-3 เป็นการแสดงค่าที่ได้จากหน่วยทดลองในขั้นตอนการจัดเตรียม 2 ค่าคือ (1) ค่าเฉลี่ยของจำนวนข้อบกพร่องที่เป็นข้อบกพร่องจริง ซึ่งเทคนิคซีบีอาร์มีค่าสูงกว่าการใช้เทคนิคโอโออาร์ทีในการตรวจสอบซอฟต์แวร์ โดยที่ค่าส่วนเบี่ยงเบนมาตรฐานของทั้งสองเทคนิคแสดงให้เห็นว่าทั้งสองเทคนิคมีการกระจายของข้อมูลที่ใกล้เคียงกัน (2) ค่าเฉลี่ยของระยะเวลาที่ใช้ซึ่งเทคนิคซีบีอาร์มีค่าต่ำกว่าการใช้เทคนิคโอโออาร์ที ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยที่ค่าส่วนเบี่ยงเบนมาตรฐานแสดงให้เห็นว่า เทคนิคซีบีอาร์มีการกระจายของข้อมูลสูงกว่าเทคนิคโอโออาร์ที

สำหรับขั้นตอนการประชุมตรวจสอบนั้นผู้วิจัยกำหนดระยะเวลาในการตรวจสอบไว้ไม่เกิน 60 นาที ซึ่งผลที่ได้จากการตรวจสอบในขั้นตอนการประชุมตรวจสอบของกลุ่มทดลองทั้งหมด 16 กลุ่ม ผู้วิจัยสามารถแสดงค่าสถิติของจำนวนข้อบกพร่องที่พบในขั้นตอนการประชุมตรวจสอบ ระยะเวลาที่ใช้ในการตรวจสอบขั้นตอนการประชุมตรวจสอบ จำนวนผลบวกปลอมที่กลุ่มทดลองพบในขั้นตอนการจัดเตรียม จำนวนผลบวกปลอมที่พบในขั้นตอนการประชุมตรวจสอบ และจำนวนข้อบกพร่องที่พบเป็นครั้งแรกในขั้นตอนการประชุมตรวจสอบ ของเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีได้ ดังตารางที่ 4-4

ตารางที่ 4-4 ตารางแสดงค่าสถิติจำนวนข้อบกพร่องที่เป็นข้อบกพร่องจริงในขั้นตอนการประชุมตรวจสอบ เวลาที่ใช้ในการตรวจสอบขั้นตอนการประชุมตรวจสอบ จำนวนผลบวกปลอมที่พบในขั้นตอนการจัดเตรียม จำนวนผลบวกปลอมที่พบในขั้นตอนการประชุมตรวจสอบ และจำนวนข้อบกพร่องที่พบเป็นครั้งแรกในขั้นตอนการประชุมตรวจสอบ ของเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที

ตัวแปร	จำนวนหน่วยทดลอง	เทคนิคการอ่านซอฟต์แวร์	ค่าต่ำสุด / ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
จำนวนข้อบกพร่องที่เป็นข้อบกพร่องจริง	16	โอโออาร์ที	5 / 13	7.63	2.504
	16	ซีบีอาร์	9 / 17	11.75	2.435
ระยะเวลาที่ใช้	16	โอโออาร์ที	40 / 60	49.3750	7.28869
	16	ซีบีอาร์	35 / 60	49.5000	8.33238
จำนวนผลบวกปลอม (ขั้นตอนการจัดเตรียม)	16	โอโออาร์ที	4 / 23	12.88	6.728
	16	ซีบีอาร์	8 / 25	15.25	5.497
จำนวนผลบวกปลอม (ขั้นตอนการประชุมตรวจสอบ)	16	โอโออาร์ที	1 / 11	4.88	3.314
	16	ซีบีอาร์	1 / 19	6.63	6.413
จำนวนข้อบกพร่องที่พบเป็นครั้งแรกในขั้นตอนการประชุมตรวจสอบ	16	โอโออาร์ที	0 / 0	0	0
	16	ซีบีอาร์	0 / 2	.50	.756

จากตารางที่ 4-4 เป็นการแสดงค่าที่ได้จากขั้นตอนการประชุมตรวจสอบทั้งหมด 5 ค่าคือ (1) ค่าเฉลี่ยของจำนวนข้อบกพร่องที่เป็นข้อบกพร่องจริงในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ซึ่งเทคนิคซีบีอาร์มีค่าสูงกว่าเทคนิคโอโออาร์ที โดยที่ค่าส่วนเบี่ยงเบนมาตรฐานแสดงให้เห็นว่าทั้งสองเทคนิคมีการกระจายของข้อมูลใกล้เคียงกัน (2) ค่าเฉลี่ยของระยะเวลาที่ใช้ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ของเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมีค่าใกล้เคียงกัน และค่าส่วนเบี่ยงเบนมาตรฐานแสดงให้เห็นว่าทั้งสองเทคนิคมีการกระจายของข้อมูลใกล้เคียงกัน (3) ค่าเฉลี่ยของจำนวนผลบวกปลอมที่กลุ่มทดลองพบในขั้นตอนการจัดเตรียม ของเทคนิคซีบีอาร์สูงกว่าเทคนิคโอโออาร์ที ซึ่งจากค่าส่วนเบี่ยงเบนมาตรฐานแสดงให้เห็นว่าทั้งสองเทคนิคมีการกระจายของข้อมูลใกล้เคียงกัน (4) ค่าเฉลี่ยของจำนวนผลบวกปลอมที่พบในขั้นตอนการประชุมตรวจสอบของเทคนิคซีบีอาร์มีค่าสูงกว่าของเทคนิคโอโออาร์ที และจากค่าส่วนเบี่ยงเบนมาตรฐานแสดงให้เห็นว่าเทคนิคซีบีอาร์มีการกระจายของข้อมูลสูง

กว่าเทคนิคโอโออาร์ที (5) ค่าเฉลี่ยของจำนวนข้อบกพร่องที่พบเป็นครั้งแรกในขั้นตอนการประชุมตรวจสอบของเทคนิคโอโออาร์ทีมีค่าใกล้เคียงกัน กล่าวคือขั้นตอนการประชุมตรวจสอบแทบไม่พบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมเลย

การแจกแจงข้อมูลที่กล่าวมาข้างต้นเป็นข้อมูลดิบที่เก็บได้จากหน่วยทดลอง แต่สำหรับวัตถุประสงค์ของงานวิจัยนี้เป็นการเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ประสิทธิผลในการตรวจสอบซอฟต์แวร์ และประสิทธิผลในการกำจัดผลบวกปลอม ดังนั้นในการตอบวัตถุประสงค์งานวิจัย ผู้วิจัยจะนำข้อมูลที่ได้จากการทดลองมาคำนวณเพื่อตอบวัตถุประสงค์ โดยในขั้นตอนการจัดเตรียมนั้นผู้วิจัยสนใจตัวแปร 2 ตัวคือ (1) ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ซึ่งสามารถวัดค่าตัวแปรนี้ได้จากจำนวนข้อบกพร่องเฉลี่ยที่ผู้ตรวจสอบพบในระยะเวลา 1 นาที (2) ประสิทธิผลในการตรวจสอบซอฟต์แวร์ ซึ่งสามารถวัดค่าตัวแปรนี้ได้จากร้อยละของจำนวนข้อบกพร่องที่ผู้ตรวจสอบพบเทียบกับจำนวนข้อบกพร่องทั้งหมดที่ผู้วิจัยใส่ไว้ โดยผู้วิจัยสามารถแสดงค่าสถิติประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม ของเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ได้ ดังตารางที่ 4-5

ตารางที่ 4-5 ตารางแสดงค่าสถิติประสิทธิภาพและประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม ของเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด / ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
ประสิทธิภาพในการตรวจสอบ	โอโออาร์ที	48	.011 / .111	.06146	.025626
	ซีบีอาร์	48	.033 / .172	.09833	.031516
ประสิทธิผลในการตรวจสอบ	โอโออาร์ที	48	5 / 50	27.5000	11.51558
	ซีบีอาร์	48	15 / 75	40.0000	13.35144

จากตารางที่ 4-5 ซึ่งเป็นการแสดงค่าสถิติของขั้นตอนการจัดเตรียมนั้น แสดงให้เห็นว่า (1) ค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของเทคนิคซีบีอาร์มีค่าสูงกว่าเทคนิคโอโออาร์ที โดยที่จากค่าส่วนเบี่ยงเบนมาตรฐานแสดงให้เห็นว่า ทั้งสองเทคนิคมีการกระจายของข้อมูลใกล้เคียงกัน (2) ค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของเทคนิคซีบีอาร์มีค่าสูงกว่าเทคนิคโอโออาร์ที โดยที่ค่าส่วนเบี่ยงเบนมาตรฐานของเทคนิคซีบีอาร์สูงกว่าเทคนิคโอโออาร์ทีเล็กน้อย กล่าวคือเทคนิคซีบีอาร์มีการกระจายของข้อมูลสูงกว่าเทคนิคโอโออาร์ทีเล็กน้อย

การตอบวัตถุประสงค์ในส่วนของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบนั้น ผู้วิจัยสนใจตัวแปรทั้งหมด 3 ตัวคือ (1)ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ซึ่งสามารถคำนวณได้จากจำนวนข้อบกพร่องเฉลี่ยที่คณะผู้ตรวจสอบพบในระยะเวลา 1 นาที (2) ประสิทธิผลในการตรวจสอบซอฟต์แวร์ ซึ่งสามารถคำนวณได้จากร้อยละของจำนวนข้อบกพร่องที่คณะผู้ตรวจสอบพบเทียบกับจำนวนข้อบกพร่องทั้งหมดที่ผู้วิจัยใส่ไว้ และ (3) ประสิทธิผลในการกำจัดผลบวกปลอม ซึ่งสามารถคำนวณได้จากร้อยละของผลบวกปลอมที่ถูกกำจัดในขั้นตอนการประชุมตรวจสอบจากจำนวนผลบวกปลอมทั้งหมดที่พบในขั้นตอนการจัดเตรียม โดยผู้วิจัยสามารถแสดงค่าสถิติของประสิทธิภาพ ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ และประสิทธิผลในการกำจัดผลบวกปลอม ของเทคนิคโอโอเอาร์ทีและเทคนิคซีบีอาร์ได้ ดังตารางที่ 4-6

ตารางที่ 4-6 ตารางแสดงค่าสถิติของประสิทธิภาพ ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ และประสิทธิผลในการกำจัดผลบวกปลอมในขั้นตอนการประชุมตรวจสอบ ของเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด / ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
ประสิทธิภาพในการตรวจสอบ	โอโออาร์ที	16	.100 / .236	.15488	.043721
	ซีบีอาร์	16	.150 / .362	.24438	.067180
ประสิทธิผลในการตรวจสอบ	โอโออาร์ที	16	25 / 65	38.1250	12.51784
	ซีบีอาร์	16	45 / 85	58.7500	12.17433
ประสิทธิผลในการกำจัดผลบวกปลอม	โอโออาร์ที	16	22.22 / 89.47	60.4538	21.38121
	ซีบีอาร์	16	24.00 – 91.67	62.6963	23.07523

จากตารางที่ 4-6 ซึ่งเป็นการแสดงค่าสถิติจากขั้นตอนการประชุมตรวจสอบ แสดงให้เห็นว่า (1) ค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ที่ได้จากเทคนิคซีบีอาร์มีค่าสูงกว่าเทคนิคโอโออาร์ที โดยที่เทคนิคซีบีอาร์มีค่าส่วนเบี่ยงเบนมาตรฐานสูงกว่าเทคนิคโอโออาร์ทีเล็กน้อย แสดงให้เห็นว่าเทคนิคซีบีอาร์มีการกระจายของข้อมูลสูงกว่าโอโออาร์ที (2) ค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ที่ได้จากเทคนิคซีบีอาร์มีค่าสูงกว่าเทคนิคโอโออาร์ที โดยที่ทั้งสองเทคนิคมีค่าส่วนเบี่ยงเบนมาตรฐานใกล้เคียงกัน กล่าวคือมีการ

กระจายของข้อมูลใกล้เคียงกัน และ (3) ค่าเฉลี่ยของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ที่ได้จากเทคนิคซีบีอาร์มีค่าสูงกว่าเทคนิคไอไออาร์ทีเล็กน้อย และเทคนิคซีบีอาร์มีค่าส่วนเบี่ยงเบนมาตรฐานสูงกว่าเทคนิคไอไออาร์ทีเล็กน้อย แสดงให้เห็นว่าเทคนิคซีบีอาร์มีการกระจายของข้อมูลสูงกว่าเทคนิคไอไออาร์ที

4.2.2 การตรวจสอบการแจกแจงของข้อมูล ผู้วิจัยตรวจสอบการแจกแจกของข้อมูลว่าเป็นการแจกแจงปกติหรือไม่ ถ้าข้อมูลมีการการแจกแจงแบบปกติผู้วิจัยจะทดสอบสมมติฐานแบบอิงพารามิเตอร์ (Parametric Test) แต่ถ้าข้อมูลมีการแจกแจงแบบไม่ปกติ ผู้วิจัยจะทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์ (Non Parametric Test) (กัลยา วาณิชย์บัญชา, 2544) โดยงานวิจัยนี้ผู้วิจัยสนใจตัวแปร 5 ตัวคือ (1) ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม (2) ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม (3) ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ (4) ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ (5) ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ

ผู้วิจัยต้องตรวจสอบการแจกแจงข้อมูลทั้ง 5 ข้อนี้ว่ามีการแจกแจงแบบปกติหรือไม่ โดยการกำหนดสมมติฐานของการทดสอบได้ดังนี้

1. H_0: ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม มีการแจกแจงแบบปกติ

   H_1: ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม ไม่ได้มีการแจกแจงแบบปกติ

2. H_0: ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม มีการแจกแจงแบบปกติ

   H_1: ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม ไม่ได้มีการแจกแจงแบบปกติ

3. H_0: ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ มีการแจกแจงแบบปกติ

   H_1: ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ไม่ได้มีการแจกแจงแบบปกติ

4. H_0: ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ มีการแจกแจงแบบปกติ

   H_1: ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ไม่ได้มีการแจกแจงแบบปกติ

5. H_0: ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ มีการแจกแจงแบบปกติ

   H_1: ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ไม่ได้มีการแจกแจงแบบปกติ

ในการตรวจสอบการแจกแจงของข้อมูลเชิงปริมาณโดยใช้สถิติทดสอบนั้น มีสถิติทดสอบที่ใช้คือ Kolmogorov-Smirnov สำหรับหน่วยตัวอย่างมากกว่า 50 หน่วย และ Shapiro-Wilk สำหรับหน่วยตัวอย่างน้อยกว่า 50 หน่วย (กัลยา วานิชย์บัญชา, 2546) สำหรับงานวิจัยนี้ตัวอย่างในแต่ละกลุ่มน้อยกว่า 50 หน่วย ดังนั้นงานวิจัยนี้จึงใช้เทคนิค Shapiro-Wilk ในการตรวจสอบการแจกแจงข้อมูล โดยจะปฏิเสธ H_0 ถ้าค่า Sig. (Significance) ของการทดสอบน้อยกว่าระดับนัยสำคัญที่กำหนด โดยงานวิจัยนี้กำหนดระดับนัยสำคัญเท่ากับ 0.05

ตารางที่ 4-7 ตารางแสดงค่าสถิติทดสอบการแจกแจงปกติ (Normality Test) ของทั้ง 5 ตัวแปร

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	Shapiro-Wilk		
		Statistic	df	Sig.
ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม	โอโออาร์ที	.963	24	.506
	ซีบีอาร์	.921	24	.062
ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม	โอโออาร์ที	.961	24	.468
	ซีบีอาร์	.936	24	.131
ประสิทธิภาพของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ	โอโออาร์ที	.956	8	.768
	ซีบีอาร์	.954	8	.751
ประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ	โอโออาร์ที	.859	8	.118
	ซีบีอาร์	.857	8	.113
ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ	โอโออาร์ที	.950	8	.710
	ซีบีอาร์	.913	8	.376

จากตารางที่ 4-7 พบว่าค่า Sig. ของตัวแปรทุกตัวมีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนดไว้ข้างต้น ดังนั้นจึงยอมรับ H_0 ของตัวแปรทุกตัว กล่าวคือทุกตัวแปรมีการแจกแจงข้อมูลแบบปกติ ดังนั้นผู้วิจัยจึงใช้วิธีการทดสอบสมมติฐานแบบอิงพารามิเตอร์ (Parametric Test) โดยเลือกใช้การทดสอบสมมติฐานแบบที (t-test) เนื่องจากการทดสอบสมมติฐานนี้เป็นการทดสอบความแตกต่างระหว่างค่าเฉลี่ยของ 2 ประชากร (กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ที และเทคนิคซีบีอาร์) โดยที่ขนาดตัวอย่างมีขนาดเล็ก (กัลยา วาณิชย์บัญชา, 2544)

4.3 การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน

ผู้วิจัยต้องการเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน คือเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที ซึ่งจากการตรวจสอบการแจกแจงของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมนั้นเป็นแบบปกติ ดังนั้นผู้วิจัยจึงเลือกการทดสอบสมมติฐานแบบที (t-test) โดยใช้โปรแกรม SPSS

ก่อนที่จะทดสอบสมมติฐานจะต้องตรวจสอบก่อนว่าค่าความแปรปรวนของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีค่าเท่ากันหรือไม่ โดยการตั้งสมมติฐานดังนี้

$$H_0 : \sigma^2_{OORT} = \sigma^2_{CBR}$$
$$H_1 : \sigma^2_{OORT} \neq \sigma^2_{CBR}$$

ในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบเอฟ (F-test) โดยดูได้จากตารางที่ 4-8 ซึ่งเป็นตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances)

ตารางที่ 4-8 ตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม

	Levene's Test for Equality of Variances	
	F	Sig.
ค่าประสิทธิภาพ	.045	.833

เนื่องจากในที่นี้เป็นการทดสอบสองด้าน จึงเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญถ้าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2546) ซึ่งในที่นี้กำหนดค่า

ระดับนัยสำคัญคือ 0.05 และค่า Sig. คือ 0.833 จะเห็นได้ว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงยอมรับ H_0 กล่าวคือค่าประสิทธิภาพของเทคนิคโอโออาร์ที และเทคนิคซีบีอาร์มีความแปรปรวนเท่ากัน

การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมระหว่างการใช้เทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีการกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{OORT} = \mu_{CBR}$$

$$H_1 : \mu_{OORT} > \mu_{CBR}$$

โดยในที่นี้ μ_{OORT} หมายถึงค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมที่ได้จากการใช้เทคนิคโอโออาร์ที และ μ_{CBR} หมายถึงค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมที่ได้จากการใช้เทคนิคซีบีอาร์ เนื่องจากผู้วิจัยคาดว่าเทคนิคโอโออาร์ทีซึ่งเป็นเทคนิคที่จัดทำขึ้นสำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการใช้แผนภาพยูเอ็มแอลโดยเฉพาะ น่าจะมีประสิทธิภาพสูงกว่าเทคนิคซีบีอาร์

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน (จากผลการทดสอบสมมติฐานก่อนหน้า) ซึ่งสามารถแสดงตารางค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมได้ ดังตารางที่ 4-9

ตารางที่ 4-9 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิภาพ	-4.447	46	.000	-.036875	.008291	-.053565	-.020185

จากสมมติฐานข้างต้น จะปฏิเสธสมมติฐาน H_0 เมื่อค่า $\frac{\text{Sig. (2 - tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่า $t > 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2546)

ซึ่งผลจากตารางที่ 4-9 แสดงให้เห็นว่าไม่สามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าน้อยกว่า 0.05 แต่ค่า t มีค่าน้อยกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ

4.4 การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน

ผู้วิจัยต้องการเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน คือเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ซึ่งจากการตรวจสอบการแจกแจงของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมนั้นเป็นแบบปกติ ดังนั้นผู้วิจัยจึงเลือกการทดสอบสมมติฐานแบบที (t-test) โดยใช้โปรแกรม SPSS

ก่อนที่จะทดสอบสมมติฐานจะต้องตรวจสอบก่อนว่าค่าความแปรปรวนของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีเท่ากันหรือไม่ โดยการตั้งสมมติฐานดังนี้

$$H_0 : \sigma^2_{OORT} = \sigma^2_{CBR}$$
$$H_1 : \sigma^2_{OORT} \neq \sigma^2_{CBR}$$

ในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบเอฟ (F-test) โดยดูได้จากตารางที่ 4-10ซึ่งเป็นตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances)

ตารางที่ 4-10 ตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม

	Levene's Test for Equality of Variances	
	F	Sig.
ค่าประสิทธิผล	.212	.647

จากตารางที่ 4-10 แสดงค่า Sig. ซึ่งเท่ากับ 0.647 ซึ่งจะเห็นได้ว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนดไว้เท่ากับ 0.05 ดังนั้นจึงยอมรับ H_0 กล่าวคือค่าประสิทธิผลของเทคนิคโอโออาร์ที และเทคนิคซีบีอาร์มีความแปรปรวนเท่ากัน

การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างการใช้เทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีการกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{OORT} = \mu_{CBR}$$

$$H_1 : \mu_{OORT} > \mu_{CBR}$$

โดยในที่นี้ μ_{OORT} หมายถึงค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมที่ได้จากการใช้เทคนิคโอโออาร์ที และ μ_{CBR} หมายถึงค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมที่ได้จากการใช้เทคนิคซีบีอาร์ เนื่องจากผู้วิจัยคาดว่าเทคนิคโอโออาร์ทีซึ่งเป็นเทคนิคที่จัดทำขึ้นสำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการใช้แผนภาพยูเอ็มแอลโดยเฉพาะ น่าจะมีประสิทธิผลสูงกว่าเทคนิคซีบีอาร์

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน (จากผลการทดสอบสมมติฐานก่อนหน้า) ซึ่งสามารถแสดงตารางค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมได้ ดังตารางที่ 4-11

ตารางที่ 4-11 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิผล	-3.473	46	.001	-12.50000	3.59901	-19.74444	-5.25556

ซึ่งผลจากตารางที่ 4-11 แสดงให้เห็นว่าไม่สามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าน้อยกว่า 0.05 แต่ค่า t มีค่าน้อยกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ

4.5 การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน

ผู้วิจัยต้องการเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน คือเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที ซึ่งจากการตรวจสอบการแจกแจงของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบนั้นเป็นแบบปกติ ดังนั้นผู้วิจัยจึงเลือกการทดสอบสมมติฐานแบบที (t-test) โดยใช้โปรแกรม SPSS

ก่อนที่จะทดสอบสมมติฐานจะต้องตรวจสอบก่อนว่าค่าความแปรปรวนของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีเท่ากันหรือไม่ โดยการตั้งสมมติฐานดังนี้

$$H_0 : \sigma^2_{OORT} = \sigma^2_{CBR}$$
$$H_1 : \sigma^2_{OORT} \neq \sigma^2_{CBR}$$

ในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบเอฟ (F-test) โดยดูได้จากตารางที่ 4-12 ซึ่งเป็นตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances)

ตารางที่ 4-12 ตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ

	Levene's Test for Equality of Variances	
	F	Sig.
ค่าประสิทธิภาพ	1.228	.286

จากตารางที่ 4-12 แสดงค่า Sig. ซึ่งเท่ากับ 0.286 ซึ่งจะเห็นได้ว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนดไว้เท่ากับ 0.05 ดังนั้นจึงยอมรับ H_0 กล่าวคือค่าประสิทธิภาพของเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีความแปรปรวนเท่ากัน

การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีการกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{OORT} = \mu_{CBR}$$
$$H_1 : \mu_{OORT} > \mu_{CBR}$$

โดยในที่นี้ μ_{OORT} หมายถึงค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากการใช้เทคนิคโอโออาร์ที และ μ_{CBR} หมายถึงค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากการใช้เทคนิคซีบีอาร์ เนื่องจากผู้วิจัยคาดว่าเทคนิคโอโออาร์ทีซึ่งเป็นเทคนิคที่จัดทำขึ้นสำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการใช้แผนภาพยูเอ็มแอลโดยเฉพาะ น่าจะมีประสิทธิภาพสูงกว่าเทคนิคซีบีอาร์

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน (จากผลการทดสอบสมมติฐานก่อนหน้า) ซึ่งสามารถแสดงตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบได้ ดังตารางที่ 4-13

ตารางที่ 4-13 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิภาพ	-3.158	14	.007	-.089500	.028339	-.150281	-.028719

ซึ่งผลจากตารางที่ 4-13 แสดงให้เห็นว่าไม่สามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าน้อยกว่า 0.05 แต่ค่า t มีค่าน้อยกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ

4.6 การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน

ผู้วิจัยต้องการเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน คือเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที ซึ่งจากการตรวจสอบการแจกแจงของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบนั้นเป็นแบบปกติ ดังนั้นผู้วิจัยจึงเลือกการทดสอบสมมติฐานแบบที (t-test) โดยใช้โปรแกรม SPSS

ก่อนที่จะทดสอบสมมติฐานจะต้องตรวจสอบก่อนว่าค่าความแปรปรวนของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีเท่ากันหรือไม่ โดยการตั้งสมมติฐานดังนี้

$$H_0 : \sigma^2_{OORT} = \sigma^2_{CBR}$$
$$H_1 : \sigma^2_{OORT} \neq \sigma^2_{CBR}$$

ในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบเอฟ (F-test) โดยดูได้จากตารางที่ 4-14 ซึ่งเป็นตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances)

ตารางที่ 4-14 ตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ

	Levene's Test for Equality of Variances	
	F	**Sig.**
ค่าประสิทธิผล	.013	.910

จากตารางที่ 4-14 แสดงค่า Sig. ซึ่งเท่ากับ 0.910 ซึ่งจะเห็นได้ว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนดไว้เท่ากับ 0.05 ดังนั้นจึงยอมรับ H_0 กล่าวคือค่าประสิทธิผลของเทคนิคโอโออาร์ที และเทคนิคซีบีอาร์มีความแปรปรวนเท่ากัน

การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีการกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{OORT} = \mu_{CBR}$$
$$H_1 : \mu_{OORT} > \mu_{CBR}$$

โดยในที่นี้ μ_{OORT} หมายถึงค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากการใช้เทคนิคโอโออาร์ที และ μ_{CBR} หมายถึงค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากการใช้เทคนิคซีบีอาร์ เนื่องจากผู้วิจัยคาดว่าเทคนิคโอโออาร์ทีซึ่งเป็นเทคนิคที่จัดทำขึ้นสำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการใช้แผนภาพยูเอ็มแอลโดยเฉพาะ น่าจะมีประสิทธิผลสูงกว่าเทคนิคซีบีอาร์

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน (จากผลการทดสอบสมมติฐานก่อนหน้า) ซึ่งสามารถแสดงตารางแสดง

ค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบได้ ดังตารางที่ 4-15

ตารางที่ 4-15 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิผล	-3.341	14	.005	-20.62500	6.17364	-33.86614	-7.38386

ซึ่งผลจากตารางที่ 4-15 แสดงให้เห็นว่าไม่สามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าน้อยกว่า 0.05 แต่ค่า t มีค่าน้อยกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ

4.7 การเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน

ผู้วิจัยต้องการเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน คือเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที ซึ่งจากการตรวจสอบการแจกแจงของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบนั้นเป็นแบบปกติ ดังนั้นผู้วิจัยจึงเลือกการทดสอบสมมติฐานแบบที (t-test) โดยใช้โปรแกรม SPSS

ก่อนที่จะทดสอบสมมติฐานจะต้องตรวจสอบก่อนว่าค่าความแปรปรวนของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีเท่ากันหรือไม่ โดยการตั้งสมมติฐานดังนี้

$$H_0 : \sigma^2_{OORT} = \sigma^2_{CBR}$$
$$H_1 : \sigma^2_{OORT} \neq \sigma^2_{CBR}$$

ในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบเอฟ (F-test) โดยดูได้จากตารางที่ 4-16 ซึ่งเป็นตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances)

ตารางที่ 4-16 ตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances) ของประสิทธิผลในกำจัดผลบวกปลอมของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ

	Levene's Test for Equality of Variances	
	F	Sig.
ค่าประสิทธิผล ในการกำจัดผลบวกปลอม	.002	.961

จากตารางที่ 4-16 แสดงค่า Sig. ซึ่งเท่ากับ 0.961 ซึ่งจะเห็นได้ว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนดไว้เท่ากับ 0.05 ดังนั้นจึงยอมรับ H_0 กล่าวคือค่าประสิทธิผลในการกำจัดผลบวกปลอมของเทคนิคโอโออาร์ที และเทคนิคซีบีอาร์มีความแปรปรวนเท่ากัน

จากตารางที่ 4-4 แสดงให้เห็นว่าขั้นตอนการประชุมตรวจสอบช่วยในการกำจัดผลบวกปลอมได้ ดังนั้นผู้วิจัยจึงกำหนดสมมติฐานเพื่อทดสอบว่าขั้นตอนการประชุมตรวจสอบนั้น ช่วยในการกำจัดผลบวกปลอมอย่างมีนัยสำคัญหรือไม่ ดังต่อไปนี้

1)

$$H_0 : \mu_{FR\_OORT} = 0$$
$$H_1 : \mu_{FR\_OORT} \neq 0$$

2)

$$H_0 : \mu_{FR\_CBR} = 0$$
$$H_1 : \mu_{FR\_CBR} \neq 0$$

การทดสอบสมมติฐานนี้เป็นการทดสอบค่าเฉลี่ยสำหรับ 1 กลุ่มตัวอย่าง โดยในที่นี้ μ_{FR_OORT} หมายถึงค่าเฉลี่ยประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ที่ได้จากการใช้เทคนิคโอโออาร์ทีในการตรวจสอบ และ μ_{FR_CBR} หมายถึงค่าเฉลี่ยประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ที่ได้จากการใช้เทคนิคซีบีอาร์ในการตรวจสอบ ซึ่งผลการทดสอบโดยการใช้สถิติทดสอบที (t-test) แสดงได้ดังตารางที่ 4-17

ตารางที่ 4-17 ตารางแสดงค่าสถิติทดสอบค่าเฉลี่ยของประสิทธิผลในการกำจัดผลบวกปลอม ของขั้นตอนการประชุมตรวจสอบ ที่ได้จากการใช้เทคนิคโอโออาร์ที

	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
ประสิทธิผลในการกำจัดผลบวกปลอมของเทคนิคโอโออาร์ที	7.997	7	.000	60.45375	42.5786	78.3289
ประสิทธิผลในการกำจัดผลบวกปลอมของเทคนิคซีบีอาร์	7.685	7	.000	62.69625	43.4049	81.9876

จากตารางที่ 4-17 ค่า Sig. (2-tailed) ของเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีค่าน้อยกว่าค่าระดับนัยสำคัญที่กำหนดไว้ 0.05 ดังนั้นผู้วิจัยจึงปฏิเสธสมมติฐาน H_0 ของทั้ง 2 เทคนิค กล่าวคือ ขั้นตอนการประชุมตรวจสอบของเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ช่วยในการกำจัดผลบวกปลอมได้

การเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบระหว่างการใช้เทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีการกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{OORT} = \mu_{CBR}$$

$$H_1 : \mu_{OORT} > \mu_{CBR}$$

โดยในที่นี้ μ_{OORT} หมายถึงค่าเฉลี่ยของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ที่ได้จากการใช้เทคนิคโอโออาร์ที และ μ_{CBR} หมายถึงค่าเฉลี่ยของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ที่ได้จากการใช้เทคนิคซีบีอาร์ เนื่องจากผู้วิจัยคาดว่าเทคนิคโอโออาร์ทีซึ่งเป็นเทคนิคที่จัดทำขึ้นสำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ โดยการใช้แผนภาพยูเอ็มแอลโดยเฉพาะ น่าจะมีประสิทธิผลในการกำจัดผลบวกปลอมสูงกว่าเทคนิคซีบีอาร์

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน (จากผลการทดสอบสมมติฐานก่อนหน้า) ซึ่งสามารถแสดงตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบได้ ดังตารางที่ 4-18

ตารางที่ 4-18 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิผลในการกำจัดผลบวกปลอม	-.202	14	.843	-2.24250	11.12218	-26.09720	21.61220

ซึ่งผลจากตารางที่ 4-18 แสดงให้เห็นว่าไม่สามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่ามากกว่าค่าระดับนัยสำคัญ และค่า t มีค่าน้อยกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่า เทคนิคโอโออาร์ที่ไม่ได้มีค่าเฉลี่ยประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ

4.8 การวิเคราะห์ข้อมูลเพิ่มเติม (Exploration)

หลังจากผู้วิจัยทดสอบสมมติฐานเรียบร้อยแล้ว จะเห็นว่าผลที่ได้ไม่ตรงตามที่ผู้วิจัยคาดไว้ กล่าวคือประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ และประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ที่ได้จากเทคนิคเทคนิคโอโออาร์ที่ไม่ได้มีค่าสูงกว่าเทคนิคซีบีอาร์ และจากตารางที่ 4-5 และตารางที่ 4-6 จะเห็นได้ว่าค่าเฉลี่ยประสิทธิภาพและประสิทธิผลในการตรวจสอบ และค่าประสิทธิผลในการกำจัดผลบวกปลอมที่ได้จากเทคนิคซีบีอาร์มีค่าสูงกว่าเทคนิคโอโออาร์ที ดังนั้นผู้วิจัยจึงต้องการทดสอบสมมติฐานเพิ่มเติมว่าเทคนิคซีบีอาร์ มีค่าประสิทธิผลและประสิทธิภาพในการตรวจสอบสูงกว่าเทคนิคโอโออาร์ที่อย่างมีนัยสำคัญหรือไม่ นอกจากนี้ผู้วิจัยยังสนใจที่จะวิเคราะห์ลงไปใน (1) ข้อบกพร่องแต่ละข้อ (2) ประเภทของแผนภาพ (3) ประเภทของข้อบกพร่องที่ได้จากทั้งสองเทคนิคมีค่าแตกต่างกันอย่างไร

4.8.1 การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน จากตารางที่ 4-5 แสดงให้เห็นว่าค่าเฉลี่ยประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ที่ได้จากเทคนิคโอโออาร์ทีมีค่า

น้อยกว่าค่าเฉลี่ยประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมที่ได้จากเทคนิคซีบีอาร์ จึงกำหนดสมมติฐานดังนี้

$$H_0 : \mu_{CBR} \leq \mu_{OORT}$$

$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน ซึ่งสามารถแสดงตารางค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมได้ ดังตารางที่ 4-19

ตารางที่ 4-19 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิภาพ	4.447	46	.000	.036875	.008291	.020185	.053565

ผลจากตารางที่ 4-19 แสดงให้เห็นว่าสามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าน้อยกว่า 0.05 และค่า t มีค่ามากกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่าเทคนิคซีบีอาร์มีค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมสูงกว่าเทคนิคไอโออาร์ที ที่ระดับนัยสำคัญ 0.05

4.8.2 การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน จากตารางที่ 4-5 แสดงให้เห็นว่าค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ที่ได้จากเทคนิคไอโออาร์ทีมีค่าน้อยกว่าค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมที่ได้จากเทคนิคเทคนิคซีบีอาร์ จึงกำหนดสมมติฐานดังนี้

$$H_0 : \mu_{CBR} \leq \mu_{OORT}$$

$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน ซึ่งสามารถแสดงตารางค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมได้ ดังตารางที่ 4-20

ตารางที่ 4-20 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิผล	3.473	46	.001	12.50000	3.59901	5.25556	19.74444

ผลจากตารางที่ 4-20 แสดงให้เห็นว่าสามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าน้อยกว่า 0.05 และค่า t มีค่ามากกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่าเทคนิคซีบีอาร์มีค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมสูงกว่าเทคนิคโอโออาร์ที ที่ระดับนัยสำคัญ 0.05

4.8.3 การเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน จากตารางที่ 4-6 แสดงให้เห็นว่าค่าเฉลี่ยประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ที่ได้จากเทคนิคโอโออาร์ทีมีค่าน้อยกว่าค่าเฉลี่ยประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากเทคนิคเทคนิคซีบีอาร์ จึงกำหนดสมมติฐานดังนี้

$$H_0 : \mu_{CBR} \leq \mu_{OORT}$$

$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน ซึ่งสามารถแสดงตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบได้ ดังตารางที่ 4-21

ตารางที่ 4-21 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิภาพ	3.158	14	.007	.089500	.028339	.028719	.150281

ผลจากตารางที่ 4-21 แสดงให้เห็นว่าสามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าน้อยกว่า 0.05 และค่า t มีค่ามากกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่าเทคนิคซีบีอาร์มีค่าเฉลี่ยของประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคโอโออาร์ที ที่ระดับนัยสำคัญ 0.05

4.8.4 การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน จากตารางที่ 4-6 แสดงให้เห็นว่าค่าเฉลี่ยประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ที่ได้จากเทคนิคโอโออาร์ทีมีค่าน้อยกว่าค่าเฉลี่ยประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบที่ได้จากเทคนิคเทคนิคซีบีอาร์ จึงกำหนดสมมติฐานดังนี้

$$H_0 : \mu_{CBR} \leq \mu_{OORT}$$

$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน ซึ่งสามารถแสดงตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบได้ ดังตารางที่ 4-22

ตารางที่ 4-22 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิผล	3.341	14	.005	20.62500	6.17364	7.38386	33.86614

ผลจากตารางที่ 4-22 แสดงให้เห็นว่าสามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าน้อยกว่า 0.05 และค่า t มีค่ามากกว่า 0 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่าเทคนิคซีบีอาร์มีค่าเฉลี่ยของประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

4.8.5 การเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ระหว่างการใช้เทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกัน จากตารางที่ 4-6 แสดงให้เห็นว่าค่าเฉลี่ยประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ที่ได้จากเทคนิคโอโออาร์ทีมีค่าน้อยกว่าค่าเฉลี่ยประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบที่ได้จากเทคนิคเทคนิคซีบีอาร์จึงกำหนดสมมติฐานดังนี้

$$H_0 : \mu_{CBR} \leq \mu_{OORT}$$

$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน ซึ่งสามารถแสดงตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบได้ ดังตารางที่ 4-23

ตารางที่ 4-23 ตารางแสดงค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ

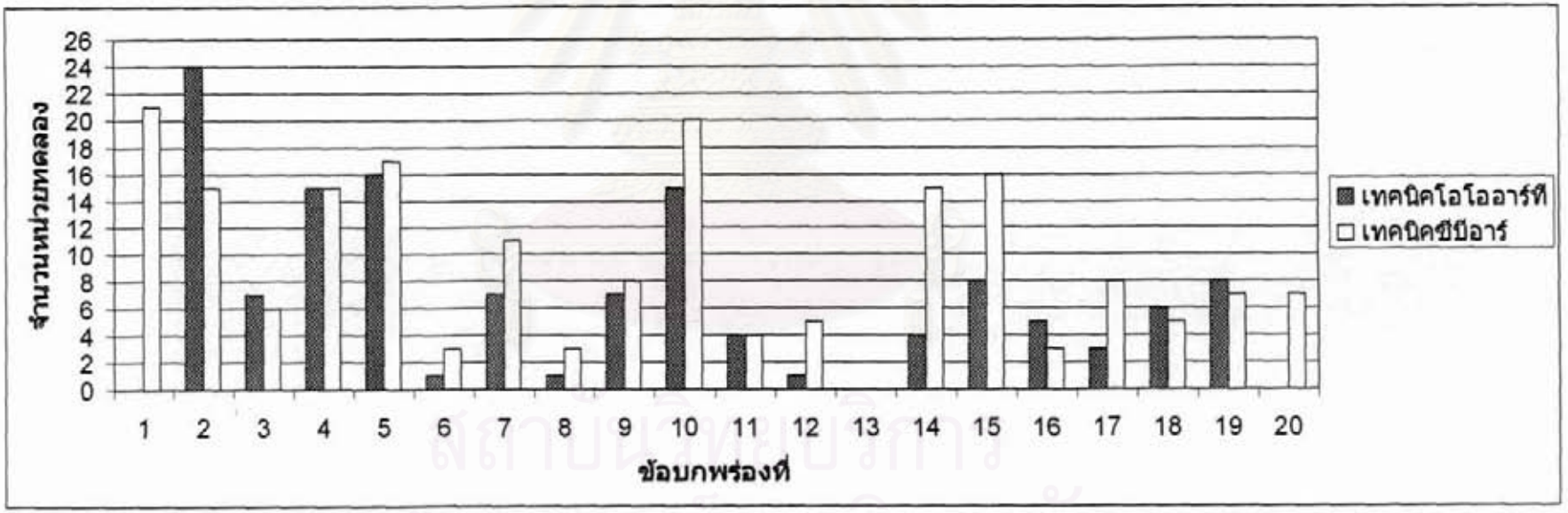
	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิผลในการกำจัดผลบวกปลอม	.202	14	.843	2.24250	11.12218	-21.61220	26.09720

ผลจากตารางที่ 4-23 แสดงให้เห็นว่าไม่สามารถปฏิเสธ H_0 ได้ เนื่องจากถึงแม้ค่า t ซึ่งมีค่าเท่ากับ 0.202 จะมีค่ามากกว่า 0 แต่ค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่าเท่ากับ 0.422 ซึ่งมีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนดไว้ 0.05 ดังนั้นจึงยอมรับสมมติฐาน H_0 กล่าวคือค่าเฉลี่ยประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ของทั้งสองเทคนิคมีค่าไม่แตกต่างกัน ซึ่งจากตารางที่ 4-6 ก็แสดงให้เห็นว่าค่าเฉลี่ยประสิทธิผลในการกำจัดผลบวกปลอมของเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมีค่าใกล้เคียงกันมาก

4.8.6 การเปรียบเทียบความถี่ของหน่วยทดลองที่ตรวจพบข้อบกพร่องแต่ละข้อ ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ผู้วิจัยต้องการวิเคราะห์ลงไปในแต่ละข้อบกพร่องว่า จำนวนหน่วยทดลองของทั้งสองเทคนิคที่สามารถตรวจพบข้อบกพร่องแต่ละข้อแตกต่างกันอย่างไร ซึ่งสามารถแสดงได้ดังตารางที่ 4-24 และสามารถแสดงในรูปของแผนภูมิแท่งได้ดังรูปที่ 4-1

ตารางที่ 4-24 ตารางแสดงความถี่ของหน่วยทดลองที่ตรวจพบข้อบกพร่องแต่ละข้อ จำแนกตามเทคนิคการอ่านซอฟต์แวร์

ข้อบกพร่องที่ / จำนวนหน่วยทดลอง	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
เทคนิคโอโออาร์ที	0	24	7	15	16	1	7	1	7	15	4	1	0	4	8	5	3	6	8	0
เทคนิคซีบีอาร์	21	15	6	15	17	3	11	3	8	20	4	5	0	15	16	3	8	5	7	7
รวม	21	39	13	30	33	4	18	4	15	35	8	6	0	19	24	8	11	11	15	7



รูปที่ 4-1 แสดงแผนภูมิเปรียบเทียบความถี่ของหน่วยทดลองที่ตรวจพบข้อบกพร่องแต่ละข้อ ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์

จากรูปที่ 4-1 แสดงให้เห็นว่าจำนวนของหน่วยทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจพบข้อบกพร่องแต่ละข้อ จะมีค่ามากกว่าจำนวนของหน่วยทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบ โดยมีข้อบกพร่องเพียง 5 ข้อจากทั้งหมด 20 ข้อที่หน่วยทดลองที่ใช้เทคนิคโอโออาร์ทีสามารถตรวจพบได้มากกว่าเทคนิคซีบีอาร์

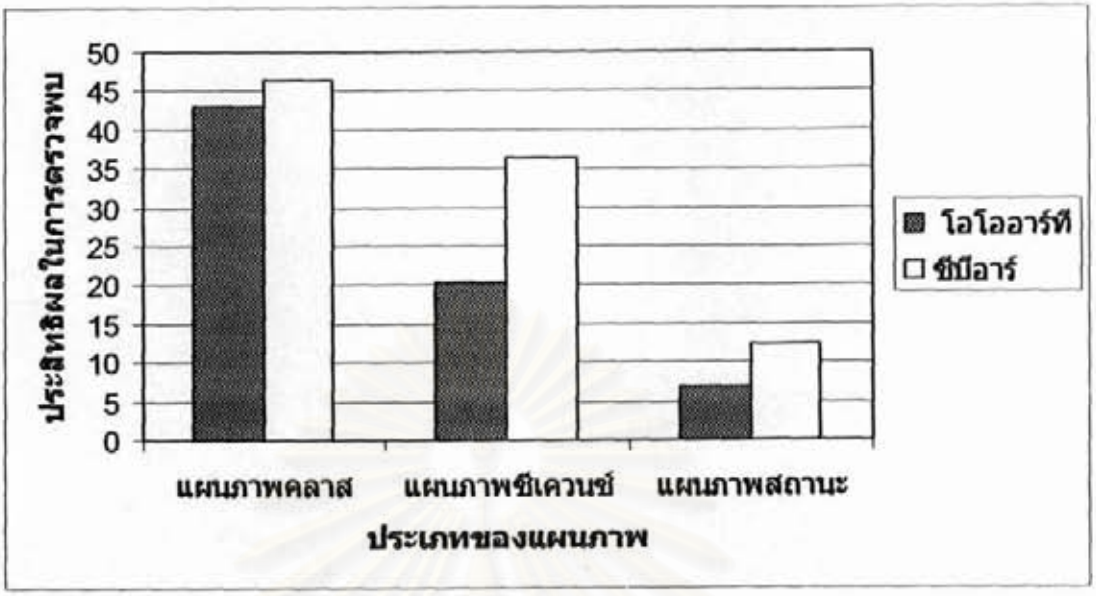
4.8.7 การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของทั้งสองเทคนิค โดยการจำแนกตามประเภทของแผนภาพ ผู้วิจัยต้องการวิเคราะห์ว่าประสิทธิผลในการตรวจพบข้อบกพร่องที่อยู่ในแผนภาพแต่ละประเภท ของทั้งสองเทคนิคมีค่าแตกต่างกันอย่างมีนัยสำคัญหรือไม่อย่างไร ในการวิเคราะห์นี้จะไม่นำแผนภาพยูสเคสมาวิเคราะห์ เนื่องจากในแผนภาพยูสเคสมีจำนวนข้อบกพร่องเพียง 1 ข้อเท่านั้น โดยสามารถคำนวณค่าประสิทธิผลในการตรวจพบได้จาก

$$\text{ประสิทธิผลในการตรวจพบ} = \left(\frac{\text{จำนวนข้อบกพร่องที่พบในแผนภาพ}}{\text{จำนวนข้อบกพร่องทั้งหมดในแผนภาพ}} \right) \times 100$$

โดยสามารถแสดงค่าสถิติของประสิทธิผลในการตรวจพบข้อบกพร่องที่อยู่ในแผนภาพแต่ละประเภทของทั้งสองเทคนิคได้ ดังตารางที่ 4-25 และสามารถเปรียบเทียบค่าเฉลี่ยของประสิทธิผลในการตรวจพบข้อบกพร่องที่อยู่ในแผนภาพแต่ละประเภทของทั้งสองเทคนิคได้ ดังรูปที่ 4-2

ตารางที่ 4-25 ตารางแสดงค่าสถิติของประสิทธิผลในการตรวจพบข้อบกพร่องที่อยู่ในแผนภาพแต่ละประเภทของทั้งสองเทคนิค

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด / ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาส	โอโออาร์ที	24	11.11 / 77.78	43.05	18.04
	ซีบีอาร์	24	0 / 88.89	46.30	20.11
ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์	โอโออาร์ที	24	0 / 85.71	20.24	24.17
	ซีบีอาร์	24	0 / 85.71	36.31	22.28
ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะ	โอโออาร์ที	24	0 / 33.33	6.94	13.83
	ซีบีอาร์	24	0 / 33.33	12.50	16.48



รูปที่ 4-2 แสดงแผนภูมิเปรียบเทียบค่าเฉลี่ยของประสิทธิผลในการตรวจพบข้อบกพร่องที่อยู่ในแผนภาพแต่ละประเภท ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์

จากรูปที่ 4-2 แสดงให้เห็นว่าเทคนิคซีบีอาร์มีประสิทธิผลในการตรวจพบข้อบกพร่อง ในทุกแผนภาพสูงกว่าเทคนิคโอโออาร์ที ผู้วิจัยจึงต้องการทดสอบว่าเทคนิคซีบีอาร์มีประสิทธิผลในการตรวจพบสูงกว่าเทคนิคโอโออาร์ทีอย่างมีนัยสำคัญหรือไม่ ซึ่งก่อนทดสอบสมมติฐานผู้วิจัยต้องตรวจสอบการแจกแจงข้อมูลก่อนว่ามีการแจกแจงแบบปกติหรือไม่ โดยผู้วิจัยตั้งสมมติฐานในการตรวจสอบการแจกแจงข้อมูลดังนี้

1. H_0: ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาส มีการแจกแจงแบบปกติ
   H_1: ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาส ไม่ได้มีการแจกแจงแบบปกติ
2. H_0: ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์ มีการแจกแจงแบบปกติ
   H_1: ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์ ไม่ได้มีการแจกแจงแบบปกติ
3. H_0: ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะ มีการแจกแจงแบบปกติ
   H_1: ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะ ไม่ได้มีการแจกแจงแบบปกติ

ในการตรวจสอบการแจกแจงข้อมูลจะใช้เทคนิค Shapiro-Wilk ในการตรวจสอบ โดยจะปฏิเสธ H_0 ถ้าค่า Sig. (Significance) ของการทดสอบน้อยกว่าระดับนัยสำคัญที่กำหนด (กัลยา วานิชย์บัญชา, 2546) โดยงานวิจัยนี้กำหนดระดับนัยสำคัญเท่ากับ 0.05

ตารางที่ 4-26 ตารางแสดงค่าสถิติทดสอบการแจกแจงปกติ (Normality Test) ของทั้ง 4 แผนภาพ

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	Shapiro-Wilk		
		Statistic	df	Sig.
ประสิทธิผลของการตรวจพบข้อบกพร่องในแผนภาพคลาส	โอโออาร์ที	.952	24	.292
	ซีบีอาร์	.944	24	.203
ประสิทธิผลของการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์	โอโออาร์ที	.811	24	.000
	ซีบีอาร์	.920	24	.059
ประสิทธิผลของการตรวจพบข้อบกพร่องในแผนภาพสถานะ	โอโออาร์ที	.503	24	.000
	ซีบีอาร์	.616	24	.000

จากตารางที่ 4-26 แสดงว่าค่าประสิทธิผลในการตรวจพบแผนภาพคลาสมีการแจกแจงข้อมูลแบบปกติ จึงใช้การทดสอบสมมติฐานแบบอิงพารามิเตอร์ โดยเลือกใช้การทดสอบสมมติฐานแบบที ส่วนค่าประสิทธิผลในการตรวจพบแผนภาพซีเควนซ์ และแผนภาพสถานะไม่ได้มีการแจกแจงข้อมูลแบบปกติ จึงใช้การทดสอบสมมติฐานแบบไม่อิงพารามิเตอร์ โดยเลือกใช้การทดสอบแบบแมนวิทนีย์ (Mann-Whitney Test)

4.8.7.1 การเปรียบเทียบประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาส ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที จากการตรวจสอบการแจกแจงข้อมูล เห็นได้ว่าประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาสมีการแจกแจงแบบปกติ ผู้วิจัยจึงทดสอบสมมติฐานโดยการทดสอบแบบที โดยก่อนที่จะทดสอบสมมติฐานจะต้องตรวจสอบก่อนว่าค่าความแปรปรวนของประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาส ของเทคนิคซีบีอาร์และเทคนิคโอโออาร์ทีมีค่าเท่ากันหรือไม่ โดยการตั้งสมมติฐานดังนี้

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

$$H_1 : \sigma^2_{OORT} \neq \sigma^2_{CBR}$$

ในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบเอฟ (F-test) โดยดูได้จากตารางที่ 4-27 ซึ่งเป็นตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances)

ตารางที่ 4-27 ตารางแสดงค่าลีวีนเทส (Levene's Test for Equality of Variances) ของประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาส

	Levene's Test for Equality of Variances	
	F	Sig.
ค่าประสิทธิผล	.116	.735

เนื่องจากในที่นี้เป็นการทดสอบสองด้าน จึงเปรียบเทียบค่า Sig. กับค่าระดับนัยสำคัญ ถ้าค่า Sig. มีค่าน้อยกว่าค่าระดับนัยสำคัญจะปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2546) ซึ่งในที่นี้กำหนดค่าระดับนัยสำคัญคือ 0.05 และค่า Sig. คือ 0.735 จะเห็นได้ว่าค่า Sig. มีค่ามากกว่าค่าระดับนัยสำคัญที่กำหนด ดังนั้นจึงยอมรับ H_0 กล่าวคือค่าประสิทธิผลของเทคนิคโอโออาร์ที และเทคนิคซีบีอาร์มีความแปรปรวนเท่ากัน

การเปรียบเทียบประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาสระหว่างการใช้เทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์มีการกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{CBR} = \mu_{OORT}$$
$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดย μ_{CBR} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาสที่ได้จากการใช้เทคนิคซีบีอาร์ และ μ_{OORT} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาสที่ได้จากการใช้เทคนิคโอโออาร์ที โดยในการทดสอบสมมติฐานนี้จะใช้สถิติทดสอบที (t-test) แบบความแปรปรวนของทั้งสองเทคนิคมีค่าเท่ากัน (จากผลการทดสอบสมมติฐานก่อนหน้า) ซึ่งสามารถแสดงตารางค่าสถิติทดสอบที (t-test for Equality of Means) ของประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาสได้ ดังตารางที่ 4-28

ตารางที่ 4-28 ตารางแสดงค่าสถิติทดสอบประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาส

	t-test for Equality of Means						
	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidential Interval of the Difference	
						Lower	Upper
ค่าประสิทธิผล	.588	46	.560	3.24167	5.51486	-7.85916	14.34250

จากสมมติฐานข้างต้น จะปฏิเสธสมมติฐาน H_0 เมื่อค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ < ค่าระดับนัยสำคัญ และค่า t > 0 โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2546) ซึ่งผลจากตารางที่ 4-23 แสดงให้เห็นว่าไม่สามารถปฏิเสธ H_0 ได้ เนื่องจากค่า $\frac{\text{Sig. (2 - tailed)}}{2}$ มีค่ามากกว่า 0.05 ดังนั้นจากการทดสอบสมมติฐานนี้แสดงให้เห็นว่าเทคนิคซีบีอาร์ไม่ได้มีค่าเฉลี่ยของประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาสสูงกว่าเทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

4.8.7.2 การเปรียบเทียบประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์ ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที จากการตรวจสอบการแจกแจงข้อมูลเห็นได้ว่า ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์ไม่ได้มีการแจกแจงแบบปกติ ผู้วิจัยจึงทดสอบสมมติฐานโดยการทดสอบแบบแมนวิทนีย์ ซึ่งกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{CBR} = \mu_{OORT}$$
$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดย μ_{CBR} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์ที่ได้จากการใช้เทคนิคซีบีอาร์ และ μ_{OORT} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์ ที่ได้จากการใช้เทคนิคโอโออาร์ที ซึ่งสามารถแสดงค่าสถิติทดสอบได้ดังตารางที่ 4-29

ตารางที่ 4-29 ตารางแสดงค่าสถิติทดสอบประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์

	แผนภาพซีเควนซ์
Mann-Whitney U	163.000
Wilcoxson W	463.000
Z	-2.627
Exact Sig. (1-tailed)	.004

เมื่อพิจารณาจากตารางที่ 4-29 จะเห็นได้ว่าค่า Exact Sig. (1-tailed) มีค่าน้อยกว่า 0.05 ดังนั้นจึงปฏิเสธสมมติฐาน H_0 กล่าวคือค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์ที่ได้จากการใช้เทคนิคซีบีอาร์ มีค่าสูงกว่าที่ได้จากการใช้เทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

4.8.7.3 การเปรียบเทียบประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะ ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที จากการตรวจสอบการแจกแจงข้อมูลเห็นได้ว่า ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะไม่ได้มีการแจกแจงแบบปกติ ผู้วิจัยจึงทดสอบสมมติฐานโดยการทดสอบแบบแมนวิทนีย์ ซึ่งกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{CBR} = \mu_{OORT}$$

$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดย μ_{CBR} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะที่ได้จากการใช้เทคนิคซีบีอาร์ และ μ_{OORT} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะ ที่ได้จากการใช้เทคนิคโอโออาร์ที ซึ่งสามารถแสดงค่าสถิติทดสอบได้ดังตารางที่ 4-30

ตารางที่ 4-30 ตารางแสดงค่าสถิติทดสอบประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะ

	แผนภาพสถานะ
Mann-Whitney U	240.000
Wilcoxson W	540.000
Z	-1.257
Exact Sig. (1-tailed)	.171

เมื่อพิจารณาจากตารางที่ 4-30 จะเห็นได้ว่าค่า Exact Sig. (1-tailed) มีค่ามากกว่า 0.05 ดังนั้นจึงยอมรับสมมติฐาน H_0 กล่าวคือค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพสถานะ ที่ได้จากการใช้เทคนิคเทคนิคโอโออาร์ทีไม่ได้มีค่าสูงกว่าเทคนิคซีบีอาร์ที่ระดับนัยสำคัญ 0.05

จากการทดสอบสมมติฐานเพื่อเปรียบเทียบค่าประสิทธิผลในการตรวจพบข้อบกพร่องในแต่ละแผนภาพแสดงให้เห็นว่า ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาสและแผนภาพสถานะของเทคนิคซีบีอาร์ไม่ได้สูงกว่าเทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05 แต่ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีควนซ์ที่ได้จากการใช้เทคนิคซีบีอาร์ มีค่าสูงกว่าที่ได้จากการใช้เทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

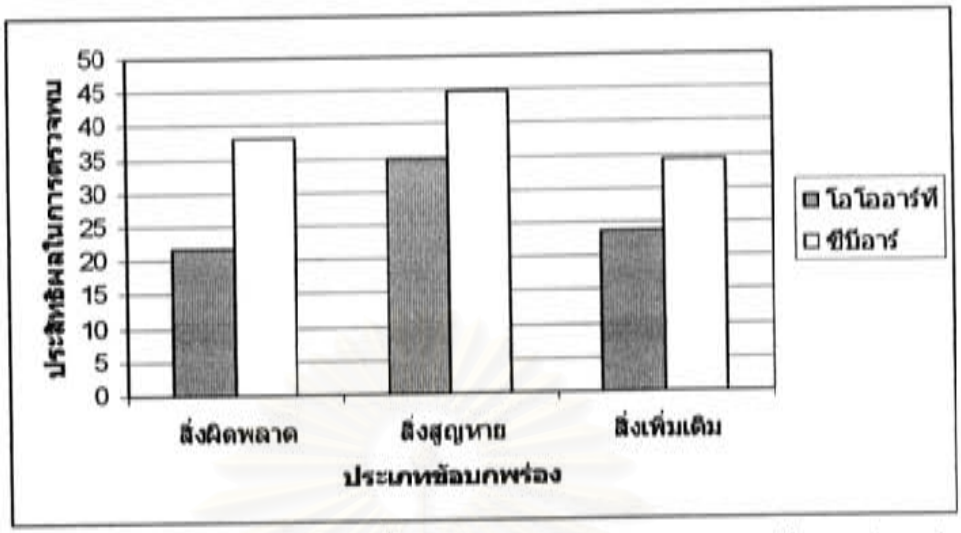
4.8.8 การเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของทั้งสองเทคนิค โดยการจำแนกตามประเภทของข้อบกพร่อง ผู้วิจัยต้องการวิเคราะห์ว่าประสิทธิผลในการตรวจพบข้อบกพร่องแต่ละประเภท ของทั้งสองเทคนิคมีค่าแตกต่างกันอย่างมีนัยสำคัญหรือไม่อย่างไร โดยคำนวณค่าประสิทธิผลในการตรวจพบได้จาก

$$\text{ประสิทธิผลในการตรวจพบ} = \left(\frac{\text{จำนวนข้อบกพร่องแต่ละประเภท}}{\text{จำนวนข้อบกพร่องทั้งหมดแต่ละประเภท}} \right) \times 100$$

โดยสามารถแสดงค่าสถิติของประสิทธิผลในการตรวจพบข้อบกพร่องแต่ละประเภทของทั้งสองเทคนิคได้ ดังตารางที่ 4-31 และสามารถเปรียบเทียบค่าเฉลี่ยของประสิทธิผลในการตรวจพบข้อบกพร่องแต่ละประเภทของทั้งสองเทคนิคได้ ดังรูปที่ 4-3

ตารางที่ 4-31 ตารางแสดงค่าสถิติของประสิทธิผลในการตรวจพบข้อบกพร่องแต่ละประเภทของทั้งสองเทคนิค

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	จำนวนหน่วยทดลอง	ค่าต่ำสุด / ค่าสูงสุด	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน
ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาด	โอโออาร์ที	24	0 / 62.50	21.88	19.24
	ซีบีอาร์	24	0 / 100	38.02	21.64
ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหาย	โอโออาร์ที	24	12.50 / 62.50	34.90	13.28
	ซีบีอาร์	24	12.50 / 75.00	44.79	17.26
ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติม	โอโออาร์ที	24	0 / 50.00	23.96	20.16
	ซีบีอาร์	24	0 / 50.00	34.38	16.17



รูปที่ 4-3 แสดงแผนภูมิเปรียบเทียบค่าเฉลี่ยของประสิทธิผลในการตรวจพบข้อบกพร่องแต่ละประเภท ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์

จากรูปที่ 4-3 แสดงให้เห็นว่าเทคนิคซีบีอาร์มีประสิทธิผลในการตรวจพบข้อบกพร่องทุกประเภทสูงกว่าเทคนิคโอโออาร์ที ผู้วิจัยต้องการทดสอบว่าเทคนิคซีบีอาร์มีประสิทธิผลในการตรวจพบสูงกว่าเทคนิคโอโออาร์ทีอย่างมีนัยสำคัญหรือไม่ ซึ่งก่อนทดสอบสมมติฐานผู้วิจัยต้องตรวจสอบการแจกแจงข้อมูลก่อนว่ามีการแจกแจงแบบปกติหรือไม่ ผู้วิจัยตั้งสมมติฐานในการตรวจสอบการแจกแจงข้อมูลดังนี้

1. H_0: ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาด มีการแจกแจงแบบปกติ
   H_1: ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาด ไม่ได้มีการแจกแจงแบบปกติ
2. H_0: ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหาย มีการแจกแจงแบบปกติ
   H_1: ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหาย ไม่ได้มีการแจกแจงแบบปกติ
3. H_0: ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติม มีการแจกแจงแบบปกติ
   H_1: ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติม ไม่ได้มีการแจกแจงแบบปกติ

ในการตรวจสอบการแจกแจงข้อมูลจะใช้เทคนิค Shapiro-Wilk ในการตรวจสอบ โดยจะปฏิเสธ H_0 ถ้าค่า Sig. (Significance) ของการทดสอบน้อยกว่าระดับนัยสำคัญที่กำหนด (กัลยา วานิชย์บัญชา, 2546) โดยงานวิจัยนี้กำหนดระดับนัยสำคัญเท่ากับ 0.05

ตารางที่ 4-32 ตารางแสดงค่าสถิติทดสอบการแจกแจงปกติ (Normality Test) ของทั้ง 4 แผนภาพ

ตัวแปร	เทคนิคการอ่านซอฟต์แวร์	Shapiro-Wilk		
		Statistic	df	Sig.
ประสิทธิผลของการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาด	โอโออาร์ที	.866	24	.004
	ซีบีอาร์	.935	24	.124
ประสิทธิผลของการตรวจพบข้อบกพร่องประเภทสิ่งสูญหาย	โอโออาร์ที	.878	24	.008
	ซีบีอาร์	.930	24	.098
ประสิทธิผลของการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติม	โอโออาร์ที	.806	24	.000
	ซีบีอาร์	.762	24	.000

จากตารางที่ 4-32 แสดงว่าค่าประสิทธิผลในการตรวจพบข้อบกพร่องทุกประเภท ไม่ได้มีการแจกแจงข้อมูลแบบปกติ จึงใช้การทดสอบสมมติฐานแบบไม่อิงพารามิเตอร์ โดยเลือกใช้การทดสอบแบบแมนวิทนีย์ (Mann-Whitney Test)

4.8.8.1 การเปรียบเทียบประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาด ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที จากการตรวจสอบการแจกแจงข้อมูลเห็นได้ว่า ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาดไม่ได้มีการแจกแจงแบบปกติ ผู้วิจัยจึงทดสอบสมมติฐานโดยการทดสอบแบบแมนวิทนีย์ ผู้วิจัยกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{CBR} = \mu_{OORT}$$

$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดย μ_{CBR} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาดที่ได้จากการใช้เทคนิคซีบีอาร์ และ μ_{OORT} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาด ที่ได้จากการใช้เทคนิคโอโออาร์ที ซึ่งสามารถแสดงค่าสถิติทดสอบได้ดังตารางที่ 4-33

ตารางที่ 4-33 ตารางแสดงค่าสถิติทดสอบประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาด

	สิ่งผิดพลาด
Mann-Whitney U	162.500
Wilcoxson W	462.500
Z	-2.633
Exact Sig. (1-tailed)	.004

เมื่อพิจารณาจากตารางที่ 4-33 จะเห็นได้ว่าค่า Exact Sig. (1-tailed) มีค่าน้อยกว่า 0.05 ดังนั้นจึงปฏิเสธสมมติฐาน H_0 กล่าวคือค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งผิดพลาด ที่ได้จากการใช้เทคนิคซีบีอาร์มีค่าสูงกว่าที่ได้จากการใช้เทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

4.8.8.2 การเปรียบเทียบประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหาย ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที จากการตรวจสอบการแจกแจงข้อมูลเห็นได้ว่าประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหายไม่ได้มีการแจกแจงแบบปกติ ผู้วิจัยจึงทดสอบสมมติฐานโดยการทดสอบแบบแมนวิทนีย์ ผู้วิจัยกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{CBR} = \mu_{OORT}$$
$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดย μ_{CBR} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหายที่ได้จากการใช้เทคนิคซีบีอาร์ และ μ_{OORT} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหาย ที่ได้จากการใช้เทคนิคโอโออาร์ที ซึ่งสามารถแสดงค่าสถิติทดสอบได้ดังตารางที่ 4-34

ตารางที่ 4-34 ตารางแสดงค่าสถิติทดสอบประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหาย

	สิ่งสูญหาย
Mann-Whitney U	194.500
Wilcoxson W	494.500
Z	-2.024
Exact Sig. (1-tailed)	.022

เมื่อพิจารณาจากตารางที่ 4-34 จะเห็นได้ว่าค่า Exact Sig. (1-tailed) มีค่าน้อยกว่า 0.05 ดังนั้นจึงปฏิเสธสมมติฐาน H_0 กล่าวคือค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งสูญหาย ที่ได้จากการใช้เทคนิคซีบีอาร์มีค่าสูงกว่าที่ได้จากการใช้เทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

4.8.8.3 การเปรียบเทียบประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติม ระหว่างเทคนิคซีบีอาร์และเทคนิคโอโออาร์ที จากการตรวจสอบการแจกแจงข้อมูลเห็นได้ว่า ประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติมไม่ได้มีการแจกแจงแบบปกติ ผู้วิจัยจึงทดสอบสมมติฐานโดยการทดสอบแบบแมนวิทนีย์ ผู้วิจัยกำหนดสมมติฐานไว้ดังนี้

$$H_0 : \mu_{CBR} = \mu_{OORT}$$
$$H_1 : \mu_{CBR} > \mu_{OORT}$$

โดย μ_{CBR} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติมที่ได้จากการใช้เทคนิคซีบีอาร์ และ μ_{OORT} หมายถึงค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติม ที่ได้จากการใช้เทคนิคโอโออาร์ที ซึ่งสามารถแสดงค่าสถิติทดสอบได้ดังตารางที่ 4-35

ตารางที่ 4-35 ตารางแสดงค่าสถิติทดสอบประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติม

	สิ่งเพิ่มเติม
Mann-Whitney U	205.000
Wilcoxson W	505.000
Z	-1.839
Exact Sig. (1-tailed)	.040

เมื่อพิจารณาจากตารางที่ 4-35 จะเห็นได้ว่าค่า Exact Sig. (1-tailed) มีค่าน้อยกว่า 0.05 ดังนั้นจึงปฏิเสธสมมติฐาน H_0 กล่าวคือค่าเฉลี่ยประสิทธิผลในการตรวจพบข้อบกพร่องประเภทสิ่งเพิ่มเติม ที่ได้จากการใช้เทคนิคซีบีอาร์มีค่าสูงกว่าที่ได้จากการใช้เทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

จากการทดสอบสมมติฐานเพื่อเปรียบเทียบค่าประสิทธิผลในการตรวจพบข้อบกพร่องแต่ละประเภทแสดงให้เห็นว่า ประสิทธิผลในการตรวจพบข้อบกพร่องทุกประเภทซึ่งได้แก่ สิ่งผิดพลาด สิ่งสูญหาย และสิ่งเพิ่มเติม ที่ได้จากการใช้เทคนิคซีบีอาร์มีค่าสูงกว่าที่ได้จากการใช้เทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 บทนำ

บทนี้นำเสนอสรุปผลการวิเคราะห์เพื่อตอบวัตถุประสงค์ของงานวิจัย การอภิปรายประเด็นต่างๆที่เกิดขึ้นในงานวิจัย การนำงานวิจัยนี้ไปใช้ประโยชน์ในเชิงทฤษฎีและเชิงประยุกต์ ข้อจำกัดของงานวิจัยและข้อเสนอแนะเพื่อเป็นโอกาสในการศึกษาต่อไปในภายภาคหน้า

5.2 การทดลองและลักษณะของหน่วยทดลอง

งานวิจัยนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) โดยใช้หน่วยทดลองเป็นนิสิตปริญญามหาบัณฑิต ที่ผ่านการเรียนรายวิชาที่มีเนื้อหาด้านการออกแบบและพัฒนาซอฟต์แวร์เชิงวัตถุ และจบการศึกษาด้านคอมพิวเตอร์ในระดับปริญญาบัณฑิต โดยมีหน่วยกิตในรายวิชาด้านคอมพิวเตอร์ไม่ต่ำกว่า 35 หน่วยกิต ทั้งหมด 48 คนสำหรับใช้ในการทดลองขั้นตอนการจัดเตรียมและแบ่งเป็นกลุ่มทดลองกลุ่มละ 3 คน ทั้งหมด 16 กลุ่ม สำหรับใช้ในการทดลองขั้นตอนการประชุมตรวจสอบ ซึ่งหน่วยทดลองจะถูกแบ่งออกเป็น 2 กลุ่มตามเทคนิคการอ่านซอฟต์แวร์ที่แตกต่างกันกลุ่มละ 24 คน

5.3 บทสรุป

งานวิจัยนี้เป็นการเปรียบเทียบประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ และประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ซึ่งสามารถสรุปผลการเปรียบเทียบได้ดังนี้

5.3.1 ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ผลการเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์พบว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ ซึ่งต่างจากที่ผู้วิจัยคาดไว้ก่อนที่จะทดลองว่า เทคนิคโอโออาร์ทีน่าจะมีค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมสูงกว่าเทคนิคซีบีอาร์

ในการวิเคราะห์ค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมนั้นจะพิจารณาจาก

(1) ระยะเวลาที่ใช้ในการตรวจสอบ ถ้าระยะเวลาที่ใช้ในการตรวจสอบน้อยจะส่งผลให้ค่าประสิทธิภาพสูง ซึ่งจากผลการทดลองเห็นได้ว่าหน่วยทดลองที่ใช้เทคนิคโอโออาร์ทีจะใช้ระยะเวลาในการตรวจสอบซอฟต์แวร์มากกว่าเทคนิคซีบีอาร์ (ดูตารางที่ 4-3 ประกอบ) ซึ่งน่าจะมีผลมาจากการที่เทคนิคโอโออาร์ทีเป็นเทคนิคการตรวจสอบซอฟต์แวร์ที่มีขั้นตอนในการตรวจสอบอย่างละเอียด ทำให้การที่หน่วยทดลองตรวจสอบซอฟต์แวร์โดยการใช้เอกสารสถานการณ์ (Scenario) เป็นเอกสารแนะนำในการตรวจสอบซอฟต์แวร์ (ดูรายละเอียดในภาคผนวก ข) จะใช้ระยะเวลานานในการอ่านและปฏิบัติตาม เนื่องจากเอกสารสถานการณ์ (Scenario) ประกอบด้วยหลายหน้ากระดาษซึ่งจะแสดงขั้นตอนในการตรวจสอบซอฟต์แวร์ที่ละเอียด และยังพบว่ามีขั้นตอนในการตรวจสอบที่ซ้ำซ้อนกัน กล่าวคือในการตรวจพบข้อบกพร่องแต่ละข้อ อาจจะตรวจพบได้ในหลายๆขั้นตอนของการตรวจสอบดังตารางที่ 3-4 ซึ่งต่างจากเทคนิคซีบีอาร์ที่มีการใช้เอกสารเช็กลิสต์เป็นเอกสารแนะนำในการตรวจสอบ เนื่องจากเอกสารเช็กลิสต์ที่ใช้ในการทดลองมีเพียงหนึ่งหน้ากระดาษเท่านั้น ซึ่งส่งผลให้ในการตรวจสอบซอฟต์แวร์นั้น หน่วยทดลองไม่เสียเวลามากนักในการอ่านเอกสารเช็กลิสต์เพื่อทำความเข้าใจในการตรวจสอบและปฏิบัติตาม และเอกสารเช็กลิสต์ไม่มีการตรวจสอบที่ซ้ำซ้อนเท่าเอกสารสถานการณ์ (Scenario)

(2) จำนวนข้อบกพร่องที่พบ ถ้าหน่วยทดลองพบข้อบกพร่องเป็นจำนวนมาก จะส่งผลให้ค่าประสิทธิภาพสูงตามไปด้วย ซึ่งจากผลการทดลองจะเห็นได้ว่าหน่วยทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบซอฟต์แวร์ ไม่ได้ปฏิบัติตามขั้นตอนในเอกสารสถานการณ์ (Scenario) ได้อย่างครบถ้วน จึงไม่สามารถตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ได้ครบทุกแผนภาพ ส่งผลต่อเนื่องให้พบจำนวนข้อบกพร่องได้น้อย ต่างจากเทคนิคซีบีอาร์ที่หน่วยทดลองส่วนใหญ่จะสามารถปฏิบัติตามเอกสารเช็กลิสต์ได้อย่างครบถ้วน จึงมีโอกาสพบข้อบกพร่องได้มากกว่า (ดูตารางที่ 4-3 ประกอบ)

5.3.2 ประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ผลการเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์พบว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ ซึ่งต่างจากที่ผู้วิจัยคาดไว้ก่อนที่จะทดลองว่า เทคนิคโอโออาร์ทีน่าจะมีค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมสูงกว่าเทคนิคซีบีอาร์

ในการวิเคราะห์ค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียม จะพิจารณาจากจำนวนข้อบกพร่องที่แต่ละหน่วยทดลองตรวจพบ กล่าวคือถ้าหน่วยทดลองพบจำนวนข้อบกพร่องมากจะส่งผลให้ค่าประสิทธิผลสูงตามไปด้วย โดยเมื่อตรวจสอบลงไปในแต่ละหน่วย

ทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบซอฟต์แวร์แล้วพบว่า หน่วยทดลองส่วนใหญ่ไม่สามารถตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ได้ครบทุกแผนภาพ จึงส่งผลให้พบจำนวนข้อบกพร่องได้น้อย ซึ่งต่างจากเทคนิคซีบีอาร์ที่หน่วยทดลองส่วนใหญ่จะสามารถปฏิบัติตามเอกสารเช็กลิสต์ได้อย่างครบถ้วน จึงมีโอกาสพบข้อบกพร่องได้มากกกว่า แต่ผู้วิจัยคาดว่าถ้าไม่มีการกำหนดระยะเวลาในการตรวจสอบแล้ว เทคนิคโอโออาร์ทีน่าจะมีประสิทธิผลสูงกว่าเทคนิคซีบีอาร์ เนื่องจากเทคนิคโอโออาร์ทีมีขั้นตอนในการตรวจสอบที่ละเอียดจึงน่าจะหาข้อบกพร่องได้มากกว่าเทคนิคซีบีอาร์ และจะเห็นได้ว่าหน่วยทดลองที่ใช้เทคนิคซีบีอาร์ในการตรวจสอบส่วนใหญ่สามารถปฏิบัติตามขั้นตอนในเอกสารเช็กลิสต์ได้อย่างครบถ้วน แต่ไม่สามารถตรวจพบข้อบกพร่องได้ครบทุกข้อ

5.3.3 ประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ผลการเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์พบว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ ซึ่งต่างจากที่ผู้วิจัยคาดไว้ก่อนที่จะทดลองว่า เทคนิคโอโออาร์ทีน่าจะมีค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคซีบีอาร์

ในการวิเคราะห์ค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบนั้นจะพิจารณาจาก

(1) ระยะเวลาที่ใช้ในการตรวจสอบ ถ้าในการตรวจสอบใช้ระยะเวลาน้อยจะส่งผลให้ค่าประสิทธิภาพในการตรวจสอบสูง ซึ่งจากผลการทดลองพบว่าทั้งสองเทคนิคใช้ระยะเวลาในการตรวจสอบซอฟต์แวร์ที่ใกล้เคียงกันมาก (ดูตารางที่ 4-4 ประกอบ) ดังนั้นในการทดลองนี้ระยะเวลาในการตรวจสอบของขั้นตอนการประชุมตรวจสอบไม่ได้ส่งผลกับค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ

(2) จำนวนข้อบกพร่องที่ตรวจพบ กล่าวคือถ้าหน่วยทดลองสามารถตรวจพบข้อบกพร่องได้เป็นจำนวนมาก จะส่งผลให้ค่าประสิทธิภาพในการตรวจสอบสูงตามไปด้วย โดยข้อบกพร่องที่บันทึกในขั้นตอนการประชุมตรวจสอบนั้น ส่วนใหญ่แล้วมาจากข้อบกพร่องที่แต่ละหน่วยทดลองในกลุ่มตรวจพบตั้งแต่ขั้นตอนการจัดเตรียม กล่าวคือการตรวจสอบขั้นตอนการประชุมตรวจสอบเป็นขั้นตอนที่หน่วยทดลองในกลุ่มจะนำผลการตรวจสอบซอฟต์แวร์ที่ได้มาประชุมร่วมกันในกลุ่มว่า หน่วยทดลองคนอื่นๆในกลุ่มเห็นด้วยหรือไม่ว่าข้อบกพร่องที่แต่ละหน่วยทดลองพบในขั้นตอน

การจัดเตรียมเป็นข้อบกพร่องจริง ถ้าเห็นด้วยจะบันทึกข้อบกพร่องนั้นลงในเอกสารบันทึกข้อบกพร่องที่ใช้ในขั้นตอนการประชุมตรวจสอบ โดยผลจากการทดลองพบว่ากลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบนั้น ตรวจพบข้อบกพร่องในขั้นตอนการจัดเตรียมน้อย จึงส่งผลให้การสรุปผลในขั้นตอนการประชุมตรวจสอบนั้นได้จำนวนข้อบกพร่องน้อยตามไปด้วย (ดูตารางที่ 4-4 ประกอบ) ทำให้ค่าประสิทธิภาพในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบต่ำ

ถึงแม้ว่าในการตรวจสอบขั้นตอนการประชุมตรวจสอบ จะให้หน่วยทดลองในกลุ่มตรวจสอบเพื่อหาข้อบกพร่องร่วมกันอีกครั้ง แต่จากผลการทดลองในตารางที่ 4-4 แสดงให้เห็นว่าในขั้นตอนการประชุมตรวจสอบนั้น กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบไม่พบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมเลย ส่วนเทคนิคซีบีอาร์นั้นเฉลี่ยแล้วพบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมไม่ถึงหนึ่งข้อ กล่าวคือขั้นตอนการประชุมตรวจสอบไม่ได้ช่วยให้พบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมเลย ซึ่งสอดคล้องกับงานวิจัยของนักวิจัยบางกลุ่ม (Votta, 1993; Porter และคณะ, 1995; Johnson และ Tjahjono, 1998) ที่กล่าวว่าขั้นตอนการประชุมตรวจสอบไม่ได้ช่วยให้พบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมเลย

5.3.4 ประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบ ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ผลการเปรียบเทียบประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์พบว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ ซึ่งต่างจากที่ผู้วิจัยคาดไว้ก่อนที่จะทดลองว่าเทคนิคโอโออาร์ทีจะมีค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคซีบีอาร์

ในการวิเคราะห์ค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์นั้นจะพิจารณาจากจำนวนข้อบกพร่องที่พบ กล่าวคือถ้ากลุ่มทดลองสามารถตรวจพบข้อบกพร่องได้เป็นจำนวนมาก จะส่งผลให้ค่าประสิทธิผลสูงตามไปด้วย ซึ่งจากผลการทดลองแสดงให้เห็นว่า กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบนั้น ตรวจพบข้อบกพร่องในขั้นตอนการจัดเตรียมน้อย จึงส่งผลให้การสรุปผลในขั้นตอนการประชุมตรวจสอบนั้นได้จำนวนข้อบกพร่องน้อยตามไปด้วย (ดูตารางที่ 4-4 ประกอบ) ทำให้ค่าประสิทธิผลในการตรวจสอบซอฟต์แวร์ของขั้นตอนการจัดเตรียมรวมกับขั้นตอนการประชุมตรวจสอบต่ำ และถึงแม้ว่าในการตรวจสอบขั้นตอนการประชุมตรวจสอบ จะให้หน่วยทดลองในกลุ่มตรวจสอบเพื่อหาข้อบกพร่องร่วมกันอีกครั้ง แต่จากผลการทดลองแสดงให้

เห็นว่า ในขั้นตอนการประชุมตรวจสอบนั้น กลุ่มทดลองที่ใช้เทคนิคโอโออาร์ทีในการตรวจสอบไม่มีการพบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมเลย ส่วนเทคนิคซีบีอาร์นั้นเฉลี่ยแล้วพบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมไม่ถึงหนึ่งข้อ

5.3.5 ประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบ ระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ ผลการเปรียบเทียบประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบระหว่างเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์พบว่า เทคนิคโอโออาร์ทีไม่ได้มีค่าประสิทธิผลในการกำจัดผลบวกปลอมสูงกว่าเทคนิคซีบีอาร์ในทางสถิติ ซึ่งต่างจากที่ผู้วิจัยคาดไว้ก่อนที่จะทดลองว่าเทคนิคโอโออาร์ทีจะมีค่าประสิทธิผลในการกำจัดผลบวกปลอมของขั้นตอนการประชุมตรวจสอบสูงกว่าเทคนิคซีบีอาร์

งานวิจัยนี้มีค่าประสิทธิผลในการกำจัดผลบวกปลอมของเทคนิคซีบีอาร์มีค่าประมาณ 63% และเทคนิคโอโออาร์ทีมีค่าประมาณ 61% และจากงานวิจัยของ Sabalaiauskaite (2004) แสดงค่าประสิทธิผลในการกำจัดผลบวกปลอมของเทคนิคซีบีอาร์เฉลี่ยที่ 73% และเทคนิคพีบีอาร์ที่ 64% จะเห็นได้ว่าแต่ละเทคนิคมีค่าประสิทธิผลในการกำจัดผลบวกปลอมที่ไม่แตกต่างกันมากนัก ดังนั้นผู้วิจัยคาดว่าประสิทธิผลในการกำจัดผลบวกปลอมไม่ได้ขึ้นกับเทคนิคการอ่านซอฟต์แวร์ แต่น่าจะมีสาเหตุมาจาก (1) ความเข้าใจในเอกสารคำแนะนำของหน่วยทดลอง (2) ความรู้ทางด้านการออกแบบซอฟต์แวร์เชิงวัตถุของหน่วยทดลอง ซึ่งเมื่อเข้าสู่ขั้นตอนการประชุมตรวจสอบมีการประชุมร่วมกับหน่วยทดลองอื่นๆในกลุ่มแล้ว จึงส่งผลให้หน่วยทดลองมีความเข้าใจในเอกสารคำแนะนำ และการออกแบบซอฟต์แวร์เชิงวัตถุมากขึ้น จึงสามารถกำจัดผลบวกปลอมได้

ถึงแม้ว่าจากการทดลองของนักวิจัยบางกลุ่ม (Votta, 1993; Porter และคณะ, 1995; Johnson และ Tjahjono, 1998) ที่ได้ศึกษาประสิทธิผลของการตรวจสอบซอฟต์แวร์ในขั้นตอนการประชุมตรวจสอบ แสดงให้เห็นว่าขั้นตอนการประชุมตรวจสอบไม่ได้ช่วยให้พบข้อบกพร่องเพิ่มจากขั้นตอนการจัดเตรียมเลย แต่จากงานวิจัยนี้แสดงให้เห็นว่า ขั้นตอนการประชุมตรวจสอบช่วยในการกำจัดผลบวกปลอมได้ ซึ่งช่วยให้เอกสารแสดงการออกแบบซอฟต์แวร์มีคุณภาพมากขึ้น เนื่องจากถ้าปล่อยให้เกิดผลบวกปลอมจะทำให้การแก้ไขข้อบกพร่องผิดพลาด ซึ่งจะส่งผลต่อการพัฒนาซอฟต์แวร์ ทำให้ซอฟต์แวร์ที่ได้นั้นไม่มีคุณภาพ

5.3.6 สรุปการวิเคราะห์ข้อมูลเพิ่มเติม (Exploration) จากการวิเคราะห์ข้อมูลเพิ่มเติมพบว่า เทคนิคซีบีอาร์มีประสิทธิภาพและประสิทธิผลในการตรวจสอบสูงกว่าเทคนิคโอโออาร์ที ที่ระดับนัยสำคัญ 0.05 โดยที่ประสิทธิผลในการกำจัดผลบวกปลอมของทั้งสองเทคนิคมีค่าไม่แตกต่างกัน และเมื่อวิเคราะห์ค่าประสิทธิผลในการตรวจพบข้อบกพร่องโดยจำแนกตามแผนภาพแสดงให้เห็นว่าเทคนิคซีบีอาร์มีประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์สูง

กว่าเทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05 แต่ประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาสและแผนภาพสถานะของทั้งสองเทคนิคมีค่าไม่แตกต่างกัน นอกจากนี้ผู้วิจัยได้วิเคราะห์ค่าประสิทธิผลในการตรวจพบข้อบกพร่องโดยจำแนกตามประเภทของข้อบกพร่อง ซึ่งผลที่ได้แสดงให้เห็นว่าเทคนิคซีบีอาร์มีประสิทธิผลในการตรวจพบข้อบกพร่องแต่ละประเภทซึ่งได้แก่ สิ่งผิดพลาด สิ่งสูญหาย และสิ่งเพิ่มเติม สูงกว่าเทคนิคโอโออาร์ทีที่ระดับนัยสำคัญ 0.05

5.4 การนำงานวิจัยไปประยุกต์ใช้ (Contribution)

งานวิจัยนี้นอกจากนำไปใช้ในเชิงทฤษฎีแล้ว ยังสามารถนำไปประยุกต์ใช้ในทางปฏิบัติได้ดังต่อไปนี้

5.4.1 การนำงานวิจัยไปใช้ในเชิงทฤษฎี (Theoretical Contribution) งานวิจัยนี้เป็นการต่อยอดองค์ความรู้ด้านการตรวจสอบซอฟต์แวร์ (Software Inspection) เนื่องจากผู้วิจัยทบทวนวรรณกรรมในอดีต (Literature Review) แล้วพบว่า มีการนำเทคนิคโอโออาร์ทีมาศึกษาน้อยมาก ทั้งที่เทคนิคโอโออาร์ทีเป็นเทคนิคสำหรับใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุโดยเฉพาะ โดยมีงานวิจัยที่เกี่ยวข้องกับเทคนิคโอโออาร์ทีคืองานวิจัยของ Conradi และคณะ (2003) ที่ศึกษาเทียบเปรียบเทียบประสิทธิภาพในการตรวจสอบซอฟต์แวร์ระหว่างเทคนิคโอโออาร์ทีกับเทคนิคอาร์แอนด์ไอ (R&I Techniques) ซึ่งเป็นเทคนิคที่ใช้ในการตรวจสอบซอฟต์แวร์ของบริษัทอีริคสันประเทศนอร์เวย์ และงานวิจัยของ Melo และคณะ (2001) กล่าวว่าเทคนิคโอโออาร์ทีสามารถค้นหาข้อบกพร่องได้เป็นจำนวนมากและสามารถนำไปใช้ในอุตสาหกรรมซอฟต์แวร์ได้ แต่ยังไม่มีการนำเทคนิคโอโออาร์ทีไปศึกษาโดยการเปรียบเทียบกับเทคนิคการอ่านซอฟต์แวร์อื่นเลย ดังนั้นผู้วิจัยจึงศึกษาว่าระหว่างการนำเทคนิคโอโออาร์ทีและเทคนิคซีบีอาร์ มาใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยวิธีการเชิงวัตถุแล้ว เทคนิคไหนจะให้ประสิทธิภาพและประสิทธิผลในการตรวจสอบมากกว่ากัน เพื่อเป็นแนวทางให้กับผู้ที่สนใจในด้านของการตรวจสอบซอฟต์แวร์ได้นำข้อมูลเหล่านี้ไปใช้ในการศึกษาต่อไป

5.4.2 การนำงานวิจัยไปใช้ในเชิงประยุกต์ (Practical Contribution) จากผลการทดลองของงานวิจัยนี้แสดงให้เห็นว่า เทคนิคซีบีอาร์มีประสิทธิผลในการตรวจสอบ ประสิทธิภาพในการตรวจสอบ และประสิทธิผลในการกำจัดผลบวกปลอมสูงกว่าเทคนิคโอโออาร์ที ซึ่งเมื่อสำรวจผลการวิเคราะห์ข้อมูลเพิ่มเติม โดยการจำแนกตามประเภทของแผนภาพ แสดงให้เห็นว่าการตรวจสอบในขั้นตอนการจัดเตรียมซึ่งมีระยะเวลาจำกัดนั้น แม้ทั้งสองเทคนิคจะมีประสิทธิผลในการตรวจพบข้อบกพร่องในแผนภาพคลาส และแผนภาพสถานะที่ไม่แตกต่างกัน แต่ในการตรวจพบข้อบกพร่องในแผนภาพซีเควนซ์นั้น เทคนิคซีบีอาร์มีประสิทธิผลในการตรวจพบข้อบกพร่องสูงกว่าเทคนิคโอ

โออาร์ที และเมื่อสำรวจลงไปโดยการจำแนกตามประเภทของข้อบกพร่องแล้วพบว่า เทคนิคซีบีอาร์มีประสิทธิผลในการตรวจพบข้อบกพร่องทุกประเภทสูงกว่าเทคนิคโอโออาร์ที

ในการเลือกเทคนิคการอ่านซอฟต์แวร์ไปใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์ที่ออกแบบโดยวิธีการเชิงวัตถุนั้น ถ้าจำกัดระยะเวลาที่ใช้ในการตรวจสอบซอฟต์แวร์หรือผู้ตรวจสอบมีประสบการณ์ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุมากแล้ว องค์กรรับจ้างพัฒนาซอฟต์แวร์ควรจะนำเทคนิคซีบีอาร์ไปใช้ในการตรวจสอบซอฟต์แวร์แทนที่เทคนิคโอโออาร์ที เนื่องจากเป็นเทคนิคการอ่านซอฟต์แวร์ที่ไม่ใช้เวลาในการตรวจสอบมาก แต่ก็สามารถพบข้อบกพร่องได้ เนื่องจากเอกสารเช็กลิสต์ซึ่งเป็นเอกสารคำแนะนำสำหรับเทคนิคซีบีอาร์ ไม่ได้มีคำอธิบายในการตรวจสอบที่ละเอียดจึงไม่เสียเวลาในการอ่านและปฏิบัติตามมากนัก จำเป็นต้องอาศัยประสบการณ์ของผู้ตรวจสอบ

ถ้าองค์กรรับจ้างพัฒนาซอฟต์แวร์ต้องการเอกสารแสดงการออกแบบซอฟต์แวร์ที่มีความถูกต้องมากที่สุดโดยไม่คำนึงถึงระยะเวลาที่ใช้ในการตรวจสอบ หรือผู้ตรวจสอบยังไม่ใคร่มีประสบการณ์ตรวจสอบเอกสารดังกล่าว เทคนิคโอโออาร์ทีเป็นตัวเลือกที่เหมาะสมในการนำไปใช้ตรวจสอบซอฟต์แวร์ เนื่องจากเอกสารสถานการณ์ (Scenario) ซึ่งเป็นเอกสารคำแนะนำสำหรับเทคนิคโอโออาร์ที มีขั้นตอนการตรวจสอบที่ละเอียด จึงต้องใช้เวลานานในการอ่านเพื่อทำความเข้าใจและปฏิบัติตาม

งานวิจัยนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) ดังนั้นระบบที่นำมาใช้ในงานวิจัยนี้จึงเป็นระบบที่มีขนาดเล็กไม่มีความซับซ้อนมากนัก ซึ่งผู้วิจัยคาดว่าถ้าระบบที่นำมาตรวจสอบมีความซับซ้อนมากกว่างานวิจัยนี้แล้ว เทคนิคโอโออาร์ทีน่าจะสามารถตรวจพบข้อบกพร่องได้มากกว่าการนำเทคนิคซีบีอาร์ไปใช้ในการตรวจสอบ

5.5 ข้อจำกัดและข้อเสนอแนะของงานวิจัย

งานวิจัยนี้มีข้อจำกัดบางประการ ซึ่งผู้วิจัยขอสรุปและเสนอแนะแนวทาง ดังต่อไปนี้

1. การนำผลการทดลองไปใช้ให้คำนึงด้วยว่า หน่วยทดลองของงานวิจัยนี้เป็นนิสิตในระดับปริญญามหาบัณฑิต สาขาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ไม่ใช่ผู้ตรวจสอบมืออาชีพจากองค์กรรับจ้างพัฒนาซอฟต์แวร์ ดังนั้นในการนำไปขยายผลกับผู้ตรวจสอบซอฟต์แวร์ในองค์กรรับจ้างพัฒนาซอฟต์แวร์อาจได้ผลที่แตกต่างไป

2. ในการทดลองผู้วิจัยได้กำหนดระยะเวลาที่ใช้ในการทดลองแต่ละขั้นตอนไว้ แต่ปกติแล้วการตรวจสอบซอฟต์แวร์ในขั้นตอนการจัดเตรียม เป็นขั้นตอนที่ให้ผู้ตรวจสอบนำเอกสารกลับไปตรวจสอบแล้วนำมาส่ง จึงไม่มีการกำหนดระยะเวลาที่แน่นอน ดังนั้นเมื่อนำผลที่ได้จากการ

ทดลองไปขยายผลกับการตรวจสอบซอฟต์แวร์ในองค์กรรับจ้างพัฒนาซอฟต์แวร์ อาจให้ผลที่แตกต่างออกไป

3. เอกสารแสดงการออกแบบซอฟต์แวร์ที่นำมาใช้ในการทดลองเป็นระบบที่มีขนาดเล็กและมีความซับซ้อนไม่มากนัก เพื่อให้หน่วยทดลองสามารถค้นหาข้อบกพร่องได้ในระยะเวลาที่กำหนด ซึ่งเมื่อนำไปขยายผลกับระบบจริงที่องค์กรรับจ้างพัฒนาซอฟต์แวร์จัดทำขึ้นอาจให้ผลที่แตกต่างไป

4. ในการทดลองนั้นผู้วิจัยใช้เอกสารแสดงการออกแบบซอฟต์แวร์ที่มีความถูกต้องสมบูรณ์มากำหนดข้อบกพร่องเอง ซึ่งผลที่ได้นั้นเมื่อนำไปขยายผลกับเอกสารแสดงการออกแบบซอฟต์แวร์ ที่กลุ่มผู้ผลิตซอฟต์แวร์จัดทำขึ้นจริงซึ่งอาจมีลักษณะข้อบกพร่องที่แตกต่างไปแล้ว อาจให้ผลที่แตกต่างไป

5. ควรมีการศึกษาต่อไปโดยเปลี่ยนแผนแบบการทดลอง เช่น นำผู้ตรวจสอบซอฟต์แวร์จากองค์กรรับจ้างพัฒนาซอฟต์แวร์มาเป็นหน่วยทดลอง เพิ่มระยะเวลาที่ใช้ในการตรวจสอบ ขั้นตอนการจัดเตรียม เปลี่ยนระบบที่นำมาใช้ในการทดลองให้มีขนาดใหญ่ซับซ้อน หรือนำเอกสารแสดงการออกแบบซอฟต์แวร์จากองค์กรรับจ้างพัฒนาซอฟต์แวร์จริงมาใช้ในการทดลอง เพื่อดูว่าประสิทธิภาพและประสิทธิผลในการตรวจสอบซอฟต์แวร์จะเปลี่ยนแปลงไปอย่างไร

6. งานวิจัยนี้เป็นการเปรียบเทียบเทคนิคการอ่านซอฟต์แวร์ ระหว่างเทคนิคไอไออาร์ทีและเทคนิคซีบีอาร์เท่านั้น จึงควรมีการต่อยอดโดยการนำเทคนิคการอ่านซอฟต์แวร์เทคนิคอื่นๆ มาเปรียบเทียบกับเทคนิคไอไออาร์ทีโดยใช้ระบบเดียวกับของงานวิจัยนี้ เนื่องจากที่กล่าวมาแล้วข้างต้นว่า จากการทบทวนวรรณกรรมผู้วิจัยพบว่างานวิจัยที่เป็นการศึกษาเทคนิคไอไออาร์ทียังมีไม่มากเท่าไรนัก

รายการอ้างอิง

ภาษาไทย

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

กัลยา วาณิชย์บัญชา. 2546. การใช้ SPSS for Windows ในการวิเคราะห์ข้อมูล. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

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

บุญประเสริฐ สุรักษ์รัตนสกุล. 2546. วิธีการตรวจสอบวากยสัมพันธ์ของแผนภาพยูเอ็มแอล. วิทยานิพนธ์ปริญญามหาบัณฑิต. ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์.

ราชบัณฑิตยสถาน. 2546. ศัพท์คอมพิวเตอร์และเทคโนโลยีสารสนเทศ ฉบับราชบัณฑิตยสถาน. พิมพ์ครั้งที่ 6. กรุงเทพมหานคร: สำนักพิมพ์สหมิตรพริ้นติ้ง.

ศิริชัย พงษ์วิชัย. 2548. การวิเคราะห์ข้อมูลทางสถิติด้วยคอมพิวเตอร์. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

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

คณะกรรมการวิจัยแห่งชาติ. 2547. Research On-Line: ตำราชุดฝึกอบรมหลักสูตร "นักวิจัย" [On-Line]. Available from: http://www.elearning.nrct.net/file/Research1-9.pdf.

ภาษาอังกฤษ

Ahrendt, W., Barr, T., Beckert, B., Giese, M., Hahnle, R., Menzel, W., Mostowski, W. and Schmitt, P. H. 2002. The Key System: Integrating Object-Oriented Design and Formal Methods. Lecture Notes In Computer Science, Publisher: Springer Berlin / Heidelberg 2306: 327.

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. 2000. Software Product Improvement with Inspection. Proceedings of The 26th EUROMICRO Conference (EUROMICRO'00) 2: 2262.

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

Briand, L., El Emam K., Laitenberger O. and Fussbroich T. 1998. Using Simulation to Build Inspection Efficiency Benchmarks for Development Projects. Proceedings of the 20th International Conference on Software Engineering, Kyoto, Japan : 340-349.

Broy, M. and Denert, E. 2002. Software Pioneers: Contributions to Software Engineering. New York: Springer-Verlag.

Carnegie Mellon Software Engineering Institute. 2002. Research On-Line: Capability Maturity Model Integration (CMMI℠), Version 1.1 [On-Line]. Available from: http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr012.pdf.

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., Marjara, A., Hantho, O., Frotveit, T. and Skåtevik, B. 1999. A study of inspections and testing at Ericsson, Norway. Proc. PROFES'99 published by VTT : 263-284.

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.

Dunsmore, A., Roper M. and Wood M. 2001. Systematic Object-Oriented Inspection - An Empirical Study. Proceedings of the 23 International Conference on Software Engineering : 135-144.

Dunsmore, A. 2002. Investigating Effective Inspection of Object-Oriented Code. Ph.D. Thesis. University of Strathclyde.

Dunsmore, A., Roper M. and Wood M. 2003. The Development and Evaluation of Three Diverse Techniques for OO Code Inspection. IEEE Transactions on Software Engineering 8: 677-686.

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

Fagan, M. E. 1986. Advances in Software Inspection. IEEE Transactions on Software Engineering 7: 744-751.

Fisher, M. and Light, W. R. 1979. "Definitions in Software Quality Management" in Software Quality Management. New York: Petrocelli Books.

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.

Halling, M. and Biffl, St. 2001. Using Reading Techniques to Focus Inspection Performance. Proc. Euromicro 2001 Workshop on Software Product and Process Improvement, Warsaw, IEEE Comp. Soc. Press : 248-257.

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

IEEE Standard. 1990. A Glossary of Software Engineering Terminology. USA: IEEE 610.12-1990, Institute of Electrical and Electronic Engineers.

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

Kenett, R. S. and Baker, E. R. 1999. Software Process Quality Management and Control. New York: Marcel Dekker.

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.

Laitenberger, O. and DeBaud, J. M. 2000. An Encompassing Life Cycle Centric Survey of Software Inspection. The Journal of Systems and Software 50, 1: 5-31.

Lang, N. A. and Jacob, Peter H. M. 2005. Research On-Line: RMIS: MIddleware For Transparent Object-ORIENTED Modling in Multi-Simulator Systems [On-Line]. Available from: http://www.informs-cs.org/wsc05papers/036.pdf.

Larman, C. 2002. Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and the Unified Process. Second Edition. USA: Prentice-Hall International.

McCall, J. A. 1979. An Introduction to Software Quality Metrics. New York: A Petrocelli Book.

Miller, J., Wood, M. and Roper, M. 1998. Further experiences with scenarios and checklists. Empirical Software Engineering Journal 3, 1: 37-64.

Melo, W., Shull, F., Travassos, G. H. 2001. Research On-Line: Software Review Guidelines [On-Line]. Available from: http://cronos.cos.ufrj.br/publicacoes/reltec/es55601.pdf.

Mrdalj, S., Scazzero, J. and Jovanovic V. 2004. Research On-Line: 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. 2002. 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.

Pfleeger S. L. 1998. Software Engineering: Theory and Practice. USA: Prentice-Hall.

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.

Regnell, B., Runeson, P. and Thelin, T. 2000. Are the Perspectives Really Different? Further Experimentation on Scenario-Based Reading of Requirements. Empirical Software Engineering: An International Journal 5, 4: 331-356.

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.

Sabaliauskaite, G., Matsukawa, F., Kusumoto, S. and Inoue K. 2003. Further Investigations of Reading Techniques for Object-Oriented Design Inspection. Information and Software Technology 45, 9: 571-585.

Sabaliauskaite, G. 2004. Investigating Defect Detection in Object-Oriented Design and Cost – Effectiveness of Software. Ph.D. Thesis. Faculty of Engineering Science of Osaka University.

Sabaliauskaite, G., Kusumoto, S. and Inoue K. 2004. Assessing Defect Detection Performance of Interacting Teams in Object-Oriented Design Inspection. Information and Software Technology 46, 13: 875-886.

Schach, S. R. 1999. Classical and Object-Oriented Software Engineering with UML and C++. Fourth Edition. Singapore: McGraw-Hill.

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.

Shull, F., Rus, I. and Basili, V. 2000. How Perspective Based Reading can Improve Requirements Reading. IEEE Computer 33, 7: 73-79.

Software Engineering Technical Committee of the IEEE Computer Society. 1983. Software Engineering Technical Committee of the IEEE Computer Society, IEEE Standard Glossary of Software Engineering Terminology, IEEE-STD-729-1983. New York: IEEE.

Stone, B. 2001. Research On-Line: OO Design and the Unified Modelling Language[On-Line]. Available from: http://www.eeng.dcu.ie/~alife/gmit/CA212/week10uml.ppt.

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

Thelin, T. 2002. Empirical Evaluations of Usage-Based Reading and Fault Content Estimation for Software Inspections. Ph.D. Thesis. Lund University.

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, V. 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, V. 2002. Research On-Line: Reading Techniques for OO Design Inspections[On-Line]. Available from: http://www.cs.umd.edu /Library/TRs/CS-TR-4353/CS-TR-4353.pdf#search= %22Reading%20Techniques %20for% 20OO%20Design%20%22.

United States Air Force. 1987. Software Quality Indicators: Management Quality Insight. Washington DC.: United States Air Force.

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

Unhelkar, B. 2005. Verification and Validation for Quality of UML 2.0 Models. New Jersey: John Wiley & Sons.

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

Wang, Y. and Lu, Y. Research On-Line: Chapter 3 : SQA for Object Models [On-Line]. Available from: http://www.cis.ksu.edu/SELab/SQA01/c3/841team.html.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. and Wesslen, A. 2000. Experimentation in Software Engineering: an Introduction. International Series in Software Engineering Vol. 6. New York: Springer-Verlag.

Wohlin, C., Aurum, A., Petersson, H., Shull, F. and Ciolkowski, M. 2002. Software Inspection Benchmarking – A Qualitative and Quantitative Comparative Opportunity. Proceedings of the Eighth IEEE Symposium on Software Metrics : 118-127.

Yeh H. T. 1993. Software Process Quality. New York: McGraw-Hill.

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

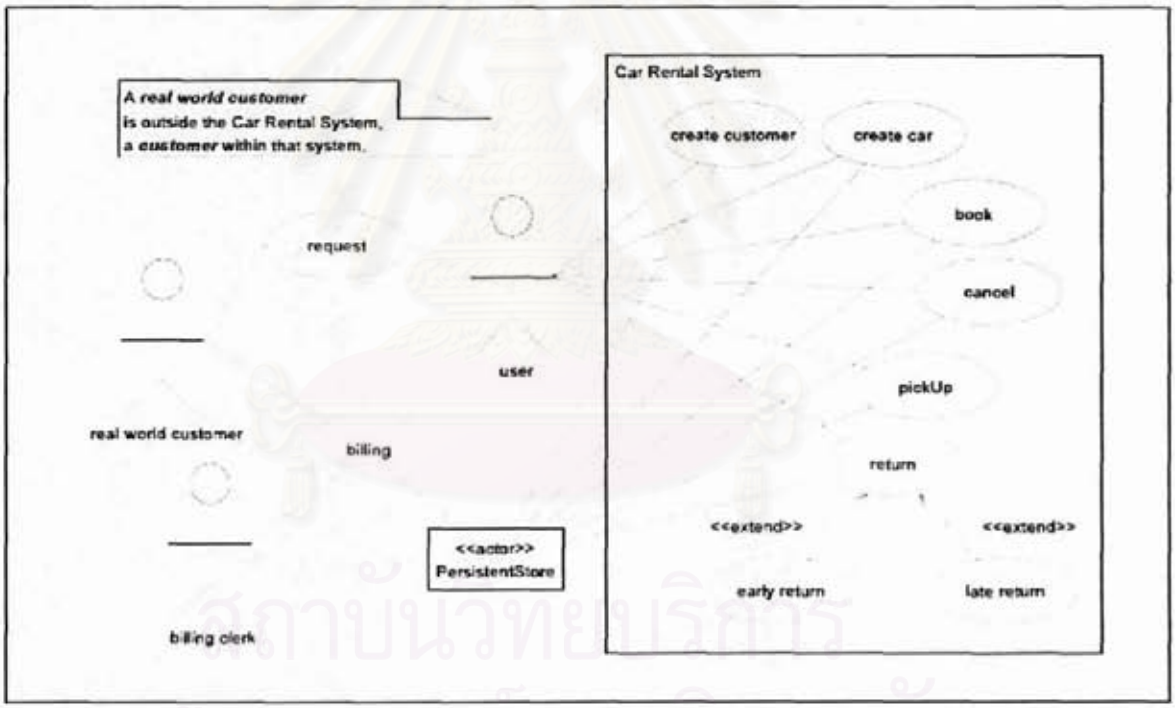
ภาคผนวก ก

เอกสารแสดงการออกแบบซอฟต์แวร์

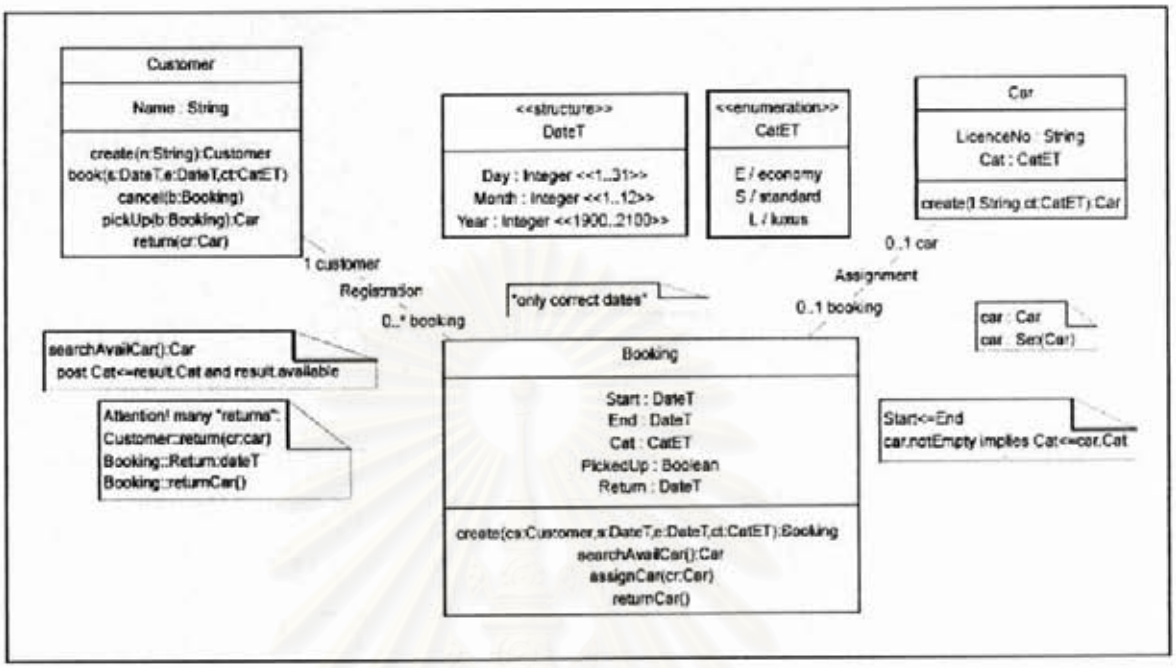
1. ต้นแบบเอกสารแสดงการออกแบบซอฟต์แวร์

เอกสารแสดงการออกแบบซอฟต์แวร์ที่ใช้ในการทดลองนี้นำมาจาก กรณีศึกษาจากเอกสารการสอนของมหาวิทยาลัยเบรเมน (University of Bremen) คณะวิทยาศาสตร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ (Department for Mathematics and Computer Science) ซึ่งประกอบด้วย แผนภาพยูสเคส (Use Case Diagram) แผนภาพคลาส (Class Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) และแผนภาพสถานะ (State Machine Diagram)

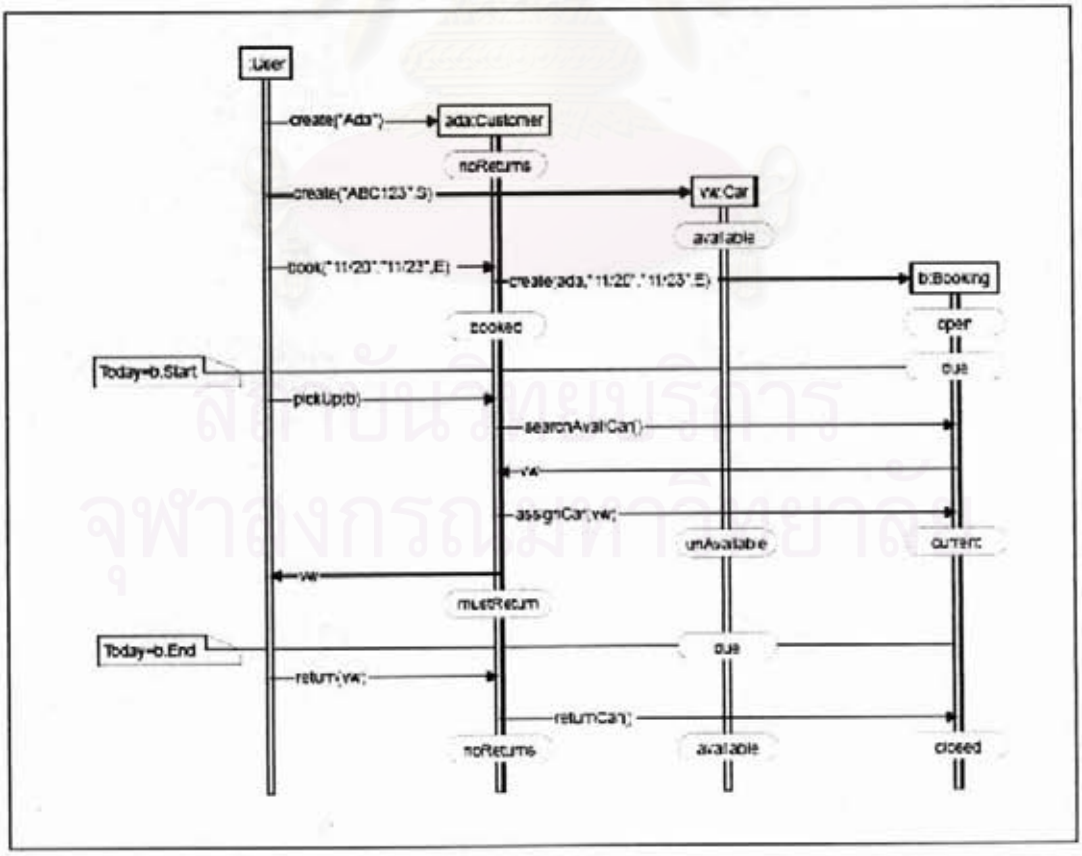
1.1 แผนภาพยูสเคส

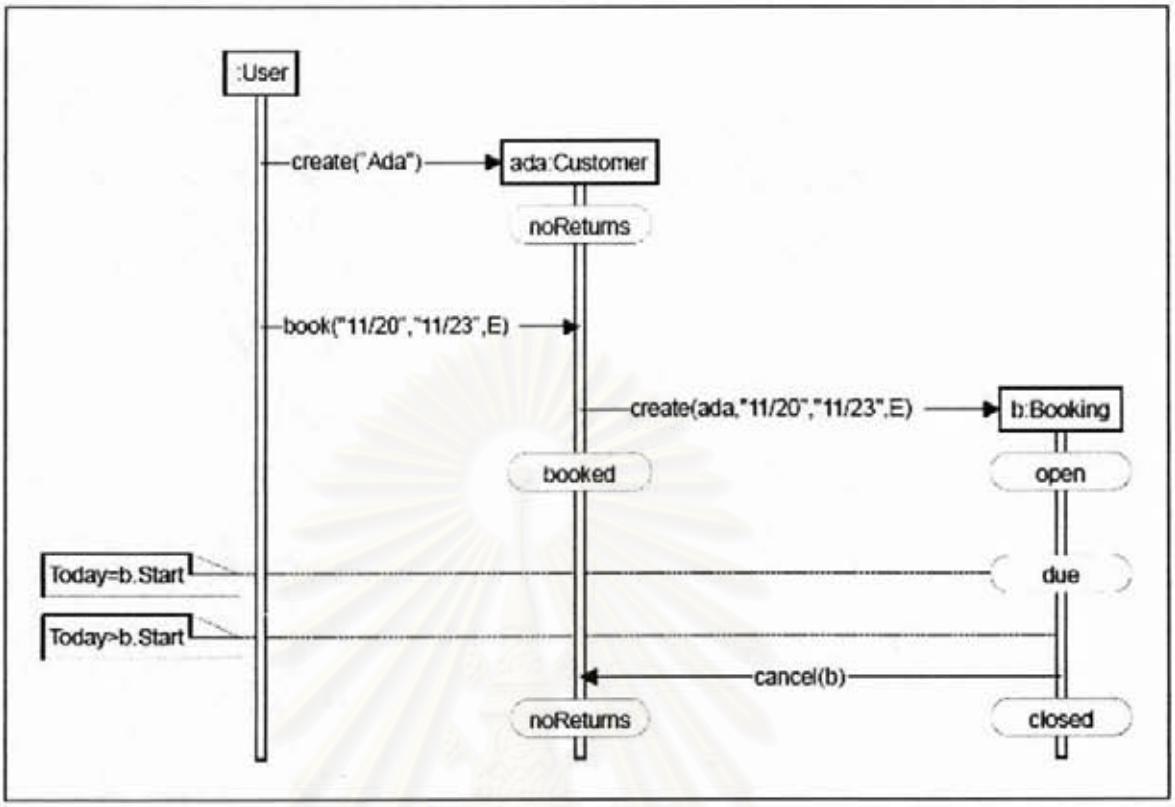


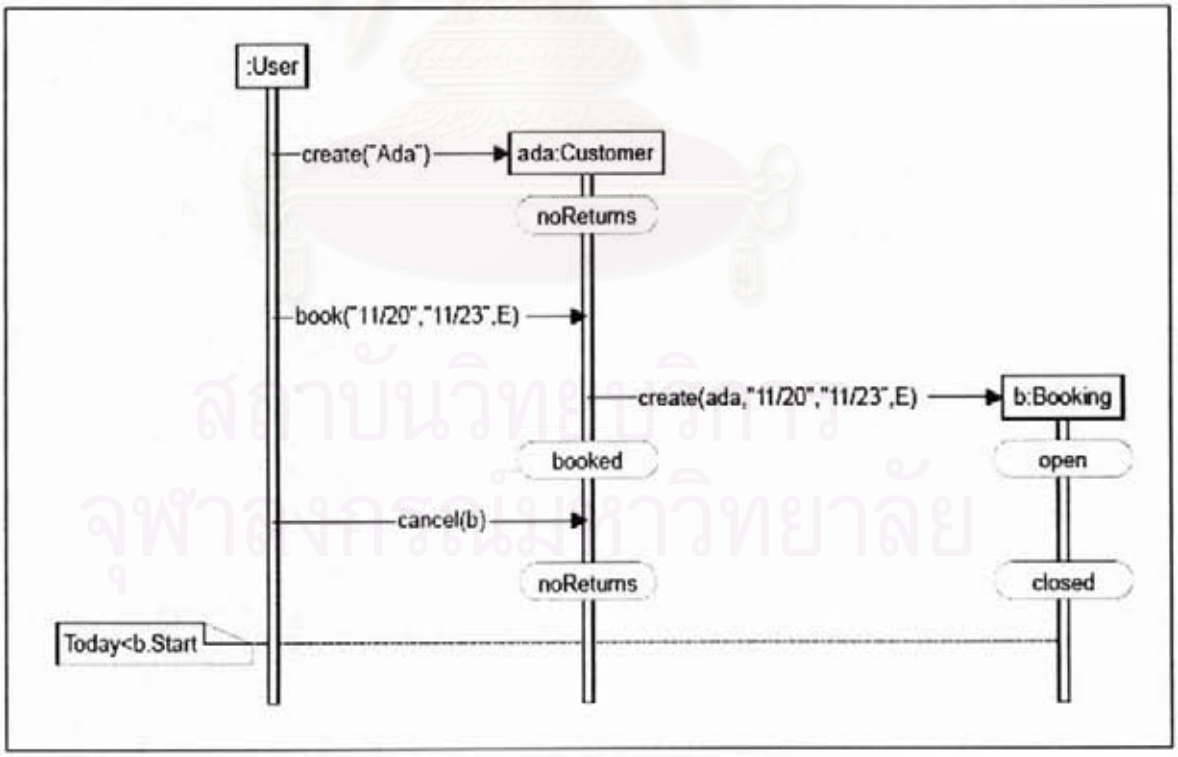
1.2 แผนภาพคลาส

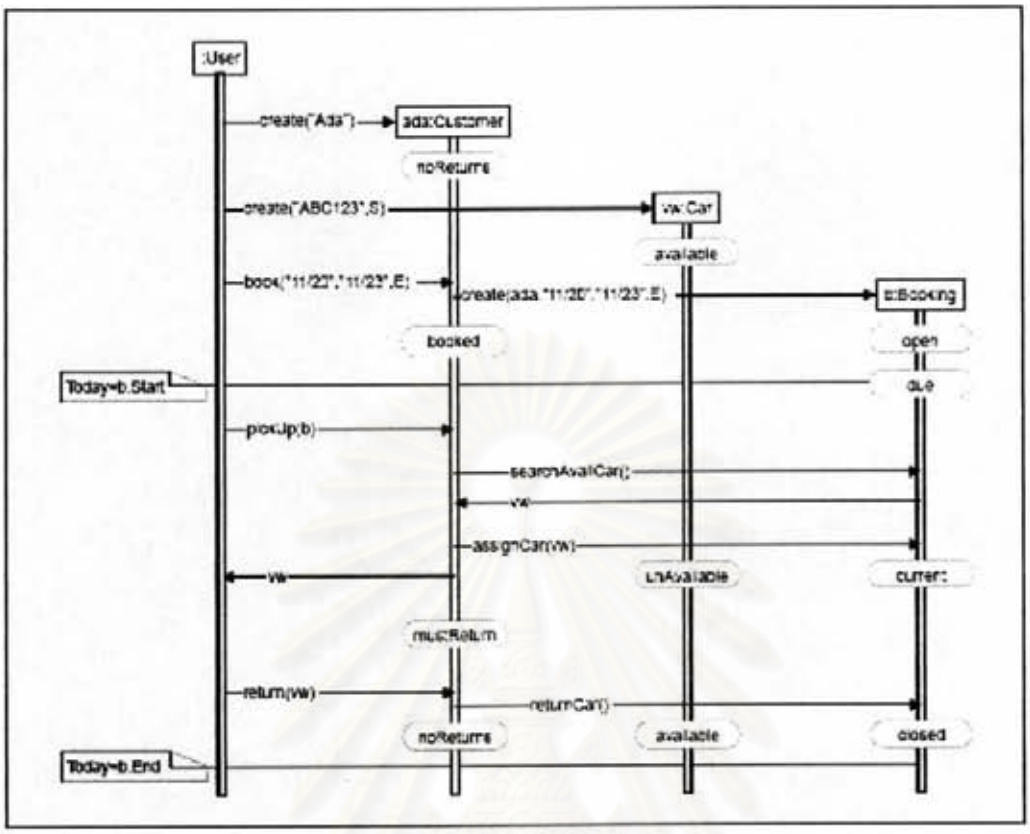


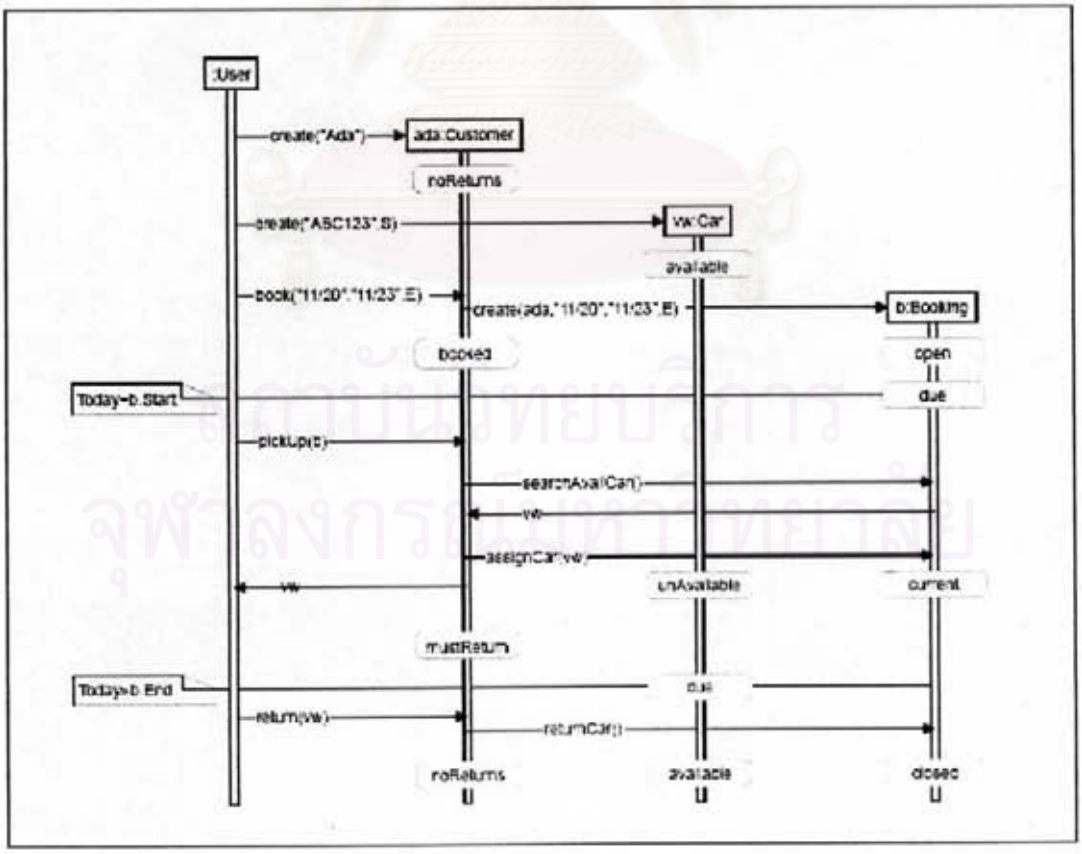
1.3 แผนภาพซีเควนซ์

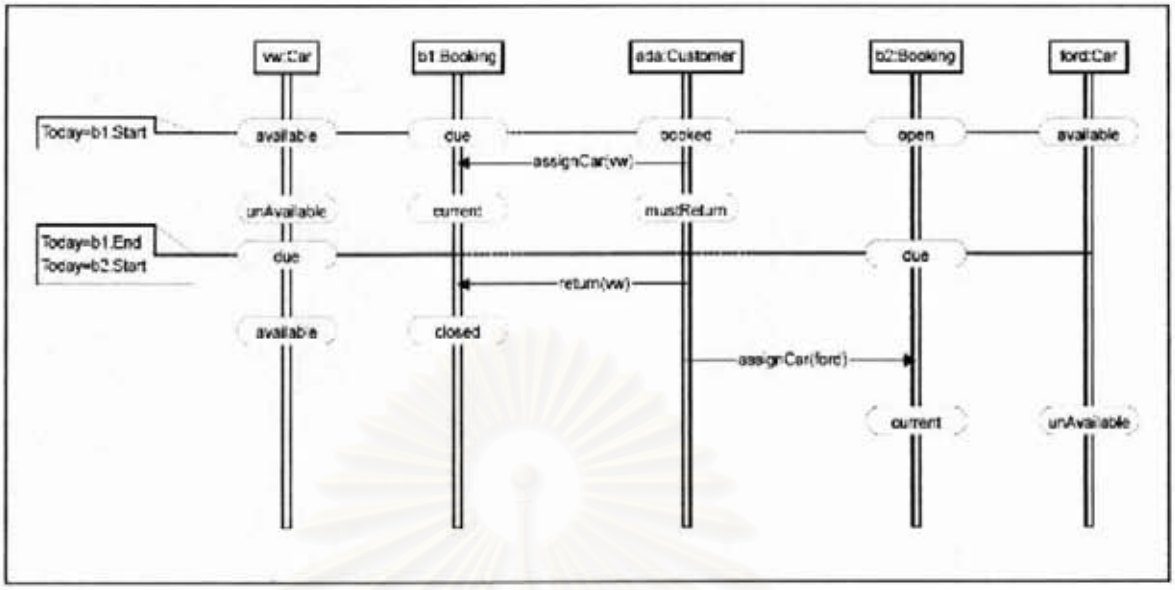


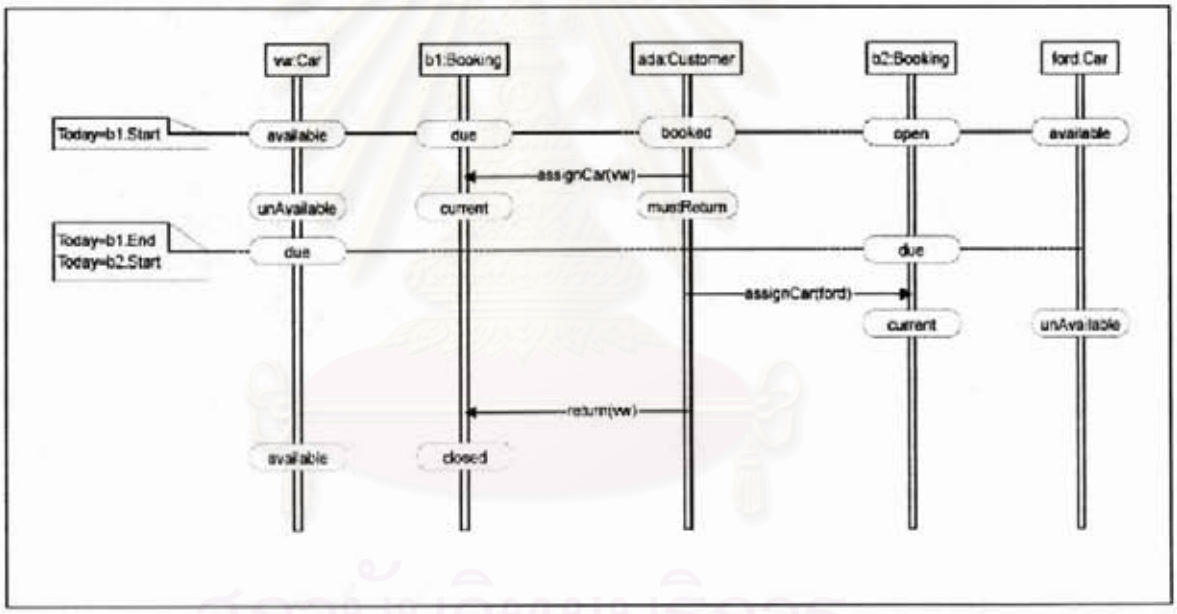

:User
create("Ada")
ada:Customer
noReturns
book("11/20","11/23",E)
create(ada,"11/20","11/23",E)
b:Booking
booked
open
Today=b.Start
due
Today>b.Start
cancel(b)
noReturns
closed


:User
create("Ada")
ada:Customer
noReturns
book("11/20","11/23",E)
create(ada,"11/20","11/23",E)
b:Booking
booked
open
cancel(b)
noReturns
closed
Today<b.Start

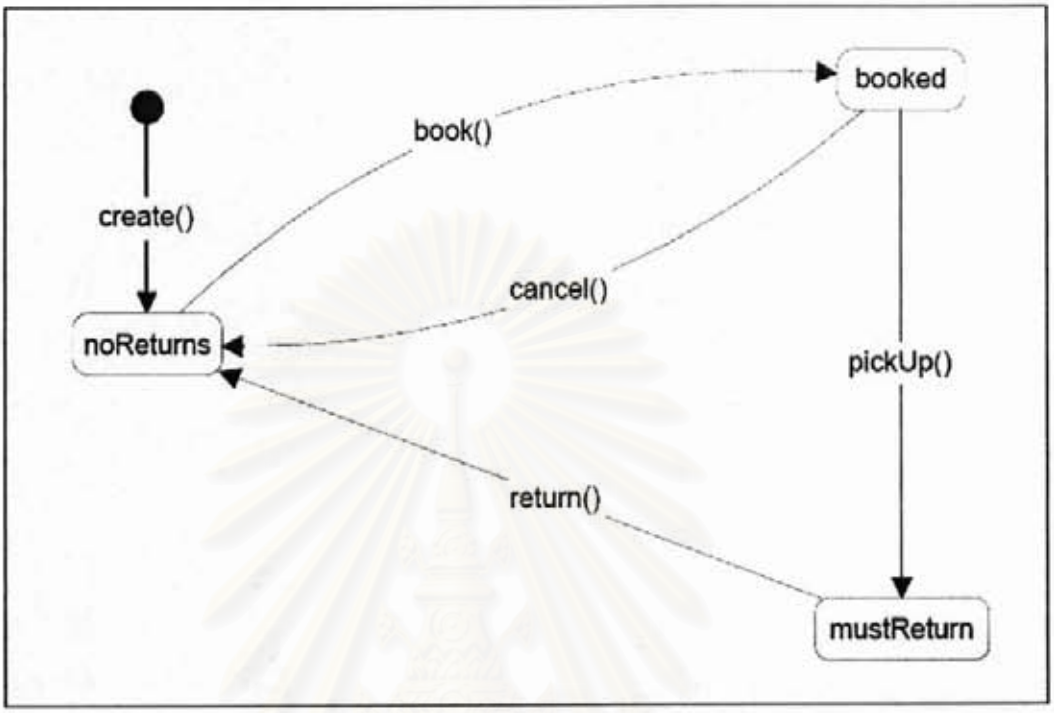

:User
ada:Customer
noReturns
vw:Car
available
b:Booking
open
booked
due
Today=b.Start
pickUp(b)
searchAvailCar()
vw
assignCar(vw)
unAvailable
current
mustReturn
return(vw)
returnCar()
noReturns
available
closed
Today=b.End

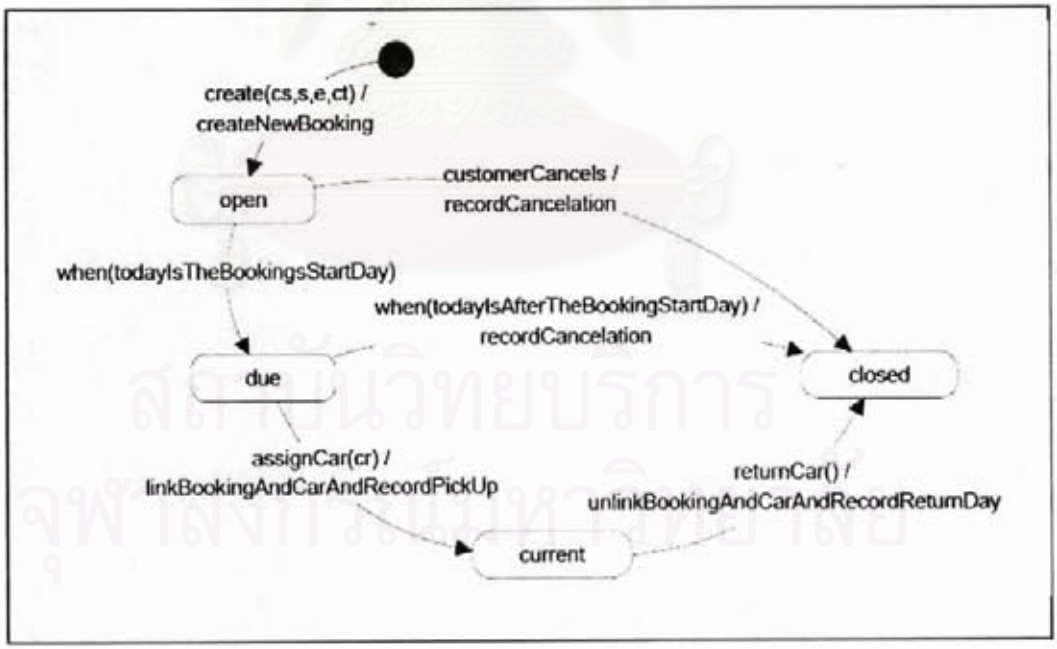

:User
ada:Customer
noReturns
vw:Car
available
b:Booking
open
booked
due
Today=b.Start
pickUp(b)
searchAvailCar()
vw
assignCar(vw)
unAvailable
current
mustReturn
Today=b.End
return(vw)
returnCar()
noReturns
available
closed

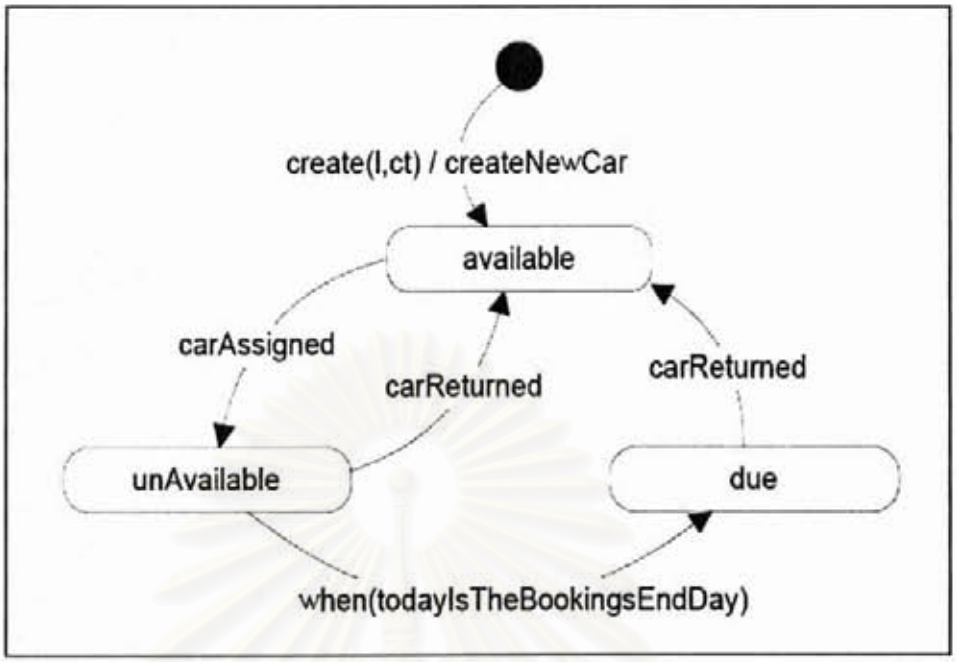




1.4 แผนภาพสถานะ







2. เอกสารแสดงการออกแบบซอฟต์แวร์ที่ปรับปรุงแล้ว

จากต้นฉบับเอกสารแสดงการออกแบบซอฟต์แวร์ข้างต้น จะเห็นได้ว่าประกอบด้วยแผนภาพหลายแผนภาพ ดังนั้นผู้วิจัยจึงได้ปรับปรุงเอกสารแสดงการออกแบบซอฟต์แวร์ให้เหมาะสมกับระยะเวลาที่กำหนดในการทดลอง เพื่อให้หน่วยทดลองสามารถค้นหาข้อบกพร่องได้ทันตามระยะเวลาที่กำหนด ซึ่งจะประกอบด้วยเอกสารแสดงความต้องการซอฟต์แวร์ แผนภาพยูสเคสและคำอธิบายยูสเคส แผนภาพคลาสและคำอธิบายคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ

2.1 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)

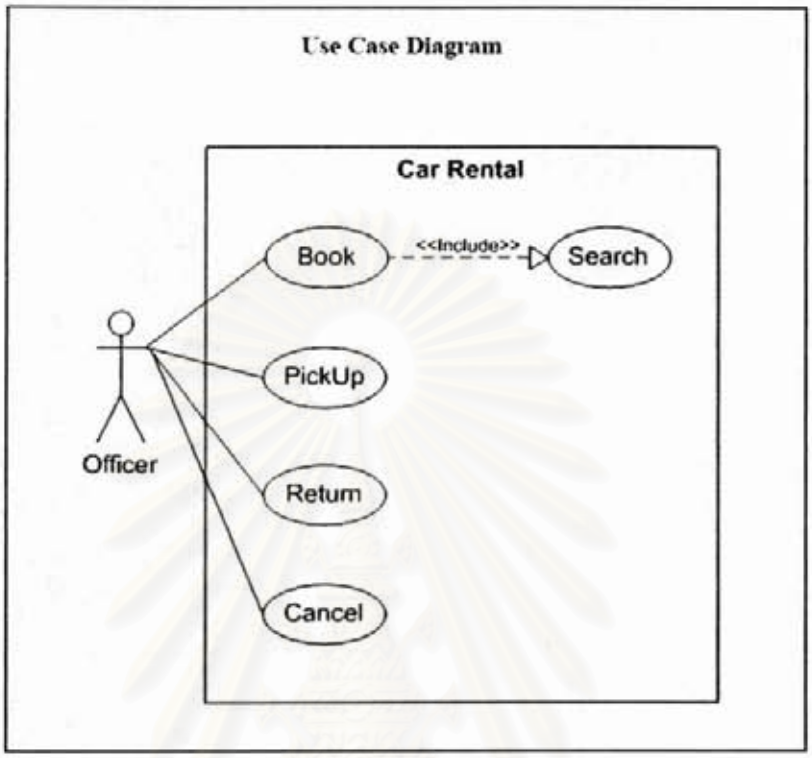
8. เมื่อลูกค้ายกเลิกรายการการเช่ารถ หรือนำรถที่เช่ามาคืนเรียบร้อยแล้ว Officer จะทำการลบข้อมูลของลูกค้านั้นทิ้ง

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

2.2 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)

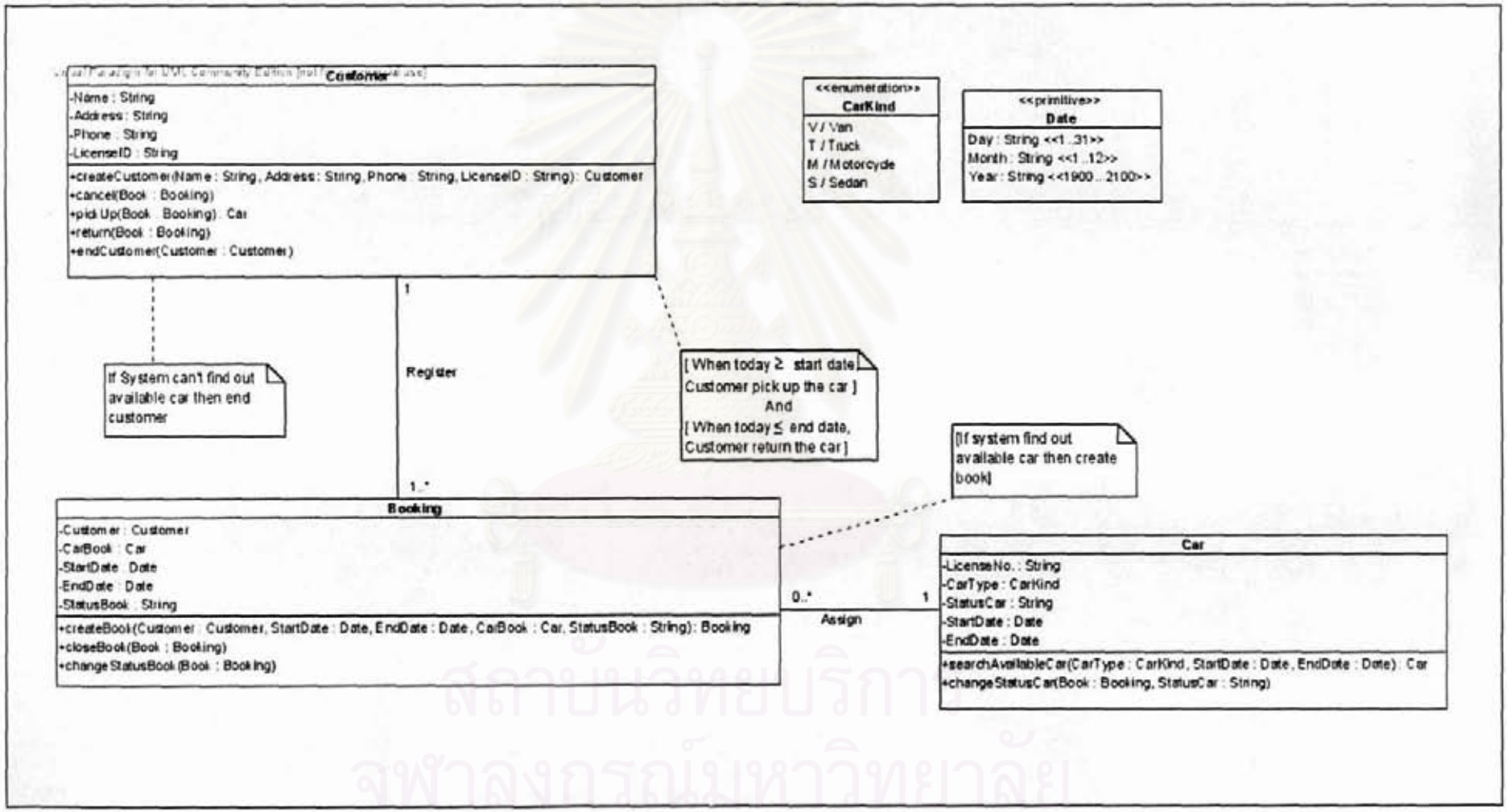
2. Search	
Goal	ค้นหารถที่มีสถานะว่างในวันที่ลูกค้าต้องการเช่า โดยค้นหาตามประเภทรถที่ลูกค้าต้องการ
Actor	Officer
Precondition	ลูกค้าแจ้งข้อมูลต่างๆที่ใช้ในการเช่ารถเรียบร้อยแล้ว
Postcondition	ได้รับผลการค้นหารถเรียบร้อยแล้ว
Main Success Scenario	1. Officer ค้นหารถที่ว่าง โดยค้นหาตามประเภทรถที่ลูกค้าต้องการ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

2.3 แผนภาพคลาสและคำอธิบายคลาส



Class Description

1. Customer				
Description	ลูกค้าที่ต้องการเช่ารถ			
Superclass	-			
Subclass	-			
Attributes				
Name	**Description**	**Data Type**	**Initial Value**	**Visibility**
Name	ชื่อลูกค้า	String	-	Private
Address	ที่อยู่ลูกค้า	String	-	Private
Phone	เบอร์โทรศัพท์ลูกค้า	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. Booking

Description	รายการการเช่ารถ
Superclass	-
Subclass	-

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. eloseBook			
Description	ฟังก์ชันที่ใช้ลบรายการการเช่ารถ		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Book	รายการการเช่าที่ต้องการลบ	Booking
Return Type	-		
3. changeStatusBook			
Description	ฟังก์ชันที่ใช้เปลี่ยนสถานะรายการการเช่ารถ		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Book	รายการการเช่าที่ต้องการเปลี่ยนสถานะ	Booking
Return Type	-		

3. 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
	StartDate	วันเริ่มต้นการเช่ารถ	Date
	EndDate	วันสิ้นสุดการเช่ารถ	Date
Return Type	Car		

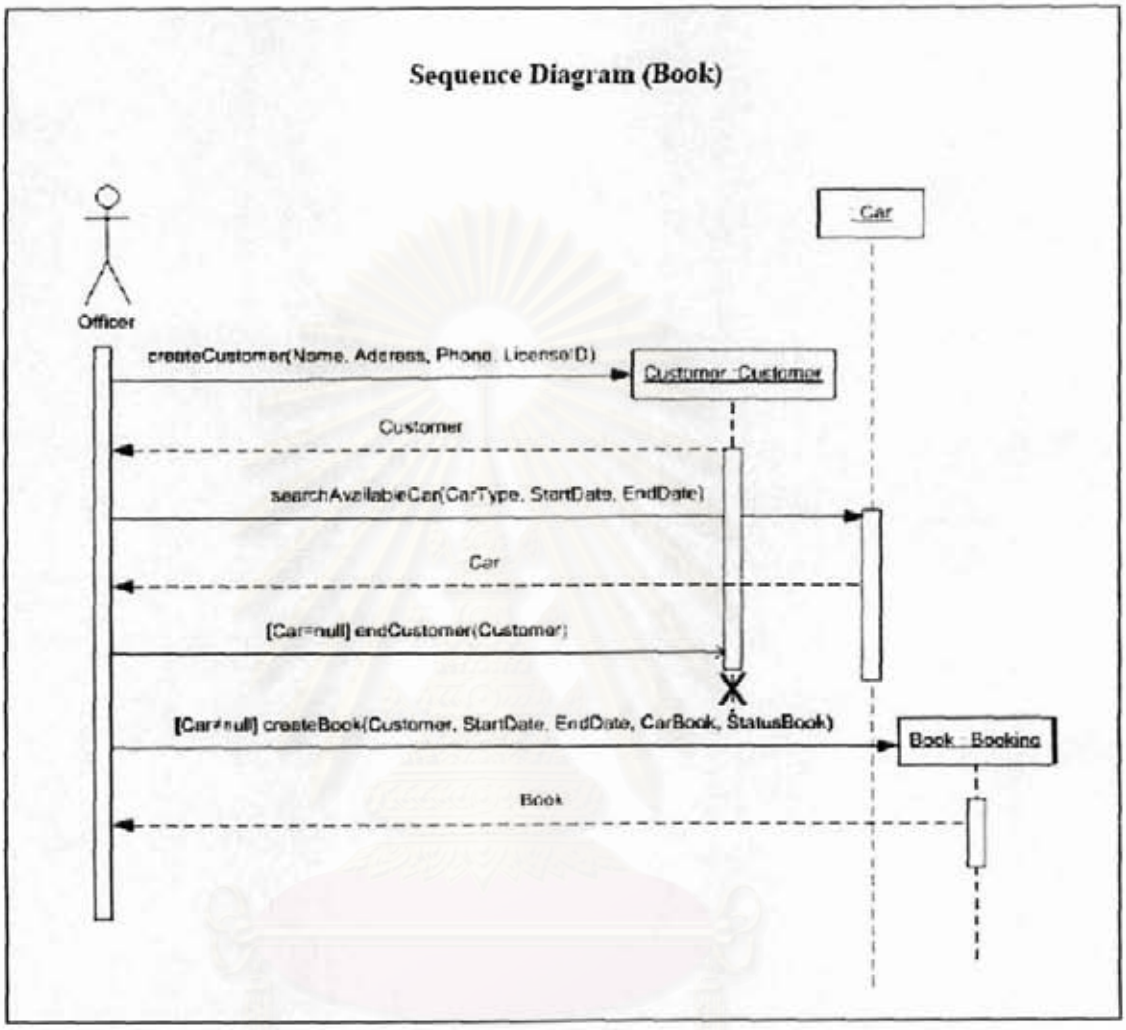
2. changeStatusCar			
Description	ฟังก์ชันที่ใช้เปลี่ยนสถานะของรถ ซึ่งแบ่งออกเป็น Available และ PickUp		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Book	รายการการเช่าที่ต้องการเปลี่ยนสถานะรถ	Booking
	StatusCar	สถานะรถ	String
Return Type	-		

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

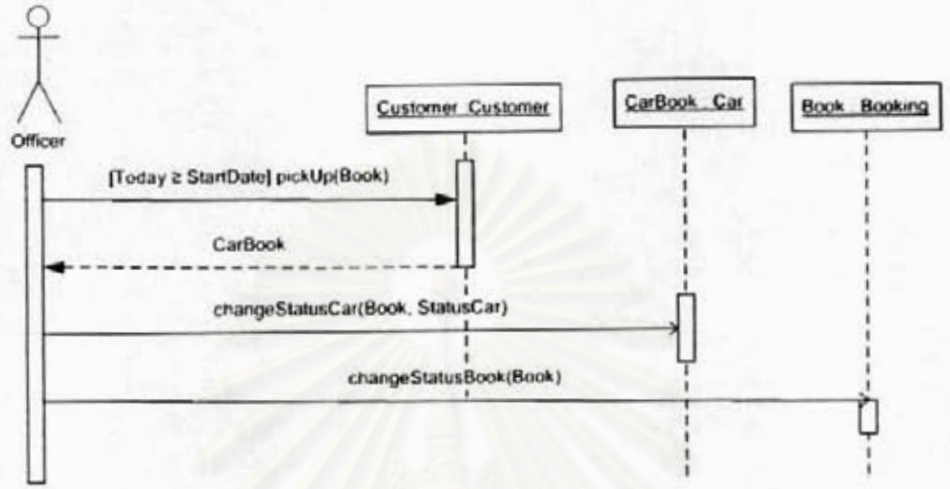
ข้อจำกัด :

1. สามารถทำรายการการเช่ารถได้ ก็ต่อเมื่อระบบสามารถค้นหารถตามที่ลูกค้าต้องการเช่าได้
2. ลูกค้าสามารถมารับรถได้ภายในระยะเวลาที่เช่า
3. ลูกค้าต้องนำรถมาคืนไม่เกินวันสุดท้ายของการเช่ารถ

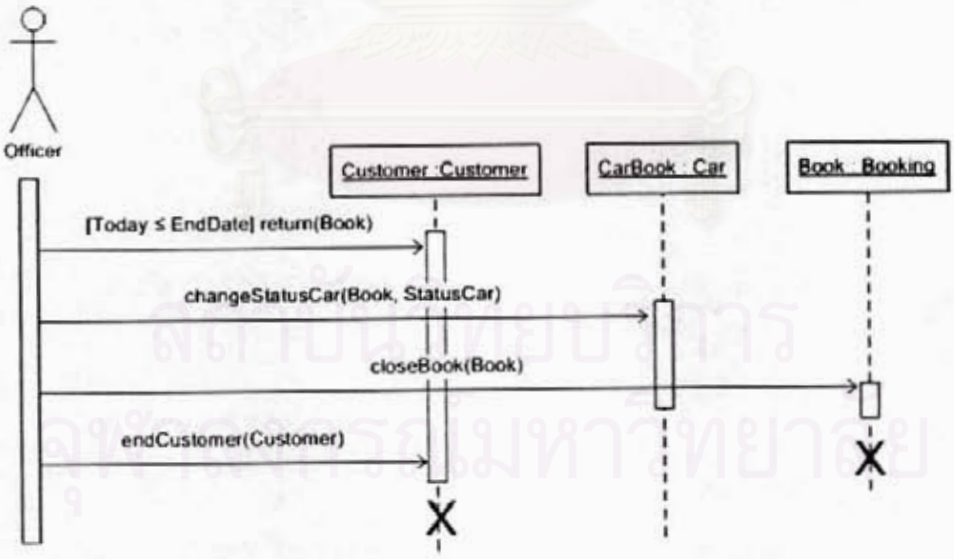
2.4 แผนภาพซีเควนซ์

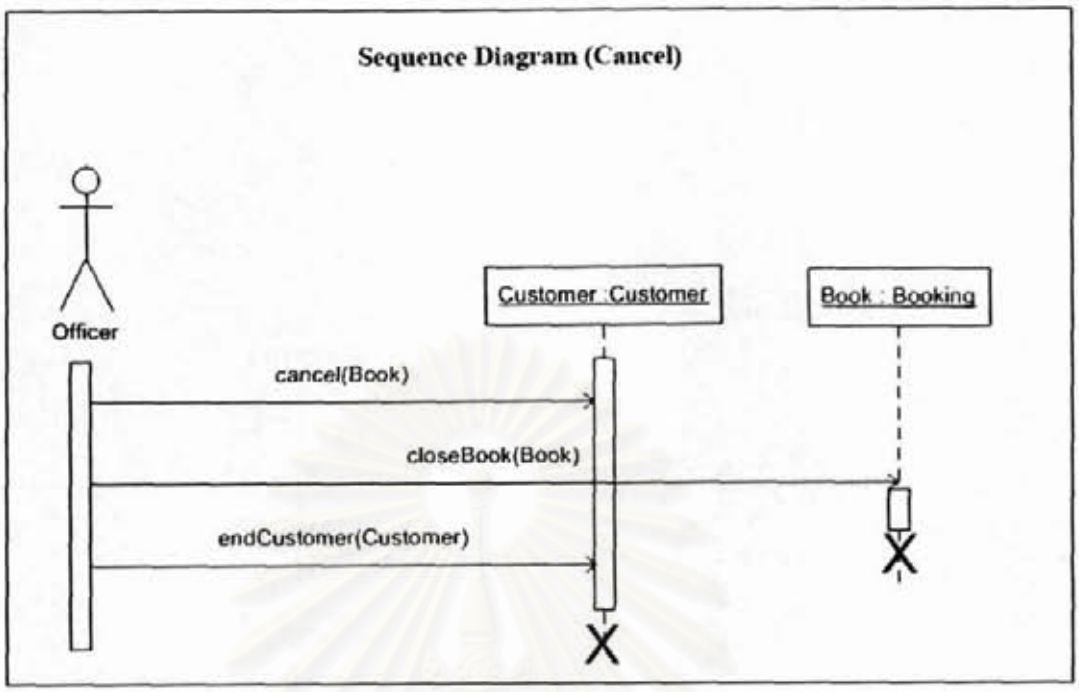


Sequence Diagram (Pick Up)

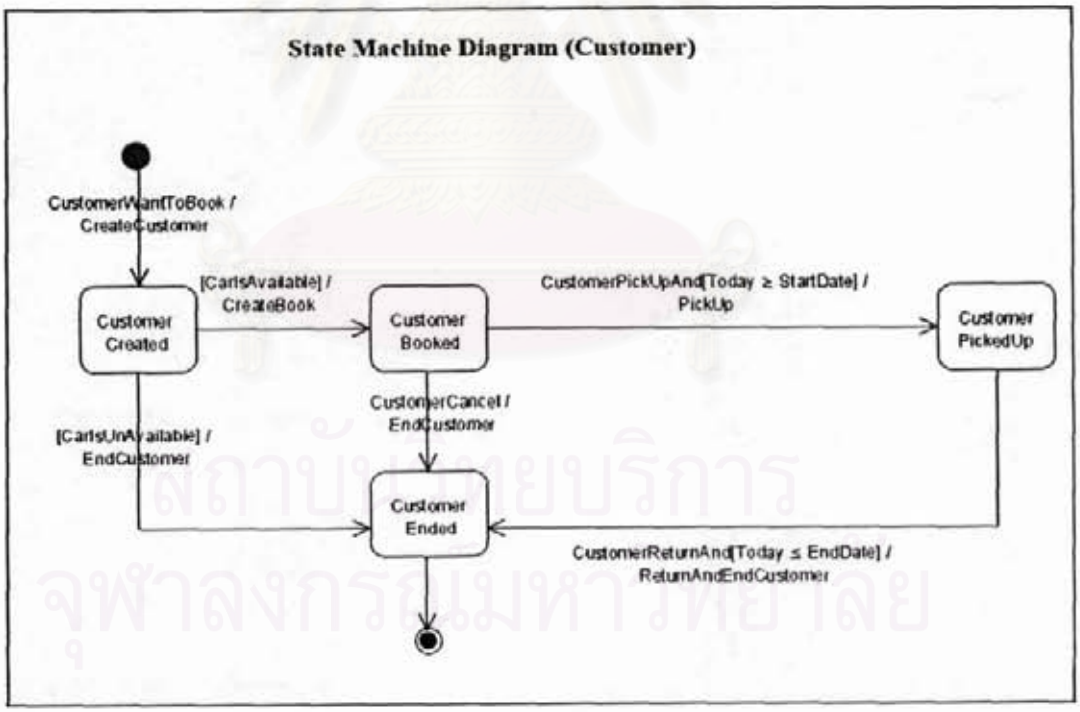


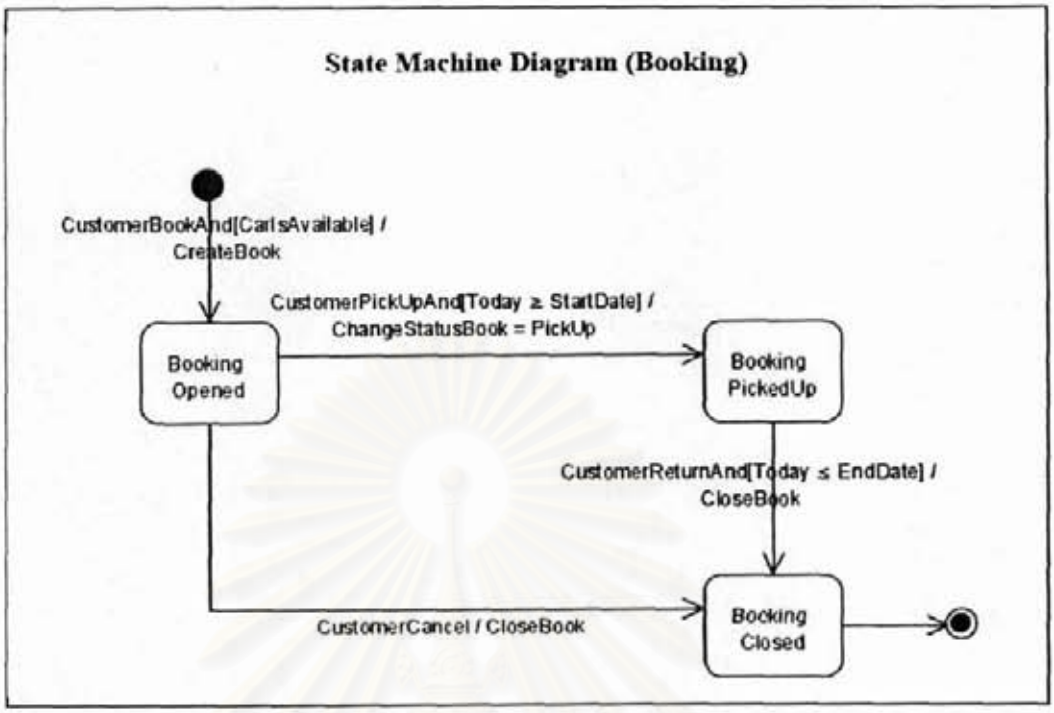
Sequence Diagram (Return)

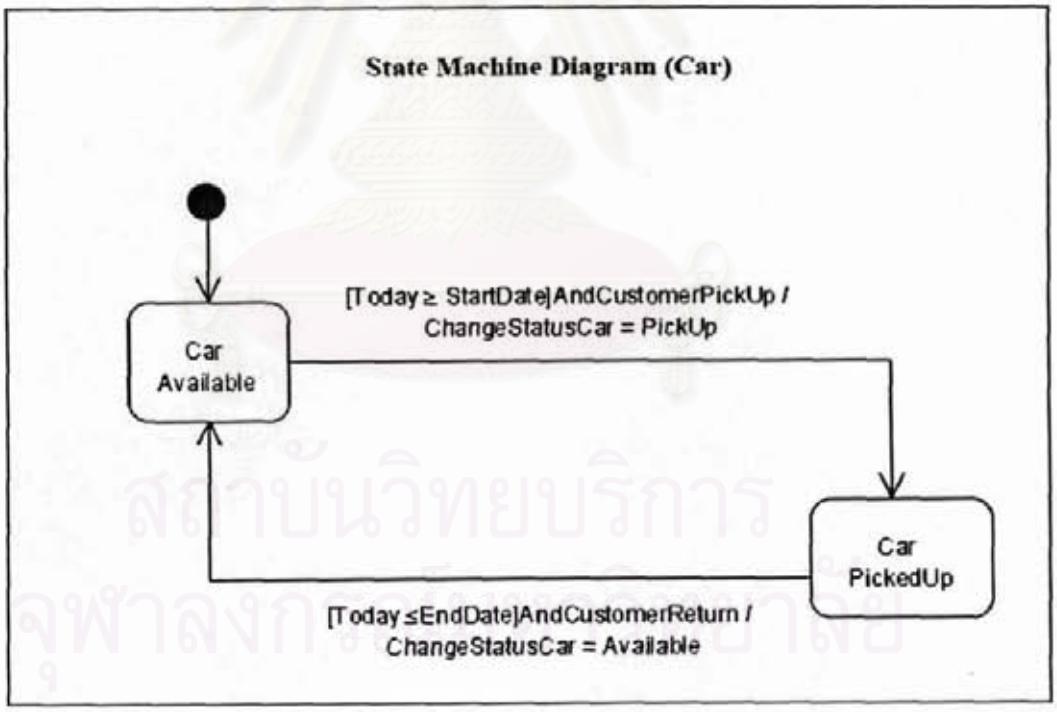




2.5 แผนภาพสถานะ



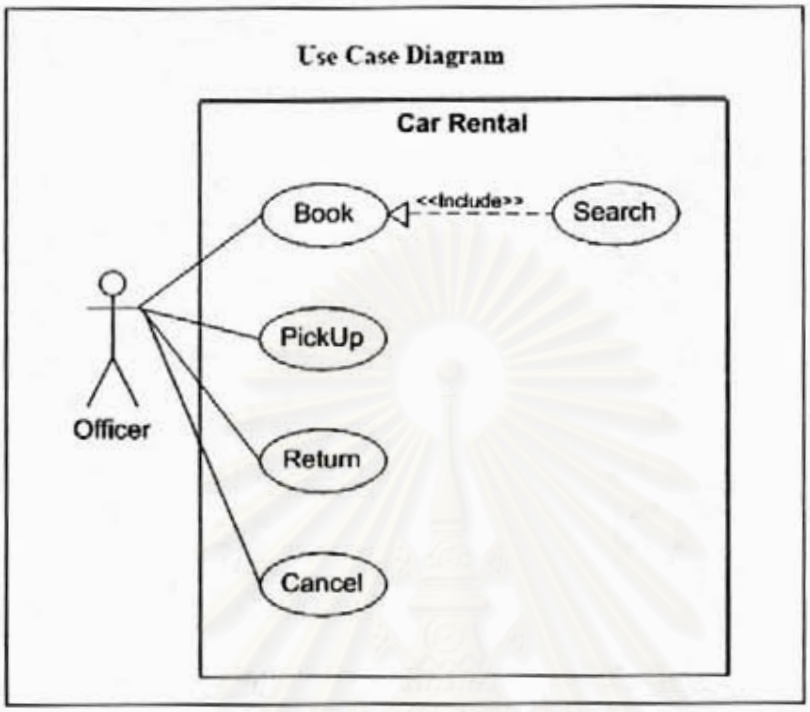




3. เอกสารแสดงการออกแบบซอฟต์แวร์ที่ใช้ในการทดลอง

หลังจากปรับปรุงเอกสารแสดงการออกแบบซอฟต์แวร์ ให้มีความถูกต้องและเหมาะสมแล้ว ผู้วิจัยได้ใส่ข้อบกพร่องลงไปในเอกสารแสดงการออกแบบซอฟต์แวร์ดังนี้

3.1 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)

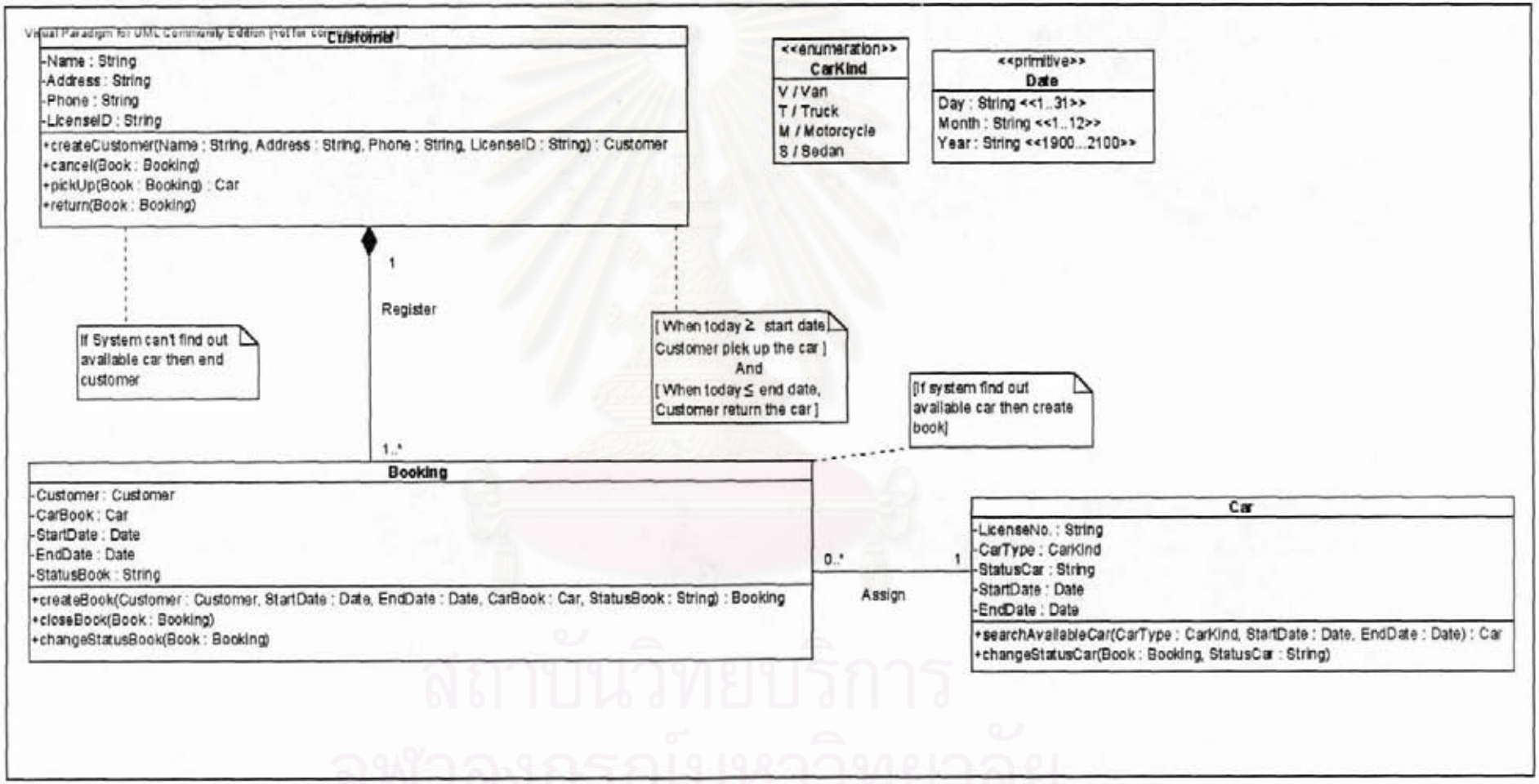
2. Search	
Goal	ค้นหารถที่มีสถานะว่างในวันที่ลูกค้าต้องการเช่า โดยค้นหาตามประเภทรถที่ลูกค้าต้องการ
Actor	Officer
Precondition	ลูกค้าแจ้งข้อมูลต่างๆที่ใช้ในการเช่ารถเรียบร้อยแล้ว
Postcondition	ได้รับผลการค้นหารถเรียบร้อยแล้ว
Main Success Scenario	1. Officer ค้นหารถที่ว่าง โดยค้นหาตามประเภทรถที่ลูกค้าต้องการ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

3.2 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	-			

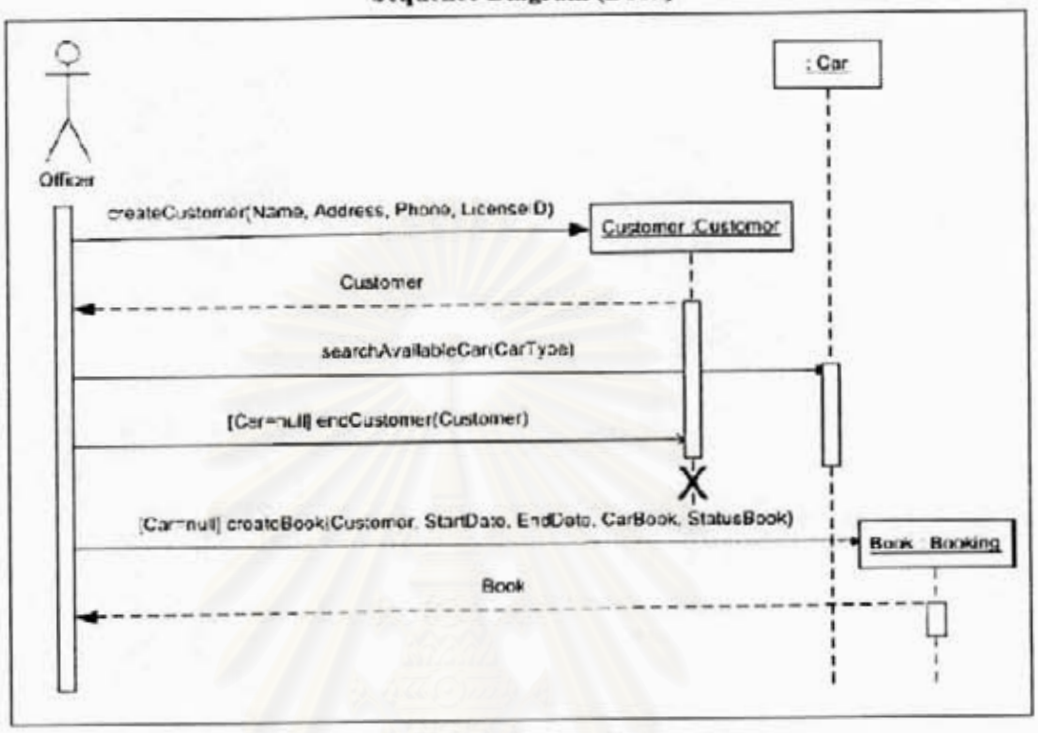
Association	
1. Register	
Description	ลูกค้าต้องการลงทะเบียนการเช่ารถ
Association Type	Composition
From - To	Customer – Booking
Cardinality	1 : 1..*
2. Assign	
Description	มอบหมายรถให้กับรายการการเช่า
Association Type	Association
From - To	Car – Booking
Cardinality	1 : 0..*
3. Search	
Description	ค้นหารถที่ลูกค้าต้องการ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

ข้อจำกัด :

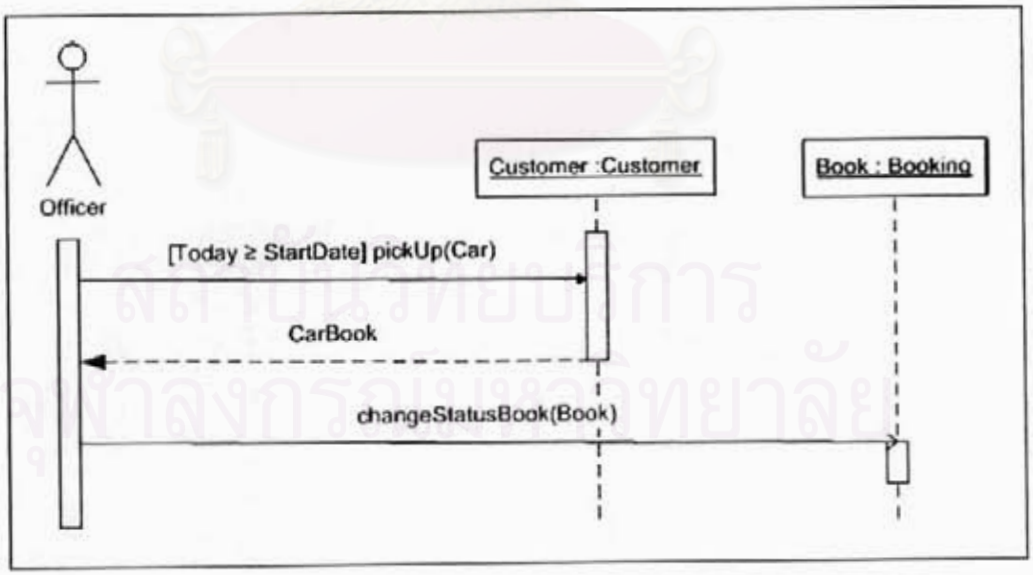
1. สามารถทำรายการการเช่ารถได้ ก็ต่อเมื่อระบบสามารถค้นหารถตามที่ลูกค้าต้องการเช่าได้
2. ลูกค้าสามารถมารับรถได้ภายในระยะเวลาที่เช่า

3.3 แผนภาพซีเควนซ์

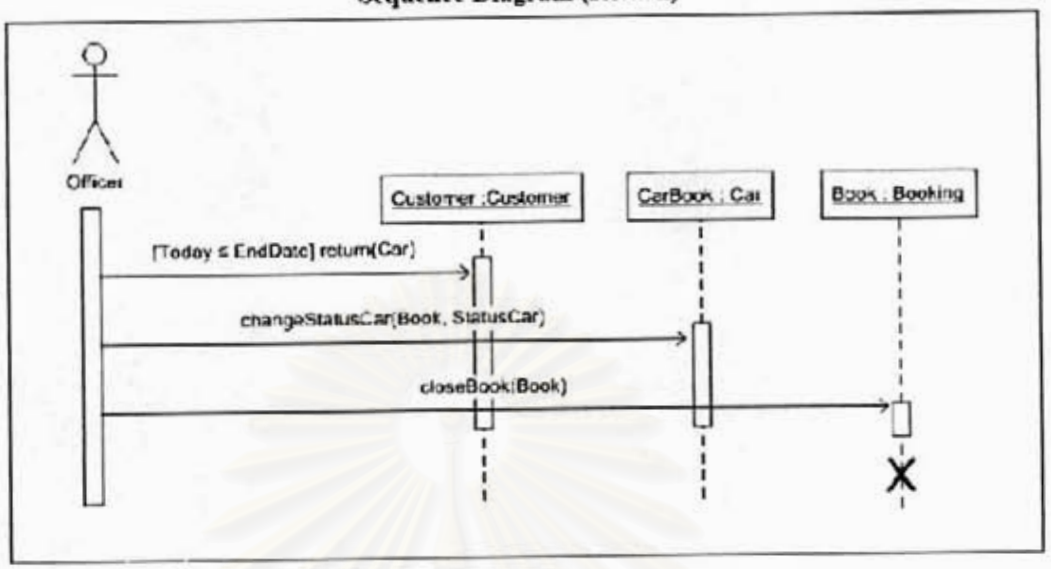
Sequence Diagram (Book)



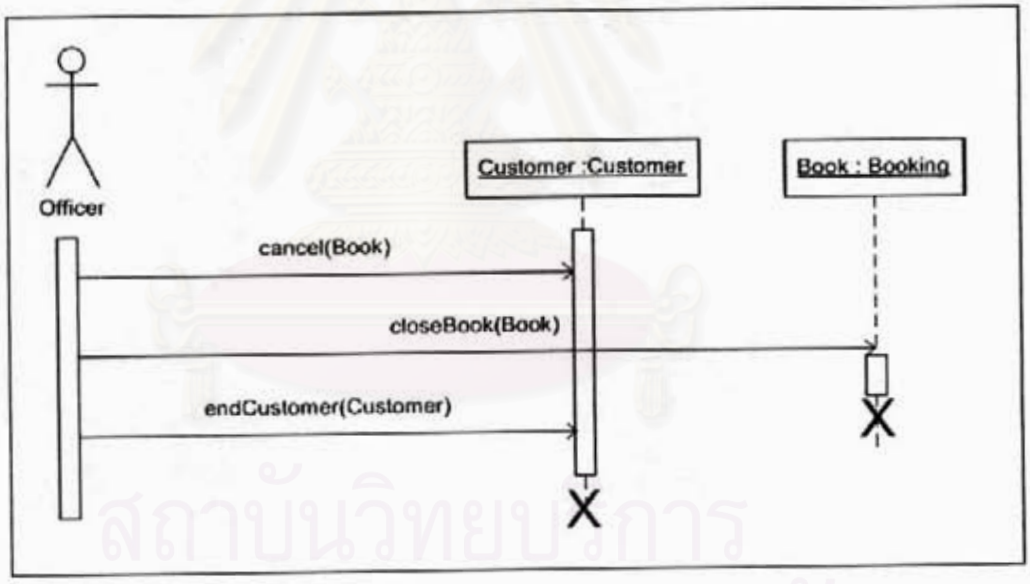
Sequence Diagram (Pick Up)



Sequence Diagram (Return)

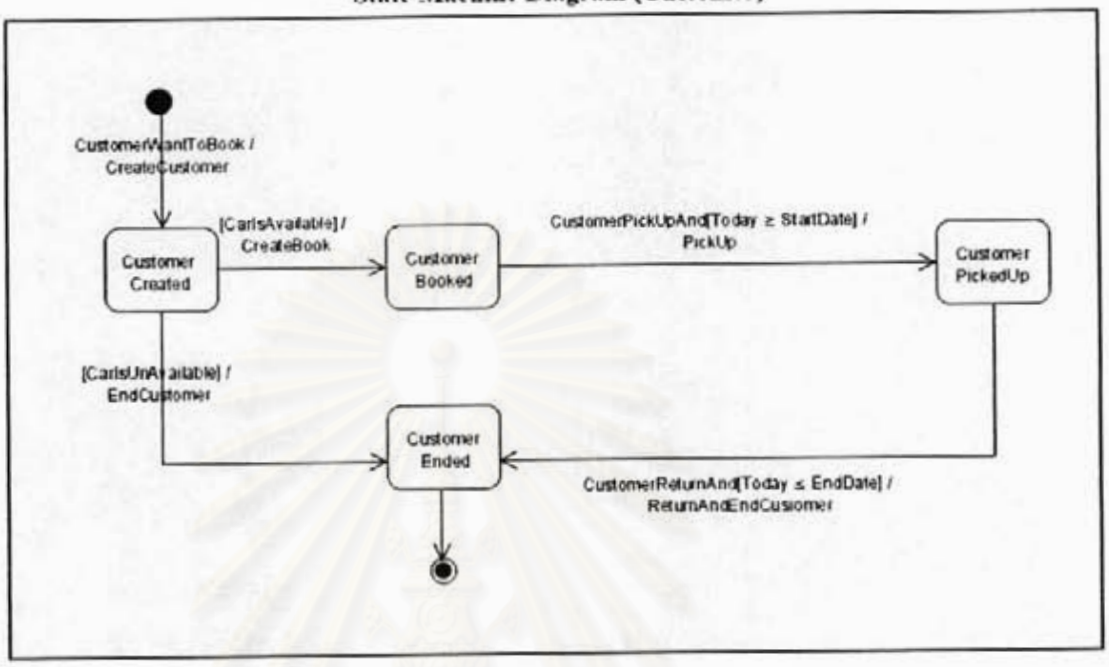


Sequence Diagram (Cancel)

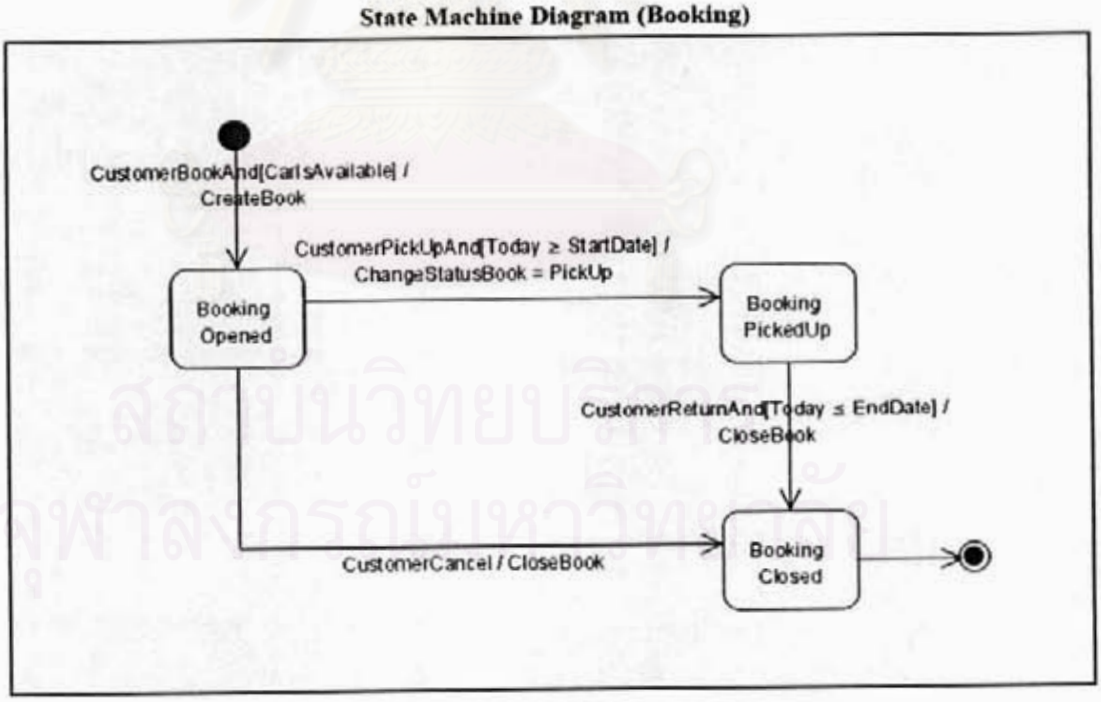


3.4 แผนภาพสถานะ

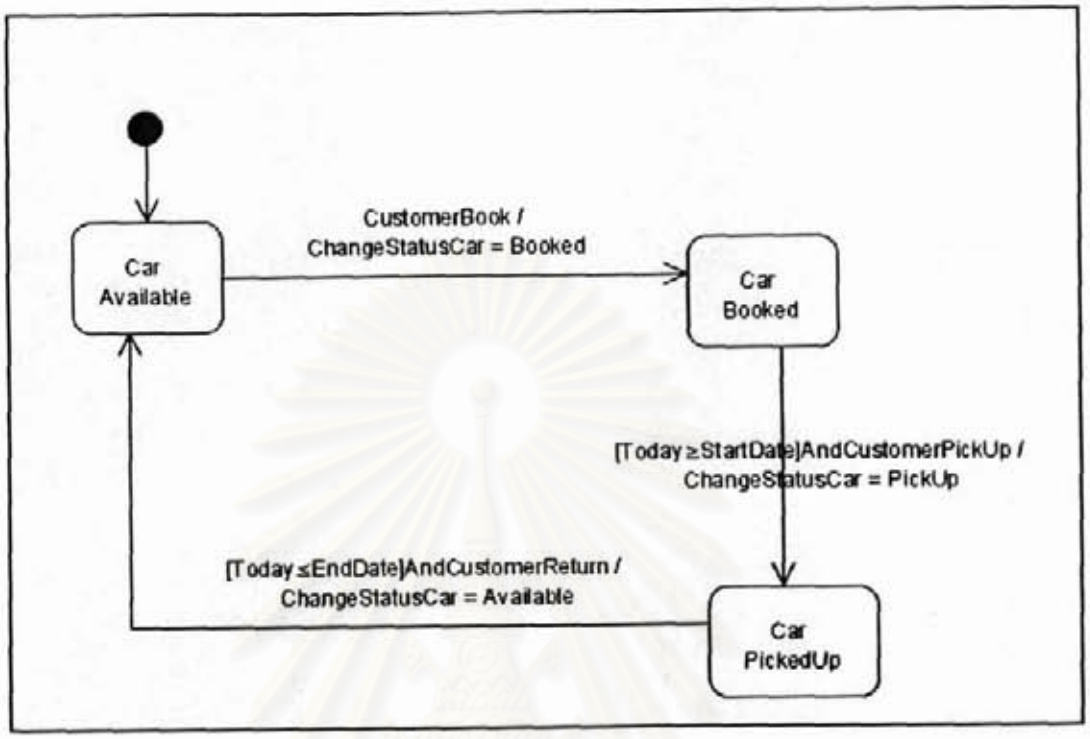
State Machine Diagram (Customer)



State Machine Diagram (Booking)



State Machine Diagram (Car)



ภาคผนวก ข

เอกสารคำแนะนำ

1. ต้นฉบับเอกสารสถานการณ์ (Scenario) เป็นเอกสารคำแนะนำสำหรับเทคนิคโอโออาร์ทีใช้ในการตรวจสอบเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุ นำมาจากงานวิจัยของ Travassos และคณะ (2002)

APPENDIX B.1 – OORT's 3.0 – Horizontal Reading – English Version

Reading 1 – Sequence x Class Diagrams

Goal: To verify that the class diagram for the system describes classes and their relationships in such a way that the behaviors specified in the sequence diagrams are correctly captured. To do this, you will first check that the classes and objects specified in the sequence diagram appear in the class diagram. Then you will check that the class diagram describes relationships, behaviors, and conditions that capture the dynamic services as described on the sequence diagram.

Inputs to process:

1. A class diagram (possibly divided into packages) that describes the classes of a system and how they are associated.
2. Sequence diagrams that describe the classes, objects, and possibly actors of a system and how they collaborate to capture services of the system.

I. **Take a sequence diagram and read it to understand the system services described and how the system should implement those services.**

INPUTS: Sequence diagram (SD).
OUTPUTS: System objects (marked in blue on SD);
Services of the system (marked in green on SD);
Conditions on the services (marked in yellow on SD).

A. For each sequence diagram, underline the system objects and classes, and any actors, with a blue pen.

B. Underline the information exchanged between objects (the horizontal arrows) with a green pen. Consider whether this information represents *messages* or *services* of the system. If the information exchanged is very detailed, at the level of messages, you should abstract several messages together to understand the services they work to provide. Example 2 provides an illustration of messages being abstracted into services. Annotate the sequence diagram by writing down these services, and underline them in green also.

C. Circle any of the following constraints on the messages and services with a yellow pen: restrictions on the number of classes/objects to which a message can be sent, restrictions on the global values of an attribute, dependencies between data, or time constraints that can affect the state of the object. Also circle any conditions that determine under what circumstances a message will be sent. The sequence diagram in Example 2 contains several examples of constraints and conditions on messages. The conditions concerning payment type and payment time determine when messages authorize_payment and new_payment_type_request will be sent , while the restrictions on response_time for message authorize_payment represent time constraints.

II. **Identify and inspect the related class diagrams, to identify if the corresponding system objects are described accurately.**

INPUTS: Sequence diagrams, with objects, services, and constraints marked;
Class diagrams.

OUTPUTS: Discrepancy reports.

A. Verify that every object, class, and actor used in the sequence diagram is represented by a concrete class in a class diagram. For classes and actors, simply find the name on the class diagram. For objects, find the name of the class from which the object is instantiated. Check for the following discrepancies and mark on the discrepancy report form :

1) **If a class or object cannot be found on the class diagram, it means that the information is inconsistent between both documents, it is present in one and absent in the other.**
2) **If an actor cannot be found, determine whether that actor needs to be represented as a class to perform the necessary behavior. If it does, then information that is present in the sequence diagram is missing from the class diagram.**

B. Verify that for every green-marked service or message on the sequence diagram, there is a corresponding behavior on the class diagram. Verify that there are class behaviors in the class diagram that encapsulate the higher-level services provided by the sequence diagram. To do this, make sure that the class or object that *receives* the message on the sequence diagram, or should be responsible for the service, possesses an associated behavior on the class diagram. Also make sure that there exists some kind of association (on the class diagram) between the two classes that the message connects (on the sequence diagram). Remember that in both cases, you may need to trace upwards through any inheritance trees in which the class belongs to find the necessary features. Finally, verify that for each service, the messages described by the sequence diagram are sufficient to achieve that service. Check for the following discrepancies, and mark on the **discrepancy report form**:

1) **Make sure that for each message on the sequence diagram the receiving class contains an appropriate behavior on the class diagram. If not, it means that there is an inconsistency between the diagrams. A behavior is present in the sequence diagram, but missing on the class diagram.**
2) **Make sure that there are appropriate behaviors for the system services? If not, there is a service present on the sequence diagram that is not represented on the class diagram.**
3) **Make sure there is an association on the class diagram between the two classes between which the message is sent. If not, an association is present in the sequence diagram, because of the message exchange, but not present in the class diagram.**
4) **Make sure that there are not any behaviors missing, which would prevent the service from being achieved. If there are, it means that something is missing from the sequence diagram.**

C. Verify that the constraints identified in the sequence diagram can be fulfilled according to the class diagram. Check for the following discrepancies, if any of the following statements are not true then information on the sequence diagram has not been represented in the class diagram. Mark this on the discrepancy report form.

1) **If the sequence diagram places restrictions on the number of objects that can receive a message, make sure that constraint appears as cardinality information for the appropriate association in the class diagram.**
2) **If the sequence diagram specifies a range of permissible values for data, make sure that constraint appears as a value range on an attribute in the class diagram.**
3) **If the sequence diagram contains information concerning the dependencies between data or objects (e.g. "a 'Bill' object cannot exist unless at least one 'Purchase' object exists") make sure this information is included on the class diagram. (It may be as a constraint on a class or relation on the class diagram or by cardinality constraints on relationships.)**
4) **If the sequence diagram contains timing constraints that could affect the state of an object (e.g. "if no input is received within 5 minutes then the window should be closed") make sure this information is included as a constraint on a class or relation on the class diagram? (For example, the class diagram in Example 3 contains a timing constraint for the class "Credit_Card_System" since it applies to all instantiations of this class. The conditional expressions from Example 2 should not appear in the class diagram because they do not affect the state of a class.)**

D. Finally, for each class, message, and data identified above, think about whether, *based on your previous experience*, it results in a reasonable design. For example, think about quality attributes of the design such as cohesion (do all the behaviors and attributes of a class really belong together?) and coupling (are the relations between classes appropriate?). Check for the following discrepancies:

1) **Make sure that it is logical for the class to receive this message with these data.**
2) **Make sure you can verify that the constraints are feasible.**
3) **Make sure all of the necessary attributes are defined. If not, the diagrams may contain incorrect facts.**
4) **For the classes specified in the sequence diagram, make sure the behaviors and attributes specified for them on the class diagram make sense.**
5) **Make sure the *name* of the class is appropriate for the domain, and for its attributes and behaviors.**
6) **Make sure the relationships with other classes are appropriate.**
7) **Make sure the relationships are of the right *type*. a(For example, has a composition relationship been used where an association makes sense?) If not, you have found an incorrect fact because something in the design contradicts your knowledge of the domain.**

Reading 2 -- State diagrams x Class description

Goal: To verify that the classes are defined in a way that can capture the functionality specified by the state diagrams.

Inputs to Process:

1. A set of class descriptions that lists the classes of a system along with their attributes and behaviors.
2. State diagrams that describes the internal states in which an object may exist, and the possible transitions between states.

For **each state diagram,** perform the following steps:

I. **Read the state diagram to understand the possible states of the object and the actions that trigger transitions between them.**

INPUTS: State diagram (SD).

OUTPUTS: Object States (marked in blue on SD);
Transition Actions (marked in green on SD);
Discrepancy reports.

A. Determine which class is being modeled by this state diagram.

1) **If you can't determine the class that is being modeled, then something has been omitted or is ambiguous. Indicate this on a discrepancy report form.**

B. Trace the sequence of states and the *transition actions* (system changes during the lifetime of the object, which trigger a transition from one state to another) through the state diagram. Begin at the start state (filled circle) and follow the transitions until you reach an end state (double circle). Make sure you have covered all transitions.

C. Underline the name of each state, as you come to it, with a blue pen.

D. Highlight transition actions (represented by arrows) as you come to them using a green pen. For example, the state diagram provided in Example 5 contains seven transition actions. The arrow leading from the state labeled "authorizing" back to itself represents an action that does not actually change the state of the object.

E. Think about the states and actions you have just identified, and how they fit together.

1) **Make sure that you can understand and describe what is going on with the object just by reading the state machine. If you cannot, then the state machine is ambiguous. Indicate this on the discrepancy report form.**

II. **Find the class or class hierarchy, attributes, and behaviors on the class description that correspond to the concepts on the state diagram.**

INPUTS: Class description (CD);
Object States (marked in blue on SD);
Transition Actions (marked in green on SD).

OUTPUTS: Relevant object attributes (marked in blue on CD);
Relevant object behaviors (marked in green on CD);
Discrepancy reports.

A. Use the class description to find the class or class hierarchy that corresponds to this state diagram.

1) **If you can't find the corresponding class fill out a discrepancy report form because you have found an inconsistency. The state machine describes a class that has not been described on the class description.**

B. Find how the responsible class encapsulates the blue-underlined states described on the state diagram. States may be encapsulated:

- 1 attribute explicitly. (An attribute exists whose possible values correspond to system states, e.g. attribute "mode" with possible values "on", "off".)
- 1 attribute implicitly. (An object is considered to be in a specific state depending on the value of some attribute, but the state is not recorded explicitly. E.g. if a>5 the object behaves one way, for other values of a another behavior is appropriate, but nothing explicitly records the current state.)
- a combination of attributes.
- class type. (E.g. subclasses "fixed rate loan" and "variable rate loan" can be considered states of parent class "loan".) Remember to check the corresponding class and all parents in its inheritance hierarchy.

Mark each blue-underlined state with a star (*) when it is found.

1) **If there are any unstarred states then something is missing from the class description. If you can determine from your semantic knowledge of the domain, that the extra state does not make sense, then indicate this on the discrepancy report form, otherwise just indicate that the two diagrams are inconsistent.**

C. For each green-highlighted transition action on the state diagram, verify that there are class behaviors capable of achieving that transition. Remember to look both in the currently selected class and any classes higher in the inheritance hierarchy.

(Keep in mind the following possible exceptions: 1) The transition depends on a global attribute, outside of the class hierarchy. 2) In instances of poor design, i.e. high coupling and public class attributes, behaviors in associated classes can modify the value of a variable in the class directly.)
If the transition action is an *event* (i.e. a transition occurs when something happens) look for a behavior or set of class behaviors that achieve that event.

If the transition action is a *constraint* (i.e. a transition occurs when some expression becomes true or false) look for behaviors that can change the value of the constraint expression. For example, note the constraints "[payment ok]" and "[payment not ok]" in example 5. These describe when the actions they describe can happen, based on the status of payment.

Check for the following discrepancies, and fill out a **discrepancy report form** if you find any:

1) **Make sure that all actions are encapsulated by the class description. If they are not, then something is represented in the state diagram, but not in the class description.**

2) **Make sure that all of the constraints are encapsulated by the class description. If they are not, then something is represented on the state diagram, but not in the class description.**

3) **Make sure all of the data need to verify a constraint is present in the class description. If it is not all there, then you have found information in the state diagram that is not in the class description.**

III. **Compare the class description to the state diagram to make sure that the class, as described, can capture the appropriate functionality.**

INPUTS: Object States (marked in blue on SD);
Transition Actions (marked in green on SD)

OUTPUTS: Discrepancy reports.

A. Consider the system functionality in which this class participates, as described by the class description, and the states in which it may exist, as described by the state diagram.

1) **Using your semantic knowledge of this class and the behaviors it should encapsulate, make sure that all states are described. If not, something is missing and the class as described cannot behave, as it should. Indicate this on a discrepancy report form.**

Reading 3 -- Sequence x State diagrams

Goal: To verify that every state transition for an object can be achieved by the messages sent and received by that object.

Inputs to Process:

1. Sequence diagrams that describe the classes, objects, and possibly actors of a system and how they collaborate to capture services of the system.
2. State diagrams that describe the internal states in which an object may exist, and the possible transitions between states.

For **each state diagram**, perform the following steps:

I. **Read the state diagram to understand the possible states of the object and the actions that trigger transitions between them.**

INPUTS: State diagram (SD).

OUTPUTS: Transition Actions (marked and labeled in green on SD);
Discrepancy reports.

A. Determine which class is being modeled by this state diagram.

1) ***If you can't determine the class that is being modeled, then something has been omitted or is ambiguous. Indicate this on a discrepancy report form.***

B. Trace the sequence of states and the *transition actions* (system changes during the lifetime of the object, which trigger a transition from one state to another) through the state diagram. Begin at the start state and follow the transitions until you reach the end state. Make sure you have covered all transitions.

C. Highlight transition actions (represented by arrows) as you come to them using a green pen. For example, the state diagram provided in Example 5 contains seven transition actions. The arrow leading from the state labeled "authorizing" back to itself represents an action that does not actually change the state of the object. Give each action a unique label [A1, A2, ...].

D. Think about the states and actions you have just identified, and how they fit together.

1) **Make sure that you can understand and describe what is going on with the object just by reading the state machine. If you cannot, then the state machine is ambiguous. Indicate this on the discrepancy report form.**

II. **Read the sequence diagrams to understand how the transition actions are achieved by messages that are sent and received by the relevant object.**

INPUTS: State diagram (SD);
Transition Actions (marked and labeled in green on SD);
Sequence diagrams (SqD).

OUTPUTS: Object messages (marked and labeled in green on SqD);
Discrepancy reports.

A. Take the sequence diagrams and choose the ones that use the object modeled by the state diagram; use only this subset of the sequence diagrams in the remainder of this step.
   1) **If there are no sequence diagrams that have this class in them, then fill out a discrepancy report because there is information in a state diagram that does not appear on the sequence diagrams.**

For each sequence diagram identified in the previous step:

B. Read the diagram to identify the system service being described and the messages that this object receives.
C. Think about which object states on the state diagram are *semantically* related to the system service. Highlight the state transitions leading to and from these states, and use this subset for the remainder of this step.
D. Map the object messages on the sequence diagram to the state transitions on the state diagram. Each transition action may map to one message, or a sequence of messages. To do this, you will need to think about the *semantics* behind the system messages. Are they contributing to achieving some larger system service or functionality? Do they have something to do with the types of states this object should be in? When you have made a mapping, mark the related messages and transition actions with a star (*). Label the messages with the same label given to their associated action on the state diagram.
   1) **Make sure, semantically, that you could do this mapping. If you cannot, then there are messages needed for a state transition that are not in the sequence diagram. Fill out a discrepancy report form, because information included in one diagram is not included in the other one.**
E. Look for constraints and conditions on the messages you just mapped to state transitions. An example constraint might be "t>0", that is, whether or not a message is sent depends on the value of some attribute t. Look to see that any constraints/conditions found are captured somehow on the state diagram. This information might be captured by: 1) state information (i.e. the fact that t>0 corresponds to a particular state of the system; 2) transition information (i.e. some state transition occurs when t>0 becomes true or false; 3) nothing (i.e. this information is not relevant or important for the state diagram). If any of the following occur, then fill out a **discrepancy report form**:
   1) **Make sure that you can find a correspondence between the conditions and constraints on the state and sequence diagrams. If not, then one diagram has information that is not on the other.**
   2) **For the information that appears on both diagrams, make sure that it is consistent. If it is not, then you have found the same information represented on two different diagrams in an inconsistent way.**

III. **Review the marked-up diagrams to make sure that all transition actions are accounted for.**

INPUTS: Transition Actions (marked and labeled in green on SD);
Object messages (marked and labeled in green on SqD);.

OUTPUTS: Discrepancy reports.

A. Review the state diagram looking for unstarred transition actions that could not be associated with object messages.
   1) **If the transition action was labeled with a constraint, see if you can find a message or sequence of messages capable of satisfying the constraint. If not, you have found information represented in one diagram but not in the other. The state diagram requires system services that are not described on any sequence diagram. Fill out a discrepancy report.**
   2) **If the transition action was labeled with an event, see if you can find a message, a sequence of messages, or some event performed by an actor that achieves the transition action. If not, you have found information represented in one diagram but not in the other. The state diagram requires system services that are not described on any sequence diagram. Fill out a discrepancy report.**

B. If the starred messages and transition actions identified in the previous step appear on the same sequence diagram, make sure they appear in a logical order. That is, suppose the messages that achieve action A1 appear before the messages that achieve action A2 on one sequence diagram. This means that A1 must take place chronologically before A2. Then you should make sure that A1 could be reached before A2 on the state diagram as well.
   1) **If the order does not match, the fill out a discrepancy report form, information is represented on two diagrams, but in an inconsistent way.**

Reading 4 -- Class diagrams x Class descriptions

Goal: To verify that the detailed descriptions of classes contain all the information necessary according to the class diagram, and that the description of classes make semantic sense.

Inputs to Process:

3. A class diagram (possibly divided into packages) that describes the classes of a system and how they are associated.
4. A set of class descriptions that lists the classes of a system along with their attributes and behaviors.

I. Read the class diagram to understand the necessary properties of the classes in the system.

INPUTS: Class diagram.
Class description.

OUTPUTS: Discrepancy reports.

For **each class on the class diagram**, perform the following steps:

A. Find the relevant class description. Mark the class on the class description with a blue symbol (*) when found.

1) **If you can't find the description, fill out a discrepancy report form, because a class present on the class diagram is not present in the class description.**

B. Check the name and textual description of the class to ensure that they provide a *meaningful* description of the class that you are considering at this time. Also check that the description is using an adequate abstraction level.

1) **Using your knowledge, make sure you can understand the purpose of this class from the high-level description. If not, the description may be too ambiguous to be used for the design model. Fill out a discrepancy report reporting a discrepancy because: *outside knowledge*.**

C. Verify that all the attributes are described along with basic types.

1) **Make sure that the same set of attributes is present in both the class description and the class diagram. If not, fill out a discrepancy report form because information is present in one document but not present in the other.**

2) **Make sure this class can *meaningfully* encapsulate all these attributes, that is, does it make sense to have these attributes in the class description, and that the basic types assigned to the attributes *feasible* according to the description of the attribute. If not, fill out a discrepancy report form indicating a discrepancy because: *outside knowledge*.**

D. Verify that all the behaviors and constraints are described.

1) **Make sure the same set of behaviors and constraints is present in both the class description and the class diagram, and that they use the same style or level of granularity (e.g. pseudocode) to describe the behaviors. If not, then information on one diagram is not present on the other, or it is inconsistent between the two.**

2) **Make sure this class can *meaningfully* encapsulate all these behaviors. Make sure the constraints make sense for this class. Make sure that behaviors can accomplish their tasks using the attributes that have been defined (for this or some other class). If not, fill in a discrepancy report indicating a discrepancy because: *outside knowledge*.**

3) **Make sure the constraints are satisfiable using the attributes and behaviors that have been defined. If not, you have found a situation where the behaviors and constraints as defined cannot be satisfied using the attributes and behaviors that have been defined. Indicate this on a discrepancy report form as a discrepancy because: *outside knowledge*. Describe the situation.**

4) **Make sure that the behaviors for this class do not rely *excessively* on the attributes of other classes to accomplish their functionality. (Note that you must make a value judgement about what is meant by "excessive reliance." You should compare the number of references to other classes for this class with the rest of the system, and consider the type of**

functionality addressed to determine if such reliance is really necessary.) If they do, then you have found a possibly poor design situation. Fill out a discrepancy report form indicating this situation.

E. If the class diagram specifies any inheritance mechanisms for this class, verify that they are correctly described.

1) **Make sure the inheritance relationship is included on the class description. If it is not, fill out a discrepancy report form. Information on the class diagram is not on the class description.**

2) **Use the class hierarchy to find the parents of this class. Make sure that, *semantically*, a <class name> is a type of <parent name>, and that it makes sense to have this class at this point of the hierarchy. If not, you have uncovered a potential style issue: the hierarchy should not be defined in this way. Fill in a discrepancy report describing the problem: *outside information*.**

F. Verify that all the class relationships (association, aggregation and composition) are correctly described with respect to multiplicity indications.

1) **Make sure that the object roles are captured on the class description, and that the correct graphical notation is used on the class diagram. If you find a problem, fill out a discrepancy report form indicating if information is omitted in one diagram, or if the notation is incorrect.**

2) ***Semantically*, make sure the relationships make sense given the role and the objects related. For example, if a composition relationship is involved, do the connected objects really seem like a "whole-part" relationship? If they don't make sense then you have uncovered a potential style issue: the relationships should not be defined in this way. Fill in a discrepancy report describing the problem: *outside information*.**

3) **If cardinalities are important, make sure they are described in the class description. Given your understanding of the relationship, make sure the quantities of objects used are enough. If not, fill in a discrepancy report because information in one diagram is not present in the other.**

4) **Make sure that there is some attribute representing the relationship. If not, fill in a discrepancy report indicating that information in one diagram is not present in the other.**

5) **Make sure that the relationship uses a feasible basic type or structure of basic types (if multiple cardinality is involved). If not, fill in a discrepancy report form indicating a discrepancy because: *outside information*.**

II. **Review the class descriptions for extraneous information.**

INPUTS: Class description.

OUTPUTS: Discrepancy reports.

A. Review the class descriptions to make sure that all classes described actually appear in the class diagram.

1) **Make sure there are no unstarred classes on the class description. If there are any, fill out a discrepancy report form because a class on the class description is not present on the class diagram.**

APPENDIX B.2 – OORTs 3.0 – Vertical Reading - English Version

Reading 5 -- Class Descriptions x Requirements Description

Goal: To verify that the concepts and services that are described by the functional requirements are captured appropriately by the class descriptions.

Inputs to Process:

1. A set of functional requirements that describes the concepts and services that is necessary in the final system.
2. A set of class descriptions that lists the classes of a system along with their attributes and behaviors.

I. **Read the requirements description to understand the functionality described.**

INPUTS: Set of functional requirements (FR).

OUTPUTS: Candidate classes/objects/attributes (marked in blue in FRs);
Candidate services (marked in green in FRs);
Constraints or conditions on services (marked in yellow in FRs).

A. Read over the each functional requirement to understand the functionality that it describes.

B. Find the nouns in the requirement; they are candidates to become classes, objects, or attributes in the system design. Underline the nouns with a blue pen.

C. Find the verbs, or descriptions of actions, which are candidates to be services or behaviors in the system. Underline the verbs or action descriptions with a green pen.

D. Look for descriptions of constraints or conditions on the nouns and verbs you identified in the preceding two steps. Especially pay attention to non-functional requirements, which typically contain restrictions and conditions on system functionality. For example, examine whether relationships between the concepts have been identified. Ask whether there are explicit constraints or limitations on the way actions are performed. Try to notice if definite quantities have been specified at any point in the requirement (see Example 4). Underline these conditions and constraints with a yellow pen.

II. **Compare the class descriptions to the requirements to verify if the requirements were captured appropriately.**

INPUTS: Set of functional requirements (FR);
Class description (CD).

OUTPUTS: Corresponding concepts have been marked on the FR and CD;
Discrepancy reports.

A. For each green-underlined action description in the functional requirements, try to find an associated behavior or combination of behaviors in the class description. Use syntactic clues (e.g. a behavior name that is similar or synonymous to an action description) to help your search, but make sure the *semantic* meaning of the function in the requirements and high-level design is the same. When found, mark both the name of the behavior(s) in the class description and the description of the activity in the requirements with a green symbol (*).

1) **Make sure the classes receive the right information for accomplishing the required behaviors. Make sure *feasible* results are produced. If not, the classes cannot implement the functionality appropriately. Indicate this on the discrepancy report form and mark whether it is because of *omitted functionality* or *incorrect* or *ambiguous information*.**

B. For each blue-underlined noun in the functional requirements, try to find an associated class in the class description. An associated class may be named after a concept from the requirements, may describe a general class of which the concept is a particular instance (i.e. an object), or may contain the concept as an attribute. Use syntactic clues (e.g. a class name that is similar to the name of a concept) to help your search, but make sure the *semantic* meaning of the concepts in the requirements and design is the same.

C. If the concept in the functional requirements corresponds to a class name in the class description, mark both the name of the class in the class description and the concept in the requirements description with a blue symbol (*).

1) **Make sure the class descriptions contain sufficient information regarding the concepts that play some role in this functionality and the class names have some connection to the nouns you had marked. If not, or if the classes are using ambiguous information to describe the concepts indicate this on the discrepancy report form.**

2) **Make sure these classes encapsulate (blue-marked) attributes concerned with the nouns you marked and (green-marked) behaviors concerned with the verbs or actions descriptions you had marked. Also make sure that all identified constraints and conditions for these classes regarding this requirement are described. If not, you have found important information from the requirements omitted from the design. Indicate this on the discrepancy report form.**

D. If the concept in the functional requirements corresponds to an attribute in the class description, mark both the name of the attribute in the class description and the concept in the requirements description with a blue symbol (*).

1) **Make sure the class description is using *feasible* types to represent information; given the requirements description and that the (yellow-underlined) constraints and conditions on the attributes were observed in their definition. If not, you have found incorrect information in the design. Indicate this on the discrepancy report form.**

III. **Review the class description and functional requirements to make sure that all appropriate concepts correspond between the documents.**

INPUTS: Set of functional requirements (FR);
Class description (CD).

OUTPUTS: Discrepancy reports.

A. Look for descriptions of functionality in the requirements that have been omitted from the design.

1) **Make sure that there are no unstarred nouns or verbs in the requirements. If there is one, make sure that it should have been included in the design, and was not there merely for clarification. If it should have been in the design, then information has been omitted from the design. Indicate this on the discrepancy report form.**

Reading 6 -- Sequence Diagrams x Use-cases

Goal: To verify that sequence diagrams describe an appropriate combination of objects and messages that work to capture the functionality described by the use case.

Inputs to process:

1. A use case that describes important concepts of the system (which may eventually be represented as objects, classes, or attributes) and the services it provides.
2. One or more sequence diagrams that describe the objects of a system and the services it provides. There may be multiple sequence diagrams for a given use case since a use case will typically describe multiple "execution paths" through the system functionality. The correct set of sequence diagrams for a use case must be selected by using traceability information, or by someone with semantic knowledge about the system. Finding the correct set of sequence diagrams without traceability information or knowledge of the system will be hard.
3. The class descriptions of all classes in the sequence diagram.

I. **Identify the functionality described by a use case, and important concepts of the system that are necessary to achieve that functionality.**

INPUTS: Use case (UC)

OUTPUTS: System concepts (marked in blue on UC).
Services provided by system (marked in green on UC).
Data necessary for achieving services (marked in yellow on UC).

A. Read over the use case to understand the functionality that it describes.

B. Find the nouns included in the use case; they describe concepts of the system. Underline and number each unique noun with a blue pen as it is found. (That is, if a particular noun appears several times, label the noun with the same number each time.)

C. For each noun identify the verbs that describe actions applied *to* or *by* the nouns. Underline the identified services and number them (in the order they must be performed) with a green pen. Look for the constraints and conditions that are necessary in order for this set of actions to be performed. As an example, consider Example 1, in which constraints and conditions have been highlighted. In this use case, there is an example of both a constraint ("The Customer can only wait for 30 seconds for the authorization process") and a condition ("time of payment is the same as the purchase time").

D. Also identify any information or data that is required to be sent or received in order to perform the actions. Label the data in yellow as "Di,j" where subscripts i and j are the numbers given to the nouns between which the information is exchanged.

II. **Identify and inspect the related sequence diagrams, to identify if the corresponding functionality is described accurately and whether behaviors and data are represented in the right order.**

INPUTS: Use case, with concepts, services, and data marked.
Sequence diagram (SD).

OUTPUTS: System concepts (marked in blue on SD);
Services provided by system (marked in green on SD);
Data exchanged between objects (marked in yellow on SD).

A. For each sequence diagram, underline the system objects with a blue pen. Number them with the corresponding number from the use case.

B. Identify the *services* described by the sequence diagrams. To do this, you will need to examine the information exchanged between objects and classes on the sequence diagrams (the horizontal arrows). If the information exchanged is very detailed, at the level of messages, you may need to abstract several messages together to understand the services they work to provide. Underline the identified services and number them (in the order they occur in the diagram) with a green pen. Look for the condition that activates the actions.

C. Identify the information (or data) that is exchanged between system classes. Label the data in yellow as "Di,j" where subscripts i and j are the numbers given to the objects between which the information is exchanged.

III. **Compare the marked-up diagrams to determine whether they represent the same domain concepts.**

INPUTS: Use case, with concepts, services, and data marked;
Sequence diagram, with objects, services, and data marked.

OUTPUTS: Discrepancy reports.

A. For each of the blue-marked nouns on the use case, search the sequence diagram to see if the same noun is represented. Mark the noun on the use case and the sequence diagram with a blue star (*) if it can be found on the sequence diagram.

1) **If there are any unstarred nouns on the use case, it means that a concept was used to describe functionality on the use case but it was not represented on the sequence diagram. For each of the nouns on the sequence diagram, find the corresponding class on the class description and check whether the unstarred noun is an attribute. If the unstarred noun does not appear as an attribute of any of these classes, you have found an omission.(Is this correct? We are referring to the Class Description here, but that is not part of this technique) A concept was described on the use case but has not appeared in the system design. Fill in a discrepancy report because necessary functionality has been omitted.**

2) **If there are any unstarred nouns on the sequence diagram you have found an extraneous noun, or a noun describing a lower-level concept, on the sequence diagram. Think about whether the concept is necessary for the high-level design, and whether it represents a level of detail that is appropriate at this time. If it does not, fill in a discrepancy report because this information is extraneous.**

B. Identify the services described by the sequence diagram, and compare them with the description used on the use case. Are the classes/objects exchanging messages in the same order specified on the use case? Were the data that appear on messages on the sequence diagram correctly described on the use case? Is it possible for you to understand the expected functionality just by reading the sequence diagram?
   1) **Make sure that the classes exchange messages in the same specified order. If not think about whether this represents a defect. Usually, switching the order of messages may have an effect on the functionality. But sometimes messages can be switched without affecting the outcome; other times, messages can be performed in parallel, or conditions may ensure that only one or the other message is executed anyway. If changing the order will change the functionality, fill in a discrepancy report because the information on the design is incorrect.**
   2) **Make sure that the data exchanged are all in the correct message and that the messages go between the correct classes (i.e. do the labels "Di,j" for the data match between diagrams). Make sure the messages make sense for the objects sending and receiving them, and for achieving the relevant services. If not, it means that the sequence diagram is using information incorrectly. Fill in a discrepancy report describing the problem.**

C. Are all the constraints and conditions from the use case being observed in this sequence diagram? Is some detail from the use case missing here?
   1) **Make sure that the constraints are observed. Make sure all of the behavior and data on the sequence diagram are directly concerned with the use-case. If not, it means that the sequence diagram is using information incorrectly. Fill in a discrepancy report describing the problem.**

Reading 7 – State Diagrams x Requirements Description and Use-cases

Goal: To verify that the state diagrams describe appropriate states of objects and events that trigger state changes as described by the requirements and use cases.

Inputs to process:
1. The set of all state diagrams, each of which describes an object in the system.
2. A set of functional requirements that describes the concepts and services that is necessary in the final system.
3. The set of use cases that describe the important concepts of the system

For **each state diagram**, do the following steps:

I. **Read the state diagram to basically understand the object it is modeling.**

II. **Read the requirements description to determine the possible states of the object, which states are adjacent to each other, and events that cause the state changes.**

INPUTS: State Diagrams (SD)
Requirements Description (RD)

OUTPUTS: Object States (marked in blue on SD)
Adjacency Matrix

A. Put away the state diagram and erase any (*) from that are in the requirements from previous iterations of this step. Now, read through the requirements looking for places where the concept is described or for any functional requirements in which the concept participates or is affected. When you locate one of these, mark it in pencil with a (*) so that it will be easier to use for the remainder of the step. Focus on these parts of the RD for the rest of the step.
B. Locate descriptions of all of the different states that this object can be in. To locate a state, look for attribute vales or combinations of attribute values that can cause the object to behave in a different way. When you locate a state underline it with a blue pen and give it a number.
C. Now identify which one of the numbered states is the Initial state. Using a blue pen, mark it with an "I". Likewise mark the end state with an "E".
D. When you have found all of the states, on a separate sheet of paper, create a matrix with 1..N across the top and 1..N down the left side, where 1..N represents the numbers that you gave to the states in the previous step.
E. For each pair of states, if the object can change from the state represented by the number on the left hand side to the state represented by the number on the top row, then mark the box at the intersection of the row and column. If you can determine the event(s) that cause the state change put that in the box, if not just put a check mark (the event will be determined in a later step). If you can determine that it is not possible for the transition to happen then place an X in the box. If you cannot make a definite determination then leave the box blank for now.
F. For any event that you have identified above, if there are any constraints described in the requirements, then write those by the event in the matrix.

III. **Read the Use cases and determine the events that can cause state changes.**

INPUT: Use Cases

OUTPUT: Completed Adjacency Matrix

A. Read through the use cases and find the ones in which the object participates. Focus on these for the rest of the step.

B. For each box in the adjacency matrix that has a check mark in it, look through the use cases and determine what event(s) can cause that transition. These events may not be obvious and may require you to abstract the use-cases and think about what is actually going on with each object. Erase the check mark and write this event(s) in its place.

C. For each box that is blank in the adjacency matrix, see if any event that can cause that transition is described in the use cases. If it is, then write that event in the box; if not then place an X in the box.

IV. **Read the state diagram to determine if the states described are consistent with the requirements and if the transitions are consistent with the requirements and use cases.**

INPUT: Requirements Description;

State Diagram (SD);

Adjacency Matrix (AM).

OUTPUT: Discrepancy Reports

A. For each state that is marked and numbered in the requirements description, find the corresponding state on the state diagram and using a blue pen, mark it with the same number used in the requirements. Be careful, because the same state may have a different name in the requirements than it has on the state diagram. To determine if two different names are talking about the same state, you must use your understanding of the requirement's description of the state and the information contained in the state diagram. This may be an iterative process where if states appear to be missing, you must go back and look again at what you have identified and make sure that it is correct. If you find any problems, fill out a discrepancy report.

   1) **Make sure you can find all of the states. If a state is missing, look to see if two or more states that you marked in the requirements were combined into one state on the state diagram. If not, then information has been omitted from the design. If so, then make sure this combination makes sense. If it does not, then the design has incorrect information in it.**

   2) **Make sure that there are no extra states in the state diagram. Look to see if one state that you marked in the requirements has been split into two or more states in the state diagram. If not, then information in the design is extraneous. If so, make sure that this split makes sense. If it does not then the design has incorrect information.**

B. Once you have all of the states labeled with numbers, using the AM, compare the transition events marked on the matrix to the ones on the SD. For any box on the AM that is marked with an event, check the corresponding states on the SD to make sure they have an event to transition between them, and check to ensure that the event is the same.

   1) **Make sure all of the events on the AM appear on the SD. If not, information has been omitted from the design. If there are extra events on the state diagram, then the design has extraneous information in it.**

C. For each constraint that was marked on the AM, find it on the SD.

   1) **Make sure you can find all of the constraints that are on the AM. If you cannot, then information has been omitted from the design. If there are extra constraints on the state diagram, then the design has extraneous information.**

2. เอกสารสถานการณ์ (Scenario) ที่ใช้ในการทดลอง เป็นเอกสารสถานการณ์ (Scenario) ที่ผู้วิจัยปรับปรุงขึ้นมาเพื่อใช้ในการทดลอง

Reading 1 : Sequence Diagram x Class Diagram			
วัตถุประสงค์	เพื่อตรวจสอบว่า Class และความสัมพันธ์ใน Class Diagram ถูกนำเสนอได้ตรงตามที่ถูกระบุใน Sequence Diagram		
ขั้นตอน	แผนภาพ	สิ่งที่ต้องทำ	ข้อบกพร่อง
1	Sequence Diagram	- ขีดเส้นใต้ที่ Object หรือ Class ด้วยสีน้ำเงิน และที่ Message ด้วยสีเขียว - วงกลม Constraint และ Condition ของ Message ด้วยสีเหลือง	
2	Sequence Diagram → Class Diagram	ตรวจสอบทุกๆ 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	ตรวจสอบทุกๆ 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 ดังนั้นให้ตรวจสอบว่าใน Class Diagram มีความสัมพันธ์ (Association) เชื่อมระหว่าง Class คู่นั้นหรือไม่	Class ที่มีความสัมพันธ์กันใน Sequence Diagram แต่ใน Class Diagram ไม่แสดงความสัมพันธ์ (Association) เชื่อมระหว่าง Class คู่นั้น
5	Sequence Diagram → Class Diagram	ถ้าใน Sequence Diagram มีการแสดง Constraint หรือ Condition (เหลือง) ให้ตรวจสอบว่า Class Diagram แสดงไว้ด้วยหรือไม่	มี Constraint หรือ Condition (เหลือง) ใน Sequence Diagram แต่ไม่ได้แสดงไว้ใน Class Diagram
6	Class Diagram	ตรวจสอบชื่อ Class, Attribute และ Method ว่าสื่อความหมายได้ชัดเจนหรือไม่	ชื่อ Class, Attribute และ Method สื่อความหมายไม่ชัดเจน
7	Class Diagram	ตรวจสอบประเภทของความสัมพันธ์ระหว่าง 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	- ขีดเส้นใต้ที่ชื่อ State ด้วยสีน้ำเงิน - วงกลมที่ 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	ตรวจสอบว่าทุกๆ 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	ถ้ามีการกำหนดค่าที่เป็นไปได้ของ Attribute ใน Class Description ให้ตรวจสอบว่าตรงกับที่แสดงใน 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	เขียนอักษรกำกับแต่ละ 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) เมื่อพบแล้วให้ทำคอกอัน 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 ที่ทำคอกจันไว้ในขั้นตอนก่อนหน้าปรากฏใน Sequence Diagram เดียวกันแล้ว ให้ตรวจสอบว่า 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 ให้ตรวจสอบว่าใน Class Description ได้อธิบายไว้อย่างถูกต้อง และตรวจสอบชื่อของ 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	ตรวจสอบว่าทุก 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)* - วงกลมด้วยสีเหลืองที่ Constraint และ Condition	
2	Requirement Description → Class Description	ค้นหา Method ใน Class Description ที่สอดคล้องกับคำกริยา (เขียว) ใน Requirement Description เมื่อพบแล้วให้ทำคอลอันสีเขียวที่ชื่อของ Method และที่คำกริยา	ไม่พบ Method ที่สอดคล้องกับคำกริยา
3	Requirement Description → Class Description	ค้นหา Class หรือ Attribute ใน Class Description ที่สอดคล้องกับคำนาม (น้ำเงิน) ใน Requirement Description เมื่อพบแล้วให้ทำคอลอันสีน้ำเงินที่คำนาม และที่ Class หรือ Attribute	ไม่พบ Class หรือ Attribute ที่สอดคล้องกับคำนาม
4	Requirement Description → Class Description	ถ้าคำนามใน Requirement Description สอดคล้องกับ Class ใน Class Description ให้ตรวจสอบว่า - Class (คอลอันน้ำเงิน) มีข้อมูลที่ครบถ้วนสำหรับฟังก์ชันการทำงาน -Attribute (คอลอันน้ำเงิน) ของ Class เหล่านี้เป็นคำนาม - Method (คอลอันเขียว) ของ Class เป็นคำกริยา - Constraint สำหรับ Class เหล่านี้ตรงกับใน Requirement Description	- Class (คอลอันน้ำเงิน) มีข้อมูลสำหรับฟังก์ชันการทำงานไม่ครบถ้วน - Attribute (คอลอันน้ำเงิน) ของ Class ไม่ใช่คำนาม - Method (คอลอันเขียว) ของ Class ไม่ใช่คำกริยา - Constraint ใน Class Description ไม่ตรงตามใน Requirement Description
5	Requirement Description → Class Description	ถ้าคำนามใน Requirement Description สอดคล้องกับ Attribute ใน Class Description ให้ตรวจสอบว่า - ประเภทของ Attribute (คอลอันน้ำเงิน) กำหนดไว้อย่างเหมาะสม	- ประเภทของ Attribute (คอลอันน้ำเงิน) ใน Class Description ไม่เหมาะสม
6	Requirement Description	ตรวจสอบว่าคำนามหรือคำกริยาทุกคำใน Requirement Description ถูกทำคอลอัน	มีคำนามหรือคำกริยาใน Requirement Description ที่ไม่ได้ถูกทำคอลอัน

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	- ขีดเส้นใต้และใส่ตัวเลขกำกับด้วยสีน้ำเงินไว้ที่คำนาม (คำนามเดียวกันไม่ต้องขีดซ้ำ) - ขีดเส้นใต้และใส่ตัวเลขกำกับด้วยสีเขียวที่คำกริยา (คำกริยาเดียวกันไม่ต้องขีดซ้ำ) - วงกลมด้วยสีเหลืองที่ Constraint และ Condition	
2	Use Cases	อ่านดูว่าในการทำแต่ละฟังก์ชันต้องใช้ข้อมูลใดบ้าง จากนั้นเขียน Di,j ด้วยสีแดงกำกับที่ข้อมูล โดยที่ i และ j คือตัวเลขที่กำกับอยู่บนคำนามที่ข้อมูลส่งหากัน	
3	Sequence Diagram	- ขีดเส้นใต้ที่ Object ด้วยสีน้ำเงิน และใส่ตัวเลขกำกับโดยให้เลขสอดคล้องกับ Use Cases - ขีดเส้นใต้ด้วยสีเขียวและใส่ตัวเลขกำกับตามลำดับการส่ง Message - วงกลมด้วยสีเหลืองที่ 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	- ตรวจสอบลำดับการส่ง Message (เขียว) ใน Sequence Diagram ว่าตรงกับใน Use Cases หรือไม่ - ตรวจสอบข้อมูลของแต่ละ 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	ตรวจสอบว่า Constraint และ Condition (เหลือง) ใน Sequence Diagram ตรงกับใน Use Cases (เหลือง) หรือไม่	Constraint หรือ Condition (เหลือง) ใน Sequence Diagram ไม่ตรงกับใน Use Cases (เหลือง)

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	วงกลมด้วยสีแดงที่ฟังก์ชันการทำงานของระบบ	
2.	Requirement Description	ขีดเส้นใต้ด้วยดินสอและใส่หมายเลขกำกับที่คำอธิบาย 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	- *ถ้าสามารถกำหนด* Event ที่เป็นสาเหตุให้เปลี่ยน State ระหว่างคู่ใด ได้ให้*ใส่ Event* ไว้ที่ช่องนั้น (*ถ้า* Event นั้นมี Constraint หรือ Condition อธิบายไว้ใน Requirement Description ให้*เขียน*ลงใน Adjacency Matrix ด้วย) - *ถ้าคิดว่าเป็นไปไม่ได้ที่จะ*เปลี่ยน State ระหว่าง State คู่นั้นให้*ใส่เครื่องหมายกากบาท* - *ถ้า*ไม่แน่ใจให้*ปล่อย*ช่องนั้นว่างไว้	
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 เพื่อหมดแล้วให้เขียนตัวเลขด้วยดินสอกำกับโดยใช้ตัวเลขเดียวกับใน 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) ต้นฉบับ ผู้วิจัยอ้างอิงเอกสารเช็กลิสต์จาก 2 งานวิจัยคือ Sabaliauskaite (2004) และงานวิจัยของ Wang และ Lu (2001)

3.1 เอกสารเช็กลิสต์จากงานวิจัยของ Sabaliauskaite (2004)

CHECKLIST

Locate the following diagrams: Class Diagrams, Activity Diagrams, Sequence Diagrams and Component Diagrams. Answer the questions related to corresponding diagrams. When you detect a defect, mark it on the diagram and fill the necessary information in the Defect registration form. If you have any comments, write them in Comment form.

Class Diagrams		
1.	Aren't there any inconsistencies between Class Diagrams and Requirements Specification?	☐ yes ☐ no
2.	Are all the necessary Classes and Associations defined?	☐ yes ☐ no
3.	Are there no redundant elements in Class Diagrams?	☐ yes ☐ no
4.	Is the multiplicity of all Associations defined?	☐ yes ☐ no
5.	Do Class Diagrams have enough information for software code development?	☐ yes ☐ no
Activity Diagrams		
6.	Aren't there any inconsistencies between Activity Diagrams and Requirements Specification?	☐ yes ☐ no
7.	Are all the necessary States and Transitions defined?	☐ yes ☐ no
8.	Is the order of the States correct?	☐ yes ☐ no
9.	Are there no redundant elements in Activity Diagrams?	☐ yes ☐ no
Sequence Diagrams		
10.	Aren't there any inconsistencies between Sequence Diagrams, Class Diagrams and Requirements Specification?	☐ yes ☐ no
11.	Are all the necessary Objects and Messages shown?	☐ yes ☐ no
12.	Is every Class from Class Diagrams included in any of Sequence Diagrams and vice versa?	☐ yes ☐ no
13.	Is every Use Case from Use-Case Diagrams implemented in one of Sequence Diagrams?	☐ yes ☐ no
14.	Are there no redundant elements in Sequence Diagrams?	☐ yes ☐ no
Component Diagrams		
15.	Aren't there any inconsistencies between Component Diagrams and other diagrams?	☐ yes ☐ no
16.	Are all the Classes from Class Diagrams included in one of Components?	☐ yes ☐ no
17.	Are the necessary Software Components and their relationships depicted?	☐ yes ☐ no
18.	Are there no redundant elements in Component Diagrams?	☐ yes ☐ no
19.	Do Component Diagrams implement all use cases of the system from Use-Case Diagrams?	☐ yes ☐ no
20.	Do Component Diagrams have enough information for implementation?	☐ yes ☐ no

3.2 เอกสารเช็กลิสต์จากงานวิจัยของ Wang และ Lu (2001)

3.2. Check list of the use case diagram

Does it have a system name?

Does it have a use case description?

Does it have preconditions?

Does it have clean system boundary?

Does it list all the actors?

Does it list some unnecessary actors?

Does it capture all the possible functionality of system?

3.3. Check list of the Sequence Diagram

Does sequence diagram follow the time line?

Is the activation bar length correct?

Is the life line length correct?

Is the message notation clear and logic?

Does the diagram show the correct objects' creators?

Does each sequence diagram can find a use case in use case diagram?

Is there an actor to initiate the sequence diagram?

Does each sequence diagram keep synchronized with class diagram?

Is correct guard condition stated?

3.4. Check list of the collaboration diagram

Is this collaboration diagram too big? (rule of thumb)

Does the collaboration diagram have correct sequence number?

Does the whole diagram focused on the object role instead of message time line?

Is the guard condition correctly set?

Is the initiating operation identified?

Does the whole collaborate diagram can successfully accomplish the initating operation?

Are the interactions between objects correctly identified?

3.5. Check list of the class diagram

Does each use case can be accomplished in this class diagrams?

Is aggregation relationship correct?

Is aggregation mixed up with composition?

Is generalization relationship correct?

Is the navigability direction consistent with ownership?

Is the multiplicity number correct?

Is this class diagram too big need to group into different packages (rule of thumb)?

3.6. Check list of state-chart diagram

Does the state chart diagram clearly say which subject it describes?

Does the diagram identify all the possible states?

Are all the possible transitions between states identified?

Are all the events triggering each state's transition identified?

Are the necessary guard conditions labeled out?

Are the necessary action of event identified?

Does the state chart have initiate and ending state?

4. เอกสารเช็กลิสต์ (Checklist) ที่ใช้ในการทดลอง เป็นเอกสารเช็กลิสต์ที่ผู้วิจัยปรับปรุงขึ้นมาเพื่อใช้ในการทดลอง

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 ค
เอกสารบันทึกข้อบกพร่อง

1. เอกสารบันทึกข้อบกพร่องที่ใช้ในขั้นตอนการจัดเตรียม

เอกสารบันทึกข้อบกพร่อง (Preparation)

รหัสผู้ตรวจสอบ.....
เวลาที่ใช้.....

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

2. เอกสารบันทึกข้อบกพร่องที่ใช้ในขั้นตอนการประชุมตรวจสอบ

เอกสารบันทึกข้อบกพร่อง (Meeting)

รหัสผู้ตรวจสอบ 1).....

2).....

3).....

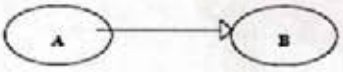
เวลาที่ใช้ .....

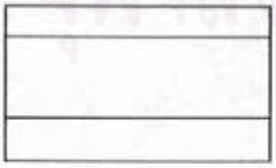
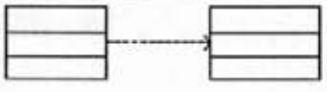
ข้อ	Reading / ข้อ	ชื่อเอกสาร	ตำแหน่งที่พบ	คำอธิบาย
1.				
2.				
3.				
4.				
5.				
6.				
7.				
8.				
9.				
10.				

ภาคผนวก ง

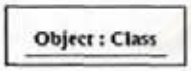
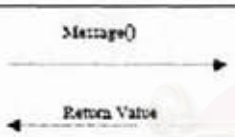
เอกสารแสดงความหมายสัญลักษณ์ของแผนภาพยูเอ็มแอล

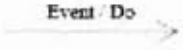
เอกสารนี้ประกอบด้วยสัญลักษณ์ของแผนภาพยูสเคส แผนภาพคลาส แผนภาพซีเควนซ์ และแผนภาพสถานะ จัดทำขึ้นเพื่อให้หน่วยทดลองสามารถอ่านเอกสารแสดงการออกแบบซอฟต์แวร์เชิงวัตถุให้เข้าใจมากยิ่งขึ้น

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 ตามด้วยเครื่องหมายทัพภาคคู่และชื่อ 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 จ

ผลการทดลองของหน่วยทดลองที่ใช้เทคนิคซีบีอาร์

กลุ่มที่ 1

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
01	นส. สุพรรษา ธราพร	7	71	0.099	35
02	นส. จุฑารัตน์ คุมกลาง	11	67	0.164	55
03	นาย ชนัดถ์ ถนัดพจนามาตย์	9	90	0.100	45

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
13	21	14	55	0.236	65	33.33

กลุ่มที่ 2

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
04	นส. พรภัทร ตฤณตียกูล	15	87	0.172	75
05	นส. ขนิษฐา สาธิตไพศาลกุล	9	83	0.108	45
06	นาย ชนาวุธ เปรุณาวิน	7	73	0.096	35

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
17	15	3	47	0.362	85	80

กลุ่มที่ 3

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
07	นส. เพ็ญพรรณ บูรณประเสริฐสุข	7	88	0.080	35
08	นส. ราณี สมใจมิตร	8	83	0.096	40
09	นส. อรชนก ช่อสมบัติ	7	65	0.108	35

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
11	12	1	42	0.262	55	91.67

กลุ่มที่ 4

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
10	นาย เอกองค์ อธิปธรรมวาร	10	90	0.111	50
11	นส. สุดาทิพย์ สุขสอาด	3	90	0.033	15
12	นาย ภมร วรรธกะวิกานต์	7	90	0.078	35

Meeting						
จำนวน ข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
12	16	6	58	0.207	60	62.5

กลุ่มที่ 5

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
13	นาย ปฤสันต์ ผูกสถิตย์	10	90	0.111	50
14	นส. สุนันทา เปี่ยมพริ้ง	9	90	0.100	45
15	นาย อนาวิล อมรเดชากุล	10	90	0.111	50

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
10	8	2	49	0.204	50	75

กลุ่มที่ 6

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
16	นาย ฉัตรไชย ชนะศิรางกูล	3	84	0.036	15
17	นส. จุฬาภา ชูอำไพ	8	78	0.103	40
18	นส. เจตปรียา เบกไธสง	10	87	0.115	50

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
9	11	4	60	0.15	45	63.64

กลุ่มที่ 7

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
19	นาย อัครเดช อุดมชัยพร	7	87	0.080	35
20	นาย ทักพงศ์ สมส่งกุล	7	83	0.084	35
21	นส. ศิริกุล โตสัมพันธ์มงคล	11	87	0.126	55

Meeting						
จำนวน ข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
11	25	19	35	0.314	55	24

กลุ่มที่ 8

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
43	นาย ประสาน พึ่งวัฒนาพงศ์	7	67	0.104	35
44	นาย นพปฏล คำนึงกิจ	4	72	0.056	20
45	นส.ศิรินทร์ วาทกิจ	6	67	0.089	30

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
11	14	4	50	0.22	55	71.43

ผลการทดลองของหน่วยทดลองที่ใช้เทคนิคโอโออาร์ที

กลุ่มที่ 1

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
22	นส. วชิรา กาญจนฤทธิไกร	2	85	0.024	10
23	นส. อรนุช จันทร์ศรี	10	90	0.111	50
24	นาย สาธร ชลานุมาศ	6	90	0.067	30

Meeting						
จำนวน ข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
13	18	6	55	0.236	65	66.67

กลุ่มที่ 2

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
25	นาย ธนาคม อุบลเพ็ง	2	90	0.022	10
26	นาย เด่นชัย เตพิมลรัตน์	10	90	0.111	50
27	นาย อิศเรศ บุญทรง	6	85	0.071	30

Meeting						
จำนวน ข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
6	23	11	40	0.15	30	52.17

กลุ่มที่ 3

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
28	นาย กิตติชัย พัฒนาปัญญาสัตย์	1	90	0.011	5
29	นาย สาธิต หิรัญยูปกรณ์	5	90	0.056	25
30	นาย ชัยวัฒน์ จินาวงศ์	7	90	0.078	35

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
7	14	4	55	0.127	35	71.43

กลุ่มที่ 4

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
31	นาย สุภกิจ จิตเจริญ	3	90	0.033	15
32	นส. เบญ กานต์จรรยา	3	90	0.033	15
33	นส. กมลวรรณ ธีระวรรณ	5	90	0.056	25

Meeting						
จำนวน ข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
5	10	6	45	0.111	25	40

กลุ่มที่ 5

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
34	นส. ปาณิศา เย็นนะสา	7	90	0.078	35
35	นส. มนัญญา ภาคศักดิ์ศรี	7	90	0.078	35
36	นส. กรวิภา เกตุเรืองไรจน์	7	90	0.078	35

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
8	9	7	50	0.16	40	22.22

กลุ่มที่ 6

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
37	นาย ชาคริต สวัสดิรักษ์	8	90	0.089	40
38	นส. อุไรภรณ์ เมืองทอง	6	90	0.067	30
39	นส. สุกาวดี แก้วกล้า	6	90	0.067	30

Meeting						
จำนวน ข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
9	6	2	50	0.18	45	66.67

210

กลุ่มที่ 7

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
40	นส. นวพร ชาติบัญชาชัย	4	90	0.044	20
41	นส. ทัศนาพร วงศ์วิเชียรชัย	4	90	0.044	20
42	นาย วีระยุทธ์ คุ้มโภคา	7	90	0.078	35

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
6	19	2	60	0.100	30	89.47

กลุ่มที่ 8

Preparation					
No.	ชื่อ - สกุล	จำนวนข้อบกพร่องที่พบ	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)
46	นส.ธันยาภรณ์ ตันทรงเจริญ	5	90	0.056	25
47	นส.ณิชยา ตั้งจิตมั่นคงชัย	6	90	0.067	30
48	นส.พรทิพย์ โสภณชัย	5	90	0.056	25

Meeting						
จำนวนข้อบกพร่องที่พบ	จำนวนผลบวกปลอม (Preparation)	จำนวนผลบวกปลอม (Meeting)	เวลาที่ใช้ (นาที)	ประสิทธิภาพ (ข้อบกพร่อง / 1 นาที)	ประสิทธิผล (%)	ประสิทธิผลในการกำจัดผลบวกปลอม (%)
7	4	1	40	0.175	35	75

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวอุมาพร นิลเอวะ เกิดวันที่ 14 สิงหาคม พ.ศ. 2525 สำเร็จการศึกษาวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ จากภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีพ.ศ. 2547 จากนั้นได้เข้าศึกษาต่อในระดับปริญญาโท สาขาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย