การตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล

Use Case Validation for Object-Oriented Software Development with UML

ปรณัฐ เตี้ยนวล
Poranat Tianual

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์
A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Science
Prince of Songkla University
2562

การตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุ ด้วยยูเอ็มแอล

Use Case Validation for Object-Oriented Software Development with UML

ปรณัฐ เตี้ยนวล
Poranat Tianual

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์
A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Science
Prince of Songkla University
2562

ชื่อวิทยานิพนธ์ การตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล
ผู้เขียน นายปรณัฐ เตี้ยนวล
สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

.....
(ผู้ช่วยศาสตราจารย์ ดร. อำนาจ เปาะทอง)

คณะกรรมการสอบ

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. ฐิมาพร เพชรแก้ว)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. อำนาจ เปาะทอง)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. สุภาภรณ์ กานต์สมเกียรติ)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

.....
(ศาสตราจารย์ ดร. ดำรงศักดิ์ ฟ้ารุ่งสาง)
คณบดีบัณฑิตวิทยาลัย

ขอรับรองว่า ผลงานวิจัยนี้มาจากการศึกษาวิจัยของนักศึกษาเอง และได้แสดงความขอบคุณบุคคลที่มีส่วนช่วยเหลือแล้ว

ลงชื่อ .....
(ผู้ช่วยศาสตราจารย์ ดร. อำนาจ เปาะทอง)
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

ลงชื่อ .....
(นายปรณัฐ เตี้ยนวล)
นักศึกษา

ข้าพเจ้าขอรับรองว่า ผลงานวิจัยนี้ไม่เคยเป็นส่วนหนึ่งในการอนุมัติปริญญาในระดับใดมาก่อน และไม่ได้ถูกใช้ในการยื่นขออนุมัติปริญญาในขณะนี้

ลงชื่อ .....
(นายปรณัฐ เตี้ยนวล)
นักศึกษา

ชื่อวิทยานิพนธ์ การตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล
ผู้เขียน นายปรณัฐ เตี้ยนวล
สาขาวิชา วิทยาการคอมพิวเตอร์
ปีการศึกษา 2561

บทคัดย่อ

ในยุคสมัยปัจจุบัน วิศวกรรมซอฟต์แวร์เชิงวัตถุได้รับความนิยมและมีบทบาทมากในกลุ่มนักพัฒนาซอฟต์แวร์ โดยเฉพาะวิศวกรรมซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล การตรวจสอบความถูกต้องของยูสเคส ซึ่งเป็นแบบจำลองยูเอ็มแอลที่สร้างขึ้นในการวิเคราะห์ฟังก์ชันงานของระบบซอฟต์แวร์ที่จะพัฒนา และใช้เป็นต้นแบบที่สำคัญซึ่งส่งสารสนเทศต่อไปยังการวิเคราะห์และออกแบบในกระบวนการดำเนินงานด้วยยูเอ็มแอลในขั้นตอนถัดไป ดังนั้นงานวิจัยนี้จึงนำเสนอการตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล โดยการสร้างกรณีทดสอบจากคุณลักษณะความต้องการและจากคำอธิบายยูสเคส แล้วนำกรณีทดสอบที่ได้มาเปรียบเทียบกัน โดยความสอดคล้องกันของกรณีทดสอบจะเป็นตัวบอกความถูกต้องของยูสเคส ทั้งนี้ได้มีการพัฒนาระบบต้นแบบเพื่อประเมินประสิทธิภาพในการทำงานของแนวคิดที่นำเสนอ และเลือกระบบให้บริการยืม-คืนของระบบห้องสมุดอัตโนมัติมาใช้เป็นกรณีตัวอย่างในการประเมินนี้ ผลการดำเนินการโดยนักศึกษาระดับบัณฑิตศึกษา จำนวน 4 ราย และผู้เชี่ยวชาญที่เป็นนักวิเคราะห์ระบบงานคอมพิวเตอร์ จำนวน 4 ราย พบว่าในด้านการสร้างกรณีทดสอบ แนวคิดที่นำเสนอมีประสิทธิภาพมากกว่านักศึกษาระดับบัณฑิตศึกษาโดยเฉลี่ยเป็นร้อยละ 20.35 และมีประสิทธิภาพมากกว่าผู้เชี่ยวชาญโดยเฉลี่ยเป็นร้อยละ 5.23 ในแง่ของความสมบูรณ์ของกรณีทดสอบ ในขณะที่ในด้านการตรวจสอบความถูกต้องของยูสเคส แนวคิดที่นำเสนอมีประสิทธิภาพมากกว่านักศึกษาระดับบัณฑิตศึกษาโดยเฉลี่ยเป็นร้อยละ 66.67 และมีประสิทธิภาพมากกว่าผู้เชี่ยวชาญโดยเฉลี่ยเป็นร้อยละ 62.50 ในแง่ของความถูกต้องของยูสเคส

Thesis Title	Use Case Validation for Object-Oriented Software Development with UML
Author	Mr. Poranat Tianual
Major Program	Computer Science
Academic Year	2018

ABSTRACT

Nowadays, object-oriented software engineering has gained popularity from many software developers especially object-oriented software engineering with UML. The validation of use case models generated during an analysis phase is very important since information from the analysis phase will pass to later analysis and design in the UML process. Therefore, this research proposes the technique for detecting defects in use case models during an analysis phase. The proposed technique is generating test cases from requirements specifications and test cases from use cases descriptions, and then comparing these two sets of test cases. The validity of use cases is based on the conformity of these test cases. The simple tools were created for demonstrating and evaluating the proposed technique. The library circulation system was selected as a case study for benchmarking. The four graduate students and four analysis specialists were selected as the subjects for the evaluation. The results show that the effectiveness in term of completeness of manually decision table generation is less than the proposed technique at 20.35% for graduate students and 5.23% for specialists respectively. The effectiveness in term of correctness of manually fault detection is less than the proposed technique at 66.67% for graduate students and 62.50% for specialists respectively.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลือและสนับสนุนจากบุคคลหลายฝ่าย ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณอย่างสูง ดังนี้

ผู้ช่วยศาสตราจารย์ ดร.อำนาจ เปาะทอง อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาให้ความรู้ คำแนะนำ และช่วยเหลือในการแก้ปัญหาต่างๆ ให้แก่ผู้วิจัยเสมอมา พร้อมทั้งตรวจทานละแก้ไขวิทยานิพนธ์ให้แก่ผู้วิจัย

ผู้ช่วยศาสตราจารย์ ดร.ฐิมาพร เพชรแก้ว ประธานกรรมการสอบวิทยานิพนธ์ ที่กรุณาให้ข้อเสนอแนะ และช่วยตรวจทานแก้ไขวิทยานิพนธ์ให้มีความถูกต้องสมบูรณ์

ผู้ช่วยศาสตราจารย์ ดร.สุภาภรณ์ กานต์สมเกียรติ กรรมการในการสอบวิทยานิพนธ์ ที่กรุณาให้ข้อเสนอแนะ และช่วยตรวจทานแก้ไขวิทยานิพนธ์ให้มีความถูกต้องสมบูรณ์

อาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์ทุกท่านที่ให้ความรู้ทางด้านวิชาการ และคำแนะนำในการทำวิทยานิพนธ์

นักศึกษาปริญญาโทและปริญญาเอกที่ช่วยเป็นผู้ประเมินนำร่องประสิทธิภาพของระบบ นักวิเคราะห์ระบบ และผู้เข้าร่วมทดลองประเมินระบบทุกท่าน ตลอดถึงศูนย์คอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ ที่อนุเคราะห์ข้อมูลสำหรับกรณีตัวอย่าง

เจ้าหน้าที่ภาควิชาวิทยาการคอมพิวเตอร์ และเจ้าหน้าที่บัณฑิตวิทยาลัยทุกท่านที่ให้ความช่วยเหลือ และอำนวยความสะดวกเกี่ยวกับเอกสารและอุปกรณ์ต่างๆ

เพื่อนๆ พี่ๆ และน้องๆ ในภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ ทั้งที่ยังศึกษาอยู่ และที่จบการศึกษาไปแล้ว รวมถึง เพื่อนๆ จากภาควิชาอื่นๆ ที่ช่วยประเมินประสิทธิภาพของระบบเบื้องต้น ให้คำปรึกษา และกำลังใจในการทำวิทยานิพนธ์

คุณพัศชนันท์ เจียจิรโชติ ที่เป็นกำลังใจและพลังงานในการทำงานวิจัยในทุกๆ วัน

คุณพ่อ คุณแม่ และครอบครัว ที่สนับสนุนผู้วิจัยเสมอมา

ผู้วิจัยขอขอบคุณทุกท่านเป็นอย่างสูงมา ณ โอกาสนี้

ปรณัฐ เตี้ยนวล

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

รายการตาราง

ตาราง **หน้า**

รายการตาราง (ต่อ)

ตาราง **หน้า**

รายการภาพประกอบ

ภาพประกอบ **หน้า**

รายการภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของงานวิจัย

ในการผลิตซอฟต์แวร์นั้น การควบคุมคุณภาพของซอฟต์แวร์ถือเป็นสิ่งสำคัญที่นักพัฒนาระบบต้องคำนึงถึง เพราะนอกจากจะเป็นการเพิ่มความน่าเชื่อถือให้กับผลิตภัณฑ์ซอฟต์แวร์แล้วยังสามารถลดต้นทุนในกระบวนการผลิตได้ ซอฟต์แวร์นอกจากจะถูกตรวจสอบว่าตรงตามความต้องการหรือไม่ ยังต้องมีการตรวจสอบว่ามีข้อผิดพลาดอยู่ในผลิตภัณฑ์ที่จะส่งมอบให้ผู้ใช้หรือไม่ ซอฟต์แวร์ที่ไม่มีคุณภาพจะต้องถูกปรับปรุงแก้ไขก่อนที่จะส่งมอบให้กับผู้ใช้ ทำให้การทดสอบซอฟต์แวร์เข้ามามีบทบาทสำคัญในกระบวนการผลิตซอฟต์แวร์ ซึ่งการทดสอบนั้นมีได้หลายระดับและทำได้ในทุกๆ ขั้นตอนของการผลิต หากมีการตรวจสอบและคัดกรองข้อผิดพลาดได้เร็ว ก็จะยิ่งลดต้นทุนในการผลิตและลดข้อผิดพลาดได้มากขึ้น

การวิเคราะห์ความต้องการ (Requirements Analysis) เป็นขั้นตอนหนึ่งในกระบวนการวิศวกรรมความต้องการ โดยในขั้นตอนนี้นักวิเคราะห์ระบบ (System Analyst) จะวิเคราะห์ความต้องการที่เก็บรวบรวมมาจากผู้ใช้ด้วยแนวทางและวิธีการพัฒนาซอฟต์แวร์ที่เลือกใช้ แนวทางการวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design) นักวิเคราะห์ระบบมักเริ่มสร้างแบบจำลองด้วยแบบจำลองยูสเคส (Use Case Model) โดยอาศัยภาษายูเอ็มแอล (UML : Unified Modeling Language) โดยปกติแล้วแบบจำลองยูสเคสสามารถนำมาสร้างกรณีทดสอบสำหรับใช้ในขั้นตอนการทดสอบเพื่อการยอมรับของผู้ใช้งาน (User Acceptance Test) ซึ่งเป็นการทดสอบระบบขั้นตอนสุดท้ายเพื่อให้แน่ใจว่าระบบที่พัฒนาใช้งานได้จริง และผ่านการยอมรับโดยผู้ใช้งานระบบ แต่การทดสอบขั้นตอนนี้สามารถกระทำได้ก็ต่อเมื่อมีการพัฒนาระบบเสร็จสมบูรณ์แล้ว ซึ่งมักจะเป็นเวลานานหลังจากที่มีการสร้างแบบจำลองยูสเคสและสร้างกรณีทดสอบ การพัฒนาซอฟต์แวร์ตามแนวทางเชิงวัตถุด้วยยูเอ็มแอล แบบจำลองยูสเคสนับว่ามีความสำคัญมากเปรียบได้กับการวางฐานรากของบ้านหรืออาคาร เพราะยูสเคสจะถูกนำไปออกแบบส่วนต่างๆ ของระบบซอฟต์แวร์ต่อไป ดังนั้น ถ้านักวิเคราะห์สามารถตรวจสอบความถูกต้องของยูสเคสได้ ก็จะลดความผิดพลาดที่จะเกิดขึ้นในผลิตภัณฑ์ซอฟต์แวร์ได้ตั้งแต่ช่วงต้นของกระบวนการ

การดำเนินงานวิจัยในครั้งนี้จึงเสนอแนวคิดวิธีการตรวจสอบความถูกต้องของแบบจำลองยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล เพื่อให้นักวิเคราะห์สามารถนำกรณีทดสอบที่ได้จากการวิเคราะห์ความต้องการ หรือข้อกำหนดคุณลักษณะความต้องการ มาใช้ตรวจสอบเชิงกระบวนการทำงานได้ตั้งแต่ในขั้นตอนการวิเคราะห์ความต้องการโดยไม่จำเป็นต้อง

รอให้ระบบพัฒนาจนเสร็จ โดยสามารถระบุได้ตั้งแต่ในขั้นตอนการวิเคราะห์ความต้องการว่าแบบจำลองยูสเคสสอดคล้องกับความต้องการของผู้ใช้หรือไม่

1.2 งานวิจัยที่เกี่ยวข้อง

1) ระบบสนับสนุนสำหรับการสร้างกรณีทดสอบเบื้องต้นบนพื้นฐานของคุณลักษณะความต้องการที่เขียนอยู่ในรูปแบบฟอร์ม (สิริมา, 2557)

งานวิจัยนี้มีวัตถุประสงค์คือการสร้างเครื่องมือสำหรับช่วยสร้างกรณีทดสอบสำหรับใช้ในขั้นตอน Acceptance Test โดยจะรับความต้องการเข้ามาในระบบด้วยรูปแบบที่กำหนดไว้ จากนั้นสกัดข้อมูลจากความต้องการเพื่อสร้างเป็นกรณีทดสอบด้วยการสร้างตารางตัดสินใจจากค่าที่เป็นไปได้ของข้อมูลนำเข้า วิธีการนี้สามารถนำมาใช้ได้จริง และสามารถนำมาประยุกต์ใช้กับคำอธิบายยูสเคสได้

2) Defects in Automotive Use Cases (Törner, 2006)

งานวิจัยนี้มีจุดประสงค์คือการจำแนกประเภท จัดกลุ่ม และระบุความรุนแรงของข้อผิดพลาดในแบบจำลองยูสเคส โดยใช้ระบบอุตสาหกรรมรถยนต์เป็นกรณีศึกษาและเพื่อการประเมินแนวคิด ใช้งานวิจัยนี้เป็นแนวทางในการพิจารณาข้อผิดพลาดในยูสเคสที่ควรจะต้องถูกตรวจสอบ และแยกประเด็นที่ควรต้องตรวจสอบ ทำให้ได้ข้อสรุปว่าในงานวิจัยนี้ จะมุ่งเน้นไปที่การตรวจสอบข้อผิดพลาดในยูสเคสในประเด็นความถูกต้อง และความครบถ้วนของยูสเคส

3) Using acceptance tests as a support for clarifying requirements: A series of experiments (Ricca, 2007)

งานวิจัยนี้มีวัตถุประสงค์คือการตรวจสอบความถูกต้องของความต้องการที่ได้รับมาจากผู้ใช้โดยการนำความต้องการ กับกรณีทดสอบ Acceptance Test มาใช้เครื่องมือช่วย คือ FIT Table (Framework for Integration Test) ซึ่งเป็นเครื่องมือที่สามารถจำลองการทำงานของซอฟต์แวร์ภายใต้เงื่อนไขของความต้องการได้ และนำผลการทำงานมาเปรียบเทียบกับกรณีทดสอบเพื่อหาข้อผิดพลาดของความต้องการ ทำให้นักพัฒนาระบบสามารถทำการ Acceptance Testing เบื้องต้นได้แม้ซอฟต์แวร์จะยังไม่ได้รับการพัฒนาขึ้นมา และในงานวิจัยนี้ก็ได้ทำการทดลองเพื่อวัดประสิทธิภาพการทำงานของ FIT Table และพบว่าสามารถเพิ่มความเข้าใจที่นักพัฒนาระบบมีต่อความต้องการ และเพิ่มความถูกต้องของความต้องการได้จริง นำแนวคิดของวิธีการนี้มาประยุกต์ใช้ได้ คือ จากการนำกรณีทดสอบ Acceptance Test กลับมาทดสอบความถูกต้องของความต้องการ นำมาปรับให้เป็น การนำกรณีทดสอบ Acceptance มาเป็นตัวทดสอบความถูกต้องของยูสเคส

4) Generation of Test Requirements from Aspectual Use Cases (Xu, 2007)

งานวิจัยนี้มีจุดประสงค์คือสร้างความต้องการสำหรับการทดสอบระบบจากยูสเคสเชิงลักษณะ (Aspect-Oriented Use Case) โดยใช้วิธีการรูปนัย (Formal Methods) แปลงยูสเคสให้อยู่ในรูปเพทริเนท (Petri Net) จากนั้น สร้างความต้องการสำหรับการทดสอบระบบจากแบบจำลองเพทริเนท ใช้งานวิจัยนี้ประกอบการพิจารณาแนวทางในการสร้างกรณีทดสอบจากยูสเคส โดยแนวทางนี้ใช้เครื่องมือช่วยคือวิธีการทางรูปนัย และสร้างกรณีทดสอบเพื่อทดสอบระบบ

5) An Approach to Generate Test Goals from Use Case Scenarios (Mahmood, 2013)

งานวิจัยนี้มีจุดประสงค์คือทดสอบระบบโดยใช้เหตุการณ์ยูสเคส (Use Case Scenario) ด้วยการแบ่งยูสเคสของระบบให้ออกมาเป็นเหตุการณ์ย่อย จากนั้นสร้างกรณีการดำเนินการ (Execution Contract) จากเหตุการณ์ย่อยโดยใช้แผนภาพลำดับขั้น (Sequence Diagram) เป็นเครื่องมือช่วย และสร้างเป้าหมายของการทดสอบจากกรณีการดำเนินการนั้น ใช้งานวิจัยนี้ประกอบการพิจารณาแนวทางในการสร้างกรณีทดสอบจากยูสเคส โดยแนวทางนี้ใช้เครื่องมือช่วยคือแผนภาพลำดับขั้น และสร้างกรณีทดสอบเพื่อการทดสอบระบบ

6) Automatic Early Defects Detection in Use Case Documents (Liu, 2014)

งานวิจัยนี้มีจุดประสงค์คือหาความผิดพลาดในเอกสารยูสเคสแบบอัตโนมัติ ดำเนินการโดยการใช้การประมวลภาษาธรรมชาติ (NLP : Natural Language Processing) มาเป็นเครื่องมือในการแยกศัพท์และหาความผิดพลาด ใช้งานวิจัยนี้เพื่อประกอบการพิจารณาแนวทางในการตรวจสอบความถูกต้องของยูสเคส โดยแนวทางนี้ใช้การประมวลผลภาษาธรรมชาติเป็นเครื่องมือช่วยในการหาข้อผิดพลาดในยูสเคสแบบอัตโนมัติ

1.3 วัตถุประสงค์ของงานวิจัย

1) เพื่อพัฒนาแนวคิดสำหรับการตรวจสอบความถูกต้องของยูสเคส (Use Case Validation) บนพื้นฐานของข้อกำหนดคุณลักษณะความต้องการของซอฟต์แวร์ (Requirements Specification) และกรณีทดสอบเพื่อการยอมรับซอฟต์แวร์ (Acceptance Test Case)

2) เพื่อพัฒนาเครื่องมือสำหรับตรวจสอบความถูกต้องของยูสเคสจากวิธีการที่นำเสนอ

1.4 ขอบเขตการดำเนินงานของงานวิจัย

1) กำหนดให้ข้อกำหนดคุณลักษณะความต้องการซอฟต์แวร์ในรูปแบบพื้นฐานซึ่งถูกระบุโดยผู้ใช้และถูกกลั่นกรองโดยนักวิเคราะห์แล้ว เป็นข้อกำหนดที่ถูกต้อง เช่น ข้อกำหนดคุณลักษณะในรูปแบบฟอร์ม (Form-Based Requirements Specification) ที่ใช้ในงานวิจัยนี้

2) พัฒนาแนวคิดสำหรับการตรวจสอบความถูกต้องของยูสเคส โดยประกอบด้วยวิธีการดังนี้

2.1) การออกแบบกรอบแนวคิดในการตรวจสอบความถูกต้องของแบบจำลองยูสเคส (Use Case Validation Framework)

2.2) การออกแบบโครงสร้างการทำงานและขั้นตอนวิธีในการตรวจสอบความถูกต้องของแบบจำลองยูสเคส (Procedure and Algorithm Design for Use Case Validation)

3) พัฒนาเครื่องมือสำหรับตรวจสอบความถูกต้องตามที่ได้ออกแบบเอาไว้

<u>หมายเหตุ</u> สารสนเทศที่ผิดพลาดซึ่งถูกยืนยันโดยผู้ใช้หรือผู้ต้องการซอฟต์แวร์และนักวิเคราะห์ในช่วงการวิเคราะห์ความต้องการแล้วและนำไปสู่การแก้ไขในภายหลังไม่รวมอยู่ในขอบเขตงานวิจัยนี้

1.5 ขั้นตอนและระยะเวลาการดำเนินงาน

1.5.1 ขั้นตอนการดำเนินงาน

1) ศึกษางานวิจัยและเอกสารที่เกี่ยวข้อง
2) วิเคราะห์และเสนอกรอบแนวคิด
3) ศึกษาเทคนิคและเครื่องมือที่ใช้ในการพัฒนาระบบ
4) ออกแบบระบบสนับสนุนกรอบแนวคิด
5) พัฒนาและทดสอบระบบ
6) สรุปผล จัดทำรายงานวิจัย และเขียนบทความวิจัยเพื่อเผยแพร่

1.5.2 ระยะเวลาที่ใช้ในงานวิจัย

พฤศจิกายน 2559 – เมษายน 2562

1.5.3 แผนการดำเนินงานวิจัย

ระยะเวลาการดำเนินงานวิจัย แสดงดังตารางที่ 1.1

ตารางที่ 1.1 ระยะเวลาการดำเนินงานวิจัย

กิจกรรม	2559	2560		2561		2562
ดำเนินการ	11-12	1-6	7-12	1-6	7-12	1-4
1	←	—	—	→		
2	←	—	→			
3	←	—	—	→		
4		←	—	—	→	
5		←	—	—	—	→
6			←	—	—	→

1.6 สถานที่และเครื่องมือที่ใช้

1.6.1 สถานที่

ห้องวิจัยกลุ่มวิศวกรรมซอฟต์แวร์ ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่

1.6.2 เครื่องมือที่ใช้

1) ด้านซอฟต์แวร์

ซอฟต์แวร์ระบบและซอฟต์แวร์ประยุกต์ดังต่อไปนี้

1. Microsoft Windows 7 Professional 32 bits
2. Microsoft Visual Studio 2013
3. Microsoft SQL Server Management Studio 2012

2) ด้านฮาร์ดแวร์

เครื่องคอมพิวเตอร์ส่วนบุคคลที่มีคุณลักษณะดังนี้

1. หน่วยประมวลผล - Intel Core 2 Quad Q6600 2.40Ghz
2. หน่วยความจำ - 3.00 GB

1.7 ประโยชน์ที่คาดว่าจะได้รับ

1) เพิ่มประสิทธิภาพและความน่าเชื่อถือในการวิเคราะห์ระบบของนักวิเคราะห์ระบบ

2) เพิ่มความถูกต้องของแบบจำลองยูสเคส

3) ลดต้นทุนในการผลิตซอฟต์แวร์

4) สามารถนำกรณีทดสอบเพื่อการยอมรับซอฟต์แวร์ที่ได้จัดทำไว้ในช่วงการวิเคราะห์ระบบมาใช้ประโยชน์ในการตรวจสอบยืนยันความถูกต้องของยูสเคส

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

เนื้อหาในบทนี้กล่าวถึงทฤษฎีและหลักการที่เกี่ยวข้องกับงานวิจัย โดยส่วนแรกจะกล่าวถึงกระบวนการวิศวกรรมซอฟต์แวร์ ส่วนที่สองจะกล่าวถึงการทดสอบซอฟต์แวร์ ส่วนที่สามจะกล่าวถึงการวิเคราะห์และออกแบบเชิงวัตถุ และส่วนที่สี่จะกล่าวถึงยูเอ็มแอล

2.1 กระบวนการวิศวกรรมซอฟต์แวร์

2.1.1 วัฏจักรชีวิตของโครงการซอฟต์แวร์

วัฏจักรชีวิตของโครงการซอฟต์แวร์ (SPLC : Software Project Life Cycle) เป็นวงจรที่แสดงกระบวนการที่ควบคุมการพัฒนาซอฟต์แวร์และควบคุมโครงการซอฟต์แวร์ IEEE Std 1074-2006 ได้กำหนดมาตรฐานสำหรับวงจรชีวิตของโครงการซอฟต์แวร์ประกอบด้วยกลุ่มกิจกรรมในส่วนต่างๆ ดังนี้ (IEEE Computer Society, 2006)

2.1.1.1 กลุ่มกิจกรรมในส่วนการบริหารโครงการ

กลุ่มกิจกรรมในส่วนการบริหารโครงการ (Project Management Section of Activity Groups) เป็นกลุ่มกิจกรรมที่รวมการเริ่มต้น การติดตาม และการควบคุมโครงการซอฟต์แวร์ตลอดทั้งวัฏจักรชีวิตโครงการ ครอบคลุมกลุ่มกิจกรรมต่อไปนี้

1) กลุ่มกิจกรรมการเริ่มโครงการ

กลุ่มกิจกรรมการเริ่มโครงการ (Project Initiation Activity Group) เป็นการสร้างและปรับปรุงโครงสร้างพื้นฐานของโครงการพัฒนาหรือปรับปรุงซอฟต์แวร์ ประกอบไปด้วย การสร้างโครงการซอฟต์แวร์ (SPLC Development) การดำเนินประมาณการ (Estimation) การจัดสรรทรัพยากรของโครงการ (Project Resources Allocation) การระบุตัวชี้วัด (Metrics Definition) และการระบุวัตถุประสงค์ด้านความปลอดภัย (Security Objectives Determination)

2) กลุ่มกิจกรรมการวางแผนโครงการ

กลุ่มกิจกรรมการวางแผนโครงการ (Project Planning Activity Group) เป็นการวางแผนการจัดการโครงการทั้งหมด รวมถึงคาดการณ์เหตุการณ์ที่อาจเกิดขึ้นได้ เป็นการดำเนินงานที่สามารถทำซ้ำได้เท่าที่ต้องการตลอดทั้งวัฏจักรชีวิตของโครงการ ประกอบไปด้วย การวางแผนประเมิน (Evaluation Planning) การวางแผนจัดการการตั้งค่า (Configuration Management Planning) การวางแผนเปลี่ยนถ่ายระบบ (System Transition Planning)

การวางแผนการติดตั้ง (Installation Planning) การวางแผนการทำเอกสาร (Documentation Planning) การวางแผนการฝึกอบรม (Training Planning) การวางแผนการบริหารโครงการ (Project Management Planning) การวางแผนการรวมระบบ (Integration Planning) และการวางแผนการออกใช้งาน (Release Management Planning)

3) กลุ่มกิจกรรมการติดตามและควบคุมโครงการ

กลุ่มกิจกรรมการติดตามและควบคุมโครงการ (Project Monitoring and Control Activity Group) เป็นการติดตามและจัดการโครงการ ในกลุ่มกิจกรรมนี้ การดำเนินงานของโครงการจะถูกติดตาม รายงาน และประเมินผลเทียบกับแผนการดำเนินกงานที่วางไว้ และจัดการความเสี่ยงที่อาจเกิดขึ้นในโครงการ ประกอบไปด้วย การจัดการความเสี่ยง (Risk Managing) การจัดการโครงการ (Project Managing) การระบุความต้องการในการพัฒนาวัฏจักรชีวิตของโครงการ (Identify SPLCP Improvement Needs) การเก็บรักษาบันทึก (Records Retaining) การรวบรวมและวิเคราะห์ข้อมูลตัวชี้วัด (Metric Data Collecting and Analyzing) และ การปิดโครงการ (Project Closing)

2.1.1.2 กลุ่มกิจกรรมในส่วนการดำเนินงานก่อนการพัฒนา

กลุ่มกิจกรรมในส่วนการดำเนินงานก่อนการพัฒนา (Pre-Development Section of Activity Groups) เป็นกลุ่มกิจกรรมที่ทำการรวบรวม และจัดสรรความต้องการก่อนการพัฒนาระบบจะเริ่มต้น ครอบคลุมกลุ่มกิจกรรมต่อไปนี้

1) กลุ่มกิจกรรมการสำรวจแนวคิด

กลุ่มกิจกรรมการสำรวจแนวคิด (Concept Exploration Activity Group) เป็นการเริ่มต้นการพัฒนาด้วยการระบุแนวความคิดหรือความต้องการของซอฟต์แวร์ที่จะสร้าง ประกอบไปด้วย การระบุแนวคิดหรือความต้องการ (Ideas or Needs Identification) กำหนดวิธีการที่เป็นไปได้ (Potential Approaches Formulating) ศึกษาความเป็นไปได้ (Feasibility Studies Conducting) และปรับปรุงแนวคิดหรือความต้องการ (Idea or Need Refining)

2) กลุ่มกิจกรรมการจัดสรรระบบ

กลุ่มกิจกรรมการจัดสรรระบบ (System Allocation Activity Group) เป็นการเชื่อมโยงฟังก์ชันที่ต้องการเข้าไปหาซอฟต์แวร์ ฮาร์ดแวร์ และผู้ใช้ ประกอบไปด้วย การวิเคราะห์ฟังก์ชันของระบบ (System Functions Analyzing) การพัฒนาสถาปัตยกรรมของระบบ (System Architecture Development) และการจัดสรรความต้องการของระบบ (System Requirements Allocating)

3) กลุ่มกิจกรรมการนำเข้าซอฟต์แวร์

กลุ่มกิจกรรมการนำเข้าซอฟต์แวร์ (Software Importation Activity Group) เป็นการนำซอฟต์แวร์ที่มีอยู่มาใช้รวมกับระบบใหม่ที่จะพัฒนา เนื่องจากความต้องการของซอฟต์แวร์บางส่วนเหมาะต่อการนำซอฟต์แวร์ที่มีอยู่กลับมาใช้ใหม่ หรือจากการนำซอฟต์แวร์จากภายนอก

โครงการมาใช้ ซึ่งอาจมาจากภายนอกองค์กรที่พัฒนาซอฟต์แวร์ ซอฟต์แวร์ที่นำเข้ามาประกอบไปด้วย คลังโปรแกรม (Code Library) โปรแกรมขับอุปกรณ์ (Driver) สิ่งอำนวยความสะดวกเพิ่มเติม หรือระบบทั้งระบบที่จะนำมารวมกับโครงการที่จะพัฒนา กิจกรรมในกลุ่มนี้ประกอบไปด้วย การระบุความต้องการของซอฟต์แวร์ที่จะนำเข้า (Imported Software Requirements Identification) การประเมินแหล่งที่มาของซอฟต์แวร์ (Software Import Sources Evaluating) ระบุวิธีการนำเข้า (Software Import Method Defining) และการนำเข้าซอฟต์แวร์พร้อมเอกสารเข้าสู่โครงการ (Software Importing)

2.1.1.3 กลุ่มกิจกรรมในส่วนการดำเนินงานในการพัฒนา

กลุ่มกิจกรรมในส่วนการดำเนินงานในการพัฒนา (Development Section of Activity Groups) เป็นกลุ่มกิจกรรมที่เกิดขึ้นในช่วงการพัฒนาซอฟต์แวร์ ครอบคลุมกลุ่มกิจกรรมต่อไปนี้

1) กลุ่มกิจกรรมการวิเคราะห์ความต้องการ

กลุ่มกิจกรรมความต้องการของซอฟต์แวร์ (Software Requirements Activity Group) เป็นกลุ่มกิจกรรมที่ทำหน้าที่ปรับแต่งความต้องการของซอฟต์แวร์ที่ได้รับการจัดสรรมาจากความต้องการของระบบโดยรวม ประกอบไปด้วย การระบุและพัฒนาความต้องการ (Software Requirements Defining and Development) การระบุความต้องการส่วนติดต่อ (Interface Requirements Defining) และจัดลำดับความสำคัญและเชื่อมประสานความต้องการเข้าไว้ด้วยกัน (Software Requirements Prioritizing and Integrating)

2) กลุ่มกิจกรรมการออกแบบ

กลุ่มกิจกรรมการออกแบบ (Design Activity Group) เป็นการพัฒนาแบบจำลองของระบบให้ตอบรับกับความต้องการ ในการออกแบบระดับสถาปัตยกรรมจะเน้นไปที่ซอฟต์แวร์ที่นำมาประกอบกันเป็นระบบ โครงสร้าง และการเชื่อมต่อของซอฟต์แวร์เหล่านั้น ในการออกแบบระดับรายละเอียดจะเน้นไปที่โครงสร้างข้อมูล และกระบวนการขั้นตอนวิธีของซอฟต์แวร์ ประกอบไปด้วย การออกแบบสถาปัตยกรรม (Architectural Designing) ออกแบบฐานข้อมูล (Database Designing) ออกแบบส่วนติดต่อ (Interfaces Designing) และออกแบบรายละเอียด (Detailed Designing)

3) กลุ่มกิจกรรมการดำเนินงานพัฒนาโปรแกรมต้นฉบับ

กลุ่มกิจกรรมการพัฒนาโปรแกรมต้นฉบับ (Implementation Activity Group) เป็นการพัฒนาซอฟต์แวร์ขึ้นมาจากรายละเอียดที่ได้ออกแบบเอาไว้ ประกอบไปด้วย การสร้างโปรแกรมต้นฉบับ (Source Code Creating) การสร้างเอกสารการปฏิบัติการ (Operating Documentation Creating) การเชื่อมประสาน (Integration) และการจัดการนำซอฟต์แวร์ออกใช้งาน (Software Releases Managing)

2.1.1.4 กลุ่มกิจกรรมในส่วนการดำเนินงานหลังการพัฒนา

กลุ่มกิจกรรมในส่วนการดำเนินงานหลังการพัฒนา (Post-Development Section of Activity Groups) เป็นกลุ่มกิจกรรมที่เกิดขึ้นภายหลังการพัฒนาซอฟต์แวร์ ครอบคลุมกลุ่มกิจกรรมต่อไปนี้

1) กลุ่มกิจกรรมการติดตั้งระบบ

กลุ่มกิจกรรมการติดตั้งระบบ (Installation Activity Group) เป็นการส่งมอบและติดตั้งผลิตภัณฑ์ซอฟต์แวร์จากสภาพแวดล้อมในการพัฒนาไปสู่สภาพแวดล้อมการทำงานของผู้ใช้ ประกอบไปด้วย การกระจายซอฟต์แวร์ (Software Distribution) การติดตั้งซอฟต์แวร์ (Software Installation) และการทดสอบเพื่อการยอมรับผลิตภัณฑ์ซอฟต์แวร์ในสิ่งแวดล้อมที่ใช้งานจริง (Software Accepting in Operational Environment)

2) กลุ่มกิจกรรมการใช้งานและการสนับสนุน

กลุ่มกิจกรรมการใช้งานและการสนับสนุน (Operation and Support Activity Group) คือการใช้งานระบบของผู้ใช้ และการสนับสนุนผู้ใช้ ทั้งการช่วยเหลือในทางเทคนิค และการให้คำปรึกษาในการใช้งาน ประกอบด้วย การใช้งานระบบ (System Operation) การให้ความช่วยเหลือทางเทคนิคและการให้คำปรึกษาแก่ผู้ใช้ (Technical Assistance Providing and Consulting) และบันทึกคำร้องที่ผู้ใช้ขอความช่วยเหลือ (Support Request Log Maintaining)

3) กลุ่มกิจกรรมการบำรุงรักษาระบบ

กลุ่มกิจกรรมการบำรุงรักษาระบบ (Maintenance Activity Group) เป็นการระบุสิ่งที่ต้องการปรับปรุงและการหาผลสรุปของความผิดพลาดที่เกิดขึ้นกับระบบ ประกอบไปด้วย การระบุสิ่งที่ต้องการปรับปรุง (Software Improvement Needs Identification) การรายงานปัญหา (Problem Reporting) และการทำกระบวนการตามวัฏจักรชีวิตของโครงการซ้ำ (SPLCP Repetition)

4) กลุ่มกิจกรรมการยุติการใช้งาน

กลุ่มกิจกรรมการยุติการใช้งาน (Retirement Activity Group) เป็นการนำระบบที่มีอยู่ออกจากการใช้งาน ไม่ว่าจะเป็นการหยุดการทำงาน หรือ การแทนที่ด้วยเวอร์ชั่นใหม่กว่าหรือด้วยระบบอื่น ประกอบไปด้วย การแจ้งเตือนผู้ใช้ (User Notification) การดำเนินการแบบขนาน (Parallel Operations Conducting) และการเลิกใช้งานระบบ (System Retiring)

2.1.1.5 กลุ่มกิจกรรมในส่วนการดำเนินงานในการสนับสนุน

กลุ่มกิจกรรมการดำเนินงานในการสนับสนุน (Support Section of Activity Groups) เป็นการรับประกันความครบถ้วนสมบูรณ์และคุณภาพของระบบ ครอบคลุมกลุ่มกิจกรรมต่อไปนี้

1) กลุ่มกิจกรรมการประเมิน

กลุ่มกิจกรรมการประเมิน (Evaluation Activity Group) เป็นกลุ่มกิจกรรมที่ทำระหว่างกระบวนการในวัฏจักรชีวิตของโครงการซอฟต์แวร์เพื่อค้นหาข้อผิดพลาดในผลิตภัณฑ์หรือกระบวนการที่ใช้ รวมทั้งการทบทวนและตรวจสอบ การทดสอบ และการรายงานผล

2) กลุ่มกิจกรรมการจัดการการตั้งค่าซอฟต์แวร์

กลุ่มกิจกรรมการจัดการการตั้งค่าซอฟต์แวร์ (Software Configuration Management Activity Group) เป็นการระบุตัวตนวัตถุต่างๆ ในโครงการซอฟต์แวร์ เช่น โค้ด ต้นฉบับ เอกสาร แผนงาน คำอธิบาย เป็นต้น เพื่อให้บริการการควบคุมและการสร้างรายงานสถานะของวัตถุเหล่านั้น ทำให้การจัดการโครงการมีความชัดเจน สามารถมองเห็นได้จากภายนอก ประกอบไปด้วยการกำหนดการระบุการตั้งค่า (Configuration Identification Development) ควบคุมการตั้งค่า (Configuration Control) และการขึ้นบัญชีสถานะ (Status Accounting)

3) การจัดทำเอกสาร

การจัดทำเอกสาร (Documentation Development Activity Group) คือการวางแผน ออกแบบ ดำเนินงาน แก้ไข ผลิต แจกจ่าย และบำรุงรักษาเอกสารของโครงการ มีจุดประสงค์เพื่อจัดหาเอกสารซอฟต์แวร์ตามแต่ละช่วงเวลาให้แก่ผู้ใช้ที่ต้องการ ประกอบไปด้วย การจัดทำเอกสาร (Documentation Implementation) และการผลิตและแจกจ่ายเอกสาร (Documentation Producing and Distribution)

4) การดำเนินการฝึกอบรม

การดำเนินการฝึกอบรม (Training Activity Group) ในการสร้างผลิตภัณฑ์ซอฟต์แวร์ที่มีคุณภาพนั้นรวมไปถึงการที่มีบุคคลที่มีความรู้และทักษะ ดังนั้น ผู้ใช้อาจต้องการรับการฝึกอบรมการติดตั้ง ใช้งาน และบำรุงรักษาซอฟต์แวร์ ประกอบไปด้วย การพัฒนาสื่อวัสดุในการฝึกอบรม (Training Materials Development) การพัฒนารายการฝึกอบรม (Training Program Development) และการดำเนินการฝึกอบรม (Training Program Implementation)

2.1.2 กิจกรรมพื้นฐานของกระบวนการวิศวกรรมซอฟต์แวร์

กลุ่มของกิจกรรมที่นำไปสู่การผลิตผลิตภัณฑ์ซอฟต์แวร์นั้นมีความหลากหลาย แต่อย่างไรก็ตามแนวทางส่วนใหญ่จะมีกิจกรรมพื้นฐานดังนี้ (Sommerville, 2011)

2.1.2.1 การกำหนดคุณลักษณะความต้องการ

การกำหนดคุณลักษณะความต้องการของซอฟต์แวร์ (Software Specification) เป็นขั้นตอนการทำความเข้าใจและระบุการทำงานและบริการที่ต้องการจากระบบซอฟต์แวร์ พร้อมทั้งข้อจำกัดทั้งในการทำงานของระบบ และในการพัฒนา

2.1.2.2 การออกแบบซอฟต์แวร์และการสร้างซอฟต์แวร์

การออกแบบซอฟต์แวร์และการสร้างซอฟต์แวร์ (Software Design and Implementation) เป็นขั้นตอนการพัฒนาหรือสร้างระบบให้ได้ตรงตามคุณลักษณะความต้องการที่ได้กำหนดไว้ กิจกรรมในขั้นตอนนี้รวมถึงการออกแบบซอฟต์แวร์ และการลงมือสร้างซอฟต์แวร์

2.1.2.3 การตรวจสอบความถูกต้องของซอฟต์แวร์

การตรวจสอบความถูกต้องของซอฟต์แวร์ (Software Validation) เป็นขั้นตอนการตรวจสอบความถูกต้อง ทั้งการตรวจสอบว่ากระบวนการผลิตซอฟต์แวร์ถูกต้องตามขั้นตอนและวิธีการที่เหมาะสมอย่างถูกต้อง (Verification) และการตรวจสอบว่าผลิตภัณฑ์ซอฟต์แวร์ถูกต้องสอดคล้องกับความต้องการของผู้ใช้ (Validation) โดยการตรวจสอบความถูกต้องเป็นกิจกรรมที่สามารถทำได้ระหว่างกิจกรรมทุกๆ กิจกรรม

2.1.2.4 การวิวัฒนาการซอฟต์แวร์

การวิวัฒนาการซอฟต์แวร์ (Software Evolution) การเปลี่ยนแปลงเป็นสิ่งที่สามารถเกิดขึ้นได้ตลอดเวลา ตั้งแต่ในกระบวนการการผลิตซอฟต์แวร์ ไปจนถึงหลังจากที่ได้ผลิตภัณฑ์ซอฟต์แวร์ออกมาแล้ว ซึ่งการเปลี่ยนแปลงนั้นเป็นไปได้ตั้งแต่ การเปลี่ยนคุณลักษณะความต้องการให้สอดคล้องกับความต้องการที่เปลี่ยนไปของผู้ใช้ไปจนถึงการแก้ไขข้อผิดพลาดของซอฟต์แวร์ที่พัฒนาออกมาแล้ว

2.1.3 กระบวนการวิศวกรรมความต้องการ

วิศวกรรมความต้องการ (Requirements Engineering) เป็นกระบวนการเพื่อระบุว่าระบบซอฟต์แวร์ที่จะพัฒนาต้องให้บริการอะไรบ้าง และภายใต้เงื่อนไขอะไร ประกอบด้วยขั้นตอนหลักดังนี้ (Sommerville, 2011)

2.1.3.1 การศึกษาความเป็นไปได้

การศึกษาความเป็นไปได้ (Feasibility Study) เป็นการรวบรวมคุณลักษณะของซอฟต์แวร์ที่ต้องการมาศึกษาหาความเป็นไปได้ ว่าสามารถทำได้ด้วยเทคโนโลยีทั้งในด้านฮาร์ดแวร์และซอฟต์แวร์ในปัจจุบันหรือไม่ ศึกษาว่าสามารถดำเนินการพัฒนาซอฟต์แวร์ภายใต้ทรัพยากรที่จำกัด ศึกษาความคุ้มค่าการลงทุนในมุมมองของธุรกิจ เป็นขั้นตอนที่จะตัดสินว่าจะดำเนินโครงการต่อหรือไม่

2.1.3.2 การรวบรวมและวิเคราะห์ความต้องการ

การรวบรวมและวิเคราะห์ความต้องการ (Requirements Elicitation and Analysis) เป็นขั้นตอนการรับความต้องการของระบบผ่านทางการสำรวจระบบเก่าที่มีอยู่ การสอบถามพูดคุยกับผู้ใช้ การวิเคราะห์งาน และวิธีการอื่นๆ รวมไปถึงการสร้างแบบจำลองความต้องการเพื่อช่วยให้ผู้ใช้และนักพัฒนามีความเข้าใจระบบมากยิ่งขึ้น

2.1.3.3 การกำหนดคุณลักษณะความต้องการ

การกำหนดคุณลักษณะความต้องการ (Requirements Specification) เป็นขั้นตอนการแปลงสารสนเทศที่ได้จากการวิเคราะห์และสอบถามความต้องการให้อยู่ในรูปเอกสารที่ระบุเซตของความต้องการ และแยกประเภทของความต้องการออกเป็น 2 ประเภท คือ ความต้องการของผู้ใช้ (User Requirements) และความต้องการของระบบ (System Requirements)

2.1.3.4 การตรวจสอบความถูกต้องของความต้องการ

การตรวจสอบความถูกต้องของความต้องการ (Requirements Validation) เป็นขั้นตอนการตรวจสอบความต้องการเพื่อยืนยันความครบถ้วน สมบูรณ์ ความคงเส้นคงวา และการนำไปใช้ได้จริง ในขั้นตอนนี้หากมีการตรวจพบข้อผิดพลาดของความต้องการ จะต้องได้รับการแก้ไขก่อนนำไปเข้าสู่กระบวนการถัดไปในการพัฒนา

2.1.4 การกำหนดคุณลักษณะความต้องการ

ในกระบวนการวิศวกรรมความต้องการ การกำหนดคุณลักษณะความต้องการ (Requirements Specification) เป็นขั้นตอนการเขียนรายละเอียดคุณลักษณะความต้องการของผู้ใช้ และความต้องการของระบบ ให้อยู่ในรูปเอกสารความต้องการที่มีความชัดเจนและสามารถนำไปพัฒนาซอฟต์แวร์ได้ เล่มเอกสารอธิบายคุณลักษณะความต้องการของซอฟต์แวร์ (Software Requirements Specification : SRS) อาจถูกเขียนขึ้นมาโดยตัวแทนของนักพัฒนา ตัวแทนของผู้ใช้ หรือทั้ง 2 ฝ่ายเขียนร่วมกัน ประกอบไปด้วยรายละเอียดดังนี้

2.1.4.1 บทนำ

บทนำ (Introduction) เป็นส่วนที่บอกถึงรายละเอียดของทั้งคุณลักษณะความต้องการ ทั้งเป้าหมาย ขอบเขต หรือคำจำกัดความ

2.1.4.2 คำอธิบายโดยรวม

คำอธิบายโดยรวม (Overall Description) เป็นส่วนที่บอกถึงปัจจัยทั่วไปที่มีผลต่อซอฟต์แวร์ ความสัมพันธ์กับซอฟต์แวร์อื่น ฟังก์ชันงานของซอฟต์แวร์ ข้อจำกัดต่างๆ หรือความต้องการที่อาจเพิ่มเข้ามาในอนาคต

2.1.4.3 คุณลักษณะความต้องการเฉพาะ

คุณลักษณะความต้องการเฉพาะ (Specific Requirements) เป็นส่วนที่ใช้ระบุคุณลักษณะความต้องการทั้งหมดเพื่อให้เพียงพอต่อการนำไปสู่การออกแบบ พัฒนา และทดสอบ

2.1.4.4 ข้อมูลสนับสนุน

ข้อมูลสนับสนุน (Supporting Information) คือข้อมูลสนับสนุนเพื่อความสะดวกในการใช้งานคุณลักษณะความต้องการ ประกอบไปด้วย สารบัญ ดัชนี และภาคผนวก

แนวทางการอธิบายคุณลักษณะความต้องการของซอฟต์แวร์ตามมาตรฐาน IEEE Std 830-1998 มีรูปแบบดังภาพประกอบ 2.1 (IEEE, 1998) ซึ่งสามารถประยุกต์ส่วนคุณลักษณะความต้องการให้เหมาะกับการใช้งานได้ตามภาพประกอบ 2.2 และ 2.3 (IEEE, 1998)

1. Introduction
- 1.1. Purpose
- 1.2. Scope
- 1.3. Definitions, acronyms, and abbreviations
- 1.4. References
- 1.5. Overview

2. Overall description
- 2.1. Product perspective
- 2.2. Product functions
- 2.3. User characteristics
- 2.4. Constraints
- 2.5. Assumptions and dependencies

3. Specific requirements

Appendices
Index

ภาพประกอบ 2.1 รูปแบบข้อกำหนดความต้องการด้านซอฟต์แวร์ตามมาตรฐาน IEEE Std 830-1998

3. Specific requirements
- 3.1. External interface requirements
  - 3.1.1. User interfaces
  - 3.1.2. Hardware interfaces
  - 3.1.3. Software interfaces
  - 3.1.4. Communications interfaces
- 3.2. Functional requirements
  - 3.2.1. Mode 1
    - 3.2.1.1. Functional requirement 1.1
  - 3.2.2. Mode 2
    - 3.2.1.1. Functional requirement 2.1
- 3.3. Performance requirements
- 3.4. Design constraints
- 3.5. Software system attributes
- 3.6. Other requirements

ภาพประกอบ 2.2 รูปแบบข้อกำหนดความต้องการด้านซอฟต์แวร์ส่วนที่ 3 จัดระเบียบตามหมวดหมู่การใช้งาน

3. Specific requirements
- 3.1. External interface requirements
  - 3.1.5. User interfaces
  - 3.1.6. Hardware interfaces
  - 3.1.7. Software interfaces
  - 3.1.8. Communications interfaces
- 3.2. Classes/Objects
  - 3.2.1. Classes/Objects 1
    - 3.2.1.1. Attributes
    - 3.2.1.2. Functions
    - 3.2.1.3. Messages
  - 3.2.2. Classes/Objects 2
- 3.3. Performance requirements
- 3.4. Design constraints
- 3.5. Software system attributes
- 3.6. Other requirements

ภาพประกอบ 2.3 รูปแบบข้อกำหนดความต้องการด้านซอฟต์แวร์ส่วนที่ 3 จัดระเบียบตามอ็อบเจกต์

2.2 การทดสอบซอฟต์แวร์

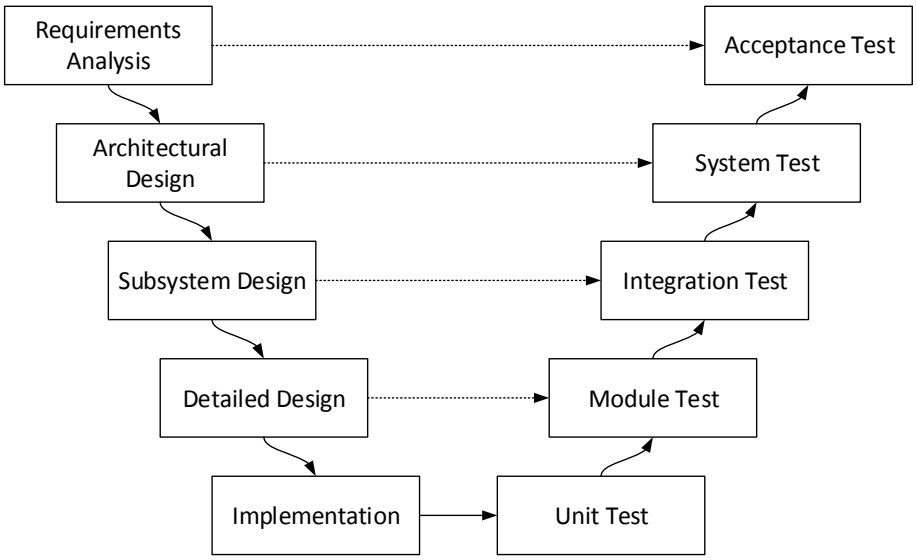
การทดสอบซอฟต์แวร์ (Software Testing) เป็นกิจกรรมที่จัดขึ้นเพื่อประเมินคุณภาพของซอฟต์แวร์ มีวัตถุประสงค์เพื่อให้มั่นใจว่าซอฟต์แวร์จะทำงานตามความต้องการและไม่ทำในสิ่งที่ไม่ควรทำหรือสิ่งที่ไม่ต้องการ โดยการดำเนินการทดสอบซอฟต์แวร์ด้วยชุดข้อมูลที่กำหนดเพื่อหาข้อผิดพลาดและปัญหาที่อาจเกิดขึ้น แล้วทำการแก้ไขข้อผิดพลาดหรือปัญหาดังกล่าว

2.2.1 ระดับของการทดสอบซอฟต์แวร์

การทดสอบซอฟต์แวร์สามารถทำควบคู่ไปกับกิจกรรมต่างๆ ในกระบวนการพัฒนาซอฟต์แวร์ได้ทุกขั้นตอน ตั้งแต่ขั้นตอนรับความต้องการไปจนถึงขั้นตอนการส่งมอบผลิตภัณฑ์ซอฟต์แวร์ ซึ่งในแต่ละกิจกรรมนั้นก็จะมีวิธีการทดสอบซอฟต์แวร์ที่แตกต่างกันออกไป สามารถแสดงให้เห็นถึงความสัมพันธ์ของกิจกรรมการผลิตซอฟต์แวร์กับการทดสอบซอฟต์แวร์ตามแบบจำลองวี (V-Model) แสดงดังภาพประกอบ 2.4 (Ammann, 2008)

2.2.1.1 การทดสอบระดับหน่วยซอฟต์แวร์

การทดสอบระดับหน่วยซอฟต์แวร์ (Unit Testing) เป็นการทดสอบเพื่อตัดสินความถูกต้องและหาข้อผิดพลาดของหน่วยโปรแกรม ถือว่าเป็นการทดสอบระดับต่ำที่สุด เพราะมักมีการพิจารณาจากรหัสต้นฉบับ ถูกดำเนินการโดยโปรแกรมเมอร์ โดยการทดสอบระดับนี้สามารถสร้างกรณีทดสอบและทำการทดสอบได้ในขั้นตอนการลงมือพัฒนา



ภาพประกอบ 2.4 แบบจำลองวีแสดงระดับการทดสอบซอฟต์แวร์และขั้นตอนการพัฒนาซอฟต์แวร์ (Ammann, 2008)

2.2.1.2 การทดสอบส่วนโมดูล

การทดสอบส่วนโมดูล (Module Testing) เป็นการทดสอบเพื่อตัดสินการทำงานของแต่ละโมดูลที่มีการเชื่อมโยงหน่วยโปรแกรมในกลุ่มเข้าด้วยกันว่ามีข้อผิดพลาดหรือไม่ สามารถทำงานได้ตรงตามที่ต้องการหรือไม่ หรือการเชื่อมต่อประสานระหว่างระบบย่อยเข้าด้วยกันถูกดำเนินการโดยโปรแกรมเมอร์ โดยการทดสอบระดับนี้สามารถสร้างกรณีทดสอบได้ตั้งแต่ขั้นตอนการออกแบบรายละเอียด และจะสามารถทดสอบได้เมื่อพัฒนาแต่ละโมดูลเสร็จ

2.2.1.3 การทดสอบการเชื่อมประสาน

การทดสอบการเชื่อมประสาน (Integration Testing) เป็นการทดสอบเพื่อตัดสินว่าการเชื่อมต่อ และการติดต่อสื่อสารระหว่างโมดูลภายในระบบย่อยมีข้อผิดพลาดหรือไม่ ถูกดำเนินการทดสอบโดยสมาชิกในทีมพัฒนาซอฟต์แวร์ โดยการทดสอบระดับนี้สามารถสร้างกรณีทดสอบได้ตั้งแต่ขั้นตอนการออกแบบระบบย่อย และจะสามารถทดสอบได้เมื่อพัฒนาระบบย่อยเสร็จแล้ว

2.2.1.4 การทดสอบทั้งระบบ

การทดสอบทั้งระบบ (System Testing) เป็นการทดสอบขั้นสุดท้ายก่อนส่งมอบผลิตภัณฑ์ให้ลูกค้า เพื่อตัดสินว่าระบบที่นำระบบย่อยมารวมกันแล้วสามารถทำงานได้ตรงตามข้อกำหนดของซอฟต์แวร์หรือไม่ การทดสอบระดับนี้มีเพื่อหาข้อผิดพลาดจากการออกแบบและจากข้อกำหนดของซอฟต์แวร์ ในการทดสอบระดับนี้มักจะไม่ให้ผู้พัฒนาเป็นผู้ทำการทดสอบ แต่จะให้ทีมงานทดสอบจากภายนอกมาเป็นผู้ทดสอบ โดยการทดสอบระดับนี้สามารถสร้างกรณีทดสอบได้ตั้งแต่ขั้นตอนการออกแบบสถาปัตยกรรมของระบบ และจะสามารถทดสอบได้เมื่อมีการพัฒนาระบบเสร็จแล้ว

2.2.1.5 การทดสอบเพื่อการยอมรับ

การทดสอบเพื่อการยอมรับ (Acceptance Testing) เป็นการทดสอบเพื่อตัดสินว่าซอฟต์แวร์ที่ผลิตออกมาเสร็จแล้วนั้นตอบสนองตรงตามความต้องการของผู้ใช้หรือไม่ หรือตรวจสอบว่าซอฟต์แวร์ทำในสิ่งที่ผู้ใช้ต้องการหรือไม่ ในการทดสอบระดับนี้ต้องใช้ผู้ใช้ หรือกลุ่มบุคคลที่มีความรู้ ความชำนาญ ในด้านที่เกี่ยวข้องกับขอบเขตการใช้งานซอฟต์แวร์มาเป็นผู้ทำการทดสอบ โดยการทดสอบระดับนี้สามารถสร้างกรณีทดสอบได้ตั้งแต่ขั้นตอนการวิเคราะห์ความต้องการและจะสามารถทดสอบได้เมื่อมีการพัฒนาซอฟต์แวร์จนเสร็จสมบูรณ์ อยู่ในขั้นการส่งมอบให้ผู้ใช้

2.2.2 กระบวนการทดสอบซอฟต์แวร์

ในการทดสอบซอฟต์แวร์นั้นจะมีกระบวนการดังต่อไปนี้ (Sommerville, 2011)

2.2.2.1 การออกแบบกรณีทดสอบ

การออกแบบกรณีทดสอบ (Test Case) เป็นการออกแบบตัวกำหนดข้อมูลนำเข้าและผลลัพธ์ที่คาดหวังของสถานการณ์ต่างๆ ในการทำงานของระบบ

2.2.2.2 การเตรียมข้อมูลทดสอบ

การเตรียมข้อมูลทดสอบ (Test Data) เป็นการสร้างข้อมูลทดสอบโดยมีกรณีทดสอบเป็นตัวกำหนดรูปแบบและประเภทข้อมูล ซึ่งข้อมูลทดสอบนี้จะเป็นข้อมูลจริงที่ใช้ในการทดสอบระบบ

2.2.2.3 การใช้งานซอฟต์แวร์ด้วยข้อมูลทดสอบ

ใช้งานซอฟต์แวร์ด้วยข้อมูลทดสอบ เป็นการทดสอบการทำงานของซอฟต์แวร์โดยการสั่งใช้งานซอฟต์แวร์และใช้ข้อมูลทดสอบที่เตรียมไว้เป็นข้อมูลนำเข้า

2.2.2.4 การเปรียบเทียบผลลัพธ์กับกรณีทดสอบ

การเปรียบเทียบผลลัพธ์กับกรณีทดสอบ เป็นการนำผลลัพธ์ที่ได้จากการทดสอบการใช้งานซอฟต์แวร์มาเปรียบเทียบกับผลลัพธ์ที่คาดหวังที่ของกรณีทดสอบว่ามีความสอดคล้องกันหรือไม่

2.2.3 เทคนิคการทดสอบซอฟต์แวร์

ในการทดสอบซอฟต์แวร์แต่ละระดับนั้นจะมีเทคนิคที่ใช้แตกต่างกันตามความเหมาะสม ปัจจุบันเทคนิคที่นิยมใช้มี 2 วิธี ดังนี้ (Glenford, 2012)

2.2.3.1 การทดสอบแบบกล่องขาว

การทดสอบแบบกล่องขาว (White-Box Testing) เป็นการทดสอบเชิงโครงสร้างที่เน้นไปที่การทดสอบการทำงานภายในโปรแกรม โดยใช้การเปรียบเทียบเป็นกล่องสีขาวที่สามารถมองเห็นรายละเอียดข้างในได้ การสร้างกรณีทดสอบมักวิเคราะห์จากรหัสต้นฉบับ (Source Code)

หรือขั้นตอนวิธี (Algorithm) ของโปรแกรม การทดสอบด้วยเทคนิคนี้จำเป็นต้องให้ผู้ที่มีความรู้ความชำนาญในเชิงเทคนิคเป็นผู้ดำเนินการทดสอบ เนื่องจากมีการพิจารณาลงไปถึงระดับรหัสต้นแบบ ขั้นตอนวิธี หรือการดำเนินการทางตรรกะ ซึ่งมีความซับซ้อน ละเอียดอ่อน และต้องใช้ความรู้ในเชิงเทคนิคสูง

2.2.3.2 การทดสอบแบบกล่องดำ

การทดสอบแบบกล่องดำ (Black-Box Testing) เป็นการทดสอบเชิงฟังก์ชัน มีตัวขับเคลื่อนการทดสอบเป็นข้อมูล ทั้งข้อมูลนำเข้าและข้อมูลส่งออก (Input/Output Driven) ทำการทดสอบโดยอ้างอิงจากคำอธิบายรายละเอียดและความต้องการของซอฟต์แวร์ มีวัตถุประสงค์เพื่อค้นหาส่วนที่ทำงานผิดพลาดไปจากรายละเอียดของข้อกำหนดคุณลักษณะซอฟต์แวร์ ดำเนินการทดสอบโดยไม่สนใจขั้นตอนการทำงานหรือพฤติกรรมภายในโปรแกรม จะสนใจเฉพาะผลลัพธ์ที่ได้เท่านั้น ดังนั้น การทดสอบด้วยเทคนิคนี้จึงไม่จำเป็นต้องให้ผู้ที่มีความรู้ในเชิงเทคนิคเป็นผู้ดำเนินการ แต่เป็นผู้ที่มีความรู้ในด้านรายละเอียดความต้องการของซอฟต์แวร์ เช่น ผู้ใช้ระบบ การทดสอบด้วยเทคนิคนี้เปรียบเทียบเป็นกล่องดำที่มองไม่เห็นว่าข้างในเป็นอะไรหรือมีอะไรอยู่ เนื่องจากการทดสอบแบบ Black-Box ใช้ข้อมูลนำเข้าและข้อมูลส่งออกเป็นตัวขับเคลื่อน ดังนั้น หากต้องการทำการทดสอบแบบละเอียดจะต้องใช้ข้อมูลที่เป็นไปได้ทั้งหมดเป็นข้อมูลนำเข้าและข้อมูลส่งออก ซึ่งในทางปฏิบัติจริงเป็นไปได้ยาก หรือเป็นไปไม่ได้เลย ดังนั้น การสร้างกรณีทดสอบมักพิจารณาถึงความครอบคลุมในส่วนทำงานที่สำคัญ ซึ่งระบุไว้ในข้อกำหนดคุณลักษณะของซอฟต์แวร์

2.2.4 หลักเกณฑ์ในการสร้างกรณีทดสอบแบบ White-Box Testing

2.2.4.1 การครอบคลุมคำสั่ง

การครอบคลุมคำสั่ง (Statement Coverage) เป็นเกณฑ์ในการสร้างกรณีทดสอบโดยมีหลักการว่า ให้ทุกๆ คำสั่งภายในโปรแกรมได้ทำงานอย่างน้อย 1 ครั้ง เช่น

```
private int func1(int A, int B, int C){
        int result = 0;
        if( (A % 2) == 0) && (B > 0) )
                result++;
        if( (A == 0) || (C == 0) )
                result = 0;
        return result;
}
```

ภาพประกอบ 2.5 ตัวอย่างชุดคำสั่งภาษา C#

จากชุดคำสั่งในภาพประกอบ 2.5 ถ้ากำหนดให้

A = 4, B = 2, C = 0

จะเป็นหนึ่งกรณีทดสอบที่สามารถเรียกให้ทุกๆ คำสั่งในชุดคำสั่งได้ทำงาน

2.2.4.2 การครอบคลุมการตัดสินใจ

การครอบคลุมการตัดสินใจ (Decision Coverage) เป็นเกณฑ์ในการสร้างกรณีทดสอบโดยมีหลักการว่า ให้ทุกๆ การตัดสินใจภายในโปรแกรม เช่น switch-case do-while if-else มีผลลัพธ์เป็น จริง (True) และ เท็จ (False) อย่างน้อยอย่างละ 1 ครั้ง

จากชุดคำสั่งในภาพประกอบ 2.5 ถ้ากำหนดให้

A = 4, B = 1, C = 3

จะได้เหตุการณ์ว่า การตัดสินใจที่1 ให้ผลลัพธ์เป็นจริง การตัดสินใจที่ 2 ให้ผลลัพธ์เป็นเท็จ (1) และถ้ากำหนดให้

A = 0, B =0, C = 0

จะได้เหตุการณ์ว่า การตัดสินใจที่ 1 ให้ผลลัพธ์เป็นเท็จ การตัดสินใจที่ 2 ให้ผลลัพธ์เป็นจริง (2) เมื่อนำกรณีทดสอบ (1) และ (2) มาทดสอบ ก็จะได้ตามเกณฑ์การครอบคลุมการตัดสินใจ

2.2.4.3 การครอบคลุมเงื่อนไข

การครอบคลุมเงื่อนไข (Condition Coverage) เป็นเกณฑ์ในการสร้างกรณีทดสอบโดยมีหลักการว่า ให้ทุกๆ ผลลัพธ์ที่เป็นไปได้ของเงื่อนไขได้รับการทดสอบอย่างน้อย 1 ครั้ง

จากชุดคำสั่งในภาพประกอบ 2.5 มี 4 เงื่อนไข : A หาร 2 ลงตัว, B > 0, A = 0, และ C = 0 ดังนั้น กรณีทดสอบตามเกณฑ์การครอบคุลมเงื่อนไข จะต้องกำหนดกรณีทดสอบให้ A หาร 2 ลงตัว, A หาร 2 ไม่ลงตัว, A = 0, A ≠ 0, B > 0, B <= 0, C = 0, และ C ≠ 0 เช่น ถ้ากำหนดให้

A = 0, B = 1, C = 0

จะได้ว่า ทุกเงื่อนไขในการตัดสินใจให้ผลลัพธ์เป็นจริง (1) และถ้ากำหนดให้

A = 1, B = 0, C = 1

จะได้ว่า ทุกเงื่อนไขในการตัดสินใจให้ผลลัพธ์เป็นเท็จ (2) เมื่อนำกรณีทดสอบ (1) และ (2) มาทดสอบ ก็จะได้ตามเกณฑ์การครอบคลุมเงื่อนไข

2.2.4.4 การครอบคลุมเงื่อนไขและการตัดสินใจ

การครอบคลุมเงื่อนไขและการตัดสินใจ (Decision/Condition Coverage) เป็นเกณฑ์ในการสร้างกรณีทดสอบโดยมีหลักการว่า ผลลัพธ์ที่เป็นไปได้ของแต่ละเงื่อนไข และผลลัพธ์ที่เป็นไปได้ของแต่ละการตัดสินใจ ได้รับการทดสอบอย่างน้อย 1 ครั้ง

จากชุดคำสั่งในภาพประกอบ 2.5 มี 4 เงื่อนไข : A หาร 2 ลงตัว, B > 0, A = 0, และ C = 0 และมี 2 การตัดสินใจ : ((A % 2) == 0) && (B > 0)) และ ((A == 0) || (C == 0)) ดังนั้น กรณีทดสอบตามเกณฑ์การครอบคลุมเงื่อนไขและการตัดสินใจ จะต้องกำหนดกรณีทดสอบให้ครอบคลุมผลลัพธ์การตัดสินใจ (A หาร 2 ลงตัว และ B > 0), (A = 0 หรือ C = 0), (A หาร 2 ไม่ลงตัว และ B <= 0), และ (A ≠ 0 หรือ C ≠ 0) และครอบคลุมผลลัพธ์ของเงื่อนไข (A หาร 2 ลงตัว), (A หาร 2 ไม่ลงตัว), (A = 0), (A ≠ 0), (B > 0), (B <= 0), (C = 0), และ (C ≠ 0) อย่างน้อย 1 กรณี เช่น ถ้ากำหนดให้

A = 3, B = 1, C = 2

จะได้เหตุการณ์ว่า การตัดสินใจที่ 1 ให้ผลลัพธ์เป็นเท็จ การตัดสินใจที่ 2 ให้ผลลัพธ์เป็นเท็จ (1) โดยมีผลลัพธ์ของทุกเงื่อนไขเป็นเท็จ และถ้ากำหนดให้

A = 0, B = 1, C= 0

จะได้เหตุการณ์ว่า การตัดสินใจที่ 1 ให้ผลลัพธ์เป็นจริง การตัดสินใจที่ 2 ให้ผลลัพธ์เป็นจริง (2) โดยมีผลลัพธ์ของทุกเงื่อนไขเป็นจริง เมื่อนำกรณีทดสอบ (1) และ (2) มาทดสอบก็จะได้ตามเกณฑ์การครอบคลุมเงื่อนไขและการตัดสินใจ

2.2.4.5 การครอบคุลมหลายเงื่อนไข

การครอบคลุมหลายเงื่อนไข (Multi-Condition Coverage) เป็นเกณฑ์ในการสร้างกรณีทดสอบโดยมีหลักการว่า ผลลัพธ์ทั้งหมดที่เป็นไปได้ของทุกเงื่อนไขในทุกการตัดสินใจได้รับการทดสอบอย่างน้อย 1 ครั้ง เช่น

จากชุดคำสั่งในภาพประกอบ 2.5 กรณีทดสอบตามเกณฑ์การครอบคลุมหลายเงื่อนไข ต้องครอบคลุมกรณีดังต่อไปนี้

A หาร 2 ลงตัว, B > 0
A หาร 2 ลงตัว, B <= 0
A หาร 2 ไม่ลงตัว, B > 0
A หาร 2 ไม่ลงตัว, B <= 0
A = 0, C = 0
A = 0, C ≠ 0
A ≠ 0, C = 0
A ≠ 0, C ≠ 0

เช่น ถ้ากำหนดให้ข้อมูลที่นำมาใช้เป็นกรณีทดสอบต่อไปนี้

A = 0, B = 1, C = 0

จะได้เหตุการณ์ว่า การตัดสินใจที่ 1 ให้ผลลัพธ์เป็นจริง การตัดสินใจที่ 2 ให้ผลลัพธ์เป็นจริง (1) โดยมีผลลัพธ์ของทุกเงื่อนไขเป็นจริง ถ้ากำหนดให้

A = 0, B = 0, C = 0

จะได้เหตุการณ์ว่า การตัดสินใจที่ 1 ให้ผลลัพธ์เป็นเท็จ การตัดสินใจที่ 2 ให้ผลลัพธ์เป็นจริง (2) โดยมีผลลัพธ์ของเงื่อนไขคือ A เป็นจริง B เป็นเท็จ C เป็นจริง ถ้ากำหนดให้

A = 3, B = 2, C = 0

จะได้เหตุการณ์ว่า การตัดสินใจที่ 1 ให้ผลลัพธ์เป็นเท็จ การตัดสินใจที่ 2 ให้ผลลัพธ์เป็นจริง (3) โดยมีผลลัพธ์ของเงื่อนไขคือ A เป็นเท็จ B เป็นจริง C เป็นจริง และถ้ากำหนดให้

A = 1, B = 0, C = 3

จะได้เหตุการณ์ว่า การตัดสินใจที่ 1 ให้ผลลัพธ์เป็นเท็จ การตัดสินใจที่ 2 ให้ผลลัพธ์เป็นเท็จ (4) โดยมีผลลัพธ์ของทุกเงื่อนไขเป็นเท็จ เมื่อนำกรณีทดสอบ (1) ถึง (4) มาทดสอบ ก็จะได้ตามเกณฑ์การครอบคลุมหลายเงื่อนไข

2.2.5 หลักเกณฑ์ในการสร้างกรณีทดสอบแบบ Black-Box Testing

2.2.5.1 การแบ่งส่วนอย่างสมมูล

การแบ่งส่วนอย่างสมมูล (Equivalence Partitioning) เป็นเกณฑ์การสร้างกรณีทดสอบโดยมีวัตถุประสงค์ในการลดขนาดของกรณีทดสอบลงให้มีขนาดเล็ก แต่ให้ผลการทดสอบ

ที่ครอบคลุมกรณีทดสอบอื่นๆ ที่เป็นไปได้ ด้วยการสร้างคลาสสมมูล (Equivalence Class) ซึ่งประกอบด้วยคลาสสมมูลที่สมเหตุสมผล (Valid Equivalence Class) และคลาสสมมูลที่ไม่สมเหตุสมผล (Invalid Equivalence Class) มีแนวทางในการสร้างดังนี้

1) ถ้าเงื่อนไขของข้อมูลนำเข้าเป็นช่วงของค่าใดๆ ให้ระบุคลาสสมมูลที่สมเหตุสมผล 1 คลาส และคลาสสมมูลที่ไม่สมเหตุสมผล 2 คลาส เช่น จากเงื่อนไข

จำนวนของวัตถุที่สามารถเป็นไปได้ มีตั้งแต่ 1 ถึง 999

จะได้ว่า คลาสสมมูลที่สมเหตุสมผล คือ (1 <= จำนวนของวัตถุ <= 999) และคลาสสมมูลที่ไม่สมเหตุสมผล คือ (จำนวนของวัตถุ < 1) และ (จำนวนของวัตถุ > 999)

2) ถ้าเงื่อนไขของข้อมูลนำเข้าเป็นเป็นการระบุจำนวนของค่า (Number of Values) ให้ระบุคลาสสมมูลที่สมเหตุสมผล 1 คลาส และคลาสสมมูลที่ไม่สมเหตุสมผล 2 คลาส เช่น จากเงื่อนไข

รถ 1 คันสามารถมีเจ้าของได้ 1 ถึง 6 คน

คลาสสมมูลที่สมเหตุสมผล คือ (มีเจ้าของ 1 ถึง 6 คน) คลาสสมมูลที่ไม่สมเหตุสมผลคือ (ไม่มีเจ้าของ) และ (มีเจ้าของมากกว่า 6 คน)

3) ถ้าเงื่อนไขของข้อมูลนำเข้าระบุเซตของข้อมูลนำเข้า ให้ระบุคลาสสมมูลที่สมเหตุสมผลตามสมาชิกแต่ละตัว และคลาสสมมูลที่ไม่สมเหตุสมผล 1 คลาสเป็นข้อมูลที่ไม่อยู่ในเซต เช่น จากเงื่อนไข

ประเภทของรถ คือ BUS, TRUCK, TAXICAB, CAR, MOTOCYCLE

คลาสสมมูลที่ไม่สมเหตุสมผลคือ (TRAILER) เป็นต้น

4) ถ้าเงื่อนไขของข้อมูลนำเข้าระบุว่า “ต้องเป็น....” เช่น จากเงื่อนไข

ตัวอักขระตัวแรกของชื่อต้องเป็นพยัญชนะ

ให้ระบุคลาสสมมูลที่สมเหตุสมผล 1 คลาส (เป็นพยัญชนะ) และคลาสสมมูลที่ไม่สมเหตุสมผล 1 คลาส (เป็นอักขระใดๆ ที่ไม่ใช่พยัญชนะ)

2.2.5.2 การวิเคราะห์ขอบเขตของข้อมูล

การวิเคราะห์ขอบเขตของข้อมูล (Boundary Value Analysis) เป็นเกณฑ์การสร้างกรณีทดสอบโดยพิจารณาจากเงื่อนไขของขอบเขต (Boundary Conditions) โดยการพิจารณาแบ่งออกเป็น 3 กรณี คือ อยู่พอดีกับขอบเขต อยู่เหนือขอบเขต และอยู่ใต้ขอบเขต มีแนวทางในการสร้างดังต่อไปนี้

1) ถ้าเงื่อนไขของข้อมูลนำเข้าระบุพิสัย (Range) ของข้อมูล ให้สร้างกรณีทดสอบที่ปลายของพิสัย และสร้างกรณีทดสอบด้วยข้อมูลนำเข้าที่มีค่าเกินปลายของพิสัย เช่น ถ้าเงื่อนไขของข้อมูลนำเข้าระบุพิสัย คือ -1.0 ถึง 1.0 จะได้กรณีทดสอบ คือ 1.0, -1.0, -1.001, และ 1.001

2) ถ้าเงื่อนไขของข้อมูลนำเข้าเป็นการระบุค่าของเลขจำนวน ให้สร้างกรณีทดสอบด้วยค่าสูงสุด ค่าต่ำสุด ค่าสูงสุด+1 และค่าต่ำสุด-1 ของจำนวนนั้น เช่น ถ้าเงื่อนไขคือ

ข้อมูลนำเข้าสามารถมีได้ตั้งแต่ 1 ถึง 255 ระเบียน

จะต้องสร้างกรณีทดสอบสำหรับ 0, 1, 255, และ 256 ระเบียน

3) ใช้แนวทางเดียวกับข้อ 1) กับเงื่อนไขของข้อมูลส่งออกแต่ละเงื่อนไข

4) ใช้แนวทางเดียวกับข้อ 2) กับเงื่อนไขของข้อมูลส่งออกแต่ละเงื่อนไข

5) ถ้าข้อมูลนำเข้าหรือข้อมูลส่งออกเป็นเซตแบบเรียงลำดับ (Ordered Set) เช่น แฟ้มข้อมูลแบบตามลำดับ (Sequential File) หรือตาราง ให้เน้นความสนใจไปที่สมาชิกตัวแรกและสมาชิกตัวสุดท้ายภายในเซต

2.2.5.3 การสร้างกราฟเหตุและผล

การสร้างกราฟเหตุและผล (Cause-Effect Graphing) เป็นเกณฑ์ในการสร้างกรณีทดสอบโดยการแสดงความสัมพันธ์ผลลัพธ์และองค์ประกอบที่ทำให้เกิดผลลัพธ์นั้น ใช้รายละเอียดความต้องการซอฟต์แวร์มาสร้างเป็นกราฟ เชื่อมโยงความสัมพันธ์ด้วยหลักการตรรกศาสตร์บูลลีน (Boolean Logic) ซึ่งมีความสัมพันธ์ 4 แบบ คือ

1) เอกลักษณ์ (Identity) เช่น ระบุว่า ถ้า A เป็น 1 แล้ว B ต้องเป็น 1 มิฉะนั้น B เป็น 0

2) นิเสธ (Not) เช่น ระบุว่า ถ้า A เป็น 1 แล้ว B ต้องเป็น 0, มิฉะนั้น B เป็น 1

3) หรือ (Or) เช่น ระบุว่า ถ้า A B หรือ C เป็น 1 แล้ว D ต้องเป็น 1, มิฉะนั้น D เป็น 0

4) และ (And) เช่น ระบุว่า ถ้าทั้ง A และ B เป็น 1 แล้ว C ต้องเป็น 1 มิฉะนั้น C เป็น 0

ตัวอย่างการสร้างกราฟเหตุและผลสามารถกระทำได้ดังแสดงในภาพประกอบ 2.6

2.2.5.4 การเดาข้อผิดพลาด

การเดาข้อผิดพลาด (Error Guessing) เป็นการใช้ความรู้ ประสบการณ์ และสัญชาตญาณของมนุษย์ ในการเดาข้อผิดพลาด โดยจะพิจารณาถึงประเด็นที่มีความเสี่ยงต่อการเกิดข้อผิดพลาด จากนั้น ก็จะสร้างกรณีทดสอบที่ครอบคลุมประเด็นดังกล่าว เนื่องจากเทคนิคและวิธีการเป็นของเฉพาะบุคคล ดังนั้นจึงระบุขั้นตอนออกมาแบบชัดเจนได้ยาก แต่มีแนวคิดพื้นฐาน

คือ ระบุรายการข้อผิดพลาดที่เป็นไปได้ หรือสถานการณ์ที่เสี่ยงต่อการเกิดข้อผิดพลาด จากนั้นก็สร้างกรณีทดสอบตามรายการนั้น

ความต้องการ : “ตัวอักขระในสดมภ์ที่ 1 ต้องเป็นตัวอักษร ‘A’ หรือ ‘B’ตัวอักขระในสดมภ์ที่ 2 ต้องเป็นตัวเลข 1 ตัว การแก้ไขไฟล์จึงจะทำได้ ถ้าตัวอักขระตัวแรกผิด แสดงข้อความเตือน X12 ถ้าตัวอักขระตัวที่สองผิด ให้แสดงข้อความเตือน X13”

เหตุ คือ : 1 ตัวอักขระในสดมภ์ที่ 1 เป็น ‘A’

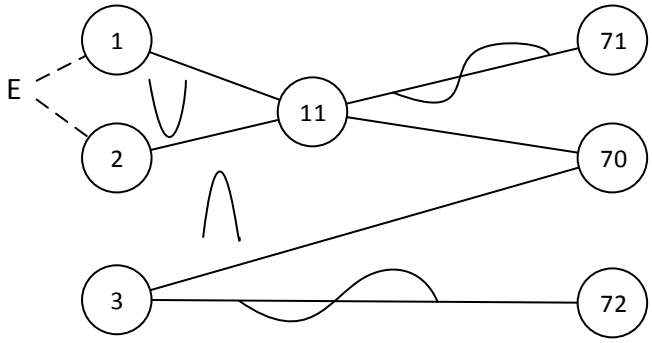
2 ตัวอักขระในสดมภ์ที่ 1 เป็น ‘B’

3 ตัวอักขระในสดมภ์ที่ 2 เป็นตัวเลข 1 ตัว

ผล คือ : 70 เกิดการแก้ไขไฟล์

71 แสดงข้อความเตือน X12

72 แสดงข้อความเตือน X13



จะเห็นได้ว่าโหนดสื่อกลางหมายเลข 11 ถูกสร้างขึ้นมาในขั้นตอนการสร้างกราฟ เพื่อแทนค่าความจริงของการ OR ระหว่าง โหนด 1 กับโหนด 2 โดย ถ้าเป็นจริง ไปโหนด 70 ถ้าเป็นเท็จ ไปโหนด 71 และ มีตัวควบคุม E เพื่อคุมไม่ให้ 1 และ 2 เป็นจริงพร้อมกัน

ภาพประกอบ 2.6 ตัวอย่างการสร้างกราฟเหตุและผลอย่างง่าย (Glenford, 2012)

2.3 การวิเคราะห์และการออกแบบเชิงวัตถุ

การโปรแกรมเชิงวัตถุ (Object-Oriented Programming) คือกระบวนการพัฒนาระบบโดยมีแนวคิดพื้นฐานมาจากอ็อบเจกต์ (Object) หรือวัตถุ มีหลักการว่า ให้มองทุกอย่างภายในระบบเป็นวัตถุ โดยภายในแต่ละอ็อบเจกต์นั้นจะมีลักษณะประจำ (Attribute) และเมท็อด (Method) อ็อบเจกต์จะทำงานร่วมกันโดยการส่งสาร (Message) หากัน และมีความสัมพันธ์ระหว่างกันในรูปแบบต่างๆ และเนื่องจากการโปรแกรมเชิงวัตถุนั้นมีแนวคิดพื้นฐานที่แตกต่างจากการโปรแกรมแบบดั้งเดิม ดังนั้นจึงมีวิธีการโปรแกรมที่แตกต่างออกไป นั่นคือจะใช้การวิเคราะห์เชิงวัตถุ (Object-Oriented Analysis) และการออกแบบเชิงวัตถุ

(Object-Oriented Design) ในการพัฒนาระบบแทนวิธีการแบบดั้งเดิมนั่นเอง การโปรแกรมเชิงวัตถุได้ถูกพัฒนาขึ้นมาพร้อมกับภาษาเชิงวัตถุ (Object-Oriented Language) ซึ่งถือเป็นภาษาเชิงโปรแกรมยุคที่ 3 ในช่วงท้ายๆ มีวิธีการมาจากการออกแบบให้ข้อมูลเป็นตัวขับเคลื่อน (Data-Driven Design Methods) ในลักษณะข้อมูลแบบนามธรรม (Data Abstraction) เป็นตัวต้นแบบเพื่อจะนำมาขยายความและสืบทอดคุณสมบัติออกมาเป็นคลาส หรืออ็อบเจกต์ชิ้นใหม่เพื่อแก้ปัญหาตามที่ระบบต้องการต่อไป ทำให้การโปรแกรมเชิงวัตถุมีความยืดหยุ่นสูง สามารถนำมาใช้ในกระบวนการพัฒนาซอฟต์แวร์ได้ด้วยภาษาเชิงวัตถุใดๆ

2.3.1 ลักษณะพื้นฐานเฉพาะของระบบเชิงวัตถุ

ลักษณะพื้นฐานเฉพาะของระบบเชิงวัตถุ (Characteristics of Object-Oriented Systems) มีรายละเอียดดังนี้ (Dennis, 2015)

2.3.1.1 คลาสและอ็อบเจกต์

คลาส (Class) เป็นแม่แบบทั่วไปที่ใช้ในการกำหนดและสร้างอ็อบเจกต์เป็นตัวกำหนดคุณสมบัติ พฤติกรรม และข้อมูลของอ็อบเจกต์

อ็อบเจกต์ (Object) เป็นตัวแทนที่สร้างมาจากคลาส อ็อบเจกต์ที่สร้างจากคลาสเดียวกันจะใช้สมาชิกที่ได้รับสืบทอดมาจากคลาสร่วมกันได้ ซึ่งสมาชิกประกอบไปด้วย ลักษณะประจำตัวดำเนินการ เมท็อด ความสัมพันธ์ และพฤติกรรม โดยสมาชิกที่แต่ละอ็อบเจกต์นั้นใช้สามารถมีรายละเอียดที่แตกต่างกันได้

2.3.1.2 เมท็อดและสาร

เมท็อด (Method) เป็นการกระทำการต่างๆ ภายในอ็อบเจกต์ ทำหน้าที่กำหนดพฤติกรรมของอ็อบเจกต์

สาร (Message) เป็นสารสนเทศที่รับ–ส่งระหว่างอ็อบเจกต์เพื่อสั่งการ และควบคุมการทำงาน เป็นฟังก์ชันการทำงาน หรือลำดับการเรียกจากออบเจกต์หนึ่งถึงอีกออบเจกต์หนึ่ง

2.3.1.3 การห่อหุ้มและการซ่อนสารสนเทศ

การห่อหุ้ม (Encapsulation) เป็นการกำหนดความสามารถการเข้าถึง หรือเปลี่ยนแปลงค่าสมาชิกภายในคลาส ทำให้คลาสอื่นหรืออ็อบเจกต์ที่ไม่ได้รับสิทธิ์ไม่สามารถเข้าถึงหรือแก้ไขสมาชิกนั้นๆได้โดยตรง สามารถเข้าถึงได้ผ่านการเรียกใช้เมท็อดภายในคลาสนั้นเท่านั้น

การซ่อนสารสนเทศ (Information Hiding) เป็นการซ่อนข้อมูลภายในโมดูล โดยจะส่งเฉพาะข้อมูลที่โมดูลอื่นต้องการในรูปแบบสารผ่านการเรียกใช้จากเมท็อดเท่านั้น

2.3.1.4 การสืบทอด

การสืบทอด (Inheritance) คือการที่คลาสแม่ (Superclass) สืบทอดคุณสมบัติมายังคลาสลูก (Subclass) โดยภายในคลาสลูกจะมีลักษณะประจำและเมท็อดที่มาจากคลาสแม่

และสามารถกำหนดใหม่เพิ่มเข้ามาในตัวคลาสลูกเองได้ ทำให้สามารถสร้างคลาสที่มีคุณสมบัติใกล้เคียงกัน แต่มีรายละเอียดการทำงานต่างกันได้

2.3.1.5 ภาวะพหุสัณฐานและการยึดเหนี่ยวแบบพลวัต

ภาวะพหุสัณฐาน (Polymorphism) คือการที่สามารถตอบสนองต่อเมท็อดเดียวกันด้วยวิธีการที่แตกต่างกัน เกิดจากที่คลาสสืบทอดคุณสมบัติมาจากคลาสแม่ทำการแทนที่ (Override) เมท็อดที่ได้รับทอดมา ทำให้มีการกระทำการและผลลัพธ์ที่แตกต่างกันออกไป

การยึดเหนี่ยวแบบพลวัต (Dynamic Binding) คือการกำหนดชนิดของอ็อบเจกต์ในขณะ run-time ซึ่งจะมีความยืดหยุ่นสูงกว่าการยึดเหนี่ยวแบบอพลวัต (Static Binding) ซึ่งกำหนดชนิดตั้งแต่ตอนคอมไพล์ และเป็นตัวสนับสนุนการทำงานของภาวะพหุสัณฐาน

2.3.2 การวิเคราะห์เชิงวัตถุ

การวิเคราะห์เชิงวัตถุ (Object-Oriented Analysis) คือการตอบคำถามว่าใครจะใช้ระบบ ระบบทำอะไรได้ ที่ไหนและเมื่อใด โดยการวิเคราะห์ความต้องการให้อยู่ในรูปแบบของคลาสและอ็อบเจกต์ กำหนดความสัมพันธ์ภายในระบบ โดยการวิเคราะห์เชิงวัตถุนั้นจะต้องไม่ยึดติดกับภาษาหรือข้อจำกัดในการพัฒนาใดๆ โดยมุ่งเน้นความสนใจไปที่การไหลของข้อมูลภายในระบบ การวิเคราะห์เชิงวัตถุเป็นการใช้มุมมองและแนวคิดของโลกเชิงวัตถุซึ่งเป็นโลกในจินตนาการ มาอธิบายและขยายความเป็นแบบจำลองในโลกของความเป็นจริง ประกอบด้วยกิจกรรมดังต่อไปนี้ (Dennis, 2015)

2.3.2.1 การกำหนดความต้องการ

การกำหนดความต้องการ (Requirements Determination) คือการเปลี่ยนความต้องการทางธุรกิจที่อยู่ในรูปแบบคำอธิบายด้วยภาษาทั่วไปให้อยู่ในรูปแบบที่เหมาะต่อการนำมาใช้ในการวิเคราะห์และออกแบบระบบ

2.3.2.2 การสร้างแบบจำลองการดำเนินการทางธุรกิจและฟังก์ชัน

การสร้างแบบจำลองการดำเนินการทางธุรกิจและฟังก์ชัน (Business Process and Functional Modeling) เป็นการสร้างแบบจำลองเพื่ออธิบายการทำงานโต้ตอบของระบบภายใต้สิ่งแวดล้อมการทำงาน ใช้แผนภาพ 2 แบบแทนแบบจำลองในการอธิบาย คือ แผนภาพยูสเคส (Use Case Diagram) ใช้ในการอธิบายฟังก์ชันพื้นฐานของระบบ และแผนภาพกิจกรรม (Activity Diagram) ใช้ในการอธิบายแบบจำลองการดำเนินการทางธุรกิจ

2.3.2.3 การสร้างแบบจำลองเชิงโครงสร้าง

การสร้างแบบจำลองเชิงโครงสร้าง (Structural Modeling) เป็นการสร้างแบบจำลองเพื่ออธิบายโครงสร้างของอ็อบเจกต์ที่ใช้สนับสนุนการดำเนินการทางธุรกิจ แผนภาพที่ใช้แทนแบบจำลองในกิจกรรมนี้ คือ แผนภาพคลาส (Class Diagram) และแผนภาพอ็อบเจกต์ (Object Diagram)

2.3.2.4 การสร้างแบบจำลองเชิงพฤติกรรม

การสร้างแบบจำลองเชิงพฤติกรรม (Behavioral Modeling) เป็นการอธิบายการเคลื่อนไหว หรือการทำงานของระบบที่ใช้สนับสนุนการดำเนินการทางธุรกิจ แผนภาพที่ใช้แทนแบบจำลองที่ในกิจกรรมนี้ คือ แผนภาพลำดับ แผนภาพการสื่อสาร (Communication Diagram) และแผนภาพสถานะ (State Machine Diagram)

2.3.3 การออกแบบเชิงวัตถุ

การออกแบบเชิงวัตถุ (Object-Oriented Design) เป็นการนำแบบจำลองที่ได้จากการวิเคราะห์เชิงวัตถุมาออกแบบและเพิ่มเติมข้อกำหนดต่างๆ ที่ได้มาจากความต้องการที่ไม่เป็นฟังก์ชัน (Non-Functional Requirements) เพื่อให้เกิดเป็นแบบจำลองที่มีความชัดเจนขึ้นทั้งในเชิงตรรกะและในการนำไปพัฒนาจริง และทำให้ระบบที่พัฒนาเหมาะสมและสอดคล้องกับความต้องการ ประกอบด้วยกิจกรรมดังต่อไปนี้ (Dennis, 2015)

2.3.3.1 การทวนสอบและตรวจสอบความสมเหตุสมผลของแบบจำลองการวิเคราะห์ระบบ

การทวนสอบและตรวจสอบความสมเหตุสมผลของแบบจำลองการวิเคราะห์ระบบ (Analysis Models Verification and Validation) เป็นการทวนสอบเพื่อยืนยันความถูกต้องว่าแบบจำลองการวิเคราะห์ระบบนั้นตอบสนองต่อความต้องการอย่างครบถ้วนและถูกต้อง และตรวจสอบว่าแบบจำลองแต่ละตัวมีความสอดคล้องกัน

2.3.3.2 การออกแบบคลาสและเมท็อด

ออกแบบคลาสและเมท็อด (Class and Method Design) เป็นการออกแบบรายละเอียดการทำงานภายในคลาสและเมท็อด รวมไปถึงตัวแปรและการทำงานร่วมกันระหว่างโมดูล

2.3.3.3 การออกแบบชั้นการจัดการข้อมูล

ออกแบบการชั้นจัดการข้อมูล (Data Management Layer Design) เป็นการเลือกรูปแบบโครงสร้างข้อมูลที่เหมาะกับระบบและนำมาประยุกต์ใช้ให้เหมาะกับความต้องการ

2.3.3.4 การออกแบบชั้นส่วนปฏิสัมพันธ์ระหว่างมนุษย์และคอมพิวเตอร์

ออกแบบชั้นส่วนปฏิสัมพันธ์ระหว่างมนุษย์และคอมพิวเตอร์ (Human-Computer Interaction Layer Design) เป็นการออกแบบส่วนติดต่อผู้ใช้ หน้าจอของโปรแกรมที่ใช้ทำงานในระบบ ส่วนที่ใช้รับข้อมูลและแสดงผลสารสนเทศที่ผ่านการประมวลผล

2.3.3.5 การออกแบบชั้นสถาปัตยกรรมเชิงกายภาพ

ออกแบบชั้นสถาปัตยกรรมเชิงกายภาพ (Physical Architecture Layer Design) เป็นการออกแบบในส่วนฮาร์ดแวร์ ซอฟต์แวร์ และระบบเครือข่ายของระบบ โดยอ้างอิงมาจากความต้องการที่ไม่เป็นฟังก์ชันเป็นหลัก เช่น ประสิทธิภาพ ความเร็วในการประมวลผล ความปลอดภัย

2.4 ยูเอ็มแอล

ยูเอ็มแอล (UML : Unified Modeling Language) เป็นแบบจำลองอเนกประสงค์ที่ใช้ช่วยแก้ปัญหาในการพัฒนาซอฟต์แวร์เชิงวัตถุ โดยจะใช้เป็นแบบจำลองแทนระบบเพื่อให้ผู้เกี่ยวข้องสามารถทำความเข้าใจได้ง่ายขึ้น ยูเอ็มแอลสามารถใช้ได้แทบทุกขั้นตอนของโครงการพัฒนาซอฟต์แวร์เชิงวัตถุ ตั้งแต่ขั้นตอนการรับความต้องการไปจนถึงขั้นตอนการส่งมอบ มีจุดมุ่งหมายเพื่อรวมแบบจำลองต่างๆ เข้าไว้ด้วยกันภายใต้มาตรฐานเดียวกัน ประกอบไปด้วยแบบจำลองต่างๆ ที่แสดงทั้งโครงสร้าง พฤติกรรม และการจัดการของระบบตามแต่ที่นักวิเคราะห์ระบบจะใช้นำเสนอในแต่ละด้านที่ต้องการ

2.4.1 ประวัติของยูเอ็มแอล

ยูเอ็มแอลถูกพัฒนาขึ้นเพื่อลดความยุ่งยากและรวมวิธีการพัฒนาซอฟต์แวร์เชิงวัตถุเข้าด้วยกัน โดยหลังจากภาษาเชิงวัตถุถูกประกาศใช้และได้รับการยอมรับในปี 1967 ก็ได้มีนักวิชาการเสนอแนวคิดในการพัฒนาเชิงวัตถุออกมาอย่างมากมาย โดยที่มี 4 วิธีการที่เป็นที่รู้จักดีคือ วิธีการของ Booch, วิธีการ Object Modeling Technique ของ James Rumbaugh, วิธีการ Object-oriented Software Engineering ของ Ivar Jacobson, และวิธีการของ Coad & Yourdon ต่อมาในปี 1994 - 1995 Grady Booch, James Rumbaugh, และ Ivar Jacobson นักวิชาการและเจ้าของแนวคิด 3 คนได้ร่วมมือกันและรวมแนวคิดเข้าเป็นหนึ่งเดียวภายใต้ชื่อยูเอ็มแอล และก่อตั้งองค์กร Object Management Group (OMG) ขึ้นในปี 1996 เพื่อควบคุม ดูแล และพัฒนามาตรฐานของยูเอ็ฒแอล โดยในองค์กรนี้ Booch, Rumbaugh และ Jacobson ได้ร่วมมือกับนักทฤษฎีและนักพัฒนาจากบริษัทต่างๆ มากมาย ประกอบไปด้วย Microsoft, IBM, Oracle, Sun, และ Compaq โดย OMG ได้ให้กำเนิดยูเอ็มแอลเวอร์ชั่น 1.1 ในปี 1997 ปัจจุบัน OMG ได้พัฒนามาตรฐานยูเอ็มแอลเป็นเวอร์ชั่น 2.5.1 ซึ่งได้ประกาศขึ้นมาในปี 2017 (NECTEC, 2542; OMG, 2017)

2.4.2 แบบจำลองยูเอ็มแอล

แบบจำลองยูเอ็มแอลเป็นแบบจำลองที่ใช้สำหรับอธิบายระบบซอฟต์แวร์ให้เห็นภาพอย่างง่าย ทั้งโครงสร้างของระบบ ความสัมพันธ์ และพฤติกรรม ให้อยู่ในรูปแบบที่เรียบง่ายและมีมาตรฐาน โดยแบบจำลองประกอบไปด้วยวัตถุ (Object) ที่ใช้แทนสิ่งหนึ่งสิ่งใดในระบบ เช่น คลาส (Class) หรือแพคเกจ (Package) และความสัมพันธ์ (Relation) ระหว่างวัตถุซึ่งจะถูกแทนด้วยเส้นแบบต่างๆ พร้อมคำอธิบายประกอบ ผู้ที่มีส่วนเกี่ยวข้องทั้งหมดของโครงการสามารถเข้าใจได้ตรงกันว่าระบบในแต่ละส่วนเป็นอย่างไรเพียงแค่มีความรู้พื้นฐานในด้านยูเอ็มแอลหรือในด้านการวิเคราะห์และออกแบบเชิงวัตถุ โดยไม่จำเป็นต้องมีความรู้ในด้านการโปรแกรม และด้วยการใช้สัญลักษณ์ในการอธิบายนั้นสามารถช่วยให้การวิเคราะห์และออกแบบระบบทำได้อย่างสะดวก เนื่องจากแบบจำลองยูเอ็มแอลนั้นไม่ยึดติดกับภาษาใดภาษาหนึ่ง ทำให้นักวิเคราะห์ระบบสามารถนำแบบจำลองยูเอ็มแอลมาใช้กับการวิเคราะห์และออกแบบเชิงวัตถุได้ทุกภาษา โดยแบบจำลอง

ยูเอ็มแอลจะอยู่ในรูปแผนภาพ (Diagram) เนื่องจากระบบซอฟต์แวร์อาจมีขนาดใหญ่และซับซ้อน และมีหลายมุมมองที่สามารถนำเสนอให้ผู้เกี่ยวข้องในแต่ละด้าน ตามหลักการออกแบบงานและวิธีการทางวิศวกรรมศาสตร์ จึงมีการแบ่งมุมมองออกเป็น 4 ประเภท ดังต่อไปนี้ (Rumbaugh, 2005)

2.4.2.1 มุมมองประเภทโครงสร้าง

มุมมองประเภทโครงสร้าง (Structural View) เป็นมุมมองที่ใช้แสดงโครงสร้างของระบบ ประกอบไปด้วย Static View Design View และ Use Case View

2.4.2.2 มุมมองประเภทพลวัตร

มุมมองประเภทพลวัตร (Dynamic View) เป็นมุมมองที่ใช้แสดงพฤติกรรมของระบบ ประกอบไปด้วย State Machine View, Activity View, และ Interaction View

2.4.2.3 มุมมองประเภทกายภาพ

มุมมองประเภทกายภาพ (Physical View) เป็นมุมมองที่ใช้แสดงระบบสถาปัตยกรรม และความสัมพันธ์ของ Hardware และ Software มีเพียงมุมมองเดียว คือ Deployment View

2.4.2.4 มุมมองประเภทการจัดการแบบจำลอง

มุมมองประเภทการจัดการแบบจำลอง (Model Management View) เป็นมุมมองที่ใช้อธิบายการจัดการแบบจำลอง ประกอบไปด้วย Deployment View, Profile, และ Model Management View

2.4.3 ประเภทของแผนภาพยูเอ็มแอล

แผนภาพยูเอ็มแอลสามารถแบ่งออกตามการใช้งานได้เป็น 2 ประเภทหลัก คือ แผนภาพเชิงโครงสร้าง (Structure Diagram) และแผนภาพเชิงพฤติกรรม (Behavior Diagram) ประกอบไปด้วยแผนภาพทั้งหมด 14 แผนภาพ ให้นักวิเคราะห์ระบบสามารถเลือกนำไปใช้เพื่อนำเสนอแบบจำลองของระบบตามที่ต้องการนำเสนอ

2.4.3.1 แผนภาพเชิงโครงสร้าง

แผนภาพโครงสร้าง (Structural Diagram) ประกอบไปด้วยแผนภาพดังต่อไปนี้ (OMG, 2017)

1) Class Diagram
2) Component Diagram
3) Object Diagram
4) Composite Structure Diagram
5) Deployment Diagram

6) Package Diagram

7) Profile Diagram

2.4.3.2 แผนภาพเชิงพฤติกรรม

แผนภาพเชิงพฤติกรรม (Behavioral Diagram) ประกอบด้วยแผนภาพดังต่อไปนี้ (OMG, 2017)

1) Activity Diagram

2) Use Case Diagram

3) State Machine Diagram

4) Sequence Diagram

5) Communication Diagram

6) Interaction Overview Diagram

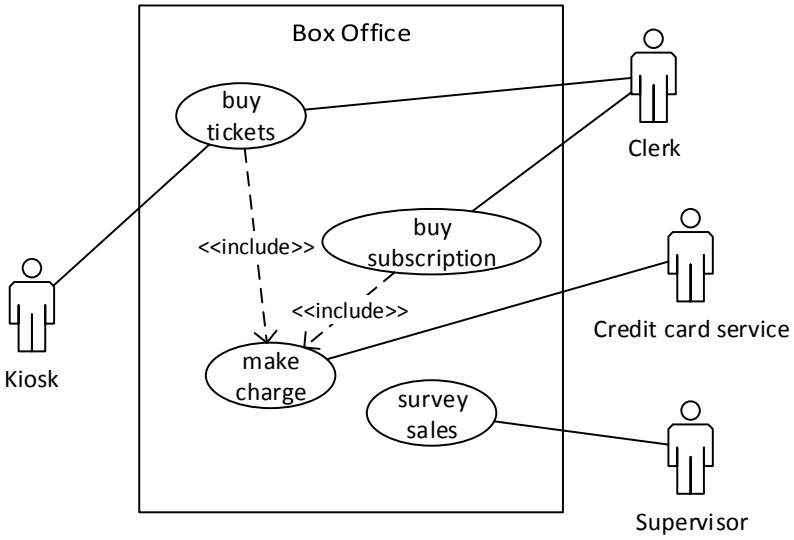
7) Timing Diagram

2.4.4 ตัวอย่างแผนภาพยูเอ็มแอล

แผนภาพยูเอ็มแอลที่มีความสำคัญต่อการวิเคราะห์และออกแบบระบบ ได้แก่ แผนภาพยูสเคส แผนภาพกิจกรรม แผนภาพคลาส แผนภาพอ็อบเจกต์ แผนภาพลำดับ แผนภาพการสื่อสาร และแผนภาพสถานะ เนื่องจากเป็นแบบจำลองที่ใช้อธิบายโครงสร้างและการทำงานพื้นฐานของระบบ มีรายละเอียดดังต่อไปนี้

2.4.4.1 แผนภาพยูสเคส

แผนภาพยูสเคส (Use Case Diagram) เป็นแผนภาพที่ใช้แสดงพฤติกรรมของระบบ ประกอบไปด้วยสัญลักษณ์ที่ใช้แทนฟังก์ชันงานของระบบ เรียกว่า ยูสเคส และสัญลักษณ์แทนผู้กระทำ (Actor) ที่เข้ามามีส่วนเกี่ยวข้องกับแต่ละยูสเคส โดยผู้กระทำนั้นสามารถเป็นได้ทั้งมนุษย์ที่มาใช้ระบบ และระบบอื่นหรือขั้นตอนการทำงานอื่นที่ทำงานร่วมกันกับยูสเคสนี้ แสดงดังภาพประกอบ 2.7 (Rumbaugh, 2005)



ภาพประกอบ 2.7 ตัวอย่างแผนภาพยูสเคส (Rumbaugh, 2005)

2.4.4.2 แผนภาพกิจกรรม

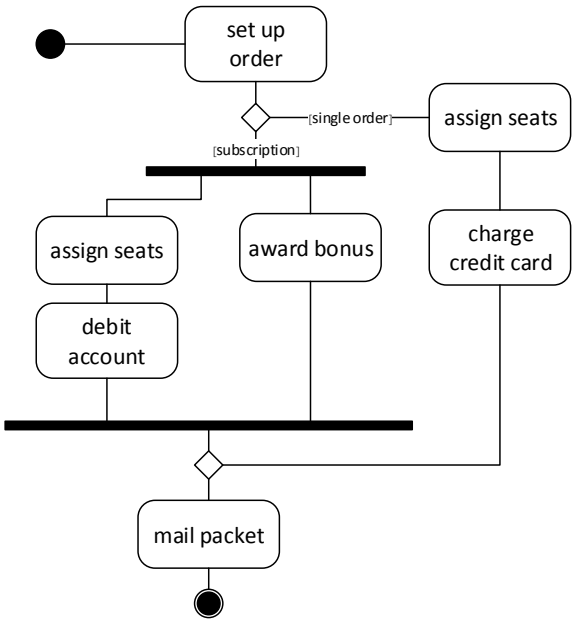
แผนภาพกิจกรรม (Activity Diagram) เป็นแผนภาพที่ใช้แสดงลำดับขั้นตอนในการดำเนินการกิจกรรม (Activity) ในแต่ละโมดูล ทั้งที่เป็นการดำเนินการตามลำดับและการทำงานแบบคู่ขนาน แสดงเงื่อนไขในการตัดสินใจและการควบคุมการทำงาน (Decision) โดยจะอยู่ในรูปแบบโหนดกิจกรรม (Activity Node) ที่เชื่อมกันด้วยลูกศรที่ใช้แสดงทิศทางการทำงาน (Control Flow) โดยกิจกรรมทั้งหมดจะดำเนินการอยู่ระหว่างโหนดเริ่มต้น (Initial Node) และโหนดสิ้นสุด (Finite Node) แสดงดังภาพประกอบ 2.8 (Rumbaugh, 2005) นอกจากนี้สามารถแบ่งช่องระบุกิจกรรมตามผู้ดำเนินกิจกรรมคล้ายช่องในสระว่ายน้ำ (Swimming Lane) เพื่อให้มีความชัดเจนเข้าใจง่ายขึ้น แสดงดังภาพประกอบ 2.9

2.4.4.3 แผนภาพคลาส

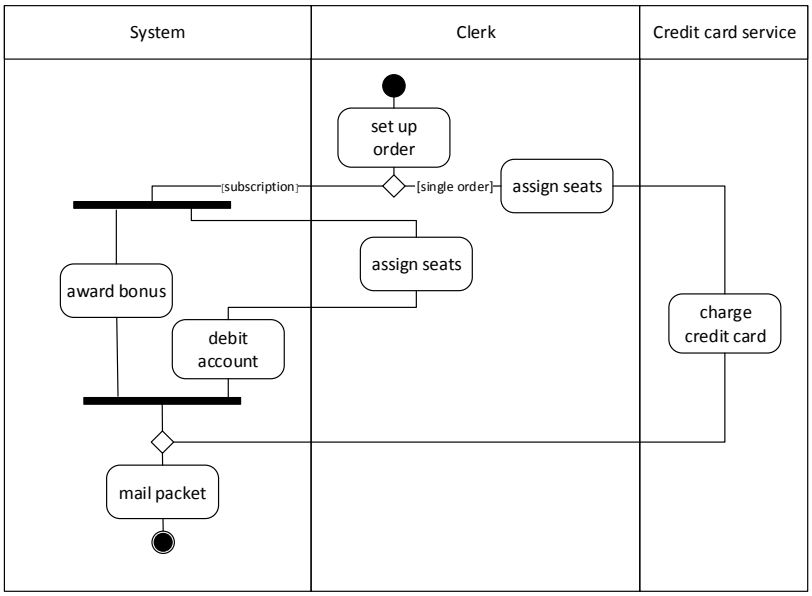
แผนภาพคลาส (Class Diagram) เป็นแผนภาพที่ใช้แสดงโครงสร้างของระบบประกอบไปด้วยสัญลักษณ์ที่ใช้แทนคลาส และเส้นที่ใช้แสดงความสัมพันธ์ระหว่างคลาส โดยภายในคลาสประกอบไปด้วยลักษณะประจำและเมธ็อดของคลาสนั้น แสดงดังภาพประกอบ 2.10 (Rumbaugh, 2005)

2.4.4.4 แผนภาพอ็อบเจกต์

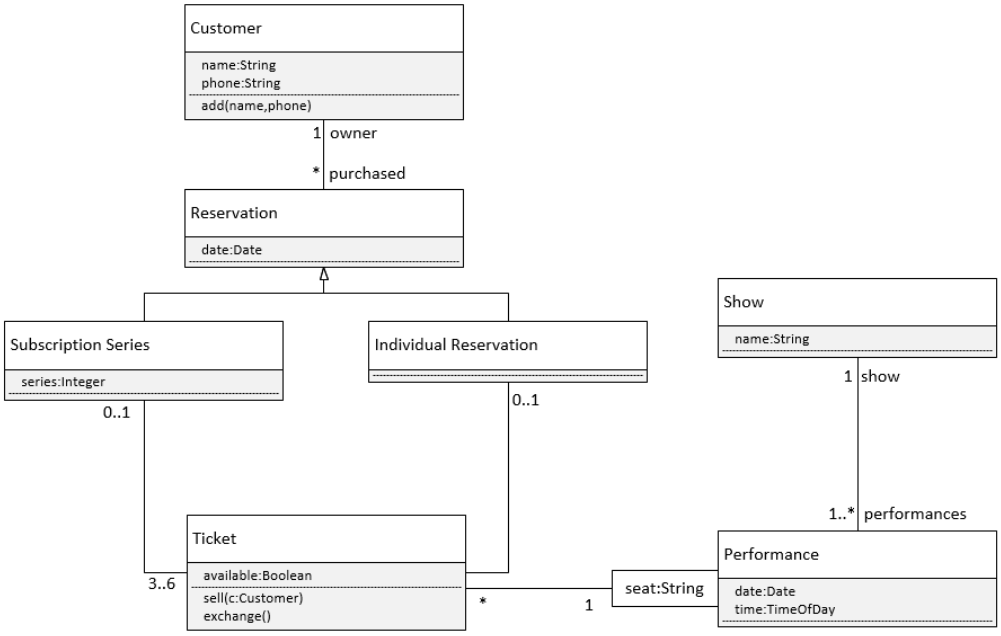
แผนภาพอ็อบเจกต์ (Object Diagram) เป็นแผนภาพที่ใช้แสดงภาพรวมของอ็อบเจกต์ของคลาสและความสัมพันธ์ต่างๆ ในขณะหนึ่ง โดยจะแสดงคุณลักษณะของกรณีตัวอย่าง (Instance Specification) ของอ็อบเจกต์ แสดงดังภาพประกอบ 2.11 (Rumbaugh, 2005)



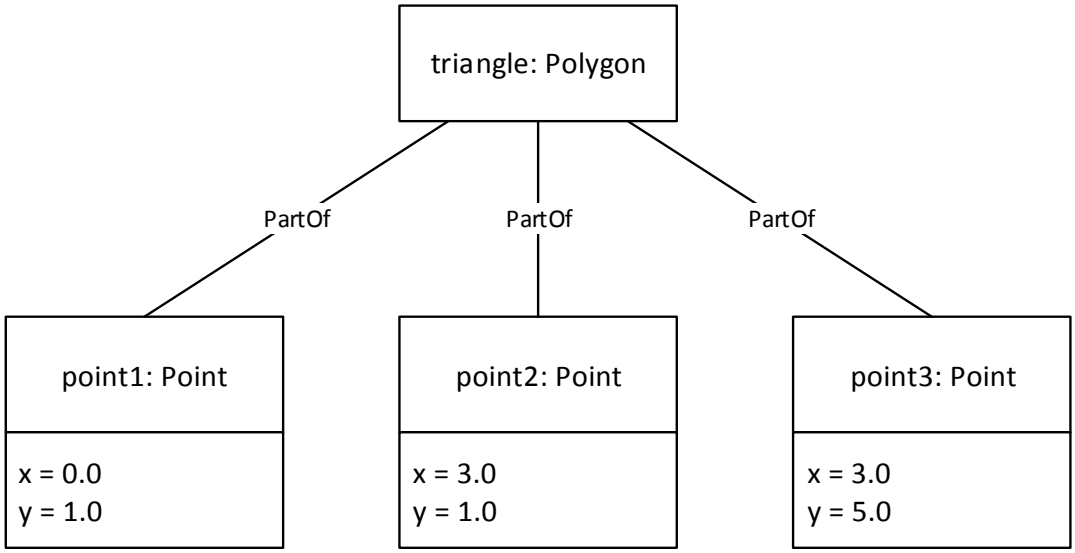
ภาพประกอบ 2.8 ตัวอย่างแผนภาพกิจกรรม (Rumbaugh, 2005)



ภาพประกอบ 2.9 ตัวอย่างแผนภาพกิจกรรมแบบแบ่งช่องระบุกิจกรรมตามผู้ดำเนินกิจกรรม



ภาพประกอบ 2.10 ตัวอย่างแผนภาพคลาส (Rumbaugh, 2005)



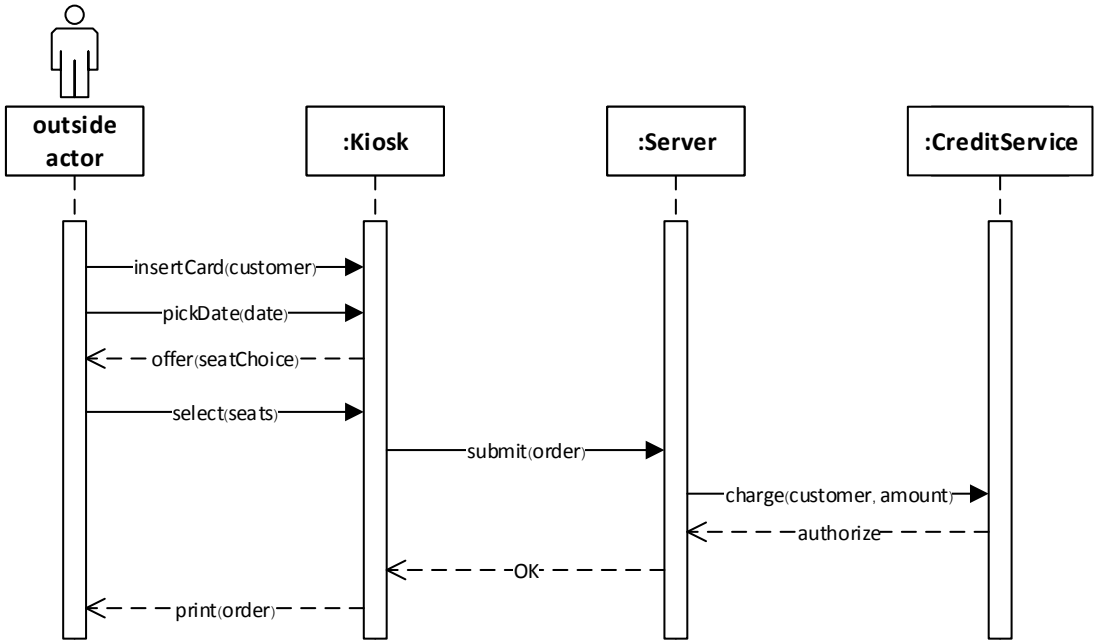
ภาพประกอบ 2.11 ตัวอย่างแผนภาพอ็อบเจกต์ (Rumbaugh, 2005)

2.4.4.5 แผนภาพลำดับ

แผนภาพลำดับ (Sequence Diagram) เป็นแผนภาพที่ใช้แสดงลำดับการส่งข้อความภายในฟังก์ชันงาน อยู่ในรูปแบบแผนภูมิ (Chart) 2 มิติ โดยแสดงวัตถุต่างๆ รวมถึงผู้กระทำภายในระบบตามแนวนอนของแผนภูมิ แต่ละวัตถุมีเส้นแนวตั้งเรียกว่าเส้นแสดงชีวิต (Lifeline) แสดงลำดับการส่งข้อความด้วยลูกศรที่ชี้จากเส้นแสดงชีวิตของวัตถุหนึ่งไปหาอีกวัตถุหนึ่ง โดยเรียงลำดับขั้นตอนจากด้านบนไปด้านล่างตามแนวตั้ง แสดงดังภาพประกอบ 2.12 (Rumbaugh, 2005)

2.4.4.6 แผนภาพการสื่อสาร

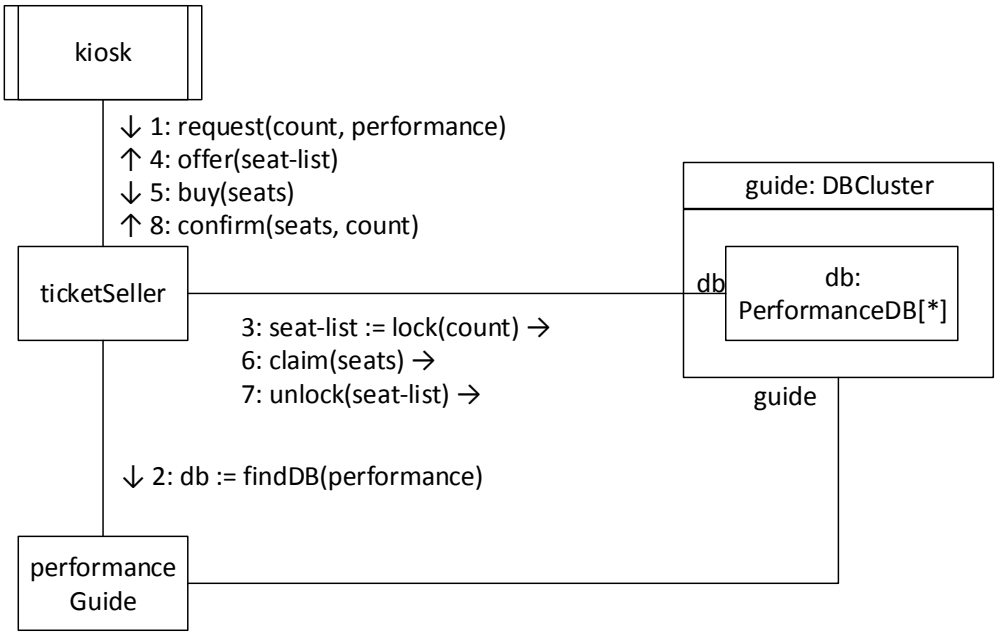
แผนภาพการสื่อสาร (Communication Diagram) เป็นแผนภาพที่ใช้แสดงการปฏิสัมพันธ์ระหว่างองค์ประกอบของระบบ โดยแสดงลำดับการส่งสารระหว่างอ็อบเจกต์ต่างๆ ภายในระบบ ซึ่งจะมีความใกล้เคียงกับแผนภาพลำดับและสามารถใช้ทดแทนกันได้ แต่มีความต่างกันตรงที่แผนภาพการสื่อสารจะแสดงภาพของความสัมพันธ์ระหว่างอ็อบเจกต์ได้ดีกว่าแผนภาพลำดับ และแผนภาพลำดับแสดงลำดับการทำงานในรูปแบบของเส้นชีวิต ในขณะที่แผนภาพการสื่อสารไม่มีการแสดงเส้นชีวิต จึงต้องแสดงลำดับการทำงานด้วยการระบุหมายเลขลำดับไว้หน้าสารที่ส่งระหว่างอ็อบเจกต์ แสดงดังภาพประกอบ 2.13 (Rumbaugh, 2005)



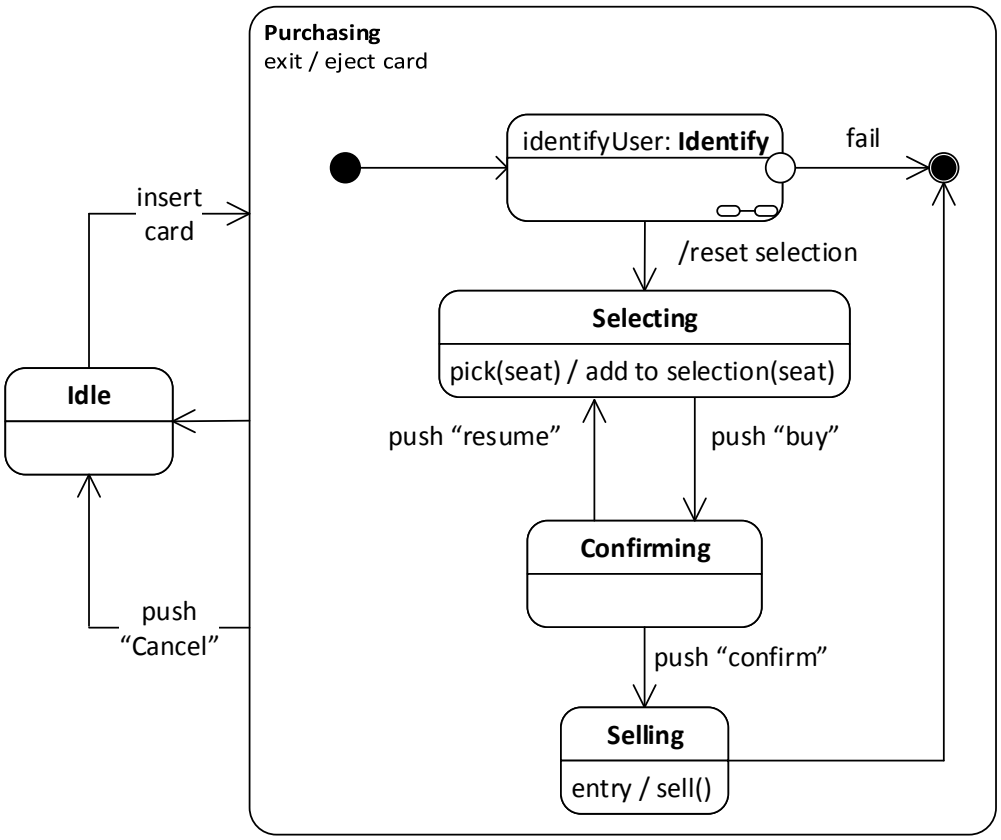
ภาพประกอบ 2.12 ตัวอย่างแผนภาพลำดับ (Rumbaugh, 2005)

2.4.4.7 แผนภาพสถานะ

แผนภาพสถานะ (State Machine Diagram) เป็นแผนภาพที่ใช้แสดงลำดับของการเปลี่ยนแปลงสถานะ (State) ของอ็อบเจกต์หรือกรณีตัวอย่างในขณะที่กำลังดำเนินการตอบสนองต่อเหตุการณ์ในระบบ โดยจะบอกถึงสถานะที่เปลี่ยนไปเมื่อเหตุการณ์นั้นถูกตอบสนอง แสดงดังภาพประกอบ 2.14 (Rumbaugh, 2005)



ภาพประกอบ 2.13 ตัวอย่างแผนภาพการสื่อสาร (Rumbaugh, 2005)



ภาพประกอบ 2.14 ตัวอย่างแผนภาพสถานะ (Rumbaugh, 2005)

บทที่ 3

การวิเคราะห์ระบบงาน

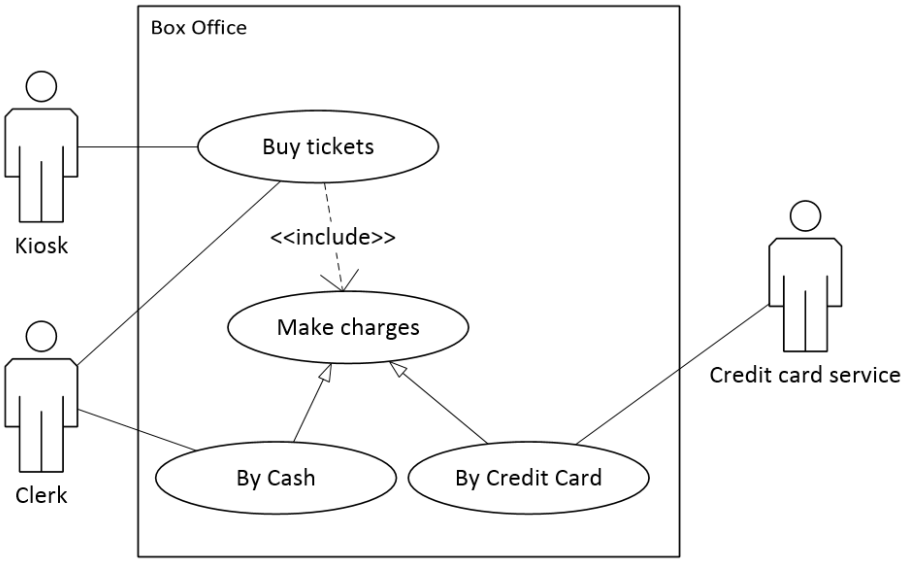
เนื้อหาในบทนี้กล่าวถึงการวิเคราะห์และออกแบบกรอบแนวคิดการตรวจสอบความถูกต้องของแบบจำลองยูสเคส โดยส่วนแรกจะกล่าวถึงการตรวจสอบเบื้องต้น ส่วนถัดไปจะกล่าวถึงกรอบแนวคิดของหลักการที่นำเสนอ และส่วนสุดท้ายกล่าวถึงขั้นตอนการตรวจสอบความถูกต้องของยูสเคส

3.1 การตรวจสอบสภาพปัญหาเบื้องต้น

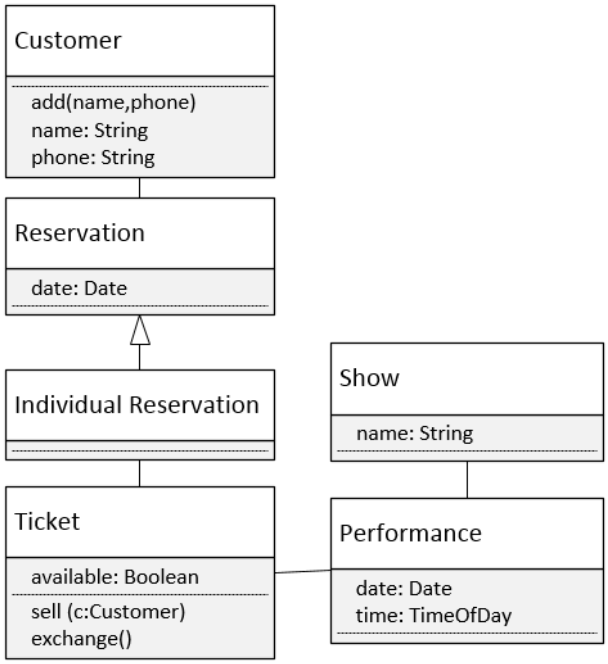
จากการวิเคราะห์ที่มาและความสำคัญของปัญหาในบทที่ 1 งานวิจัยนี้จึงได้มีการตรวจสอบเบื้องต้น 2 ประเด็น คือการส่งต่อความผิดพลาดจากแผนภาพยูสเคสไปยังแผนภาพอื่นๆ และการตรวจสอบความถูกต้องของยูสเคสโดยนักวิเคราะห์ระบบมือใหม่

3.1.1 การส่งต่อความผิดพลาด

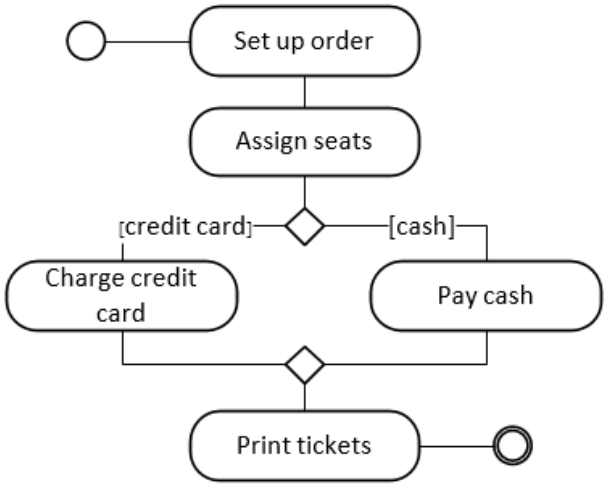
ในการตรวจสอบการส่งต่อความผิดพลาดเบื้องต้น ผู้ดำเนินการวิจัยได้ทำการเลือกใช้แผนภาพยูสเคสที่อ้างอิงมาจากตัวอย่างของ UML Reference Manual (Rumbaugh, 2005) มาเป็นต้นแบบ โดยได้กำหนดกรณีทดสอบเป็น 2 กรณี คือ A Very Simple Box Office และ A Simple Box Office จากนั้น นำแผนภาพยูสเคสในแต่ละกรณีทดสอบมาสร้างแผนภาพยูสเคสที่มีข้อผิดพลาดโดยการเปลี่ยนสัญลักษณ์แทนการดำเนินการบางอย่างภายในแผนภาพ และสุดท้ายนำแผนภาพยูสเคสทั้งหมดที่มี ทั้งแผนภาพที่ถูกต้องที่อ้างอิงมา และแผนภาพที่จงใจใส่ข้อผิดพลาดเข้าไป มาเป็นต้นแบบในการสร้างแผนภาพอื่นๆ ประกอบไปด้วย แผนภาพคลาส และแผนภาพกิจกรรม ดังแสดงตัวอย่างกรณีทดสอบแบบ A Simple Box Office ได้ตามภาพประกอบ 3.1 ถึงภาพประกอบ 3.6 โดยแผนภาพในภาพประกอบ 3.2 และภาพประกอบ 3.3 เป็นแผนภาพที่สร้างจากแผนภาพยูสเคสในภาพประกอบ 3.1 ในขณะที่ แผนภาพในภาพประกอบ 3.5 และ 3.6 เป็นแผนภาพที่สร้างจากแผนภาพยูสเคสที่มีข้อผิดพลาดในภาพประกอบ 3.4



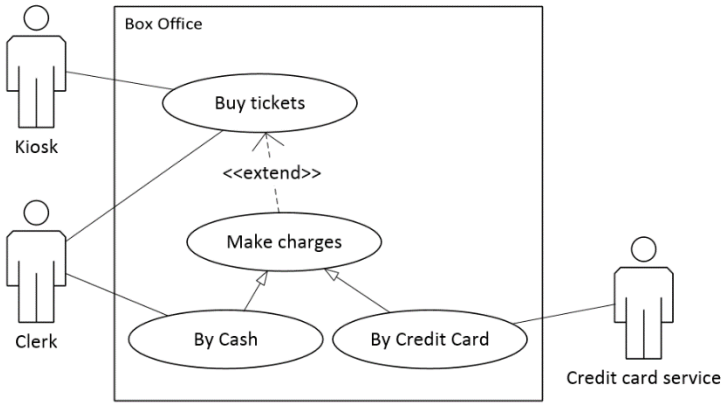
ภาพประกอบ 3.1 แผนภาพยูสเคสของกรณีทดสอบ A Simple Box Office



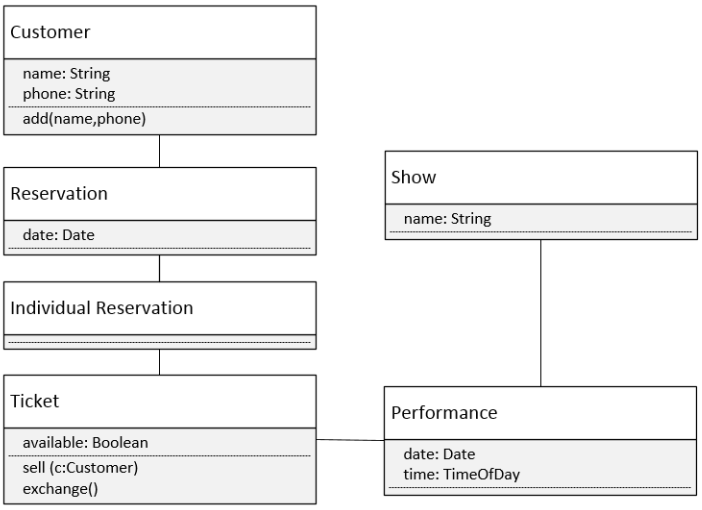
ภาพประกอบ 3.2 แผนภาพคลาสของกรณีทดสอบ A Simple Box Office



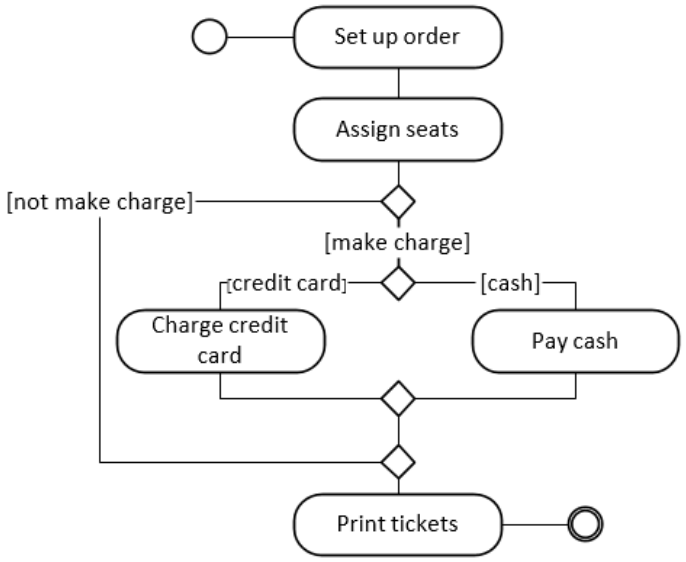
ภาพประกอบ 3.3 แผนภาพกิจกรรมของกรณีทดสอบ A Simple Box Office



ภาพประกอบ 3.4 แผนภาพยูสเคสของกรณีทดสอบ A Simple Box Office ที่มีข้อผิดพลาด



ภาพประกอบ 3.5 แผนภาพคลาสของกรณีทดสอบ A Simple Box Office ที่มีข้อผิดพลาด

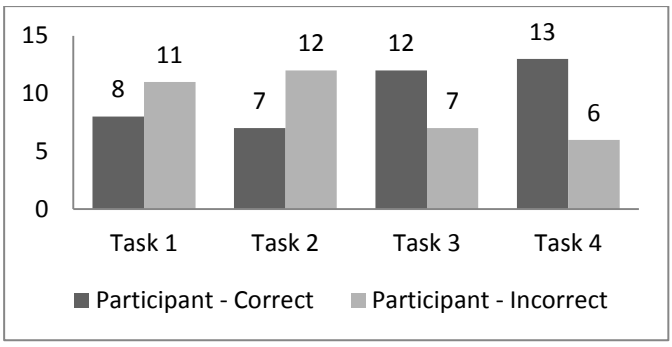


ภาพประกอบ 3.6 แผนภาพกิจกรรมของกรณีทดสอบ A Simple Box Office ที่มีข้อผิดพลาด

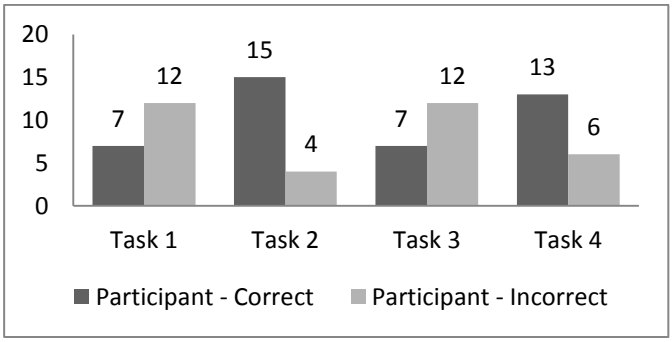
จากภาพประกอบ 3.1 - ภาพประกอบ 3.6 แสดงให้เห็นว่า แผนภาพยูสเคสที่มีความผิดพลาดนั้น หากนำมาสร้างแผนภาพในขั้นถัดไปโดยการอ้างอิงจากแผนภาพยูสเคสนั้นและไม่ได้มีการตรวจสอบที่ถี่ถ้วนเพียงพอ สามารถส่งต่อความผิดพลาดไปยังแผนภาพอื่นๆ ได้ โดยจากกรณีตัวอย่าง แผนภาพยูสเคสในภาพประกอบ 3.4 มีข้อผิดพลาดคือ ความสัมพันธ์ระหว่างเคส Buy Tickets กับเคส Make Charges เปลี่ยนเป็น Extend แทนที่จะเป็น Include ส่งผลให้แผนภาพกิจกรรมในภาพประกอบ 3.6 มีการตัดสินใจที่จะไม่ทำในส่วนการ make charge เพิ่มขึ้นมา เป็นต้น ซึ่งอาจทำให้ซอฟต์แวร์ที่ผลิตขึ้นมีข้อผิดพลาด หรือไม่ตรงตามความต้องการของผู้ใช้ได้

3.1.2 การตรวจสอบความสามารถในการตรวจจับข้อผิดพลาดของนักวิเคราะห์ระบบ

ในการตรวจสอบความสามารถในการตรวจจับข้อผิดพลาดของยูสเคสเบื้องต้นของนักวิเคราะห์ระบบ โดยเฉพาะนักวิเคราะห์ระบบที่มีประสบการณ์น้อย โดยผู้ดำเนินการวิจัยได้ใช้นักศึกษาชั้นปีที่ 4 ที่ผ่านการเรียนวิชาการวิเคราะห์และออกแบบสารสนเทศ และวิชาวิศวกรรมซอฟต์แวร์เบื้องต้น ของภาควิชาวิทยาการคอมพิวเตอร์ เป็นตัวแทนของนักวิเคราะห์ระบบในครั้งนี้ ผู้ดำเนินการวิจัยได้ทำการจำลองสถานการณ์ โดยใช้กรณีทดสอบจากข้อ 3.1.1 ทั้ง 2 กรณี ซึ่งแต่ละกรณีทดสอบประกอบไปด้วย แผนภาพยูสเคสที่ถูกต้อง 1 แผนภาพ แผนภาพยูสเคสที่มีข้อผิดพลาด 3 แผนภาพ และคุณลักษณะความต้องการที่อยู่ในรูปแบบฟอร์ม มาให้ผู้เข้าทดลองดำเนินการทดสอบ โดยการให้ผู้เข้าทดลองพิจารณาคุณลักษณะความต้องการที่ให้ แล้วระบุว่าแผนภาพใดคือแผนภาพที่ถูกต้อง และแผนภาพใดคือแผนภาพที่มีข้อผิดพลาด พร้อมทั้งระบุข้อผิดพลาดนั้น



ภาพประกอบ 3.7 ผลการตรวจสอบเบื้องต้น – กรณีทดสอบที่ 1



ภาพประกอบ 3.8 ผลการตรวจสอบเบื้องต้น – กรณีทดสอบที่ 2

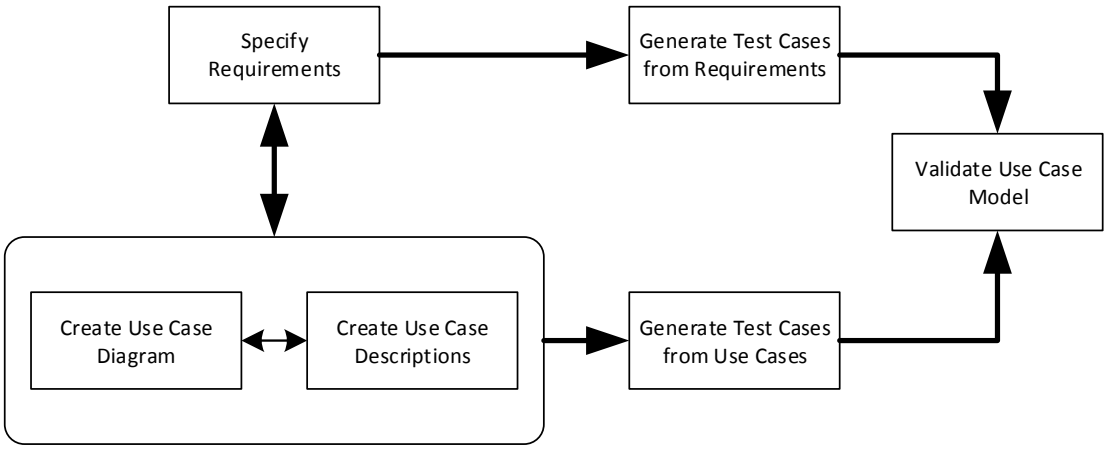
ตารางที่ 3.1 ประสิทธิภาพในการหาข้อผิดพลาดในยูสเคสของผู้เข้าร่วมทดลอง

กรณีทดสอบ	Accuracy Performance Rate				
	Task 1	Task 2	Task 3	Task 4	เฉลี่ย
กรณีทดสอบ 1	42%	39%	63%	68%	53%
กรณีทดสอบ 2	39%	79%	39%	68%	56%

จากการทดลองตามที่กล่าวมาข้างต้น ผู้เข้าร่วมทดลองสามารถแยกแยะแผนภาพยูสเคสที่มีข้อผิดพลาดออกจากแผนภาพยูสเคสที่ถูกต้องของทั้ง 2 กรณีทดสอบได้เฉลี่ย 53% และ 56% ตามลำดับ ดังแสดงได้ตามภาพประกอบ 3.7 ภาพประกอบ 3.8 และตารางที่ 3.1 ซึ่งหากเป็นระบบที่มีความซับซ้อนมากกว่ากรณีทดสอบทั้งสองกรณีนี้ มีความเป็นไปได้ที่อาจมีข้อผิดพลาดในแผนภาพยูสเคสที่ยากต่อการตรวจจับ แม้จะดำเนินการโดยนักวิเคราะห์ระบบที่มีความรู้และประสบการณ์มากกว่าผู้เข้าร่วมทดลอง

3.2 กรอบการทำงานในการตรวจสอบความถูกต้องของยูสเคส

จากวัตถุประสงค์ของงานวิจัยในบทที่ 1 และจากการตรวจสอบเบื้องต้นที่ได้นำเสนอในข้างต้น งานวิจัยนี้นำเสนอวิธีการตรวจสอบความถูกต้องของแบบจำลองยูสเคส โดยวิธีการสร้างกรณีทดสอบเพื่อการยอมรับขึ้นมา 2 ชุด ด้วยวิธีการสร้างโดยนักวิเคราะห์ระบบ ชุดแรกสร้างโดยใช้ข้อมูลจากความต้องการเชิงฟังก์ชัน (Functional Requirements) ของระบบซึ่งได้รับการตรวจสอบจากนักวิเคราะห์ระบบแล้วว่ามีความถูกต้อง และอีกชุดสร้างโดยใช้ข้อมูลจากเอกสารคำอธิบายยูสเคส (Use Case Description) ที่นักวิเคราะห์ระบบสร้างขึ้น จากนั้น นำกรณีทดสอบเพื่อการยอมรับทั้ง 2 ชุด มาเปรียบเทียบกัน ถ้ากรณีทดสอบทั้ง 2 ชุด มีผลลัพธ์ที่ไม่สอดคล้องกันในสถานการณ์เดียวกัน หรือกรณีทดสอบจากยูสเคสไม่ครอบคลุมกรณีทดสอบจากความต้องการเชิงฟังก์ชัน ก็จะสรุปได้ว่าแบบจำลองยูสเคสนั้นมีข้อผิดพลาดเกิดขึ้น ดังแสดงในภาพประกอบ 3.9



ภาพประกอบ 3.9 กรอบการทำงานการตรวจสอบความถูกต้องของยูสเคส

3.3 ขั้นตอนการตรวจสอบความถูกต้องของยูสเคส

การตรวจสอบความถูกต้องของยูสเคสประกอบไปด้วย 5 ขั้นตอนหลัก คือ 1) รับข้อมูลความต้องการเชิงฟังก์ชัน 2) รับข้อมูลคำอธิบายยูสเคส 3) สร้างตารางตัดสินใจ 4) เปรียบเทียบตารางตัดสินใจ และ 5) เปรียบเทียบความสัมพันธ์ โดยมีรายละเอียดของแต่ละขั้นตอนดังต่อไปนี้

3.3.1 การรับข้อมูลความต้องการเชิงฟังก์ชัน

ขั้นตอนการระบุคุณลักษณะความต้องการในรูปแบบฟอร์มสำหรับแต่ละฟังก์ชันงานที่ต้องการ มีรายละเอียดดังนี้

3.3.1.1 กำหนดคุณลักษณะความต้องการในรูปแบบฟอร์ม

นักวิเคราะห์ระบบหรือผู้ใช้ระบุคุณลักษณะความต้องการในรูปแบบฟอร์ม และหากมีความสัมพันธ์กับฟังก์ชันอื่นๆ ก็ให้ระบุไปในขั้นตอนนี้ด้วยว่าเป็นความสัมพันธ์ที่จำเป็น (Require) ความสัมพันธ์เพิ่มเติม (Additional) หรือความสัมพันธ์แบบเลือกใช้งาน 1 ฟังก์ชัน (Choose one)

3.3.1.2 ระบุผลลัพธ์ที่คาดหวังของฟังก์ชันพร้อมระบุค่าความถูกต้องของสถานการณ์นั้น

นักวิเคราะห์ระบบระบุผลลัพธ์ที่คาดหวังของฟังก์ชัน ซึ่งเป็นสถานการณ์ (Scenario) ที่เป็นไปได้ทั้งหมดจากการดำเนินการฟังก์ชันนั้น พร้อมทั้งระบุค่าความถูกต้อง (Validity) ของแต่ละสถานการณ์ว่าเป็นสถานการณ์ที่ถูกต้อง (Valid) หรือไม่ถูกต้อง (Invalid)

3.3.1.3 ระบุชนิดข้อมูลและค่าที่เป็นไปได้ของข้อมูลนำเข้า

นักวิเคราะห์ระบบระบุชนิดของตัวแปรแต่ละตัว ป้อนค่าที่เป็นไปได้ของแต่ละข้อมูลนำเข้า มีรายละเอียดดังนี้

1) ชนิดข้อมูลประเภทตัวเลข (Number) นักวิเคราะห์ระบบระบุค่าข้อมูลที่เป็นไปได้ ถ้าข้อมูลอยู่ในรูปแบบช่วงของข้อมูล ให้เขียนโดยใช้ตัวดำเนินการสัมพันธ์ (Relational Operators) เช่น >MIN, <MAX คั่นแต่ละค่าด้วยเครื่องหมายจุลภาค (,)

2) ชนิดข้อมูลประเภทตัวอักษรหรือข้อความ (Character or String) ถ้าข้อมูลมีค่าที่เป็นไปได้ที่เป็นไปได้เพียง 1 ค่า ให้นักวิเคราะห์ระบบระบุค่านั้นไป 1 ค่า ถ้าข้อมูลมีค่าที่เป็นไปได้มากกว่า 1 ค่า ให้นักวิเคราะห์ระบบระบุค่าข้อมูลที่เป็นไปได้ทั้งหมดเป็นตัวอักษร หรือข้อความคั่นด้วยเครื่องหมายจุลภาค (,) ถ้าข้อมูลมีค่าที่เป็นไปได้มากกว่า 1 ค่า และเป็นข้อมูลที่จำเป็นต่อการทำงานของระบบ ให้นักวิเคราะห์ระบุค่าข้อมูลที่เป็นไปได้ทั้งหมด

3) ชนิดข้อมูลประเภทตรรกะ (Boolean) มีค่าที่เป็นไปได้เพียง 2 อย่าง คือ true หรือ false เท่านั้น

3.3.2 การรับข้อมูลคำอธิบายยูสเคส

ขั้นตอนการรับคำอธิบายยูสเคส มีรายละเอียดดังนี้

3.3.2.1 กำหนดคำอธิบายยูสเคสในรูปแบบฟอร์ม

นักวิเคราะห์ระบบระบุคำอธิบายยูสเคสในรูปแบบฟอร์ม และหากมีความสัมพันธ์ระหว่างยูสเคสก็ให้ระบุไปในขั้นตอนนี้ ประเภทของความสัมพันธ์ระหว่างยูสเคสประกอบด้วย Associate, Include, Extend, และ Generalization

3.3.2.2 ระบุเส้นทางภายในยูสเคส

นักวิเคราะห์ระบบระบุเส้นทาง (Path) ภายในยูสเคส ทั้งเส้นทางหลัก (Main Path) และเส้นทางเพิ่มเติม (Additional Path) พร้อมทั้งระบุค่าความถูกต้อง (Validity) ของแต่ละเส้นทางว่าเป็นเส้นทางที่ผลลัพธ์ที่ถูกต้อง (Valid) หรือไม่ถูกต้อง (Invalid) โดยในการระบุเส้นทางเพิ่มเติมนั้น นักวิเคราะห์ต้องระบุว่าเป็นเส้นทางที่แยกออกมาจากเส้นทางใด ในขั้นตอนใด

3.3.2.3 ระบุชนิดข้อมูลและค่าที่เป็นไปได้ของข้อมูลนำเข้า

นักวิเคราะห์ระบบระบุชนิดของตัวแปรแต่ละตัว ป้อนค่าที่เป็นไปได้ของข้อมูลนำเข้าแต่ละตัว มีรายละเอียดดังนี้

1) ชนิดข้อมูลประเภทตัวเลข (Number) นักวิเคราะห์ระบบระบุค่าข้อมูลที่เป็นไปได้ ถ้าข้อมูลอยู่ในรูปแบบช่วงของข้อมูล ให้เขียนโดยใช้ตัวดำเนินการสัมพันธ์ (Relational Operators) เช่น >MIN, <MAX คั่นแต่ละค่าด้วยเครื่องหมายจุลภาค (,)

2) ชนิดข้อมูลประเภทตัวอักษรหรือข้อความ (Character or String) ถ้าข้อมูลมีค่าที่เป็นไปได้ที่เป็นไปได้เพียง 1 ค่า ให้นักวิเคราะห์ระบบระบุค่านั้นไป 1 ค่า ถ้าข้อมูลมีค่าที่เป็นไปได้มากกว่า 1 ค่า ให้นักวิเคราะห์ระบบระบุค่าข้อมูลที่เป็นไปได้ทั้งหมด เป็นตัวอักษรหรือข้อความคั่นด้วยเครื่องหมายจุลภาค (,) ถ้าข้อมูลมีค่าที่เป็นไปได้มากกว่า 1 ค่า และเป็นข้อมูลที่จำเป็นต่อการทำงานของระบบ ให้นักวิเคราะห์ระบุค่าข้อมูลที่เป็นไปได้ทั้งหมด

3) ชนิดข้อมูลประเภทตรรกะ (Boolean) มีค่าที่เป็นไปได้เพียง 2 อย่าง คือ true หรือ false เท่านั้น

3.3.3 การสร้างตารางตัดสินใจ

ในขั้นตอนนี้ ระบบจะสร้างตารางตัดสินใจจากข้อมูลนำเข้า ผลลัพธ์ที่คาดหวัง และเส้นทาง ที่ได้จากในขั้นตอนที่ 1 และขั้นตอนที่ 2 ดำเนินการโดยเริ่มจากกำหนดเซตของข้อมูลนำเข้าแต่ละตัว $Pi_{in} = \{p_1, p_2, \dots, p_{fi}\}$ โดยที่ Pi_{in} แทนข้อมูลนำเข้า และ p_{fi} แทนค่าที่เป็นไปได้ของข้อมูลนำเข้าตัวนั้น จากนั้น หาผลคูณคาร์ทีเชี่ยน (Cartesian Product) ของเซต Pi_{in} ทุกเซต และเก็บผลลัพธ์ไว้ที่เซต C ดังสมการ

$$C = Pi_1 \times Pi_2 \times \dots \times Pi_{in}$$

โดยที่แต่ละสมาชิกภายใน C คือสถานการณ์ (Scenario) ในตารางตัดสินใจ จากนั้น ให้นักวิเคราะห์ระบบระบุว่าแต่ละสถานการณ์คือสถานการณ์ที่เกิดขึ้นในผลลัพธ์ที่คาดหวังกรณีใดของคุณลักษณะความต้องการ หรือเกิดขึ้นในเส้นทางการทำงานใดของคำอธิบายยูสเคส โดยในการสร้างตารางตัดสินใจนี้จะใช้วิธีเดียวกันกับทั้งคุณลักษณะความต้องการ และคำอธิบายยูสเคส เช่น จากกรณีตัวอย่าง Simple Box-Office ในฟังก์ชัน Buy Tickets มีข้อมูลนำเข้า 3 ตัว คือ performance, count, และ seats โดยมีผลลัพธ์ที่คาดหวังดังแสดงในตาราง 3.2 และมีค่าที่เป็นไปได้ดังต่อไปนี้

$$\text{performance: } Pi_1 = \{available,\ unavailable\}$$

$$\text{count: } Pi_2 = \begin{Bmatrix} < available\ seats, \\ = available\ seats, \\ > available\ seats \end{Bmatrix}$$

$$\text{seats: } Pi_3 = \{free,\ taken\}$$

ตารางที่ 3.2 ผลลัพธ์ที่คาดหวังของฟังก์ชัน Simple Box-office

ผลลัพธ์ที่คาดหวัง
A customer has done a successful transaction.
A selected show is not available.
A selected show has not enough seats
The selected seats were taken

นำเซตของข้อมูลนำเข้าทั้ง 3 ตัวมาหาผลคูณคาร์ทีเซียนของ $Pi_1 Pi_2$ และ Pi_3 และนำมาเขียนเป็นตารางได้ดังตาราง 3.3

ตารางที่ 3.3 ตารางตัดสินใจของฟังก์ชัน Simple Box-office

performance	count	seats
available	<available seats	free
available	<available seats	taken
unavailable	<available seats	free
unavailable	<available seats	taken
available	=available seats	free
available	=available seats	taken

ตารางที่ 3.3 ตารางตัดสินใจของฟังก์ชัน Simple Box-office (ต่อ)

performance	count	seats
unavailable	=available seats	free
unavailable	=available seats	taken
available	>available seats	free
available	>available seats	taken
unavailable	>available seats	free
unavailable	>available seats	taken

จากนั้น ระบุว่าแต่ละสถานการณ์คือสถานการณ์ที่เกิดขึ้นในผลลัพธ์ที่คาดหวังกรณีใดของคุณลักษณะความต้องการ ดังแสดงในตาราง 3.4

ตารางที่ 3.4 ผลลัพธ์ที่คาดหวังและตารางตัดสินใจของฟังก์ชัน Simple Box-office

ผลลัพธ์ที่คาดหวัง	performance	count	seats
A customer has done a successful transaction	available	<available seats	free
	available	=available seats	free
A selected show is not available	unavailable	<available seats	free
	unavailable	<available seats	taken
	unavailable	=available seats	free
	unavailable	=available seats	taken
	unavailable	>available seats	free
	unavailable	>available seats	taken
A selected show has not enough seats	available	>available seats	free
	available	>available seats	taken
The selected seats were taken	available	<available seats	taken
	available	=available seats	taken

3.3.4 การเปรียบเทียบตารางตัดสินใจ

ในขั้นตอนนี้ นักวิเคราะห์ระบบจะต้องจับคู่คุณลักษณะความต้องการเข้ากับยูสเคสที่สอดคล้องกัน จากนั้น กำหนดเซต S และเซต T ขึ้นมา โดยที่สมาชิกของเซต S คือเซตของคุณลักษณะความต้องการ และสมาชิกของเซต T คือเซตของคำอธิบายยูสเคส และลำดับของสมาชิกใน T เรียงตามคู่คุณลักษณะที่สอดคล้องกันใน S จะได้ว่า เซตคุณลักษณะความต้องการคือ $S = \{F_1, F_2, \dots, F_b\}$ และเซตของคำอธิบายยูสเคสคือ $T = \{U_1, U_2, \dots, U_b\}$ และมีคู่อันดับความสัมพันธ์ระหว่างคุณลักษณะความต้องการกับยูสเคสเป็น $(F_1, U_1), (F_2, U_2), \dots, (F_b, U_b)$

สำหรับแต่ละ $F_b \epsilon S$ กำหนดเซตของแต่ละคุณลักษณะ ซึ่งมีสมาชิกคือ $R_1, R_2, \dots, R_m$ โดยที่ R_m คือเซตของผลลัพธ์ที่คาดหวังของคุณลักษณะความต้องการนั้น ในแต่ละ R_m กำหนดเซตของแต่ละข้อมูลนำเข้า $I_1, I_2, \dots, I_q$ โดยที่ I_q คือเซตของค่าที่เป็นไปได้ของข้อมูลนำเข้าแต่ละตัวในสถานการณ์นั้น ดังสมการ

$$F_b = \{R_1, R_2, \dots, R_m\}$$

$$R_m = \{I_1, I_2, \dots, I_q\}$$

สำหรับแต่ละ $U_b \epsilon T$ กำหนดเซตของแต่ละยูสเคส ซึ่งมีสมาชิกคือ $P_1, P_2, \dots, P_m$ โดยที่ P_m คือเซตของเส้นทางภายในยูสเคสนั้น ในแต่ละ P_m กำหนดเซตของแต่ละข้อมูลนำเข้า $J_1, J_2, \dots, J_q$ โดยที่ J_q คือเซตของค่าที่เป็นไปได้ของข้อมูลนำเข้าแต่ละตัวในสถานการณ์นั้น ดังสมการ

$$U_b = \{P_1, P_2, \dots, P_m\}$$

$$P_m = \{J_1, J_2, \dots, J_q\}$$

เช่น จากตัวอย่างในตาราง 2.1 – 2.3 สร้างเซตของคุณลักษณะความต้องการได้ดังนี้

$$F_1 = \{R_1, R_2, R_3, R_4\}$$

$$R_1 = \{I_1, I_2\}$$

$$I_1 = \{available, < available\ seats, free\}$$

$$I_2 = \{available, = available\ seats, free\}$$

จากนั้น เปรียบเทียบตารางตัดสินใจโดยการนำเซตของ I_q ในแต่ละผลลัพธ์ที่คาดหวัง R_m มา Intersection กับเซตของผลคูณคาร์ทีเซียนของคำอธิบายยูสเคส C เนื่องจาก C คือสถานการณ์ที่เป็นไปได้ทั้งหมดของยูสเคส ดังนั้น หาก Intersect กันแล้วได้ผลลัพธ์เท่ากับ I_q หมายความว่า มีสถานการณ์เดียวกันเกิดขึ้น ทั้งในคุณลักษณะความต้องการ และในคำอธิบายยูสเคส ในทางกลับกัน หาก Intersect แล้วได้ผลลัพธ์ไม่เท่ากับ I_q แสดงว่าในคำอธิบายยูสเคสไม่มี

สถานการณ์นี้เกิดขึ้น หรือสถานการณ์ที่เกิดขึ้นมีความแตกต่างไปจากคุณลักษณะความต้องการ จึงสรุปได้ว่ามีข้อผิดพลาดเกิดขึ้นในคำอธิบายยูสเคสนั้น

3.3.5 การเปรียบเทียบความสัมพันธ์

ในขั้นตอนนี้ เป็นการตรวจสอบความสัมพันธ์ภายในมุมมองยูสเคส เนื่องจากการแทนความสัมพันธ์ที่ผิดไปจากที่ควรจะเป็น จะส่งผลต่อการทำงานโดยรวมของระบบได้ โดยในฝั่งมุมมองยูสเคส มีความสัมพันธ์ 3 ประเภท คือ “include” “extend” และ “generalization” ในขณะที่ในฝั่งคุณลักษณะความต้องการนั้นไม่ได้มีคำหลักที่ใช้ระบุไว้อย่างชัดเจนเหมือนในฝั่งมุมมองยูสเคส ดังนั้น ในงานวิจัยฉบับนี้จะใช้คำว่า “require” “additional” และ “choose one” แทนความสัมพันธ์ในฝั่งคุณลักษณะความต้องการ โดย “require”จะใช้ในกรณีที่ในการทำงานของฟังก์ชันหนึ่งต้องการให้มีการเรียกใช้อีกฟังก์ชัน ส่วน “additional” ใช้ในกรณีที่ฟังก์ชันหนึ่งมีการทำงานร่วมกับอีกฟังก์ชันได้ แตกต่างกับ “require” ตรงที่ “additional” อาจจะมีการเรียกใช้อีกฟังก์ชันหนึ่งหรือไม่มีก็ได้ และ “choose one” ใช้ในกรณีที่ฟังก์ชันมีความสัมพันธ์กับฟังก์ชันอื่นมากกว่า 1 ฟังก์ชัน แต่ในการทำงานมีการเรียกใช้เพียง 1 ฟังก์ชันเท่านั้น

ขั้นตอนการเปรียบเทียบความสัมพันธ์ มีรายละเอียดดังนี้

3.3.5.1 การสร้างตารางความสัมพันธ์

ในขั้นตอนนี้ ระบบจะสร้างตารางความสัมพันธ์ของคุณลักษณะความต้องการเพื่อระบุฟังก์ชันหลัก ฟังก์ชันที่ถูกเรียกใช้ และความสัมพันธ์ในรูปแบบ $\{f_i, r_j, f_k\}$ โดย f_i คือ ฟังก์ชันหลัก r_j คือความสัมพันธ์ระหว่างฟังก์ชัน และ f_k คือฟังก์ชันที่ถูกเรียกใช้ จากนั้น ดำเนินการในลักษณะเดียวกันกับยูสเคส โดยสร้างตารางความสัมพันธ์ของยูสเคส เพื่อระบุเคสหลัก เคสที่ถูกเรียกใช้ เละความสัมพันธ์ ในรูปแบบ $\{u_i, q_j, u_k\}$ โดย u_i คือยูสเคสหลัก q_j คือความสัมพันธ์ระหว่างยูสเคส และ u_k คือยูสเคสที่ถูกเรียกใช้ เช่น จากกรณีตัวอย่างในภาพประกอบ 3.1 นำมาสร้างตารางความสัมพันธ์ได้ดังแสดงในตาราง 3.5

ตารางที่ 3.5 ตัวอย่างการสร้างตารางความสัมพันธ์

ฟังก์ชันหลัก	ความสัมพันธ์	ฟังก์ชันที่ถูกเรียกใช้
Buy Tickets	Require	Make Charges
Make Charges	Choose One	By Cash
Make Charges	Choose One	By Credit Card

3.3.5.2 การเปรียบเทียบตารางความสัมพันธ์

ในขั้นตอนนี้ ระบบจะเปรียบเทียบความสัมพันธ์ของคุณลักษณะความต้องการกับยูสเคส โดยใช้คู่คุณลักษณะและยูสเคสที่สัมพันธ์กันตามที่ได้จับคู่ไว้แล้วในขั้นตอนที่ 4 มาพิจารณาความสัมพันธ์กัน โดยมีข้อกำหนดดังนี้ ถ้าคุณลักษณะความต้องการมีความสัมพันธ์เป็น “require” ยูสเคสต้องมีความสัมพันธ์เป็น “include” ถ้าคุณลักษณะความต้องการมีความสัมพันธ์เป็น “additional” ยูสเคสต้องมีความสัมพันธ์เป็น “extend” และถ้าคุณลักษณะความต้องการมีความสัมพันธ์เป็น “choose one” ยูสเคสต้องมีความสัมพันธ์เป็น “generalization” ถ้าความสัมพันธ์ของคู่คุณลักษณะและยูสเคสใดไม่เป็นไปตามข้อกำหนดทั้ง 3 ข้อนี้ ถือว่ายูสเคสนั้นมีความผิดปกติเกิดขึ้น เช่น จากกรณีตัวอย่างในภาพประกอบ 3.4 นำมาสร้างตารางความสัมพันธ์ได้ดังแสดงในตาราง 3.6

ตารางที่ 3.6 ตัวอย่างการเปรียบเทียบความสัมพันธ์

ยูสเคสหลัก	ความสัมพันธ์	ยูสเคสที่ถูกเรียกใช้
Buy Tickets	Extend	Make Charges
Make Charges	Generalize	By Cash
Make Charges	Generalize	By Credit Card

เมื่อนำตารางที่ 3.5 และ 3.6 มาเปรียบเทียบกัน พบว่า คู่ความสัมพันธ์ {Buy Tickets, Extend, Make charges} นั้น ไม่สอดคล้องกับความสัมพันธ์ของคุณลักษณะความต้องการ คือ {Buy Tickets, Require, Make charges} ดังนั้น จึงสรุปได้ว่าความสัมพันธ์ระหว่างยูสเคส Buy Tickets และยูสเคส Make Charges มีข้อผิดพลาดเกิดขึ้น

บทที่ 4

การออกแบบและพัฒนาระบบ

เนื้อหาในบทนี้กล่าวถึงการออกแบบและพัฒนาระบบตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล (Use Case Validation for Object-Oriented Software Development with UML) โดยจะกล่าวถึงการออกแบบขั้นตอนวิธีการออกแบบระบบโดยรวม เครื่องมือที่ใช้พัฒนาระบบ และการพัฒนาระบบ

4.1 การออกแบบขั้นตอนวิธีสำหรับการตรวจจับข้อผิดพลาด

จากการวิเคราะห์ระบบในบทที่ 3 สามารถออกแบบขั้นตอนวิธีสำหรับการตรวจจับข้อผิดพลาดได้ดังแสดงในรหัสเทียม (Pseudo Code) ของระบบดังแสดงในภาพประกอบ 4.1 มีลำดับการทำงานโดยสรุปคือ

1) รับข้อมูลที่จำเป็นเข้ามาในระบบ และสร้างตารางตัดสินใจจากค่าที่เป็นไปได้ของข้อมูลนำเข้าด้วยวิธีการหาผลคูณคาร์ทีเชียน โดยสร้างทั้งตารางตัดสินใจจากคุณลักษณะความต้องการ และจากคำอธิบายยูสเคส

2) หาความผิดพลาดที่อาจเกิดขึ้นในแต่ละสถานการณ์ของฟังก์ชัน โดยการใช้แต่ละสถานการณ์ในคุณลักษณะความต้องการมาพิจารณาว่าในตารางตัดสินใจจากคำอธิบายยูสเคสมีสถานการณ์นี้เกิดขึ้นหรือไม่ โดยจะสามารถบอกได้ว่าแต่ละสถานการณ์เมื่อนำมาพิจารณาแล้วได้ผลลัพธ์เป็นอย่างไร

3) หาความผิดพลาดที่อาจเกิดขึ้นในความสัมพันธ์ระหว่างฟังก์ชัน โดยการพิจารณาคู่อันดับความสัมพันธ์ระหว่างคุณลักษณะความต้องการ เปรียบเทียบกับคู่อันดับความสัมพันธ์ระหว่างยูสเคสที่สัมพันธ์กัน

4) ส่งผลกลับไปแจ้งผู้ใช้ ว่ามีข้อผิดพลาดเกิดขึ้นในสถานการณ์ใด หรือมีข้อผิดพลาดเกิดขึ้นในความสัมพันธ์ใดหรือไม่

Algorithm 1 Algorithm for validating use case

Input: expected results, inputs, specification's relation, use case's relation

Output: Use case validating results

Get expected results as R_s
Get inputs from requirements as Pi_n
repeat
  Get possible values of each input as p_p
until No more possible values.
Get inputs from use cases as Pi_m
repeat
  Get possible values of each input as p_q
until No more possible values.
Do Cartesian Product of every Pi_n as C_r,
C_r = (I_v : I_v is the set of possible values of scenario)
Do Cartesian Product of every Pi_m as C_u,
C_u = (J_w : J_w is the set of possible values of scenario)
repeat
  Define which I_v belongs to each R_s
until No more I_v that doesn't belong to any R_s
repeat
  Let $T = I_v \cap C_u$
  if $T < I_v$ **then**
    "Scenario I_v in Expected result R_s = Warning"
  else
    "Scenario I_v in Expected result R_s = Passed"
  end if
until No more expected result.
Get specification's relation.
Get use case's relation.
if specification's relation = "Require" & use case's relation $\neq$ "Include" **then**
  "Relation Defected"
else if specification's relation = "Additional" & use case's relation $\neq$ "Extend" **then**
  "Relation Defected"
else if specification's relation = "Choose one" & use case's relation $\neq$ "Generalization" **then**
  "Relation Defected"
else
  "Relation Passed"
end if
return Use case validating results.

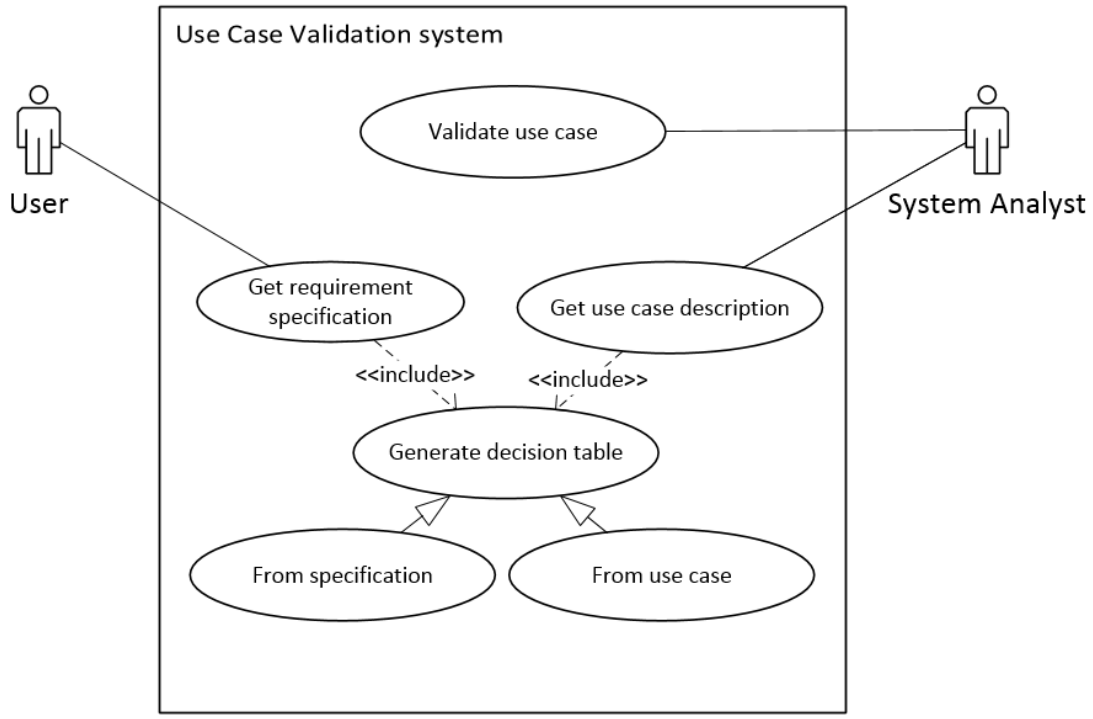
ภาพประกอบ 4.1 รหัสเทียมของระบบ

4.2 การออกแบบระบบโดยรวม

การออกแบบระบบโดยรวม ในที่นี้เพื่อกำหนดรูปแบบในการพัฒนาเครื่องมือสนับสนุนการตรวจจับข้อผิดพลาดตามขั้นตอนที่ได้นำเสนอแนวคิด ซึ่งจะเป็นประโยชน์ต่อการดำเนินงานในกระบวนการวิศวกรรมซอฟต์แวร์ที่สามารถนำไปปฏิบัติจริงได้ จากการวิเคราะห์ระบบในบทที่ 3 สามารถออกแบบกิจกรรม สถาปัตยกรรม ฐานข้อมูล และส่วนติดต่อกับผู้ใช้ของระบบตรวจสอบความถูกต้องของยูสเคสได้ดังต่อไปนี้

4.2.1 ความสัมพันธ์และกิจกรรมของระบบ

ระบบตรวจสอบความถูกต้องของยูสเคสมีผู้เกี่ยวข้องทั้งหมด 2 ฝ่าย คือผู้ใช้ และนักวิเคราะห์ระบบ ซึ่งมีกิจกรรมและความสัมพันธ์ต่างๆ ระหว่างกันดังแสดงได้ตามภาพประกอบ 4.2 และมีรายละเอียดของแต่ละยูสเคสดังคำอธิบายในตารางที่ 4.1 ถึงตารางที่ 4.6



ภาพประกอบ 4.2 แผนภาพยูสเคสของระบบ

ตารางที่ 4.1 คำอธิบายกิจกรรม Validate Use Case

ชื่อ	Validate use case
เป้าหมาย	เพื่อให้นักวิเคราะห์ระบบตรวจสอบความถูกต้องของยูสเคส
ผู้ดำเนินการ	นักวิเคราะห์ระบบ (System Analyst)
การดำเนินการ	นักวิเคราะห์ระบบทำการระบุยูสเคสที่จะทำการตรวจสอบ จากนั้นระบบจะดำเนินการตรวจสอบและแสดงผลลัพธ์ออกมา
เงื่อนไขหรือข้อยกเว้น	ต้องมีตารางตัดสินใจของคุณลักษณะความต้องการและตารางตัดสินใจของยูสเคสที่เพียงพอต่อการตรวจสอบ
ผลลัพธ์	ผลการตรวจสอบความถูกต้อง ชื่อฟังก์ชันและข้อมูลนำเข้าที่มีข้อผิดพลาด ความสัมพันธ์ที่มีข้อผิดพลาด

ตารางที่ 4.2 คำอธิบายกิจกรรม Get Requirement Specification

ชื่อ	Get requirement specification
เป้าหมาย	เพื่อให้ผู้ใช้ป้อนคุณลักษณะความต้องการเข้าไปในระบบ
ผู้ดำเนินการ	ผู้ใช้ (User)
การดำเนินการ	ผู้ใช้จะต้องป้อนคุณลักษณะความต้องการเข้าไปในแบบฟอร์มที่ระบบเตรียมไว้ให้ พร้อมทั้งให้ข้อมูลรายละเอียดข้อมูลนำเข้าและผลลัพธ์ที่คาดหวังของแต่ละฟังก์ชันงาน
เงื่อนไขหรือข้อยกเว้น	-
ผลลัพธ์	คุณลักษณะความต้องการในรูปแบบฟอร์ม

ตารางที่ 4.3 คำอธิบายกิจกรรม Get Use Case Description

ชื่อ	Get use case description
เป้าหมาย	เพื่อให้นักวิเคราะห์ระบบป้อนคำอธิบายยูสเคสเข้าไปในระบบ
ผู้ดำเนินการ	นักวิเคราะห์ระบบ (System Analyst)
การดำเนินการ	นักวิเคราะห์ระบบจะต้องป้อนคำอธิบายยูสเคสเข้าไปในแบบฟอร์มที่ระบบเตรียมไว้ให้ พร้อมทั้งให้ข้อมูลรายละเอียดข้อมูลนำเข้าและเส้นทางการทำงานของยูสเคส
เงื่อนไขหรือข้อยกเว้น	-
ผลลัพธ์	คำอธิบายยูสเคสในรูปแบบฟอร์ม

ตารางที่ 4.4 คำอธิบายกิจกรรม Generate Decision Table

ชื่อ	Generate decision table
เป้าหมาย	เพื่อสร้างตารางตัดสินใจ
ผู้ดำเนินการ	กิจกรรม Get requirement specification และกิจกรรม Get use case description
การดำเนินการ	ระบบจะเรียกใช้กิจกรรม From specification หรือ From use case ตามแต่ข้อมูลที่ผู้ใช้ระบบป้อนเข้ามา
เงื่อนไขหรือข้อยกเว้น	-
ผลลัพธ์	ตารางตัดสินใจ

ตารางที่ 4.5 คำอธิบายกิจกรรม From Specification

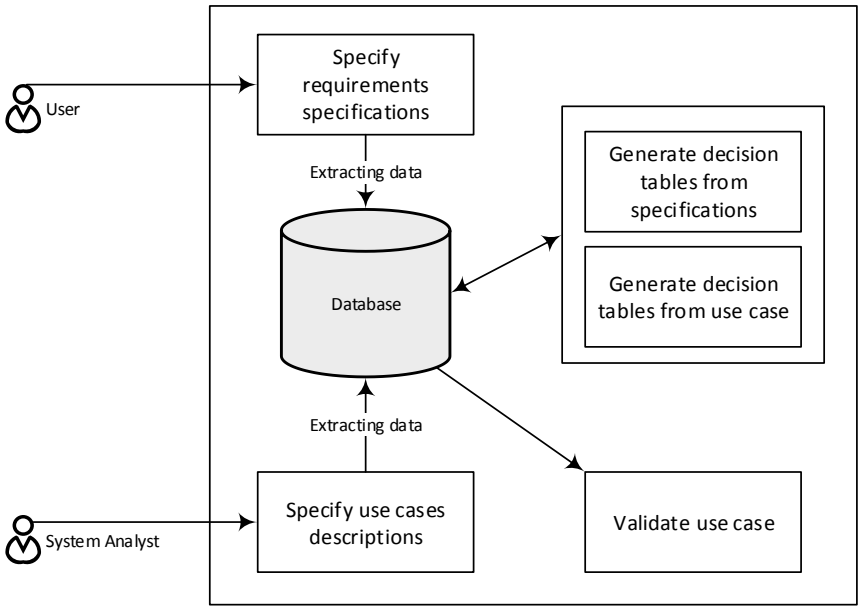
ชื่อ	From specification
เป้าหมาย	เพื่อสร้างตารางตัดสินใจจากคุณลักษณะความต้องการ
ผู้ดำเนินการ	กิจกรรม Generate decision table
การดำเนินการ	ระบบจะสร้างตารางตัดสินใจจากข้อมูลที่สกัดมาจากคุณลักษณะความต้องการโดยอัตโนมัติ
เงื่อนไขหรือข้อยกเว้น	ผู้ใช้ป้อนข้อมูลคุณลักษณะความต้องการเข้ามาในระบบ
ผลลัพธ์	ตารางตัดสินใจ

ตารางที่ 4.6 คำอธิบายกิจกรรม From Use Case

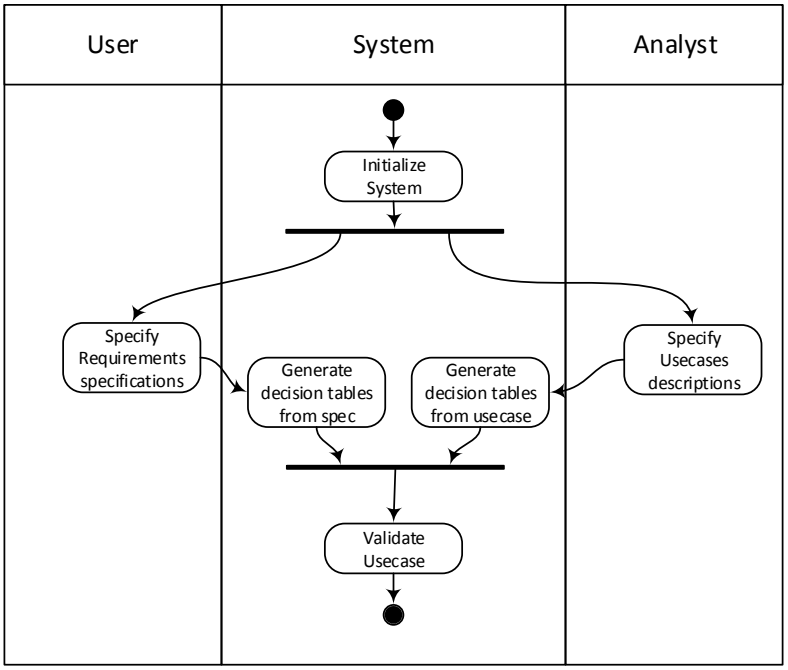
ชื่อ	From use case
เป้าหมาย	เพื่อสร้างตารางตัดสินใจจากคำอธิบายยูสเคส
ผู้ดำเนินการ	กิจกรรม Generate decision table
การดำเนินการ	ระบบจะสร้างตารางตัดสินใจจากข้อมูลที่สกัดมาจากคำอธิบายยูสเคสโดยอัตโนมัติ
เงื่อนไขหรือข้อยกเว้น	นักวิเคราะห์ระบบป้อนข้อมูลคำอธิบายยูสเคสเข้ามาในระบบ
ผลลัพธ์	ตารางตัดสินใจ

4.2.2 สถาปัตยกรรมของระบบ

ระบบตรวจสอบความถูกต้องของยูสเคสมีองค์ประกอบเพื่อรองรับการป้อนข้อมูลคุณลักษณะความต้องการและคำอธิบายยูสเคส การสร้างตารางตัดสินใจ รวมถึงการตรวจสอบความถูกต้องของยูสเคส ซึ่งสามารถแสดงสถาปัตยกรรมของระบบได้ตามภาพประกอบ 4.3 และภาพประกอบ 4.4



ภาพประกอบ 4.3 สถาปัตยกรรมของระบบ



ภาพประกอบ 4.4 แผนภาพกิจกรรมของระบบ

ระบบตรวจสอบความถูกต้องของยูสเคสประกอบด้วยองค์ประกอบหลักทั้งหมด 4 ส่วน ซึ่งมีรายละเอียดของการดำเนินการดังนี้

4.2.2.1 ส่วนตรวจสอบความถูกต้องของยูสเคส

ส่วนตรวจสอบความถูกต้องของยูสเคส (Use Case Validation Module) เป็นฟังก์ชันการทำงานที่เป็นเป้าหมายหลักของระบบ คือการตรวจสอบความถูกต้องของยูสเคส ซึ่งในการทำงานของฟังก์ชันนี้ต้องการตารางตัดสินใจที่ได้จากข้อมูลที่ผู้ใช้และนักวิเคราะห์ระบบป้อนเข้าไปในระบบ

4.2.2.2 ส่วนสร้างตารางตัดสินใจ

ส่วนสร้างตารางตัดสินใจ (Decision Table Generation Module) เป็นฟังก์ชันการทำงานที่ใช้สำหรับสร้างตารางตัดสินใจ โดยแบ่งออกเป็น 2 ฟังก์ชันย่อย คือ ฟังก์ชันสร้างตารางตัดสินใจจากคุณลักษณะความต้องการ และฟังก์ชันสร้างตารางตัดสินใจจากคำอธิบายยูสเคส

4.2.2.3 ส่วนรับคุณลักษณะความต้องการ

ส่วนรับคุณลักษณะความต้องการ (Requirements Specification Creation Module) เป็นฟังก์ชันที่ใช้รับคุณลักษณะความต้องการจากผู้ใช้ในรูปแบบฟอร์มและนำมาสกัดเอาข้อมูลที่ระบบจะใช้ในการสร้างตารางตัดสินใจ

4.2.2.4 ส่วนรับคำอธิบายยูสเคส

ส่วนรับคำอธิบายยูสเคส (Use Case Description Creation Module) เป็นฟังก์ชันที่ใช้รับคำอธิบายยูสเคสจากนักวิเคราะห์ระบบในรูปแบบฟอร์มและนำมาสกัดเอาข้อมูลที่ระบบจะใช้ในการสร้างตารางตัดสินใจ

4.2.3 ฐานข้อมูลของระบบ

การออกแบบฐานข้อมูลของระบบจะสัมพันธ์กับแผนภาพยูสเคสของระบบตรวจสอบความถูกต้องของยูสเคส โดยแสดงโครงสร้างฐานข้อมูลด้วยแผนภาพอี-อาร์ (Entity-Relationship Diagram : E-R Diagram) ได้ดังภาพประกอบ 4.5

จากแผนภาพอี-อาร์ ซึ่งแสดงโครงสร้างฐานข้อมูลเชิงสัมพันธ์ (Relation Database Structure) สามารถแปลงเป็นแบบแผนเชิงสัมพันธ์ (Relation Schema) ดังนี้

ReqSpec(Spec_ID, Spec_Namee, Source, Destination, Require, PreCon, PostCon, Description)

Spec_Input(Spec_ID, Variable_ID, Variable_Name, Variable_Possible, Variable_Type)

Spec_Possible(Possible_ID, Spec_ID, Variable_ID, Value, Validity, Result)

Spec_Result(Spec_ID, Result_ID, Value, Validity)

Spec_Relation(Spec_ID, Sub_ID, Relation)

Spec_Decision(Spec_ID, RowNo, ColNo, CellValue)

Usecase(Usecase_ID, Usecase_Name, Actor, Trigger, Stakeholder, Description)

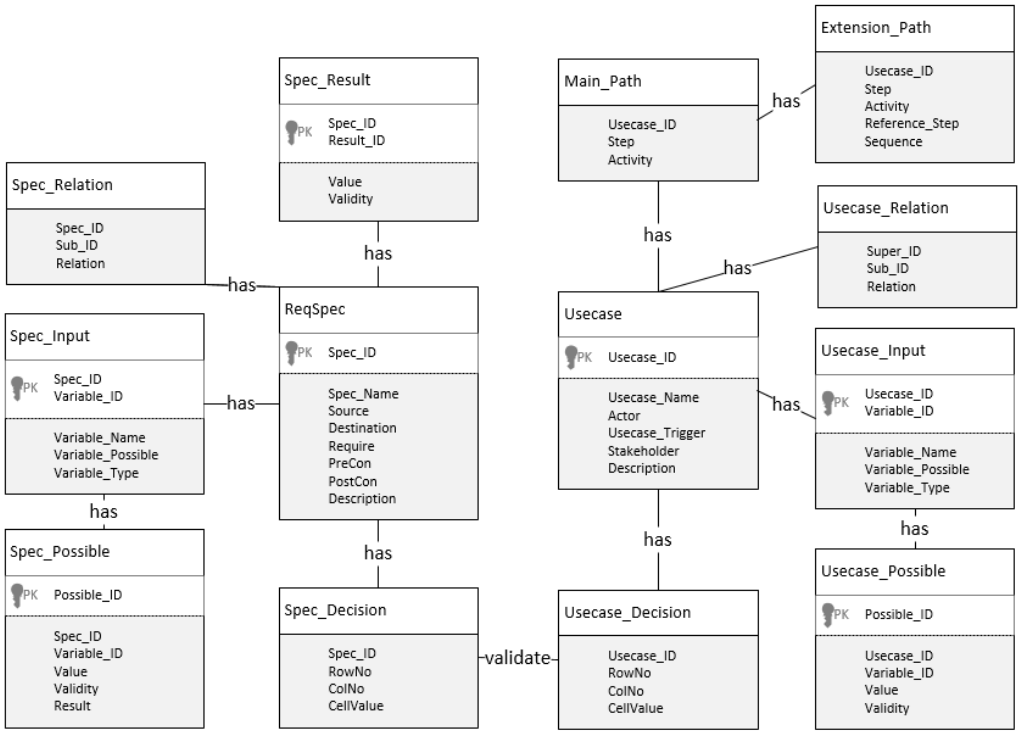
Usecase_Input(Usecase_ID, Variable_ID, Variable_Name, Variable_Possible, Variable_Type)

Usecase_Possible(Possible_ID, Usecase_ID, Variable_ID, Value, Validity)

Main_Path(Usecase_ID, Step, Activity)

Extension_Path(Usecase_ID, Step, Activity, Reference_Step, Sequence)

Usecase_Decision(Usecase_ID, RowNo, ColNo, CellValue)



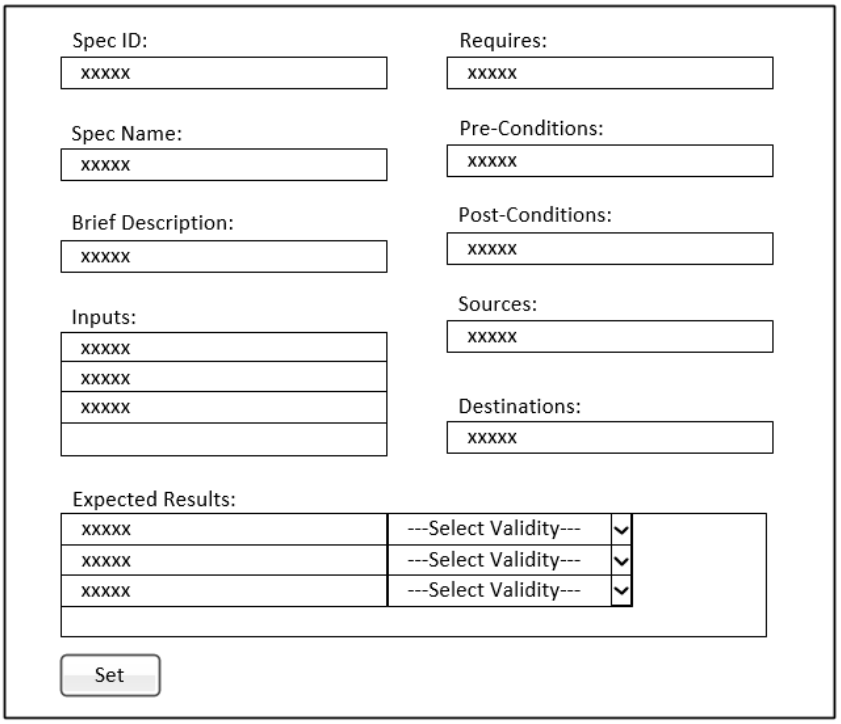
ภาพประกอบ 4.5 แผนภาพอี-อาร์ของฐานข้อมูลระบบ

4.2.4 ส่วนติดต่อกับผู้ใช้

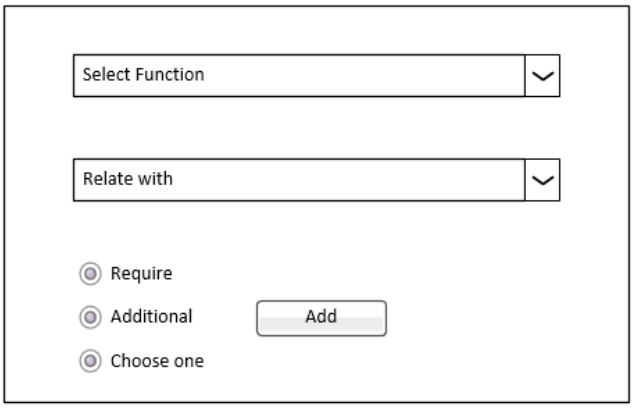
การดำเนินงานและการแสดงผลของระบบตรวจสอบความถูกต้องของยูสเคสได้รับการพัฒนาให้มีฟังก์ชันรองรับการใช้งานตามที่ได้ออกแบบไว้ข้างต้น โดยมีรายละเอียดของส่วนต่างๆ ดังต่อไปนี้

4.2.4.1 การป้อนคุณลักษณะความต้องการเข้าไปในระบบ

ในส่วนนี้ผู้ใช้จะต้องป้อนข้อมูลคุณลักษณะความต้องการลงไปในแบบฟอร์มที่ระบบจัดไว้ให้ โดยรูปแบบฟอร์มที่ใช้อ้างอิงและดัดแปลงมาจากรูปแบบของ Ian Somerville (Somerville, 2011) โดยในส่วนที่ต้องระบุผลลัพธ์ที่คาดหวัง ผู้ใช้จะต้องระบุค่าความถูกต้องของผลลัพธ์นั้นด้วยว่าเป็นผลลัพธ์ที่ถูกต้อง (Valid) หรือไม่ถูกต้อง (Invalid) เมื่อป้อนข้อมูลจนครบแล้ว ให้ผู้ใช้กดปุ่ม “Set” เพื่อทำการบันทึกข้อมูล แสดงดังภาพประกอบ 4.6 และแบบฟอร์มที่ใช้ระบุความสัมพันธ์ระหว่างคุณลักษณะความต้องการ ที่ใช้ระบุว่าแต่ละฟังก์ชันงานมีความสัมพันธ์กับฟังก์ชันงานอื่นๆ ในรูปแบบใดบ้าง โดยให้ผู้ใช้เลือก ID ของฟังก์ชันหลัก และฟังก์ชันที่มาเกี่ยวข้องด้วยจาก Drop Down List จากนั้น เลือกรูปแบบความสัมพันธ์ว่าเป็นแบบจำเป็นต้องมี (Require) เพิ่มเติม (Additional) หรือ เลือกใช้งาน 1 ฟังก์ชัน (Choose one) จากปุ่ม Radio Button จากนั้น กดปุ่ม “Add” เพื่อทำการบันทึกข้อมูล แสดงดังภาพประกอบ 4.7



ภาพประกอบ 4.6 แบบฟอร์มรับข้อมูลทั่วไปของคุณลักษณะความต้องการ

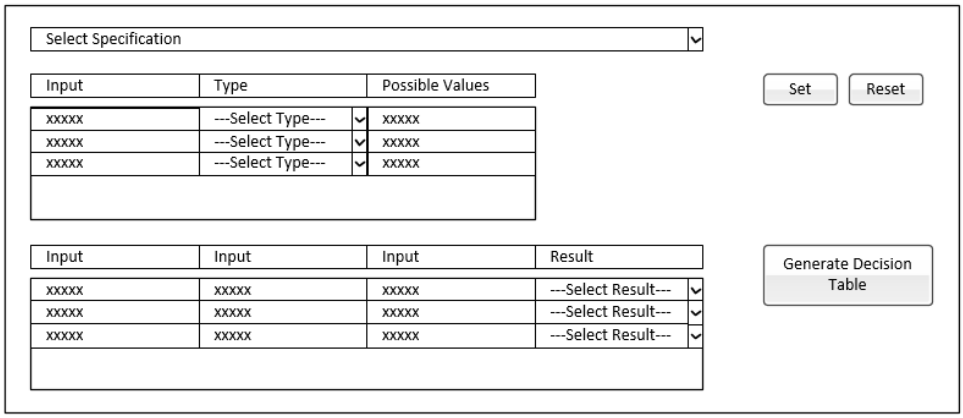


ภาพประกอบ 4.7 แบบฟอร์มรับความสัมพันธ์ของคุณลักษณะความต้องการ

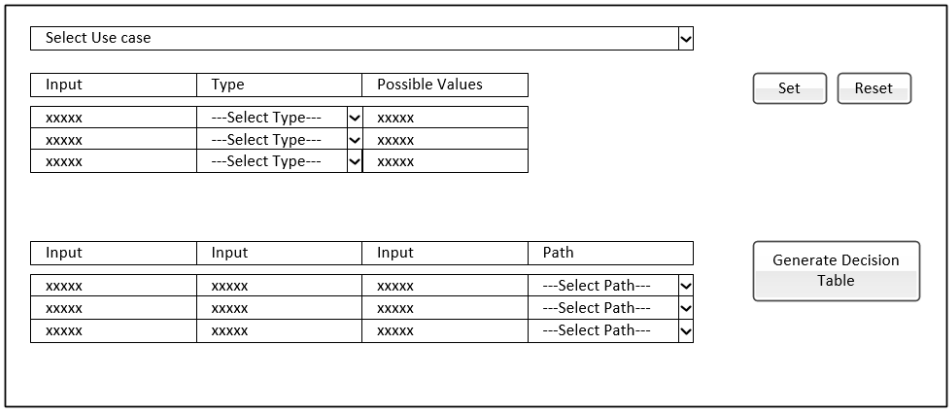
4.2.4.2 การกำหนดรายละเอียดของข้อมูลนำเข้า

ในส่วนนี้เป็นการกำหนดและระบุรายละเอียดของข้อมูลนำเข้าแต่ละตัวที่ผู้ใช้ได้ป้อนเข้ามาในแบบฟอร์มรับคุณลักษณะความต้องการ โดยจะต้องระบุชนิดและค่าที่เป็นไปได้ของข้อมูลนำเข้าแต่ละตัว เมื่อระบุรายละเอียดของข้อมูลนำเข้าจนครบถ้วนแล้วให้ดำเนินการต่อไปโดยการกดปุ่ม “Set” เพื่อสร้างตารางตัดสินใจของคุณลักษณะความต้องการนั้น โดยความแม่นยำในการตรวจสอบความถูกต้องของยูสเคสจะขึ้นอยู่กับความถูกต้องของรายละเอียดที่ระบุไว้ในขั้นตอนนี้เป็นหลัก

โดยการกำหนดรายละเอียดของข้อมูลนำเข้าทั้งข้อมูลนำเข้าของคุณลักษณะความต้องการและข้อมูลนำเข้าของคำอธิบายยูสเคสนั้นจะมีรูปแบบฟอร์มลักษณะเดียวกัน โดยระบบจะสร้างตารางตัดสินใจให้แบบอัตโนมัติจากข้อมูลนำเข้า จากนั้นผู้ใช้จะต้องทำการระบุว่าแต่ละสถานการณ์ในตารางตัดสินใจนั้น สัมพันธ์กับผลลัพธ์ที่คาดหวังใดในกรณีที่สร้างจากคุณลักษณะความต้องการ หรือสัมพันธ์กับเส้นทางใดในกรณีที่สร้างจากคำอธิบายยูสเคส แสดงดังภาพประกอบ 4.8 และภาพประกอบ 4.9



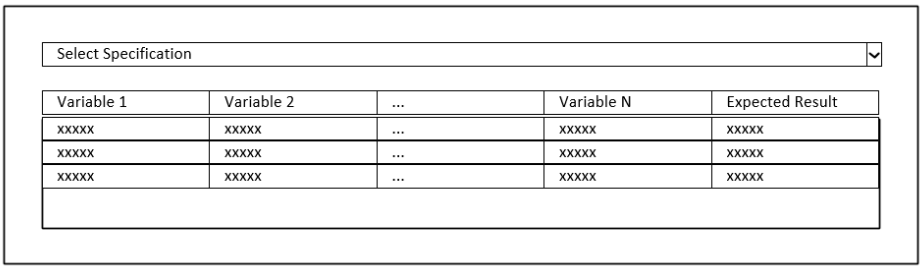
ภาพประกอบ 4.8 แบบฟอร์มรับรายละเอียดข้อมูลนำเข้าของคุณลักษณะความต้องการ



ภาพประกอบ 4.9 แบบฟอร์มรับรายละเอียดข้อมูลนำเข้าของคำอธิบายยูสเคส

4.2.4.3 การแสดงผลตารางตัดสินใจ

ในส่วนนี้จะแสดงตารางตัดสินใจที่ระบบสร้างให้โดยอัตโนมัติ โดยเลือกจาก ID ของคุณลักษณะความต้องการ หรือ ID ของยูสเคสที่ต้องการจะเรียกดู แสดงดังภาพประกอบ 4.10



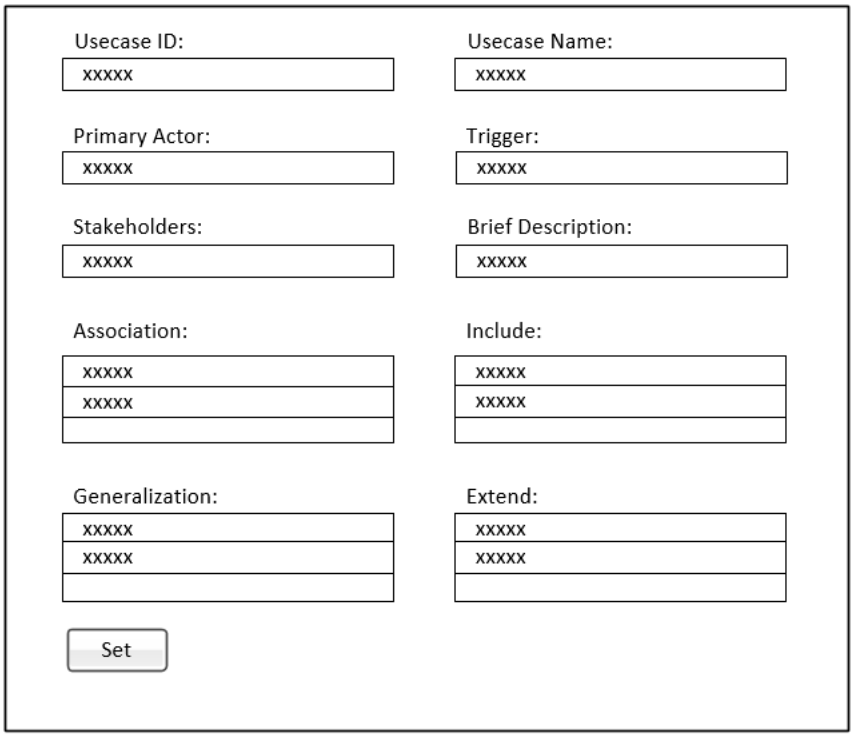
ภาพประกอบ 4.10 ส่วนแสดงผลตารางตัดสินใจ

4.2.4.4 การป้อนคำอธิบายยูสเคสไปในระบบ

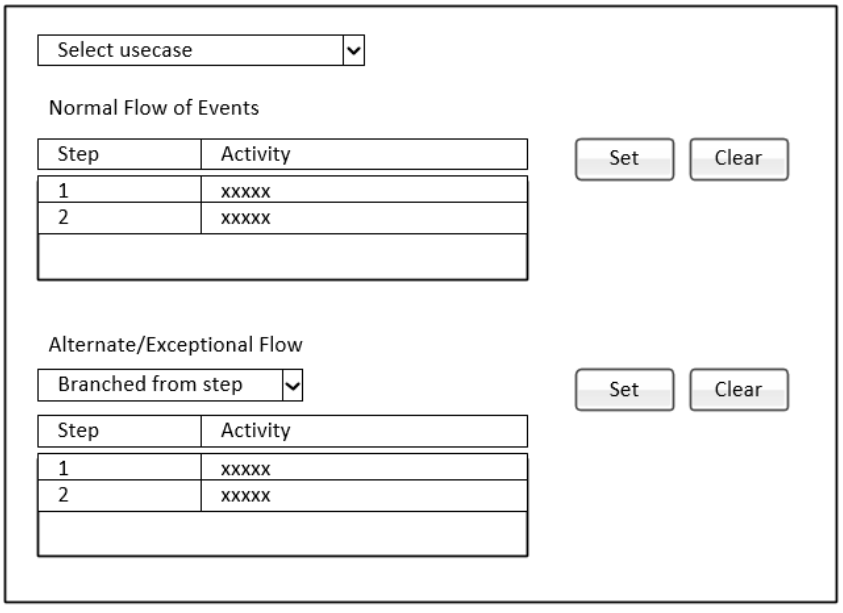
ในส่วนนี้นักวิเคราะห์ระบบจะต้องป้อนข้อมูลคำอธิบายยูสเคสลงไปในแบบฟอร์มที่ระบบจัดไว้ให้ แสดงดังภาพประกอบ 4.11 และภาพประกอบ 4.12

4.2.4.5 การตรวจสอบความถูกต้องของยูสเคส

ในส่วนนี้นักวิเคราะห์ระบบสามารถตรวจสอบความถูกต้องของยูสเคส โดยการระบุ ID ของคุณลักษณะความต้องการและ ID ของยูสเคสที่จะทำการตรวจสอบจาก Drop Down List จากนั้น ระบบจะแสดงรายการคุณลักษณะความต้องการและยูสเคสที่มีความสัมพันธ์กับ ID ที่นักวิเคราะห์ระบบเลือก นักวิเคราะห์ระบบจะต้องจับคู่ ID ของคุณลักษณะความต้องการเข้ากับ ID ของยูสเคสที่สัมพันธ์กัน โดยการเลือก ID จากในรายการที่แสดงแล้วกดปุ่ม “Pair” โดยระบบจะแสดงรายการ ID ได้จับคู่แล้ว เมื่อจับคู่จนครบทุก ID แล้ว ให้นักวิเคราะห์ระบบกดปุ่ม “Validate” เพื่อทำการตรวจสอบ โดยหากจะแก้ไขการจับคู่สามารถทำได้โดยการกดปุ่ม “Reset”



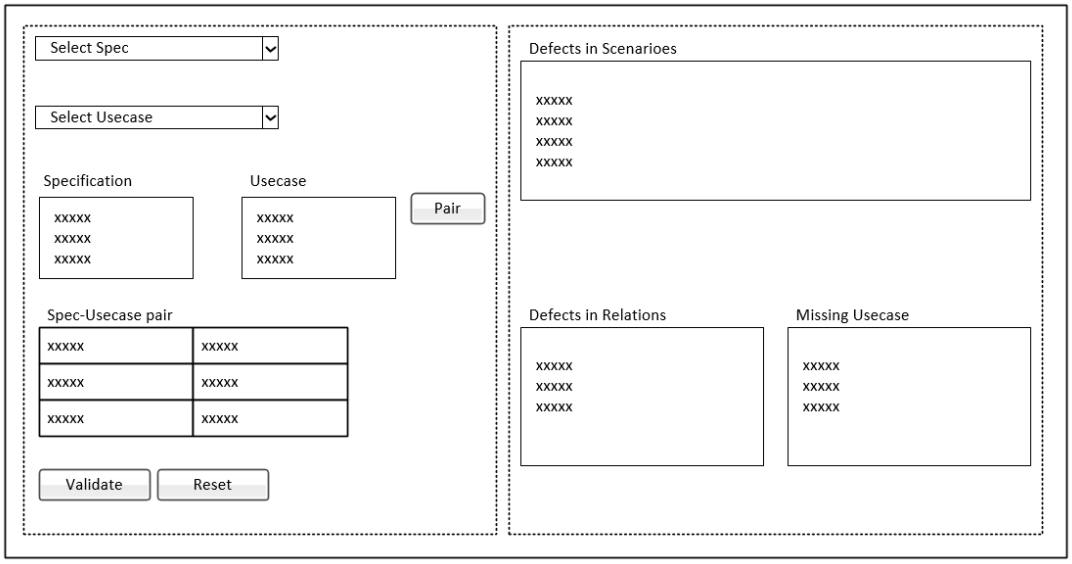
ภาพประกอบ 4.11 แบบฟอร์มรับข้อมูลทั่วไปของคำอธิบายยูสเคส



ภาพประกอบ 4.12 แบบฟอร์มรับข้อมูลเส้นทางของคำอธิบายยูสเคส

ผลของการตรวจสอบความถูกต้องจะแสดงใน 3 รายการ รายการแรกคือ “Defects in Scenario”แสดงความผิดพลาดของสถานการณ์ของแต่ละยูสเคส เช่น ผลการทำงานไม่ตรงกับคุณลักษณะความต้องการ หรือมีบางสถานการณ์ที่ขาดหายไป หากไม่มีข้อผิดพลาดระบบจะแสดงข้อความ “All Scenario Passed!” รายการที่ 2 คือ “Defects in Relationships”

แสดงความผิดพลาดในความสัมพันธ์ระหว่างยูสเคส หากไม่มีข้อผิดพลาดระบบจะแสดงข้อความ “All Relation Passed!” และรายการที่ 3 คือ “Missing Usecase” แสดงรายการคุณลักษณะความต้องการที่ไม่มียูสเคสมารองรับ หากไม่มีข้อผิดพลาดระบบจะแสดงข้อความ “All Usecase Present!” ดังแสดงในภาพประกอบ 4.13



ภาพประกอบ 4.13 แบบฟอร์มตรวจสอบความถูกต้องของยูสเคส

4.3 เครื่องมือซอฟต์แวร์ที่ใช้พัฒนาระบบ

เครื่องมือต่างๆ ที่ใช้ในการพัฒนาระบบตรวจสอบความถูกต้องของยูสเคส ประกอบด้วย

1) Microsoft Windows 7 สำหรับเป็นระบบปฏิบัติการ
2) Microsoft Visual Studio 2013 สำหรับเป็นเครื่องมือ IDE (Integrated Development Environment) ในการพัฒนาระบบ
3) Microsoft SQL Server 2012 สำหรับใช้จัดการฐานข้อมูลเชิงสัมพันธ์
4) Visual C# เป็นภาษาการโปรแกรมที่ใช้ในการพัฒนาระบบ

4.4 การพัฒนาระบบ

จากการที่ได้วิเคราะห์และออกแบบระบบตรวจสอบความถูกต้องของยูสเคส สามารถนำมาพัฒนาระบบโดยแยกการทำงานของระบบออกเป็นคลาส ซึ่งสามารถแสดงโครงสร้างดังภาพประกอบ 4.14 สามารถอธิบายการทำงานแต่ละคลาสได้ดังนี้

1) UseCaseValidation

เป็นคลาสสำหรับตรวจสอบความถูกต้องของยูสเคส โดยการดึงข้อมูลจากตารางตัดสินใจที่สร้างไว้มาดำเนินการ แบ่งการทำงานเป็น 2 เมธ็อด คือ validateScenario() สำหรับตรวจสอบเหตุการณ์ภายในยูสเคส และ validateRelation() สำหรับตรวจสอบความสัมพันธ์ระหว่างยูสเคส

2) GenerateDecisionTable

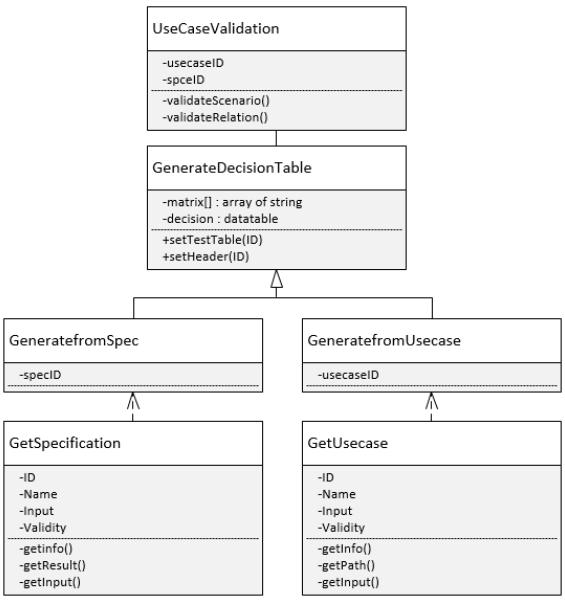
เป็นคลาสสำหรับสร้างตารางตัดสินใจ แบ่งการทำงานเป็น 2 เมธ็อด คือ setTestTable() สำหรับสร้างตารางตัดสินใจ และ setHeader() สำหรับตั้งชื่อหัวตารางตัดสินใจให้เป็นชื่อของข้อมูลนำเข้า โดยมีคลาสย่อยคือ GeneratefromSpec สำหรับสร้างตารางตัดสินใจจากคุณลักษณะความต้องการ และ GeneratefromUsecase สำหรับสร้างตารางตัดสินใจจากคำอธิบายยูสเคส

3) GetSpecification

เป็นคลาสสำหรับรับคุณลักษณะความต้องการที่เขียนอยู่ในรูปแบบฟอร์มจากผู้ใช้ แบ่งการทำงานเป็น 3 เมธ็อด คือ getInfo() สำหรับรับข้อมูลทั่วไป getResult() สำหรับรับผลลัพธ์ที่คาดหวัง และ getInput() สำหรับรับข้อมูลนำเข้าและรายละเอียดของข้อมูลนำเข้าแต่ละตัว

4) GetUsecase

เป็นคลาสสำหรับรับคำอธิบายยูสเคสจากนักพัฒนาระบบ แบ่งการทำงานเป็น 3 เมธ็อด คือ getInfo() สำหรับรับข้อมูลทั่วไป getPath() สำหรับรับเส้นทางการทำงานภายในยูสเคส และ getInput() สำหรับรับข้อมูลนำเข้าและรายละเอียดของข้อมูลนำเข้าแต่ละตัว



ภาพประกอบ 4.14 แผนภาพคลาสของระบบ

บทที่ 5

การประเมินประสิทธิภาพ

เนื้อหาในบทนี้กล่าวถึงการประเมินประสิทธิภาพของงานวิจัย ซึ่งมีการดำเนินการและผลการประเมินดังต่อไปนี้

5.1 การออกแบบการประเมินระบบ

ตามขอบเขตงานวิจัยนี้มีการดำเนินการ 2 ขั้นตอนหลัก คือ การสร้างกรณีทดสอบจากคุณลักษณะความต้องการและคำอธิบายยูสเคส และการตรวจสอบความถูกต้องของยูสเคส ดังนั้นในการประเมินระบบ ทางผู้วิจัยจึงจำเป็นต้องประเมินประสิทธิภาพของทั้ง 2 ขั้นตอนนั้น โดยจะประเมินความครบถ้วนสมบูรณ์ (Completeness) ของกรณีทดสอบ และ ความถูกต้อง (Correctness) ของการตรวจจับความถูกต้องของยูสเคส ด้วยการใช้กรณีศึกษาขั้นพื้นฐาน (Based-Line Case Study) เป็นตัวเปรียบเทียบการทำงานระหว่างผู้เข้าร่วมประเมินและการดำเนินการโดยระบบที่พัฒนาขึ้น และทำการจับเวลาที่ผู้เข้าร่วมประเมินใช้ในการสร้างกรณีทดสอบเพื่อวัดความเร็วในการดำเนินการ (Time) โดยก่อนมีการประเมินระบบจริงได้นำแบบประเมินที่สร้างขึ้นไปประเมินขั้นต้นโดยทดลองนำร่องกับนักศึกษาระดับปริญญาโท และนักศึกษาระดับปริญญาเอก ของภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์ จำนวน 4 คน ก่อนนำไปประเมินระบบกับผู้เชี่ยวชาญในขั้นถัดไป

5.1.1 การประเมินประสิทธิภาพของการตรวจสอบความถูกต้อง

ในงานวิจัยนี้ได้ประเมินประสิทธิภาพของการตรวจสอบความถูกต้องของยูสเคสโดยผู้เข้าร่วมประเมิน โดยการให้ผู้เข้าร่วมประเมินพิจารณาข้อกำหนดความต้องการและคำอธิบายยูสเคส และระบุข้อผิดพลาดในคำอธิบายยูสเคสนั้น มาเปรียบเทียบกับผลการตรวจสอบความถูกต้องของยูสเคสโดยระบบที่พัฒนาขึ้น เพื่อเป็นการเปรียบเทียบประสิทธิภาพในการตรวจสอบความถูกต้องของยูสเคส ระหว่างผู้เข้าร่วมประเมินกับระบบที่ดำเนินการด้วยวิธีการที่นำเสนอ โดยระบบที่พัฒนาขึ้นมามีการดำเนินการตรวจสอบความถูกต้องของยูสเคสโดย รับข้อมูลคุณลักษณะความต้องการและคำอธิบายยูสเคสเพื่อนำมาสร้างเป็นตารางตัดสินใจ และนำตารางตัดสินใจที่สร้างจากคุณลักษณะความต้องการมาเปรียบเทียบกับตารางตัดสินใจที่สร้างจากคำอธิบายยูสเคสโดยอัตโนมัติ เพื่อหาเหตุการณ์ที่ยูสเคสไม่ตอบสนองต่อความต้องการ หรือทำงานผิดไปจากความต้องการ

การตรวจสอบเพื่อเปรียบเทียบประสิทธิภาพในการตรวจสอบความถูกต้องของยูสเคส ได้กำหนดคะแนนจากจำนวนข้อผิดพลาดเป็น 1 คะแนนต่อรายการข้อผิดพลาด สำหรับกรณีตัวอย่างระบบให้บริการยืม-คืน จะถูกกำหนดคะแนนของแต่ละฟังก์ชันงานได้ดังตารางที่ 5.1

ตารางที่ 5.1 คะแนนของแต่ละฟังก์ชันงานของกรณีทดสอบความถูกต้อง

คะแนนของฟังก์ชัน	**Check out (คะแนน)**	**Check in (คะแนน)**	**Renew (คะแนน)**	**Hold item (คะแนน)**	**Fine (คะแนน)**	**รวม (คะแนน)**
	2	1	0	2	1	6

ผลการทดลองประเมินประสิทธิภาพที่ได้จากการตรวจสอบความถูกต้องของยูสเคสโดยระบบที่พัฒนาขึ้น ดังแสดงในตารางที่ 5.2

ตารางที่ 5.2 ผลการทดลองประเมินประสิทธิภาพ ความถูกต้องของระบบที่พัฒนาขึ้น

กลุ่มผู้ประเมิน	**Check out (คะแนน)**	**Check in (คะแนน)**	**Renew (คะแนน)**	**Hold item (คะแนน)**	**Fine (คะแนน)**	**รวม (คะแนน)**	**ความตรงกัน**
ระบบ	2	1	0	2	1	6	100%

5.1.2 การประเมินประสิทธิภาพของการสร้างกรณีทดสอบ

ในงานวิจัยนี้ได้ประเมินประสิทธิภาพของการสร้างกรณีทดสอบจากยูสเคสโดยผู้เข้าร่วมประเมิน โดยการให้ผู้เข้าร่วมประเมินแจกแจงสถานการณ์ที่ได้จากคำอธิบายยูสเคสที่เขียนอยู่ในรูปแบบฟอร์ม แล้วนำคำอธิบายยูสเคสที่เขียนอยู่ในรูปแบบฟอร์มชุดเดียวกันนี้ มาสร้างกรณีทดสอบด้วยระบบที่พัฒนาขึ้น จากนั้นนำกรณีทดสอบทั้ง 2 ชุดมาทำการประเมินประสิทธิผลความครบถ้วนสมบูรณ์ของกรณีทดสอบ เพื่อเป็นการเปรียบเทียบประสิทธิภาพในการสร้างกรณีทดสอบจากยูสเคส ระหว่างผู้เข้าร่วมประเมินกับระบบที่ดำเนินการด้วยวิธีการที่นำเสนอ โดยระบบที่พัฒนาขึ้นมามีการดำเนินการสร้างกรณีทดสอบโดยรับข้อมูลคำอธิบายยูสเคส รับข้อมูลนำเข้าของยูสเคสพร้อมทั้งระบุค่าที่เป็นไปได้ จากนั้นระบบจะสร้างตารางตัดสินใจจากค่าที่เป็นไปได้ของทุกข้อมูลนำเข้าโดยอัตโนมัติ

การตรวจสอบเพื่อเปรียบเทียบประสิทธิภาพในการสร้างตารางตัดสินใจจากคำอธิบายยูสเคส ได้กำหนดคะแนนจากจำนวนสถานการณ์ (Scenario) เป็น 1 คะแนนต่อรายการ สำหรับกรณีตัวอย่างระบบให้บริการยืม-คืน จะถูกกำหนดคะแนนของแต่ละฟังก์ชันงานได้ดังตารางที่ 5.3

ตารางที่ 5.3 คะแนนของแต่ละฟังก์ชันงานของกรณีทดสอบความครบถ้วนสมบูรณ์

คะแนนของสถานการณ์	**Check out (คะแนน)**	**Check in (คะแนน)**	**Renew (คะแนน)**	**Hold item (คะแนน)**	**Fine (คะแนน)**	**รวม (คะแนน)**
	15	7	4	13	4	43

ผลการทดลองประเมินประสิทธิภาพที่ได้จากการสร้างกรณีทดสอบโดยระบบที่พัฒนาขึ้น ดังแสดงในตารางที่ 5.4

ตารางที่ 5.4 ผลการทดลองประเมินประสิทธิภาพ ความถูกต้องของระบบที่พัฒนาขึ้น

กลุ่มผู้ประเมิน	**Check out (คะแนน)**	**Check in (คะแนน)**	**Renew (คะแนน)**	**Hold item (คะแนน)**	**Fine (คะแนน)**	**รวม (คะแนน)**	**ความตรงกัน**
ระบบ	54	8	4	36	4	106	100%

5.1.3 การเลือกกลุ่มผู้เข้าร่วมประเมิน

5.1.3.1 การประเมินขั้นต้น

การประเมินประสิทธิภาพของระบบขั้นต้นนั้น ผู้วิจัยได้คัดเลือกนักศึกษาระดับปริญญาโท และระดับปริญญาเอก จำนวน 4 ท่าน ได้แก่

1) นายสกล จันทร์ขจร
นักศึกษาระดับปริญญาเอก ภาควิชาวิทยาการคอมพิวเตอร์

2) นางสาวฟูไดละห์ ดือมอง
นักศึกษาระดับปริญญาเอก ภาควิชาวิทยาการคอมพิวเตอร์

3) นายเจษฎา ดิษโสภา
นักศึกษาระดับปริญญาโท ภาควิชาวิทยาการคอมพิวเตอร์

4) นางสาวณิชาภัทร ปิ่นโพธิ์
นักศึกษาระดับปริญญาโท ภาควิชาวิทยาการคอมพิวเตอร์

5.1.3.2 การประเมินโดยกลุ่มผู้เชี่ยวชาญ

การประเมินประสิทธิภาพของระบบโดยกลุ่มผู้เชี่ยวชาญนั้น ผู้วิจัยได้คัดเลือกผู้เชี่ยวชาญทางด้านการวิเคราะห์และพัฒนาระบบ จำนวน 4 ท่าน ได้แก่

1) นายชำนาญ อินทสโร

ตำแหน่งนักวิชาการคอมพิวเตอร์ชำนาญการพิเศษ ศูนย์คอมพิวเตอร์ ระดับและวุฒิการศึกษาสูงสุด คือ ระดับปริญญาตรี

2) นางสาวศศิธร คงหนู

ตำแหน่งนักวิชาการคอมพิวเตอร์ ศูนย์คอมพิวเตอร์ ระดับและวุฒิการศึกษาสูงสุด คือ ระดับปริญญาโท

3) นายสุทธิพงษ์ อุดมประเสริฐกุล

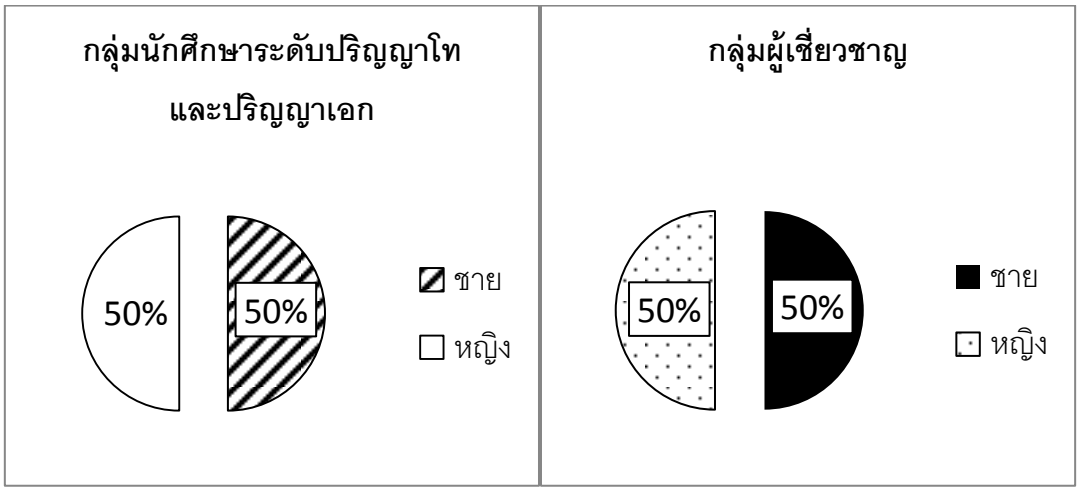
ตำแหน่งนักวิชาการคอมพิวเตอร์ ศูนย์คอมพิวเตอร์ ระดับและวุฒิการศึกษาสูงสุด คือ ระดับปริญญาโท

4) นางสาวอัญชลี พัฒนพันธ์ชัย

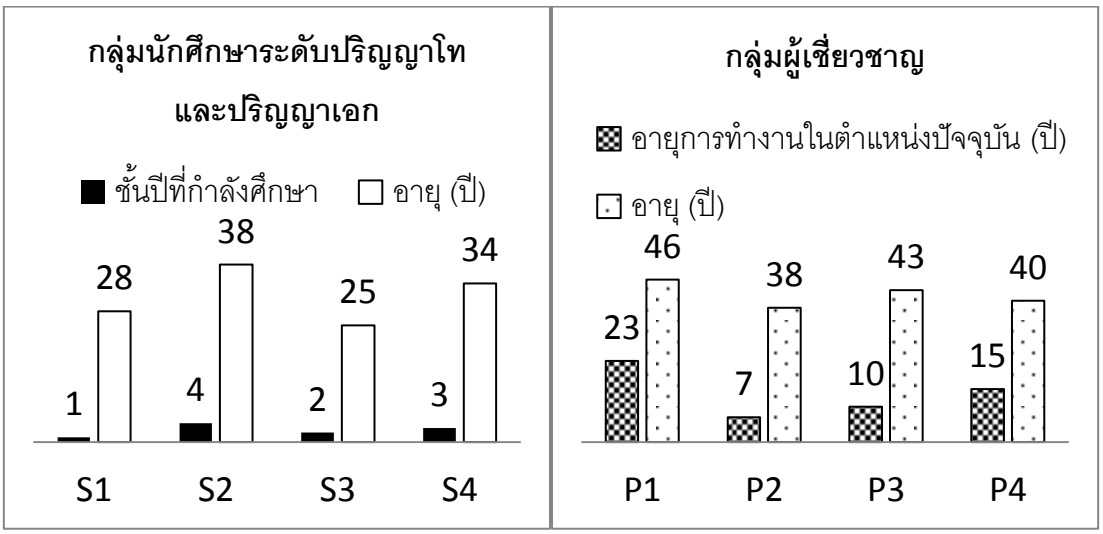
ตำแหน่งนักวิชาการคอมพิวเตอร์ชำนาญการ หน่วยเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ ระดับและวุฒิการศึกษาสูงสุด คือ ระดับปริญญาโท

ในวันที่ทำการประเมินโดยกลุ่มผู้เชี่ยวชาญ มีผู้ชี่ยวชาญ 1 ท่านที่ได้ตอบรับคำเชิญแต่ไม่สามารถเข้าร่วมทำการประเมินได้เนื่องจากปัญหาทางด้านสุขภาพ ดังนั้น ในการประเมินระบบจึงมีผู้เชี่ยวชาญมาร่วมประเมิน 4 ท่าน

ในการวิเคราะห์ข้อมูล ได้กำหนดหมายเลขรหัส S1 ถึง S4 แทนผู้เข้าร่วมประเมินในกลุ่มนักศึกษาระดับปริญญาโทและปริญญาเอก และได้กำหนดหมายเลขรหัส P1 ถึง P4 แทนผู้เข้าร่วมประเมินในกลุ่มผู้เชี่ยวชาญโดยข้อมูลส่วนตัวในด้านเพศ อายุ ชั้นปีที่กำลังศึกษาของกลุ่มนักศึกษา และอายุงานในตำแหน่งปัจจุบันของกลุ่มผู้เชี่ยวชาญที่เข้าร่วมประเมินระบบได้ดังภาพประกอบ 5.1 และภาพประกอบ 5.2 ตามลำดับ



ภาพประกอบ 5.1 แผนภูมิวงกลมแสดงอัตราส่วนร้อยละของผู้เข้าร่วมประเมินแบ่งตามเพศ



ภาพประกอบ 5.2 แผนภูมิแท่งแสดงอายุและอายุงานในตำแหน่งปัจจุบันหรือชั้นปีที่กำลังศึกษาของผู้เข้าร่วมประเมิน

5.2 ขั้นตอนการทดลองประเมินระบบ

ขั้นตอนการประเมินประสิทธิภาพของระบบ มีขั้นตอนหลักทั้งหมด 3 ขั้นตอนคือ 1) ขั้นตอนการบรรยายแบบย่อ 2) ขั้นตอนการตรวจสอบความถูกต้องของยูสเคส และ 3) ขั้นตอนการสร้างตารางตัดสินใจจากยูสเคส

ขั้นตอนที่ 1 การบรรยายแบบย่อ

ขั้นตอนการบรรยายแบบย่อ เป็นขั้นตอนการอธิบายให้ผู้เข้าร่วมประเมินเข้าใจถึงขั้นตอนและวิธีการในการประเมิน รูปแบบของคุณลักษณะความต้องการที่ใช้ และรูปแบบของคำอธิบายยูสเคสที่ใช้ จากนั้น มอบกรณีศึกษาให้แก่ผู้เข้าร่วมประเมิน เป็นแผนภาพยูสเคสและคุณลักษณะความต้องการของฟังก์ชันในระบบให้บริการยืม-คืน จำนวน 5 ฟังก์ชัน

ขั้นตอนที่ 2 การตรวจสอบความถูกต้องของยูสเคส

ขั้นตอนการตรวจสอบความถูกต้องของยูสเคส จะดำเนินการโดยมอบโจทย์สำหรับการทดลองวิจัยที่ 1 แก่ผู้เข้าร่วมประเมินเป็นคำอธิบายยูสเคสจำนวน 5 เคส ซึ่งบางเคสมีข้อผิดพลาดอยู่ ในขณะที่บางเคสไม่มีข้อผิดพลาดใดๆ ภาระงานของโจทย์สำหรับการทดลองวิจัยที่ 1 คือให้ผู้เข้าร่วมประเมินตรวจสอบความถูกต้อง และระบุข้อผิดพลาดที่พบในคำอธิบายยูสเคสว่าข้อผิดพลาดที่ตรวจพบมีอะไรบ้าง โดยในขั้นตอนนี้ให้ผู้เข้าร่วมประเมินดำเนินการด้วยมือ

ขั้นตอนที่ 3 การสร้างตารางตัดสินใจจากยูสเคส

ขั้นตอนการสร้างตารางตัดสินใจจากยูสเคส จะดำเนินการโดยมอบโจทย์สำหรับการทดลองวิจัยที่ 2 แก่ผู้เข้าร่วมประเมินเป็นคำอธิบายยูสเคสจำนวน 5 เคส ภาระงานของโจทย์สำหรับการทดลองวิจัยที่ 2 คือ ให้ผู้เข้าร่วมประเมินสร้างตารางตัดสินใจจากคำอธิบายยูสเคสที่มอบให้ โดยในขั้นตอนนี้ให้ผู้เข้าร่วมประเมินดำเนินการด้วยมือเช่นเดียวกับในขั้นตอนที่ 2

5.3 เครื่องมือที่ใช้ในการประเมินระบบ

1) ระบบตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล

2) โจทย์สำหรับทำการประเมิน

3) แบบสอบถาม

แบบสอบถามในการประเมินความพึงพอใจในการใช้งานระบบ ซึ่งใช้เฉพาะในการประเมินโดยกลุ่มผู้เชี่ยวชาญ แบ่งเป็น 3 ตอน คือ

ตอนที่ 1 ส่วนข้อมูลเกี่ยวกับผู้ตอบแบบสอบถาม

ตอนที่ 2 แบบสอบถามความพึงพใจในการใช้งานระบบ เป็นแบบสอบถามปลายปิด จำนวน 11 ข้อ

ตอนที่ 3 ส่วนคำถามปลายเปิดสำหรับข้อเสนอแนะ จำนวน 3 ข้อ

4) วิธีวิเคราะห์ข้อมูลของการประเมินความพึงพอใจ

ผู้วิจัยรวบรวมแบบสอบถามและวิเคราะห์ผลความพึงพอใจของผู้เข้าร่วมประเมิน ซึ่งผู้วิจัยกำหนดเกณฑ์ตามแนวทาง Likert Scale ดังนี้

4.50 – 5.00 ความพึงพอใจมากที่สุด

3.50 – 4.49 ความพึงพอใจมาก

2.50 – 3.49 ความพึงพอใจปานกลาง

1.50 – 2.49 ความพึงพอใจน้อย

0.50 – 1.49 ความพึงพอใจน้อยที่สุด

สถิติที่ใช้ในการวิเคราะห์ความพึงพอใจ ได้แก่

1) ค่าเฉลี่ย $(\overline{\mathrm{X}})$

$$\overline{\mathrm{X}} = \frac{\sum_{i=1}^{n} x_i}{n}$$

โดย $\overline{X}$ คือ ค่าเฉลี่ย

x_i คือ ข้อมูลแต่ละจำนวนตั้งแต่ตัวที่ 1 ถึงตัวที่ i

$\sum_{i=1}^{n} x_i$ คือ ผลรวมคะแนนทั้งหมด

n คือ จำนวนตัวอย่าง

2) ค่าส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation)

$$SD = \sqrt{\frac{\sum_{i=1}^{n} |x_i - \bar{x}|^2}{n}}$$

โดย SD คือ ส่วนเบี่ยงเบนมาตรฐาน

x_i คือ ข้อมูลแต่ละจำนวนตั้งแต่ตัวที่ 1 ถึงตัวที่ i

$\overline{x}$ คือ ค่าเฉลี่ย

n คือ จำนวนตัวอย่าง

5.4 ผลการประเมินประสิทธิภาพของระบบ

5.4.1 ผลการประเมินระบบโดยกลุ่มนักศึกษาระดับปริญญาโทและปริญญาเอก

ผลการเปรียบเทียบประสิทธิภาพของการตรวจจับข้อผิดพลาดในยูสเคส และการสร้างตารางตัดสินใจจากยูสเคส เปรียบเทียบกับระบบที่พัฒนาขึ้น สามารถแสดงดังตาราง 5.5 และตาราง 5.6

ตารางที่ 5.5 ผลการประเมินประสิทธิภาพการตรวจสอบความถูกต้องของยูสเคสโดยนักศึกษาระดับปริญญาโทและปริญญาเอก

กลุ่มผู้ประเมิน	Check out (คะแนน)	Check in (คะแนน)	Renew (คะแนน)	Hold item (คะแนน)	Fine (คะแนน)	รวม (คะแนน)	ความตรงกัน
S1	2	1	0	0	0	3	50.00%
S2	1	0	0	0	0	1	16.67%
S3	1	1	0	1	0	3	50.00%
S4	0	0	0	0	1	1	16.67%
ค่าเฉลี่ย	**1**	**0.5**	**0**	**0.25**	**0.25**	**2**	**33.33%**

ตารางที่ 5.6 ผลการประเมินประสิทธิภาพการสร้างตารางตัดสินใจจากคำอธิบายยูสเคสโดยนักศึกษาระดับปริญญาโทและปริญญาเอก

กลุ่มผู้ประเมิน	Check out (คะแนน)	Check in (คะแนน)	Renew (คะแนน)	Hold item (คะแนน)	Fine (คะแนน)	รวม (คะแนน)	ความตรงกัน
S1	11	7	4	5	4	31	72.10%
S2	11	7	4	5	4	31	72.10%
S3	14	7	4	11	4	40	93.02%
S4	14	7	4	6	4	35	81.40%
ค่าเฉลี่ย	**12.5**	**7**	**4**	**6.75**	**4**	**34.25**	**79.65%**

โดยในการทดลองประเมินนำร่องนี้ ผู้เข้าประเมินได้รับการอธิบายแบบย่อเป็นเวลา 20 นาที และใช้เวลาเฉลี่ย 1 ชั่วโมง 15 นาทีในการทำแบบประเมินทั้ง 2 ขั้นตอน

5.4.1.1 สรุปผลการประเมินประสิทธิภาพของระบบโดยกลุ่มนักศึกษาระดับปริญญาโทและปริญญาเอก

จากผลการประเมินประสิทธิภาพของระบบตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล สามารถสรุปได้ดังนี้

1) ด้านความถูกต้องของการตรวจสอบความถูกต้องของยูสเคส

ความถูกต้องของการตรวจสอบความถูกต้องโดยผู้เข้าร่วมประเมินโดยเฉลี่ยคิดเป็นร้อยละ 33.33 ของข้อผิดพลาดทั้งหมด

2) ด้านความครบถ้วนของการสร้างตารางตัดสินใจจากคำอธิบายยูสเคส

ความครบถ้วนของการสร้างตารางตัดสินใจโดยผู้เข้าร่วมประเมินโดยเฉลี่ยคิดเป็นร้อยละ 79.65 ของจำนวนเหตุการณ์ที่สามารถเกิดขึ้นได้

5.4.2 ผลการประเมินระบบโดยกลุ่มผู้เชี่ยวชาญ

ผลการเปรียบเทียบประสิทธิภาพของการตรวจจับข้อผิดพลาดในยูสเคส และการสร้างตารางตัดสินใจจากยูสเคส เปรียบเทียบกับระบบที่พัฒนาขึ้น สามารถแสดงดังตาราง 5.7 และตาราง 5.8

ตาราง 5.7 ผลการประเมินประสิทธิภาพการตรวจสอบความถูกต้องของยูสเคสโดยกลุ่มผู้เชี่ยวชาญ

กลุ่มผู้ประเมิน	Check out (คะแนน)	Check in (คะแนน)	Renew (คะแนน)	Hold item (คะแนน)	Fine (คะแนน)	รวม (คะแนน)	ความตรงกัน
P1	2	0	0	1	0	3	50.00%
P2	2	0	0	1	0	3	50.00%
P3	1	0	0	0	0	1	16.67%
P4	2	0	0	0	0	2	33.33%
ค่าเฉลี่ย	**1.75**	**0**	**0**	**0.5**	**0**	**2.25**	**37.50%**

ตารางที่ 5.8 ผลการประเมินประสิทธิภาพการสร้างตารางตัดสินใจจากคำอธิบายยูสเคสโดยกลุ่มผู้เชี่ยวชาญ

กลุ่มผู้ประเมิน	Check out (คะแนน)	Check in (คะแนน)	Renew (คะแนน)	Hold item (คะแนน)	Fine (คะแนน)	รวม (คะแนน)	ความตรงกัน
P1	15	7	4	13	4	43	100%
P2	12	7	3	13	4	39	90.70%

ตารางที่ 5.8 ผลการประเมินประสิทธิภาพการสร้างตารางตัดสินใจจากคำอธิบายยูสเคสโดยกลุ่มผู้เชี่ยวชาญ (ต่อ)

กลุ่มผู้ประเมิน	Check out (คะแนน)	Check in (คะแนน)	Renew (คะแนน)	Hold item (คะแนน)	Fine (คะแนน)	รวม (คะแนน)	ความตรงกัน
P3	15	7	4	13	4	43	100%
P4	13	7	4	10	4	38	88.37%
ค่าเฉลี่ย	**13.75**	**7**	**3.75**	**12.25**	**4**	**40.75**	**94.77%**

โดยในการทดลองประเมินระบบโดยผู้เชี่ยวชาญนี้ ผู้เข้าร่วมประเมินได้รับการอธิบายแบบย่อเป็นเวลา 20 นาที และใช้เวลาเฉลี่ย 1 ชั่วโมง 28 นาที ในการดำเนินการทั้ง 2 ขั้นตอน

5.4.2.1 สรุปผลการประเมินประสิทธิภาพของระบบโดยกลุ่มผู้เชี่ยวชาญ

จากผลการประเมินประสิทธิภาพของระบบตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล สามารถสรุปได้ดังนี้

1) ด้านความถูกต้องของการตรวจสอบความถูกต้องของยูสเคส

ความถูกต้องของการตรวจสอบความถูกต้องโดยผู้เข้าร่วมประเมินโดยเฉลี่ยคิดเป็นร้อยละ 37.50 ของข้อผิดพลาดทั้งหมด

2) ด้านความครบถ้วนของการสร้างตารางตัดสินใจจากคำอธิบายยูสเคส

ความครบถ้วนของการสร้างตารางตัดสินใจโดยผู้เข้าร่วมประเมินโดยเฉลี่ยคิดเป็นร้อยละ 94.77 ของจำนวนเหตุการณ์ที่สามารถเกิดขึ้นได้

5.4.2.2 ข้อสังเกตจากการประเมินประสิทธิภาพของระบบ

ในการสร้างตารางตัดสินใจโดยผู้ทดลองนั้น จะสร้างโดยการคำนึงถึงลำดับของข้อมูลและเงื่อนไขในเหตุการณ์ โดยอ้างอิงมาจากสถานการณ์จริง เช่น จากฟังก์ชันการยืม ในเหตุการณ์ผู้ยืมไม่ได้เป็นสมาชิก ซึ่งไม่สามารถยืมทรัพยากรได้ ผู้เข้าร่วมประเมินจะไม่นำความเป็นไปได้จากรหัสทรัพยากรมาพิจารณา ในขณะที่การสร้างตารางตัดสินใจโดยระบบนั้น ระบบจะสร้างตารางตัดสินใจที่เป็นไปได้มาทั้งหมด โดยไม่ได้คำนึงถึงลำดับของข้อมูลและเงื่อนไขในเหตุการณ์ ส่งผลให้ตารางตัดสินใจที่สร้างโดยระบบนั้นมีความเป็นไปได้ครบถ้วนมากกว่าการสร้างโดยผู้เข้าร่วมประเมิน แต่มีจำนวนสถานการณ์มากเกินความจำเป็น

5.5 ผลการประเมินความพึงพอใจของระบบ

ผลจากการประเมินความพึงพอใจต่อระบบที่พัฒนาขึ้น แสดงดังตาราง 5.9 ถึงตาราง 5.11

ตารางที่ 5.9 ค่าเฉลี่ย ส่วนเบี่ยงเบนมาตรฐาน และระดับความพึงพอใจในด้านการทำงานของระบบ

1. ความพึงพอใจด้านการทำงานของระบบ	$\bar{x}$	SD	**ระดับความพึงพอใจ**
1.1 ความถูกต้องของการทำงาน	4.50	0.5	ความพึงพอใจมากที่สุด
1.2 ความสมบูรณ์ของผลลัพธ์	4.50	0.5	ความพึงพอใจมากที่สุด
1.3 ความเร็วในการประมวลผล	5	0	ความพึงพอใจมากที่สุด
1.4 ความสามารถในการป้องกันความผิดพลาดจากผู้ใช้	3.75	0.96	ความพึงพอใจมาก

ตารางที่ 5.10 ค่าเฉลี่ย ส่วนเบี่ยงเบนมาตรฐาน และระดับความพึงพอใจในด้านการติดต่อกับผู้ใช้งาน

2. ความพึงพอใจในด้านการติดต่อกับผู้ใช้งาน	$\bar{x}$	***SD***	**ระดับความพึงพอใจ**
2.1 การใช้งานง่ายและเป็นมิตรกับผู้ใช้ เช่น การจัดลำดับข้อมูลบนหน้าจอ	3.75	0.5	ความพึงพอใจมาก
2.2 ความเหมาะสมในการวางตำแหน่งองค์ประกอบบนหน้าจอ	3.5	0.58	ความพึงพอใจมาก
2.3 ความชัดเจนในการสื่อความหมายของส่วนต่างๆ	3.5	0.58	ความพึงพอใจมาก
2.4 ในกรณีที่มีข้อผิดพลาดมีการแสดงให้เห็นอย่างชัดเจน	3.75	0.5	ความพึงพอใจมาก
2.5 ความสวยงามของการแสดงผลบนหน้าจอ เช่น การใช้สีหรือขนาดของตัวอักษร	3.5	0.58	ความพึงพอใจมาก

ตารางที่ 5.11 ค่าเฉลี่ย ส่วนเบี่ยงเบนมาตรฐาน และระดับความพึงพอใจในด้านคุณค่าของระบบ

3. ความพึงพอใจในด้านคุณค่าของระบบ	$\bar{x}$	***SD***	**ระดับความพึงพอใจ**
3.1 ช่วยลดเวลาในการตรวจสอบความถูกต้องของยูสเคส	4.25	0.58	ความพึงพอใจมาก
3.2 ช่วยเพิ่มความถูกต้อง แม่นยำ ในการตรวจสอบความถูกต้องของยูสเคส	4.25	0.58	ความพึงพอใจมาก

5.5.1 สรุปผลการประเมินความพึงพอใจของระบบ

จากผลการประเมินความพึงพอใจในการใช้งานระบบตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอลจากผู้เข้าร่วมประเมินในด้านต่างๆ สามารถสรุปได้ดังนี้

1) ความพึงพอใจในด้านการทำงานของระบบ

ความพึงพอใจของผู้เข้าร่วมประเมินในด้านความสามารถในการป้องกันความผิดพลาดจากผู้ใช้อยู่ในระดับพึงพอใจมาก ส่วนด้านอื่นๆ นอกจากนั้นอยู่ในระดับพึงพอใจมากที่สุด

2) ความพึงพอใจในด้านการติดต่อผู้ใช้งาน

ความพึงพอใจของผู้เข้าร่วมประเมินในด้านการติดต่อกับผู้ใช้งานอยู่ในระดับพึงพอใจมากในทุกๆ ด้าน

3) ความพึงพอใจในด้านคุณค่าของระบบ

ความพึงพอใจของผู้เข้าร่วมประเมินในด้านคุณค่าของระบบอยู่ในระดับพึงพอใจมากในทุกๆ ด้าน

5.5.2 ข้อเสนอแนะ

ข้อเสนอแนะที่ได้จากการประเมินความพึงพอใจของผู้ใช้ระบบตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล มีดังนี้

1) ปรับปรุงการแสดงผลในส่วนติดต่อผู้ใช้ให้สอดคล้องตามวัตถุประสงค์และให้สะดวกต่อการใช้งานมากขึ้น

2) การที่ระบบสร้างตารางตัดสินใจโดยไม่ได้คำนึงถึงลำดับของข้อมูลหรือความเป็นไปได้จริงของสถานการณ์ ทำให้จำนวนของกรณีทดสอบมีมากเกินความจำเป็น

5.6 การดำเนินการหลังการประเมินระบบ

ผู้วิจัยได้พิจารณาทบทวนการสร้างตารางตัดสินใจตามข้อเสนอแนะของผู้เข้าร่วมประเมินให้เหมาะสมยิ่งขึ้นแล้ว และได้พิจารณาปรับปรุงส่วนแสดงผลให้ใช้งานง่ายขึ้น

บทที่ 6

บทสรุปและข้อเสนอแนะ

จากเนื้อหางานวิจัยทั้งหมดที่นำเสนอในรายงานนี้ ได้กล่าวถึงความเป็นมาของงานวิจัยและศึกษาแนวคิดที่เกี่ยวข้อง นำแนวคิดที่ได้มาวิเคราะห์และออกแบบแนวทางการแก้ปัญหา จนถึงการพัฒนาระบบพร้อมทั้งทดสอบการใช้งาน ดังนั้นสำหรับในบทนี้จะกล่าวถึงผลสรุปในการทำวิจัย ปัญหาและอุปสรรคในการวิจัย รวมถึงข้อเสนอแนะสำหรับผู้ที่สนใจในอนาคต

6.1 สรุปผลการวิจัย

งานวิจัยนี้นำเสนอวิธีการตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล ขั้นตอนการตรวจสอบเริ่มจากการรับข้อมูลคุณลักษณะความต้องการในรูปแบบฟอร์ม คำอธิบายยูสเคสในรูปแบบฟอร์ม เพื่อนำมาสร้างตารางตัดสินใจ จากนั้น นำตารางตัดสินใจมาเปรียบเทียบกันเพื่อตรวจสอบว่ามีเหตุการณ์ใดในตารางตัดสินใจที่ยูสเคสไม่สอดคล้องกับคุณลักษณะความต้องการหรือไม่ ซึ่งจะระบุได้ว่าในสถานการณ์ใดมีข้อผิดพลาดเกิดขึ้น และข้อมูลนำเข้าตัวใดที่มีข้อผิดพลาดนั้น การทดสอบประสิทธิภาพของขั้นตอนวิธีที่นำเสนอได้ทดสอบใน 3 ประเด็น คือ 1) ความถูกต้อง โดยประเมินจากประสิทธิภาพในการตรวจสอบความถูกต้องของยูสเคส 2) ความครบถ้วน โดยประเมินจากประสิทธิภาพในการสร้างตารางตัดสินใจ และ 3) เวลา โดยประเมินจากเวลาที่ผู้เข้าร่วมประเมินใช้ในการดำเนินการ โดยพบว่าวิธีการที่นำเสนอสามารถตรวจจับความถูกต้องและความครบถ้วนสมบูรณ์มากกว่าการดำเนินการด้วยนักวิเคราะห์ระบบ โดยเฉพาะนักวิเคราะห์ระบบที่มีประสบการณ์น้อย อีกทั้งความรวดเร็วในการประมวลผลมากกว่าการดำเนินการด้วยคน

นอกจากนั้นงานวิจัยนี้ยังได้พัฒนาเครื่องมือซอฟต์แวร์สำหรับสนับสนุนในการตรวจสอบความถูกต้องของยูสเคสเพื่ออำนวยความสะดวกแก่นักวิเคราะห์ โดยได้ประเมินความพึงพอใจของระบบจากผู้เชี่ยวชาญที่เข้าร่วมทดลองประเมินระบบ พบว่ามีความพึงพอใจในระดับพึงพอใจมาก

6.2 ปัญหาและอุปสรรค

เนื่องจากกรณีตัวอย่างที่นำมาใช้ในการประเมินระบบนั้น ทางผู้พัฒนาขึ้นใช้งานจริงไม่ได้พัฒนาด้วยแนวทางการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล ดังนั้นผู้วิจัยจึงจำเป็นต้องสร้างคำอธิบายยูสเคสเองโดยใช้คุณลักษณะความต้องการของกรณีตัวอย่างเป็นต้นแบบ ทำให้ข้อมูลของกรณีตัวอย่างศึกษาในส่วนที่เป็นคำอธิบายยูสเคสไม่ใช่ข้อมูลที่ได้มาจากระบบที่ใช้งานจริง

6.3 ข้อเสนอแนะและงานในอนาคต

ข้อเสนอแนะและงานในอนาคตสำหรับงานวิจัยมีดังนี้

1) เครื่องมือที่พัฒนาขึ้นสนับสนุนการตรวจสอบความถูกต้องของยูสเคสยังมีข้อจำกัดในการใช้งานอยู่ และทำงานแบบอัตโนมัติได้เฉพาะในส่วนการสร้างตารางตัดสินใจและการตรวจสอบความถูกต้อง แต่ในส่วนการรับข้อมูลเข้ามาในระบบยังไม่ได้เป็นแบบอัตโนมัติ วิธีการสกัดความต้องการจากข้อกำหนดคุณลักษณะความต้องการของผู้ใช้เป็นงานที่ยาก ซึ่งอาจต้องใช้เทคนิคปัญญาประดิษฐ์ เช่นเครือข่ายประสาทเทียมมาช่วยได้ เพื่อลดการนำเข้าข้อมูลโดยนักวิเคราะห์

2) เครื่องมือที่พัฒนาขึ้นสนับสนุนการตรวจสอบความถูกต้องของยูสเคสมีส่วนแสดงผลและลำดับการทำงานที่ยังไม่เป็นมิตรกับผู้ใช้เท่าที่ควร ดังนั้นในอนาคตควรพัฒนาการออกแบบให้ดีขึ้น

3) เครื่องมือที่พัฒนาขึ้นสนับสนุนการตรวจสอบความถูกต้องของยูสเคสมีการสร้างกรณีทดสอบของทุกเหตุการณ์ที่เป็นไปได้ ซึ่งอาจมากเกินความจำเป็น ดังนั้นในอนาคตควรพัฒนากระบวนการขั้นตอนวิธีในการสร้างกรณีทดสอบให้เหมาะสมต่อการใช้งาน

บรรณานุกรม

บรรจง หะรังสี และ ญาณวรรณ สินธุภิญโญ. 2542. แนะนำ UML เบื้องต้น. NECTEC Technical Journal. 2542, 1(5). pp. 184-198.

สิริมา นาคเตี้ย. 2557. ระบบสนับสนุนสำหรับการสร้างกรณีทดสอบเบื้องต้นบนพื้นฐานของคุณลักษณะความต้องการที่เขียนอยู่ในรูปแบบฟอร์ม. วิทยานิพนธ์มหาบัณฑิต (วิทยาการคอมพิวเตอร์). มหาวิทยาลัยสงขลานครินทร์, สงขลา, ประเทศไทย.

Ammann, P., and Offutt, J. 2008. Introduction to Software Testing. Cambridge University Press: USA.

Dennis, A., Wixom, B. H., and Tegardenm, D. 2015. System Analysis & Design: An Object - Oriented Approach with UML 5th Edition. John Wiley & Sons, Inc.: USA.

Glenford, J. M., Badgett, T., and Sandler, C. 2012. The Art of Software Testing 3rd Edition. John Wiley & Sons, Inc.: USA.

IEEE (The Institute of Electrical and Electronics Engineers). 1998. IEEE Recommended Practice for Software Requirements Specifications. IEEE Standard 893. 1998 Edition. IEEE Computer Society

IEEE (The Institute of Electrical and Electronics Engineers). 2006. IEEE Standard for Developing a Software Project Life Cycle Process. IEEE Standard 1074. 2006 Edition. IEEE Computer Society

Liu, S., Sun, J., Liu, Y., and Zhang, Y. 2014. Automatic Early Defects Detection in Use Case Documents. ASE'14. Vasteras, Sweden, 15-19 September. pp. 785-790.

Mahmood, A., and Khatoon, S. 2013. An Approach to Generate Test Goals from Use Case Scenarios. Information Technology Journal. 2013, 12(8). pp.1600-1606.

OMG (Object Management Group). 2017. OMG Unified Modeling Language 2.5.1. OMG Unified Modeling Language Publication.

Ricca, F., Torchiano, M., Penta, M. D., Ceccato, M., and Tonella, P. 2007. Using acceptance tests as a support for clarifying requirements: A series of experiments. Information and Software Technology, 51. pp. 270-283.

Rumbaugh, J., Jacobson, I., and Booch, G. 2005. The Unified Modeling Language Reference Manual 2nd Edition. Pearson Education, Inc.: USA.

Sommerville, I. 2011. Software Engineering 9th Edition. Pearson Education, Inc.: USA.

Tianual, P., and Pohthong, A. 2019. Defects Detection Technique of Use Case Views during Requirements Engineering. 8th International Conference on Software and Computer Applications (ICSCA 2019). Penang, Malaysia, 19-22 February. pp. 277-281.

Törner, F. Ivarsson, M., Pettersson, F., and Öhman, P. 2006. Defects in Automotive Use Cases. ISESE'06. Rio de Janeiro, Brazil, 21–22 September. pp. 115-123.

Xu, D., and He, X. 2007. Generation of Test Requirements from Aspectual Use Cases. Workshop WTAOP. Vancouver, Canada, 12-13 March. pp. 17-22.

ภาคผนวก

ภาคผนวก ก.

ผลงานตีพิมพ์

เรื่อง	Defects Detection Technique of Use Case Views during Requirements Engineering
งานประชุมวิชาการ	2019 8th International Conference on Software and Computer Applications (ICSCA 2019)
สถานที่	ปีนัง ประเทศมาเลเซีย
วันที่	19 – 22 กุมภาพันธ์ 2562

Defects Detection Technique of Use Case Views during Requirements Engineering

Poranat Tianual
Department of Computer Science
Faculty of Science, Prince of Songkla University
Hat Yai, Songkhla, Thailand
poranat.tianual@gmail.com

Amnart Pohthong
Department of Computer Science
Faculty of Science, Prince of Songkla University
Hat Yai, Songkhla, Thailand
amnart.p@psu.ac.th

ABSTRACT
In the past decade, object-oriented software engineering (OOSE) has gained popularity from many software developers, especially OOSE with a unified modeling language (UML). Use case views are often used in most systems during an analysis phase. These views show the system functionality related to the system stakeholders. Hence, use case views seem to be a corner stone for a software system. The defects occurring in use case views will affect the later designs. If these defects can be found early, it would save time and cost in software development. Therefore, this research proposes a technique for detecting defects in use case views during an analysis phase or requirements engineering process. Correct users' requirements were created as requirements specifications in a traditional form-based style for testing comparison. The algorithms for generating decision table from form-based requirements and UML use case specification as well as the algorithm for use case view validation were invented. Two simple case studies were investigated and used as the preliminary evaluation. The nineteen fourth-year students were selected as the subjects for the preliminary investigation in order to compare between manual fault detection and our automated proposed system. They were asked to perform four tasks for each case study. The results show that the efficiency of manual fault detection is less than the proposed technique at 47% and 44% or overall average as 45.5% for the two case studies. Currently, we apply this proposed technique to more complex industrial setting and familiar software systems to software engineers.

CCS Concepts
• **Software and its engineering → Software verification and validation → Process validation → Use cases**

Keywords
Defects detection; Requirements engineering; Requirements validation; Use case view; UML.

1. INTRODUCTION
Computer hardware and software has been evolved rapidly since computer technologies were introduced [1]. Human needs and users' requirements play a key role in software construction. Errors can be occurred any time in software engineering processes during software development life cycle (SDLC) such as the IEEE-1074 standard for SDLC [2]. These SDLC processes are employed to some software development paradigms such as a traditional waterfall model, a spiral model, a prototyping model, and object-oriented paradigms [2-5]. Each paradigm also employs several techniques and methods to build a software product. Hence, defects detection seems to be hard work and a difficult task for software developers. Late error resolution would lead to cost and time factors in software development, as well as reputation of software producers. Although requirements validation is often mentioned and suggested in several software development paradigms and software project management standards such as CMMI (Capability Maturity Model Integration), it cannot guarantee that a software product will be free of errors. Requirements validation seems to be complicated activities since it may involve with all stakeholders along requirements engineering process: requirements elicitation, requirements analysis, requirements specification, requirements negotiation, and requirements management [6]. These requirements specification together with some system models will be used to design more system models in the design phase. It is important to ensure that the system models conform to its user requirements. Defects in software processes and software products should be detected during the early phase of software development in a systematic way. Therefore, efficient techniques and tool supports for defects detection and prevention are still needed for software engineering practice. Especially, software engineering methodologies using object-oriented techniques with UML introduce many views and diagrams for system modeling.

Although some researchers have proposed some techniques and methods for defects detection in UML use case views during the analysis phase [7-10], those techniques are based on formal methods and natural language processing (NLP). These methods seem to be complicated for most analysts and software engineers in software practice, especially this would be obstacles for software maintenance. Thus, defects detection in term of model checking and validation is necessary. The validation support tools for users are also needed.

In this work, we intend to propose a new technique together with its support tool for defect detection of use case views using decision and relation tables. In addition, the decision tables from our methods can be used for generating the user acceptance test cases.

The remainder of this paper is organized as follows. Section 2 discusses some related works. Section 3 described the proposed technique. Section 4 presents evaluation and results. In section 5, we conclude our work.

ICSCA '19, February 19–21, 2019, Penang, Malaysia

ACM ISBN 978-1-4503-6573-4/19/02…$15.00
https://doi.org/10.1145/3316615.3316631

2. RELATED KNOWLEDGE BACKGROUND

2.1 Object-Oriented Software Engineering

In the past decades, several new programming languages and software development techniques have been introduced and integrated to software development paradigms as well as engineering principles are also employed to software development in term of building a safety product with cost-effective solution to practical problems [11]. Object-oriented paradigms and programming languages were also introduced and have gained recognition from many software developers. In design and programming aspects, an object is an instance of abstract data types representing a real world object in a business system. A software system is viewed as a set of interacting objects. Information hiding, encapsulation, and inheritance are some major characteristics of object orientation that are expected to reduce complexity and enhance reusability in software development [3, 12]. Nowadays, there are several object-oriented programming languages such as C++, Java, and Python. OOSE introduces new techniques to traditional software engineering including object-oriented analysis, object-oriented design, object-oriented implementation, and object-oriented testing. At the beginning era of OOSE, there were limited techniques and support tools for software system development. Later UML has been invented for OOSE.

2.2 Unified Modeling Language

Software engineering takes many concepts from engineering methods such as requirements specification, software specification, and architectural design. Similar to engineering construction, system modeling and design is important for building a software system, especially for a complex system. Hence, UML emerges as a modeling language for describing software systems along software engineering processes. It has been integrated most good practices of OOSE, including an object modeling technique (OMT), object-oriented software engineering (OOSE), and object-oriented analysis and design (OOAD) [13-15]. Later, the Object Management Group (OMG) took responsibility for the further development of the UML standard [16]. UML provides three views for modeling a software system: functional, structural, and behavioral views. In order to depict these three views, a conceptual model of UML consists of three kinds of following components.

Things. The UML consists of four kinds of things: structural, behavioral, grouping, and annotation things.

Relationships. The UML consists of four kinds of relationships: dependency, association, generalization, and realization.

Diagrams. The last version of UML provides several diagrams classified into two groups. The first group is the structure diagrams consisting of class, object, package, composite structure, component, deployment, and profile diagrams. The second group is the behavioral diagrams consisting of use case, activity, state machine, sequence, communication, timing, and interaction overview diagrams.

In business needs, users' requirements vary from system to system and depend on the nature of business. Software developers can select suitable UML diagrams for their desired software systems. Hence, OOSE with UML needs good understandings about UML components and entities, well-organized software engineering processes from an analysis phase to design and implementation phases as well as well-organized change management [17]. However, verification and validation for OOSE with UML still needs more practical techniques and tools although some research work has suggested new ideas for model checking and validation as follows: The use of model checkers to verify properties of UML activity diagrams [18], Semantics and verification of data flow in UML 2.0 activities [19], an automatic method for the verification of UML class diagrams with the object constraint language [20], An approach to detect infeasible paths in UML models [21], An approach for testing UML design models to uncover inconsistencies among sequence diagrams and class diagrams [22], Automatic synthesis of communication and concurrency for exploring component-based system implementations considering UML channel semantics [23], Formal verification and validation of UML 2.0 sequence diagrams using source and destination of messages [24], and controlled experiment of manual test case derivation from UML activity diagrams and state machine diagrams [25].

The most of them focus on activity diagrams, sequence diagrams, state machine diagrams, and class diagrams. There are a few of them focus on verification and validation of use case views.

Since our research framework is based on requirements specified by use case description, we have reviewed some practical use case descriptions as following formats [26]; Alistair Cockburn's use case description format [27], Jan Kettenis's use case description format [28], Tao Yue's use case description format [29], and Alan Dennis's use case description format [30].

From above reviews of practical use case description formats, we employ Alan Dennis's format because it seems to be simple and up to date.

3. PROPOSED TECHNIQUE

In order to detect the invalid entity occurring during UML use case creation, we use the users' requirements written in a form-based specification for comparison. When creating use case views for any software system, a developer takes these requirements specifications to account. These requirements specifications represent real user requirements from an analysis phase. Therefore, the use case views that do not conform to the analysis requirements specifications mean some defects happening in this process. Defects in use case models can be categorized in many criteria [31]. However, in our preliminary investigation we focus on two criteria: the incorrect flow between use cases and the goal not achieved. For example, in our simple case study: simple box office, Fig 1(a) shows the correct use case view and Fig 1(b) shows the incorrect use case view that consists of an invalid flow and incomplete use cases.

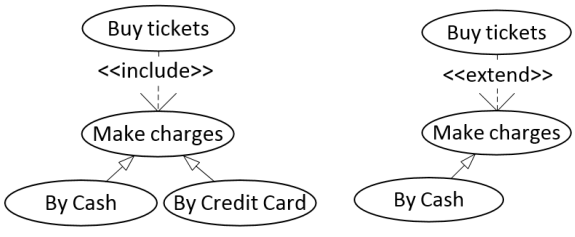


Figure 1. Example of defects in a use case view for the simple box office

In our technique, we start by creating decision tables for both the concerned requirements specification and its use case. Then we validate a use case model by comparing these generated decision tables to find the incompleteness of use case scenarios

corresponding to use case functionality. Then we compare the relations appearing in the specification with the relations of use cases to find the misuse of semantic flows occurring between use cases in the use case model.

This proposed technique consists of five main processes: 1) Get all of functional requirements. In this process, the system analyst specifies a user's requirements in a form-based specification format for each specification [2], the relationship between the requirements specification, the possible expected results of each specification and their validity. Then, the possible value for each input and its expected result are specified. 2) Get use case descriptions. In this process, the use case description and its possible value for each input, together with their validity are specified. 3) Generate decision tables for all requirements specifications and for all use case descriptions; one specification produces one decision table and one use case description produces one decision table. 4) Compare the corresponding decision tables. 5) Compare the corresponding relations.

Our proposed support tool was based on the above five processes. However, each requirements specification must be manually mapped in pair with its corresponding use case description before generating and comparing the decision tables. Giving S is the desired system and F represent all functional requirements specifications in S. Giving T is a set of all use cases in the system, S, consisting its use case elements, $U_1, U_2, \dots, U_b$. The order of corresponding elements of S and T can be written in pair as $(F_1, U_1), (F_2, U_2), \dots, (F_b, U_b)$ for all b functional requirements.

3.1 Decision table generation

In order to generate the decision tables, we begin by constructing a set of each input $Pi_{in} = \{p_1, p_2, \dots, p_{fi}\}$ where Pi_{in} denote the input of F_b, fi is a number of possible value of input Pi_{in}, and p_{fi} is the possible value of the input. And do the same way to use case $U_b$by constructing a set of each input$Pj_{in} = \{p_1, p_2, \dots, p_{ui}\}$ where Pj_{in} denote the input of U_b, ui is a number of possible value of input Pi_{in}, and p_{ui} is the possible value of the input. Then, create a Cartesian product of every Pi_{in} or Pj_{in}, called C set as shown in equation (1), the elements of C are the corresponding scenarios in each decision table.

$$C = Pi_1 \times Pi_2 \times \dots \times Pi_{in} \quad (1)$$

If the validity of every p_{fi} in Pi_{in} is "Valid" then the result of Pi_{in} is "Valid" else the result of Pi_{in} is "Invalid". For example, $Pi_1 = \{1, 2\}$, $Pi_2 = \{a, b\}$, and $Pi_3 = \{s, t, u\}$ where the validity of each value is as follows: 1-Valid, 2-Invalid, a-Valid, b-Invalid, s-Valid, t-Valid, and u-Invalid, the Cartesian product of $Pi_1 \times Pi_2 \times Pi_3$ is as shown in Table 1. Value of Pi_1, Pi_2, and Pi_3 in each row represents the possible value of each input in the scenario.

Table 1. Example of Decision Table

Pi_1	Pi_2	Pi_3	$Validity$
1	a	s	$Valid$
1	a	t	$Valid$
1	a	u	$Invalid$
...	...	...	...
2	b	u	$Invalid$

3.2 Decision table comparison

In order to compare the decision tables, we begin by constructing a set of each F_b and U_b:

3.2.1 Possible event generation for requirements specification

From each $F_b \epsilon S$, constructing a set of the function F_b, the elements of F_b are $R_1, R_2, \dots, R_m$ where R_m denote expected results of each specification and m is a number of expected results. The elements of R_m are $I_1, I_2, \dots, I_q$ where I_q denote the input, q is a number of inputs, and $I_q \epsilon Pi_{in}$. The elements of I_q are $i_1, i_2, \dots, i_x$ where i_x denote the possible value of input I_q in the expected result R_m from function F_n and x represent a number of possible value as shown in equations (2) – (4).

$$F_b = \{R_1, R_2, \dots, R_m\} \quad (2)$$

$$R_m = \{I_1, I_2, \dots, I_q\} \quad (3)$$

$$I_q = \{i_1, i_2, \dots, i_x\} \quad (4)$$

3.2.2 Possible event generation for use case description

From each $U_b \epsilon T$, constructing a set of the use case U_b, the elements of U_b are $P_1, P_2, \dots, P_m$ where P_m denote path of use case and m is a number of paths. The elements of P_m are $J_1, J_2, \dots, J_q$ where J_q denote the input, q is a number of inputs, and $J_q \epsilon Pj_{in}$. The elements of J_q are $j_1, j_2, \dots, j_y$where j_y denote the possible values of input I_m in the path P_m from use caseU_n and y represent a number of possible values as shown in equations (5) – (7).

$$U_b = \{P_1, P_2, \dots, P_m\} \quad (5)$$

$$P_m = \{J_1, J_2, \dots, J_q\} \quad (6)$$

$$J_q = \{j_1, j_2, \dots, j_y\} \quad (7)$$

Finally, the corresponding decision tables are compared by using the intersection set of input I_q and $J_q$in every expected result R_m in every function F_b as shown in equation (8).

$$\sum_{d=1}^{b} \sum_{e=1}^{m} \sum_{f=1}^{q} I_{d,e,f} \cap J_{d,e,f} \quad (8)$$

If $I_f \cap J_f = I_f$ then the path $P_e$in the use case U_d covers the expected result R_e in the specification F_d else, the use case U_d has the defects in the input J_f of path P_e. For example, the specification F_1 has the expected results as shown in Table 2(a) and the use case U_1has the paths as shown in Table 2(b). All expected results and paths in the intersection set of I_f and $J_f$are equals to I_f. This shows that all paths in the use case U_1 cover all expected results in the requirements specification F_1 and mean no defects in the use case functionality.

Table 2. Example of Decision Table comparing

Expected Result		R_1	R_2	R_3	R_4	R_5
Input	I_1	1	1	1	1	2
	I_2	a	a	a, b	a, b	a, b
	I_3	s	t	u	t, u	s, t, u

2(a)

Path		P_1	P_2	P_3	P_4	P_5
Input	J_1	1	1	1	1	2
	J_2	a	a	a, b	a, b	a, b
	J_3	s	t	u	t, u	s, t, u

2(b)

3.3 Relation table comparison

There are three main relations for the semantic flows between use cases: “include”, “extend”, and “generalization”. In general, there is no relationship explicitly specified in form-based functional requirements specification, so in our research we need the system analyst to specify them as “require”, “additional”, and “choose one” respectively. We created the relation tables for functional requirements from *S* to categorize into the main specifications, the dependent specifications, and their relations as $\{f_i, r_j, f_k\}$ where f_i represent the main specifications, f_k represent the dependent specifications, and r_j represent the relations between the specifications. Then, we do the same way for the use case descriptions from T as $\{u_i, q_j, u_k\}$ where u_i represent the base use cases, u_k represent the included use cases or the extending use cases or the specialize use cases, and q_j represent the relations between the use cases. Finally, we compare each pair of specification and use case descriptions in the relation tables $\{f_i, u_i\}$ by the following rules: if r_j is “Require” q_j must be “Include”, if r_j is “Additional” q_j must be “Extended”, and if $r_j$is “Choose one” q_j must be “Generalization”. In the other hand, if any relationship does not follow these rules, we assume that the relation has the defects. For example, there is a relation table for specification and use case as shown in Table 3, Table 3(a) is a relation table of specification and Table 3(b) is a relation table of use case. A pair of relation in Table 3(a) and Table 3(b) does not follow the rules, which is $\{\text{Func3}, \text{Require}, \text{Func5}\}$ and $\{\text{Case3}, \text{Extend}, \text{Case5}\}$. Hence, the relation of this specification and use case description has defects.

Table 3. Example of Relation Table comparing

Specification	Relation	Dependence specification
Func1	Require	Func2
Func2	Additional	Func3
Func2	Additional	Func4
Func3	Require	Func5

3(a)

Use case	Relation	Dependence use case
Case1	Include	Case2
Case2	Extended	Case3
Case2	Extended	Case4
Case3	Extended	Case5

3(b)

4. EVALUATION OF THE STUDY

4.1 Evaluation of preliminary case studies

The empirical evaluation of preliminary case studies starts with a ten-minute brief of form-based specification and use case model explanation in order to remind and reduce the gap of the subjects’ knowledge background. Then the four tasks of each case study were given to them for specifying whether the given use case model correct or incorrect. If the model is incorrect, they must describe the defects.

4.1.1 Evaluation setting

In our preliminary investigation, we used two simple case studies to evaluate the performance of our proposed technique. There are case study 1 - a very simple box office and case study 2 - a simple box office. The Case study 1 consists of two functional requirements, including "Buy Tickets" and "Make Charge" while the case study 2 consists of four functional requirements, including "Buy Tickets", "Make Charge", "By Cash", and "By Credit Card". Both of these two use case models were modified from the example in UML manual [13]. In order to design an empirical evaluation for the preliminary investigation, we intentionally generated three defected use case models for each case study from the correct use case model, called Task 1-4. The defects in the incorrect use case models were created by switching the direction of the relationship and changing the relationship stereotype from “include” to “extend”.

Our participants acting as the analyst novice subjects are 19 senior students in the major of Computer Science, Prince of Songkla University with GPA greater or equal to 2.5. There are 8 males and 11 females with the average age 21.42 years old. The participants have already taken classes in the course of system analysis and design and the course of software engineering.

4.1.2 Results

The results of our preliminary evaluation show that most of participants are confused with the use of “include” and “extend” relationship stereotypes, and the notations of the relationship in use case diagrams. Thus, they found difficulty in detecting the defects in the two simple case studies as shown in the following charts in Figure 2 and 3 where the y-axis denotes a number of participants and x-axis represents the given tasks. The accuracy rates of participants performing the tasks are shown in Table 4.

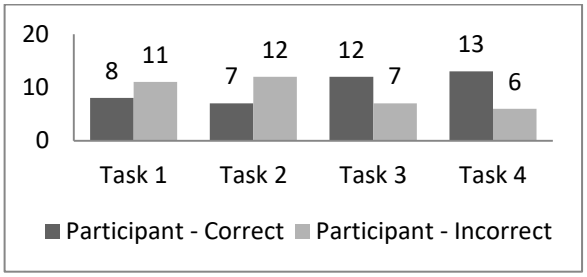


Figure 2.The results performed by the subjects for the case study 1.

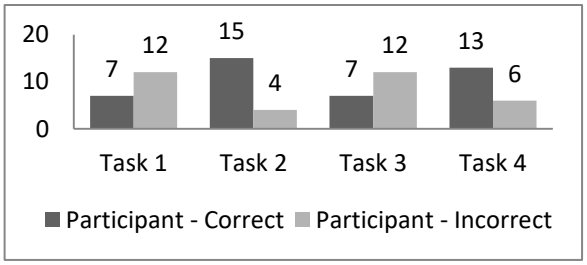


Figure 3. The results performed by the subjects for the case study 2

Table 4. Accuracy comparison

Case study	Accuracy Performance Rate				
	Task 1	Task 2	Task 3	Task 4	Average
Case study 1	42%	39%	63%	68%	53%
Case study 2	39%	79%	39%	68%	56%

4.2 Evaluation of the industrial setting case study

From the results of our preliminary investigation, we currently design the further evaluation study for the proposed technique in

more complex and familiar industrial setting system to analysts and software engineers in terms of correctness and completeness.

5. CONCLUSION

This research proposes a technique for detecting defects in use case views during use case creation. The proposed technique is based on a model-based validation using form-based specification and UML use case description. The nineteen student subjects were selected to act as the analyst novice for the preliminary evaluation in terms of correctness and time consumption. In addition to our findings of defects occurring in the later designs caused by the defects from use case views, the results from the preliminary study show that the accuracy of manually defects detection is much less than our proposed methods with much more time consumption.

6. REFERENCES

[1] G.B. Shelly and M.E. Vermaat. 2011. Discovering Computers Fundamentals: Living in a Digital World. Course Technology Cengage Learning.

[2] J.F. Peters and W. Pedrycz. 2000. Software Engineering: An Engineering Approach. John Wiley & Sons, Inc.

[3] Sommerville. 2011. Software Engineering. Pearson.

[4] W. Pree. 1995. Design Pattern for Object-Oriented Software Development. Addison-Wesley.

[5] N. Dale, D.T. Joyce, and C. Weems. 2006. Object-Oriented Data Structures Using Java. Jones and Bartlett Publishers.

[6] S. Maalem and N. Zarour. 2016. Challenge of validation in requirements engineering, Journal of innovation in digital ecosystem 3, 15-21.

[7] D. Xu and X. He. 2007. Generation of Test Requirements from Aspectual Use Cases. Proceedings of the 3rd workshop on Testing aspect-oriented programs, 17-22.

[8] A. Mahmood and S. Khatoon. 2013. An Approach to Generate Test Goals from Use Case Scenarios, Information Technology Journal 12(8), 1600-1606.

[9] S. Liu, J. Sun, Y. Liu, Y. Zhang, B. Wadhwa, J. Song Dong, and X. Wang. 2014. Automatic Early Defects Detection in Use Case Documents, ASE '14 Proceedings of the 29th ACM/IEEE international conference on Automated software engineering, 785-790.

[10] R.S. Pressman. 1992. Software Engineering: A Practitioner's Approach. McGraw-Hill.

[11] M. Shaw and D. Garlan. 1996. Software Architecture: Perspectives on An Emerging Discipline. Prentice Hall.

[12] E. Braude. 2004. Software Design from Programming to Architecture. John Wiley & Sons, Inc.

[13] J. Rumbaugh, I. Jacobson, and G. Booch. 2004. The Unified Modeling Language Reference Manual 2nd Edition, Addison-Wesley, USA.

[14] B. Bruegge and A.H. Dutoit. 2014. Object-Oriented Software Engineering Using UML. Patterns, and Java. Pearson.

[15] D. Tegarden, A. Dennis, and B.H. Wixom. 2013. Systems Analysis and Design with UML. John Wiley & Sons, Inc.

[16] Object Management Group. 2012. Information technology - Object Management Group Unified Modeling Language (OMG UML) Superstructure.

[17] J. Tomyim and A. Pohthong. 2016. Requirements Change Management Based on Object-Oriented Software Engineering with Unified Modeling Language. Proceedings of 2016 IEEE International Conference on Software Engineering and Service Science, Beijing, China, 2016, 7-10.

[18] Z. Daw and R. Cleveland. 2015. Comparing model checkers for timed UML activity diagrams. Science of Computer Programming, 277-299.

[19] H. Storrle. 2005. Semantics and Verification of Data Flow in UML 2.0 Activities. Electronic Notes in Theoretical Computer Science, 35-52.

[20] J. Cabot, R. Clariso, and D. Riera. 2014. On the verification of UML/OCL class diagram using constraint programming. The Journal of Systems and Software 2014, 1-23.

[21] D. Kundu, M. Sarma, and D. Samanta. 2015. A UML model-based approach to detect infeasible paths. The Journal of Systems and Software 2015, 71-92.

[22] O. Pilskalns, A. Andrews, A. Knight, S. Ghosh, and R. France. 2007. Testing UML designs, Information and Software Technology 2007, 892-912.

[23] H. Posadas, P. Penil, A. Nicolas, and E. Villar. 2015. Automatic synthesis of communication and concurrency for exploring component-based system implementations considering UML channel semantics. Journal of Systems Architecture 2015, 341-360.

[24] V. Lima, C. Talhi, D. Mouheb, M. Debbabi, and L. Wang. 2009. Formal Verification and Validation of UML 2.0 Sequence Diagrams using Source and Destination of Messages. Electronic Notes in Theoretical Computer Science, 143-160.

[25] M. Felderer and A. Herrmann. 2015. Manual test case derivation from UML activity diagrams and state machines: A controlled experiment. Information and Software Technology, 1-15.

[26] S. Tiwari, and A. Gupta. 2017. Investigating comprehension and learnability aspects of use cases for software specification problems. Information and Software Technology, 22-43.

[27] A. Cockburn. 2001. Writing effective use cases. vol. 1. Addison-Wesley, Boston. 2001.

[28] J. Kettenis. 2007. Getting started with use case modeling: White paper, Oracle Corporation.

[29] T. Yue , L.C. Briand ,and Y. Labiche. 2013. Facilitating the transition from Use Case Models to Analysis Models: Approach and Experiments. ACM Trans. Software Eng. Method.

[30] A. Dennis, B.H. Wixom, and D. Tegarden. 2015. System Analysis & Design: An Object-Oriented Approach with UML 5th Edition. John Wiley & Sons.

[31] F. Törner, M. Ivarsson, F. Pettersson, and P. Öhman. 2006. Defects in Automotive Use Cases. Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, 115-123

ภาคผนวก ข.

พจนานุกรมข้อมูล

พจนานุกรมข้อมูล (Data Dictionary) ในระดับรายละเอียดดังตาราง ข-1 ถึง ข-13

สัญลักษณ์ที่ใช้ในการกำหนดชนิดของข้อมูล ได้แก่

int แทน ข้อมูลจำนวนตัวเลข
nvarchar แทน ข้อมูลอักขระ

สัญลักษณ์ที่ใช้แทนคีย์ ได้แก่

PK แทน คีย์หลัก (Primary Key)
FK แทน คีย์นอก (Foreign Key)

ตารางที่ ข-1 ข้อมูล ReqSpec (ข้อมูลความต้องการ)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Spec_ID	nvarchar(50)	รหัสความต้องการ	PK
2	Spec_Name	nvarchar(50)	ชื่อความต้องการ	
3	Source	nvarchar(MAX)	แหล่งนำเข้าข้อมูล	
4	Destination	nvarchar(MAX)	ปลายทางส่งออกข้อมูล	
5	Require	nvarchar(MAX)	สิ่งที่ต้องการ	
6	PreCon	nvarchar(MAX)	เงื่อนไขก่อนดำเนินการ	
7	PostCon	nvarchar(MAX)	เงื่อนไขหลังดำเนินการ	
8	Description	nvarchar(MAX)	คำอธิบายความต้องการ	

ตารางที่ ข-2 ข้อมูล Spec_Input (ข้อมูลรายละเอียดข้อมูลนำเข้าของความต้องการ)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Spec_ID	nvarchar(50)	รหัสความต้องการ	FK (ReqSpec)
2	Variable_ID	nvarchar(50)	รหัสข้อมูลนำเข้า	PK
3	Variable_Name	nvarchar(50)	ชื่อข้อมูลนำเข้า	
4	Variable_Possible	nvarchar(MAX)	ค่าที่เป็นไปได้ทั้งหมด	
5	Variable_Type	nvarchar(10)	ประเภทของข้อมูลนำเข้า	

ตารางที่ ข-3 ข้อมูล Spec_Possible (ข้อมูลค่าที่เป็นไปได้ของความต้องการ)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Possible_ID	nvarchar(50)	รหัสค่าที่เป็นไปได้	PK
2	Spec_ID	nvarchar(50)	รหัสความต้องการ	FK (ReqSpec)
3	Variable_ID	nvarchar(50)	รหัสข้อมูลนำเข้า	FK (Spec_Input)
4	Value	nvarchar(MAX)	ค่าที่เป็นไปได้	
5	Validity	nvarchar(10)	ค่าความถูกต้อง	
6	Result	nvarchar(MAX)	ผลลัพธ์ที่คาดหวัง	

ตารางที่ ข-4 ข้อมูล Spec_Relation (ข้อมูลความสัมพันธ์ของความต้องการ)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Spec_ID	nvarchar(50)	รหัสความต้องการ	FK (ReqSpec)
2	Sub_ID	nvarchar(50)	รหัสความต้องการที่มีความสัมพันธ์ด้วย	FK (ReqSpec)
3	Relation	nvarchar(50)	ประเภทของความสัมพันธ์	

ตารางที่ ข-5 ข้อมูล Spec_Result (ข้อมูลผลลัพธ์ที่คาดหวังของความต้องการ)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Spec_ID	nvarchar(50)	รหัสความต้องการ	FK (ReqSpec)
2	Result_ID	nvarchar(50)	รหัสผลลัพธ์ที่คาดหวัง	PK
3	Value	nvarchar(MAX)	ผลลัพธ์ที่คาดหวัง	
4	Validity	nvarchar(10)	ค่าความถูกต้อง	

ตารางที่ ข-6 ข้อมูล Spec_Decision (ข้อมูลตารางตัดสินใจของความต้องการ)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Spec_ID	nvarchar(50)	รหัสความต้องการ	FK (ReqSpec)
2	RowNo	int	ตำแหน่งตามแถว	
3	ColNo	int	ตำแหน่งตามสดมภ์	
4	CellValue	nvarchar(MAX)	ค่าในตาราง	

ตารางที่ ข-7 ข้อมูล Usecase (ข้อมูลยูสเคส)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Usecase_ID	nvarchar(50)	รหัสยูสเคส	PK
2	Usecase_Name	nvarchar(50)	ชื่อยูสเคส	
3	Actor	nvarchar(50)	ผู้กระทำ	
4	Trigger	nvarchar(MAX)	สิ่งที่เรียกให้เกิดยูสเคส	
5	Stakeholder	nvarchar(MAX)	ผู้เกี่ยวข้อง	
6	Description	nvarchar(MAX)	คำอธิบาย	

ตารางที่ ข-8 ข้อมูล Main_Path (ข้อมูลเส้นทางหลัก)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Usecase_ID	nvarchar(50)	รหัสยูสเคส	FK (Usecase)
2	Step	nvarchar(50)	ขั้นตอนการทำงาน	
3	Activity	nvarchar(MAX)	การทำงาน	

ตารางที่ ข-9 ข้อมูล Extension_Path (ข้อมูลเส้นทางเลือก)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Usecase_ID	nvarchar(50)	รหัสยูสเคส	FK (Usecase)
2	Step	nvarchar(50)	ขั้นตอนการทำงาน	
3	Activity	nvarchar(MAX)	การทำงาน	
4	Reference_Step	nvarchar(50)	ขั้นตอนการทำงานที่แยกออกมา	
5	Number	nvarchar(50)	ลำดับของเส้นทางเลือก	
6	Type	nvarchar(10)	ประเภทของเส้นทางเลือก	
7	Validity	nvarchar(20)	ค่าความถูกต้อง	

ตารางที่ ข-10 ข้อมูล Usecase_Input (ข้อมูลรายละเอียดข้อมูลนำเข้าของยูสเคส)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Usecase_ID	nvarchar(50)	รหัสยูสเคส	FK (Usecase)
2	Variable_ID	nvarchar(50)	รหัสข้อมูลนำเข้า	PK
3	Variable_Name	nvarchar(50)	ชื่อข้อมูลนำเข้า	
4	Variable_Possible	nvarchar(MAX)	ค่าที่เป็นไปได้ทั้งหมด	
5	Variable_Type	nvarchar(10)	ประเภทของข้อมูลนำเข้า	

ตารางที่ ข-11 ข้อมูล Usecase_Possible (ข้อมูลค่าที่เป็นไปได้ของยูสเคส)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Possible_ID	nvarchar(50)	รหัสค่าที่เป็นไปได้	PK
2	Usecase_ID	nvarchar(50)	รหัสยูสเคส	FK (Usecase)
3	Variable_ID	nvarchar(50)	รหัสข้อมูลนำเข้า	FK (Usecase_Input)
4	Value	nvarchar(MAX)	ค่าที่เป็นไปได้	
5	Validity	nvarchar(10)	ค่าความถูกต้อง	
6	Case_Path	nvarchar(50)	เส้นทางที่สัมพันธ์กัน	

ตารางที่ ข-12 ข้อมูล Usecase_Relation (ข้อมูลความสัมพันธ์ของยูสเคส)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Super_ID	nvarchar(50)	รหัสยูสเคส	FK (Usecase)
2	Sub_ID	nvarchar(50)	รหัสยูสเคสที่มีความสัมพันธ์ด้วย	FK (Usecase)
3	Relation	nvarchar(50)	ประเภทของความสัมพันธ์	

ตารางที่ ข-13 ข้อมูล Usecase_Decision (ข้อมูลตารางตัดสินใจของยูสเคส)

ลำดับ	ชื่อเขตข้อมูล	ชนิด	คำอธิบาย	คีย์
1	Usecase_ID	nvarchar(50)	รหัสยูสเคส	FK (Usecase)
2	RowNo	int	ตำแหน่งตามแถว	
3	ColNo	int	ตำแหน่งตามสดมภ์	
4	CellValue	nvarchar(MAX)	ค่าในตาราง	

ภาคผนวก ค.

แบบฟอร์มการประเมินระบบ

<u>คำชี้แจง</u>

งานทดลองวิจัยเรื่อง: การตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล

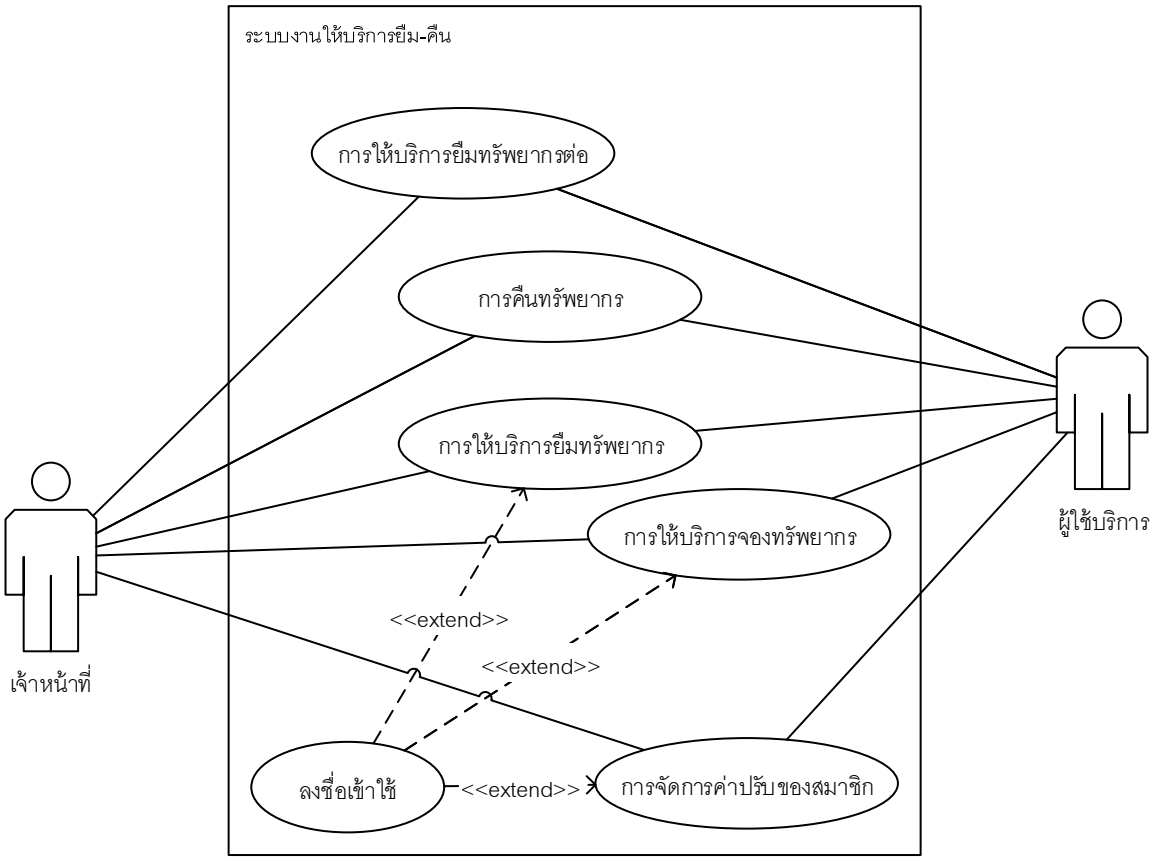
ผู้ดำเนินการวิจัย: นายปรณัฐ เตี้ยนวล

ที่มาและความสำคัญ: ปัจจุบัน การพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอลได้รับความนิยมมาก โดยเฉพาะแผนภาพยูสเคสมักถูกใช้ในการวิเคราะห์ฟังก์ชันของระบบซอฟต์แวร์ที่จะสร้างขึ้น ดังนั้น แผนภาพยูสเคสจึงเป็นสารสนเทศที่สำคัญซึ่งส่งต่อไปยังการวิเคราะห์และออกแบบในกระบวนการดำเนินงานด้วยยูเอ็มแอลในขั้นตอนถัดไป เช่น แผนภาพกิจกรรม (Activity Diagram) แผนภาพลำดับ (Sequence Diagram) และแผนภาพคลาส (Class Diagram) ด้วยเหตุนี้ หากยูสเคสนี้ผิดก็จะส่งผลความผิดพลาดตามไปด้วย ทั้งนี้ ในกระบวนการวิศวกรรมความต้องการสามารถออกแบบกรณีทดสอบได้ตั้งแต่สร้างคำอธิบายยูสเคส

กรอบแนวคิดการวิจัย: นำกรณีทดสอบที่ถูกออกแบบตั้งแต่เริ่มสร้างคำอธิบายยูสเคสมาใช้ในการทดสอบความถูกต้องของยูสเคส เทียบกับ กรณีทดสอบที่ถูกสร้างตามประเพณีนิยมในรูปแบบฟอร์ม

กรณีศึกษา: ระบบให้บริการยืม-คืน (Circulation System) ซึ่งเป็นส่วนหนึ่งของระบบห้องสมุดอัตโนมัติ (Automatic Library System for Thai Higher Education Industries – ALIST) ได้คัดเลือกฟังก์ชันการทำงานหลัก 5 ฟังก์ชันมาทดลองประเมินแนวคิดวิจัยที่นำเสนอ ได้แก่ การให้บริการยืมทรัพยากร (Check-out Item) การคืนทรัพยากร (Check-in Item) การให้บริการยืมทรัพยากรต่อ (Renew Item) การให้บริการจองทรัพยากร (Hold Item) และการจัดการค่าปรับของสมาชิก (Fine) ดังแสดงในคุณลักษณะความต้องการในรูปแบบฟอร์มต่อไปนี้

แผนภาพยูสเคสของระบบงานให้บริการยืม-คืน



ภาพประกอบ 1 แผนภาพยูสเคสของระบบ

<u>หมายเหตุ</u> ผู้ใช้บริการที่เป็นสมาชิกจะต้องดำเนินการตามระบบสมาชิกที่เชื่อมโยงมายังระบบงานให้บริการยืม-คืน

<u>ฟังก์ชันงานชื่อ:</u> การให้บริการยืมทรัพยากร (Check-out Item)

ตารางที่ 1.1 ข้อกำหนดคุณลักษณะของฟังก์ชันการให้บริการยืมทรัพยากร

Function	Check out
Description	1. ป้อนหรือสแกนรหัสสมาชิกลงในช่อง “Barcode” 2. กรณีที่มีข้อมูลสมาชิกระบบจะแสดงรายละเอียดของสมาชิกบางส่วนทางซ้ายมือ 3. กรณีที่ไม่มีข้อมูลสมาชิกหรือสมาชิกหมดอายุ ระบบจะแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบด้วย พร้อมทั้งถามว่าจะต่ออายุบัตรสมาชิกใหม่หรือไม่? ด้วย ถ้าไม่ต้องการให้ทำการล้างหน้าจอทั้งหมด 4. ระบบจะแสดงข้อมูลการยืมที่ยังไม่ได้คืน, การจอง, รายการค่าปรับ และข้อความแจ้งเตือนต่างๆ ของสมาชิก 5. กรณีที่สมาชิกมีรายการค้างค่าปรับ, มีรายการให้รับทรัพยากรที่จองหรือมีข้อความสำคัญต่างๆ ให้ระบบแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ พร้อมทั้งให้ข้อความบนแท็ปนั้นเป็นสีแดง 6. จากนั้นป้อนหรือสแกนรหัสบาร์โค้ดของทรัพยากรลงในช่อง “Barcode” 7. กรณีที่มีสมาชิกท่านอื่นจองทรัพยากรนี้อยู่ให้ระบบแสดงข้อความแจ้งเตือนพร้อมข้อมูลบางส่วนของสมาชิกที่จองทรัพยากรนี้ 8. กรณีที่ยืมเกินสิทธิ์ให้ระบบแสดงคำถามยืนยันการยืมเกินสิทธิ์ ถ้าต้องการยืมเกินสิทธิ์และผู้ใช้ไม่มีสิทธิ์ดำเนินการก็ให้แสดงหน้าจอ Login ขึ้นมา 9. กรณีที่ทรัพยากรนี้มีสมาชิกแจ้งหายไว้ ให้ระบบปรับปรุงสถานะของทรัพยากรนี้และหากสมาชิกยังไม่ได้ชำระค่าปรับในส่วนของค่าทรัพยากรให้ทำการยกเว้นค่าปรับของสมาชิกในส่วนของราคาทรัพยากรพร้อมทั้งส่งข้อความแจ้งเตือนทั้งทางข้อความและทางอีเมล์ของสมาชิกด้วย และแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ 10. กรณีที่ไม่ผ่านการตรวจสอบเงื่อนไขต่างๆ ให้แสดงข้อความแจ้งเตือนขึ้นมา 11. กรณีที่สมาชิกที่ยืมกับสมาชิกที่จองทรัพยากรนี้เป็นคนเดียวกันให้ทำการลบข้อความแจ้งเตือนให้มารับทรัพยากร และปรับสถานะเป็น รับทรัพยากรที่จองแล้ว 12. กรณีที่ไม่มีข้อมูลทรัพยากรนี้ในฐานข้อมูลระบบจะต้องแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบและแสดงหน้าจอให้เพิ่มทรัพยากรแบบเร่งด่วน (Item On The Fly) โดยผู้ใช้ต้องป้อนข้อมูลทั้งหมดที่เป็นสีแดง 13. เมื่อผ่านการตรวจสอบทุกอย่างระบบจะดำเนินการยืมทรัพยากรให้แก่สมาชิกพร้อมทั้งแสดงรายการทรัพยากรที่ยืมในส่วนของ “Checked-out”

	เป็นสีแดง 14. กรณีที่ยืมตัวเล่มที่เป็นทรัพยากรสำรอง จะแสดงหน้าจอให้ผู้ใช้ทราบว่าเป็นทรัพยากรสำรองให้ยืมกี่วัน 15. กรณีที่เกิดข้อผิดพลาดต่างๆ ให้ระบบแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ	
Inputs	- รหัสสมาชิก - รหัสทรัพยากร	ค่าที่เป็นไปได้ ตามตารางที่ 1.2
Source	- ตัวอ่านบาร์โค้ด - คีย์บอร์ด	
Outputs	- ผลลัพธ์การยืม ตามตารางที่ 1.2	
Destination	- จอมอนิเตอร์- ฐานข้อมูล	
Requires	- ความเป็นสมาชิก	
Pre-condition	- ปรากฏหน้าจอการยืม	
Post-condition	- ปรากฏหน้าจอการยืมสำเร็จ - ปรากฏข้อความแจ้งเตือนข้อผิดพลาด - บันทึกข้อมูลการยืมลงฐานข้อมูล	

ตารางที่ 1.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Output
รหัสสมาชิก	String	เป็นสมาชิกมีสิทธิ์ยืม	ยืมสำเร็จ
		เป็นสมาชิกexpire	บัตรหมดอายุ
		เป็นสมาชิกค้างเงินเกิน	ยืมไม่สำเร็จและมีข้อความเตือน
		เป็นสมาชิกถูกบล็อก	ยืมไม่สำเร็จและมีข้อความเตือน
		เป็นสมาชิกไม่มีสิทธิ์ยืม	ยืมไม่สำเร็จและมีข้อความเตือน
		ไม่เป็นสมาชิก	ไม่มีข้อมูลสมาชิก
รหัสทรัพยากร	String	AVยืมได้	ยืมสำเร็จ
		AVยืมไม่ได้typeเกินสิทธิ์	ยืมเกินสิทธิ์
		BHถูกจอง	มีสมาชิกอื่นจองทรัพยากรไว้
		LUใช้ในห้องสมุดเท่านั้น	ยืมไม่สำเร็จและมีข้อความเตือน
		LOหาย	มีสมาชิกอื่นแจ้งหายไว้
		เล่มเดียวกัน	ยืมไม่สำเร็จและมีข้อความเตือน
		OnTheFly	ยืมสำเร็จและเพิ่มทรัพยากรแบบเร่งด่วน
		เล่มสำรอง	ยืมสำเร็จและทรัพยากรเป็นเล่มสำรอง
		อื่นๆยืมไม่ได้	ยืมไม่สำเร็จและมีข้อความเตือน

<u>ฟังก์ชันงานชื่อ:</u> การคืนทรัพยากร (Check-in Item)

ตารางที่ 2.1 ข้อกำหนดคุณลักษณะของฟังก์ชันการคืนทรัพยากร

Function	Check in
Description	1. ป้อนหรือสแกนรหัสบาร์โค้ดของทรัพยากรที่ต้องการคืนในช่อง “Item Barcode” 2. กรณีที่ต้องการเปลี่ยนประเภทการคืนทรัพยากร ให้กดปุ่ม “สัญลักษณ์การแก้ไข” ระบบจะแสดงหน้าจอ login ขึ้นมา แล้วจะปรากฏหน้าจอประเภทการคืนเมื่อ login ผ่าน เมื่อเลือกเปลี่ยน Check-in Type เสร็จก็กดปุ่ม “OK” 3. กรณีที่มีค่าปรับจากการคืนทรัพยากรสายให้ระบบแสดงรายการทรัพยากรคืนสาย, จำนวนค่าปรับ และชื่อสมาชิกที่ยืมในส่วนของสีแดง 4. สามารถเลือกรายการสีแดง เพื่อแก้ไข หรือลดหย่อนค่าปรับ 5. กรณีที่ทรัพยากรหรือหมายเลขบรรณานุกรมที่คืนมีสมาชิกจอง ให้ระบบแสดงข้อความเตือนพร้อมทั้งข้อมูลสมาชิกที่จองบางส่วนให้ผู้ใช้ทราบ 6. กรณีที่ทรัพยากรนี้มีสมาชิกแจ้งหายไว้ ให้ระบบปรับปรุงสถานะของทรัพยากรชิ้นนี้และหากสมาชิกยังไม่ได้ชำระค่าปรับในส่วนของค่าทรัพยากรให้ทำการยกเว้นค่าปรับของสมาชิกในส่วนของราคาทรัพยากรพร้อมทั้งส่งข้อความแจ้งเตือนทั้งทางข้อความและทางอีเมล์ของสมาชิกด้วย และแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ 7. รายการทรัพยากรที่คืนทั้งหมดจะแสดงอยู่ในส่วนของแท็บ “Check-In” 8. การแจ้งพบหนังสือที่ได้แจ้งหายไว้ มีขั้นตอนดังนี้ 8.1 กรณีที่พบหนังสือแล้ว/ซื้อหนังสือมาคืน ให้ยิงบาร์โค้ดในหน้าคืนแล้วระบบจะตรวจสอบว่าเป็นหนังสือที่ถูกแจ้งหายไว้หรือไม่ จะมีหน้าต่างแสดงเมื่อเลือกกรณีจัดการทรัพยากรหายแล้ว กดปุ่ม ‘ตกลง’ ระบบจะทำการเปลี่ยนสถานะตัวเล่มเป็น ‘AV’ และตัดหนี้ทรัพยากรหาย การซื้อหนังสือมาคืนจะต้องนำหนังสือเล่มที่ซื้อไปติดบาร์โค้ดเดียวกับที่ได้แจ้งหายไว้ก่อน ระบบจะแสดงผลลัพธ์เหมือนคืนหนังสือปกติ 8.2 กรณีที่ซื้อหนังสือใกล้เคียงมาคืน เนื่องจากหาหนังสือที่แจ้งหายไว้ไม่พบหรือหาซื้อเล่มเดิมไม่ได้ จึงซื้อหนังสือที่ใกล้เคียงมาแทน การทำงานเช่นเดียวกับกรณีแรกแต่สถานะของตัวเล่มจะยังเป็น ‘LO’ เนื่องจากตัวเล่มจริงๆ ยังไม่พบ ส่วนหนังสือที่นำมาคืนนี้เป็นคนละบาร์โค้ดกับเล่มที่ได้แจ้งหายไว้ แต่การแจ้งพบต้องป้อนบาร์โค้ดตัวเล่มที่แจ้งหายไว้ แล้วเลือกเป็นกรณี 8.3 กรณีหาซื้อไม่ได้ต้องการชำระเงิน ในกรณีนี้อาจเป็นนักศึกษาที่แจ้งหายไว้แล้วต้องการจ่ายเงิน เนื่องจากถ้าไม่เคลียร์หนี้สินจากทางห้องสมุดทำให้ขอจบ

	การศึกษาไม่ได้ การทำงานเช่นเดียวกับกรณีที่สอง แต่จะมีการพิมพ์ใบเสร็จรับเงิน	
Inputs	- รหัสทรัพยากร	
Source	- ตัวอ่านบาร์โค้ด - คีย์บอร์ด	ค่าที่เป็นไปได้ ตามตารางที่ 2.2
Outputs	- ผลลัพธ์การคืน ตามตารางที่ 2.2	
Destination	- จอมอนิเตอร์ - ฐานข้อมูล	
Requires	-	
Pre-condition	- ปรากฏหน้าจอการคืน	
Post-condition	- ปรากฏข้อความแสดงผลการคืน - แก้ไขข้อมูลสถานะทรัพยากรในฐานข้อมูล	

ตารางที่ 2.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Output
รหัสทรัพยากร	String	คืนสำเร็จ	แสดงรายการทรัพยากรที่คืน
		คืนแล้วมีคนจอง	แสดงรายการทรัพยากรที่คืน พร้อมข้อมูลสมาชิกที่จอง
		คืนแบบ Book drop	แสดงรายการทรัพยากรที่คืน
		คืนหนังสือที่แจ้งหาย	แสดงรายการทรัพยากรที่แจ้งหายพร้อมปรับปรุงสถานะแจ้งหายเป็น ‘AV’
		ซื้อหนังสือใกล้เคียงมาคืน	แสดงรายการทรัพยากรที่แจ้งหายพร้อมปรับปรุงสถานะแจ้งหายเป็น ‘LO’
		ชำระเงินค่าหนังสือที่แจ้งหาย	แสดงรายการทรัพยากรที่แจ้งหายพร้อมปรับปรุงสถานะแจ้งหายเป็น ‘LO’และพิมพ์ใบเสร็จรับเงิน
		คืนหนังสือที่เป็น OnTheFly	แสดงรายการทรัพยากรที่คืน
		คืนสาย	แสดงรายการทรัพยากรที่คืน พร้อมแจ้งค่าปรับ

<u>ฟังก์ชันงานชื่อ:</u> การให้บริการยืมทรัพยากรต่อ (Renew Item)

ตารางที่ 3.1 ข้อกำหนดคุณลักษณะของฟังก์ชันการให้บริการยืมทรัพยากรต่อ

Function	Renew Item	
Description	1. เลือกรายการที่ต้องการยืมต่อ แล้วกดปุ่ม “Renew Item” ระบบจะแสดงหน้าจอให้เลือกว่ายืมต่อทางใด ได้แก่ ทางเว็บ OPAC และทางเคาน์เตอร์ 2. ระบบแสดงข้อมูลเพื่อให้ผู้ใช้ยืนยันการยืมต่อ เมื่อดำเนินการเสร็จสิ้น จะมีหน้าจอแสดงผลการยืมต่อ โดยแบ่งออกเป็น 2 ส่วนคือส่วนบน (สีน้ำเงิน) คือทรัพยากรที่ยืมต่อสำเร็จ และส่วนล่าง (สีแดง) คือทรัพยากรที่ยืมต่อไม่ได้ พร้อมมีการระบุเหตุผล	
Inputs	- รหัสสมาชิก - รหัสทรัพยากร	ค่าที่เป็นไปได้ ตามตารางที่ 3.2
Source	- คีย์บอร์ด - ตัวอ่านบาร์โค้ด	
Outputs	- ผลลัพธ์การยืมต่อ ตามตารางที่ 3.2	
Destination	- จอมอนิเตอร์ - ฐานข้อมูล	
Requires	- ความเป็นสมาชิก	
Pre-condition	- ปรากฏหน้าจอการยืมต่อ	
Post-condition	- ปรากฏหน้าจอผลการยืมต่อสำเร็จ - ปรากฏหน้าจอผลการยืมต่อไม่สำเร็จ - แก้ไขข้อมูลสถานะทรัพยากรในฐานข้อมูล	

ตารางที่ 3.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Output
รหัสสมาชิก	String	ยืมต่อได้	รายการทรัพยากรที่ยืมต่อสำเร็จ
รหัสทรัพยากร	String	ยืมต่อสำเร็จ	รายการทรัพยากรที่ยืมต่อสำเร็จ
		ยืมต่อไม่สำเร็จ มีคนจอง -	รายการทรัพยากรที่ยืมต่อไม่ได้
		ยืมต่อไม่สำเร็จ วันข้ามไป - ตอนปิดเทอม	รายการทรัพยากรที่ยืมต่อไม่ได้
		ยืมต่อเกินโควต้า	รายการทรัพยากรที่ยืมต่อไม่ได้

<u>ฟังก์ชันงานชื่อ:</u> การให้บริการจองทรัพยากร (Hold Item)

ตารางที่ 4.1 ข้อกำหนดคุณลักษณะของฟังก์ชันการให้บริการจองทรัพยากร

Function	Hold Item	
Description	1. เลือกแท็ป “Hold” จากนั้นเลือก “By Item” เพื่อจองแบบเจาะจงตัวเล่ม ป้อนหรือสแกนรหัสทรัพยากร 2. เลือก “By Bibliographic” เพื่อจองแบบไม่เจาะจงตัวเล่ม ป้อนหรือสแกนรหัสทรัพยากร 3. กรณีที่จองเกินสิทธิ์ ถ้าผู้ใช้ไม่มีสิทธิ์ดำเนินการให้ระบบแสดงหน้าจอ Login ขึ้นมา 4. กรณีที่ไม่ผ่านเงื่อนไขของการตรวจสอบต่างๆ ให้แสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ 5. กรณีที่ต้องการยกเลิกรายการจองให้เลือกรายการที่ต้องการ 6. กดปุ่ม “Cancel Hold Item(s)” จากนั้นระบบจะแสดงข้อความยืนยันการยกเลิกการจองพร้อมทั้งจำนวนรายการที่ต้องการยกเลิก 7. กรณีที่ยืนยันการยกเลิกการจอง ระบบจะยกเลิกรายการจอง พร้อมทั้งกำหนดสถานะการยกเลิกเป็นยกเลิกการจอง และทำการปรับปรุงลำดับการจองทรัพยากรในแต่ละรายการ 8. กรณีที่เกิดข้อผิดพลาดต่างๆ ให้ระบบแสดงข้อความแต้งเตือนให้ผู้ใช้ทราบด้วย	
Inputs	- รหัสสมาชิก - รหัสทรัพยากร	ค่าที่เป็นไปได้ ตามตารางที่ 4.2
Source	- คีย์บอร์ด - ตัวอ่านบาร์โค้ด	
Outputs	- ผลลัพธ์การจอง ตามตารางที่ 4.2	
Destination	- จอมอนิเตอร์ - ฐานข้อมูล	
Requires	- ความเป็นสมาชิก	
Pre-condition	- ปรากฏหน้าจอการจอง	
Post-condition	- ปรากฏหน้าจอผลการจอง - ปรากฏหน้าจอผลการยกเลิกการจอง	- ปรากฏข้อความแสดงข้อผิดพลาด - แก้ไขข้อมูลลำดับการยืมในฐานข้อมูล

ตารางที่ 4.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Output
รหัสสมาชิก	string	เป็นสมาชิกมีสิทธิ์ยืม	แสดงข้อมูลรายการจอง
		เป็นสมาชิกexpire	แสดงข้อความเตือนข้อผิดพลาด
		เป็นสมาชิกค้างเงินเกิน	แสดงข้อความเตือนข้อผิดพลาด
		เป็นสมาชิกถูกblock	แสดงข้อความเตือนข้อผิดพลาด
		เป็นสมาชิกไม่มีสิทธิ์ยืม	แสดงข้อความเตือนข้อผิดพลาด
		ไม่เป็นสมาชิก	แสดงข้อความเตือนข้อผิดพลาด
รหัสทรัพยากร	string	จองเล่มเดียวกัน	แสดงข้อความเตือนข้อผิดพลาด
		AVจองได้	แสดงข้อมูลรายการจอง
		AVจองไม่ได้typeเกินโควต้า	แสดงหน้าจอ login
		LUใช้ในห้องสมุดเท่านั้น	แสดงข้อความเตือนข้อผิดพลาด
		LOหาย	แสดงข้อความเตือนข้อผิดพลาด
		ยกเลิกการจอง	แสดงรายการการยกเลิกการจอง
		อื่นๆจองไม่ได้	แสดงข้อความเตือนข้อผิดพลาด

<u>ฟังก์ชันงานชื่อ:</u> การจัดการค่าปรับของสมาชิก (Fine)

ตารางที่ 5.1 ข้อกำหนดคุณลักษณะของฟังก์ชันการจัดการค่าปรับของสมาชิก

Function	Fine	
Description	1. คลิกแท็ป “Debt” 2. ในช่อง Balance จะต้องแสดงจำนวนค่าปรับทั้งหมดที่มี 3. กรณีที่ต้องการชำระค่าปรับ ให้เลือกรายการที่ต้องการชำระ ในช่อง Pay Debt จะต้องแสดงจำนวนค่าปรับที่จะชำระ หรือ ป้อนจำนวนเงินที่ต้องการชำระในกรณีที่ต้องการชำระเพียงบางส่วน 4. กดปุ่ม “OK” ระบบจะต้องแสดงคำถามยืนยันการชำระพร้อมทั้งแสดงรายการที่ต้องการชำระ 5. กรณีที่ชำระเป็นบางส่วนระบบจะตัดจากรายการที่เลือกตามลำดับ แล้วแสดงยอดเงินคงเหลือในส่วนของ Balance ของแต่ละรายการ 6. เมื่อดำเนินการชำระเรียบร้อยแล้ว ระบบจะต้องแสดงคำถามยืนยันการพิพม์ใบเสร็จรับเงิน 7. กรณีที่ต้องการลดหย่อนค่าปรับ กดปุ่ม “Deduct Debt” ระบบจะแสดงหน้าจอ “Login”เพื่อตรวจสอบสิทธิ์การเข้าใช้งาน จากนั้นให้ป้อนจำนวนเงินที่จะลดหย่อน และป้อนหมายเหตุของการลดหย่อน ซึ่งผู้ใช้จำเป็นต้องป้อนหมายเหตุของการยกเว้นนี้ แล้วกดปุ่ม “OK” 8. กรณีที่ต้องการเพิ่มค่าปรับ ให้กดปุ่ม “Add Debt” 9. ระบบแสดงหน้าจอ login ขึ้นมา เพื่อตรวจสอบสิทธิ์การเข้าใช้งาน 10. ระบบจะแสดงหน้าจอเพิ่มรายการค่าปรับขึ้นมา 11. ในการเพิ่มรายการค่าปรับผู้ใช้จำเป็นที่จะต้องป้อนหมายเหตุของการเพิ่มรายการค่าปรับ 12. จำนวนเงิน ไม่สามารถเป็นค่าว่างหรือมีค่าเป็นศูนย์ได้	
Inputs	- รหัสสมาชิก	ค่าที่เป็นไปได้ ตามตารางที่ 5.2
Source	- ตัวอ่านบาร์โค้ด - คีย์บอร์ด	
Outputs	- ผลลัพธ์การจัดการค่าปรับ ตามตารางที่ 5.2	
Destination	- จอมอนิเตอร์ - เครื่องพิมพ์ - ฐานข้อมูล	

Requires	- ความเป็นสมาชิก
Pre-condition	- ปรากฏหน้าจอการจัดการค่าปรับ
Post-condition	- ปรากฏหน้าจอผลการจัดการค่าปรับ - พิมพ์ใบเสร็จรับเงิน - แก้ไขข้อมูลรายการค่าปรับในฐานข้อมูล

ตารางที่ 5.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Output
รหัสสมาชิก	string	มีค่าปรับ	แสดงรายการค่าปรับ
ตัวเลือกดำเนินการ	string	เพิ่มค่าปรับ	แสดงรายการค่าปรับ
		ลดหย่อนค่าปรับ	แสดงรายการค่าปรับ
		ชำระค่าปรับทั้งหมดของรายการ	ดำเนินการชำระค่าปรับ
		ชำระค่าปรับบางส่วนของรายการ	แสดงยอดค่าปรับคงเหลือในแต่ละรายการ

ภาคผนวก ง.

โจทย์ปัญหาสำหรับการประเมินระบบ

<u>โจทย์สำหรับการทดลองวิจัย 1</u>

<u>คำสั่งดำเนินงาน</u> ขอให้ผู้เข้าทดลองพิจารณาคำอธิบายยูสเคส (Use Case Description) หมายเลข 1.1 ถึง หมายเลข 1.5 ซึ่งต้องมีสารสนเทศครบตามข้อกำหนดคุณลักษณะของฟังก์ชันงานที่อธิบายไว้ตามแบบฟอร์ม และกรุณาระบุข้อผิดพลาดที่พบในคำอธิบายยูสเคสต่อไปนี้

ตารางที่ 6.1 คำอธิบายยูสเคสหมายเลข 1.1

Use Case Name	Check out	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การให้บริการยืมทรัพยากร เป็นกระบวนการเพื่อให้สมาชิกสามารถยืมทรัพยากรสารสนเทศจากห้องสมุดได้ โดยสมาชิกจะต้องนำบัตรสมาชิกและทรัพยากรที่ต้องการยืมมาติดต่อที่จุดให้บริการ	
Trigger	สมาชิกนำบัตรสมาชิกและทรัพยากรที่ต้องการยืมมาติดต่อที่จุดให้บริการ	
Relationships	Association:	-
	Include:	-
	Extend:	ลงชื่อเข้าใช้
	Generalization:	-
Normal flow of events	1. ป้อนหรือสแกนรหัสสมาชิกลงในช่อง “Barcode” หากมีข้อมูลสมาชิก ระบบจะแสดงรายละเอียดของสมาชิกบางส่วนทางซ้ายมือ 2. ระบบจะแสดงข้อมูลการยืมที่ยังไม่ได้คืน, การจอง, รายการค่าปรับ และข้อความแจ้งเตือนต่างๆ ของสมาชิก 3. ป้อนหรือสแกนรหัสบาร์โค้ดของทรัพยากรลงในช่อง “Barcode” 4. ดำเนินการยืมทรัพยากรให้แก่สมาชิกพร้อมทั้งแสดงรายการ	

	ทรัพยากรที่ยืมในส่วนของ “Checked-out” เป็นสีแดง
Sub Flows	S-1: ยืมเกินสิทธิ์ 1. ระบบจะแสดงคำถามยืนยันการยืมเกินสิทธิ์ 2. แสดงหน้าจอ login
Alternate/Exception Flows	1.e1: กรณีที่ไม่มีข้อมูลสมาชิกหรือสมาชิกหมดอายุ ระบบจะแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบด้วย พร้อมทั้งถามว่าจะต่ออายุบัตรสมาชิกใหม่หรือไม่? ด้วย ถ้าไม่ต้องการให้ทำการล้างหน้าจอทั้งหมด 1.e5: กรณีที่เป็นสมาชิกแต่ไม่สามารถยืมได้ เนื่องจาก ถูกบล็อกค้างเงินเกิน หรือไม่มีสิทธิ์ยืม ให้ระบบแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ 3.e1: กรณีที่มีสมาชิกท่านอื่นจองทรัพยากรนี้อยู่ ให้ระบบแสดงข้อความแจ้งเตือนพร้อมข้อมูลบางส่วนของสมาชิกที่จองทรัพยากรนี้ 3.e2: กรณีที่ยืมเกินสิทธิ์ S-1 จะถูกกระทำ 3.e3: กรณีที่ไม่มีข้อมูลทรัพยากรนี้ในฐานข้อมูล ระบบจะต้องแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบและแสดงหน้าจอให้เพิ่มทรัพยากรแบบเร่งด่วน (Item On The Fly) โดยผู้ใช้ต้องป้อนข้อมูลทั้งหมดที่เป็นสีแดง 3.e4: กรณีที่ทรัพยากรนี้มีสมาชิกแจ้งหายไว้ ให้ระบบปรับปรุงสถานะของทรัพยากรนี้และหากสมาชิกยังไม่ได้ชำระค่าปรับในส่วนของค่าทรัพยากรให้ทำการยกเว้นค่าปรับของสมาชิกในส่วนของราคาทรัพยากรพร้อมทั้งส่งข้อความแจ้งเตือนทั้งทางข้อความและทางอีเมล์ของสมาชิกด้วย และแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ 3.e5: กรณีที่เกิดข้อผิดพลาดต่างๆ ให้ระบบแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ

ตารางที่ 6.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสสมาชิก	String	เป็นสมาชิกมีสิทธิ์ยืม	Normal flow of events
		เป็นสมาชิกexpire	1.e1 กรณีที่ไม่มีข้อมูลสมาชิกหรือสมาชิกหมดอายุ
		เป็นสมาชิกค้างเงินเกิน	Normal flow of events
		เป็นสมาชิกถูกบล็อก	1.e5 กรณีที่เป็นสมาชิกแต่ไม่สามารถยืมได้
		เป็นสมาชิกไม่มีสิทธิ์ยืม	1.e5 กรณีที่เป็นสมาชิกแต่ไม่สามารถยืมได้
		ไม่เป็นสมาชิก	1.e1 กรณีที่ไม่มีข้อมูลสมาชิกหรือสมาชิกหมดอายุ
รหัสทรัพยากร	String	AVยืมได้	Normal flow of events
		AVยืมไม่ได้typeเกินสิทธิ์	3.e2 กรณีที่ยืมเกินสิทธิ์
		BHถูกจอง	3.e1 กรณีที่มีสมาชิกท่านอื่นจองทรัพยากรนี้อยู่
		LUใช้ในห้องสมุดเท่านั้น	3.e5 กรณีที่เกิดข้อผิดพลาดต่างๆ
		LOหาย	3.e4 กรณีที่ทรัพยากรนี้มีสมาชิกแจ้งหายไว้
		เล่มเดียวกัน	3.e5 กรณีที่เกิดข้อผิดพลาดต่างๆ
		OnTheFly	3.e3 กรณีที่ไม่มีข้อมูลทรัพยากรนี้ในฐานข้อมูล
		เล่มสำรอง	3.e5 กรณีที่เกิดข้อผิดพลาดต่างๆ
		อื่นๆยืมไม่ได้	3.e4 กรณีที่ทรัพยากรนี้มีสมาชิกแจ้งหายไว้

☐ คำอธิบายยูสเคสถูกต้อง ☐ คำอธิบายยูสเคสมีข้อผิดพลาด

ข้อผิดพลาดคือ

.....
.....
.....
.....
.....

☐ ไม่สามารถระบุได้ว่าคำอธิบายยูสเคสถูกต้องหรือมีข้อผิดพลาด เนื่องจากยากต่อการดำเนินการด้วยคน

ตารางที่ 7.1 คำอธิบายยูสเคสหมายเลข 1.2

Use Case Name	Check in	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การคืนทรัพยากร เป็นกระบวนการรับคืนทรัพยากรที่สมาชิกได้ยืมจากห้องสมุด	
Trigger	สมาชิกนำสื่อที่ยืมไปมาคืนให้กับเจ้าหน้าที่ที่เคาน์เตอร์หรือที่ Book drop	
Relationships	Association:	-
	Include:	-
	Extend:	-
	Generalization:	-
Normal flow of events	1. ป้อนหรือสแกนรหัสบาร์โค้ดของทรัพยากรที่ต้องการคืนในช่อง “Item Barcode” 2. กรณีที่ต้องการเปลี่ยนประเภทการคืนทรัพยากร S-1 จะถูกกระทำ 3. รายการทรัพยากรที่คืนทั้งหมดจะแสดงอยู่ในส่วนของแท็บ “Check-In” 3.1. กรณีที่พบหนังสือแล้ว/ซื้อหนังสือมาคืน S-2 จะถูกกระทำ 3.2 กรณีที่ซื้อหนังสือใกล้เคียงมาคืน S-3 จะถูกกระทำ 3.3 กรณีที่หาซื้อหนังสือมาคืนไม่ได้ ต้องการชำระเงิน S-4 จะถูกกระทำ	
Sub Flows	S-1: เปลี่ยนประเภทการคืน 1. ระบบจะแสดงหน้าจอ login ขึ้นมา 2. หน้าจอปรากฏประเภทการคืนเมื่อ login ผ่าน 3. เมื่อเลือกเปลี่ยน Check-in Type เสร็จก็กดปุ่ม “OK” S-2: พบหนังสือแล้ว/ซื้อหนังสือมาคืน 1. ให้ยิงบาร์โค้ดในหน้าคืนแล้วระบบจะตรวจสอบว่าเป็นหนังสือที่ถูกแจ้งหายไว้หรือไม่ 2. มีหน้าต่างแสดง เมื่อเลือกกรณีจัดการทรัพยากรหายแล้ว กดปุ่ม	

	‘ตกลง’ 3. ระบบจะทำการเปลี่ยนสถานะตัวเล่มเป็น ‘AV’ และตัดหนี้ทรัพยากรหาย 4.การซื้อหนังสือมาคืนจะต้องนำหนังสือเล่มที่ซื้อไปติดบาร์โค้ดเดียวกับที่ได้แจ้งหายไว้ก่อน ระบบจะแสดงผลลัพธ์เหมือนคืนหนังสือปกติ S-3: ซื้อหนังสือใกล้เคียงมาคืน 1. การทำงานเช่นเดียวกับ S-2 แต่สถานะของตัวเล่มจะยังเป็น ‘LO’ ส่วนหนังสือที่นำมาคืนนี้เป็นคนละบาร์โค้ดกับเล่มที่ได้แจ้งหายไว้ แต่การแจ้งพบต้องป้อนบาร์โค้ดตัวเล่มที่แจ้งหายไว้ แล้วเลือกเป็นกรณี S-4: หาซื้อไม่ได้ต้องการชำระเงิน 1. การทำงานเช่นเดียวกับ S-3 แต่จะมีการพิมพ์ใบเสร็จรับเงิน S-5: แก้ไขค่าปรับ 1. สามารถเลือกรายการสีแดง เพื่อแก้ไข หรือลดหย่อนค่าปรับ
Alternate/Exception Flows	1.a1: กรณีที่มีค่าปรับจากการคืนทรัพยากรสาย ให้ระบบแสดงรายการทรัพยากรคืนสาย, จำนวนค่าปรับ และชื่อสมาชิกที่ยืมในส่วนของสีแดง หากสมาชิกต้องการแก้ไขหรือลดหย่อนค่าปรับ S-5 จะถูกกระทำ 1.a2: กรณีที่ทรัพยากรหรือหมายเลขบรรณานุกรมที่คืนมีสมาชิกจอง ให้ระบบแสดงข้อความเตือนพร้อมทั้งข้อมูลสมาชิกที่จองบางส่วนให้ผู้ใช้ทราบ

ตารางที่ 7.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสทรัพยากร	String	คืนสำเร็จ	Normal flow of events
		คืนสาย	1.a1 กรณีที่มีค่าปรับจากการคืนทรัพยากรสาย
		คืนแล้วมีคนจอง	1.a2 กรณีที่ทรัพยากรหรือหมายเลขบรรณานุกรมที่คืนมีสมาชิกจอง

		คืนแบบ Book drop	S-1 เปลี่ยนประเภทการคืน
		คืนหนังสือที่แจ้งหาย	S-2 พบหนังสือแล้ว/ซื้อหนังสือมาคืน
		ซื้อหนังสือใกล้เคียงมาคืน	S-3 ซื้อหนังสือใกล้เคียงมาคืน
		ชำระเงินค่าหนังสือที่แจ้งหาย	S-4 หาซื้อไม่ได้ต้องการชำระเงิน

☐ คำอธิบายยูสเคสถูกต้อง ☐ คำอธิบายยูสเคสมีข้อผิดพลาด

ข้อผิดพลาดคือ

.....
.....
.....
.....
.....

☐ ไม่สามารถระบุได้ว่าคำอธิบายยูสเคสถูกต้องหรือมีข้อผิดพลาด เนื่องจากยากต่อการดำเนินการด้วยคน

ตารางที่ 8.1 คำอธิบายยูสเคสหมายเลข 1.3

Use Case Name	Renew	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การให้บริการยืมทรัพยากรต่อ เป็นกระบวนการต่ออายุการยืมทรัพยากรที่สมาชิกได้ยืมไปแล้ว โดยสมาชิกสามารถยืมทรัพยากรต่อได้ถึง วิธีคือ ทาง 2web OPAC และการติดต่อที่จุดให้บริการ	
Trigger	สมาชิกมาติดต่อที่จุดให้บริการ หรือ ใช้บริการทางเว็บ OPAC	
Relationships	Association:	-
	Include:	-
	Extend:	-
	Generalization:	-
Normal flow of events	1. เลือกรายการที่ต้องการยืมต่อ แล้วกดปุ่ม “Renew Item” ระบบจะแสดงหน้าจอให้เลือกว่ายืมต่อทางใด ได้แก่ ทาง OPAC และทางเคาน์เตอร์ 2. ระบบแสดงข้อมูลเพื่อให้ผู้ใช้ยืนยันการยืมต่อ 3. แสดงผลการยืมต่อในส่วนสีน้ำเงิน	
Sub Flows	-	
Alternate/Exception Flows	3.e1: กรณีที่มีทรัพยากรที่ยืมต่อไม่ได้ แสดงในส่วนสีแดง พร้อมบอกเหตุผล	

ตารางที่ 8.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสสมาชิก	String	ยืมต่อได้	Normal flow of events
รหัสทรัพยากร	String	ยืมต่อสำเร็จ	Normal flow of events
		ยืมต่อไม่สำเร็จ - มีคนจอง	3.e1 กรณีที่มีทรัพยากรที่ยืมต่อไม่ได้
		ยืมต่อไม่สำเร็จ - วันข้ามไปตอนปิดเทอม	3.e1 กรณีที่มีทรัพยากรที่ยืมต่อไม่ได้
		ยืมต่อเกินโควต้า	3.e1 กรณีที่มีทรัพยากรที่ยืมต่อไม่ได้

☐ คำอธิบายยูสเคสถูกต้อง ☐ คำอธิบายยูสเคสมีข้อผิดพลาด

ข้อผิดพลาดคือ

.....
.....
.....
.....
.....

☐ ไม่สามารถระบุได้ว่าคำอธิบายยูสเคสถูกต้องหรือมีข้อผิดพลาด เนื่องจากยากต่อการดำเนินการด้วยคน

ตารางที่ 9.1 คำอธิบายยูสเคสหมายเลข 1.4

Use Case Name	Hold Item	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การให้บริการจองทรัพยากร เป็นกระบวนการจองทรัพยากรหรือยกเลิกการจองทรัพยากรให้แก่สมาชิกทั้งจองแบบเจาะจงตัวเล่มและแบบไม่เจาะจงตัวเล่ม โดยที่ทรัพยากรที่ต้องการจองนั้นจะต้องมีสถานะถูกยืมไปเท่านั้น	
Trigger	สมาชิกมาติดต่อที่จุดให้บริการ	
Relationships	Association:	-
	Include:	-
	Extend:	-
	Generalization:	-
Normal flow of events	1. เลือกแท็ป “Hold” จากนั้นเลือก “By Item” เพื่อจองแบบเจาะจงตัวเล่ม ป้อนหรือสแกนรหัสทรัพยากร 2. แสดงข้อมูลของทรัพยากรนั้นๆในช่องรายการด้านล่าง 3. ถ้ายกเลิกการจอง S-1 จะถูกเรียกใช้	
Sub Flows	S-1: ยกเลิกการจอง 1. กดปุ่ม “Cancel Hold Item(s)” จากนั้นระบบจะแสดงข้อความยืนยันการยกเลิกการจองพร้อมทั้งจำนวนรายการที่ต้องการยกเลิก 2. ระบบจะยกเลิกรายการจอง พร้อมทั้งกำหนดสถานะการยกเลิกเป็นยกเลิกการจอง 3. ทำการปรับปรุงลำดับการจองทรัพยากรในแต่ละรายการ	
Alternate/Exception Flows	1.a1: เลือก “By Bibliographic” เพื่อจองแบบไม่เจาะจงตัวเล่ม ป้อนหรือสแกนรหัสทรัพยากร 1.e1: กรณีที่จองเกินสิทธิ์ ถ้าผู้ใช้ไม่มีสิทธิ์ดำเนินการให้ระบบแสดงหน้าจอ Login ขึ้นมา 1.e2: กรณีที่เกิดข้อผิดพลาดต่างๆ ให้ระบบแสดงข้อความแจ้งเตือน	

ตารางที่ 9.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสสมาชิก	string	เป็นสมาชิกมีสิทธิ์ยืม	Normal flow of events
		เป็นสมาชิกexpire	1.e2 กรณีที่เกิดข้อผิดพลาดต่างๆ
		เป็นสมาชิกค้างเงินเกิน	1.e2 กรณีที่เกิดข้อผิดพลาดต่างๆ
		เป็นสมาชิกถูกblock	1.e2 กรณีที่เกิดข้อผิดพลาดต่างๆ
		เป็นสมาชิกไม่มีสิทธิ์ยืม	1.e2 กรณีที่เกิดข้อผิดพลาดต่างๆ
		ไม่เป็นสมาชิก	1.e2 กรณีที่เกิดข้อผิดพลาดต่างๆ
รหัสทรัพยากร	string	AVจองได้	Normal flow of events
		AVจองไม่ได้typeเกินสิทธิ์	1.e1 กรณีที่จองเกินสิทธิ์
		LUใช้ในห้องสมุดเท่านั้น	1.e2 กรณีที่เกิดข้อผิดพลาดต่างๆ
		LOหาย	1.e2 กรณีที่เกิดข้อผิดพลาดต่างๆ
		อื่นๆจองไม่ได้	1.e2 กรณีที่เกิดข้อผิดพลาดต่างๆ

☐ คำอธิบายยูสเคสถูกต้อง ☐ คำอธิบายยูสเคสมีข้อผิดพลาด

ข้อผิดพลาดคือ

.....
.....
.....
.....
.....
.....

☐ ไม่สามารถระบุได้ว่าคำอธิบายยูสเคสถูกต้องหรือมีข้อผิดพลาด เนื่องจากยากต่อการดำเนินการด้วยคน

ตารางที่ 10.1 คำอธิบายยูสเคสหมายเลข 1.5

Use Case Name	Fine	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การจัดการค่าปรับของสมาชิก เป็นกระบวนการของการชำระ แก้ไข ยกเว้น และการเพิ่มค่าปรับของสมาชิก	
Trigger	สมาชิกมาติดต่อที่จุดให้บริการ	
Relationships	Association:	-
	Include:	-
	Extend:	ลงชื่อเข้าใช้
	Generalization:	-
Normal flow of events	1. คลิกแท็ป “Debt” 2. ในช่อง Balance จะต้องแสดงจำนวนค่าปรับทั้งหมดที่มี 3. ในกรณีที่ต้องการชำระเงิน S-1 จะถูกกระทำ 4. ในกรณีที่ต้องการลดหย่อนค่าปรับ S-2 จะถูกกระทำ	
Sub Flows	S-1: ชำระเงิน 1. เลือกรายการที่ต้องการชำระ 2. ในช่อง Pay Debt จะต้องแสดงจำนวนค่าปรับที่จะชำระ 3. ในกรณีที่ต้องการชำระค่าปรับบางส่วน S-1a1 จะถูกกระทำ 4. ดำเนินการชำระ 5. ในกรณีที่ต้องการพิมพ์ใบเสร็จรับเงิน S-4 จะถูกกระทำ S-2: ลดหย่อนค่าปรับ 1. กดปุ่ม “Deduct Debt” 2. ระบบจะแสดงหน้าจอ “Login”เพื่อตรวจสอบสิทธิ์การเข้าใช้งาน 3. ป้อนจำนวนเงินที่จะลดหย่อน และป้อนหมายเหตุของการลดหย่อน ซึ่งผู้ใช้จำเป็นต้องป้อนหมายเหตุของการยกเว้นนี้ 4. กดปุ่ม “OK” S-3: เพิ่มค่าปรับ 1. กดปุ่ม “Add Debt”	

	2. ระบบแสดงหน้าจอ login ขึ้นมา เพื่อตรวจสอบสิทธิ์ 3. ระบบจะแสดงหน้าจอเพิ่มรายการค่าปรับขึ้นมา 4. ในการเพิ่มรายการค่าปรับผู้ใช้จำเป็นที่จะต้องป้อนหมายเหตุของการเพิ่มรายการค่าปรับ S-4: พิมพ์ใบเสร็จรับเงิน 1. พิมพ์ใบเสร็จรับเงินออกทางเครื่องพิมพ์
Alternate/Exception Flows	S-1a1: กรณีที่ชำระค่าปรับบางส่วนของรายการ ป้อนจำนวนเงินที่ต้องการชำระ ตัดยอดชำระจากรายการที่เลือกตามลำดับ และแสดงยอดคงเหลือ

ตารางที่ 10.2 ค่า inputs และ outputs ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสสมาชิก	string	เกินกำหนดส่ง	Normal flow of events
		หนังสือหาย	Normal flow of events
		หนังสือชำรุด	Normal flow of events
ตัวเลือกดำเนินการ	string	เพิ่มค่าปรับ	S-3 เพิ่มค่าปรับ
		ลดหย่อนค่าปรับ	S-2 ลดหย่อนค่าปรับ
		ชำระค่าปรับทั้งหมดของรายการ	S-1 ชำระเงิน
		ชำระค่าปรับบางส่วนของรายการ	S-1a1 กรณีที่ชำระค่าปรับบางส่วนของรายการ

☐ คำอธิบายยูสเคสถูกต้อง ☐ คำอธิบายยูสเคสมีข้อผิดพลาด

ข้อผิดพลาดคือ

.....
.....
.....
.....
.....
.....

☐ ไม่สามารถระบุได้ว่าคำอธิบายยูสเคสถูกต้องหรือมีข้อผิดพลาด เนื่องจากยากต่อการดำเนินการด้วยคน

<u>โจทย์สำหรับการทดลองวิจัย 2</u>

<u>คำสั่งดำเนินงาน</u> ขอให้ผู้เข้าทดลองพิจารณาคำอธิบายยูสเคสหมายเลข 2.1 ถึงหมายเลข 2.5 และกรุณาระบุสถานการณ์ (Scenario) ที่เป็นไปได้ในแต่ละกรณีลงในกระดาษคำตอบนี้

ตารางที่ 11.1 คำอธิบายยูสเคสหมายเลข 2.1

Use Case Name	Check out	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การให้บริการยืมทรัพยากร เป็นกระบวนการเพื่อให้สมาชิกสามารถยืมทรัพยากรสารสนเทศจากห้องสมุดได้ โดยสมาชิกจะต้องนำบัตรสมาชิกและทรัพยากรที่ต้องการยืมมาติดต่อที่จุดให้บริการ	
Trigger	สมาชิกนำบัตรสมาชิกและทรัพยากรที่ต้องการยืมมาติดต่อที่จุดให้บริการ	
Relationships	Association:	-
	Include:	-
	Extend:	ลงชื่อเข้าใช้
	Generalization:	-
Normal flow of events	1. ป้อนหรือสแกนรหัสสมาชิกลงในช่อง “Barcode” หากมีข้อมูลสมาชิก ระบบจะแสดงรายละเอียดของสมาชิกบางส่วนทางซ้ายมือ 2. ระบบจะแสดงข้อมูลการยืมที่ยังไม่ได้คืน, การจอง, รายการค่าปรับ และข้อความแจ้งเตือนต่างๆ ของสมาชิก 3. ป้อนหรือสแกนรหัสบาร์โค้ดของทรัพยากรลงในช่อง “Barcode” 4. ดำเนินการยืมทรัพยากรให้แก่สมาชิกพร้อมทั้งแสดงรายการทรัพยากรที่ยืมในส่วนของ “Checked-out” เป็นสีแดง	
Sub Flows	S-1: ยืมเกินสิทธิ์ 1. ระบบจะแสดงคำถามยืนยันการยืมเกินสิทธิ์ 2. แสดงหน้าจอ login	

Alternate/Exception Flows	1.e1: กรณีที่ไม่มีข้อมูลสมาชิกหรือสมาชิกหมดอายุ ระบบจะแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบด้วย พร้อมทั้งถามว่าจะต่ออายุบัตรสมาชิกใหม่หรือไม่? ด้วย ถ้าไม่ต้องการให้ทำการล้างหน้าจอทั้งหมด 1.e2: กรณีที่สมาชิกมีรายการค้างค่าปรับ, มีรายการให้รับทรัพยากรที่จองหรือมีข้อความสำคัญต่างๆ ให้ระบบแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ พร้อมทั้งให้ข้อความบนแท็ปนั้นเป็นสีแดง 1.e5: กรณีที่เป็นสมาชิกแต่ไม่สามารถยืมได้ เนื่องจาก ถูกบล็อก ค้างเงินเกิน หรือไม่มีสิทธิ์ยืม ให้ระบบแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ 3.e1: กรณีที่มีสมาชิกท่านอื่นจองทรัพยากรนี้อยู่ ให้ระบบแสดงข้อความแจ้งเตือนพร้อมข้อมูลบางส่วนของสมาชิกที่จองทรัพยากรนี้3.e2: กรณีที่ยืมเกินสิทธิ์ S-1 จะถูกกระทำ 3.e3: กรณีที่ไม่มีข้อมูลทรัพยากรนี้ในฐานข้อมูล ระบบจะต้องแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบและแสดงหน้าจอให้เพิ่มทรัพยากรแบบเร่งด่วน (Item On The Fly) โดยผู้ใช้ต้องป้อนข้อมูลทั้งหมดที่เป็นสีแดง 3.e4: กรณีที่ทรัพยากรนี้มีสมาชิกแจ้งหายไว้ ให้ระบบปรับปรุงสถานะของทรัพยากรนี้และหากสมาชิกยังไม่ได้ชำระค่าปรับในส่วนของค่าทรัพยากรให้ทำการยกเว้นค่าปรับของสมาชิกในส่วนของราคาทรัพยากรพร้อมทั้งส่งข้อความแจ้งเตือนทั้งทางข้อความและทางอีเมล์ของสมาชิกด้วย และแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ3.e5: กรณีที่เกิดข้อผิดพลาดต่างๆ ให้ระบบแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ 3.a1: กรณีที่ยืมตัวเล่มที่เป็นทรัพยากรสำรอง จะแสดงหน้าจอให้ผู้ใช้ทราบว่าเป็นทรัพยากรสำรองให้ยืมกี่วัน

ตารางที่ 11.2 ค่า inputs และ paths ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสสมาชิก	String	เป็นสมาชิกมีสิทธิ์ยืม	Normal flow of events
		เป็นสมาชิกexpire	1.e1 กรณีที่ไม่มีข้อมูลสมาชิกหรือ

			สมาชิกหมดอายุ
		เป็นสมาชิกมีรายการค้างค่าปรับหรือมีข้อความสำคัญต่างๆ	1.e2 กรณีที่สมาชิกมีรายการค้างค่าปรับ, มีรายการให้รับทรัพยากรที่จองหรือมีข้อความสำคัญต่างๆ
		เป็นสมาชิกถูกบล็อก	1.e5 กรณีที่เป็นสมาชิกแต่ไม่สามารถยืมได้
		เป็นสมาชิกไม่มีสิทธิ์ยืม	1.e5 กรณีที่เป็นสมาชิกแต่ไม่สามารถยืมได้
		ไม่เป็นสมาชิก	1.e1 กรณีที่ไม่มีข้อมูลสมาชิกหรือสมาชิกหมดอายุ
รหัสทรัพยากร	String	AVยืมได้	Normal flow of events
		AVยืมไม่ได้typeเกินสิทธิ์	3.e2 กรณีที่ยืมเกินสิทธิ์
		BHถูกจอง	3.e1 กรณีที่มีสมาชิกท่านอื่นจองทรัพยากรนี้อยู่
		LUใช้ในห้องสมุดเท่านั้น	3.e5 กรณีที่เกิดข้อผิดพลาดต่างๆ
		LOหาย	3.e4 กรณีที่ทรัพยากรนี้มีสมาชิกแจ้งหายไว้
		เล่มเดียวกัน	3.e5 กรณีที่เกิดข้อผิดพลาดต่างๆ
		OnTheFly	3.e3 กรณีที่ไม่มีข้อมูลทรัพยากรนี้ในฐานข้อมูล
		เล่มสำรอง	3.a1 กรณีที่ยืมตัวเล่มที่เป็นทรัพยากรสำรอง
		อื่นๆยืมไม่ได้	3.e5 กรณีที่เกิดข้อผิดพลาดต่างๆ

ตารางที่ 11.3 ตารางตัดสินใจของยูสเคส Checkout

Path	Input	
	รหัสสมาชิก	**รหัสทรัพยากร**
Normal flow of events		
S-1: ยืมเกินสิทธิ์		
1.e1: ไม่มีข้อมูลสมาชิกหรือสมาชิกหมดอายุ		
1.e2: สมาชิกมีรายการค้างค่าปรับ		
1.e5: กรณีที่เป็นสมาชิกแต่ไม่สามารถยืมได้		
3.e1: สมาชิกท่านอื่นจองทรัพยากรนี้อยู่		
3.e2: ยืมเกินสิทธิ์		
3.e3: ไม่มีข้อมูลทรัพยากรนี้ในฐานข้อมูล		
3.e4: ทรัพยากรนี้มีสมาชิกแจ้งหายไว้		
3.e5: เกิดข้อผิดพลาดต่างๆ		
3.a1: ยืมตัวเล่มที่เป็นทรัพยากรสำรอง		

ตารางที่ 12.1 คำอธิบายยูสเคสหมายเลข 2.2

Use Case Name	Check in	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การคืนทรัพยากร เป็นกระบวนการรับคืนทรัพยากรที่สมาชิกได้ยืมจากห้องสมุด	
Trigger	สมาชิกนำสื่อที่ยืมไปมาคืนให้กับเจ้าหน้าที่ที่เคาน์เตอร์หรือที่ Book drop	
Relationships	Association:	-
	Include:	-
	Extend:	-
	Generalization:	-
Normal flow of events	1. ป้อนหรือสแกนรหัสบาร์โค้ดของทรัพยากรที่ต้องการคืนในช่อง “Item Barcode” 2. กรณีที่ต้องการเปลี่ยนประเภทการคืนทรัพยากร S-1 เปลี่ยนประเภทการคืน จะถูกกระทำ 3. รายการทรัพยากรที่คืนทั้งหมดจะแสดงอยู่ในส่วนของแท็บ “Check-In” 3.1. กรณีที่พบหนังสือแล้ว/ซื้อหนังสือมาคืน S-2 จะถูกกระทำ 3.2 กรณีที่ซื้อหนังสือใกล้เคียงมาคืน S-3 จะถูกกระทำ 3.3 กรณีที่หาซื้อหนังสือมาคืนไม่ได้ ต้องการชำระเงิน S-4 จะถูกกระทำ	
Sub Flows	S-1: เปลี่ยนประเภทการคืน 1. ระบบจะแสดงหน้าจอ login ขึ้นมา 2. หน้าจอปรากฏประเภทการคืนเมื่อ login ผ่าน 3. เมื่อเลือกเปลี่ยน Check-in Type เสร็จก็กดปุ่ม “OK” S-2: พบหนังสือแล้ว/ซื้อหนังสือมาคืน 1. ให้ยิงบาร์โค้ดในหน้าคืนแล้วระบบจะตรวจสอบว่าเป็นหนังสือที่ถูกแจ้งหายไว้หรือไม่ 2. มีหน้าต่างแสดง เลือกจัดการทรัพยากรหายแล้วกดปุ่ม ‘ตกลง’	

	3. ระบบจะทำการเปลี่ยนสถานะตัวเล่มเป็น ‘AV’ และตัดหนี้ทรัพยากรหาย 4.การซื้อหนังสือมาคืนจะต้องนำหนังสือเล่มที่ซื้อไปติดบาร์โค้ดเดียวกับที่ได้แจ้งหายไว้ก่อน ระบบจะแสดงผลลัพธ์เหมือนคืนหนังสือปกติ S-3: ซื้อหนังสือใกล้เคียงมาคืน 1. การทำงานเช่นเดียวกับ S-2 แต่สถานะของตัวเล่มจะยังเป็น ‘LO’ ส่วนหนังสือที่นำมาคืนนี้เป็นคนละบาร์โค้ดกับเล่มที่ได้แจ้งหายไว้ แต่การแจ้งพบต้องป้อนบาร์โค้ดตัวเล่มที่แจ้งหายไว้ แล้วเลือกเป็นกรณีS-4: หาซื้อไม่ได้ต้องการชำระเงิน 1. การทำงานเช่นเดียวกับ S-3 แต่จะมีการพิมพ์ใบเสร็จรับเงิน S-5: ลดหย่อนค่าปรับ 1. สามารถเลือกรายการสีแดง เพื่อแก้ไข หรือลดหย่อนค่าปรับ
Alternate/Exception Flows	1.a1: กรณีที่มีค่าปรับจากการคืนทรัพยากรสายให้ระบบแสดงรายการทรัพยากรคืนสาย, จำนวนค่าปรับ และชื่อสมาชิกที่ยืมในส่วนของสีแดง หากสมาชิกต้องการแก้ไขหรือลดหย่อนค่าปรับ S-5 ลดหย่อนค่าปรับ จะถูกกระทำ 1.a2: กรณีที่ทรัพยากรหรือหมายเลขบรรณานุกรมที่คืนมีสมาชิกจอง ให้ระบบแสดงข้อความเตือนพร้อมทั้งข้อมูลสมาชิกที่จองบางส่วนให้ผู้ใช้ทราบ

ตารางที่ 12.2 ค่า inputs และ paths ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสทรัพยากร	String	คืนสำเร็จ	Normal flow of events
		คืนแล้วมีคนจอง	1.a2 กรณีที่ทรัพยากรหรือหมายเลขบรรณานุกรมที่คืนมีสมาชิกจอง
		คืนแบบ Book drop	S-1 เปลี่ยนประเภทการคืน
		คืนหนังสือที่แจ้งหาย	S-2 พบหนังสือแล้ว/ซื้อหนังสือมาคืน

		ซื้อหนังสือใกล้เคียงมาคืน	S-3 ซื้อหนังสือใกล้เคียงมาคืน
		ชำระเงินค่าหนังสือที่แจ้งหาย	S-หาซื้อไม่ได้ต้องการชำระ 4 เงิน
		ลดหย่อนค่าปรับ	S-5 ลดหย่อนค่าปรับ
		คืนหนังสือที่เป็น OnTheFly	Normal flow of events

ตารางที่ 12.3 ตารางตัดสินใจของยูสเคส Check in

Path	**Input**
	รหัสทรัพยากร
Normal flow of events	
S-1: เปลี่ยนประเภทการคืน	
S-2: พบหนังสือแล้วซื้อหนังสือมาคืน/	
S-3: ซื้อหนังสือใกล้เคียงมาคืน	
S-4: หาซื้อไม่ได้ต้องการชำระเงิน	
S-5: ลดหย่อนค่าปรับ	
1.a1: คืนทรัพยากรสาย	
1.a2: ทรัพยากรหรือหมายเลขบรรณานุกรมที่คืนมีสมาชิกจอง	
1.a3: ทรัพยากรนี้มีสมาชิกแจ้งหายไว้	

ตารางที่ 13.1 คำอธิบายยูสเคสหมายเลข 2.3

Use Case Name	Renew	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การให้บริการยืมทรัพยากรต่อ เป็นกระบวนการต่ออายุการยืมทรัพยากรที่สมาชิกได้ยืมไปแล้ว โดยสมาชิกสามารถยืมทรัพยากรต่อได้ถึง วิธีคือ ทาง 2web OPAC และการติดต่อที่จุดให้บริการ	
Trigger	สมาชิกมาติดต่อที่จุดให้บริการ หรือ ใช้บริการทางเว็บ OPAC	
Relationships	Association:	-
	Include:	-
	Extend:	-
	Generalization:	-
Normal flow of events	1. เลือกรายการที่ต้องการยืมต่อ แล้วกดปุ่ม “Renew Item” ระบบจะแสดงหน้าจอให้เลือกว่ายืมต่อทางใด ได้แก่ ทาง OPAC และทางเคาน์เตอร์ 2. ระบบแสดงข้อมูลเพื่อให้ผู้ใช้ยืนยันการยืมต่อ 3. แสดงผลการยืมต่อในส่วนสีน้ำเงิน	
Sub Flows	-	
Alternate/Exception Flows	3.e1: กรณีที่มีทรัพยากรที่ยืมต่อไม่ได้ แสดงในส่วนสีแดง พร้อมบอกเหตุผล	

ตารางที่ 13.2 ค่า inputs และ paths ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสสมาชิก	String	ยืมต่อได้	Normal flow of events
รหัสทรัพยากร	String	ยืมต่อสำเร็จ	Normal flow of events
		ยืมต่อไม่สำเร็จ - มีคนจอง	3.e1 กรณีที่มีทรัพยากรที่ยืมต่อไม่ได้
		ยืมต่อไม่สำเร็จ - วันข้ามไปตอนปิดเทอม	3.e1 กรณีที่มีทรัพยากรที่ยืมต่อไม่ได้
		ยืมต่อเกินโควต้า	3.e1 กรณีที่มีทรัพยากรที่ยืมต่อไม่ได้

ตารางที่ 13.3 ตารางตัดสินใจของยูสเคส Renew

Path	Input	
	รหัสสมาชิก	**รหัสทรัพยากร**
Normal flow of events		
3.e1: มีทรัพยากรที่ยืมต่อไม่ได้		

ตารางที่ 14.1 คำอธิบายยูสเคสหมายเลข 2.4

Use Case Name	Hold Item	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การให้บริการจองทรัพยากร เป็นกระบวนการจองทรัพยากรหรือยกเลิกการจองทรัพยากรให้แก่สมาชิกทั้งจองแบบเจาะจงตัวเล่มและแบบไม่เจาะจงตัวเล่ม โดยที่ทรัพยากรที่ต้องการจองนั้นจะต้องมีสถานะถูกยืมไปเท่านั้น	
Trigger	สมาชิกมาติดต่อที่จุดให้บริการ	
Relationships	Association:	-
	Include:	-
	Extend:	ลงชื่อเข้าใช้
	Generalization:	-
Normal flow of events	1. เลือกแท็ป “Hold” จากนั้นเลือก “By Item” เพื่อจองแบบเจาะจงตัวเล่ม ป้อนหรือสแกนรหัสทรัพยากร 2. แสดงข้อมูลของทรัพยากรนั้นๆในช่องรายการด้านล่าง 3. ถ้ายกเลิกการจอง S-1 จะถูกเรียกใช้	
Sub Flows	S-1: ยกเลิกการจอง 1. กดปุ่ม “Cancel Hold Item(s)” จากนั้นระบบจะแสดงข้อความยืนยันการยกเลิกการจองพร้อมทั้งจำนวนรายการที่ต้องการยกเลิก 2. ระบบจะยกเลิกรายการจอง พร้อมทั้งกำหนดสถานะการยกเลิกเป็นยกเลิกการจอง 3. ทำการปรับปรุงลำดับการจองทรัพยากรในแต่ละรายการ	
Alternate/Exception Flows	1.a1: เลือก “By Bibliographic” เพื่อจองแบบไม่เจาะจงตัวเล่ม ป้อนหรือสแกนรหัสทรัพยากร 1.e1: กรณีที่จองเกินสิทธิ์ ถ้าผู้ใช้ไม่มีสิทธิ์ดำเนินการให้ระบบแสดงหน้าจอ Login ขึ้นมา 1.e2: กรณีที่เกิดข้อผิดพลาดต่างๆ ให้ระบบแสดงข้อความแจ้งเตือน	

ตารางที่ 14.2 ค่า inputs และ paths ที่เป็นได้

Inputs	Types	Possible value	Response Path
รหัสสมาชิก	string	เป็นสมาชิกมีสิทธิ์ยืม	Normal flow of events
		เป็นสมาชิกexpire	1.e2 กรณีที่เกิดข้อผิดพลาด
		เป็นสมาชิกค้างเงินเกิน	1.e2 กรณีที่เกิดข้อผิดพลาด
		เป็นสมาชิกถูกblock	1.e2 กรณีที่เกิดข้อผิดพลาด
		เป็นสมาชิกไม่มีสิทธิ์ยืม	1.e2 กรณีที่เกิดข้อผิดพลาด
		ไม่เป็นสมาชิก	1.e2 กรณีที่เกิดข้อผิดพลาด
รหัสทรัพยากร	string	จองเล่มเดียวกัน	1.e2 กรณีที่เกิดข้อผิดพลาด
		AVจองได้	Normal flow of events
		AVจองไม่ได้typeเกินโควต้า	1.e1 กรณีที่จองเกินสิทธิ์
		LUใช้ในห้องสมุดเท่านั้น	1.e2 กรณีที่เกิดข้อผิดพลาด
		LOหาย	1.e2 กรณีที่เกิดข้อผิดพลาด
		อื่นๆจองไม่ได้	1.e2 กรณีที่เกิดข้อผิดพลาด

ตารางที่ 14.3 ตารางตัดสินใจของยูสเคส Hold item

Path	Input	
	รหัสสมาชิก	**รหัสทรัพยากร**
Normal flow of events		
S-1: ยกเลิกการจอง		
1.a1: จองแบบไม่เจาะจงตัวเล่ม		
1.e1: จองเกินสิทธิ์		
1.e2: เกิดข้อผิดพลาดต่างๆ		

ตารางที่ 15.1 คำอธิบายยูสเคสหมายเลข 2.5

Use Case Name	Fine	
Primary Actor	เจ้าหน้าที่, สมาชิก	
Stakeholders	เจ้าหน้าที่, สมาชิก	
Brief Description	การจัดการค่าปรับของสมาชิก เป็นกระบวนการของการชำระ แก้ไข ยกเว้น และการเพิ่มค่าปรับของสมาชิก	
Trigger	สมาชิกมาติดต่อที่จุดให้บริการ	
Relationships	Association:	-
	Include:	-
	Extend:	ลงชื่อเข้าใช้
	Generalization:	-
Normal flow of events	1. คลิกแท็ป “Debt” 2. ในช่อง Balance จะต้องแสดงจำนวนค่าปรับทั้งหมดที่มี 3. ในกรณีที่ต้องการชำระเงิน S-1 จะถูกกระทำ 4. ในกรณีที่ต้องการลดหย่อนค่าปรับ S-2 จะถูกกระทำ 5. ในกรณีที่ต้องการเพิ่มค่าปรับ S-3 จะถูกกระทำ	
Sub Flows	S-1: ชำระเงิน 1. เลือกรายการที่ต้องการชำระ 2. ในช่อง Pay Debt จะต้องแสดงจำนวนค่าปรับที่จะชำระ 3. ในกรณีที่ต้องการชำระค่าปรับบางส่วน S-1a1 จะถูกกระทำ 4. ดำเนินการชำระ 5. ในกรณีที่ต้องการพิมพ์ใบเสร็จรับเงิน S-4 จะถูกกระทำ S-2: ลดหย่อนค่าปรับ 1. กดปุ่ม “Deduct Debt” 2. ระบบจะแสดงหน้าจอ “Login”เพื่อตรวจสอบสิทธิ์การเข้าใช้งาน 3. ป้อนจำนวนเงินที่จะลดหย่อน และป้อนหมายเหตุของการลดหย่อน ซึ่งผู้ใช้จำเป็นต้องป้อนหมายเหตุของการยกเว้นนี้ 4. กดปุ่ม “OK” S-3: เพิ่มค่าปรับ	

	1. กดปุ่ม “Add Debt” 2. ระบบแสดงหน้าจอ login ขึ้นมา เพื่อตรวจสอบสิทธิ์การเข้าใช้งาน 3. ระบบจะแสดงหน้าจอเพิ่มรายการค่าปรับขึ้นมา 4. ในการเพิ่มรายการค่าปรับผู้ใช้จำเป็นที่จะต้องป้อนหมายเหตุของการเพิ่มรายการค่าปรับ S-4: พิมพ์ใบเสร็จรับเงิน 1. พิมพ์ใบเสร็จรับเงินออกทางเครื่องพิมพ์
Alternate/Exception Flows	S-1a1: กรณีที่ชำระค่าปรับบางส่วนของรายการ ป้อนจำนวนเงินที่ต้องการชำระ ตัดยอดชำระจากรายการที่เลือกตามลำดับ และแสดงยอดคงเหลือ

ตารางที่ 15.2 ค่า inputs และ path ที่เป็นไปได้

Inputs	Types	Possible value	Response Path
รหัสสมาชิก	string	เกินกำหนดส่ง	Normal flow of events
		หนังสือหาย	Normal flow of events
		หนังสือชำรุด	Normal flow of events
ตัวเลือกดำเนินการ	string	เพิ่มค่าปรับ	S-3 เพิ่มค่าปรับ
		ลดหย่อนค่าปรับ	S-2 ลดหย่อนค่าปรับ
		ชำระค่าปรับทั้งหมดของรายการ	S-1 ชำระเงิน
		ชำระค่าปรับบางส่วนของรายการ	S-1a1 กรณีที่ชำระค่าปรับบางส่วนของรายการ

ตารางที่ 15.3 ตารางตัดสินใจของยูสเคส Fine

Path	Input	
	รหัสสมาชิก	**ตัวเลือกดำเนินการ**
Normal flow of events		
S-1: ชำระเงิน		
S-2: ลดหย่อนค่าปรับ		
S-3: เพิ่มค่าปรับ		
S-4: พิมพ์ใบเสร็จรับเงิน		
S-1a1: กรณีที่ชำระค่าปรับบางส่วนของรายการ		

<u>คำรับรองการเข้าร่วมประเมินระบบ</u>

ข้าพเจ้า ชื่อ-สกุล .....

ตำแหน่ง .....

หน่วยงาน/สังกัด .....

ขอรับรองว่า ได้เข้าร่วมประเมินระบบการตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล

ลงนาม..... ....

(.....)

วันที่_____/_____/_____

ภาคผนวก จ.

แบบประเมินความพึงพอใจ

<u>คำชี้แจง</u>

งานทดลองวิจัยเรื่อง: การตรวจสอบความถูกต้องของยูสเคสสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอล

ผู้ดำเนินการวิจัย: นายปรณัฐ เตี้ยนวล

ที่มาและความสำคัญ: ปัจจุบัน การพัฒนาซอฟต์แวร์เชิงวัตถุด้วยยูเอ็มแอลได้รับความนิยมมาก โดยเฉพาะแผนภาพยูสเคสมักถูกใช้ในการวิเคราะห์ฟังก์ชันของระบบซอฟต์แวร์ที่จะสร้างขึ้น ดังนั้น แผนภาพยูสเคสจึงเป็นสารสนเทศที่สำคัญ ซึ่งส่งต่อไปยังการวิเคราะห์และออกแบบในกระบวนการดำเนินงานด้วยยูเอ็มแอลในขั้นตอนถัดไป เช่น แผนภาพกิจกรรม (Activity Diagram) แผนภาพลำดับ (Sequence Diagram) และแผนภาพคลาส (Class Diagram) ด้วยเหตุนี้ หากยูสเคสนี้ผิดก็จะส่งผลความผิดพลาดตามไปด้วย ทั้งนี้ ในกระบวนการวิศวกรรมความต้องการสามารถออกแบบกรณีทดสอบได้ตั้งแต่สร้างคำอธิบายยูสเคส

แบบประเมินแบ่งเป็น 2 ตอน คือ

1. ข้อมูลเกี่ยวกับผู้ใช้
2. แบบประเมินความพึงพอใจในการใช้งานระบบและข้อเสนอแนะ

<u>ตอนที่ 1</u> ข้อมูลเกี่ยวกับผู้ใช้

1.1 เพศ ☐ ชาย ☐ หญิง อายุ.....ปี

1.2 ระดับการศึกษาสูงสุด ☐ ปริญญาตรี ☐ ปริญญาโท

☐ ปริญญาเอก ☐ อื่นๆ ระบุ.....

1.3 วุฒิการศึกษาสูงสุด (เช่น วท.บ.คณิตศาสตร์).....

1.4 ชื่อหน่วยงาน / ภาควิชาที่สังกัด.....

1.5 ตำแหน่งงานปัจจุบัน.....

1.6 อายุงานในตำแหน่งปัจจุบัน.....ปี.....เดือน

ตอนที่ 2 แบบประเมินความพึงพอใจของผู้ใช้

กรุณาใส่เครื่องหมาย ✓ ลงในช่องที่เห็นว่าใกล้เคียงกับความจริงมากที่สุด โดยมีเกณฑ์การให้คะแนนดังนี้

ระดับความพึงพอใจ 5 = ดีมาก 4 = ดี 3 = ปานกลาง 2 = พอใช้ 1 = ปรับปรุง

ข้อ	รายการประเมินผล	ระดับความพึงพอใจ				
		5	4	3	2	1
1	**ด้านการทำงานของระบบ**					
	1.1 ความถูกต้องของการทำงาน					
	1.2 ความสมบูรณ์ของผลลัพธ์					
	1.3 ความเร็วในการประมวลผล					
	1.4 ความสามารถในการป้องกันความผิดพลาดจากผู้ใช้					

ความคิดเห็นและข้อเสนอแนะ

.....

.....

ข้อ	รายการประเมินผล	ระดับความพึงพอใจ				
		5	4	3	2	1
2	**ด้านการติดต่อกับผู้ใช้งาน**					
	2.1 การใช้งานง่ายและเป็นมิตรกับผู้ใช้ เช่น การจัดลำดับข้อมูลบนหน้าจอ					
	2.2 ความเหมาะสมในการวางตำแหน่งองค์ประกอบบนน้าจอ					
	2.3 ความชัดเจนในการสื่อความหมายของส่วนต่างๆ					
	2.4 ในกรณีที่มีข้อผิดพลาดมีการแสดงให้เห็นอย่างชัดเจน					
	2.5 ความสวยงามของการแสดงผลบนหน้าจอ เช่น การใช้สีหรือขนาดของตัวอักษร					

ความคิดเห็นและข้อเสนอแนะ

.....

.....

ข้อ	รายการประเมินผล	ระดับความพึงพอใจ				
		5	4	3	2	1
3	**ด้านคุณค่าของระบบ**					
	3.1 ช่วยลดเวลาในการตรวจสอบความถูกต้องของยูสเคส					
	3.2 ช่วยเพิ่มความถูกต้อง แม่นยำ ในการตรวจสอบความถูกต้องของยูสเคส					

ความคิดเห็นและข้อเสนอแนะ

.....

.....

ขอบคุณที่เสียสละเวลาในการตอบแบบประเมิน

ประวัติผู้เขียน

ชื่อ สกุล นายปรณัฐ เตี้ยนวล
รหัสประจำตัวนักศึกษา 5710220053
วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2556

ทุนการศึกษา

ทุนผู้ช่วยสอน บัณฑิตศึกษา มหาวิทยาลัยสงขลานครินทร์ ประจำปี 2557 – 2561

การตีพิมพ์เผยแพร่ผลงาน

Poranat Tianual and Amnart Pohthong. 2019. Defects Detection Technique of Use Case Views during Requirements Engineering. 2019 8th International Conference on Software and Computer Applications (ICSCA 2019). Penang, Malaysia, 19 – 22 February 2019.