SOFTWARE

REQUIREMENTS ENGINEERING AND DOCUMENTATION

IMAGE BY WIRESTOC ON FREEPIK : HTTPS://WWW.FREEPIK.COM/FREE-PHOTO/BUSINESSMAN-SUIT-PUTTING-LAST-PIECE-PYRAMID-USING-WOODEN-BLOCKS_17344606

วิศวกรรมความต้องการ และเอกสารโครงการซอฟต์แวร์

Bachelor of Science Program in
Software Engineering Faculty of Informatics,
Burapha University, Thailand

อภิสิทธิ์ แสงใส

รายวิชา 88823265 วิศวกรรมความต้องการและเอกสารโครงการซอฟต์แวร์

Software Requirements Engineering and Documentation

ผู้เขียน นายอภิสิทธิ์ แสงใส

สาขาวิชาวิศวกรรมซอฟต์แวร์

คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา

คำนำ

เอกสารคำสอนรายวิชา 88823265 วิศวกรรมความต้องการและเอกสารโครงการซอฟต์แวร์ ผู้เขียนได้รวบรวมและเรียบเรียง โดยมีวัตถุประสงค์เพื่อใช้ประกอบการเรียนการสอนในชั้นเรียนเท่านั้น มิได้มีวัตถุประสงค์ในเชิงพานิชย์ ผู้เขียนกำหนดเนื้อหาออกเป็น 8 บท ประกอบด้วย

ส่วนที่ 1 พื้นฐานของวิศวกรรมความต้องการ (Fundamentals of Requirements Engineering)

บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับวิศวกรรมความต้องการ

บทที่ 2 ความหมายและบทบาทของผู้มีส่วนได้ส่วนเสีย

บทที่ 3 กระบวนการวิศวกรรมความต้องการ

ส่วนที่ 2 การพัฒนาความต้องการ (Requirements Development)

บทที่ 4 ความเข้าใจในขอบเขตและการสกัดความต้องการ

บทที่ 5 การประเมินความต้องการ

บทที่ 6 การจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์

บทที่ 7 การตรวจสอบความต้องการ และการประกันคุณภาพความต้องการ

ส่วนที่ 3 การจัดการความต้องการ (Requirements Management)

บทที่ 8 การจัดการความต้องการ

ผู้เขียนได้เรียบเรียงเอกสารคำสอนเล่มนี้จากประสบการณ์สอน การประชุมและเปลี่ยนความคิดเห็นกับคณาจารย์ในสาขาวิชา และผู้ประกอบการบริษัทผู้พัฒนาซอฟต์แวร์ รวมถึงตำราทั้งจากภาษาไทยและภาษาต่างประเทศ ในท้ายนี้ผู้เขียนขอขอบคุณทุกภาคส่วน โดยเฉพาะอย่างยิ่ง คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพาที่มีส่วนทำให้ผู้เขียนสามารถรวบรวม เรียบเรียงจนเอกสารเป็นรูปเล่มได้ และหวังเป็นอย่างยิ่งว่าเอกสารคำสอนเล่มนี้ จะเป็นประโยชน์ต่อการเรียนการสอน และการเรียนรู้ของผู้อ่านทุกท่าน

รายละเอียดของรายวิชา

ชื่อสถาบันอุดมศึกษา	มหาวิทยาลัยบูรพา
วิทยาเขต/คณะ/ภาควิชา	ชลบุรี / คณะวิทยาการสารสนเทศ / สาขาวิชาวิศวกรรมซอฟต์แวร์

หมวดที่ 1 ข้อมูลโดยทั่วไป

1.1 รหัสและชื่อวิชา

88823265 วิศวกรรมความต้องการและเอกสารโครงการซอฟต์แวร์

Software Requirements Engineering and Documentation

1.2 จำนวนหน่วยกิต

3 หน่วยกิต (2-2-5)

1.3 หลักสูตรและประเภทของรายวิชา

วิทยาศาสตรบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์

1.4 คำอธิบายรายวิชา

พื้นฐานเกี่ยวกับความต้องการของระบบ กระบวนการวิศวกรรมความต้องการ วิธีการ เครื่องมือและเทคนิคในการได้มาซึ่งข้อมูลและความต้องการของระบบ การจัดการและการจัดทำเอกสารความต้องการของระบบ การวิเคราะห์ความต้องการของระบบ การระบุข้อกำหนดซอฟต์แวร์ เทคนิคการตรวจสอบยืนยันการใช้ได้ รวมถึงเทคนิคการวิเคราะห์ความต้องการ การวิเคราะห์เป้าหมาย และการวิเคราะห์ยูสเคส มาตรฐานการจัดทำเอกสารความต้องการ การสืบค้นย้อนกลับได้ การจัดการความต้องการ การจัดการกับการเปลี่ยนแปลงความต้องการ

Software requirements fundamentals, requirements engineering process, methods, tools, and techniques for eliciting, organizing and documenting software requirements, requirement analysis, requirement specification, validation techniques, need, goal, and use case analysis, requirements documentation standards, traceability, requirements management, and handling requirements changes.

หมวดที่ 2 ผลการเรียนรู้ของรายวิชา

Program Learning Outcome : PLO	Course Learning Outcomes: CLO
PLO 1 : ปฏิบัติตนด้วยความซื่อสัตย์สุจริต ตรงต่อเวลา และเคารพกฎระเบียบขององค์กรและสังคม	CLO 1 : แสดงพฤติกรรมถึงความมีวินัย ซื่อสัตย์สุจริต ตรงต่อเวลา และเคารพกฎระเบียบขององค์กรและสังคม
PLO 2 : อธิบายหลักการและทฤษฎีที่สำคัญในเนื้อหาด้านวิศวกรรมซอฟต์แวร์ได้อย่างถูกต้องตามหลักวิชาการ	CLO 2 : อธิบายนิยามและความรู้เกี่ยวกับวิศวกรรมความต้องการได้อย่างถูกต้อง
PLO 3 : ประยุกต์ความรู้เพื่อแก้ไขโจทย์ปัญหาในการพัฒนาซอฟต์แวร์ ตามแนวทางของกระบวนการพัฒนาซอฟต์แวร์ด้วยวิถีโอเพนซอร์ส	CLO 3 : ใช้เครื่องมือโอเพนซอร์สในการวิเคราะห์และออกแบบซอฟต์แวร์ได้อย่างเหมาะสม
PLO 6 : มีประสบการณ์ในการวิเคราะห์และออกแบบซอฟต์แวร์ขนาดกลางและขนาดใหญ่ตามความต้องการของผู้ใช้ เพื่อแก้ไขปัญหาโจทย์การทำงานจริงจากสถานประกอบการ	CLO 4 : วิเคราะห์และออกแบบซอฟต์แวร์ตามความต้องการที่ได้รับมอบหมาย
PLO 8 : ทำงานในทีมพัฒนาซอฟต์แวร์ทั้งในบทบาทของผู้นำและผู้ตาม เพื่อให้บรรลุเป้าหมายของการพัฒนาซอฟต์แวร์	CLO 5 : กำหนดเป้าหมายและแผนการดำเนินงานของมกุลได้
PLO 9 : สื่อสารด้วยวิธีการเขียนหรือปากเปล่าที่เกี่ยวข้องกับวิศวกรรมซอฟต์แวร์ รวมถึงเลือกใช้เครื่องมือและรูปแบบการนำเสนอที่เหมาะสมกับสถานการณ์	CLO 6 : เขียนเอกสารข้อกำหนดความต้องการซอฟต์แวร์ได้อย่างถูกต้อง CLO 7 : สาธิตการใช้งานซอฟต์แวร์ที่ได้รับผิดชอบได้

หมวดที่ 3 กำหนดการสอน และแผนการสอน

สัปดาห์ที่	ชื่อเรื่องที่สอน	ผลการเรียนรู้ของบทเรียน (LLO)	สอดคล้องกับ CLO	วิธีการประสบการณ์การเรียนรู้	สื่อการเรียนรู้/โสตทัศนูปกรณ์	การประเมินผลการเรียนรู้
1	ภาพรวมของรายวิชา • แนะนำรายวิชา • ขอบเขตเนื้อหาและแผนการเรียน • ข้อตกลงในการเรียนการสอน บทที่ 1 นิยามและพื้นฐานเกี่ยวกับความต้องการ	• ผู้เรียนเข้าใจภาพรวมรายวิชา • ผู้เรียนทราบแผนการเรียนในรายวิชาตลอดภาคการศึกษา • ผู้เรียนทราบข้อตกลงในการเรียนในรายวิชา • ผู้เรียนอธิบายความรู้พื้นฐานเกี่ยวกับวิศวกรรมความต้องการได้ • ผู้เรียนอธิบายเกี่ยวกับประเภทความต้องการได้ • ผู้เรียนอธิบายกระบวนการต่างๆ ของวิศวกรรมความต้องการได้	1, 2, 3	• บรรยาย • บรรยาย • ยกตัวอย่าง • นิสิตร่วมอภิปรายแสดงความคิดเห็น	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย • คะแนนการสอบกลางภาค • คะแนนการถาม-ตอบ
2 – 3	บทที่ 1 นิยามและพื้นฐานเกี่ยวกับความต้องการ	• ผู้เรียนอธิบายความสัมพันธ์ของวิศวกรรมความต้องการกับกิจกรรมอื่นๆ ได้ • ผู้เรียนอธิบายความจำเป็นในการทำวิศวกรรมความต้องการ ของโครงการพัฒนาซอฟต์แวร์ได้	1, 2, 3	• บรรยาย • ยกตัวอย่าง • นิสิตร่วมอภิปรายแสดงความคิดเห็น •	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย • คะแนนการสอบกลางภาค • คะแนนการถาม-ตอบ
4	บทที่ 2 ความหมายและบทบาทของผู้มีส่วนได้ส่วนเสีย	• ผู้เรียนอธิบายความหมายของ Stakeholder ได้	1, 2	• บรรยาย • นิสิตร่วมอภิปรายแสดงความคิดเห็น	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย

สัปดาห์ที่	ชื่อเรื่องที่สอน	ผลการเรียนรู้ของบทเรียน (LLO)	สอดคล้องกับ CLO	วิธีการประสบการณ์การเรียนรู้	สื่อการเรียนรู้/โสตทัศนุปกรณ์	การประเมินผลการเรียนรู้
		• ผู้เรียนอธิบายชนิดต่างๆ ของ Stakeholder ได้ • ผู้เรียนตระหนักถึงความสำคัญของบทบาทและหน้าที่ของ Stakeholder ต่อวิศวกรรมความต้องการได้				• คะแนนการสอบกลางภาค • คะแนนการถาม-ตอบ
5	บทที่ 3 กระบวนการวิศวกรรมความต้องการ	• ผู้เรียนอธิบายกระบวนการวิศวกรรมความต้องการได้ • ผู้เรียนอธิบายลำดับของกระบวนการวิศวกรรมความต้องการ และสามารถบอกถึงความสำคัญของแต่ละลำดับได้	1, 2, 3	• บรรยาย • ยกตัวอย่าง • นิสิตร่วมอภิปรายแสดงความคิดเห็น •	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย • คะแนนการสอบกลางภาค • คะแนนการถาม-ตอบ
6 - 8	บทที่ 4 ความเข้าใจในขอบเขตและการสกัดความต้องการ	• ผู้เรียนอธิบายขอบเขตและการสกัดความต้องการได้ • ผู้เรียนอธิบายเกี่ยวกับการประเมินความต้องการได้	1, 2, 3	• บรรยาย • ยกตัวอย่าง • นิสิตร่วมอภิปรายแสดงความคิดเห็น •	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย • คะแนนการสอบกลางภาค • คะแนนการถาม-ตอบ
			สอบกลางภาค			
10	บทที่ 5 การประเมินความต้องการ	• ผู้เรียนอธิบายข้อกำหนดความต้องการและการจัดทำเอกสารความต้องการได้ • ผู้เรียนอธิบายวิธีการตรวจสอบความต้องการและการประกันคุณภาพความต้องการได้	1, 2, 3, 4, 5, 6	• บรรยาย • ยกตัวอย่าง • นิสิตร่วมอภิปรายแสดงความคิดเห็น	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย • คะแนนการสอบปลายภาค • คะแนนการถาม-ตอบ

สัปดาห์ที่	ชื่อเรื่องที่สอน	ผลการเรียนรู้ของบทเรียน (LLO)	สอดคล้องกับ CLO	วิธีการประสบการณ์การเรียนรู้	สื่อการเรียนรู้/โสตทัศนุปกรณ์	การประเมินผลการเรียนรู้
11	บทที่ 6 การจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์	• ผู้เรียนอธิบายเกี่ยวกับเอกสารความต้องการ • ผู้เรียนอธิบายองค์ประกอบต่าง ๆ ของเอกสารความต้องการ • ผู้เรียนอธิบายความรู้เกี่ยวกับแม่แบบการเขียนเอกสารความต้องการ • ผู้เรียนเขียนข้อกำหนดความต้องการแบบเป็นทางการ	1, 2, 3, 4, 5, 6	• บรรยาย • ยกตัวอย่าง • นิสิตร่วมอภิปรายแสดงความคิดเห็น	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน Template เอกสาร	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย • คะแนนการสอบปลายภาค • คะแนนการถาม-ตอบ
12	บทที่ 7 การตรวจสอบความต้องการและการประกันคุณภาพความต้องการ	• ผู้เรียนอธิบายเกี่ยวกับการทบทวน และตรวจสอบความต้องการได้ • ผู้เรียนตรวจสอบความต้องการได้ • ผู้เรียนทำซอฟต์แวร์ต้นแบบต้องการแบบเป็นทางการได้	1, 2, 3, 4, 5,	• บรรยาย • ยกตัวอย่าง • นิสิตร่วมอภิปรายแสดงความคิดเห็น	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย • คะแนนการสอบปลายภาค
13	บทที่ 8 การจัดการความต้องการ	• ผู้เรียนอธิบายความต้องการที่เสถียร กับความต้องการที่แปรเปลี่ยนได้ • ผู้เรียนระบุและจัดเก็บความต้องการได้	1, 2, 3, 4, 5, 6	• บรรยาย • ยกตัวอย่าง	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการเข้าชั้นเรียน • คะแนนการทดสอบย่อย • คะแนนการสอบ • คะแนนการถาม-ตอบ
14-16	การนำเสนอบทความวิชาการภาษาอังกฤษ โดย 10 มกุล	• ผู้เรียนแปลและเรียบเรียงบทความวิชาการภาษาอังกฤษ และเรียบเรียงข้อความ เพื่อนำเสนอ ในเวลา มกุลละ 15 นาที	7	• นิสิตนำเสนอบทความ • นิสิตร่วมอภิปรายแสดงความคิดเห็น	• แฟ้มดิจิทัลนำเสนอ • ระบบมัลติมีเดีย • เอกสารคำสอน	• คะแนนการนำเสนอ
17	**สอบปลายภาค**					

สารบัญ

สารบัญ

สารบัญรูป

สารบัญรูป

สารบัญตาราง

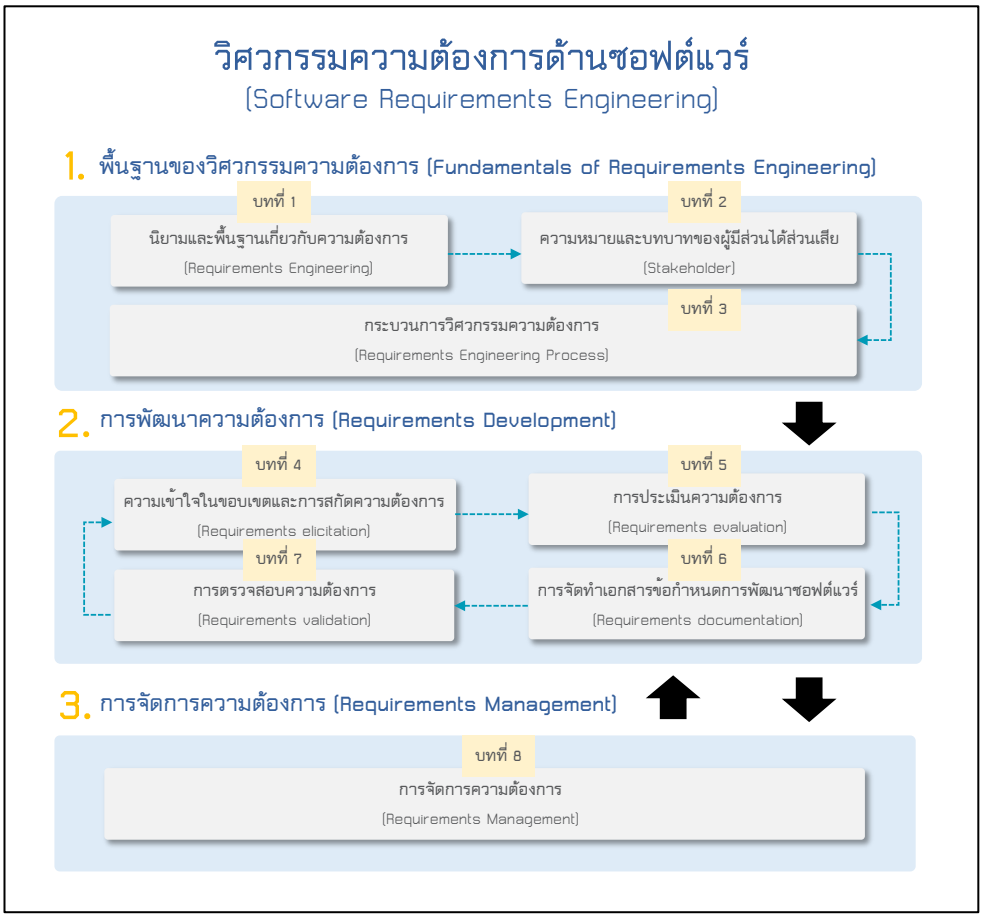
วิธีการใช้งานเอกสารคำสอนฉบับนี้

เอกสารคำสอนฉบับนี้ถูกออกแบบและเรียบเรียงให้ผู้ที่สนใจศึกษาค้นคว้า เข้าใจเนื้อหาอย่างเป็นลำดับตามกระบวนการวิศวกรรมความต้องการ แบ่งออกเป็น 3 ส่วนหลัก ได้แก่

ส่วนที่ 1) พื้นฐานของวิศวกรรมความต้องการ อธิบายเกี่ยวกับความหมายของวิศวกรรมความต้องการ คำจำกัดความที่เกี่ยวข้อง ประเภทของความต้องการ ความหมายของผู้มีส่วนได้ส่วนเสีย การวิเคราะห์ผู้มีส่วนได้ส่วนเสีย การมีปฏิสัมพันธ์กับผู้มีส่วนได้ส่วนเสีย และกระบวนการต่างๆ ที่เกี่ยวข้องกับวิศวกรรมความต้องการ เป็นต้น โดยครอบคลุมเนื้อหาของเอกสารทั้งหมดจำนวน 3 บท ได้แก่ บทที่ 1 ถึงบทที่ 3

ส่วนที่ 2) การพัฒนาความต้องการ เนื้อหาในส่วนนี้ถือเป็นหัวใจหลักของการทำวิศวกรรมความต้องการ กล่าวถึงกระบวนการในการเก็บรวมรวมและทำความเข้าใจความต้องการ การวิเคราะห์และประเมินความต้องการ การจัดทำเอกสารความต้องการ และการตรวจสอบความต้องการ ครอบคลุมเนื้อหาจำนวน 4 บท ได้แก่ บทที่ 4 ถึงบทที่ 7

และส่วนที่ 3) การจัดการความต้องการ เนื้อหาอธิบายเกี่ยวกับการจัดการกับการเปลี่ยนแปลงของความต้องการ และการติดตามความต้องการ เป็นต้น คือ บทที่ 8 ดังแสดงในภาพที่ 0 วิศวกรรมความต้องการด้านซอฟต์แวร์



ภาพที่ 0 วิศวกรรมความต้องการด้านซอฟต์แวร์

ส่วนที่ 1 พื้นฐานของวิศวกรรมความต้องการ
(Fundamentals of Requirements Engineering)

การเรียนรู้ในส่วนนี้

บทที่ 1 นิยามและพื้นฐานเกี่ยวกับความต้องการ (Requirements Engineering)

- คำจำกัดความของที่เกี่ยวข้อง
- ประเภทของความต้องการ
- คนที่เกี่ยวข้อง และกระบวนการวิศวกรรมความต้องการ
- ชนิดของโครงการซอฟต์แวร์

บทที่ 2 ความหมายและบทบาทของผู้มีส่วนได้ส่วนเสีย (Stakeholders)

- ความหมายของ Stakeholders
- การวิเคราะห์ Stakeholders
- แนววิธีในการวิเคราะห์ Stakeholders
- แผนภาพยูสเคส

บทที่ 3 กระบวนการวิศวกรรมความต้องการ (Requirements Engineering Process)

- กระบวนการพัฒนาความต้องการ (Requirements Development)
- กระบวนการจัดการความต้องการ (Requirements Management)

แผนบริหารการสอน
บทที่ 1 นิยามและพื้นฐานเกี่ยวกับความต้องการ

เนื้อหา

1. คำจำกัดความของที่เกี่ยวข้อง
2. ประเภทของความต้องการ
3. คนที่เกี่ยวข้อง และกระบวนการวิศวกรรมความต้องการ
4. ชนิดของโครงการซอฟต์แวร์

ผลลัพธ์การเรียนรู้รายบทเรียน

1. ผู้เรียนสามารถอธิบายความรู้พื้นฐานเกี่ยวกับวิศวกรรมความต้องการได้
2. ผู้เรียนสามารถอธิบายเกี่ยวกับประเภทความต้องการได้
3. ผู้เรียนสามารถอธิบายกระบวนการต่างๆ ของวิศวกรรมความต้องการได้
4. ผู้เรียนสามารถอธิบายความสัมพันธ์ของวิศวกรรมความต้องการกับกิจกรรมอื่นๆ ได้
5. ผู้เรียนสามารถอธิบายความจำเป็นในการทำวิศวกรรมความต้องการ ของโครงการพัฒนาซอฟต์แวร์ได้

วิธีการสอน

1. บรรยาย
2. ยกตัวอย่างระบบ โดยใช้กรณีตัวอย่างระบบที่เป็นที่รู้จัก
3. นิสิตร่วมอภิปรายแสดงความคิดเห็น

สื่อการสอน

1. แฟ้มดิจิทัลนำเสนอ
2. ระบบมัลติมีเดีย เช่น คอมพิวเตอร์ เครื่องเสียง เครื่องฉายภาพ
3. เอกสารประกอบคำสอน

การวัดและการประเมินผล

1. คะแนนการเข้าชั้นเรียน
2. คะแนนแบบฝึกหัด
3. คะแนนการอภิปรายของนิสิต

บทที่ 1 พื้นฐานเกี่ยวกับความต้องการ

วิศวกรรมความต้องการ (Requirements Engineering) เป็นกระบวนการ (Process) ของการสกัดความต้องการ (Elicit/Gather/Acquire) การวิเคราะห์ความต้องการ (Analyze/Evaluate) จัดทำเอกสาร (Documentation) ตลอดจนจัดการการเปลี่ยนแปลงของความต้องการ (Change) ที่ซอฟต์แวร์ควรตอบสนองผู้ใช้งาน เป็นกระบวนการส่วนแรก (Upstream) ที่สำคัญของกระบวนการพัฒนาซอฟต์แวร์ เนื่องจากช่วยให้แน่ใจว่าซอฟต์แวร์ที่ได้จะตอบสนองความต้องการของผู้ใช้งานและผู้มีส่วนได้ส่วนเสีย (Stakehoders) อย่างถูกต้องและเหมาะสม ซึ่งในบทที่ 1 จะอธิบายเกี่ยวกับ คำจำกัดความของที่เกี่ยวข้อง ประเภทของความต้องการ คนที่เกี่ยวข้อง และกระบวนการวิศวกรรมความต้องการ และชนิดของโครงการซอฟต์แวร์

ในปี ค.ศ. 2017 สถาบันบริหารโครงการ (Project Management Institute) รายงานสาเหตุหลักของความล้มเหลวในโครงการพัฒนาซอฟต์แวร์ ดังแสดงในตารางที่ 1.1 พบว่าในขณะที่หลายคนมีความกังวลเกี่ยวกับปัญหาด้านเทคนิคจะทำให้การพัฒนาซอฟต์แวร์เกิดความล้มเหลว แต่ในความเป็นจริงแล้วเกือบครึ่งของสาเหตุหลักทั้งหมดมีส่วนเกี่ยวข้องและเชื่อมโยงกับการทำวิศวกรรมความต้องการ (ลำดับที่ 1, 5, 7) ดังนั้นวิศวกรรมความต้องการจึงยังคงเป็นเรื่องที่วิศวกรต้องให้ความสำคัญเป็นอย่างมาก ถึงแม้ว่าในปัจจุบันจะมีรูปแบบการพัฒนาซอฟต์แวร์ที่หลากหลายและตอบสนองต่อการเปลี่ยนแปลงความต้องการ เช่น การพัฒนาแอปพลิเคชันแบบรวดเร็ว (Rapid Application Development) และสกรัม (Scrum) เป็นต้น

ตารางที่ 1.1 สาเหตุหลักของความล้มเหลวในโครงการพัฒนาซอฟต์แวร์

#	สาเหตุหลักความล้มเหลว
1.	**เป้าหมายและความต้องการที่ไม่ชัดเจน** (Lack of clear goals and requirements)
2.	**การจัดการโครงการที่ไม่ดี** (Poor project management)
3.	**ขาดแคลนทรัพยากร** (Lack of resources)
4.	**ปัญหาทางเทคนิค** (Technical challenges)
5.	**การเปลี่ยนแปลงความต้องการ** (Changes in requirements)
6.	**การสื่อสารที่ไม่ดี** (Poor communication)
7.	**ขาดการมีส่วนร่วมของได้ส่วนเสีย** (Lack of user involvement)

เพื่อให้มั่นใจว่าซอฟต์แวร์จะทำงานได้อย่างถูกต้อง สอดคล้องกับความต้องการของผู้ใช้งานและผู้มีส่วนได้ส่วนเสีย ต้องทำความเข้าใจว่าปัญหาของผู้ใช้งานคืออะไร ในหมู่คนที่เกี่ยวข้องกับปัญหานั้นๆ และนิยามปัญหาที่ต้องการแก้ไขให้เข้าใจตรงกัน โดยเนื้อหาในเอกสารฉบับนี้มีคำจำกัดความที่เกี่ยวข้อง ดังตารางที่ 1.2

1.1 คำจำกัดความของที่เกี่ยวข้อง (Key Definitions)

ตารางที่ 1.2 Key Definitions

#	คำนิยาม	ความหมาย
1.	**ความต้องการด้านซอฟต์แวร์** (Software requirements)	**1. IEEE Standard for Software Requirements Specification (IEEE Std 830-1998)** เป็นกระบวนการกำหนดความต้องการสำหรับระบบซอฟต์แวร์ รวมถึงฟังก์ชันที่ระบบควรดำเนินการภายใต้ข้อจำกัดต่างๆ โดยเกี่ยวข้องกับการระบุผู้มีส่วนได้ส่วนเสียในระบบ การกำหนดความต้องการและเป้าหมายของการพัฒนาระบบ ซึ่งเป้าหมายของความต้องการด้านซอฟต์แวร์คือการสร้างเอกสารข้อกำหนดความต้องการที่ชัดเจน รัดกุม และสมบูรณ์ ซึ่งสามารถใช้เป็นแนวทาในการออกแบบและพัฒนาซอฟต์แวร์ได้ **2. IEEE 1220 Application and Management of the Systems Engineering** ข้อมูลที่อธิบายเกี่ยวกับการทำงานของซอฟต์แวร์ รวมถึงข้อจำกัดต่างๆ มักอยู่ในรูปแบบเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์ (Software Requirement Specification Document: SRSD) ซึ่งชี้แจงเกี่ยวกับการทำงานของซอฟต์แวร์ที่กำลังจะพัฒนา และมีข้อกำหนดหรือเงื่อนไขในการทำงานอย่างไร โดยข้อกำหนดที่ถูกเขียนลงในเอกสารฉบับนี้จะสะท้อนความต้องการของผู้มีส่วนได้ส่วนเสียของซอฟต์แวร์
2.	**ความต้องการระดับผู้ใช้** (User requirements)	ความต้องการของผู้ใช้งานที่อธิบายเกี่ยวกับฟังก์ชันการทำงานของระบบ อาจอยู่ในแบบของข้อความบรรยาย หรือแผนภาพที่เข้าใจง่าย
3.	**ความต้องการระดับระบบ** (System requirements)	ข้อกำหนดทางเทคนิคที่กล่าวถึงการทำงานของระบบซอฟต์แวร์อย่างละเอียด มักเรียกว่า “System specification” หรือ “Function specification”
4.	**การสกัดความต้องการ** (Requirements elicitation)	กระบวนการรวบรวมความต้องการสำหรับซอฟต์แวร์ ซึ่งเกี่ยวข้องกับการระบุผู้มีส่วนได้ส่วนเสียในซอฟต์แวร์ การทำความเข้าใจความต้องการ และเข้าใจเป้าหมายของซอฟต์แวร์
5.	**การประเมินความต้องการ** (Requirements evaluation)	กระบวนการประเมินคุณภาพและความเหมาะสมของข้อกำหนดความต้องการสำหรับพัฒนาซอฟต์แวร์ เกี่ยวข้องกับการทบทวนข้อกำหนดความต้องการ เพื่อให้แน่ใจว่าครบถ้วน สอดคล้อง ตรวจสอบย้อนกลับ และตรวจสอบได้ และตอบสนองความต้องการและเป้าหมายของผู้ใช้งานและผู้มีส่วนได้ส่วนเสีย อาจมีการเจรจาต่อรองหรือปรึกษาหารือกับผู้ใช้งานและผู้มีส่วนได้ส่วนเสียหากความต้องการมีข้อขัดแย้งกันหรือมีความไม่สอดคล้องกับปัจจัยอื่นๆ

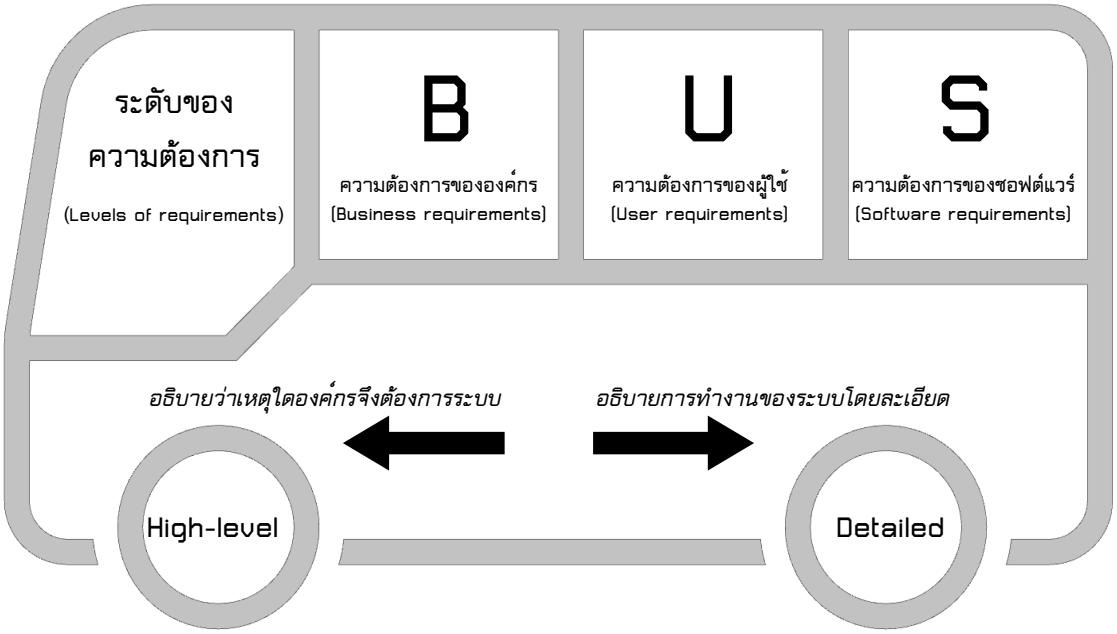
#	คำนิยาม	ความหมาย
6.	**การจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์** (Requirements specification/ Requirements documentation)	กระบวนการจัดทำเอกสารข้อกำหนดความต้องการสำหรับใช้ในการพัฒนาซอฟต์แวร์ เกี่ยวข้องกับการรวบรวมและจัดระเบียบข้อกำหนดความต้องการในลักษณะที่ชัดเจน รัดกุม และสมบูรณ์ เพื่อให้สามารถใช้เป็นแนวทางในการออกแบบและพัฒนาซอฟต์แวร์ได้
7.	**การตรวจสอบความต้องการ** (Requirements validation)	กระบวนการตรวจสอบว่าเอกสารข้อกำหนดความต้องการสำหรับใช้ในการพัฒนาซอฟต์แวร์ ถูกต้องและครบถ้วนตามความต้องการของผู้มีส่วนได้ส่วนเสีย
8.	**ผู้มีส่วนได้ส่วนเสีย** (Stakehoder)	คน กลุ่มคน อุปกรณ์หรือซอฟต์แวร์อื่นๆ รวมหมายถึงระเบียบวิธีปฏิบัติ กฎหมายต่างๆ ที่เกี่ยวข้อง ที่มีอิทธิพล หรือมีสิทธิ หรือมีผลกระทบ ในการกำหนดความต้องการต่อระบบ (ไม่ว่าทางตรงหรือทางอ้อม)
9.	**การทำงานที่พบในปัจจุบัน** (The system-as-is)	ระบบที่มีอยู่ก่อนที่จะนำระบบคอมพิวเตอร์/ซอฟต์แวร์ที่ได้พัฒนามาใช้งาน ในกรณีที่ไม่มีการใช้ระบบเดิมอยู่ อาจหมายรวมถึงการทำงานด้วยมือ (Manual) ก่อนมีระบบ
10.	**การทำงานที่เกิดขึ้นใหม่ในการพัฒนาระบบใหม่** (The system-to-be)	ระบบที่คาดว่าจะได้รับ หลังจากนำระบบคอมพิวเตอร์/ซอฟต์แวร์ ที่พัฒนาขึ้นมาตามความต้องการ และนำมาใช้งานได้สำเร็จ
11.	**เอกสารความต้องการ** (Requiremnts Document : RD)	เอกสารที่ระบุข้อกำหนดความต้องการสำหรับใช้ในการพัฒนาซอฟต์แวร์ โดยจะรวบรวมและจัดระเบียบข้อกำหนดในลักษณะที่ชัดเจน รัดกุม และครบถ้วน เพื่อให้สามารถใช้เป็นแนวทางในการออกแบบและพัฒนาซอฟต์แวร์ได้
12.	**วัตถุประสงค์ของระบบ** (System objectives)	เป้าหมายหรือผลลัพธ์ที่คาดว่าจะบรรลุได้เมื่อนำระบบมาใช้งาน อาจเป็นการกำหนดสิ่งที่ระบบควรทำก็ได้
13.	**หน้าที่ต่างๆ ของระบบ / ขอบเขตการทำงานของระบบ** (Functionalities/Services)	คุณสมบัติ/หน้าที่/การให้บริการของซอฟต์แวร์ที่ทำได้
14.	**ข้อจำกัด/เงื่อนไข** (Constraints)	ข้อจำกัด เงื่อนไข หรือคุณภาพต่างๆ ที่เกี่ยวข้องกับการทำงานของซอฟต์แวร์โดยตรง หรืออาจเกี่ยวข้องในขณะการพัฒนาซอฟต์แวร์ หรือเกี่ยวข้องกับผู้มีส่วนได้ส่วนเสีย
15.	**ความต้องการแบบฟังก์ชัน** (Functional requirements)	หน้าที่ต่างๆ ของระบบ/ขอบเขตการทำงานของระบบ ที่ System-to-be ควรจะมี (ถ้าไม่มีระบบจะทำงานไม่ได้ หรือไม่ถูกต้อง หรือไม่ครบถ้วน)
16.	**ความต้องการแบบไม่เป็นฟังก์ชัน** (Non-Functional requirements)	เป็นส่วนที่ไม่เกี่ยวข้องกับการทำงานของระบบโดยตรง แต่จะส่งเสริมให้ระบบมีการทำงานที่มีประสิทธิภาพมากขึ้น หรืออาจเป็นการระบุสิ่งที่ผู้พัฒนาสัญญาว่าจะทำให้ หรืออาจเกี่ยวข้องกับข้อจำกัด/เงื่อนไขต่างๆ ที่มีต่อระบบและระหว่างการพัฒนาระบบ
17.	**ความรู้เฉพาะทาง (Domain knowledge)**	ในบริบทของการพัฒนาซอฟต์แวร์หมายถึงความรู้และความเข้าใจเฉพาะที่จะใช้ซอฟต์แวร์ ตัวอย่างเช่น นักพัฒนาซอฟต์แวร์ที่ทำงานเกี่ยวกับระบบการจัดการเวชระเบียนจะต้องมีความเข้าใจอย่างลึกซึ้งเกี่ยวกับอุตสาหกรรมการแพทย์ ตลอดจนข้อกำหนดและความจำเป็นเฉพาะของบุคลากรทางการแพทย์

1.2 ประเภทของความต้องการ

ความต้องการถูกแบ่งออกเป็นหลายประเภท ในส่วนนี้จะจำแนกประเภทของความต้องการออกเป็น 2 มิติ ได้แก่ 1. ระดับของความต้องการ และ 2. ความต้องการของระบบ ดังต่อไปนี้

1.2.1 จำแนกประเภทของความต้องการตามระดับของความต้องการ (Levels of requirements)

ในภาพที่ 1.1 ระดับของความต้องการแบ่งออกเป็น 3 ประเภท ได้แก่ 1. ความต้องการขององค์กร (Business requirements) 2. ความต้องการของผู้ใช้ (User requirements) และ 3. ความต้องการของระบบ (System requirements) โดยความต้องการขององค์กรจะมีลักษณะที่ผู้มีส่วนได้ส่วนเสีย (Stakeholders) ทุกกลุ่ม สามารถทำความเข้าใจได้ง่าย (High Level) แต่ในทางตรงกันข้าม ความต้องการของระบบมีลักษณะที่ค่อนข้างซับซ้อน เนื่องจากต้องอธิบายรายละเอียดของระบบอย่างครบถ้วน (Detailed) เหมาะสำหรับทีมพัฒนาซอฟต์แวร์



ภาพที่ 1.1 ระดับของความต้องการ

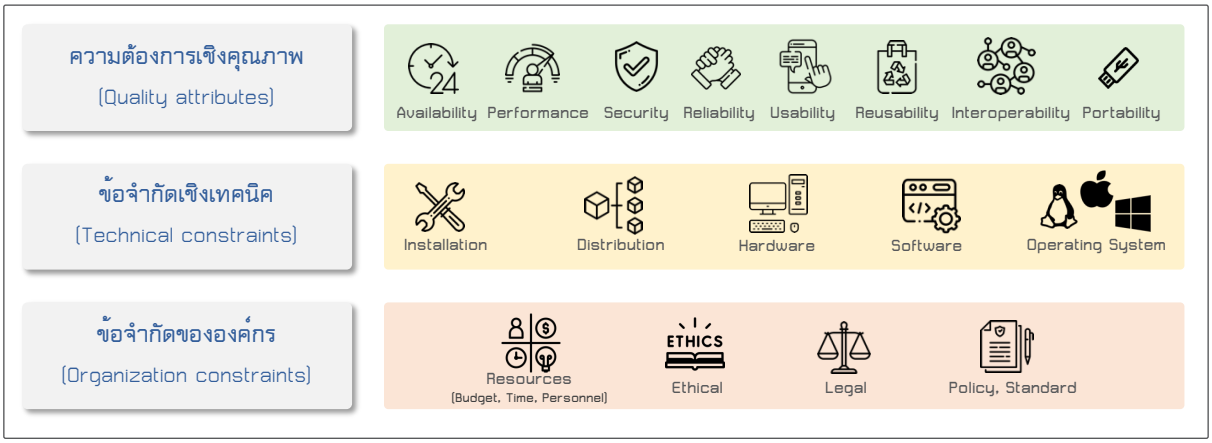
1. **ความต้องการขององค์กร/ข้อกำหนดการทางธุรกิจ (Business requirements)** - อธิบายถึงเป้าหมาย วัตถุประสงค์ และความต้องการของธุรกิจหรือองค์กรที่จะใช้ระบบซอฟต์แวร์ มักสอดคล้องกับการระบุกระบวนการทางธุรกิจ (Business processes) ที่จะนำซอฟต์แวร์มาช่วยสนับสนุน และประโยชน์ที่ซอฟต์แวร์ควรมอบให้กับธุรกิจ ความต้องการขององค์กรมักจะมาจากผู้มีส่วนได้ส่วนเสียของโครงการที่เป็นเจ้าของธุรกิจ (Business owners) ผู้จัดการ (Managers) และผู้ใช้ (End-users)

2. **ความต้องการของผู้ใช้ (User requirements)** – อธิบายถึงความต้องการ (User Needs) และความคาดหวัง (User Expectations/Want) ของผู้ใช้ (End-users) ที่ผู้ใช้ต้องสามารถดำเนินการผ่านระบบซอฟต์แวร์นั้นๆ จากมุมมองของผู้ใช้
    a. **User Needs** หมายถึง ฟังก์ชันการทำงานที่มีความจำเป็นต่อผู้ใช้ โดยทั่วไปแล้วความต้องการเหล่านี้จะถูกระบุโดยการสำรวจ (Surveys) หรือการสัมภาษณ์ (interviews)
    b. **User Expectations/Want** หมายถึง ระดับคุณภาพ (Level of quality) ที่ผู้ใช้คาดหวังจากระบบซอฟต์แวร์ ความคาดหวังเหล่านี้มักจะได้รับอิทธิพลจากประสบการณ์ก่อนหน้านี้ของผู้ใช้กับระบบซอฟต์แวร์ที่คล้ายคลึงกัน
3. **ความต้องการของซอฟต์แวร์ (Software requirements)** – อธิบายลักษณะทางเทคนิคและการทำงานของระบบซอฟต์แวร์อย่างละเอียด โดยทั่วไปมักจะถูกกำหนดโดยทีมพัฒนาซอฟต์แวร์และใช้เพื่อเป็นแนวทางในการพัฒนาระบบซอฟต์แวร์ให้สามารถทำงานได้ตามความต้องการขององค์กร (Business requirements) และความต้องการของผู้ใช้ (User requirements)

1.2.2 จำแนกประเภทของความต้องการตามความต้องการของระบบ (System requirements)

1. **ความต้องการเชิงฟังก์ชัน (Functional requirements)** - อธิบายเกี่ยวกับการทำงานของระบบโดยตรง เป็นการระบุความสามารถการทำงานของระบบ ซึ่งอาจเกี่ยวข้องกับลำดับขั้นตอนการทำงานของระบบ เงื่อนไข และข้อจำกัดต่างๆ ตัวอย่างเช่น หาก ณ ขณะนี้ ทีมกำลังพัฒนาเว็ปไซต์ e-Commerce สำหรับขายสินค้าออนไลน์ ดังนั้น ความต้องการเชิงฟังก์ชันที่เกี่ยวข้องอาจจะมี ฟังก์ชันที่ให้ลูกค้าเพิ่มสินค้าลงในตะกร้าสินค้าได้ หรือเรียกว่า “ฟังก์ชันเพิ่มสินค้าในตะกร้าสินค้า” อย่างไรก็ตาม ความต้องการนี้สามารถแยกย่อยออกเป็นความต้องการที่มีรายละเอียดมากขึ้น เช่น
    a. ระบบสามารถแสดงสินค้าที่มีอยู่พร้อมราคาและคำอธิบายสินค้า
    b. ระบบสามารถให้ลูกค้าเลือกจำนวนสินค้าแต่ละรายการ
    c. ระบบสามารถคำนวณราคารวมของสินค้าในตะกร้าสินค้า
    d. ระบบสามารถให้ลูกค้าแก้ไขหรือลบสินค้าออกจากตะกร้าสินค้า
2. **ความต้องการที่ไม่เป็นเชิงฟังก์ชัน (Non-functional requirements)** – อธิบายถึงความต้องการที่ไม่เกี่ยวกับการทำงานของระบบซอฟต์แวร์โดยตรง แต่เป็นส่วนเสริมการทำงานให้มีประสิทธิภาพมากขึ้น ตัวอย่างเช่น เว็ปไซต์ e-Commerce สำหรับขายสินค้าออนไลน์ อาจเป็นไปได้ที่จะมีผู้เข้าใช้งานเป็นจำนวนมาก ดังนั้น “เว็ปไซต์ควรรองรับการเข้าใช้งานของลูกค้าในปริมาณมากได้ โดยไม่ส่งผลทำให้ระบบเกิดความล่าช้าหรือเสียหายใดๆ กับตัวระบบ”

บางครั้งความต้องการที่ไม่เป็นเชิงฟังก์ชันอาจหมายรวมถึง คุณลักษณะด้านคุณภาพ (Quality attributes) หรือข้อจำกัดเชิงเทคนิค (Technical constraints) หรือข้อจำกัดขององค์กร (Organization constraints) ดังภาพที่ 1-2 โดยรายละเอียดแสดงดังตาราง 1.3 ถึงตารางที่ 1.4



ภาพที่ 1.2 ความต้องการที่ไม่เป็นเชิงฟังก์ชัน

ตารางที่ 1.3 ความต้องการที่ไม่เป็นเชิงฟังก์ชัน : คุณลักษณะด้านคุณภาพ (Quality attributes)

#	คำนิยาม	ความหมาย
1.	Availability	เกี่ยวข้องกับกรณีที่ข้อมูลใดๆ ในระบบถูกเรียกใช้ หรือถูกค้นหา ผ่านเมนูหนึ่งๆ ณ เวลาหนึ่งๆ ระบบจะต้องสามารถตอบสนองและให้บริการข้อมูลนั้นๆ แก่ผู้ใช้ได้ โดยผู้ใช้คนนั้นจะต้องมีสิทธิ์ในการเข้าถึงข้อมูล
2.	Performance	เกี่ยวข้องกับคุณภาพความต้องการในประเด็นเกี่ยวกับประสิทธิภาพการทำงานของระบบ ภายใต้เงื่อนไขที่กำหนด ทั้งในเรื่องของเวลาและทรัพยากรอื่นๆ
3.	Security	เกี่ยวกับความมั่นคงปลอดภัยของระบบ ป้องกันการคุกคามจากการเข้าถึงระบบซอฟต์แวร์โดยมิชอบ
4.	Reliability	เกี่ยวข้องกับกรณีที่ซอฟต์แวร์ถูกใช้งานหรือทำงานเป็นระยะเวลานาน ต้องทำให้ผู้ใช้ซอฟต์แวร์มั่นใจได้ว่าระบบดังกล่าวมีความคงทน แข็งแรง สามารถทำงานได้ตลอดเวลา
5.	Usability	เกี่ยวกับผู้ใช้สามารถเรียนรู้การใช้ได้ง่าย โดยที่ผู้ใช้สามารถใช้งานได้เอง สามารถจดจำการใช้งานได้ง่าย มีความพึงพอใจในการใช้งาน
6.	Reusability	เกี่ยวกับความสามารถในการนำกลับมาใช้ใหม่ได้
7.	Interoperability	เกี่ยวกับความสามารถในการทำงานเชื่อมต่อกับระบบอื่นได้
8.	Portability	เกี่ยวกับความสามารถที่จะใช้ซอฟต์แวร์ที่เขียนขึ้นสำหรับเครื่องมือชนิดหนึ่งกับเครื่องมืออีกชนิดหนึ่งได้

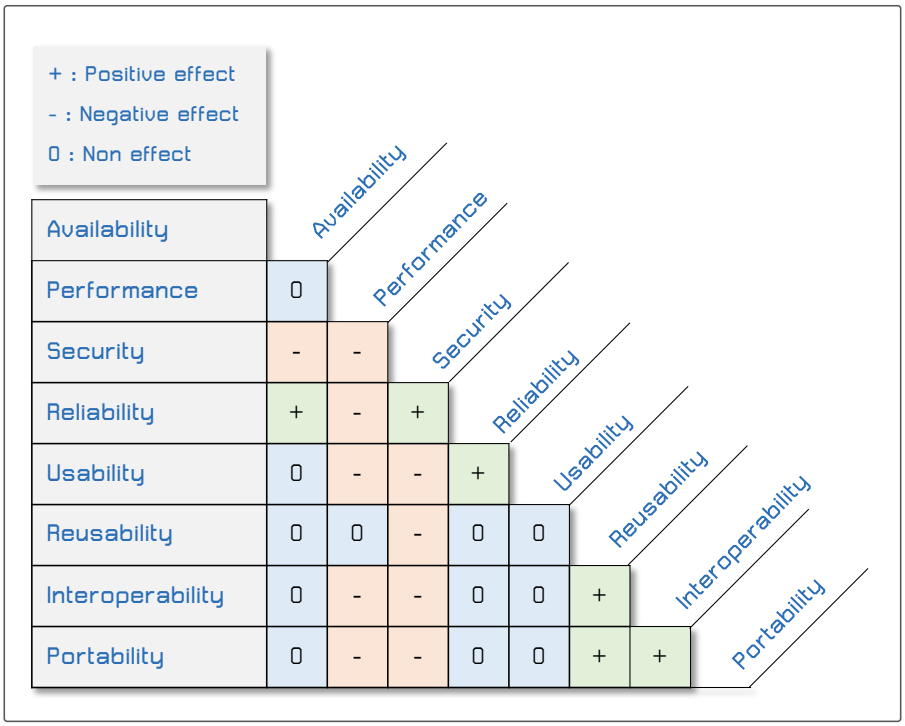
ตารางที่ 1.4 ความต้องการที่ไม่เป็นเชิงฟังก์ชัน : ข้อจำกัดเชิงเทคนิค (Technical constraints)

#	คำนิยาม	ความหมาย
1.	Installation	เกี่ยวกับการทำให้มั่นใจได้ว่าระบบจะสามารถทำงาน (Run) ได้และทำงานได้ตามเป้าหมายที่กำหนดไว้
2.	Distribution	เกี่ยวกับซอฟต์แวร์ที่สามารถทำงานแบบกระจายได้ (Distribution) สภาพแวดล้อมที่แตกต่างกันทางภูมิศาสตร์ (ติดตั้งในหลายพื้นที่ หรือหลาย Site งาน)
3.	Hardware	เกี่ยวกับข้อจำกัดในการทำงานประสานเชื่อมต่อกันระบบว่าระบบซอฟต์แวร์กับอุปกรณ์ต่างๆ
4.	Software	เกี่ยวกับข้อจำกัดที่ส่งผลกระทบต่อการออกแบบ การพัฒนา และการปรับใช้ระบบซอฟต์แวร์
5.	Operating System	เกี่ยวกับข้อจำกัดของระบบปฏิบัติการซอฟต์แวร์

ตารางที่ 1.5 ความต้องการที่ไม่เป็นเชิงฟังก์ชัน : ข้อจำกัดขององค์กร (Organization constraints)

#	คำนิยาม	ความหมาย
1.	Resources	เกี่ยวกับข้อจำกัดด้านงบประมาณ เวลา และบุคลากร
2.	Ethical	เกี่ยวกับข้อจำกัดด้านจริยธรรม
3.	Legal	เกี่ยวกับข้อจำกัดด้านกฎหมาย
4.	Policy, Standard	เกี่ยวกับข้อจำกัดด้านนโยบาย หรือมาตรฐานองค์กร

อย่างไรก็ตามความต้องการที่ไม่เป็นเชิงฟังก์ชันมีความสัมพันธ์ระหว่างกันที่หลากหลาย ทั้งส่งผลดีต่อกัน (Positive effect) ส่งผลขัดแย้งกัน (Negative effect) และไม่มีผลต่อกัน (Non effect) ดังแสดงในภาพที่ 1.3



ภาพที่ 1.3 ความสัมพันธ์ระหว่างความต้องการเชิงคุณภาพ

- **ส่งผลดีต่อกัน (Positive effect)** ตัวอย่างเช่น ความน่าเชื่อถือของระบบ (Reliability) และความสามารถในการใช้งาน (Usability) ระบบที่มีความสามารถในการใช้งานสูง โดยที่ผู้ใช้สามารถใช้งานได้เอง สามารถจดจำการใช้งานได้ง่าย มีความพึงพอใจในการใช้งาน ย่อมทำให้เกิดความน่าเชื่อถือเพิ่มมากขึ้น
- **ส่งผลขัดแย้งกัน (Negative effect)** ตัวอย่างเช่น ความมั่นคงปลอดภัย (Security) และความสามารถในการใช้งาน (Usability) ระบบที่มีความปลอดภัยสูงอาจใช้งานยาก โดยต้องมีการพิสูจน์ตัวตนหลายระดับ ในทางกลับกัน ระบบที่มีความสามารถในการใช้งานสูงอาจมีความปลอดภัยน้อย เนื่องจากอาจมีมาตรการรักษาความปลอดภัยน้อย เพื่อลดระยะเวลาในการเข้าใช้งานของผู้ใช้
- **ไม่มีผลต่อกัน (Non effect)** ตัวอย่างเช่น ความสามารถในการนำกลับมาใช้ใหม่ได้ (Reusability) และความสามารถในการใช้งาน (Usability) ถึงแม้ว่าระบบจะสามารถนำกลับมาใช้ใหม่ได้มากหรือน้อยเพียงใด ก็ไม่ส่งผลต่อความสามารถในการใช้งานระบบ

1.3 คนที่เกี่ยวข้อง และกระบวนการวิศวกรรมความต้องการ

ในกระบวนการวิศวกรรมความต้องการเรียกคน หรือกลุ่มคน หรือผู้ใดที่มีส่วนเกี่ยวข้องกับระบบที่จะพัฒนาขึ้นใหม่ (System-to-be) ว่า “ผู้มีส่วนได้ส่วนเสีย” (Stakeholders) โดยมีหน้าที่สำคัญคือกำหนดรูปร่าง (Shape) ของ System-to-be อาจครอบคลุมบุคคลดังนี้

1. เป็นผู้ตัดสินใจในเชิงกลยุทธ์
2. เป็นผู้จัดการของหน่วยงานหรือแผนกต่างๆ
3. เป็นผู้เชี่ยวชาญในการให้ข้อมูลที่เกี่ยวข้องกับระบบ (Domain expert)
4. เป็นผู้ดูแล และ/หรือเป็นผู้ใช้งาน System-to-be (Operators, End-users)
5. ทีมพัฒนา และ/หรือ ทีมรับจ้างช่วง (Subcontractor)
6. ผู้ออกใบรับรอง (Certificate Authorities)

นอกจาก Stakeholders จะเป็นคนหรือกลุ่มคนแล้ว ยังเป็นอุปกรณ์หรือซอฟต์แวร์อื่นๆ รวมหมายถึงระเบียบวิธีปฏิบัติ กฎหมายต่างๆ ที่เกี่ยวข้อง ที่มีอิทธิพล หรือมีสิทธิ หรือมีผลกระทบ ในการกำหนดความต้องการต่อระบบ (ไม่ว่าทางตรงหรือทางอ้อม)

โดยกระบวนการวิศวกรรม เป็นวิธีการจัดการที่เป็นระบบในการรวบรวม (Elicit/Gather/Acquire) เข้าใจ (Understand) วิเคราะห์ (Analyze) จัดทำเอกสาร (Document) ตรวจสอบความถูกต้อง (Validate) และจัดการความต้องการ (Manage) โดยกระบวนการความต้องการมักประกอบด้วยขั้นตอนต่อไปนี้

1.4 ชนิดของโครงการซอฟต์แวร์

1.4.1 โครงการชนิด Greenfield และ Brownfield

- **Greenfield project** หมายถึงซอฟต์แวร์ที่ถูกพัฒนาขึ้นมาใหม่
- **Brownfield project** หมายถึงซอฟต์แวร์ที่ถูกปรับปรุง เพิ่มเติม รวมกัน ของระบบ system-as-is ที่มีอยู่แล้ว

1.4.2 โครงการชนิด Customer-driven และ Market-driven

- **Customer-driven project** หมายถึงซอฟต์แวร์ที่ถูกพัฒนา โดยเกิดจากแรงผลักดัน/แรงกดดันจากลูกค้าขององค์กรนั้นๆ ว่าจำเป็นหรือถึงเวลาแล้วที่องค์กรนั้น ควรจะมีซอฟต์แวร์ใหม่
- **Market-driven project** หมายถึงซอฟต์แวร์ที่ถูกพัฒนา ภายใต้สภาวะแรงกดดัน หรือความต้องการจากส่วนแบ่งทางการตลาด (Market segment)

1.4.3 โครงการชนิด In-house และ Outsourced

- **In-house project** หมายถึงซอฟต์แวร์ที่ถูกพัฒนา โดยเจ้าหน้าที่หรือบุคลากรในองค์กรนั้นเอง เช่น จากศูนย์คอมพิวเตอร์ของบริษัทนั้นๆ

- **Outsourced project** หมายถึงซอฟต์แวร์ที่ถูกพัฒนา โดยองค์กรหรือบริษัทนั้นๆ ทำการจ้างหน่วยงานหรือบริษัท (Software house) จากภายนอก

1.4.4 โครงการชนิด Single-product และ Product-line

- **Single-product project** หมายถึง ซอฟต์แวร์หนึ่งถูกพัฒนาสำหรับจัดการเรื่องหนึ่งๆ หรือสำหรับลูกค้ากลุ่มหนึ่งๆ (Target customers)
- **Product-line project** หมายถึงซอฟต์แวร์ที่ถูกพัฒนาเป็นชุดของซอฟต์แวร์ (Product family) เพื่อใช้สำหรับจัดการงานที่ครอบคลุมงานหลากหลายแบบ

เอกสารอ้างอิง

1. A. Lamsweerde, "Requirements Engineering from System Goals to UML Models to Software Specifications (Chapter 1)", Wiley, 2009, ISBN 978-0-470-01270-3.
2. B. Berenbach, J. Kazmeier, D. J. Paulish, and A. Rudorfer, "Software & Systems Requirements Engineering in Practice (Chapter 1)", McGrawHill, 2009, ISBN 978-0-07-160547-2.
3. G. Kotonya and I. Sommerville, "Requirements Engineering: Processes and Techniques (Chapter 1)", John Wiley & Sons, 1997, ISBN 0-471-97208-8.
4. I. F. Alexander and R. Stevens, "Writing Better Requirements (Chapter 1)", Addison Wesley, 2002, ISBN 0-321-13163-0.
5. T. T. Barker, "Writing Software Documentation: A Task-Oriented Approach", Longman, 2002, ISBN 0-321-10328-9.

แบบฝึกหัด

โจทย์ : วิเคราะห์ความต้องการของ “ระบบขายตั๋วออนไลน์” และ “ระบบการลา” ที่กำหนดให้ และพิจารณาว่าเป็นความต้องการลักษณะใด พร้อมเติมคำตอบลงในตารางที่กำหนดให้

ข้อที่ 1 ระบบขายตั๋วออนไลน์

ข้อที่	ความต้องการ
1.	ผู้ใช้ต้องสามารถเรียกดูกิจกรรม (Event) จำแนกตามชื่อกิจกรรม วันที่จัด หรือสถานที่จัดงานได้
2.	การค้นหากิจกรรม (Event) ด้วยคำสำคัญ (Keywords) ก็ควรจะทำได้ด้วยเช่นกัน
3.	ผู้ใช้จะเลือก/ปรับเปลี่ยนราคาตั๋ว (Ticket price) และระบุจำนวนตั๋วที่ต้องการ
4.	ระบบควรแสดงหน้าจอการจองตั๋วภายใน 5 วินาที หลังจากที่ผู้ใช้เข้าสู่หน้าจอ
5.	ผู้ใช้ต้องสามารถจองที่นั่งและซื้อตั๋วได้ตลอดเวลา (24/7)
6.	เฉพาะลูกค้าที่ลงทะเบียนแล้วเท่านั้นถึงจะสามารถซื้อตั๋วได้
7.	ในทุกๆ วัน ระบบจะต้องสร้างรายงานสรุปยอดขาย (Daily reports of ticket sales) โดยรายงานดังกล่าวจะถูกเก็บไว้ในระบบเป็นเวลา 2 ปี ก่อนทำลายทิ้ง
8.	ในทุกๆ คืน (24.00 น.) ข้อมูลทั้งหมดภายในระบบจะได้รับการสำรองข้อมูล (Backup)
9.	ไม่ควรจองตั๋วซ้ำกับตั๋วที่เคยถูกจองไปก่อนหน้านี้แล้ว
10.	สามารถสร้างรายงานสรุปยอดขายเฉพาะกิจ (Ad hoc) ตามที่ผู้ใช้ระบุช่วงวันและเวลาในการดูรายงาน

ข้อที่ 2 ระบบการลา

01

เนื่องด้วย**ระเบียบสำนักนายกรัฐมนตรี** ว่าด้วยการลาของข้าราชการ พ.ศ. 2555 ซึ่งมีผลต่อการพิจารณาผลการปฏิบัติงานของบุคลากรภายในองค์กร หากไม่มีการบันทึกใบลาเป็นลายลักษณ์อักษร ส่งผลให้บุคลากรขาดงาน หรือละทิ้งการปฏิบัติราชการโดยไม่มีเหตุผลสมควร ถือเป็นความผิดทางวินัย

ด้วยเหตุนี้ทางสาขาวิชาวิศวกรรมซอฟต์แวร์ คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา จึงมีแนวคิดที่จะพัฒนา เพื่อช่วยในการบริหารจัดการข้อมูลการลา และ**ง่ายต่อการสืบค้นข้อมูล** 02 03 รวมถึง**ลดระยะเวลาในการอนุมัติการลา** ทั้งนี้เพื่อช่วยลดปริมาณการใช้กระดาษ ให้สอดคล้องกับ 04 แนวคิดของ**สำนักงานไร้กระดาษ** ซึ่งระบบการลาจะดำเนินการตั้งแต่เริ่มต้น**บันทึกการลา** 05 การอนุมัติการลาผ่านระบบ ตลอดจนการ**รายงานสถิติข้อมูลการลา** และนำข้อมูลไปใช้ประกอบการประเมินผลการปฏิบัติราชการ 06

บริบทที่โจทย์กำหนด

อักษรย่อ	ชื่อเต็ม	ความหมาย
F	Functional	ความต้องการหลัก (จำเป็นต้องทำ)
N	Non-Functional	ความต้องการรอง (ถ้าทำได้จะส่งผลดี)
S	Support	เหตุผลสนับสนุน (หลักการและเหตุผล)

ตารางคำตอบสำหรับ ระบบขายตั๋วออนไลน์

เลขกำกับ	F	N	S
01			
02			
03			
04			
05			
06			
07			
08			
09			
10			

ตารางคำตอบสำหรับ ระบบการลา

เลขกำกับ	F	N	S
01			
02			
03			
04			
05			
06			

แผนบริหารการสอน
บทที่ 2 ความหมายและบทบาทของผู้ที่ได้ประโยชน์/ผู้มีส่วนได้ส่วนเสีย

เนื้อหา

1. ความหมายของ Stakeholders
2. การวิเคราะห์ Stakeholders
3. แนววิธีในการวิเคราะห์ Stakeholders
4. แผนภาพยูสเคส (Use case diagram)

ผลลัพธ์การเรียนรู้รายบทเรียน

1. ผู้เรียนอธิบายความหมายของ Stakeholder ได้
2. ผู้เรียนอธิบายชนิดต่างๆ ของ Stakeholder ได้
3. ผู้เรียนตระหนักถึงความสำคัญของบทบาทและหน้าที่ของ Stakeholder ต่อวิศวกรรมความต้องการได้

วิธีการสอน

1. บรรยาย
2. ยกตัวอย่าง
3. นิสิตร่วมอภิปรายแสดงความคิดเห็น

สื่อการสอน

1. แฟ้มดิจิทัลนำเสนอ
2. ระบบมัลติมีเดีย
3. เอกสารประกอบการสอน

การวัดและการประเมินผล

1. คะแนนการเข้าชั้นเรียน
2. คะแนนการทดสอบย่อย
3. คะแนนการสอบกลางภาค
4. คะแนนการถาม-ตอบ

บทที่ 2 ความหมายและบทบาทของผู้ที่ได้ประโยชน์/ผู้มีส่วนได้ส่วนเสีย

ผู้มีส่วนได้ส่วนเสีย (Stakeholders) เป็นกลุ่มคนหรือองค์กรที่มีผลต่อกระบวนการพัฒนาซอฟต์แวร์และความสำเร็จของโครงการ ซึ่งมีความสำคัญที่ต้องให้ความสนใจและเข้าใจอย่างลึกซึ้ง เพื่อให้การวางแผนและการพัฒนาซอฟต์แวร์เป็นไปอย่างมีประสิทธิภาพ ในบทนี้จะกล่าวถึง ความหมายของ Stakeholders การวิเคราะห์ Stakeholders การจัดการปัญหาอุปสรรคในการได้มาขององค์ความรู้ การมีปฏิสัมพันธ์กับ Stakeholders และบทบาทและหน้าที่ของ Stakeholders ต่อวิศวกรรมความต้องการ

2.1 ความหมายของ Stakeholders

ผู้มีส่วนได้ส่วนเสีย (Stakeholders) หมายถึงคน (กลุ่มคน) อุปกรณ์หรือซอฟต์แวร์อื่นๆ รวมถึงระเบียบวิธีปฏิบัติ กฎหมายต่างๆ ที่เกี่ยวข้อง มีอิทธิพล หรือมีสิทธิ หรือมีผลกระทบ ในการกำหนดความต้องการ (ไม่ว่าทางตรงหรือทางอ้อม)

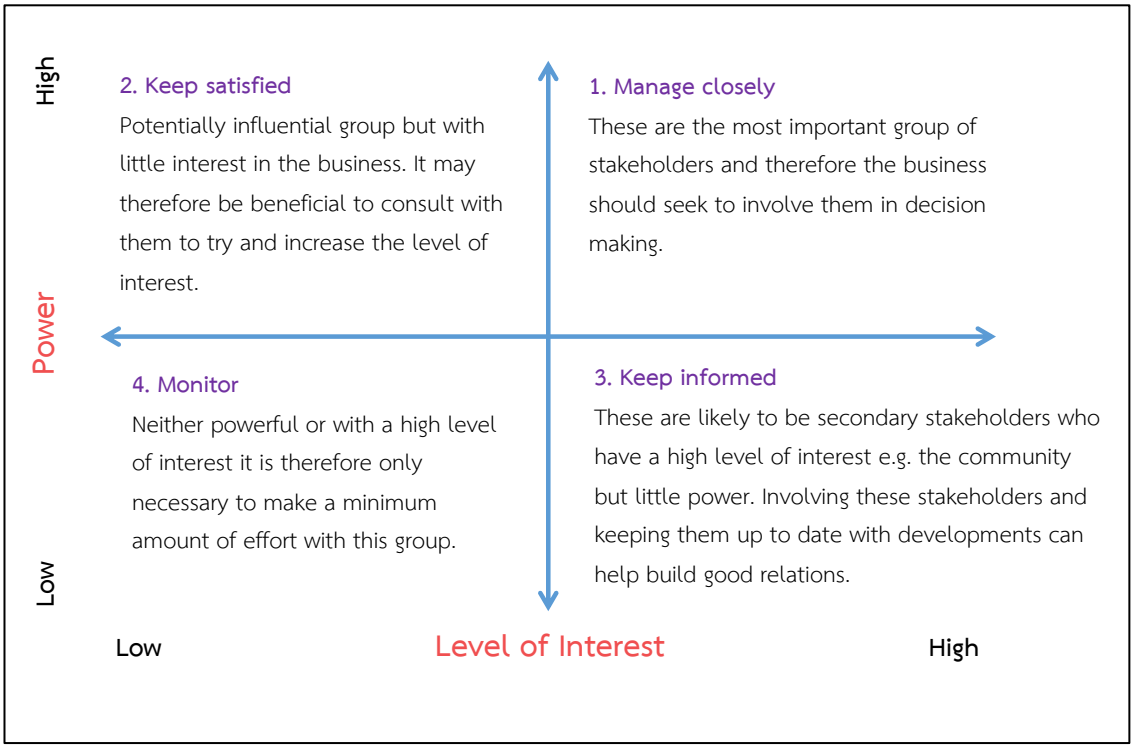
ภาพที่ 2.1 ผู้มีส่วนได้ส่วนเสีย (Stakeholders)

จากภาพที่ 2-1 ผู้มีส่วนได้ส่วนเสีย (Stakeholders) อาจเป็นผู้บริหารระดับสูง (CEO) ผู้ใช้งานระบบ (End user) ผู้เชี่ยวชาญ (Technician) นักพัฒนาระบบ (Developer) ผู้ดูแลระบบ (Sysadmin) นโยบาย/ข้อกำหนดต่างๆ (Policies/Regulations) ระบบเดิมหรือระบบต่างๆ ที่ให้บริการอยู่ (Legacy systems) การประกันคุณภาพซอฟต์แวร์ (Software quality assurance) หรือ ผู้ดำเนินการ (Operator)

2.2 การวิเคราะห์ Stakeholders

การวิเคราะห์ Stakeholders เป็นขั้นตอนสำคัญในกระบวนการพัฒนาซอฟต์แวร์ เป็นการรวบรวมข้อมูลและทำความเข้าใจเกี่ยวกับกลุ่มผู้มีส่วนได้ส่วนเสียในโครงการ ผู้เกี่ยวข้องทั้งภายในและภายนอกองค์กร ช่วยให้เข้าใจความต้องการของผู้เกี่ยวข้อง ความรู้สึก และความคาดหวังที่อาจมีผลต่อโครงการ ในขั้นตอนนี้ จะต้องระบุผู้เกี่ยวข้องที่สำคัญ รวมถึงความสำคัญและผลกระทบของพวกเขาต่อโครงการ การสื่อสารและสร้างความเข้าใจร่วมกันกับ Stakeholders เป็นสิ่งสำคัญ เพื่อให้ทุกคนเข้าใจว่าเรากำลังพัฒนาซอฟต์แวร์ให้ตรงกับความต้องการของพวกเขา

การกำหนดผู้มีส่วนได้ส่วนเสีย (Stakeholder mapping) เป็นขั้นตอนที่เกี่ยวกับการระบุ วิเคราะห์ และจัดหมวดหมู่ Stakeholders ที่เกี่ยวข้องในโครงการนั้นๆ เพื่อเข้าใจความสำคัญของผู้มีส่วนได้ส่วนเสียและสิ่งที่ผู้มีส่วนได้ส่วนเสียต้องการจากซอฟต์แวร์ที่กำลังพัฒนา กระบวนการนี้สามารถสร้างการตระหนักถึงความต้องการและความคาดหวังของ Stakeholders แต่ละกลุ่ม เพื่อให้มีการสื่อสารและการทำงานร่วมกันที่มีประสิทธิภาพมากขึ้นในขณะที่พัฒนาซอฟต์แวร์ นอกจากนี้ การจัดการ Stakeholder mapping ยังช่วยให้สามารถปรับปรุงความต้องการและการวางแผนให้เหมาะสมกับผู้ที่มีผลต่อโครงการ

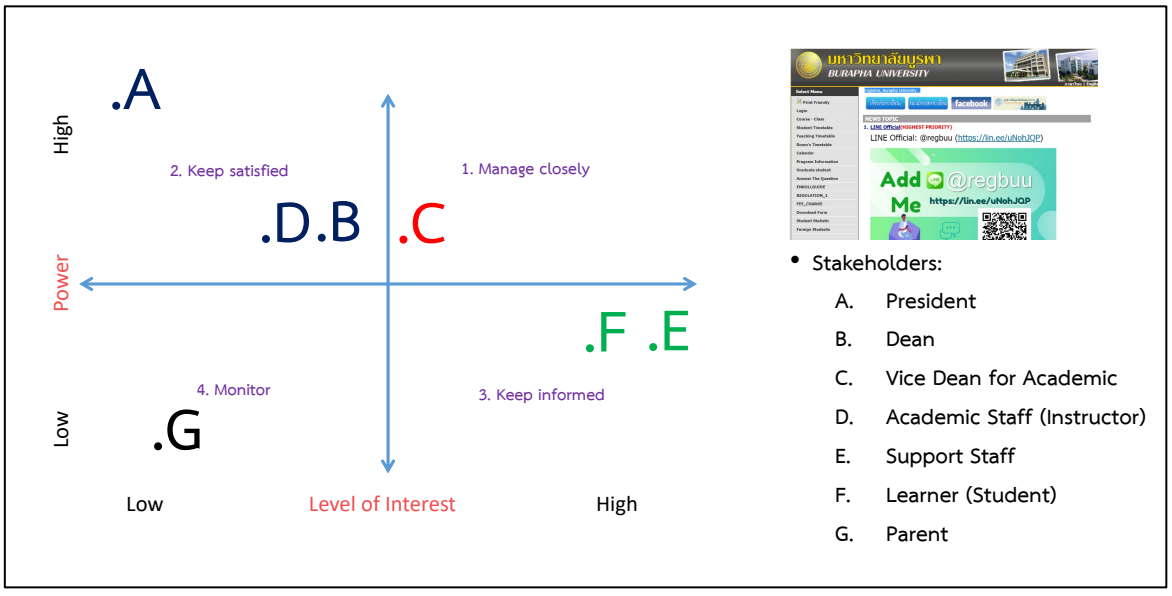


ภาพที่ 2.2 Stakeholder mapping โดยใช้ Mendelow's Matrix

Mendelow's Matrix เป็นเครื่องมือทางธุรกิจที่ใช้ในการวิเคราะห์และทำ Stakeholder mapping โดยให้ความสำคัญกับ**ระดับของอิทธิพลหรืออำนาจตัดสินใจ (Power)** และ**ระดับความสนใจในกิจการของ**

องค์กรหรือมีความเกี่ยวข้องกับระบบ (Level of Interest) แนวคิดหลักของ Mendelow's Matrix นี้คือการตีความความสำคัญของ Stakeholders โดยแยกเป็นสี่กลุ่มหลัก ดังนี้

1. **Manage closely: High Power, High Interest (อิทธิพลสูง, ความสนใจสูง)**
   กลุ่มนี้มีอิทธิพลมาก และมีความสนใจสูงในผลของโครงการ จึงจำเป็นต้องสร้างความพึงพอใจและร่วมมือกับกลุ่มนี้ในการตัดสินใจในการวางแผน กล่าวได้ว่าคนในกลุ่มนี้เป็นทั้งผู้ที่มีอำนาจตัดสินใจและเป็นผู้ที่เกี่ยวข้องกับโครงการหรือระบบที่กำลังพัฒนาอยู่ หรืออาจถูกจำแนกเป็น Primary stakeholders
2. **Keep satisfied: High Power, Low Interest (อิทธิพลสูง, ความสนใจต่ำ)**
   กลุ่มนี้มีอิทธิพลมาก แต่มีความสนใจต่ำ เพราะฉนั้นสามารถปรึกษา/หารือเกี่ยวกับสิ่งต่างๆ เพื่อใช้ประกอบการตัดสินใจ กล่าวคือคนในกลุ่มนี้เป็นผู้มีอำนาจตัดสินใจ แต่ไม่ได้เป็นผู้ที่เกี่ยวข้องกับโครงการหรือระบบที่กำลังพัฒนาอยู่โดยตรง จึงเพียงแค่ขอคำปรึกษาหรือทำให้พึงพอใจเพียงเท่านั้น
3. **Keep informed: Low Power, High Interest (อิทธิพลต่ำ, ความสนใจสูง)**
   กลุ่มนี้มีความสนใจสูง แต่มีอิทธิพลน้อย มักเป็นผู้ปฏิบัติงานหรือผู้ใต้บังคับบัญชาหรือเป็นบุคคลภายนอก มักไม่มีอำนาจตัดสินใจ แต่มักมีความรู้เกี่ยวข้องกับโครงการหรือระบบที่กำลังพัฒนาหรืออาจถูกจำแนกเป็น Secondary stakeholders
4. **Monitor: Low Power, Low Interest (อิทธิพลต่ำ, ความสนใจต่ำ)**
   กลุ่มนี้มีอิทธิพลน้อยและไม่มีความสนใจ ส่วนใหญ่ไม่ต้องให้ความสนใจมากนัก



ภาพที่ 2.3 Stakeholder mapping ของระบบ E-registrar System, Burapha University (www.reg.buu.ac.th)

ภาพที่ 2.3 แสดงตัวอย่างการทำ Stakeholder mapping ระบบ E-registrar (www.reg.buu.ac.th) โดยมีผู้มีส่วนได้ส่วนเสียกลุ่มที่ 1 (Manage closely) ได้แก่ รองคณบดีฝ่ายวิชาการ (Vice Dean for

Academic) กลุ่มที่ 2 (Keep satisfied) ได้แก่ คณบดี (Dean) และอาจารย์ (Academic Staff) กลุ่มที่ 3 (Keep informed) ได้แก่ เจ้าหน้าที่ (Support Staff) และนิสิต (Student) และกลุ่มที่ 4 (Monitor) คือ ผู้ปกครอง (Parent)

การวิเคราะห์ Stakeholders ช่วยลดความขัดแย้งและความเข้าใจผิดพลาดในโครงการ และช่วยให้เราสามารถกำหนดความต้องการของระบบที่ชัดเจนและเป็นไปตามความต้องการของผู้มีส่วนได้ส่วนเสีย ดังนั้นในการวิเคราะห์หรือระบุ (Identify) Stakeholders ของ System-to-be ข้อมูลเกี่ยวกับแต่ละ Stakeholder จำเป็นต้องถูกวิเคราะห์ในประเด็นต่อไปนี้ และแสดงในตารางที่ 2.1

- ตำแหน่งงานของ Stakeholders ในองค์กร
- บทบาทและอำนาจในการตัดสินใจของ System-to-be
- ระดับของความเชี่ยวชาญของข้อมูลที่เกี่ยวข้องกับ System-to-be (Domain expertise)
- ระดับ/ความลึก/ความใส่ใจ (Exposure) ในการรับรู้ปัญหา
- การมีอิทธิพลต่อการยอมรับระบบใหม่ (Influence)
- การมีเหตุผล/นัยยะแอบแฝง และการมีผลประโยชน์ทับซ้อน (Conflict of interest)

ตารางที่ 2.1 Stakeholders Register

ID	Name	Position	Contract Information	Impact/Influence			
				Major Requirements	Power (1-5)	Level of Interest (1-5)	Role(s) in Project
1.	A	CTO	088-888xxx, A@email.com	Students will be able to enroll to undergraduate courses	5	1	Product Owner
2.	B	End user	099-99xxx, B@email.com	Register student and enrol courses pages of the system will load within 5 seconds	2	5	Product Owner

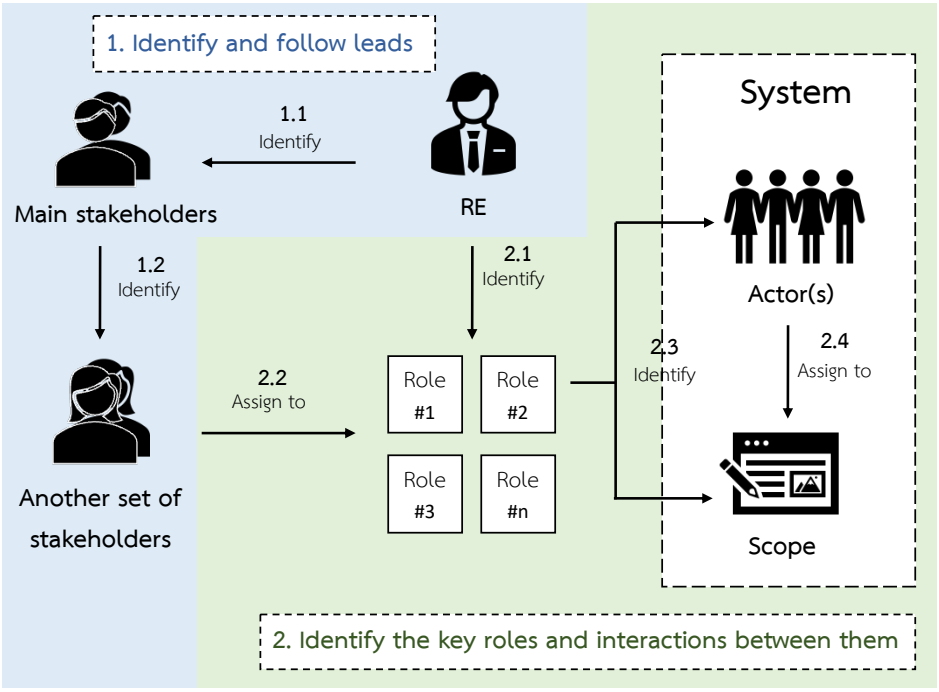
ตารางที่ 2.1 แสดงการทำ Stakeholders Register โดยระบุ/ลงทะเบียนข้อมูลต่างๆ ของผู้มีส่วนได้ส่วนเสียได้แก่ รหัส (ID) ชื่อ-สกุล (Name) ตำแหน่งในองค์กร (Position) ช่องทางการติดต่อ (Contract Information) ความต้องการหลัก (Major Requirements) ระดับของอิทธิพลหรืออำนาจตัดสินใจ (Power) ระดับความสนใจในกิจการขององค์กรหรือมีความเกี่ยวข้องกับระบบ (Level of Interest) และบริบาทของผู้มีส่วนได้ส่วนเสียในโครงการ (Role(s) in Project) โดยแสดงข้อมูลตัวอย่างในตารางที่ 2.2

ตารางที่ 2.2 Stakeholders Register ของระบบ E-registrar System, Burapha University (www.reg.buu.ac.th)

ID	Name	Position	Contract Information	Impact/Influence			
				Major Requirements	Power (1-5)	Level of Interest (1-5)	Role(s) in Project
1	Apisit Saengsai	Instructor	088902xxxx apisit.sa@buu.ac.th	1. xxxx (25/01) 2. xxxx (25/01)	3	3	Steering Committee
2	Robert Abraham	Dean	083506xxxx Robert.sa@buu.ac.th	1. xxxx (22/02) 2. xxxx (25/04)	5	1	Customer PM
3	Mike Murphy	Support Staff	087707xxxx Mike.sa@buu.ac.th	1. xxxx (25/01) 2. xxxx (25/01)	2	5	Steering Committee
...	...	...	...	...	...	...	...

2.3 แนววิธีในการวิเคราะห์ Stakeholders

การวิเคราะห์ Stakeholder มี 2 ขั้นตอนหลักๆ ได้แก่ การกำหนดผู้รับผิดชอบหลักและขยายผลไปยังผู้ที่เกี่ยวของอื่นๆ (Identify and follow leads) และการกำหนดบทบาทสำคัญและสรรหาผู้มีส่วนได้ส่วนที่เกี่ยวข้องกับบทบาทนั้นๆ (Identify the key roles and interactions between them) เพื่อหาความต้องการ/ขอบเขตของระบบ และประเภทผู้ใช้งาน (Actors) ที่เกี่ยวข้อง ดังแกลงในภาพที่ 2.4



ภาพที่ 2.4 แนววิธีที่ใช้ในการวิเคราะห์ Stakeholder

2.3.1 การกำหนดผู้รับผิดชอบหลักและขยายผลไปยังผู้ที่เกี่ยวของอื่นๆ (Identify and follow leads)

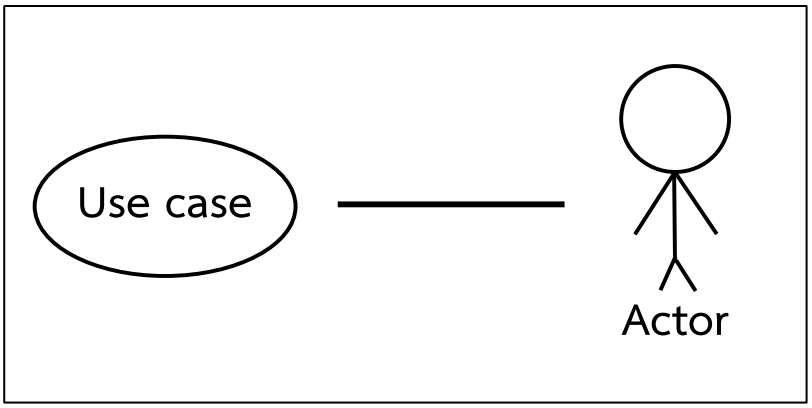
เป็นกระบวนการที่มีวัตถุประสงค์ในการรับรู้ว่าใครเป็นเจ้าของหรือผู้รู้สำคัญ (Main stakeholders) ในเนื้อหาที่เกี่ยวข้องกับระบบที่กำลังพัฒนาอยู่ (System-to-be) ในแต่ละแง่มุม กระบวนการนี้ทำให้สามารถติดตามและสะสมความรู้จากผู้ที่มีบทบาทสำคัญเหล่านี้ได้มากที่สุด โดยทั่วไปมักเริ่มต้นด้วยการประชุมกลุ่มคนที่คาดว่ามีความเกี่ยวข้อง แล้วขยายผลไปยังผู้มีส่วนได้ส่วนเสียกลุ่มอื่นๆ (Another set of stakeholders) โดยการถามคำถาม (ในกรณีของกลุ่มเล็กอาจใช้การระดมความคิดเห็น) เพื่อเข้าใจข้อมูลเบื้องต้น และสร้างความรู้สึกของการร่วมมือกันระหว่างทีมพัฒนาและกลุ่มผู้ใช้ระบบใหม่หรือผู้ที่มีผลกระทบต่อระบบใหม่นี้ โดยในกระบวนการประชุมเริ่มต้นจะต้องค้นหาผู้แทนหรือผู้รับผิดชอบหลัก (Main stakeholder/ Representative) สำหรับแต่ละหัวข้อ โดยใช้ชุดคำถามและการถามซ้ำๆ เพื่อให้ทีมพัฒนามั่นใจว่ากลุ่มคนใดเกี่ยวข้องกับระบบใหม่ในส่วนใด และใครเป็นผู้รับผิดชอบหลัก ผลลัพธ์หลักที่ได้จากการประชุมนี้คือรายชื่อของบุคคลที่ชัดเจนในการติดต่อและความคิดเห็นในแต่ละด้านที่เกี่ยวข้องกับการพัฒนาระบบใหม่ (System-to-be)

2.3.2 การกำหนดบทบาทสำคัญและสรรหาผู้มีส่วนได้ส่วนที่เกี่ยวข้องกับบทบาทนั้นๆ (Identify the key roles and interactions between them)

เป็นการระบุบทบาทหลักที่มีอิทธิพล/อำนาจตัดสินใจในโครงการซอฟต์แวร์ ซึ่งต้องทำหลังจากที่ทีมพัฒนาได้รับข้อมูลเกี่ยวกับผู้เกี่ยวข้องหลัก (Main stakeholders) มาแล้ว โดยกำหนดกรอบของบทบาท (Role) ที่สำคัญๆ ของโครงการ อาทิเช่น ผู้บริหาร ผู้รับผิดชอบโครงการ ผู้รับผิดชอบแต่ละฝ่าย และผู้ใช้งาน เป็นต้น ในขั้นตอนถัดไป ดำเนินการจัดการประชุมเบื้องต้น (Initial workshop) โดยมีวาระหลักที่ต้องพิจารณา คือ การกำหนดว่าผู้เกี่ยวข้องใดจะเป็นตัวแสดงบทบาทใด (Actor) ในระบบใหม่ และมีขอบเขต (Scope) ของระบบใหม่ที่กำลังพัฒนาเช่นไร โดยอาจจะใช้ทั้งการออกแบบระบบ (System design) และการออกแบบหน้าจอต้นแบบ (Prototype) มาใช้ในขั้นตอนนี้ แต่เน้นการหาคำตอบในเรื่อง "ปัญหาหรือที่มาของระบบ"

2.4 แผนภาพยูสเคส (Use case diagram)

แผนภาพยูสเคสเป็นเครื่องมือที่สามารถนำเสนอผู้ใช้งานในระบบจะมีบทบาทอะไรบ้าง (Actor) และสามารถใช้งานฟังก์ชัน/ขอบเขตใดบ้าง (Function/Scope) ง่ายต่อการสื่อสารระหว่างผู้ใช้งานและทีมผู้พัฒนาระบบ ดังแสดงในภาพที่ 2.5



ภาพที่ 2.5 แผนภาพยูสเคส

2.4.1 สัญลักษณ์ของแผนภาพยูสเคส

- **Actor** : ให้ระบุผู้ใช้หรือองค์กรที่มีบทบาทในระบบ อาจเป็นผู้ใช้จริงๆ หรือระบบอื่นๆ
- **Use case** : ให้ระบุการกระทำหรือการทำงานที่สำคัญของระบบ แต่ละ Use Case ควรมีชื่อที่อธิบายการกระทำแบบสั้น กระชับ และมีความหมาย
- **Relationship** : ให้เชื่อมโยง Actor กับ Use Case ที่สัมพันธ์กัน โดยใช้เส้นตรงลากเชื่อมโยงกัน

2.4.2 หลักการในการกำหนดชื่อ Actor

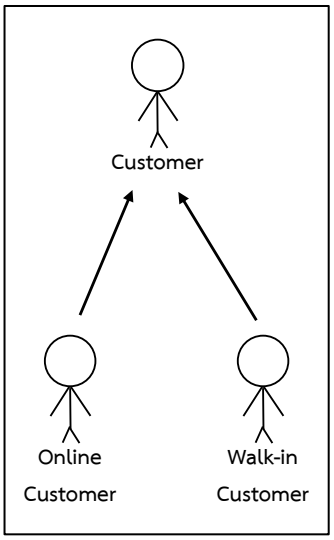
- ควรตั้งชื่อที่ขึ้นต้นด้วยคำนาม (Noun) ในลักษณะสามานยนาม (Common Noun) เช่น นักเรียน เจ้าหน้าที่ห้องสมุด เป็นต้น โดยชื่อดังกว่างต้องสื่อถึงความรับผิดชอบ
- Actor ควรอยู่ทางด้านซ้าย หรือด้านขวาสุดของแผนภาพ
- Actor อาจเป็นระบบหรืออุปกรณ์ภายนอกก็ได้

2.4.3 หลักการในการกำหนดชื่อ Use case

- ควรตั้งชื่อที่ขึ้นต้นด้วยคำกริยา (Verb) เช่น จองห้อง สั่งอาหาร ชำระเงิน เพิ่มข้อมูลลูกค้า เป็นต้น
- ชื่อ Use case ควรสั้น กระชับ และเข้าใจได้
- อาจกำหนดรหัสของ Use case ร่วมด้วย เช่น UC01-จองห้อง / UC02-สั่งอาหาร

2.4.4 ลักษณะของความสัมพันธ์ระหว่างยูสเคส

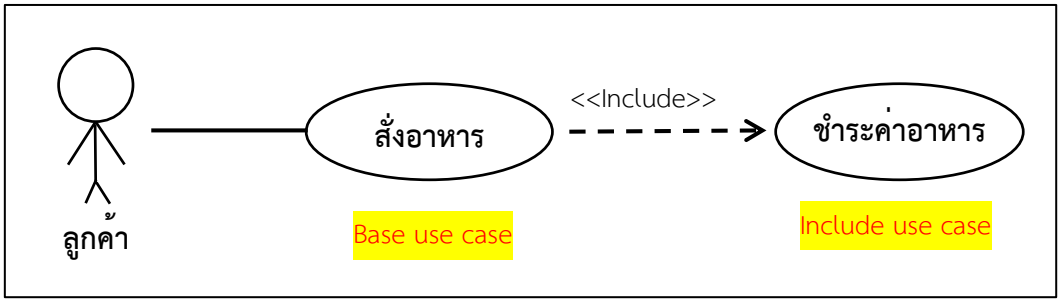
1. **ความสัมพันธ์แบบคุณลักษณะร่วม (Generalization)** Actor สามารถสืบทอดคุณสมบัติพื้นฐาน (Inherit) จาก Actor อื่นๆ ดังภาพที่ 2.6



ภาพที่ 2.6 ความสัมพันธ์แบบคุณลักษณะร่วม (Generalization)

จากภาพที่ 2.6 เห็นได้ว่า Online Customer และ Walk-in Customer สืบทอดจาก Customer ดังนั้น ทั้ง Online Customer และ Walk-in Customer จึงมีความสมามารถเหมือนกันทุกประการกับ Customer และยังสามารถมีความสามารถเพิ่มเติมของตนเองได้อีกด้วย

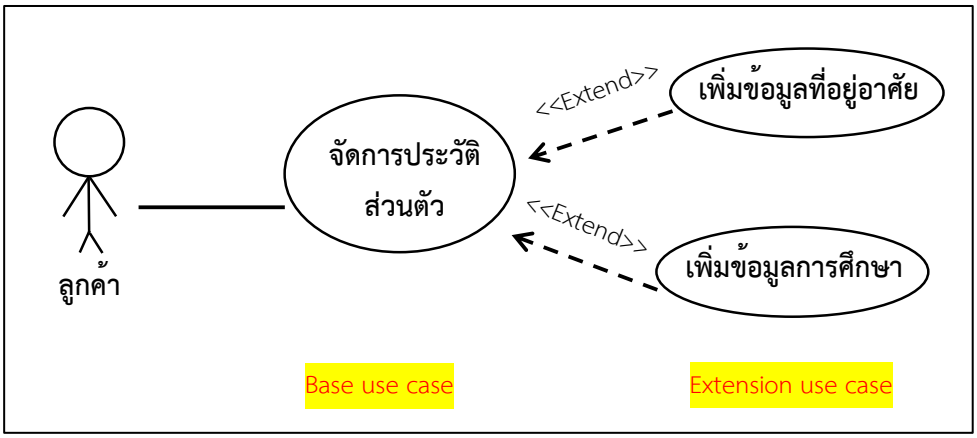
2. **ความสัมพันธ์แบบรวม (Include)** ในกรณีที่ยูสเคสหนึ่งจำเป็นต้องอาศัยการทำงานของยูสเคสอื่น จะใช้สัญลักษณ์จะเป็นลูกศรจากยูสเคสหนึ่ง (Base use case) ไปยังยูสเคสหนึ่งที่ถูกเรียกใช้ (Include use case) ดังภาพที่ 2.7



ภาพที่ 2.7 ความสัมพันธ์แบบรวม (Include)

จากภาพที่ 2.7 อธิบายได้ว่า หากลูกค้าต้องการจะดำเนินการชำระค่าอาหาร ลูกต้องดำเนินการสั่งอาหารก่อนทุกครั้ง

3. **ความสัมพันธ์แบบขยาย (Extend)** ในกรณีที่ยูสเคสหนึ่งมีผลต่อการทำงานตามปกติของยูสเคสหนึ่ง โดยที่ยูสเคสที่มา Extend (Extension use case) นั้นจะมีผลให้การดำเนินงานของยูสเคสปกติ (Base use case) ที่ถูก Extend นั้นจะถูกรบกวน หรือมีการเปลี่ยนแปลงไป ดังภาพที่ 2.8



ภาพที่ 2.8 ความสัมพันธ์แบบขยาย (Extend)

จากภาพที่ 2.8 ลูกค้าสามารถจัดการประวัติส่วนตัวของตนเองได้ และสามารถเพิ่มข้อมูลต่างๆ ได้ เช่น ข้อมูลที่อยู่อาศัย และข้อมูลการศึกษา โดยลูกค้าไม่จำเป็นต้องทำทุกอย่างพร้อมกัน ลูกค้าสามารถเลือกว่าจะเพิ่มข้อมูลที่อยู่อาศัย หรือข้อมูลการศึกษาก่อนหลังตามความต้องการของลูกค้า แต่เมื่อทำกิจกรรมใดกิจกรรมหนึ่งแล้ว (เพิ่มข้อมูลที่อยู่อาศัย หรือเพิ่มข้อมูลการศึกษา) จะพักการทำกิจกรรมจัดการประวัติส่วนตัวไว้ชั่วคราว

2.4.5 หลักการวิเคราะห์ระบบด้วยยูสเคส

ก่อนเริ่มลงมือเขียนแผนภาพยูสเคส ต้องวิเคราะห์ 3 ข้อนี้ก่อน เพื่อใช้เป็นต้องมูลตั้งต้น ได้แก่

1. What : อะไรคืองานที่ต้องทำ หรือคาดว่าจะต้องทำ
2. Who : ใครหรือผู้ใดเป็นผู้กระทำ เกี่ยวข้อง หรือมีส่วนได้ส่วนเสียกับงานเหล่านั้น
3. Why : เพราะเหตุใดจึงต้องกระทำงานเหล่านั้น

เมื่อสามารถอธิบายทั้ง 3 ข้อข้างต้นได้แล้ว จึงเริ่มขั้นตอนในการจัดทำแผนภาพยูสเคสต่อไปนี้

Step 1 : หา Use case และ Actor ของระบบ

Step 2 : สถานการณ์จำลอง (Scenario) ที่อาจเกิดขึ้นได้ในระบบ

Step 3 : เขียนแผนภาพยูสเคส

ตัวอย่างการวิเคราะห์ยูสเคส : การถอนเงินผ่านตู้ ATM

ผู้ใช้งานสอดบัตร ATM เข้าสู่เครื่องรับบัตร หากบัตรใช้งานได้จึงเข้าสู่หน้าจอ Mainn menu หากใช้งานไม่ได้ ATM จะถูกปล่อยคืน (Reject) ออกมา หากบัตรใช้ได้ผู้ใช้งานต้องใส่รหัสข้อมูล PIN (รหัสผ่าน) ที่ลงทะเบียนไว้ หลังจากนั้น เมื่อผู้ใช้ต้องการถอนเงิน ผู้ใช้ต้องระบุจำนวนเงินที่ต้องการถอนเงิน หากมีเงินในบัญชีมากกว่าหรือเท่ากับเงินที่ผู้ใช้ระบุ ผู้ใช้สามารถนำเงินออกจากเครื่อง ATM ได้ โดยแสดงการวิเคราะห์ Scenario ดังภาพที่ 2.9

1. **What** : ตรวจสอบบัตร ATM / ตรวจสอบรหัสผู้ใช้งาน / ถอนเงิน / ตรวจสอบยอดเงินในบัญชี
2. **Who** : ผู้ใช้ (ลูกค้า)
3. **Why** : เพื่อผู้ใช้งานสามาถถอนเงินได้อย่างถูกต้อง

Step 1 : หา Use case และ Actor ของระบบ

Use case : ถอนเงิน / ตรวจสอบบัตร ATM / ตรวจสอบรหัสผู้ใช้งาน / ตรวจสอบยอดเงินในบัญชี

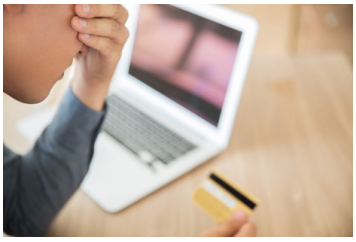
Actor : ผู้ใช้

Step 2 : สถานการณ์จำลอง (Scenario) ที่อาจเกิดขึ้นได้ในระบบ

Scenario #1

บัตรสามารถใช้งานได้ ระบบจะแสดงหน้าจอให้ใส่ PIN

Scenario #2



บัตรเสีย ตู้ ATM จะคืนบัตร

Scenario #3

ผู้ใช้ใส่ PIN ที่เคยลงทะเบียนไว้

Scenario #4

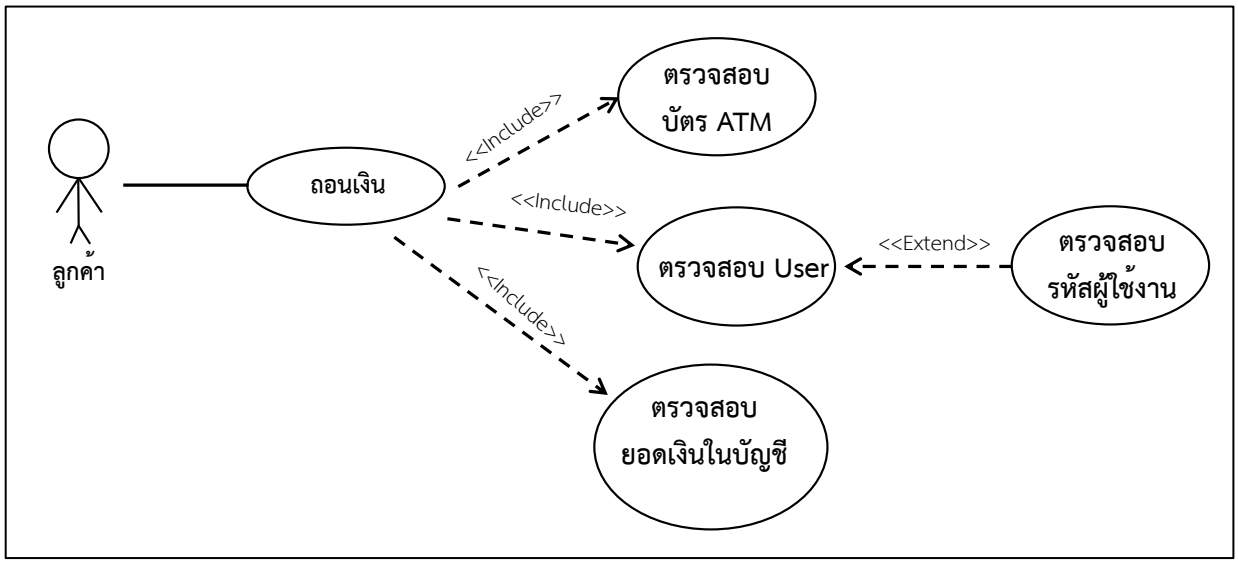
ผู้ใช้ระบุจำนวนเงินที่ต้องการถอน ATM จะตรวจสอบว่าเงินพอให้ถอนหรือไม่

ที่มา https://www.freepik.com/free-photo/hand-inserting-atm-card-into-bank-machine-withdraw-money-businessman-men-hand-puts-credit-card-into-atm_1190147.htm#query=ATM&position=19&from_view=search&track=sph, วันที่สืบค้น 20 กรกฎาคม 2566.

ที่มา https://www.freepik.com/free-photo/businessman-looking-credit-card-stress_1027037,, วันที่สืบค้น 20 กรกฎาคม 2566.

ภาพที่ 2.9 Scenario ระบบ ATM

Step 3 : เขียนแผนภาพยูสเคส



ภาพที่ 2.10 Use cae : ระบบ ATM

จากภาพที่ 2.10 แสดงการเขียน Use case ของระบบ ATM โดยมี 1 Actor คือ ผู้ใช้ และประกอบด้วย 5 use case ได้แก่ 1) ถอนเงิน 2) ตรวจสอบบัตร ATM 2) ตรวจสอบ User 3) ตรวจสอบรหัสผู้ใช้งาน และ 4) ตรวจสอบยอดเงินในบัญชี

เอกสารอ้างอิง

1. A. van Lamsweerde, "Requirements Engineering: From System Goals to UML Models to Software Specifications (Chapter 2)", Wiley, 2009, ISBN 978-0-470-01270-3.
2. I. F. Alexander and R. Stevens, "Writing Better Requirements (Chapter 2)", Addison Wesley, 2002, ISBN 0-321-13163-0.
3. G. Kotonya and I. Sommerville, "Requirements Engineering: Processes and Techniques (Chapter 1 & 2)", John Wiley & Sons, 1997, ISBN 0-471-97208-8.

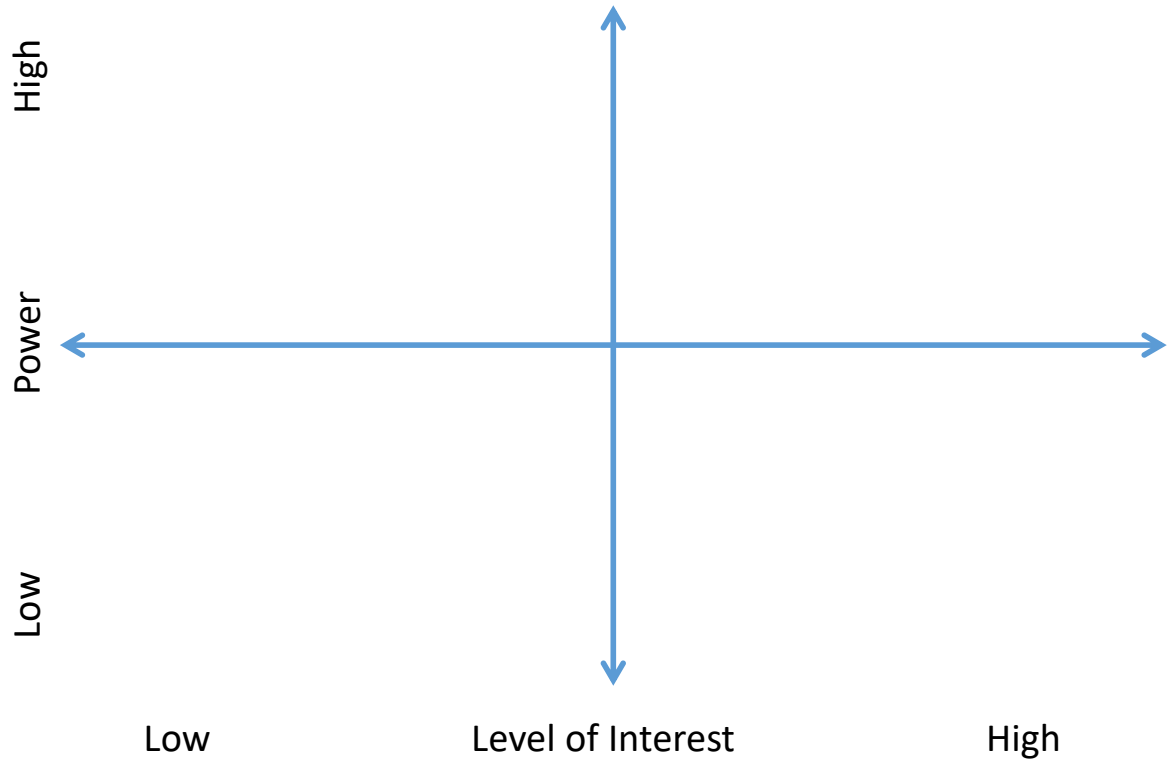
แบบฝึกหัด

ข้อที่ 1

โจทย์ : ให้นิสิตจัดทำ Stakeholder mapping จำนวน 1 ภาพ โดยให้นิสิตเลือก ระบบ/ซอฟต์แวร์/เว็ปไซต์/เกมส์ หรืออื่นๆ มา 1 ระบบ เพื่อใช้ในการวิเคราะห์ Stakeholder mapping และจัดทำตาราง Stakeholders Register จำนวน 1 ตาราง ให้สอดคล้องกับ ภาพ Stakeholder mapping

หัวข้อ	**รายละเอียด**
ระบบ/ซอฟต์แวร์/ เว็ปไซต์/เกมส์	

การวิเคราะห์ Stakeholder mapping

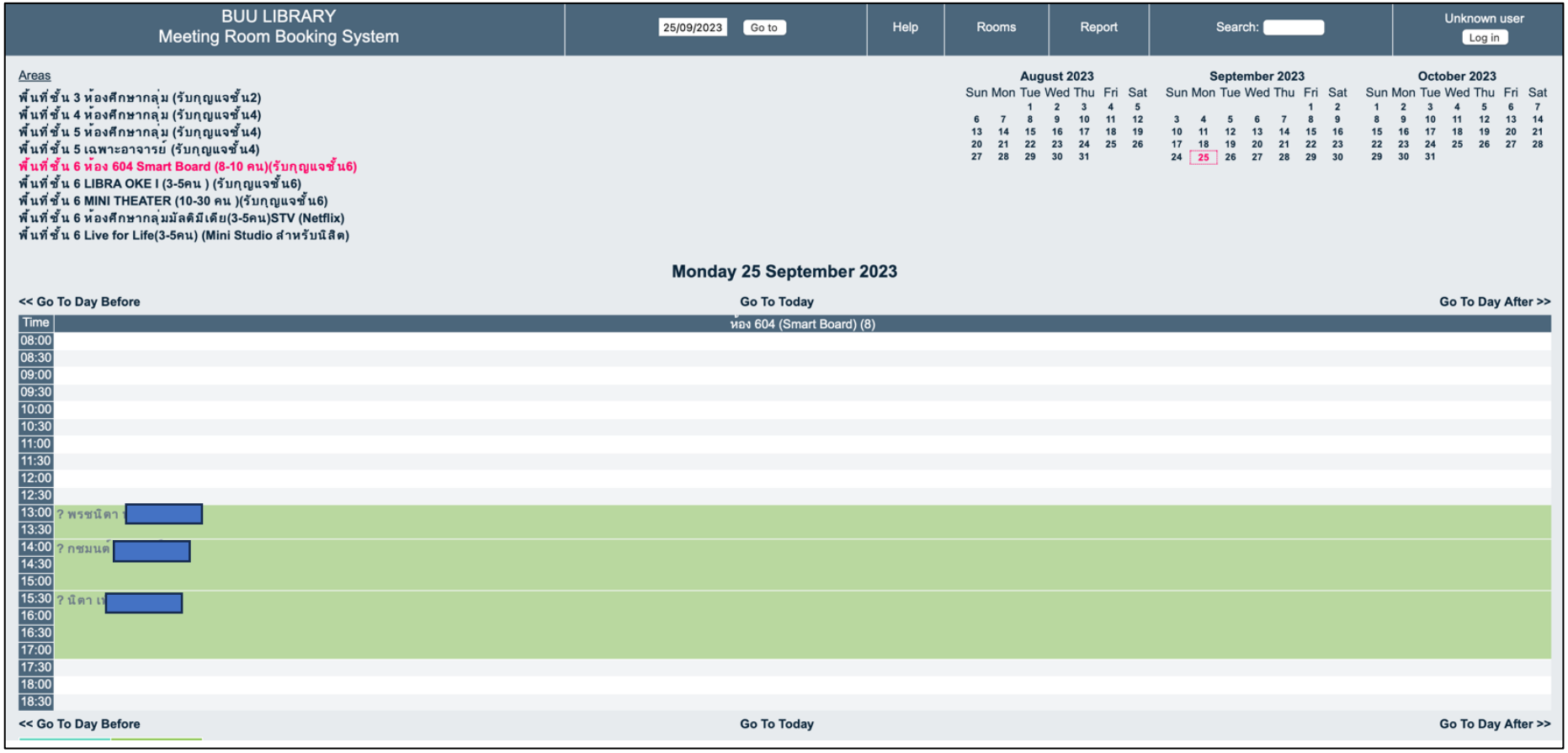


ตาราง Stakeholders Register

ID	Name	Position	Contract Information	Impact/Influence			
				Major Requirements	Power (1-5)	Level of Interest (1-5)	Role(s) in Project

ข้อที่ 2

โจทย์ : ให้นิสิตจัดทำ Use case จำนวน 1 ภาพ โดยให้นิสิตวิเคราะห์ระบบจองห้องประชุมของสำนักหอสมุด มหาวิทยาลัยบูรพา (BUU LIBRARY Meeting Room Booking System) : http://www.lib.buu.ac.th/roombooking/day.php?year=2023&month=09&day=25&area=8&room=18



Use case : BUU LIBRARY Meeting Room Booking System

What	
Who	
Why	

Step 1 : หา Use case และ Actor ของระบบ

Use case :

Actor :

Step 2 : สถานการณ์จำลอง (Scenario) ที่อาจเกิดขึ้นได้ในระบบ (อย่างน้อย 5 Scenario)

Scenario 1 :

Scenario 2 :

Scenario 3 :

Scenario 4 :

Scenario 5 :

Step 3 : เขียนแผนภาพยูสเคส

แผนบริหารการสอน
บทที่ 3 กระบวนการวิศวกรรมความต้องการ

เนื้อหา

1. กระบวนการพัฒนาความต้องการ (Requirements Development)
2. กระบวนการจัดการความต้องการ (Requirements Management)

ผลลัพธ์การเรียนรู้รายบทเรียน

1. ผู้เรียนอธิบายกระบวนการวิศวกรรมความต้องการได้
2. ผู้เรียนอธิบายลำดับของกระบวนการวิศวกรรมความต้องการ และสามารถบอกถึงความสำคัญของแต่ละลำดับได้

วิธีการสอน

1. บรรยาย
2. ยกตัวอย่าง
3. นิสิตร่วมอภิปรายแสดงความคิดเห็น

สื่อการสอน

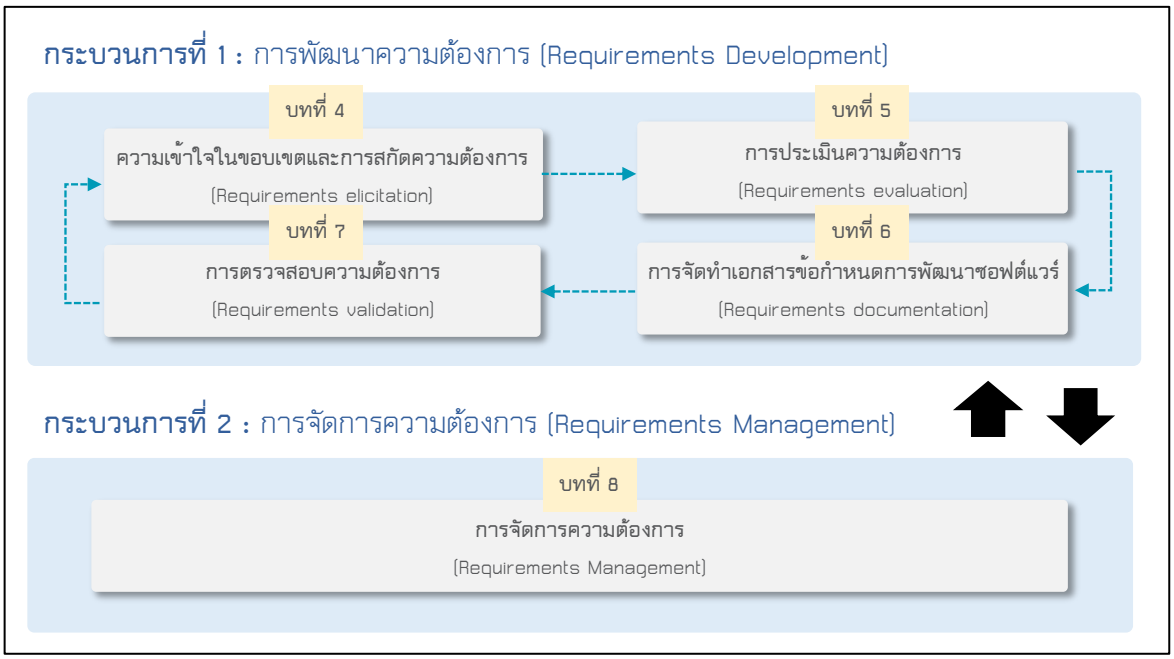
1. แฟ้มดิจิทัลนำเสนอ
2. ระบบมัลติมีเดีย
3. เอกสารคำสอน

การวัดและการประเมินผล

1. คะแนนการเข้าชั้นเรียน
2. คะแนนการทดสอบย่อย
3. คะแนนการสอบกลางภาค
4. คะแนนการถาม-ตอบ

บทที่ 3 กระบวนการวิศวกรรมความต้องการ

ในบทนี้จะนำเสนอกระบวนการหรือกิจกรรมต่างๆ ที่เกี่ยวของกับการทำวิศวกรรมความต้องการ โดยมักเริ่มต้นจากการค้นหาและเข้าใจปัญหา จากนั้นจึงกำหนดความต้องการที่จะแก้ไขปัญหา (Requirement) และทำการตรวจสอบอีกครั้งเพื่อให้แน่ใจว่าความต้องการดังกล่าว สอดคล้องกับการแก้ไขปัญหาหรือไม่ หรือมีความคุ้มค่าหรือไม่ หรือมีความเป็นไปได้หรือไม่ หลายครั้งจำเป็นต้องมีการเจรจาต่อรองกับผู้มีส่วนได้ส่วนเสีย เนื่องจากบางความต้องการอาจดำเนินการได้ยาก หรือต้องใช้ทรัพยากรจำนวนมากเกินกว่าที่โครงการกำหนด



ภาพที่ 3.1 กระบวนการวิศวกรรมความต้องการ

3.1 กระบวนการพัฒนาความต้องการ (Requirements Development)

จากภาพที่ 3.1 การพัฒนาความต้องการเป็นกระบวนการแรกของกระบวนการวิศวกรรมความต้องการ โดยประกอบด้วยกิจกรรมหลักจำนวน 4 กิจกรรม โดยมีรายละเอียดดังนี้

3.1.1 ความเข้าใจในขอบเขตและการสกัดความต้องการ (Requirements elicitation)

การสกัดความต้องการ โดยการสอบถามหรือได้มาของความต้องการจากผู้มีส่วนได้ส่วนเสีย (Stakeholders) ซึ่งอาจเกิดจากการประชุม นำมาจากเอกสารการทำงาน การสำรวจ/สอบถาม การสร้างต้นแบบ หรือสังเกตจากการทำงานจริง หรือในบางครั้งขั้นต้อนนี้อาจเรียกว่า การรวบรวมความต้องการ (Fact Finding) โดยรายละเอียดจะถูกนำเสนอใน**บทที่ 4 ความเข้าใจในขอบเขตและการสกัดความต้องการ (Requirements elicitation)** ของหนังสือเล่มนี้

3.1.2 การประเมินความต้องการ (Requirements evaluation)

เป็นการวิเคราะห์และการตกลงหรือหาข้อยุติร่วมกัน ระหว่างทีมพัฒนาระบบฯ กับ Stakeholders ในขั้นตอนนี้ จะได้ผลลัพธ์เป็นเซตของความต้องการที่มีความเห็นตรงกัน หรือเห็นชอบร่วมกัน โดยรายละเอียดจะถูกนำเสนอใน**บทที่ 5 การประเมินความต้องการ (Requirements evaluation)** ของหนังสือเล่มนี้

3.1.3 การจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์ (Requirements documentation)

เป็นขั้นตอนในการเขียนรายละเอียดความต้องการในรูปแบบเอกสาร (Requirement specification) อาจเป็นการบรรยายความ ประกอบกับเขียนแผนภาพแบบจำลองต่างๆ ที่ทั้งสองฝ่ายเข้าใจตรงกัน โดยรายละเอียดจะถูกนำเสนอใน**บทที่ 6 จัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์ (Requirements documentation)** ของหนังสือเล่มนี้

3.1.4 การตรวจสอบความต้องการ (Requirements validation)

เป็นการตรวจสอบความต้องการในเรื่องความถูกต้อง ความสม่ำเสมอ และความครบถ้วน หากไม่ถูกต้องให้ดำเนินการปรับปรุงและตรวจสอบใหม่อีกครั้ง โดยรายละเอียดจะถูกนำเสนอใน**บทที่ 7 การตรวจสอบความต้องการ (Requirements validation)** ของหนังสือเล่มนี้

3.2 กระบวนการจัดการความต้องการ (Requirements Management)

จากภาพที่ 3.1 หลักจากพัฒนาความต้องการแล้ว ยังคงต้องบริหารจัดการความต้องการให้ความถูกต้องและเป็นปัจจุบันอยู่เสมอ เป็นขั้นตอนในติดตามความต้องการตลอดระยะเวลาของโครงการ เพื่อตรวจสอบว่ามีการทำงานตามความต้องการหรือไม่ และประเมินผลลัพธ์ที่ได้ตรงกับความต้องการหรือไม่ โดยรายละเอียดจะถูกนำเสนอใน**บทที่ 8 การจัดการความต้องการ (Requirements Management)** ของหนังสือเล่มนี้

เอกสารอ้างอิง

1. A. Lamsweerde, "Requirements Engineering from System Goals to UML Models to Software Specifications (Chapter 1)", Wiley, 2009, ISBN 978-0-470-01270-3.
2. B. Berenbach, J. Kazmeier, D. J. Paulish, and A. Rudorfer, "Software & Systems Requirements Engineering in Practice (Chapter 1)", McGrawHill, 2009, ISBN 978-0-07-160547-2.
3. G. Kotonya and I. Sommerville, "Requirements Engineering: Processes and Techniques (Chapter 2)", John Wiley & Sons, 1997, ISBN 0-471-97208-8.
4. I. F. Alexander and R. Stevens, "Writing Better Requirements (Chapter 1)", Addison Wesley, 2002, ISBN 0-321-13163-0.
5. T. T. Barker, "Writing Software Documentation: A Task-Oriented Approach", Longman, 2002, ISBN 0-321-10328-9.

แบบฝึกหัด

1. ให้นิสิตใช้ความรู้ในบทที่ 3 กระบวนการวิศวกรรมความต้องการ ทั้งกระบวนการพัฒนาความต้องการ (Requirements Development) และกระบวนการจัดการความต้องการ (Requirements Management) วิเคราะห์ว่ากิจกรรมใดมีความสำคัญต่อการทำวิศวกรรมความต้องการ**มากที่สุด เพราะเหตุใด**

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

2. และในกิจกรรมใดมีความจำเป็นต้องติดต่อประสานงานกับผู้มีส่วนได้ส่วนเสีย (Stakeholders) **จำนวนมากที่สุด เพราะเหตุใด**

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

ส่วนที่ 2 การพัฒนาความต้องการ

(Requirements Development)

ผลลัพธ์การเรียนรู้ของส่วนนี้

บทที่ 4 ความเข้าใจในขอบเขตและการสกัดความต้องการ (Requirements elicitation)

- ขั้นตอนในการทำความเข้าใจในขอบเขตและการสกัดความต้องการ
- ประเด็นปัญหาของการสกัดความต้องการ
- เทคนิคการสกัดความต้องการแบบ Artifact-drive
- เทคนิคการสกัดในมุมมองของ Stakeholder-driven
- การรวบรวมความต้องการจากแหล่งอื่นๆ

บทที่ 5 การประเมินความต้องการ (Requirements evaluation)

- ประเภทของความไม่สอดคล้องกัน/ขัดแย้งกันของความต้องการ
- การจัดการความไม่สอดคล้องกัน/ขัดแย้งกันของความต้องการ
- การจัดลำดับความสำคัญความต้องการ (Requirements prioritization)

บทที่ 6 การจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์ (Requirements documentation)

- ความต้องการระดับผู้ใช้ (User requirements)
- ความต้องการระดับระบบ (System requirements)
- คุณภาพและข้อบกพร่องที่ควรหลีกเลี่ยงในการจัดทำเอกสารความต้องการ
- ตัวอย่างการเขียนข้อกำหนดความต้องการด้านซอฟต์แวร์

บทที่ 7 การตรวจสอบความต้องการ (Requirements validation)

- ทบทวนความต้องการ
- สมาชิกของทีมทบทวน
- รายการทบทวน

แผนบริหารการสอน
บทที่ 4 ความเข้าใจในขอบเขตและการสกัดความต้องการ

เนื้อหา

1. ขั้นตอนในการทำความเข้าใจในขอบเขตและการสกัดความต้องการ
2. ประเด็นปัญหาของการสกัดความต้องการ
3. เทคนิคการสกัดความต้องการแบบ Artifact-drive
4. เทคนิคการสกัดในมุมมองของ Stakeholder-driven
5. การรวบรวมความต้องการจากแหล่งอื่นๆ

ผลลัพธ์การเรียนรู้รายบทเรียน

1. นิสิตมีความรู้และเข้าใจในขอบเขตและการสกัดความต้องการ
2. นิสิตมีความรู้และเข้าใจการประเมินความต้องการ

วิธีการสอน

1. บรรยาย
2. ยกตัวอย่างระบบ โดยใช้กรณีตัวอย่างระบบที่เป็นที่รู้จัก
3. นิสิตร่วมอภิปรายแสดงความคิดเห็น

สื่อการสอน

1. แฟ้มดิจิทัลนำเสนอ
2. ระบบมัลติมีเดีย เช่น คอมพิวเตอร์ เครื่องเสียง เครื่องฉายภาพ
3. เอกสารประกอบคำสอน

การวัดและการประเมินผล

1. คะแนนการเข้าชั้นเรียน
2. คะแนนแบบฝึกหัด
3. คะแนนการอภิปรายของนิสิต

บทที่ 4 ความเข้าใจในขอบเขตและการสกัดความต้องการ

“ส่วนที่ยากที่สุดของการพัฒนาระบบสารสนเทศคือ การตัดสินใจว่าจะสร้างระบบอะไร”

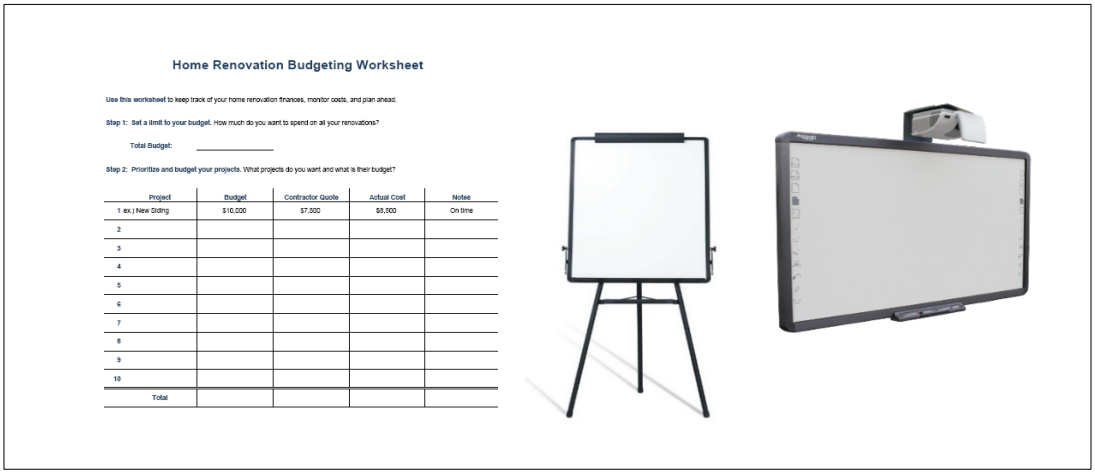
Dr. Fredrick P. Brooks, Jr

การสกัดความต้องการ (Elicitation) เป็นกระบวนการในการค้นหาปัญหา วิเคราะห์ เข้าใจ และนำมาระบุเป็นความต้องการตอบสนองต่อการทำงานของระบบ หรือองค์กรภายใต้ข้อจำกัดที่มีอยู่ เทคนิคการสกัดมีหลายชนิด และมีความรู้ที่ควรทราบหลายเรื่องดังต่อไปนี้

4.1 ขั้นตอนในการทำความเข้าใจในขอบเขตและการสกัดความต้องการ

การสกัด (Elicitation) จำเป็นต้องมีปฏิสัมพันธ์หรือการประสานงานกับ Stakeholders เพื่อรวบรวม (Capture) ปัญหา/ความต้องการ การวิเคราะห์ (Analysis) เป็นการกลั่นกรองความต้องการของ Stakeholders เพื่อเป็นข้อกำหนดของซอฟต์แวร์อย่างเป็นทางการ (Specification) ประกอบด้วย 4 ขั้นตอนคร่าวๆ ดังต่อไปนี้

1. **การรวบรวมความต้องการร่วมกัน (Collaborative requirements gathering)**
   - จัดให้มีการประชุมทั้งทีมวิศวกรความต้องการและ Stakeholders
   - มีข้อกำหนดการประชุมต่างๆ สำหรับการเตรียมตัวเข้าร่วมประชุม
   - กำหนดวาระการประชุมที่ชัดเจนที่ทำให้การประชุมนั้นเป็นทางการ และครอบคลุมประเด็นเนื้อหาที่สำคัญทั้งหมด
   - หาสถานที่ที่สามารถอยู่ร่วมกันได้ และทบทวนให้แน่ใจว่าทั้งทีมวิศวกรรมซอฟต์แวร์และ Stakeholders ดู/ใช้/รับทราบข้อมูลชุดเดียวกัน
   - ประธานการประชุม (Facilitator) อาจจะเป็นลูกค้า นักพัฒนา หรือบุคคลภายนอก ควรจะมีผู้ที่มีความรู้ในเชิงลึกเพื่อจะทำการสกัดความต้องการ ให้เข้าถึงความต้องการที่แท้จริง
   - มีการกำหนดเครื่องมือที่จะใช้ในการประชุม อาจมีการใช้ Mind map กระดาษโน้ต โปรเจคเตอร์ Work sheet เพื่อรวบรวมปัญหาต่างๆ



Home Renovation Budgeting Worksheet

Use this worksheet to keep track of your home renovation finances, monitor costs, and plan ahead.

Step 1: Set a limit to your budget. How much do you want to spend on all your renovations?

Total Budget: _____

Step 2: Prioritize and budget your projects. What projects do you want and what is their budget?

	Project	Budget	Contractor Quote	Actual Cost	Notes
1	ex.) New Siding	$10,000	$7,500	$8,000	On time
2					
3					
4					
5					
6					
7					
8					
9					
10					
	Total				

ภาพที่ 4.1 ตัวอย่างเครื่องมือในการรวบรวมสกัดความต้องการ

จากภาพที่ 4.1 แสดงตัวอย่างเครื่องมือในการรวบรวมสกัดความต้องการ เช่น Work sheet, Flip board และ Electronic board

2. การวิเคราะห์ความต้องการอย่างมีคุณภาพ (Quality Function Deployment)

Quality Function Deployment (QFD) เป็นเทคนิคการจัดการคุณภาพ ที่นำความต้องการของลูกค้ามาทำให้เป็นข้อกำหนดรายละเอียดของซอฟต์แวร์ (Specification) โดยมีจุดมุ่งหมายเพื่อสร้างความพึงพอใจให้กับลูกค้า จากกระบวนการทางวิศวกรรมซอฟต์แวร์เพื่อให้บรรลุเป้าหมาย หากทีม RE สามารถวิเคราะห์ความต้องการจาก Stakeholders ได้ว่าความต้องการนั้นเป็นความต้องการประเภทไหน จะทำให้ Software specification มีความสมบูรณ์ ซึ่งความต้องการของลูกค้า แบ่งเป็น 3 ประเภทดังนี้

2.1 ความต้องการปกติ (Normal requirements)

Normal requirements หมายถึงวัตถุประสงค์ และเป้าหมายที่ถูกระบุไว้ในซอฟต์แวร์ หรือระบบ โดยจะเกิดขึ้นในระหว่างที่มีการประชุมกับ Stakeholders ข้อกำหนดเหล่านี้จะสามารถสร้างความพึงพอใจให้แก่ Stakeholders

2.2. ความต้องการที่คาดหวัง (Expected requirements)

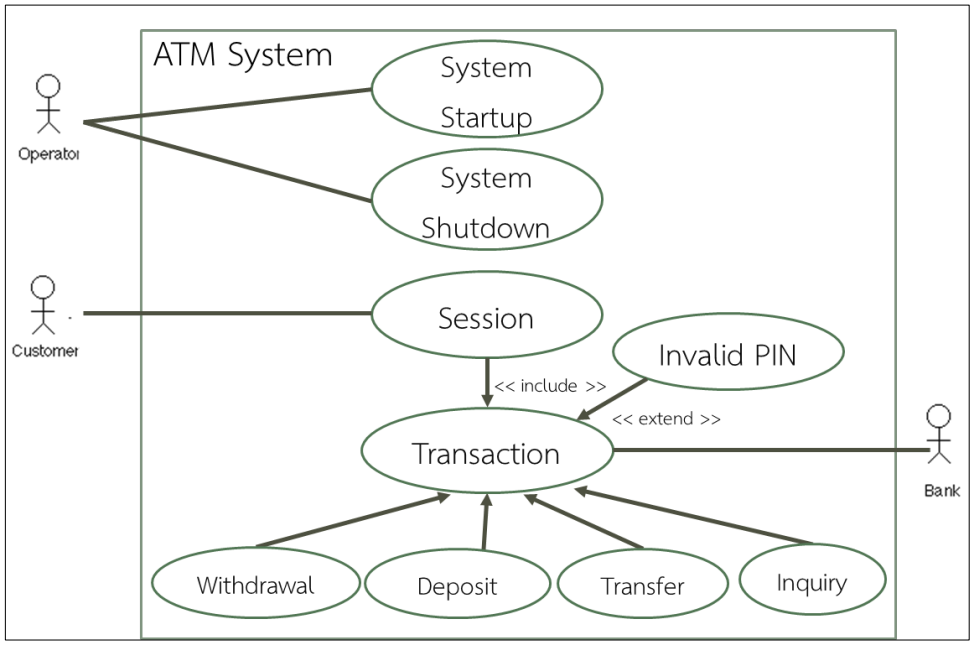
Expected requirements หมายถึงความต้องการที่ Stakeholders อาจไม่ได้ระบุไว้อย่างชัดเจน อาทิ เช่น ความสะดวกในการปฏิสัมพันธ์ของมนุษย์กับตัวเครื่อง ความสะดวกในการติดตั้งซอฟต์แวร์ เป็นต้น

2.3 ความต้องการที่น่าทึ่ง (Exciting requirements)

Exciting requirements คือ คุณลักษณะที่เกินกว่าความคาดหวังของ Stakeholders อาทิเช่น ซอฟต์แวร์สำหรับสมาร์ทโฟน หรือแท็บเล็ตที่มาพร้อมกับคุณลักษณะมาตรฐาน ได้แก่ หน้าจอสัมผัส ข้อความภาพเสียง (Visual voice mail)

3. การจัดทำ Scenario ที่เกี่ยวข้อง

Scenarios บางครั้งอาจเรียกว่า Use case คือ การมองระบบให้เป็นภาพ ที่แสดงถึงกระบวนการทำงานของระบบ เนื่องจากการรวบรวมความต้องการประเภท Functional system นั้นเริ่มเป็นรูปเป็นร่างมากขึ้น แต่การนำสื่อเหล่านี้ไปทำอีกกิจกรรมหนึ่งเพื่อทำให้เข้าใจการทำงานนั้นเป็นเรื่องยาก ดังนั้นนักพัฒนา และผู้ใช้งานจึงสร้างชุดของสถานการณ์ขึ้นมา (Set of scenarios) เพื่อที่จะระบุหัวข้อของการใช้งานของระบบให้เข้าใจง่ายมากขึ้น



ภาพที่ 4.2 ตัวอย่าง Usage scenario : Use case

จากภาพที่ 4.2 Use case ที่อธิบายถึงระบบ ATM System แสดงให้เห็นถึงการทำงานกันระหว่าง Operator Customer และ Bank ที่กระทำกับระบบ ทำให้เกิดเหตุการณ์ คือ Scenario ที่เกี่ยวข้อง

4. การสกัดเพื่อให้ได้มาซึ่งชิ้นงานต่างๆ (Elicitation work products)

ในการสกัดความต้องการจะทำให้ได้ Work product หรือชิ้นงานต่างๆ ที่ถูกพัฒนาขึ้นมา ซึ่งอาจอยู่ในรูปแบบที่หลากหลาย เช่น กระดาษโน้ต กระดาษบันทึก กระดาษ Post-it หรือ Prototype อย่างง่าย โดยความต้องการควรจะเป็นสิ่งที่เป็นไปได้ หรือสามารถทำได้ ซึ่งจะขึ้นอยู่กับขนาดของระบบที่จะพัฒนา อาทิเช่น ความต้องการของลูกค้าควรอยู่ในขอบเขตของระบบ หรือซอฟต์แวร์ รายการของลูกค้า ผู้ใช้ และ Stakeholders ที่มีส่วนร่วมในการสกัดความต้องการ และต้นแบบ (Prototype) จะถูกพัฒนาเพื่อกำหนดความต้องการให้ดียิ่งขึ้น เป็นต้น ซึ่งในแต่ละ Work product จะถูกตรวจทาน โดยทุกคนที่มีส่วนร่วมในการสกัดความต้องการ

4.2 ประเด็นปัญหาของการสกัดความต้องการ

ประเด็นปัญหาของการสกัดความต้องการส่วนใหญ่เกิดขึ้นได้จากหลายสาเหตุ ขึ้นอยู่กับสถานการณ์และปัจจัยต่างๆ โดยมีประเด็นปัญหาหลักๆ ดังต่อไปนี้ๆ

4.2.1 คนที่มีความรู้ไม่เพียงพอกับการสกัดความต้องการ (The missing ignoramus)

การสกัดความต้องการเป็นบุคคลที่มีประสบการณ์และได้รับการฝึกอบรมจากเทคนิคการสกัดความต้องการ แต่อย่างไรก็ตามทีมที่จะเข้าไปสกัดความต้องการอาจจะประกอบด้วยผู้ที่มีประสบการณ์และไม่มีประสบการณ์ อย่างไรก็ตามจะต้องกล้าที่จะถามว่า “สิ่งนี้หมายความว่าอะไร” (What) / "ทำไมต้องทำหรือทำไปเพื่ออะไร" (Why) / “ใครเป็นผู้รับผิดชอบหรือมีส่วนเกี่ยวข้อง” (Who) / “ดำเนินการเมื่อใด” (“When”) / มีส่วนเกี่ยวข้องกับส่วนอื่นๆ หรือไม่ (Where) และต้องดำเนินการอย่างไร (How)

4.2.2 ไม่ใช่ Stakeholder ที่แท้จริง (The wrong stakeholder)

Stakeholders หรือ Domain expert ไม่อาจรู้ระบบทั้งหมดขององค์กร และในบางครั้งไม่สามารถทราบได้ว่า Stakeholders ที่ให้ Requirements นั้นเป็นตัวจริงหรือไม่ ทราบเรื่องของระบบหรือไม่ และมีความรู้ความเข้าใจเกี่ยวกับขอบเขตของซอฟต์แวร์หรือไม่

4.2.3 นักวิเคราะห์ขาดประสบการณ์ (Untrained analysts)

นักวิเคราะห์ที่มีประสบการณ์ หรือมีทักษะทางด้านธุรกิจเป็นอย่างดี (Business-savvy) จะทำให้เข้าใจระบบมากขึ้น ซึ่งนักวิเคราะห์จะมีหน้าที่แค่รวมองค์กร โครงการ หรือรวมสิ่งที่ต้องการ แต่ไม่สามารถรวมความคิดหรือตัดสินใจในการแก้ปัญหาได้ จึงเป็นเรื่องที่ยากมากสำหรับคนที่ไม่มีประสบการณ์ที่จะแยกความต้องการออกจากปัญหา

4.2.4 ไม่ระบุระดับความสำคัญของความต้องการ (Not identifying requirements level)

ความต้องการคือ เมื่อมีการรวบรวมความต้องการมากขึ้น การไม่ระบุความสำคัญของความต้องการจะทำให้ความต้องการมีความซับซ้อนมากขึ้น สิ่งที่สำคัญคือ เมื่อข้อมูลที่ Stakeholders ให้มาถูกรวบรวม ควรจะต้องบันทึกข้อมูล และระบุระดับความสำคัญของข้อมูล

4.2.5 การแยกปัญหาออกจากความต้องการ (Problems separating context from requirements)

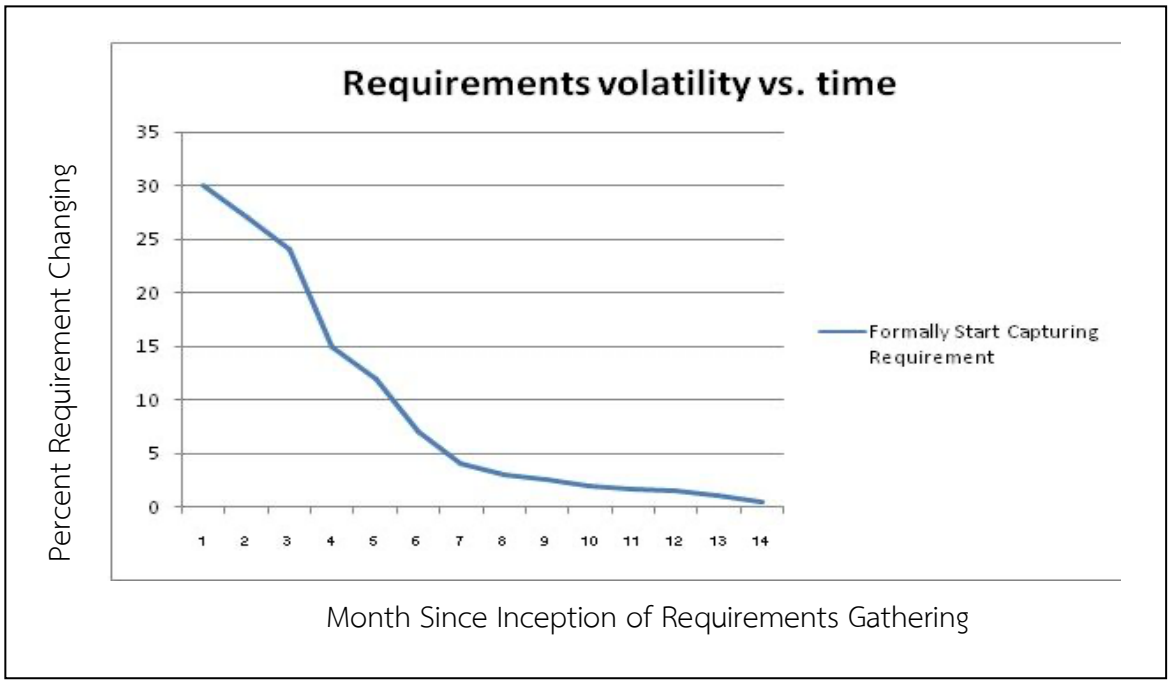
การร้องขอสิ่งที่ต้องการจาก Stakeholders บางครั้งอาจเป็นเรื่องที่ยากมากเมื่อ Stakeholders พูดในเรื่องที่ไม่มีประโยชน์หรือไม่เกี่ยวข้องกับระบบและ Stakeholders อาจจะสับสนระหว่างปัญหากับสิ่งที่ต้องการ

4.2.6 ความล้มเหลวในการเก็บรวบรวมข้อมูล (Failure to collect enough information)

เกี่ยวกับระยะเวลาที่จำกัดของโครงการ ยกตัวอย่างเช่น หากเป็นโครงการพัฒนาระบบภาษีอากร นักบัญชีอาจจะกำลังยุ่งมากในช่วงการเตรียมเก็บภาษี และสามารถพบกับนักวิเคราะห์ระบบได้เพียงหนึ่งชั่วโมงต่อสัปดาห์เท่านั้น วิธีแก้ไขคือ ควรตกลงเวลาในการเข้าพบ นัดหมาย หรือสอบถามในการสกัดความต้องการในครั้งต่อไป เนื่องจากมีเวลาที่จำกัดที่ทำให้ไม่สามารถเข้าพบได้ในทุกครั้งที่ต้องการ

4.2.7 มีการเปลี่ยนแปลงความต้องการบ่อยเกินไป (Requirements are too volatile)

โครงการส่วนใหญ่จะต้องมีการเปลี่ยนแปลงความต้องการ 1 เปอร์เซ็นต์ ถึง 3 เปอร์เซ็นต์ต่อเดือน ในความหมาย หรือการตีความของความต้องการ กล่าวโดย โจนส์ ในปี 2008 และ รอยซ์ ในปี 1998 หากความต้องการมีการเปลี่ยนแปลงอย่างรวดเร็ว ความต้องการที่ถูกระบุไว้แล้วแต่แรก อาจจะไม่สามารถทำให้บรรลุผลสำเร็จได้ จึงจำเป็นต้องรอจนกว่าระดับของความต้องการจะเสถียรลง เพื่อให้ได้ความต้องการพื้นฐานที่จะถูกระบุไว้ในระบบ แสดงในรูปที่ 4.3



ภาพที่ 4.3 กราฟการเปลี่ยนแปลงความต้องการ

4.2.8 ขอบเขตของระบบไม่ถูกกำหนดอย่างชัดเจน (System boundaries are not identified)

ควรมีการนิยามขอบเขตของระบบให้มีความชัดเจนแน่นอน เพื่อไม่ก่อให้เกิดความซ้ำซ้อนในการทำงาน และการกระจายงานเกินขอบเขตของระบบ ความเข้าใจในขอบเขตของระบบไม่ตรงกันระหว่างทีมผู้พัฒนากับลูกค้า ทำให้งานที่ออกมาไม่ตรงกับที่ลูกค้าต้องการ

4.2.9 ความเข้าใจของความต้องการซอฟต์แวร์ยังไม่ชัดเจน (Understanding of product needs is incomplete)

นักวิเคราะห์ระบบควรสอบถามข้อมูลจาก Stakeholders เกี่ยวกับระบบที่ต้องทำ หากยังมีความเข้าใจระบบไม่ชัดเจนมากพอ เพื่อให้ทราบถึงการทำงานที่ถูกต้องของระบบที่ Stakeholders ต้องการอยากให้เป็นนักวิเคราะห์ระบบ สอบถามข้อมูลเพื่อช่วยจำกัดขอบเขตความต้องการของซอฟต์แวร์ให้แคบลง และสร้างต้นแบบ (Prototyping) เพื่ออธิบายรายละเอียดของระบบในความเข้าใจของนักวิเคราะห์ระบบให้ Stakeholders ทำการตรวจสอบ

ข้อควรคำนึงในการระบุขอบเขตความต้องการของระบบ มีดังนี้ เป็นสิ่งที่ลูกค้าต้องการและจำเป็นต้องมีหรือไม่ สามารถสร้างขึ้นได้ (ด้วยเทคโนโลยี เวลา และงบประมาณที่มี) หรือไม่ สามารถอธิบายคุณสมบัติของผลิตภัณฑ์ที่ต้องนำเสนอได้หรือไม่ และสามารถแสดงให้ลูกค้าเห็นว่าเพราะเหตุใดจึงสมควรเสียเงินซื้อสินค้านั้น

4.2.10 ผู้ใช้เข้าใจว่าคอมพิวเตอร์สามารถทำได้ทุกอย่าง (User misunderstand what computers can do)

ผู้ใช้เข้าใจว่าซอฟต์แวร์สามารถทำงานได้ทุกอย่าง โดยไม่จำกัดชนิดของงาน ซึ่งงานบางอย่างเทคโนโลยียังไม่ก้าวล้ำไปถึง ซึ่งเทคโนโลยีบางอย่างอาจจะยังอยู่ในขั้นตอนของการวิจัย และยังไม่สามารถนำมาใช้ได้จริง

4.2.11 วิศวกรความต้องการควรเข้าใจรายละเอียดของงานอย่างถ่องแท้ (The requirements engineer has deep domain

นักวิเคราะห์ควรศึกษาข้อมูลของระบบ เพื่อลดระยะเวลาในการสื่อสารกับ Stakeholders กล่าวคือสามารถทำความเข้าใจระบบได้อย่างรวดเร็ว เข้าใจรายละเอียดเนื้อหาของระบบอย่างลึกซึ้ง เพื่อใช้ในขั้นตอนการออกแบบ ขั้นตอนการพัฒนาและทดสอบระบบ

4.2.12 Stakeholders มีการใช้ภาษาทางเทคนิคที่ต่างกัน (Stakeholders speak different natural and technical languages)

Stakeholders มาจากสถานที่หรือสายงานที่แตกต่างกัน ฉะนั้นการสื่อสารกันระหว่างการประชุมร่วมกัน เพื่อให้มีความเข้าใจตรงกันจึงทำได้ยาก นักวิเคราะห์ระบบจะต้องอำนวยความสะดวก และทำให้การประชุมเป็นไปอย่างราบรื่น เพื่อที่จะสามารถเก็บข้อมูลความต้องการให้ได้มากที่สุด และนักวิเคราะห์ระบบจะต้องนำข้อมูลเหล่านั้นมาวิเคราะห์ และนำเสนอแนวคิดหลักของระบบในรูปแบบที่จะทำให้ผู้ใช้เข้าใจได้ง่าย เช่น แผนภาพ (Diagram) และตาราง (Table) เพื่อสื่อถึงภาพรวมของระบบ

4.2.13 Stakeholders ละเลยข้อมูลที่สำคัญ (Stakeholders omit important, Well-understood, Tacit information)

Stakeholders หรือ Domain expert อาจจะไม่ได้นำข้อมูลบางอย่างที่จำเป็นต้องอธิบายมารวบรวมเอาไว้ในเนื้อหา เพราะคิดว่าข้อมูลนั้นเป็นข้อมูลขั้นพื้นฐาน ซึ่งอาจไม่จำเป็นต้องเอามานำเสนอ ทำให้ข้อมูลไม่ครบถ้วนสมบูรณ์ ตัวอย่างเช่น “ในการทอดไข่เจียว จะต้องเริ่มจากการเจียวไข่ จากนั้นนำไข่ลงกระทะแล้วทอด” ขั้นตอนที่ขาดหายไป เช่น การตั้งกระทะ และรอน้ำมันให้ร้อนก่อนนำไข่ลงทอด เป็นต้น

4.2.14 Stakeholders มีมุมมองที่ขัดแย้งกัน (Stakeholders have conflicting views)

การประชุมการสกัดความต้องการจะต้องมีการสนับสนุนให้ระดมสมอง เพื่อกลั่นกรองความต้องการ เมื่อ Stakeholders มีมุมมองที่ขัดแย้งกัน ทำให้เกิดการถกเถียง และมีความขัดแย้งที่ต้องแก้ไขตามมา ซึ่งการแก้ไขความขัดแย้งระหว่าง Stakeholders ไม่ควรทำในขณะที่ประชุม หรือสอบถามข้อมูลความต้องการ เพราะอาจทำให้ได้ข้อมูลความต้องการที่ละเอียดมากขึ้น

4.3 เทคนิคการสกัดความต้องการแบบ Artifact-drive

เพื่อสนับสนุนการเก็บรวบความต้องการ มีเทคนิคในการสกัดความต้องการอยู่หลายวิธี ในทีนี้จะยกตัวอย่างการใช้หลักการ Artifact-drive 7 เทคนิค โดยมีรายละเอียดดังนี้

1. Background study
2. Data collection
3. Questionnaires
4. Repertory grids & Card sorts for concept driven acquisition
5. Storyboards and scenarios for problem world exploration
6. Mockups and prototype for early feedback
7. Knowledge reuse

อธิบายรายละเอียดทางเทคนิคการสกัดความต้องการแบบ Artifact-driven ดังนี้

4.3.1 ศึกษาจากข้อมูลเดิมที่มีอยู่ (Background study)

- เรียกอีกอย่างหนึ่งว่า “Content analysis” ศึกษาได้จากสถานการณ์ภายในองค์กร และขอบเขตความรู้พื้นฐาน (Underlying domain) เทคนิคที่สำคัญ มีดังนี้ การจัดเก็บรวบรวม (Collection) การอ่าน (Reading) และการสังเคราะห์ (Synthesize) ที่มีความเกี่ยวข้องกับเอกสารประกอบหรือหนังสืออ้างอิง (Documentation)
- เรียนรู้เกี่ยวกับองค์กร หรือหน่วยงานที่มีการจัดการอย่างเป็นระบบ (Organization) ศึกษาได้จากเอกสาร เช่น ผังองค์กร (Organization chart) แผนธุรกิจ (Business plan) คู่มือนโยบาย (Policy manuals) รายงานทางการเงิน (Financial report) นัดประชุมที่สำคัญๆ (Minutes of important meetings) รายละเอียดของงาน (Job description) รูปแบบธุรกิจ (Business forms) และสิ่งอื่นๆ ที่คล้ายกัน
- เรียนรู้เกี่ยวกับขอบเขตของความรู้ (Domain) ศึกษาได้จากหนังสือเรียน (Books) การสำรวจ (Surveys) บทความที่ตีพิมพ์ (Published articles) ศึกษากฎระเบียบข้อบังคับ (Regulation enforced)
- เรียนรู้ System-as-is ให้ละเอียดขึ้น ศึกษาได้จากเอกสารที่มีการไหลของข้อมูล กระบวนการการทำงาน (Work procedures) กฎทางธุรกิจ (Business rules) รูปแบบการแลกเปลี่ยนระหว่างองค์กรหน่วยเล็กๆ และหน่วยงานอื่นๆ

ประโยชน์ของการใช้พื้นฐานความรู้เดิม ช่วยให้สามารถเตรียมความพร้อมก่อนที่จะประชุม หรือพบปะกับ Stakeholders เรียนรู้ในขั้นพื้นฐาน เกี่ยวกับศัพท์เทคนิค วัตถุประสงค์ (Objectives) นโยบาย (Policies) ที่จะนำมาพิจารณาในการแบ่งหน้าที่ความรับผิดชอบให้กับ Stakeholders และอื่นๆ ทำให้ได้มาซึ่ง Meta-

knowledge นำมาใช้ในการจัดพื้นที่เอกสาร เพื่อคัดแยกเอกสาร และมุ่งเน้นไปทางด้านที่เกี่ยวข้องกับระบบเท่านั้น ข้อจำกัดของการใช้พื้นฐานความรู้เดิม จำนวนของเอกสารที่ต้องพิจารณามีจำนวนมากทำให้ต้องมีการสกัดเฉพาะข้อมูลสำคัญ และเอกสารบางส่วนอาจเป็นเอกสารที่ไม่ถูกต้อง หรือล้าสมัย

4.3.2 การเก็บรวบรวมข้อมูล (Data collection)

การเก็บรวบรวมข้อมูลจะรวบรวมเฉพาะข้อเท็จจริง เพื่อประสิทธิภาพในการสกัดความต้องการที่ถูกต้องซึ่งข้อมูลที่ถูกรวบรวม ได้แก่ ข้อมูลทางการตลาด สถิติการใช้งาน ประสิทธิภาพของข้อมูล (Performance figures) ค่าเฉลี่ยต้นทุนและอื่นๆ โดยใช้วิธีการสุ่มตัวอย่างทางสถิติเพื่อให้ข้อมูลได้ข้อสรุปที่น่าเชื่อถือเนื่องจากข้อมูลดังกล่าวอาจมีปริมาณมาก การจะได้ข้อมูลที่มีประโยชน์ และถูกต้องอาจใช้เวลานานและจะเป็นประโยชน์อย่างมาก ต่อการสกัดความต้องการแบบ Non-function โดยเฉพาะข้อมูลที่เกี่ยวข้องกับความสามารถในการใช้งานของซอฟต์แวร์ (Usability) ประสิทธิภาพ (Performance) และการคำนวณค่าใช้จ่าย (Cost) ซึ่งหมายถึงเวลาที่ใช้ในการเข้าถึงข้อมูล

- ข้อดี คือ วิธีนี้เหมาะสำหรับการรวบรวมข้อมูลที่เกี่ยวข้องกับ Non-functional requirements
- ข้อจำกัด คือ ใช้เวลานานในการเก็บรวบรวมข้อมูล เพื่อให้ได้มาถึงความต้องการ ซึ่งยากในการตีความ และยากต่อการทำความเข้าใจ

4.3.3 แบบสอบถาม (Questionnaires)

เทคนิคนี้จะประกอบด้วย รายการคำถามที่เฉพาะเจาะจง เพื่อเลือก Stakeholders กำหนดกลุ่มเป้าหมายที่มีขนาดใหญ่ ลักษณะของแบบสอบถามจะต้องมีคุณภาพ เพราะแบบสอบถามถือว่าเป็นประโยชน์อย่างมาก ในการช่วยเตรียมความพร้อมสำหรับการสัมภาษณ์

ลักษณะของแบบสอบถามที่ดี

แต่ละคำถามต้องเป็นคำถามในบริบทสั้นๆ ต้องการคำตอบที่เป็นมาตรฐาน เป็นที่ยอมรับในขั้นต้น (Pre-established) จากคำตอบที่คิดว่าสามารถเป็นไปได้ โดยทั่วไปจะเป็นคำถามแบบหลายทางเลือก (Multiple choices) คือ เลือกคำตอบที่ถูกต้องมาเพียงแค่คำตอบเดียว จากรายการของคำตอบ ในการทำแบบสอบถามที่เป็นการประเมินควรมีแบบที่เป็นสัดส่วนถ่วงดุลน้ำหนัก (Weighted) เช่น Very high, High, Low และควรมีการออกแบบแบบสอบถามอย่างระมัดระวัง

- ข้อดีของ Questionnaires ทำให้ได้มาซึ่งข้อมูลแบบ Subjective information แบบสอบถามจะครอบคลุมคำถาม และคำตอบที่สามารถเป็นไปได้ วิธีนี้ใช้ต้นทุนในการทำงานค่อนข้างต่ำ
- ข้อจำกัดข้อมูลที่ได้มาอาจเกิดจากอคติ (Bias) จากผู้ตอบทำให้ความรู้ที่ได้มีการเบี่ยงเบนจากสิ่งที่ควรจะเป็น มีผู้คนแค่บางส่วนเท่านั้นที่เต็มใจทำแบบสอบถาม

4.3.4 การ์ดซอร์ท (Card sorts for concept driven acquisition)

Stakeholders จะได้รับการ์ดชุดหนึ่ง (Cards) เป็นชุดของบัตรที่แสดงข้อความบางส่วน หรือภาพ หรือทั้งสองอย่าง โดยบัตรจะถูกให้จัดกลุ่มตามหลักเกณฑ์ และมีการอธิบายเหตุผลในแต่ละกลุ่ม

ที่มา https://www.steptwo.com.au/papers/kmc_cardsortingoptions/ วันที่สืบค้น 20 กรกฎาคม 2566.

ภาพที่ 4.4 Card sort : การ์ด

ที่มา https://www.pinterest.com/pin/154740937168766002/ วันที่สืบค้น 20 กรกฎาคม 2566.

ภาพที่ 4.5 Card sort : จัดกลุ่ม

- ข้อดีของการใช้เทคนิค Card sort คือ ราคาถูก ง่ายต่อการใช้ และบางครั้งมีประสิทธิภาพในการเก็บข้อมูลบางอย่างที่ขาดหายไป
- ข้อเสีย คือ อาจจะไม่รับประกันความถูกต้อง

4.3.5 การวิเคราะห์ความต้องการจาก Storyboards & Scenarios (Storyboards and scenarios for problem world exploration)

1. Storyboards

เพื่อสกัดความต้องการ หรือตรวจสอบข้อมูลที่เป็นประโยชน์ บนพื้นฐานของตัวอย่างที่มีความชัดเจน เป็นรูปธรรม ใช้การบรรยาย การเล่าเรื่องราวหรือที่เรียกกันว่า Storyboards บอกเล่าเรื่องราวของ System-as-is และ System-to-be ให้ออกมาเป็นเรื่องราวในลำดับภาพรวม แต่ละภาพรวมอาจเป็นตัวแทนในส่วนของการนำเสนอแบบรวบรัด เพื่อให้ทำความเข้าใจได้ง่าย ยกตัวอย่างเช่น การสเก็ตภาพ ทำสไลด์โชว์ ทำการสำรวจปัญหา (Exploring) โดยการใช้ คำถามประเภท Who, What, Why, How, What if แบ่งเป็น 2 ประเภท คือ

- **Passive mode** : Stakeholders เป็นผู้บอกเล่าเรื่องราว ซึ่ง Storyboard จะถูกนำมาใช้ในการช่วยอธิบาย หรือการตรวจสอบว่าใช่หรือไม่
- **Active mode** : Stakeholders มีส่วนร่วมในเรื่องราวนั้น ๆ ซึ่ง Storyboard ถูกนำมาใช้สำหรับการทำงานร่วมกัน ตรงไปตรงมา

2. Scenarios

Scenarios ถูกนำมาใช้กันอย่างแพร่หลายในหลาย ๆ ขั้นตอนของ Software lifecycle เพื่อใช้ในการอธิบายว่า System-as-is ทำงานอย่างไร เช่น ใช้เพื่อเล่าลำดับเหตุการณ์ในงานจริง ๆ (Real life interaction sequences) ใช้เพื่อสำรวจ/วิเคราะห์ (Explore) วิธีการที่ System-to-be ควรจะทำ และใช้เพื่ออธิบายลำดับเหตุการณ์สมมติ ในส่วนของการปฏิสัมพันธ์กันระหว่างวัตถุ 2 วัตถุ (Hypothetical interaction sequences) เป็นต้น ซึ่งเป็นวิธีพื้นฐานที่ดีสำหรับการสกัดความต้องการ แบ่งเป็น 4 ประเภท คือ

- **Positive scenarios** : แสดงให้เห็นสิ่งที่ควรเกิดขึ้นเป็นพฤติกรรมหนึ่งๆ ที่ระบบควรแสดงอย่างครอบคลุม
- **Negative scenarios** : แสดงให้เห็นถึงสิ่งต่างๆ ที่ระบบสารสนเทศไม่สามารถทำงานได้ หรือไม่สามารถตอบสนองได้
- **Normal scenarios** : เป็นการทำให้เห็นภาพลำดับเหตุการณ์ทั้งหมดของการมีปฏิสัมพันธ์ ที่ทุกอย่างเป็นไปตามที่คาดหวังไว้
- **Abnormal scenarios** : เป็นการทำให้เห็นภาพลำดับความต้องการของการมีปฏิสัมพันธ์ภายใต้สภาวะแวดล้อมที่ผิดปกติ

4.3.6 ต้นแบบภาพร่างและการประมาณการความคิดเห็น (Mockup and prototype for early feedback)

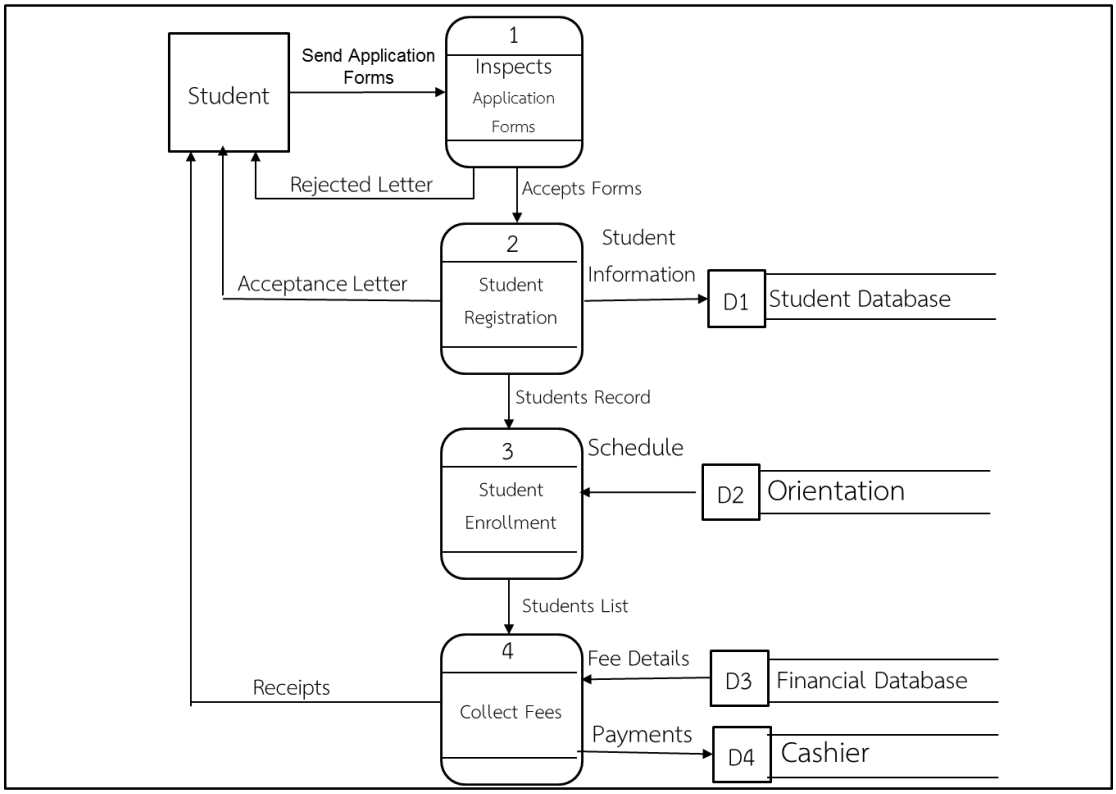
Stakeholders พบว่าการอธิบายถึงรายละเอียดของระบบต้นฉบับ (Textual system) นั้นเป็นเรื่องยาก เพื่ออธิบายรายละเอียดของ Textual system ให้ง่ายขึ้น ควรแสดงภาพร่างของโครงการ เพื่อให้เห็นถึงภาพรวมของระบบ ภาพร่างของ Software product ของการดำเนินการอาจช่วยให้เข้าใจลักษณะบางส่วน และชี้แจงผู้อื่นให้เห็นถึงคุณลักษณะที่ไม่เพียงพอ หรือขาดหายไป

การสร้าง Prototype เป็นเครื่องมือหนึ่งในการสกัดความต้องการเป็นตัวดำเนินการของ System-to-be ซึ่งมีเป้าหมายเพื่อให้วิศวกรความต้องการได้รับ Feedback จาก Stakeholders ก่อนที่จะมีการสกัดความต้องการเพิ่มเติม รวมถึง Software prototype จะช่วยให้ข้อกำหนดมีความชัดเจนมากขึ้น มี 2 ชนิด คือ

- Functional prototype
- User interface prototype

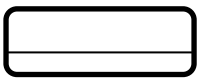
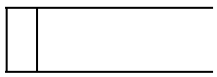
1. **Functional prototype**

จะแสดงในด้านที่เกี่ยวข้องกับการทำงานของซอฟต์แวร์ เช่น แผนภาพ Data Flow Diagrams (DFDs) แผนภาพ Use case



ภาพที่ 4.6 ตัวอย่างแผนภาพ Data Flow Diagrams (DFDs)

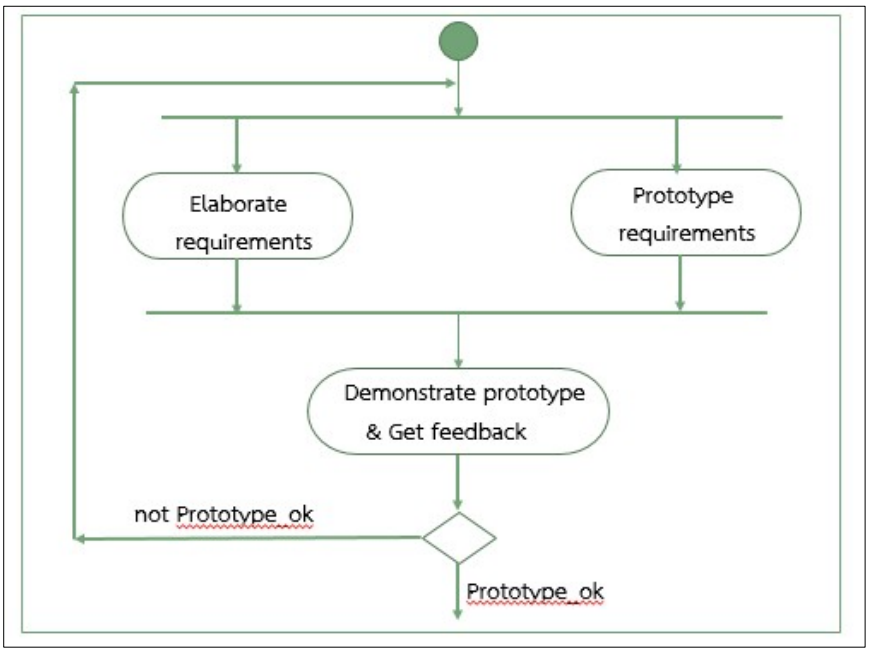
จากแผนภาพแสดงการไหลของข้อมูลของระบบ บอกให้รู้ว่าข้อมูลมาจากที่ใด ข้อมูลไปที่ใด ข้อมูลเก็บที่ใด และจะเกิดเหตุการณ์ใดบ้างกับข้อมูล โดยมีสัญลักษณ์ ดังนี้

-  สิ่งที่อยู่ภายนอก (External entity)
-  การประมวลผล (Process)
-  แหล่งเก็บข้อมูล (Data store)

2. User interface prototype

แสดงในมุมมองแบบ Static และ Dynamic ของการปฏิสัมพันธ์ระหว่างผู้ใช้กับซอฟต์แวร์ เป็นการจัดรูปแบบข้อมูลที่ถูกบันทึกให้แสดงทางหน้าจอ และอยู่ในรูปแบบของ Dialog box เป็นเครื่องมือพื้นฐานของการวิเคราะห์ความต้องการที่ดีและมีประสิทธิภาพ

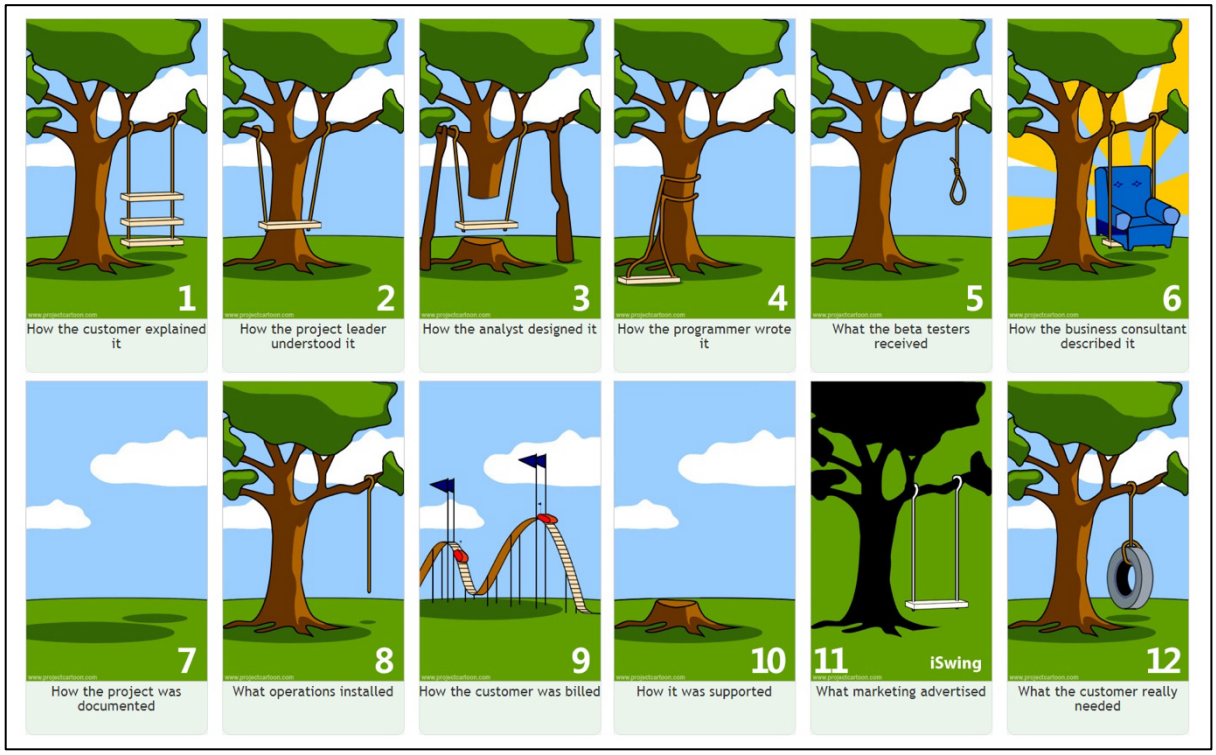
ในการทำ Prototype ให้มีต้นทุนที่คุ้มค่า (Cost effective) ควรใช้ภาษาระดับสูงในการเขียนโปรแกรม อาทิเช่น Logic programming languages เพื่อสามารถสร้าง Prototype ได้อย่างรวดเร็ว



ภาพที่ 4.7 Requirements prototyping

ข้อดี คือ ช่วยสกัดความต้องการ และตรวจสอบได้ดี ทำให้มองเห็นระบบเป็นรูปธรรม ช่วยให้เข้าใจการทำงานของระบบมากขึ้นไม่ยุ่งยาก กรณีที่ต้องปรับเปลี่ยนความต้องการ สามารถค้นพบความต้องการที่ซ่อนอยู่ได้

ข้อจำกัด คือ ไม่ครอบคลุมความต้องการบางลักษณะ โดยเฉพาะ Non-functional requirements และ Prototype ไม่ได้ครอบคลุมทุก ๆ ด้านของ Software-to-be จึงอาจทำให้ Stakeholders เข้าใจผิด และมีความคาดหวังต่อระบบสูงเกินไป ดังเช่นตัวอย่างปัญหาดังภาพที่ 4.8



ที่มา https://pmac-agpc.ca/project-management-tree-swing-story วันที่สืบค้น 20 กรกฎาคม 2566.

ภาพที่ 4.8 ตัวอย่างปัญหาในการพัฒนาซอฟต์แวร์

จากภาพที่ 4.8 แสดงให้เห็นถึงความเข้าใจในความต้องการที่แตกต่างกัน ทั้งในส่วนของ ลูกค้า และวิศวกรความต้องการ มาจนถึงนักพัฒนา สาเหตุที่เป็นต้นตอของปัญหาในการพัฒนาซอฟต์แวร์ที่เจอมากที่สุดก็คือ มาจากความต้องการ

4.3.7 การนำองค์ความรู้กลับมาใช้ซ้ำ (Knowledge reuse)

เป็นการนำองค์ความรู้ทั้งที่เป็นองค์ความรู้ทั่วไป (Reuse of domain-independent knowledge) และความรู้เฉพาะด้านกลับมาใช้ซ้ำ (Reuse of domain-specific knowledge)

4.4 เทคนิคการสกัดในมุมมองของ Stakeholder-driven

เทคนิคในส่วนนี้ต้องอาศัยความรู้เพิ่มเติม ในเรื่องของการมีปฏิสัมพันธ์กับ Stakeholders เพื่อให้ได้รับข้อมูลที่เพียงพอ มีความเกี่ยวข้องกับองค์กร ขอบเขต และปัญหากับ System-as-is โดยเฉพาะการมีปฏิสัมพันธ์โดยตรงกับเจ้าของข้อมูลนั้นๆ เพื่อให้สอดคล้องกับความต้องการของ Stakeholders โดยใช้เทคนิคต่างๆ ดังนี้

4.4.1 การสัมภาษณ์ (Interviews)

การสัมภาษณ์โดยทั่วไปถือว่าเป็นเทคนิคขั้นแรกสำหรับสอบถามความต้องการโดยเลือกเจาะจง Stakeholders ที่มีส่วนเกี่ยวข้อง ที่รู้ขอบเขตของข้อมูลที่ชัดเจน เช่น ผู้จัดการ พนักงาน และผู้ใช้งาน ฯลฯ ทำการเขียนรายงานจากบันทึกการสัมภาษณ์ ตรวจสอบความถูกต้องและความละเอียดของรายงานการสัมภาษณ์ ซึ่งเทคนิคโดยพื้นฐานที่ทำเป็นหลักในการสกัดความต้องการ ทั่วไปการสัมภาษณ์แบ่งได้ 2 วิธี ได้แก่

- การสัมภาษณ์แบบมีโครงสร้างหรือการสัมภาษณ์แบบปิด คือ การสัมภาษณ์โดยมีการเตรียมการตั้งคำถามตามวัตถุประสงค์ที่ระบุไว้
- การสัมภาษณ์แบบไม่มีโครงสร้างหรือการสัมภาษณ์แบบเปิด คือ การสัมภาษณ์โดยไม่มีการเตรียมคำถามไว้ก่อนหรือเป็นการอภิปรายกันอย่างไม่เป็นทางการ

การสัมภาษณ์ที่มีประสิทธิภาพ จะต้องเริ่มตั้งคำถามก่อน และปิดท้ายด้วยการอภิปราย (Open-ended) วัตถุประสงค์ของการสัมภาษณ์ ควรกำหนดผู้สัมภาษณ์ให้เหมาะสม เพื่อให้ได้ผลลัพธ์ที่น่าเชื่อถือและมองเห็นภาพได้ง่าย เตรียมการสัมภาษณ์สามารถมุ่งเน้นไปที่ปัญหาให้มีความเหมาะสมสำหรับผู้ให้สัมภาษณ์และถูกกาลเทศะ จากนั้นเริ่มการสัมภาษณ์ให้ผู้ให้สัมภาษณ์มีความรู้สึกสบายใจที่จะถูกตั้งคำถาม (Comfortable) โดยการพยายามสร้างความสัมพันธ์อันดีกับผู้ให้สัมภาษณ์ ทำให้เหมือนเป็น Partner กันตลอดระยะการสัมภาษณ์ และควรจับใจความสำคัญของข้อกังวลและปัญหาในงานของผู้ให้สัมภาษณ์ ถามคำถามแบบ Open-ended จนจบการสัมภาษณ์ ควรหลีกเลี่ยงคำถามที่มีลักษณะ ดังนี้ มีความลำเอียง ฟังดูแล้วไม่ฉลาด ได้คำตอบมาแล้ว และถามแล้วไม่ได้คำตอบจากผู้ให้สัมภาษณ์

4.4.2 การสังเกตและการศึกษาแบบตั้งสมมติฐาน (Observation and ethnographic studies)

เทคนิคนี้เป็นหลักฐานที่ทำให้ทั้งทีมพัฒนา และผู้คนที่เกี่ยวข้องเกิดความเข้าใจได้โดยการสังเกต แบ่งออกเป็น โดยทางอ้อม (Passive observation) และโดยทางตรง (Active observation)

การสังเกตแบบไปเรื่อย ๆ (Passive) วิศวกรความต้องการจะไม่ยุ่งเกี่ยวกับผู้ที่ปฏิบัติงาน โดยจะทำได้เพียงพิจารณาหรือสังเกตจากภายนอก และบันทึกสิ่งที่เกิดขึ้นผ่านกล้องวีดีโอ ในขณะที่ทำการเก็บรวบรวมข้อมูลที่บันทึกได้มานั้นจะต้องนำมาแปลความหมาย หรือการทำความเข้าใจ (ตีความ)

การสังเกตแบบคล่องแคล่ว (Active) วิศวกรความต้องการจะเข้าไปมีส่วนร่วมกับการทำงานขององค์กรที่ไปสังเกตการณ์ ในบางครั้งผู้ถูกสัมภาษณ์ อาจกลายเป็นส่วนหนึ่งของทีมผู้พัฒนาซอฟต์แวร์

- ข้อดีของ Observation and ethnographic studies คือ สามารถค้นพบความรู้บางอย่างที่ไม่ปรากฏในเทคนิคอื่น ๆ มุ่งเน้นการทำความเข้าใจการทำงานของระบบเดิมที่ผู้ใช้กำลังใช้งานอยู่ในปัจจุบัน ซึ่งวิธีนี้จะชี้ให้เห็นปัญหาและโอกาสที่จะแก้ไขกับ System-to-be ได้ มีการสร้างความสัมพันธ์ที่ดี มีความไว้วางใจในการซักถาม และตรวจสอบความถูกต้องของเนื้อหากับผู้ที่ปฏิบัติงานจริง
- ข้อจำกัดของ Observation and ethnographic studies คือ ผู้สังเกตการณ์จะต้องมีประสบการณ์ อาจต้องเสียค่าใช้จ่ายในการดำเนินการรวบรวมความต้องการ การสังเกตการณ์อาจเกิดขึ้นในช่วงเวลาที่เฉพาะในช่วงเวลาที่แตกต่างกัน และภายใต้เงื่อนไขการทำงานที่แตกต่างกัน ดังนั้นข้อสรุปอาจจะไม่ถูกต้อง ขณะที่ทำการสังเกตการณ์ แนวโน้มการทำงานอาจจะแตกต่างไปจากเดิมเมื่อเวลาเปลี่ยนไป ทำให้การวิเคราะห์ และประเมินคุณสมบัติที่ผิดพลาด ซึ่งเกิดความเข้าใจผิดในการทำความเข้าใจ

4.4.3 การประชุมกลุ่ม (Group session)

เป็นเทคนิคของการทำงานร่วมกันภายใต้สมติฐานเดียวกัน รวมถึงการทำความเข้าใจ และมีการตัดสินใจร่วมกันเพื่อที่จะใช้เวลาในการพัฒนาซอฟต์แวร์ให้น้อยลง โดยในแต่ละกิจกรรมจะมีผู้รับผิดชอบ และการนำเสนอเพื่อพิจารณาความถูกต้องของซอฟต์แวร์สามารถทำได้ 2 วิธี

- การประชุมกลุ่มแบบมีโครงสร้าง
- การประชุมกลุ่มแบบไม่มีโครงสร้าง

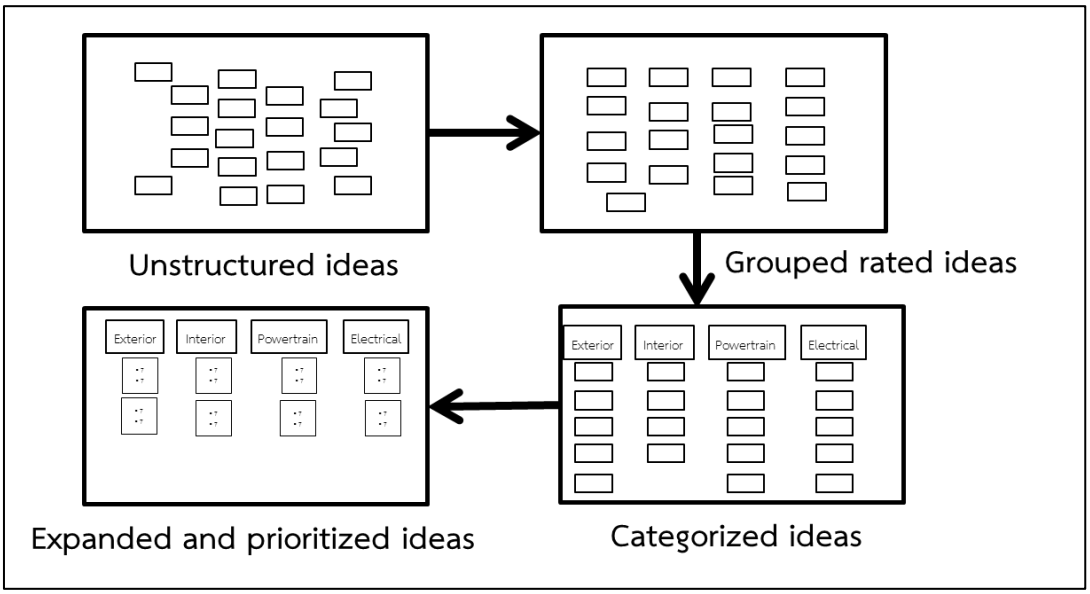
1. การประชุมกลุ่มแบบมีโครงสร้าง

ผู้มีส่วนร่วมแต่ละคนจะถูกกำหนดบทบาทหน้าที่ไว้อย่างชัดเจน เช่น หัวหน้า ผู้ดูแล ผู้รายงาน ผู้ใช้งาน และผู้จัดการ หรือนักพัฒนา โดยจะทำงานร่วมกันเพื่อให้ได้ความต้องการที่ละเอียดตามบทบาทของแต่ละหน้าที่ ซึ่งอาจมีมุมมองที่ต่างกัน ดังนั้นการเพิ่มเติมข้อมูลดังกล่าว ควรมุ่งเน้นไปที่คุณสมบัติที่แท้จริงของซอฟต์แวร์ การประสานงานภายในกลุ่มที่จะเกิดขึ้น อาจมีการใช้เทคนิคต่างๆ ในการดำเนินงาน เช่น การร่วมกันพัฒนาโปรแกรม (Joint Application Development : JAD) หรือการปรับปรุงคุณภาพของการทำงาน (Quality Function Deployment : QFD)

2. การประชุมกลุ่มแบบไม่มีโครงสร้าง

ในการประชุมกลุ่มที่ไม่มีโครงสร้าง หรือที่เรียกว่า การระดมสมอง ผู้เข้าร่วมจะได้รับการกำหนดบทบาทไว้อย่างชัดเจน โดยมีขั้นตอน ดังนี้

- **ขั้นตอนแรก** ผู้เข้าร่วมประชุมจะต้องแสดงความคิดเห็นให้มากที่สุดเท่าที่จะเป็นไปได้เกี่ยวกับการปรับปรุงงานหรือปัญหาที่มีอยู่ และการแสดงความคิดเห็นจะต้องเป็นอิสระ โดยปราศจากอคติ
- **ขั้นตอนที่สอง** ผู้มีส่วนร่วมต้องทำงานร่วมกันเพื่อประเมินความชัดเจนของข้อมูล ภายใต้กฎเกณฑ์ที่ตกลงร่วมกัน เช่น ประสิทธิภาพ ค่าใช้จ่ายในการเปลี่ยนแปลงความคิด และการจัดลำดับความสำคัญ



ภาพที่ 4.9 ขั้นตอนของการระดมสมอง Stages of brainstorming session

จากภาพที่ 4.9 เป็นภาพที่อธิบายถึงขั้นตอนการระดมสมอง โดยรูป Unstructured ideas อธิบายเกี่ยวกับการช่วยกันระดมความคิดซึ่งได้มาเป็นแนวคิดที่ยังไม่มีโครงสร้าง โดย Grouped rated ideas เป็นการจัดกลุ่มแนวคิดที่คล้ายกันหรือเหมือนกัน Categorized ideas เป็นการจัดประเภทของแนวคิด และ Expanded and prioritized ideas เป็นการเรียงลำดับความสำคัญของแนวคิด

- ข้อดีของ Group session คือ ในการประชุมกลุ่มจะมีประโยชน์เป็นอย่างมาก มีลักษณะที่ไม่ค่อยมีความเป็นทางการมีปฏิสัมพันธ์กันระหว่างการประชุมจะทำให้ผู้เข้าร่วมประชุมกล้าที่จะเปิดเผยปัญหาของระบบเดิม (System-as-is) หรือปัญหาที่อาจจะเกิดในระบบใหม่ (System-to-be) ในความเห็นต่าง หรือมีความเห็นเหมือนกัน ล้วนส่งผลต่อการสกัดความต้องการต่อ System-to-be ทั้งสิ้น เพื่อได้ข้อยุติในเวลาอันรวดเร็ว

- ข้อจำกัดของ Group session คือ ภาคส่วนที่เกี่ยวข้องอาจมีเวลาว่างที่ไม่ตรงกัน ทำให้การนัดหมายประชุมเป็นกลุ่มอาจทำได้ยากหรือถ้าทำได้ องค์ประชุมก็มักจะไม่ครบถ้วนตามที่ควรจะเป็น วิศวกรความต้องการต้องมีประสบการณ์ รวมถึงรู้จักเทคนิค และทักษะการสื่อสารเป็นอย่างดี จึงจะสามารถสกัดความต้องการที่แท้จริงได้
- ความเสี่ยงที่เกี่ยวข้องกับกลุ่มคน/ผู้แทนการประชุมอาจเปลี่ยนไป ไม่ซ้ำเดิม ในแต่ละครั้งของการประชุม อาจส่งผลต่อความมือคติ และความไม่เพียงพอหรือไม่สมบูรณ์ของข้อมูล โดยเฉพาะอย่างยิ่งการอยู่ภายใต้อิทธิพลของคนบางคน มีความยากลำบากที่จะติดต่อสื่อสารกับคนอื่น ๆ โครงสร้างการประชุมที่ไม่เหมาะสม อาจทำให้เกิดความยากลำบากในการทำงานให้เป็นรูปธรรม และเสียเวลาไปโดยเปล่าประโยชน์

4.5 การรวบรวมความต้องการจากแหล่งอื่นๆ

ความต้องการหาได้โดยตรงจาก Stakeholders ส่วนใหญ่มาจากการสัมภาษณ์ และการประชุม มีหลายแหล่งที่สามารถนำมาเป็นความต้องการ ในบทนี้ RE จะพิจารณาสิ่งเหล่านี้ นอกจากนี้ยังมองไปที่ความต้องการของผู้ใช้ในโครงการ ซึ่งเป็นแหล่งข้อมูลที่เป็นไปได้ (Possible sources) หลักที่สำคัญในการรวบรวมความต้องการขึ้นอยู่กับผู้ใช้งาน และบางครั้งอาจเป็น Stakeholders คนอื่นๆ เอกสาร และข้อมูลจากแหล่งอื่นๆ จะสามารถบอกความต้องการของผู้ใช้ได้ Stakeholders เท่านั้น ที่จะสามารถบอกได้ว่าความต้องการที่แท้จริงคืออะไร โดยจะรวบรวมความต้องการจากแหล่งอื่นๆ จากแหล่งข้อมูลที่เป็นไปได้ ดังนี้

4.5.1 รายงานต่าง ๆ ที่เกี่ยวกับปัญหา (Problem reports)

องค์กรที่มีประสิทธิภาพจะมีแบบฟอร์มสำหรับการรายงานปัญหาที่เกิดขึ้นหรือข้อบกพร่องซอฟต์แวร์ ซึ่งปัญหาที่รายงานโดยผู้ใช้จะสามารถนำมาเป็นความต้องการได้ สิ่งที่ผู้พัฒนาควรทำอันดับแรก คือ การจัดเรียงรายงานเหล่านั้นให้เป็นกลุ่มเพื่อระบุปัญหาสำคัญของผู้ใช้ ผู้พัฒนาสามารถสอบถามผู้ใช้เกี่ยวกับปัญหาให้ชัดเจน เพื่อชี้แจงความต้องการที่แท้จริงของผู้ใช้

4.5.2 ทีมช่วยเหลือและทีมสนับสนุน (Helpdesk and support team)

องค์กรที่ให้บริการซอฟต์แวร์จะมีทีมช่วยเหลือ ทำหน้าที่ในการเก็บบันทึกข้อมูลการแก้ไขปัญหา และทีมสนับสนุนทำหน้าที่ในการแก้ไขปัญหา หลายองค์กรจะมีสิ่งอำนวยความสะดวกที่คล้ายกัน เพื่อสนับสนุนการดำเนินงานของตนเอง ช่วยให้นำไปสู่การเก็บความต้องการที่ดี และช่วยให้ผู้พัฒนาประหยัดเวลา หากทีมช่วยเหลือและทีมสนับสนุนบอกความต้องการไม่ได้ อาจจะต้องมีการฝึกอบรมหรือฝึกใช้เครื่องมือที่ใช้ในการเก็บรวบรวมความต้องการ

เอกสารอ้างอิง

1. A. Lamsweerde, "Requirements Engineering from System Goals to UML Models to Software Specifications (Chapter 2)", Wiley, 2009, ISBN 978-0-470-01270-3.
2. B. Berenbach, J. Kazmeier, D. J. Paulish, and A. Rudorfer, "Software & Systems Requirements Engineering in Practice (Chapter 3)", McGrawHill, 2009, ISBN 978-0-07-160547-2.
3. G. Kotonya and I. Sommerville, "Requirements Engineering: Processes and Techniques (Chapter 3)", John Wiley & Sons, 1997, ISBN 0-471-97208-8.
4. I. F. Alexander and R. Stevens, "Writing Better Requirements (Chapter 4)", Addison Wesley, 2002, ISBN 0-321-13163-0.

แบบฝึกหัด

ตัวอย่างสถานการณ์ : บริษัท SE-FOOD

บริษัท SE-FOOD เป็นบริษัทขนาดเล็กที่อยู่ในจังหวัดชลบุรี เปิดกิจการทำไอศครีมจากนมแพะยาวนานกว่าเกือบสิบ ดำเนินธุรกิจผลิตไอศครีมจากนมแพะ และส่งขายให้กับร้านค้า ร้านอาหาร และโรงเรียนทั่วทั้งภาคตะวันออกของประเทศไทย โดยมีคุณบุญชู (Office Supervisor) เป็นผู้ประสานงานกิจกรรมทุกอย่างของบริษัท โดยมีพนักงานอีกจำนวน 2-3 คน ในการดำเนินการตามคำสั่งซื้อและใบแจ้งหนี้ของลูกค้า และจัดทำบัญชีค่าใช้จ่ายต่างๆ ที่เกี่ยวข้องกับวัสดุที่ต้องใช้ในการดำเนินงานด้วยโปรแกรม Excel และโปรแกรม COTS Payroll system เพียงเท่านั้น

บริษัทขยายการดำเนินกิจการอย่างรวดเร็ว พร้อมทั้งมีแผนเปิดช่องทางการตลาดและการเพิ่มยอดขายอีก 15% ซึ่ง คุณบุญชู มีความกังวลเกี่ยวกับความสามารถของทีมในการรับมือกับการขยายกิจการในครั้งนี้ จึงพิจารณาความเป็นไปได้ในการพัฒนาระบบไอทีแบบบูรณาการ เพื่ออำนวยความสะดวกในการทำงาน ช่วยในการควบคุมคลังสินค้า (Stock control) โดยนำเรียนเรื่องนี้กับทีมผู้บริหารทั้งหมด (MD, Production Manager, Logistics Manager, Office Supervisor) โดยมีความต้องการของระบบ ดังนี้

1. ลูกค้าใหม่ทั้งหมดจะต้องมีการเพิ่มเข้าไปในระบบได้อย่างง่ายดาย
2. ระบบจะต้องบันทึกใบสั่งซื้อของลูกค้า ถึงประมาณวันละ 50 ความต้องการ และจะถูกเก็บรักษาไว้เป็นเวลาห้าปี
3. เจ้าหน้าที่ที่เกี่ยวข้อง (Office staff) สามารถทำ 1. และ 2. ได้ และลูกค้าสามารถดำเนินการด้วยตัวเองผ่านทางเว็บไซต์
4. ระบบสามารถสร้างใบแจ้งหนี้โดยอัตโนมัติตามใบสั่งซื้อ
5. เจ้าหน้าที่สำนักงาน (Office staff) จะต้องสามารถแก้ไขรายละเอียดของการสั่งซื้อและใบแจ้งหนี้ได้
6. ส่งออกรายงานประจำสัปดาห์ของใบแจ้งหนี้ที่ค้างชำระ
7. เจ้าหน้าที่สำนักงานจะต้องสามารถพิมพ์ใบแจ้งหนี้หรือส่งอีเมลให้กับลูกค้า
8. ผู้จัดการฝ่ายผลิต (Production Manager) สามารถดูคำสั่งซื้อของลูกค้า เพื่อให้สามารถวางแผนการผลิตได้
9. การชำระเงินในใบแจ้งหนี้ควรทำได้หลายวิธี
10. ผู้บริหาร (MD) ต้องการดูการใช้จ่ายเงินผ่านระบบ Payroll system

คำสั่ง ให้นิสิตวิเคราะห์ ตัวอย่างสถานการณ์ : บริษัท SE-FOOD พร้อมระบุว่าผู้ใดบ้างที่มีส่วนได้ส่วนเสีย เทคนิคที่ใช้ในการสกัดความต้องการที่สอดคล้องกับผู้มีส่วนได้ส่วนเสีย และมีประเด็นคำถามใดบ้าง

#	ผู้มีส่วนได้ส่วนเสีย	เทคนิคที่ใช้ในการสกัดความต้องการ	ประเด็นคำถาม
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			
11.			
12.			
13.			
14.			
15.			
16.			
17.			
18.			
19.			
20.			
21.			
22.			
23.			
24.			
25.			
26.			
27.			

แผนบริหารการสอน
บทที่ 5 การประเมินความต้องการ

เนื้อหา

1. ประเภทของความไม่สอดคล้องกัน/ขัดแย้งกันของความต้องการ
2. การจัดการความไม่สอดคล้องกัน/ขัดแย้งกันของความต้องการ
3. การจัดลำดับความสำคัญความต้องการ (Requirements prioritization)

ผลลัพธ์การเรียนรู้รายบทเรียน

1. นิสิตมีความรู้และเข้าใจในขอบเขตและการสกัดความต้องการ
2. นิสิตมีความรู้และเข้าใจการประเมินความต้องการ

วิธีการสอน

1. บรรยาย
2. ยกตัวอย่างระบบ โดยใช้กรณีตัวอย่างระบบที่เป็นที่รู้จัก
3. นิสิตร่วมอภิปรายแสดงความคิดเห็น

สื่อการสอน

1. แฟ้มดิจิทัลนำเสนอ
2. ระบบมัลติมีเดีย เช่น คอมพิวเตอร์ เครื่องเสียง เครื่องฉายภาพ
3. เอกสารประกอบคำสอน

การวัดและการประเมินผล

1. คะแนนการเข้าชั้นเรียน
2. คะแนนแบบฝึกหัด
3. คะแนนการอภิปรายของนิสิต

บทที่ 5 การประเมินความต้องการ

ในบทนี้จะกล่าวถึงขั้นตอนที่สำคัญต่อความสำเร็จ หรือความล้มเหลวของระบบหรือโครงการ ซึ่งเป็นการวิเคราะห์และประเมินความต้องการในหลายบริบทตัวอย่างเช่น

- ความต้องการของผู้มีส่วนได้ส่วนเสียบางคนอาจไม่สอดคล้องกับคนอื่นๆ โดยเฉพาะในกรณีที่ผู้มีส่วนได้ส่วนเสียมีหลายเป้าหมายและข้อกังวลที่มาก จึงจำเป็นที่ต้องมีการตรวจสอบและแก้ไขความไม่สอดคล้องกันเช่น ในกรณีที่มีข้อขัดแย้งกันจะต้องมีการจัดการให้เกิดเจรจาต่อรอง เพื่อให้ทุกฝ่ายยอมรับ
- ตัวเลือกของทางเลือกที่อาจจะต้องมีการนำมาเปรียบเทียบ เพื่อเลือกตัวเลือกดีที่สุดสำหรับระบบ

5.1 ประเภทของความไม่สอดคล้องกัน/ขัดแย้งกันของความต้องการ

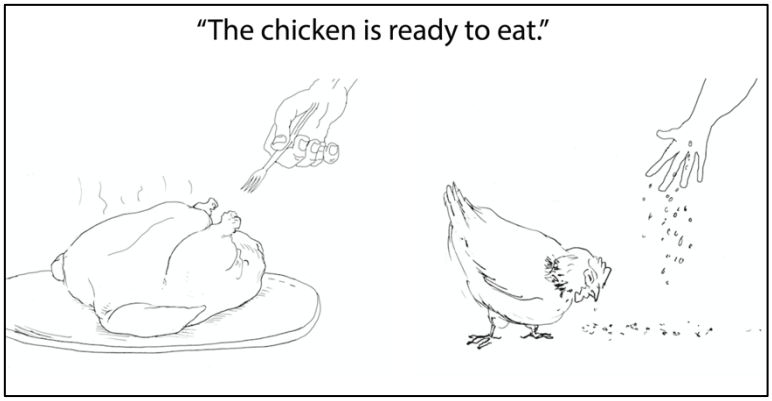
วิศวกรความต้องการต้องเผชิญกับความขัดแย้ง ความขัดแย้งเกิดจากความกังวนหรือมีมุมมองที่หลากหลาย ซึ่งความขัดแย้งนี้จะต้องถูกตรวจพบและแก้ไขในที่สุด ในส่วนนี้จะเป็นการกำหนดชนิดของความไม่สอดคล้องหรือความขัดแย้งที่วิศวกรความต้องการอาจจะพบในช่วงของการเก็บความต้องการของโครงการเพื่อใช้สำหรับจัดการกับความขัดแย้งนั้นๆ

5.1.1 คำศัพท์ที่ขัดแย้งกัน (Terminology clash)

ความต้องการมีแนวคิดเดียวกัน แต่อาจจะถูกตั้งชื่อหรือใช้ข้อความที่แตกต่างกันไปตามการทำงานที่ต่างกัน ตัวอย่างเช่น “**จัดทำ**เอกสารความต้องการ” กับ “**เขียน**เอกสารความต้องการ” มีความหมายก็คือการนำความต้องการที่ได้จากลูกค้ามาจัดทำเป็นเอกสารความต้องการ

5.1.2 การตั้งชื่อที่ขัดแย้งกัน (Designation clash)

ความต้องการตั้งชื่อหรือใช้ข้อความเดียวกัน แต่มีความหมายที่แตกต่างกันตามการทำงานที่ต่างกัน ตัวอย่างเช่น “The chicken is **ready to eat**” หมายถึงไก่พร้อมที่จะกินอาหาร หรือไก่ที่เป็นอาหารพร้อมทานแล้ว ดังภาพที่ 5.1



ภาพที่ 5.1 The chicken is ready to eat

5.1.3 โครงสร้างที่ขัดแย้งกัน (Structure clash)

ความต้องการมีแนวคิดเดียวกัน แต่อาจได้รับโครงสร้างที่แตกต่างกันตามการทำงานที่แตกต่างกัน เช่น “ต้องจ่ายเงินค่าลงทะเบียนเข้าร่วมกิจกรรมภายใน 7 วันหลังทำการลงทะเบียน” กับ “ต้องจ่ายเงินค่าลงทะเบียนก่อนเวลา 23.59 น. ของวันที่ 6 หลังจากวันลงทะเบียน”

5.1.4 ความขัดแย้งที่รุนแรง (Strong conflict)

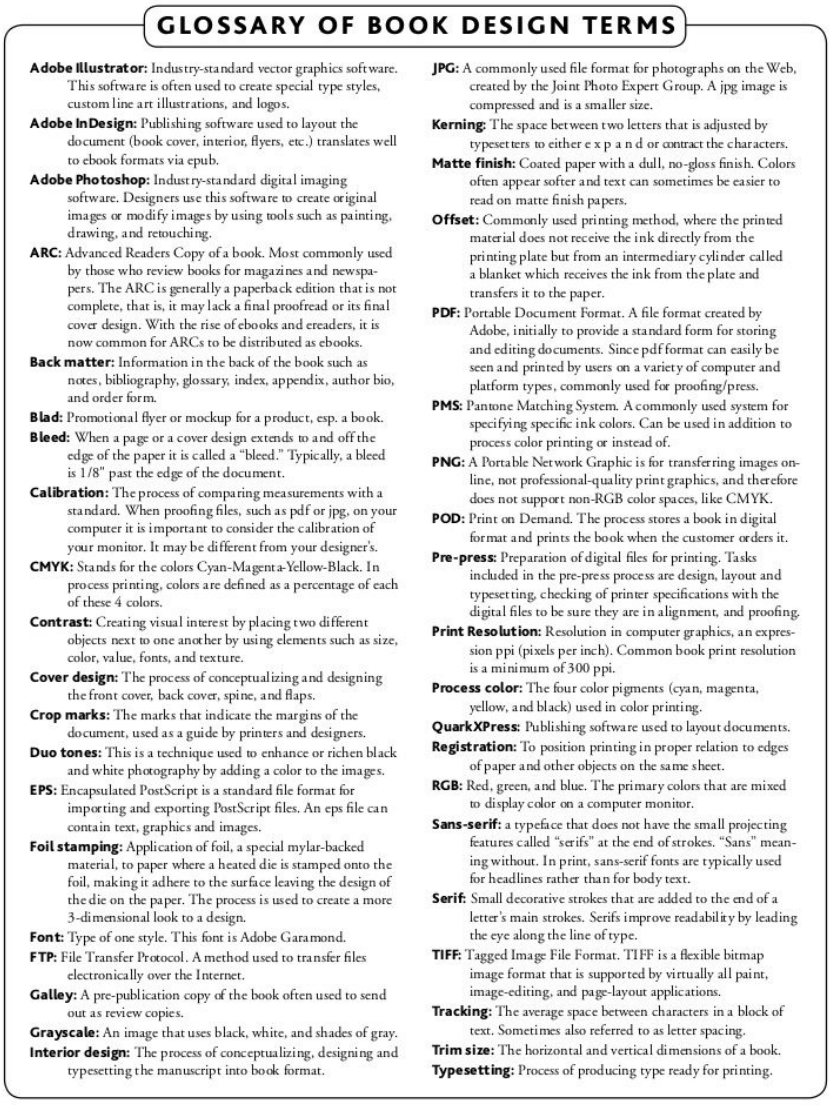
เกิดจากแนวคิดที่ไม่ตรงกัน หรือมีข้อขัดแย้งกันอย่างรุนแรง โดยนักวิศกรความต้องการไม่สามารถตัดสินใจได้ ตัวอย่างเช่น ในช่วงตอนต้นของเอกสาร RD ระบุว่าให้ผู้ใช้งานระบบต้องเป็นสมาชิกและ Login เข้าสู่ระบบจึงจะใช้งานระบบได้ แต่เมื่ออ่านเอกสารต่อไปกลับพบว่ามีการระบุให้ผู้ใช้ทั่วไปสามารถใช้งานระบบได้เหมือนกัน

5.1.5 ความแตกต่างหรือความขัดแย้งที่ไม่รุนแรง (Weak conflict or divergence)

ความต้องการมีความขัดแย้งที่ไม่รุนแรงมากนัก แต่จะสามารถหาข้อยุติได้ต้องขึ้นอยู่กับสถานการณ์และสภาพแวดล้อม ในช่วงต้นของเอกสาร RD ระบุไว้ว่าให้ผู้ที่คืนหนังสือจ่ายค่าปรับเมื่อคืนหนังสือไม่ตรงเวลา 5 บาทต่อ 1 วัน แต่กลับพบว่าหน้าหนึ่งของเอกสารระบุว่าให้ผู้ที่คืนหนังสือจ่ายค่าปรับเมื่อคืนหนังสือไม่ตรงเวลา 3 บาทต่อ 1 วัน

5.2 การจัดการความไม่สอดคล้องกัน/ขัดแย้งกันของความต้องการ

ความขัดแย้งกันของคำศัพท์ การตั้งชื่อ และโครงสร้าง จะจัดการได้อย่างมีประสิทธิภาพที่สุด โดยการใช้คำศัพท์อย่างระมัดระวังและรอบคอบ เช่น นิยามคำศัพท์ทั้งหมดที่นำมาใช้ให้ชัดเจน เป็นที่ยอมรับและเข้าใจตรงกันทุกฝ่าย ควรจะทำหมวดหมู่ของคำศัพท์ (Glossary of terms)



GLOSSARY OF BOOK DESIGN TERMS

Adobe Illustrator: Industry-standard vector graphics software. This software is often used to create special type styles, custom line art illustrations, and logos.

Adobe InDesign: Publishing software used to layout the document (book cover, interior, flyers, etc.) translates well to ebook formats via epub.

Adobe Photoshop: Industry-standard digital imaging software. Designers use this software to create original images or modify images by using tools such as painting, drawing, and retouching.

ARC: Advanced Readers Copy of a book. Most commonly used by those who review books for magazines and newspapers. The ARC is generally a paperback edition that is not complete, that is, it may lack a final proofread or its final cover design. With the rise of ebooks and ereaders, it is now common for ARCs to be distributed as ebooks.

Back matter: Information in the back of the book such as notes, bibliography, glossary, index, appendix, author bio, and order form.

Blad: Promotional flyer or mockup for a product, esp. a book.

Bleed: When a page or a cover design extends to and off the edge of the paper it is called a "bleed." Typically, a bleed is 1/8" past the edge of the document.

Calibration: The process of comparing measurements with a standard. When proofing files, such as pdf or jpg, on your computer it is important to consider the calibration of your monitor. It may be different from your designer's.

CMYK: Stands for the colors Cyan-Magenta-Yellow-Black. In process printing, colors are defined as a percentage of each of these 4 colors.

Contrast: Creating visual interest by placing two different objects next to one another by using elements such as size, color, value, fonts, and texture.

Cover design: The process of conceptualizing and designing the front cover, back cover, spine, and flaps.

Crop marks: The marks that indicate the margins of the document, used as a guide by printers and designers.

Duo tones: This is a technique used to enhance or richen black and white photography by adding a color to the images.

EPS: Encapsulated PostScript is a standard file format for importing and exporting PostScript files. An eps file can contain text, graphics and images.

Foil stamping: Application of foil, a special mylar-backed material, to paper where a heated die is stamped onto the foil, making it adhere to the surface leaving the design of the die on the paper. The process is used to create a more 3-dimensional look to a design.

Font: Type of one style. This font is Adobe Garamond.

FTP: File Transfer Protocol. A method used to transfer files electronically over the Internet.

Galley: A pre-publication copy of the book often used to send out as review copies.

Grayscale: An image that uses black, white, and shades of gray.

Interior design: The process of conceptualizing, designing and typesetting the manuscript into book format.

JPG: A commonly used file format for photographs on the Web, created by the Joint Photo Expert Group. A jpg image is compressed and is a smaller size.

Kerning: The space between two letters that is adjusted by typesetters to either e x p a n d or contract the characters.

Matte finish: Coated paper with a dull, no-gloss finish. Colors often appear softer and text can sometimes be easier to read on matte finish papers.

Offset: Commonly used printing method, where the printed material does not receive the ink directly from the printing plate but from an intermediary cylinder called a blanket which receives the ink from the plate and transfers it to the paper.

PDF: Portable Document Format. A file format created by Adobe, initially to provide a standard form for storing and editing documents. Since pdf format can easily be seen and printed by users on a variety of computer and platform types, commonly used for proofing/press.

PMS: Pantone Matching System. A commonly used system for specifying specific ink colors. Can be used in addition to process color printing or instead of.

PNG: A Portable Network Graphic is for transferring images online, not professional-quality print graphics, and therefore does not support non-RGB color spaces, like CMYK.

POD: Print on Demand. The process stores a book in digital format and prints the book when the customer orders it.

Pre-press: Preparation of digital files for printing. Tasks included in the pre-press process are design, layout and typesetting, checking of printer specifications with the digital files to be sure they are in alignment, and proofing.

Print Resolution: Resolution in computer graphics, an expression ppi (pixels per inch). Common book print resolution is a minimum of 300 ppi.

Process color: The four color pigments (cyan, magenta, yellow, and black) used in color printing.

QuarkXPress: Publishing software used to layout documents.

Registration: To position printing in proper relation to edges of paper and other objects on the same sheet.

RGB: Red, green, and blue. The primary colors that are mixed to display color on a computer monitor.

Sans-serif: a typeface that does not have the small projecting features called "serifs" at the end of strokes. "Sans" meaning without. In print, sans-serif fonts are typically used for headlines rather than for body text.

Serif: Small decorative strokes that are added to the end of a letter's main strokes. Serifs improve readability by leading the eye along the line of type.

TIFF: Tagged Image File Format. TIFF is a flexible bitmap image format that is supported by virtually all paint, image-editing, and page-layout applications.

Tracking: The average space between characters in a block of text. Sometimes also referred to as letter spacing.

Trim size: The horizontal and vertical dimensions of a book.

Typesetting: Process of producing type ready for printing.

ภาพที่ 5.2 หมวดหมู่ของคำศัพท์

อย่างไรก็ตาม การจัดการความขัดแย้งที่รุนแรงและไม่รุนแรง ต้องทราบประเภทของความขัดแย้ง และยังต้องทราบสาเหตุของขัดแย้งด้วย จากนั้นดำเนินการสอบถามไปยัง Stakeholders ที่เกี่ยวข้อง ผ่านการเจรจาต่อรอง เพื่อให้ทุกฝ่ายยอมรับร่วมกัน ในบางครั้งอาจต้องใช้อำนาจตัดสินใจของผู้บริหารระดับสูงหากไม่สามารถตกลงกันได้ โดยทั่วไปสาเหตุของความขัดแย้งมีสองสาเหตุ คือ

1. ผู้มีส่วนได้ส่วนเสียหลายคนมีวัตถุประสงค์ที่แตกต่างกัน บางครั้งวัตถุประสงค์บางอย่างไม่สามารถเข้ากันได้ จึงทำให้เกิดความขัดแย้งกันระหว่างความต้องการ ดังนั้น จึงควรวิเคราะห์วัตถุประสงค์ให้ตรงกันและแก้ปัญหาความขัดแย้งต่างๆ ให้ตรงตามความต้องการ

2. ความขัดแย้งกันระหว่างหลายๆ มุมมอง เช่น
   - ความปลอดภัย (Security) กับความสะดวกสบาย (Usability)
   - ความลับ (Confidentiality) กับความเชื่อใจ (Accountability)
   - การบำรุงรักษา (Maintainability) กับค่าใช้จ่ายในการพัฒนา (Development costs)

5.3 การจัดลำดับความสำคัญความต้องการ (Requirements prioritization)

ความต้องการที่เกิดจากการสกัดความต้องการ (Elicitation) และการประเมินผลความต้องการจะนำไปจัดลำดับความสำคัญอยู่เสมอ เหตุผลที่ต้องมีการจัดลำดับความสำคัญ เนื่องจากทรัพยากรณ์ขององค์กร/บริษัทมีอยู่อย่างจำกัด อาจทำให้ไม่สามารถพัฒนาระบบให้สำเร็จทั้งหมดภายในคราวเดียวกัน ข้อมูลการจัดลำดับความสำคัญสามารถนำไปใช้ในการจัดการความขัดแย้งได้ โดยวิธีที่ง่ายและตรงไปตรงมาที่สุดของการจัดลำดับความสำคัญของความต้องการ คือการรวบรวม Stakeholders ในกระบวนการและให้ Stakeholders เป็นผู้จัดลำดับความต้องการภายใต้ข้อจำกัดต่างๆ ที่ทุก Stakeholders รับทราบและเห็นพ้องต้องกัน ทั้งนี้อาจมีเกณฑ์พิจารณาที่เกี่ยวข้องกับปัจจัยอื่นๆ ประกอบด้วย ดังแสดงในตารางที่ 5.1

ตารางที่ 5.1 เกณฑ์การประเมินความสำคัญ

#	เกณฑ์การประเมิน	รายละเอียด
1.	ความสอดคล้องกับยุทธศาสตร์ (Strategic alignment)	เป็นการประเมินว่าความต้องการนั้นสอดคล้องกับยุทธศาสตร์ขององค์กร ทำให้บรรลุวิสัยทัศน์ พันธกิจ และผลในระยะยาว
2.	การแสดงผลประโยชน์ที่เป็นรูปธรรม (Potential benefit)	การประเมินความต้องการจากประโยชน์ขององค์กร ที่องค์กรจะได้รับ เช่น ลดต้นทุน หรือสร้างกำไรให้องค์กร
3.	ทรัพยากรที่นำมาใช้ได้ (Resource availability)	ประเมินความต้องการจากทรัพยากรณ์ที่นำมาใช้ โดยบางความต้องการอาจใช้ทรัพยากรณ์มากกว่าความต้องการอื่น ซึ่งทรัพยากรณ์หมายรวมถึง บุคคล ต้นทุน และเวลาในการปฏิบัติงาน
4.	ความยากทางเทคนิคและความเสี่ยง (Technical difficulty and risk)	ความต้องการบางข้ออาจมีความซับซ้อนทางเทคนิค ที่จะทำให้โครงการประสบความสำเร็จและความเสี่ยงที่อาจเกิดขึ้น

เอกสารอ้างอิง

1. A. Lamsweerde, "Requirements Engineering from System Goals to UML Models to Software Specifications (Chapter 3)", Wiley, 2009, ISBN 978-0-470-01270-3.
2. B. Berenbach, J. Kazmeier, D. J. Paulish, and A. Rudorfer, "Software & Systems Requirements Engineering in Practice", McGrawHill, 2009, ISBN 978-0-07-160547-2.
3. G. Kotonya and I. Sommerville, "Requirements Engineering: Processes and Techniques", John Wiley & Sons, 1997, ISBN 0-471-97208-8.
4. I. F. Alexander and R. Stevens, "Writing Better Requirements", Addison Wesley, 2002, ISBN 0-321-13163-0.

แบบฝึกหัด

คำสั่ง ให้นิสิตยกตัวอย่างประเภทของความไม่สอดคล้องกัน/ขัดแย้งกันของความต้องการ อย่างน้อย 1 ตัวอย่าง ในแต่ละประเภทความไม่สอดคล้องกัน/ขัดแย้งกันของความต้องการที่กำหนด

1. **คำศัพท์ที่ขัดแย้งกัน (Terminology clash)**

.....

.....

.....

.....

2. **การตั้งชื่อที่ขัดแย้งกัน (Designation clash)**

.....

.....

.....

.....

3. **โครงสร้างที่ขัดแย้งกัน (Structure clash)**

.....

.....

.....

.....

4. **ความขัดแย้งที่รุนแรง (Strong conflict)**

.....

.....

.....

.....

5. **ความแตกต่างหรือความขัดแย้งที่ไม่รุนแรง (Weak conflict or divergence)**

.....

.....

.....

.....

แผนบริหารการสอน
บทที่ 6 การจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์

เนื้อหา

1. ความต้องการระดับผู้ใช้ (User requirements)
2. ความต้องการระดับระบบ (System requirements)
3. คุณภาพและข้อบกพร่องที่ควรหลีกเลี่ยงในการจัดทำเอกสารความต้องการ
4. ตัวอย่างการเขียนข้อกำหนดความต้องการด้านซอฟต์แวร์

ผลลัพธ์การเรียนรู้รายบทเรียน

1. ผู้เรียนอธิบายเกี่ยวกับเอกสารความต้องการ
2. ผู้เรียนอธิบายองค์ประกอบต่างๆ ของเอกสารความต้องการ
3. ผู้เรียนอธิบายความรู้เกี่ยวกับแม่แบบการเขียนเอกสารความต้องการ
4. ผู้เรียนเขียนข้อกำหนดความต้องการแบบเป็นทางการ

วิธีการสอน

1. บรรยาย
2. ยกตัวอย่างระบบ โดยใช้กรณีตัวอย่างระบบที่เป็นที่รู้จัก
3. นิสิตร่วมอภิปรายแสดงความคิดเห็น

สื่อการสอน

1. แฟ้มดิจิทัลนำเสนอ
2. ระบบมัลติมีเดีย เช่น คอมพิวเตอร์ เครื่องเสียง เครื่องฉายภาพ
3. เอกสารประกอบคำสอน

การวัดและการประเมินผล

1. คะแนนการเข้าชั้นเรียน
2. คะแนนแบบฝึกหัด
3. คะแนนการอภิปรายของนิสิต

บทที่ 6 การจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์

ท้ายที่สุดแล้วความต้องการทั้งหมดที่เก็บรวบรวมจะถูกนำมาเขียนเป็นเอกสารข้อกำหนดความต้องการด้านซอฟต์แวร์ (Software Requirement Specificattions : SRS) และอาจถูกแก้ไขซ้ำจนกว่าจะมีความถูกต้องสมบูรณ์ และเป็นปัจจุบัน

เอกสารข้อกำหนดความต้องการด้านซอฟต์แวร์ (Software Requirement Specificattions : SRS)

เป็นเอกสารทางการในการกำหนดว่า ผู้พัฒนาระบบควรจะดำเนินการอย่างไร ซึ่งจะมีรายละเอียดทั้งความต้องการของผู้ใช้ (User requirement) และความต้องการระบบ (System requirement) จัดเป็นเอกสารที่มีความสำคัญมาก เสมือนสัญญาหรือข้อตกลงว่าผู้รับจ้างจะพัฒนาซอฟต์แวร์ใด มีความสามารถอะไร รวมถึงรายละเอียดสำคัญอื่นๆ

เอกสารข้อกำหนดความต้องการด้านซอฟต์แวร์มีแนวทางในการเขียน 2 ส่วนคือ

6.1 ความต้องการระดับผู้ใช้ (User requirements)

สามารถอธิบายได้ทั้งความต้องการแบบฟังก์ชันและไม่เป็นฟังก์ชัน โดยสามารถความต้องการในระดับนี้ผู้ที่ไม่มีความรู้ด้านซอฟต์แวร์ก็สามารถเข้าใจได้ อาจใช้รูปภาพ หรือตารางอธิบายเพิ่มเติมโดยมีหลักการทั่วๆ ไป ดังนี้

- ใช้ Template หรือแม่แบบของเอกสารที่เป็นมาตรฐาน เข้าใจได้ง่าย
- ใช้ระดับภาษาที่เข้าใจได้ง่าย
- หลีกเลี่ยงการใช้คำศัพท์ที่เกี่ยวข้องกับคอมพิวเตอร์ที่เข้าใจได้ยาก

6.2 ความต้องการระดับระบบ (System requirements)

ในส่วนนี้จะสามารถลงรายละเอียดของความต้องการในเชิงลึก หรือทางเทคนิคได้มากกว่า User requirements ซึ่งขึ้นอยู่กับแต่ละระบบและสัญญาจ้าง โดยมักจะระบุถึงประเด็นต่างๆ ดังนี้

- ขอบเขตการทำงานของระบบ
- บันทึกกระบวนการการทำงานทั้งหมดตั้งแต่ต้นจนจบ
- ขั้นตอนการบำรุงรักษา

6.3 คุณภาพและข้อบกพร่องที่ควรหลีกเลี่ยงในการจัดทำเอกสารความต้องการ

การทำอย่างตั้งใจและใช้ความรอบครอบอย่างประณีต เพื่อให้ได้มาของเอกสารความต้องการ (Requirements documents) ที่ดี ทำได้ค่อนข้างยาก เนื่องจากปัจจัยในเชิงคุณภาพมีความหลากหลายและยุ่งยากในการจัดการปัจจัยที่เกี่ยวกับคุณภาพ ในส่วนนี้จะเป็นการอธิบายให้เห็นถึงปัจจัยดังกล่าว รวมถึงอธิบายถึงข้อบกพร่องใดที่ควรหลีกเลี่ยง ดังแสดงในตารางที่ 6.1

ตารางที่ 6.1 คุณสมบัติหรือลักษณะที่ดีของความต้องการ

#	คุณภาพที่พึงประสงค์	ความหมาย
1.	Completeness	การระบุและรวบรวมความต้องการทั้งหมดที่เกี่ยวข้องในโครงการหรือซอฟต์แวร์อย่างครอบคลุม โดยไม่ตกหล่น หรือขาดหายไป
2.	Consistency	ควรเขียนความต้องการที่มีความสอดคล้องกัน มีความหมายไปในทิศทางเดียวกัน สอดคล้องกัน
3.	Adequacy	เอกสารความต้องการควรระบุความต้องการในรายละเอียดที่เพียงพอ เพื่อให้มีความเหมาะสม และเป็นประโยชน์ต่อโครงการหรือซอฟต์แวร์ที่กำลังพัฒนา
4.	Unambiguity	จะต้องเขียนหรืออธิบายให้ชัดเจน ไม่คลุมเครือ โดยเฉพาะอย่างยิ่งคำจำกัดความ หรือการนิยามเทอมที่เกี่ยวข้อง เพื่อให้ความต้องการถูกอธิบายอย่างชัดเจนและไม่มีความกำกวมหรือความสับสนในการอ่านหรือเข้าใจ
5.	Measurability	จะต้องถูกเขียนหรือสร้างขึ้นมา ในระดับที่จะทำให้สามารถประเมินหรือวัดผล เพื่อตรวจและทวนสอบ รวมถึงทดสอบได้
6.	Pertinence	ความต้องการต้องสอดคล้องกับวัตถุประสงค์ของระบบ ภายใต้บริบทของ Problem world และ System world
7.	Feasibility	จะต้องมีความเป็นไปได้ (ไม่เพ้อฝัน) ภายใต้ข้อจำกัดต่างๆ (Constraints) เช่น งบประมาณ (Budget) เวลา (Schedule) และเทคโนโลยี
9.	Good structuring	เอกสารความต้องการ (Requirements document) จะต้องมีโครงสร้างที่เชื่อมต่อกันในองค์ประกอบที่เกี่ยวข้องกัน
10.	Modifiability	เอกสารความต้องการจะต้องสามารถนำมาปรับแก้ไข เปลี่ยนแปลง และเพิ่มเติมได้ (Revise, adapt, and extend) ในระยะเวลาที่เหมาะสม และไม่ทำให้ System-to-be เสียหาย
11.	Traceability	เนื้อหาต่างๆ ในเอกสารความต้องการ ไม่ว่าจะเป็นการเขียนใหม่ หรือการปรับแก้ จะต้องสามารถตรวจสอบย้อนกลับได้

ข้อบกพร่องที่ควรหลีกเลี่ยง (Defects to avoid) หมายถึงคุณสมบัติหรือลักษณะที่ไม่พึงประสงค์ของความต้องการ ที่ควรจะไม่มีปรากฏในความต้องการ ข้อบกพร่องเหล่านี้อาจทำให้เกิดข้อผิดพลาดของซอฟต์แวร์ ความล่าช้าในโครงการ และเสียค่าใช้จ่ายเพิ่มเติม ดังแสดงในตารางที่ 6.2

ตารางที่ 6.2 คุณสมบัติหรือลักษณะที่ไม่พึงประสงค์ของความต้องการ

#	ข้อบกพร่องที่ควรหลีกเลี่ยง	ความหมาย
1.	Omission	RD items ไม่ได้ระบุในส่วนที่สำคัญๆ ของ Business process (ที่เกี่ยวข้องกับ วัตถุประสงค์ความต้องการ หรือสมมติฐาน)
2.	Contradiction	การนิยาม RD items อธิบายไม่ได้ไปในทิศทางเดียวกัน หรือมีความขัดแย้งกัน
3.	Inadequacy	การกำหนดรายละเอียดของ RD items ไม่ละเอียดเพียงพอที่จะเข้าใจได้
4.	Unmeasurability	การอธิบายหรือนิยามหรือกำหนด RD items ไม่สามารถประเมิน หรือวัดในเชิงเปรียบเทียบได้ ซึ่งส่งผลทำให้ไม่สามารถทดสอบซอฟต์แวร์ได้ (Testing)
5.	Noise	การอธิบาย RD item ซึ่งประกอบไปด้วยข้อมูลที่ไม่จำเป็น หรือไม่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ใหม่เลย
6.	Overspecification	การกำหนด RD items ที่สูงเกินความจำเป็น
7.	Unfeasibility	RD items ที่ไม่สามารถทำได้ภายใต้ข้อกำหนดต่างๆ เช่น วงเงินงบประมาณที่จำกัด ภายใต้กรอบเวลาที่จำกัด หรือภายใต้ข้อจำกัดทางเทคโนโลยี
8.	Unintelligibility	การอธิบาย RD items ในแนววิธีหรือด้วยข้อความที่ทำให้ผู้อ่านหรือที่เกี่ยวข้องไม่เข้าใจในข้อความเหล่านั้น หรือประกอบด้วยข้อความที่ยากเกินกว่าจะทำให้เข้าได้
9.	Poor structuring	RD items ถูกจัดวาง หรือจัดเรียงแบบไม่เหมาะสม ซึ่งอาจส่งผลให้เข้าใจยาก
10.	Forward reference and remorse	RD items ใช้ข้อความ หรือ Features ต่างๆ ที่ยังไม่ได้ถูกนิยามอย่างชัดเจน หรือถูกกำหนดในเวลาที่สายเกินไป
11.	Poor modifiability	การปรับแก้ RD items บางรายการส่งผลกระทบกับทั้งระบบ ซึ่งทำให้ทั้งระบบจำเป็นต้องถูกปรับแก้ไปด้วย
12.	Opacity	RD items ถูกอธิบายแบบไม่ชัดเจน เช่น สัดส่วน (Ratio) สายบังคับบัญชา (Authoring) หรือการขึ้นต่อกันของข้อมูลบางอย่าง (Dependency)

6.4 ตัวอย่างการเขียนข้อกำหนดความต้องการด้านซอฟต์แวร์

เอกสารข้อกำหนดความต้องการด้านซอฟต์แวร์

อ้างอิงจากมาตรฐาน IEEE 830-1998

IEEE Recommended Practice for Software Requirements Specifications

สามารถดาวน์โหลด Template เอกสารข้อกำหนดความต้องการด้านซอฟต์แวร์ได้จากลิงค์ที่กำหนดให้ด้านล่าง

https://drive.google.com/drive/folders/1O1H9_Xo6XrZmSsqcewJQW7PbVxAGF1p5?usp=sharing

***** ตัวอย่างต่อไปนี้เป็นเพียงแนวคิดเพื่อใช้ในการศึกษาเกี่ยวกับการเขียนข้อกำหนดความต้องการด้านซอฟต์แวร์ *****
หากนำไปใช้จริง จำเป็นจะต้องวิเคราะห์ความต้องการให้เหมาะสมกับบริบทต่างๆ และสามารถลด-เพิ่ม หัวข้อต่างๆ ได้
ทั้งนี้ ผู้เขียนใช้ตัวอย่างจากหนังสือ Fundamental of software engineering & digital transformation และขอขอบคุณไว้ ณ ที่นี้

ตารางที่ 6.3 องค์ประกอบของเอกสาร

บทที่	เนื้อหา
	หน้าปก ● ชื่อโครงการ/ชื่อระบบ ● ชื่อองค์กร ● เวอร์ชันของซอฟต์แวร์
	ประวัติการแก้ไข ● ชื่อผู้แก้ไข ● วันที่แก้ไข ● เหตุผล/รายละเอียดในการปรับแก้ ● เวอร์ชัน
	สารบัญต่างๆ ● สารบัญ ● สารบัญรูปภาพ ● สารบัญตาราง
1.	**บทนำ (Introduction)** 1.1 วัตถุประสงค์ (Purpose) 1.2 สัญลักษณ์ที่ใช้ในเอกสาร (Document conventions) 1.3 กลุ่มเป้าหมายและคำแนะนำการอ่าน (Intended audience and reading suggestions) 1.4 ขอบเขต (Product scope) 1.5 การอ้างอิง (References)

บทที่	เนื้อหา
2.	**คำอธิบายโดยรวม (Overall description)** 4.1 มุมมองของระบบ (Product perspective) 4.2 ฟังก์ชันของระบบ (Product functions) 4.3 ประเภทและคุณลักษณะของคลาส (User Classes and charecteristics) 4.4 สภาพแวดล้อมการทำงาน (Opereting environment) 4.5 ข้อจำกัดในการออกแบบและการนำไปใช้งาน (Design and implementation constraints) 4.6 เอกสารสำหรับผู้ใช้ (User documentation) 4.7 สมมติฐานและการพึ่งพา (Assumptions and dependencies)
3.	**ข้อกำหนดการเชื่อมโยงภายนอก (External interface requirement)** 3.1 ส่วนต่อประสานผู้ใช้ (User interfaces) 3.2 ส่วนต่อประสานฮาร์ดแวร์ (Hardware interfaces) 3.3 ส่วนต่อประสานซอฟต์แวร์ (Software interfaces) 3.4 ส่วนต่อประสานการสื่อสาร (Communication interfaces)
4.	**คุณสมบัติของระบบ (System feature)**
5.	**ข้อกำหนดที่ไม่ใช่ฟังก์ชัน (Non-function)** 5.1 ข้อกำหนดด้านประสิทธิภาพ (Performance requirements) 5.2 ข้อกำหนดด้านความปลอดภัย (Safety requirements) 5.3 ข้อกำหนดด้านความมั่นคง (Security requirements) 5.4 คุณสมบัติด้านคุณภาพของซอฟต์แวร์ (Software quality attribute) 5.5 กฎทางธุรกิจ (Business rules)
	ข้อกำหนดอื่นๆ (Other requirements) ภาคผนวก ก อภิธานศัพท์ (Glossary) ภาคผนวก ข แบบจำลองการวิเคราะห์ (Analysis models) ภาคผนวก ค รายการความต้องการที่กำหนดภายหลัง (To be determined list)

บทนำ (Introduction)

6.4.1 วัตถุประสงค์ (Purpose)

คำอธิบาย	ใช้สำหรับอธิบายวัตถุประสงค์ของระบบ
ตัวอย่าง	*1. เพื่อพัฒนาการให้บริการอิเล็กทรอนิกส์ ในการออกใบอนุญาตกิจการที่เป็นอันตรายต่อสุขภาพ* *2. เพื่อเชื่อมต่อการให้บริการผ่านทางเว็บเซอร์วิส*

6.4.2 สัญลักษณ์ที่ใช้ในเอกสาร (Document conventions)

คำอธิบาย	อธิบายสัญลักษณ์หรือรูปภาพที่มีความสำคัญเป็นพิเศษ
ตัวอย่าง	*ข้อกำหนดความต้องการด้านซอฟต์แวร์ฉบับนี้อ้างอิงกับมาตรฐานยูเอ็มเอล (UML) และผังงาน (Flow chart)*

6.4.3 กลุ่มเป้าหมายและคำแนะนำการอ่าน (Intended audience and reading suggestions)

คำอธิบาย	อธิบายประเภทของกลุ่มเป้าหมาย และแนะนำลำดับในการอ่านเอกสาร
ตัวอย่าง	*เอกสารนี้มีไว้สำหรับวิศวกรซอฟต์แวร์ นักพัฒนาซอฟต์แวร์ โปรแกรมเมอร์ และผู้จัดการโครงการ ก่อนอ่านเอกสารนี้แนะนำให้อ่านเอกสารวิสัยทัศน์ กฎระเบียบที่เกี่ยวข้องเพื่อดูภาพรวมของการทำงาน*

6.4.4 ขอบเขต (Product scope)

คำอธิบาย	อธิบายสั้นๆ เกี่ยวกับขอบเขตการทำงานของซอฟต์แวร์ เชื่อมโยงซอฟต์แวร์กับเป้าหมายขององค์กรหรือกลยุทธ์ทางธุรกิจ วิสัยทัศน์ พันธกิจ รวมถึงตัวชี้วัดขององค์กร
ตัวอย่าง	*เนื่องจากปัญหาของการให้บริการภาครัฐส่วนใหญ่เกี่ยวข้องกับการจัดเก็บข้อมูล ส่งผลให้เกิดความผิดพลาดในการปฏิบัติงาน ทำให้เจ้าหน้าที่นั้นปฏิบัติงานไม่ต่อเนื่อง ประชาชนผู้รับบริการไม่ได้รับความสะดวกสบายทและไม่สามารถติดตามผลของการทำธุรกรรมได้อีก ทั้งนี้ภาครัฐประกาศนโยบายรัฐบาลดิจิทัล การให้บริการแบบอิเล็กทรอนิกส์ ซึ่งเป็นการเก็บรวบรวมข้อมูลให้อยู่ในรูปแบบของฐานข้อมูลสามารถรวบรวมข้อมูลต่างๆ ที่เกี่ยวข้องกันเข้าไว้ด้วยกันอย่างเป็นระบบ มีความสัมพันธ์ระหว่างข้อมูลต่างๆ ที่ชัดเจนและเปิดโอกาสให้ผู้ใช้สามารถใช้งานและดูแลป้องกันข้อมูลเหล่านั้นได้อย่างมีประสิทธิภาพ* *โดยหน่วยงานปกครองส่วนท้องถิ่นแห่งหนึ่งประสบปัญหาของการจัดเก็บข้อมูลขององค์กรเช่น พบข้อผิดพลาดในการเก็บข้อมูลของบุคคลทั่วไปหรือนิติบุคคลที่ได้ขอออกใบอนุญาตกิจการที่*

เป็นอันตรายต่อสุขภาพ ข้อผิดพลาดในการเก็บข้อมูลนั้นมีสาเหตุมาจากการที่เก็บข้อมูลของผู้ที่มาขอใบอนุญาตกิจการที่เป็นอันตรายสูญหาย และไม่ทราบว่าการขอใบอนุญาตกิจการนั้นอยู่ในขั้นตอนใดแล้ว ส่งผลให้เกิดข้อผิดพลาดในการปฏิบัติงาน ทำให้เจ้าหน้าที่นั้นปฏิบัติงานไม่ต่อเนื่อง ดังนั้นหน่วยงานภาครัฐจึงได้พัฒนาโครงการระบบออกใบอนุญาตกิจการที่เป็นอันตรายต่อสุขภาพ เพื่อให้ผู้รับใบอนุญาตสามารถยื่นเสนอขอใบอนุญาตผ่านอินเทอร์เน็ต และโทรศัพท์เคลื่อนที่รวมทั้งระบบปฏิบัติการ iOS และ Android สามารถชำระค่าธรรมเนียมได้หลายช่องทางทั้งเงินสดหรือ Mobile Banking ยังมีการพัฒนาระบบเชื่อมต่อกันภายในหน่วยงาน และหน่วยงานภายนอกด้วยเว็บเซอร์วิส

6.4.5 การอ้างอิง (References)

คำอธิบาย	แสดงรายการเอกสารหรือที่อยู่บนเว็บอื่นๆ ที่ SRS นี้อ้างอิงถึงสิ่งเหล่านี้ อาจรวมถึงคู่มือยุทธศาสตร์ขององค์กรที่แสดง วิสัยทัศน์ พันธกิจ และตัวชี้วัด เป็นต้น
ตัวอย่าง	• *ข้อบัญญัติองค์การบริหารส่วนตำบลเรื่องการควบคุมกิจการที่เป็นอันตรายต่อสุขภาพ* • *พระราชบัญญัติการบริหารงานและการให้บริการภาครัฐผ่านระบบดิจิทัล พ.ศ 2562 (https://www.dga.or.th/policy-standard/policy-regulation/dga-dg-256/dga-046/)* • *แผนยุทธศาสตร์ขององค์กร* • *พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ 2562 (https://www.ratchakitcha.soc.go.th/DATA/PDF/2562/A/069/T_0052.PDF)* • *กรอบแนวทางการเชื่อมโยงรัฐบาลอิเล็กทรอนิกส์แห่งชาติภาครัฐ (http://sql.ldd.go.th/intraaccount/Th_e-GIF/TH-eGif_EGA.pdf)* • *คู่มือการจัดทำมาตรฐานเพื่อการเชื่อมโยงข้อมูลระหว่างหน่วยงานภาครัฐ (https://www.dla.go.th/upload/ebook/column/2554/5/622_3335.pdf)*

คำอธิบายโดยรวม (Overall description)

6.4.6 มุมมองของระบบ (Product perspective)

คำอธิบาย	อธิบายบริบทหรือแผนภาพที่แสดงส่วนประกอบหลักของระบบโดยรวมการเชื่อมต่อระหว่างระบบย่อยส่วนต่อประสานและอินเตอร์เฟสภายนอกจะเป็นประโยชน์ต่อการพัฒนาซอฟต์แวร์
ตัวอย่าง	

6.4.7 ฟังก์ชันของระบบ (Product functions)

คำอธิบาย	สรุปฟังก์ชันหลักที่ซอฟต์แวร์ต้องทำ เช่น แผนภาพกระแสข้อมูลระดับบนสุดหรือแผนภาพคลาสออบเจ็กต์
ตัวอย่าง	*ระบบนี้เป็นการเก็บรวบรวมข้อมูลของผู้ขอออกใบอนุญาตกิจการที่เป็นอันตรายต่อสุขภาพ โดยจะเก็บรวมรวมฐานข้อมูลจากการกรอกข้อมูลในเว็บแอปพลิเคชัน หลังจากนั้นข้อมูลที่ถูกกรอกไปจะถูกส่งข้อมูลไปที่ฐานข้อมูลเพื่อเก็บรวบรวมข้อมูลและประมวลผลต่อไป ระบบนี้แบ่งผู้ใช้งานระบบออกเป็น 4 ส่วนคือ* ***1. ผู้ขอรับใบอนุญาต (User)*** *แบ่งเป็นบุคคลธรรมดา และนิติบุคคล สามารถลงทะเบียนและกรอกข้อมูลที่ใช้ในการขอออกใบอนุญาตกิจการที่เป็นอันตรายต่อสุขภาพ Upload เอกสารที่ต้องใช้ประกอบการขออนุญาต สามารถตรวจสอบสถานการณ์ออกใบอนุญาตได้โดย ในกรณีที่ใช้แอปพลิเคชันบนโทรศัพท์เคลื่อนที่สามารถชำระค่าธรรมเนียมผ่านทางโมบายแบงก์กิ้งได้*

*2. **เจ้าหน้าที่ฝ่ายออกใบอนุญาต (Licensing Officer)** สามารถดูข้อมูลของผู้ขอรับใบอนุญาตสืบค้นข้อมูลตรวจสอบความถูกต้องของเอกสารตามกฎในการออกใบอนุญาตกิจการที่เป็นอันตรายต่อสุขภาพและออกใบอนุญาต*

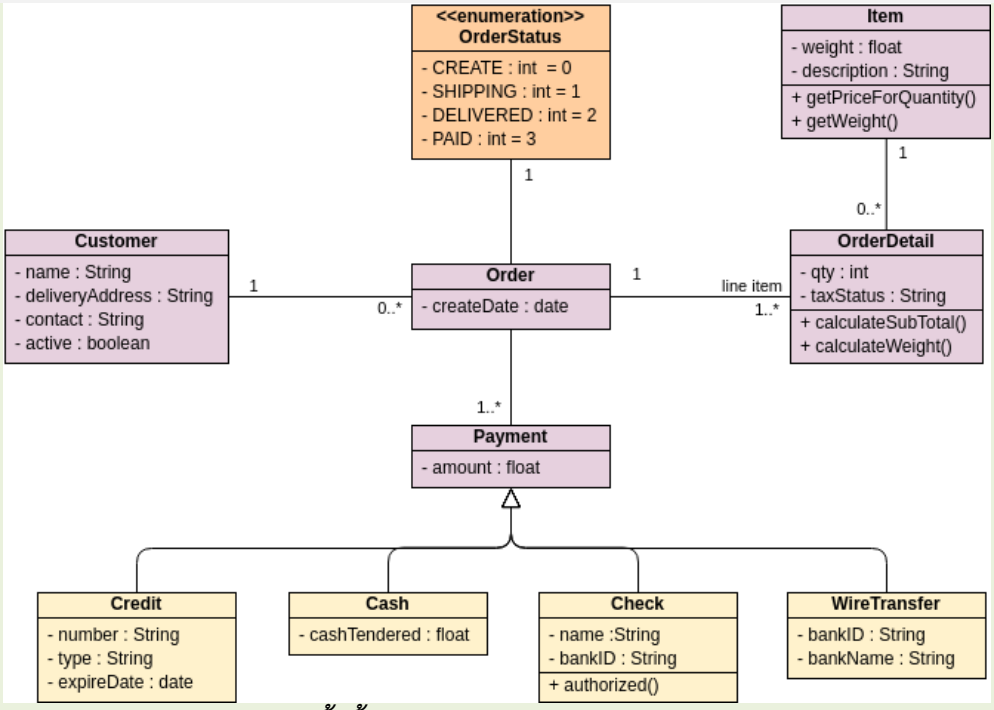
*3. **เจ้าหน้าที่ฝ่ายการเงิน (Accountant)** สามารถรับเงินค่าธรรมเนียมและออกใบเสร็จรับเงินให้ผู้ขอรับใบอนุญาต*

*4. **ผู้ดูแลระบบ (Administrator)** สามารถเข้าถึงข้อมูลทั้งหมดของระบบ สามารถดูข้อมูลผู้ขอรับใบอนุญาต กำหนด Username, Password และเพิ่มบริการต่างๆ ได้*

6.4.8 ประเภทและคุณลักษณะของคลาส (User Classes and charecteristics)

คำอธิบาย ระบุคลาสผู้ใช้ต่างๆ ที่คาดว่าจะใช้ซอฟต์แวร์นี้

ตัวอย่าง



ทั้งนี้ให้เขียนอธิบายแผนภาพด้วย

6.4.9 สภาพแวดล้อมการทำงาน (Opereting environment)

คำอธิบาย	อธิบายสภาพแวดล้อมที่ซอฟต์แวร์จะทำงาน รวมถึงเพลตฟอร์มฮาร์ดแวร์ระบบปฏิบัติการและเวอร์ชันและส่วนประกอบซอฟต์แวร์หรือแอปพลิเคชันอื่นๆ ที่ต้องทำงานร่วมกัน
ตัวอย่าง	• *ในส่วนของแอปพลิเคชันเซิร์ฟเวอร์ ระบบที่พัฒนาต้องสามารถทำงานได้บนระบบปฏิบัติการ Windows Server (Version xxx) และ IIS (Version xxx)* • *ในส่วนของเครื่อง Database server ระบบที่พัฒนาต้องสามารถทำงานได้กับฐานข้อมูล MysQL บนระบบปฏิบัติการ Windows Server (Version xxx)* • *ในส่วนของผู้ใช้เว็บแอปพลิเคชัน ระบบที่พัฒนาต้องสามารถรองรับการใช้งานของผู้ใช้บน Browser ดังต่อไปนี้ Google Chrome (Version xxx), Mozilla Firefox (Version xxx), Internet Explorer (Version xxx) ทั้งในส่วนของเครื่องคอมพิวเตอร์แบบตั้งโต๊ะ หรือโน้ตบุ๊ค หรือ โทรศัพท์เคลื่อนที่ (Smartphone)* • *ในส่วนผู้ใช้งาน Mobile application ระบบที่พัฒนาต้องสามารถรองรับการใช้งานของผู้ใช้บนระบบปฏิบัติการ iOS (Version xxx) หรือ Android (Version xxx) หรือ Window Phone (Version xxx)*

6.4.10 ข้อจำกัดในการออกแบบและการนำไปใช้งาน (Design and implementation constraints)

คำอธิบาย	อธิบายรายการหรือปัญหาใดๆ ที่จะจำกัดตัวเลือกที่มีให้สำหรับนักพัฒนา ซึ่งอาจรวมถึงนโยบายขององค์กรหรือกฎข้อบังคับ ข้อจำกัดของฮาร์ดแวร์ (ข้อกำหนดด้านเวลาความต้องการหน่วยความจำ) การเชื่อมต่อกับแอปพลิเคชันอื่นๆ เทคโนโลยีเฉพาะเครื่องมือและฐานข้อมูลที่จะใช้ การดำเนินการคู่ขนานข้อกำหนดด้านภาษา โปรโตคอลการสื่อสาร ข้อพิจารณาด้านความปลอดภัย อนุสัญญาการออกแบบหรือมาตรฐานการเขียนโปรแกรม)
ตัวอย่าง	• *ระบบที่พัฒนาต้องใช้ภาษา PHP (Version xxx)* • *ระบบใช้ฐานข้อมูล MySQL (Version xxx)* • *ระบบต้องทำงานบนเครื่องเซิร์ฟเวอร์ที่ใช้ CPU Intel Core i8 หน่วยความจำ 8 GB ฮาร์ดดิสก์ขนาด 1 TB* • *ระบบต้องรองรับโปรโตคอล HTTP และ HTTPS* • *ระบบต้องรองรับการเข้ารหัสข้อมูลสำหรับ Password ของผู้ใช้ด้วยอัลกอริทึม MD5*

6.4.11 เอกสารสำหรับผู้ใช้ (User documentation)

คำอธิบาย	แสดงรายการส่วนประกอบเอกสารผู้ใช้ (เช่น คู่มือผู้ใช้/วิธีใช้/บทช่วยสอน)
ตัวอย่าง	• *ต้องมีคู่มือการใช้งานสำหรับผู้ใช้ ประกอบด้วย ผู้ขอใบอนุญาต เจ้าหน้าที่ออกใบอนุญาต* • *เจ้าหน้าที่การเงิน และผู้ดูแลระบบ* • *ต้องมีคู่มือการในติดตั้งระบบ การสำรองและการกู้คืนข้อมูล*

6.4.12 สมมติฐานและการพึ่งพา (Assumptions and dependencies)

คำอธิบาย	ระบุปัจจัยที่อาจส่งผลต่อข้อกำหนดที่ระบุไว้ รวมถึงปัญหาเกี่ยวกับการพัฒนาหรือสภาพแวดล้อมการดำเนินงานหรือข้อจำกัด
ตัวอย่าง	• *การปรับเปลี่ยนกฎหมาย กฎระเบียบที่เกี่ยวข้อง เช่น อัตราค่าธรรมเนียม เป็นต้น* • *ฮาร์ดแวร์ทั้งหมดที่จำเป็นและเพียงพอ เพื่อรองรับการอัพโหลดของผู้ขอรับใบอนุญาต* • *ในกรณีที่มีผู้ขอรับใบอนุญาตเข้าใช้งานพร้อมๆ กันเป็นจำนวนมากจนทำให้เครื่องเซิร์ฟเวอร์มีปัญหา* • *เครื่องเซิร์ฟเวอร์โดนโจมตีจากไวรัสหรือแฮ็กเกอร์* • *ระบบเครือข่ายอินเทอร์เน็ตใช้งานไม่ได้* • *ผู้ให้บริการเว็บเซอร์วิสปรับเปลี่ยนรูปแบบการให้บริการ*

ข้อกำหนดการเชื่อมโยงภายนอก (External interface requirement)

6.4.13 ส่วนต่อประสานผู้ใช้ (User interfaces)

คำอธิบาย	อธิบายลักษณะส่วนต่อประสานระหว่างซอฟต์แวร์และผู้ใช้ ซึ่งอาจรวมถึงภาพหน้าจอตัวอย่างมาตรฐาน GUI หรือคู่มือสไตล์ซอฟต์แวร์ที่ต้องปฏิบัติตามข้อจำกัด การจัดวางหน้าจอปุ่มและฟังก์ชันมาตรฐาน (เช่น วิธีใช้) ที่จะปรากฏบนทุกหน้าจอแป้นพิมพ์ลัดมาตรฐาน การแสดงข้อความแสดงข้อผิดพลาด เป็นต้น
ตัวอย่าง	• *การออกแบบส่วนต่อประสานผู้ใช้ให้ใช้มาตรฐาน HTML 4.0 หรือ HTML 5.0 และ Cascading Style Sheets (css) เพื่อเว็บเพจสามารถนำไปแสดงบน Browser ต่างชนิดกันและให้ใช้มาตรฐาน W3C* • *การออกแบบเว็บไซต์ด้วย เทคนิค Responsive Web Design ที่จะทำให้เว็บไซต์สามารถแสดงผลได้อย่างเหมาะสมบนอุปกรณ์ที่แตกต่างกัน โดยใช้โค้ดร่วมกันและเข้าถึงเว็บเดียวกัน*

6.4.14 ส่วนต่อประสานฮาร์แวร์ (Hardware interfaces)

คำอธิบาย	อธิบายลักษณะทางตรรกะและทางกายภาพของแต่ละส่วนต่อประสานระหว่างซอฟต์แวร์และส่วนประกอบฮาร์ดแวร์ของระบบ ซึ่งอาจรวมถึงประเภทอุปกรณ์ที่รองรับลักษณะของข้อมูล และการควบคุมการโต้ตอบระหว่างซอฟต์แวร์และฮาร์ดแวร์และโปรโตคอลการสื่อสารที่ใช้
ตัวอย่าง	*ไม่มีส่วนต่อประสานฮาร์ดแวร์โดยตรงสำหรับเว็บแอปพลิเคชันที่ทำงานบนแอปพลิเคชันเซิร์ฟเวอร์ที่ติดตั้งภายในองค์กรโดยผ่านทาง Firewall*

6.4.15 ส่วนต่อประสานซอฟต์แวร์ (Software interfaces)

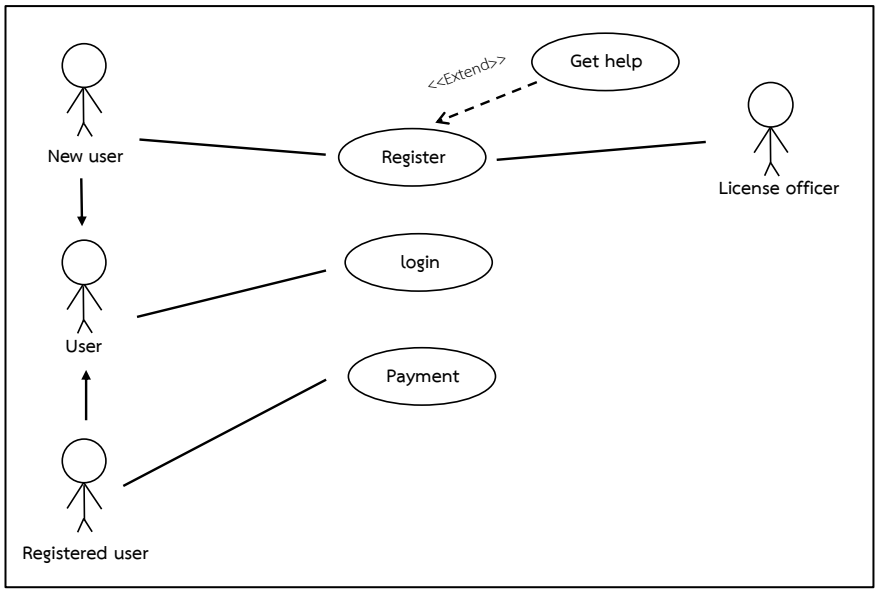
คำอธิบาย	อธิบายส่วนต่อประสานระหว่างซอฟต์แวร์นี้กับส่วนประกอบซอฟต์แวร์เฉพาะอื่นๆ (ชื่อและเวอร์ชัน) รวมถึงฐานข้อมูล ระบบปฏิบัติการ เครื่องมือไลบรารี อธิบายบริการที่จำเป็นและลักษณะของการสื่อสารอ้างถึงเอกสารที่อธิบายโปรโตคอล อินเตอร์เฟสการเขียนโปรแกรมแอปพลิเคชันโดยละเอียด ระบุข้อมูลที่จะแลกเปลี่ยนระหว่างส่วนประกอบซอฟต์แวร์
ตัวอย่าง	• *การสื่อสารระหว่างฐานข้อมูลและแพลตฟอร์มประกอบด้วย การดำเนินการเกี่ยวกับการอ่านและการแก้ไขข้อมูล เฟรมเวิร์คจะถูกใช้สำหรับผู้ใช้โดยใช้ ODBC (Open Database Connectivity)* • *ข้อมูลจะถูกรวบรวมโดยใช้เว็บฟอร์ม PHP และถูกใช้เพื่อเข้าถึงฐานข้อมูล เพื่อตรวจสอบความถูกต้องของการนำเข้าข้อมูลเพื่อการประมวลผลและแสดงผล*

6.4.16 ส่วนต่อประสานการสื่อสาร (Communication interfaces)

คำอธิบาย	อธิบายข้อกำหนดที่เกี่ยวข้องกับฟังก์ชันการสื่อสารที่จำเป็นสำหรับซอฟต์แวร์นี้ รวมถึงอีเมล เว็บบราวเซอร์ โปรโตคอลการสื่อสารของเซิร์ฟเวอร์และเครือข่าย แบบฟอร์มอิเล็กทรอนิกส์และอื่นๆ กำหนดการจัดรูปแบบข้อความที่เกี่ยวข้อง ระบุมาตรฐานการสื่อสารที่จะใช้ เช่น FTP หรือ HTTP ระบุปัญหาด้านความปลอดภัยในการสื่อสาร หรือการเข้ารหัสอัตราการถ่ายโอนข้อมูลและกลไกการซิงโครไนซ์
ตัวอย่าง	• *ระบบจะใช้โปรโตคอล TCPIP และโปรโตคอล HTTP สำหรับเว็บไซต์ ข้อมูลฟอร์มผู้ใช้จะถูกถ่ายโอนโดยใช้วิธี HTTP-POST และข้อมูลการค้นหาจะถูกถ่ายโอนโดยใช้วิธี HTTP-GET ข้อมูลรหัสผ่านจะถูกเข้ารหัสโดยใช้โปรโตคอล HTTPS ในกรณีที่มีการเข้าใช้งานระบบและการชำระค่าธรรมเนียม* • *การอัพโหลดเอกสารใช้โปรโตคอล FTP* • *การเชื่อมต่อกับหน่วยงานภายนอกใช้โปรโตคอล SOAP และรับ-ส่งข้อมูลด้วย XML* • *การเชื่อมต่อสำหรับโมบายล์แอปพลิเคชันผ่านทางเครือข่ายโทรศัพท์เคลื่อนที่ 4G หรือ 5G*

คุณสมบัติของระบบ (System feature)

เทมเพลตนี้แสดงให้เห็นถึงการจัดระเบียบข้อกำหนดการใช้งานสำหรับซอฟต์แวร์ตามคุณลักษณะของระบบบริการหลักที่ซอฟต์แวร์มีให้



ภาพที่ 6.1 ตัวอย่าง Usecase ระบบการออกใบอนุญาต

6.4.17 คุณสมบัติของ Use Case Register (ลงทะเบียน)

1. คำอธิบายและลำดับความสำคัญ (Description and Priority)

คำอธิบาย	ระบุคำอธิบายสั้นๆ ของคุณสมบัติและระบุว่ามีลำดับความสำคัญสูง ปานกลาง หรือต่ำ
ตัวอย่าง	*ผู้ใช้ที่ไม่ได้ลงทะเบียนจะต้องลงทะเบียนเพื่อใช้ฟังก์ชันอื่นๆ คุณลักษณะนี้มีความสำคัญสูง*

2. สิ่งกระตุ้น/ลำดับการตอบสนอง (Stimulus/Response Sequences)

คำอธิบาย	แสดงรายการลำดับของการกระทำของผู้ใช้และการตอบสนองของระบบที่กระตุ้นพฤติกรรม ที่กำหนดไว้สำหรับคุณลักษณะนี้
ตัวอย่าง	• *ปุ่ม "ลงทะเบียน" /คลิกลิงก์ : แบบฟอร์มข้อมูลลงทะเบียนจะปรากฏขึ้น* • *คลิกปุ่ม "ส่ง" : ข้อมูลที่ลงทะเบียนจะได้รับการตรวจสอบ ถ้ามีปัญหาข้อความแสดงความช่วยเหลือ (Help) จะแสดงเป็นป้ายกำกับหรือกล่องโต้ตอบ การลงทะเบียนที่สำเร็จจะส่งต่อผู้ใช้ไปยังหน้าหลักของผู้ใช้*

3. ข้อกำหนดการทำงาน (Functional Requirements)

คำอธิบาย ระบุรายการข้อกำหนดการใช้งานโดยละเอียดที่เกี่ยวข้องกับความสามารถของซอฟต์แวร์ที่ต้องมีเพื่อให้ผู้ใช้สามารถใช้บริการที่มีให้โดยคุณลักษณะนี้ หรือเพื่อดำเนินการตามกรณีการใช้งานรวมถึงวิธีที่ซอฟต์แวร์ควรตอบสนองต่อเงื่อนไขข้อผิดพลาดที่คาดการณ์ไว้ หรือการนำเข้าข้อมูลที่ไม่ถูกต้อง ข้อกำหนดควรกระชับครบถ้วน ไม่คลุมเครือ ตรวจสอบได้

ตัวอย่าง

REQ-1 :	
ชื่อยูเคส	*Register (ลงทะเบียน)*
ผู้ใช้งาน	*1. New user* *2. License officer*
วัตถุประสงค์	*เพื่อให้ผู้ขอใบอนุญาตสามารถลงทะเบียนได้ และเมื่อลงทะเบียนสำเร็จแล้วสามารถเข้าสู่ระบบ และดูรายละเอียดประเภทใบอนุญาตก่อนยื่นเรื่องได้*
เงื่อนไขก่อนหน้า	*ผู้ขอใบอนุญาตยังไม่มีการสมัครใช้งานระบบ*
เงื่อนไขภายหลัง	*มีบัญชีขอรับใบอนุญาต*
ขั้นตอนการทำงาน	*1. กรอกข้อมูลส่วนตัวผู้ขอใบอนุญาต โดยใช้เลขที่บัตรประชาชน* *2. กรอก Username และ Password* *3. ยืนยันสิทธิ์การเข้าใช้งานระบบโดย License officer*

6.4.18 คุณสมบัติของ User Case Get Help (ขอความช่วยเหลือ)

1. คำอธิบายและลำดับความสำคัญ (Description and Priority)

คำอธิบาย	ระบุคำอธิบายสั้นๆ ของคุณสมบัติและระบุว่ามีลำดับความสำคัญสูง ปานกลาง หรือต่ำ
ตัวอย่าง	*ผู้ที่ลงทะเบียนสามารถขอความช่วยเหลือจากระบบได้กรณีที่มีปัญหา คุณลักษณะนี้มีลำดับความสำคัญต่ำ*

2. สิ่งกระตุ้น/ลำดับการตอบสนอง (Stimulus/Response Sequences)

คำอธิบาย	แสดงรายการลำดับของการกระทำของผู้ใช้และการตอบสนองของระบบที่กระตุ้นพฤติกรรม ที่กำหนดไว้สำหรับคุณลักษณะนี้
ตัวอย่าง	• *คลิกปุ่ม/ลิงค์ "ดาวน์โหลด" : ระบบจะดาวน์โหลดเอกสารที่เลือกไปยังคอมพิวเตอร์ของผู้ขอรับใบอนุญาต* • *คลิกปุ่ม/ลิงค์ "คำถามที่พบบ่อย (FAQ)" : คำถามและคำตอบที่เป็นประโยชน์จะปรากฏขึ้น*

3. ข้อกำหนดการทำงาน (Functional Requirements)

คำอธิบาย	ระบุรายการข้อกำหนดการใช้งานโดยละเอียดที่เกี่ยวข้องกับความสามารถของซอฟต์แวร์ที่ต้องมีเพื่อให้ผู้ใช้สามารถใช้บริการที่มีให้โดยคุณลักษณะนี้ หรือเพื่อดำเนินการตามกรณีการใช้งานรวมถึงวิธีที่ซอฟต์แวร์ควรตอบสนองต่อเงื่อนไขข้อผิดพลาดที่คาดการณ์ไว้ หรือการนำเข้าข้อมูลที่ไม่ถูกต้อง ข้อกำหนดควรกระชับครบถ้วน ไม่คลุมเครือ ตรวจสอบได้

ตัวอย่าง

REQ-2 :	
ชื่อยูเคส	*Get Help (ขอความช่วยเหลือ)*
ผู้ใช้งาน	*New user*
วัตถุประสงค์	*เพื่อให้ผู้ขอใบอนุญาตสามารถดาวน์โหลดคู่มือการใช้งาน และกฎระเบียบต่างๆ รวมถึงดูคำถามที่พบบ่อย (FAQ)*
เงื่อนไขก่อนหน้า	*ผู้ขอรับใบอนุญาตยังไม่มีการสมัครใช้งานระบบและต้องการคู่มือการใช้งาน*
เงื่อนไขภายหลัง	*ผู้ขอรับใบอนุญาตสามารถลงทะเบียนได้อย่างถูกต้อง*
ขั้นตอนการทำงาน	*1. ผู้ใช้ที่ลงทะเบียนจะเลือกหนึ่งในประเภทเอกสารที่ต้องการแล้วคลิกปุ่ม “ดาวน์โหลด” ระบบจะดาวน์โหลดเอกสารไปยังโฟลเดอร์ที่ระบุโดยผู้ใช้* *2. ผู้ใช้ที่จะลงทะเบียนจะคลิกปุ่ม “FAQ” ระบบจะแสดงคำถามคำตอบที่พบบ่อย*

6.4.19 คุณลักษณะของระบบ 3 (และอื่นๆ) ทำเช่นเดียวกันจนครบทุก Use Case

ข้อกำหนดที่ไม่ใช่ฟังก์ชัน (Non-function)

6.4.20 ข้อกำหนดด้านประสิทธิภาพ (Performance requirements)

คำอธิบาย	หากมีข้อกำหนดด้านประสิทธิภาพสำหรับซอฟต์แวร์ภายใต้สถานการณ์ต่างๆ ให้ระบุไว้ที่นี่ และอธิบายเหตุผลเพื่อช่วยให้นักพัฒนาเข้าใจเจตนาและตัดสินใจเลือกการออกแบบที่เหมาะสม
ตัวอย่าง	• *ระบบนี้ต้องรองรับผู้ใช้จำนวนมากพร้อมๆ กัน ไม่ต่ำกว่า 100 Concurent USers ดังนั้น จึงต้องมีการตอบสนองผู้ใช้ในระยะเวลาที่รวดเร็ว* • *ระบบต้องสามารถทำงานได้ตลอดเวลา (24 ชั่วโมง 7 วัน)* • *ระบบต้องสามารถทำงานเข้ากันได้กับฮาร์ดแวร์ที่มีอยู่*

6.4.21 ข้อกำหนดด้านความปลอดภัย (Safety requirements)

คำอธิบาย	ระบุข้อกำหนดที่เกี่ยวข้องกับการความเสียหาย หรืออันตรายที่อาจเกิดขึ้นจากการใช้ซอฟต์แวร์ กำหนดมาตรการป้องกันหรือการดำเนินการใดๆ ที่ต้องดำเนินการ ตลอดจนการดำเนินการที่ต้องป้องกัน อ้างถึงนโยบายหรือข้อบังคับภายนอกใดๆ ที่ระบุปัญหาด้านความปลอดภัยที่ส่งผลต่อการออกแบบ หรือการใช้งานของซอฟต์แวร์และแนวทางการแก้ไข
ตัวอย่าง	• *ความล้มเหลวของฟังก์ชันซอฟต์แวร์ที่สำคัญด้านความปลอดภัย จะต้องได้รับการตรวจสอบ คัดแยก และกู้คืน เพื่อป้องกันไม่ให้เกิดเหตุการณ์ร้ายแรง* • *ซอฟต์แวร์จะต้องจัดการข้อผิดพลาดเพื่อสนับสนุนฟังก์ชันที่สำคัญด้านความปลอดภัย* • *ซอฟต์แวร์จะต้องจัดให้มีกลไกการป้องกันความผิดพลาด เพื่อป้องกันการเผยแพร่ข้อผิดพลาด* • *ซอฟต์แวร์ต้องสามารถยุติการทำงานได้หากทำเพื่อไปเพื่อให้ระบบเกิดความปลอดภัย*

6.4.22 ข้อกำหนดด้านความมั่นคง (Security requirements)

คำอธิบาย	ระบุข้อกำหนดใดๆ เกี่ยวกับปัญหาด้านความมั่นคงหรือความเป็นส่วนตัว การป้องกันข้อมูลที่ใช้หรือสร้างโดยซอฟต์แวร์ การพิสูจน์ตัวตนผู้ใช้ อ้างถึงนโยบายหรือข้อบังคับภายนอกใดๆ ที่มีปัญหาด้านความมั่นคงที่ส่งผลกระทบต่อซอฟต์แวร์ และการรับรองความมั่นคงหรือความเป็นส่วนตัวที่ต้องเป็นไปตามข้อกำหนด
ตัวอย่าง	***1. การรักษาความลับ (Confidentiality)*** ● *ผู้ใช้จะลงทะเบียนเข้าสู่ระบบโดยใช้หมายเลขบัตรประชาชน จะได้รับการตรวจสอบความถูกต้องก่อนที่จะจัดเก็บลงฐานข้อมูล* ● *เฉพาะเจ้าหน้าที่รับผิดชอบกับระบบนี้เท่านั้นที่จะมีสิทธิเข้าถึงข้อมูลผู้ขอรับใบอนุญาตได้* ***2. ความรับผิดชอบ (Accountability)*** ● *กิจกรรมการแก้ไขบัญชีทั้งหมดจะถูกบันทึก* ● *ระบบต้องมีการคุ้มครองข้อมูลส่วนบุคคลตาม พ.ร.บ คุ้มครองข้อมูลส่วนบุคคล พ.ศ. 2562*

6.4.23 คุณสมบัติด้านคุณภาพของซอฟต์แวร์ (Software quality attribute)

คำอธิบาย	ระบุลักษณะคุณภาพเพิ่มเติมสำหรับซอฟต์แวร์ที่จะมีความสำคัญต่อผู้ใช้หรือผู้พัฒนา สิ่งที่ต้องพิจาณา ได้แก่ ความสามารถในการปรับตัว ความพร้อมใช้งาน ความถูกต้อง ความยืดหยุ่น ความสามารถในการทำงานร่วมกัน การบำรุงรักษา ความน่าเชื่อถือ การนำกลับมาใช้ใหม่ มีความทนทาน ความสามารถในการทดสอบและการใช้งาน เป็นต้น
ตัวอย่าง	***1. ความพร้อมใช้งาน (Availability)*** ● *ระบบจะพร้อมใช้งาน 7 วัน 24 ชั่วโมง ซึ่งสามารถตรวจสอบได้โดยดูจากรายงานการทำงานของระบบหรือประวัติการเข้าใช้งาน* ***2. การใช้งาน (Usability)*** ● *ระบบจะประเมินความสามารถในการใช้งานผ่านการสำรวจผู้ใช้งานทุกประเภท* ***3. ประสิทธิภาพ (Performance)*** ● *การทดสอบด้านเวลาการตอบสนอง (Response Time)* ***4. ฟังก์ชันการทำงาน (Functionality)*** ● *ฟังก์ชันการทำงานจะถูกประเมินผ่านการตอบกลับของผู้ใช้จากระบบ* ***5. ความน่าเชื่อถือ (Reliability)*** ● *การทดสอบความมั่นคงปลอดภัยจะดำเนินการอย่างเพียงพอ*

6.4.24 กฎทางธุรกิจ (Business rules)

ระบุหลักการปฏิบัติงานใดๆ เกี่ยวกับซอฟต์แวร์ เช่น บุคคลหรือบทบาทใดสามารถทำหน้าที่ได้ภายใต้สถานการณ์เฉพาะสิ่งเหล่านี้ไม่ใช่ข้อกำหนดด้านการทำงานในตัวมันเอง แต่อาจบ่งบอกถึงข้อกำหนดการทำงานบางอย่างเพื่อบังคับใช้กฎ

ข้อกำหนดอื่นๆ (Other requirements)

กำหนดข้อกำหนดอื่นใดที่ไม่ครอบคลุมถึงส่วนอื่นใน SRS ซึ่งอาจรวมถึงข้อกำหนดฐานข้อมูล ข้อกำหนดความเป็นสากลข้อกำหนดทางกฎหมาย วัตถุประสงค์การใช้ซ้ำสำหรับโครงการ และอื่นๆ เพิ่มส่วนใหม่ที่เกี่ยวข้องกับโปรเจ็กต์

6.4.25 ภาคผนวก ก อภิธานศัพท์ (Glossary)

กำหนดคำศัพท์ทั้งหมดที่จำเป็นในการให้ความหมายอย่างถูกต้อง รวมถึงคำย่อและตัวย่อ ทั้งนี้ อาจจะสร้างอภิธานศัพท์แยกต่างหาก ซึ่งครอบคลุมหลายโครงการหรือทั้งองค์กรก็ได้

6.4.26 ภาคผนวก ข แบบจำลองการวิเคราะห์ (Analysis models)

หรือรวมถึงโมเดลการวิเคราะห์ที่เกี่ยวข้อง เช่น แผนภาพกระแสการไหลของข้อมูล (Data Flow Diagram) แผนภาพคลาส (Class Diagram) แผนภาพการเปลี่ยนสถานะ (State Diagram) หรือแผนภาพความสัมพันธ์ของเอนทิตี (ER-Diagram) เป็นต้น

6.4.27 ภาคผนวก ค รายการความต้องการที่กำหนดภายหลัง (To be determined list)

ในหัวข้อนี้อาจจะเกิดขึ้นในภายหลัง โดยเป็นการรวบรวมรายการต่างๆ ที่เป็นการเพิ่มเติมความเข้าใจนำไปสู่การพัฒนาที่สมบูรณ์มากขึ้น

เอกสารอ้างอิง

1. A. Lamsweerde, "Requirements Engineering from System Goals to UML Models to Software Specifications (Chapter 4)", Wiley, 2009, ISBN 978-0-470-01270-3.
2. B. Berenbach, J. Kazmeier, D. J. Paulish, and A. Rudorfer, "Software & Systems Requirements Engineering in Practice ", McGrawHill, 2009, ISBN 978-0-07-160547-2.
3. I. F. Alexander and R. Stevens, "Writing Better Requirements (Chapter 7)", Addison Wesley, 2002, ISBN 0-321-13163-0.
4. P. Tantatsanawong, “Fundamental of software engineering & digital transformation”, IDC Premier”, 2022, ISBN 978-6-16-487305-6.

แบบฝึกหัด

คำสั่ง ให้ใช้ความรู้ในบทที่ 6 การจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์ และจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์ โดยใช้ Template ที่กำหนดให้ ซึ่งให้เลือกกรณีศึกษาในภาคผนวก ก จำนวน 1 กรณีศึกษา เพื่อใช้เป็นโจทย์ในการจัดทำเอกสารข้อกำหนดการพัฒนาซอฟต์แวร์ โดยกรณีศึกษาประกอบไปด้วย

- *Case study 1: Library Management*
- *Case study 2: Train Control*
- *Case study 3: Meeting Scheduling*

สามารถดาวน์โหลด Template เอกสารข้อกำหนดความต้องการด้านซอฟต์แวร์ได้จากลิงค์ที่กำหนดให้ด้านล่าง

https://drive.google.com/drive/folders/1O1H9_Xo6XrZmSsqcewJQW7PbVxAGF1p5?usp=sharing

แผนบริหารการสอน
บทที่ 7 การตรวจสอบความต้องการ และการประกันคุณภาพความต้องการ

เนื้อหา

1. ทบทวนความต้องการ
2. สมาชิกของทีมทบทวน
3. รายการทบทวน

ผลลัพธ์การเรียนรู้รายบทเรียน

1. ผู้เรียนอธิบายเกี่ยวกับการทบทวน และตรวจสอบความต้องการได้
2. ผู้เรียนตรวจสอบความต้องการได้

วิธีการสอน

1. บรรยาย
2. ยกตัวอย่างระบบ โดยใช้กรณีตัวอย่างระบบที่เป็นที่รู้จัก
3. นิสิตร่วมอภิปรายแสดงความคิดเห็น

สื่อการสอน

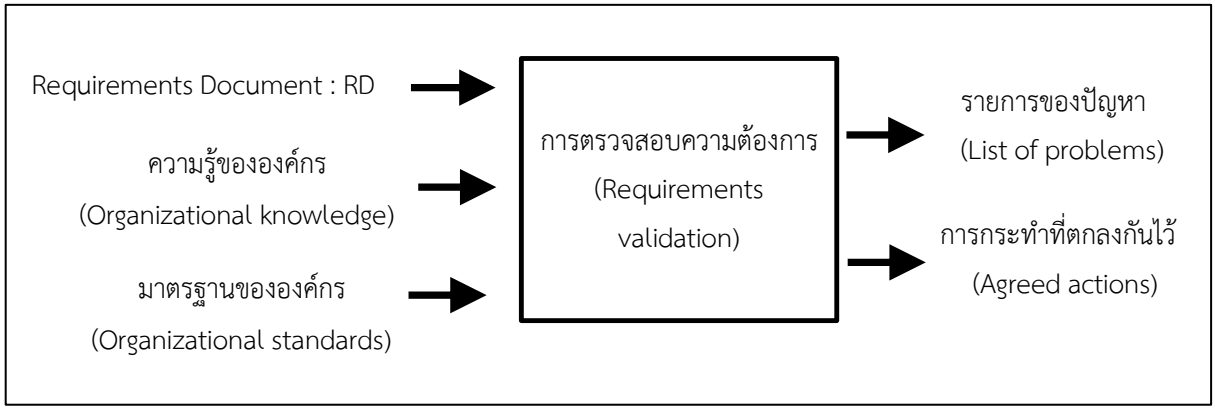
1. แฟ้มดิจิทัลนำเสนอ
2. ระบบมัลติมีเดีย เช่น คอมพิวเตอร์ เครื่องเสียง เครื่องฉายภาพ
3. เอกสารประกอบคำสอน

การวัดและการประเมินผล

1. คะแนนการเข้าชั้นเรียน
2. คะแนนแบบฝึกหัด
3. คะแนนการอภิปรายของนิสิต

บทที่ 7 การตรวจสอบความต้องการ และการประกันคุณภาพความต้องการ

การตรวจสอบความต้องการเป็นขั้นตอนสุดท้ายของการพัฒนาความต้องการ เป็นจุดมุ่งหมายของการตรวจสอบความต้องการ ที่จะรับรองว่าเอกสารความต้องการนั้นจะมีความสอดคล้อง ความสมบูรณ์ และความถูกต้อง



ภาพที่ 7.1 Inputs และ Outputs ของกระบวนการการตรวจสอบความต้องการ

จากภาพที่ 7-1 เป็นการแสดงถึง Inputs และ Outputs ของกระบวนการการตรวจสอบความต้องการ ซึ่ง Inputs หรือ ปัจจัยในการผลิตของกระบวนการการตรวจสอบความต้องการมีดังนี้

1) เอกสารความต้องการ (The requirements document) ควรเป็นเวอร์ชันที่สมบูรณ์ของเอกสารมากกว่าฉบับร่างที่ยังไม่เสร็จ ควรจะจัดรูปแบบและระเบียบตามมาตรฐานขององค์กร

2) มาตรฐานขององค์กร (The organizational standards) กระบวนการการตรวจสอบความต้องการควรสอดคล้องกับมาตรฐานของบริษัท ดังนั้นสิ่งที่เป็นมาตรฐานที่เกี่ยวข้องกับ RD ควรเข้าสู่กระบวนการตรวจสอบ

3) ความรู้ขององค์กร (Organizational Knowledge) เป็นปัจจัยที่ไม่เป็นรูปธรรม แต่ในทางปฏิบัติเป็นสิ่งที่สำคัญมาก คนที่เกี่ยวข้องในการตรวจสอบความต้องการ ต้องทราบว่าองค์กรมีศัพท์เฉพาะของวิธีปฏิบัติและทักษะของคนเกี่ยวข้องในการพัฒนาและใช้ระบบ เพราะความรู้นี้เป็นสิ่งสำคัญมากในการทำกระบวนการการตรวจสอบ

และในลำดับถัดมาคือ Outputs หรือผลที่ได้รับจากกระบวนการตรวจสอบความต้องการซึ่งประกอบไปด้วย 2 สิ่งดังต่อไปนี้

1) รายการของปัญหา (A problem list) คือรายการของรายงานปัญหาเกี่ยวกับเอกสารความต้องการ ควรจะจัดเป็นประเภทปัญหาเช่น ความคลุมเครือ ความไม่สมบูรณ์ เป็นต้น แต่ก็เป็นเรื่องยากที่จะจำแนกปัญหาต่างๆ ออกมาเป็นหมวดหมู่

2) การกระทำที่ตกลงกันไว้ (Agreed actions) สิ่งที่คนหรือองค์กรตกลงที่จะทำเพื่อแก้ไขหรือปรับปรุงสถานการณ์หรือเพื่อรับมือกับปัญหาที่เกิดขึ้นหรือกำลังเกิดขึ้นในอนาคต การทำ Agreed actions มักเกิดในบริบททางธุรกิจหรือองค์กรเพื่อแก้ไขปัญหาหรือเพื่อทำงานร่วมกันในโครงการหรือกิจกรรมที่ต้องการความร่วมมือจากหลายฝ่าย สามารถเป็นส่วนหนึ่งของหนังสือหรือข้อตกลงทางธุรกิจที่ระบุว่าใครจะทำอะไร อย่างไร และเมื่อใด เพื่อให้กิจกรรมหรือโครงการมีความสำเร็จตามวัตถุประสงค์ที่กำหนดไว้ในข้อตกลง

7.1 ทบทวนความต้องการ

การทบทวนความต้องการเป็นเทคนิคในการตรวจสอบความต้องการที่มีการใช้อย่างกว้างขวาง เกี่ยวข้องกับกลุ่มคนที่อ่านและวิเคราะห์ความต้องการ ซึ่งจะต้องมองหาปัญหา เพื่อประชุมหาข้อตกลงในการแก้ไขปัญหาที่พบ

ขั้นตอนของรูปแบบกระบวนการทบทวนความต้องการที่สำคัญ ดังนี้

1) ทบทวนแผน : ทีมทบทวนจะเลือกเวลา และสถานที่สำหรับการประชุมทบทวน

2) กระจายเอกสาร : กระจายเอกสารความต้องการและเอกสารอื่นๆ ที่เกี่ยวข้องให้กับสมาชิกในทีม

3) เตรียมพร้อมสำหรับการทบทวน : ร่วมกันอ่านและแสดงความคิดเห็นความต้องการที่จะบ่งบอกถึงความขัดแย้ง การละเว้น หรือปัญหาอื่นๆ

4) ประชุมทบทวน : ร่วมกันแสดงความคิดเห็นหาวิธีการเพื่อดำเนินการแก้ไขปัญหาที่เกิดขึ้น

5) ติดตามการดำเนินงาน : หัวหน้าการทบทวนจะตรวจสอบว่าการกระทำที่ได้ตรงลงกันไว้ ได้รับการดำเนินการหรือไม่

6) แก้เอกสาร : ปรับปรุงเอกสารความต้องการที่สะท้อนให้เห็นถึงการทำงานที่ได้ตกลงกันไว้ ซึ่งขั้นตอนนี้อาจจะได้รับการยอมรับ หรือต้องกลับไปทบทวนอีกครั้ง

การประชุมอย่างเป็นทางการในการทบทวนความต้องการ ประธานควรเป็นบุคคลที่ไม่มีส่วนเกี่ยวข้องในการผลิตความต้องการที่กำลังจะทำให้ถูกต้อง ในระหว่างการประชุมวิศวกรความต้องการจะนำเสนอแต่ละความต้องการตามลำดับ รวมไปถึงความคิดเห็นต่างๆ นอกจากนี้ต้องมีการระบุปัญหาที่พบและบันทึกปัญหาสำหรับการประชุมในครั้งถัดไป ซึ่งในทีมควรมีสมาชิกที่ทำหน้าที่เลขาเพื่อคอยจดบันทึกความต้องการหรือปัญหาต่างๆ ที่ถูกถกขึ้นในที่ประชุม เวลาในการประมาณการเวลาในการทวนสอบความต้องการ อาจจะเป็น 20-40 ความต้องการ ต่อ 1 ชั่วโมง ขึ้นอยู่กับขนาดของความต้องการ

7.2 สมาชิกของทีมทบทวน

การเลือกสมาชิกที่เหมาะสมสำหรับทีมทบทวนความต้องการเป็นสิ่งที่สำคัญ จะเป็นการดีที่เอกสารจะถูกตรวจสอบโดยทีมงานที่มีความแตกต่างกัน ที่มาจากบุคคลที่มีประสบการณ์ที่ต่างกัน ถ้าเป็นไปได้ ควรจะมีผู้ใช้ระบบหรือตัวแทนของลูกค้า วิศวกรออกแบบระบบ และวิศวกรความต้องการอยู่ในทีม

ข้อดีของผู้ที่มีส่วนได้เสียที่เกี่ยวข้อง

1) ผู้ที่มีประสบการณ์ที่แตกต่างกันจะนำมาซึ่งความสามารถที่แตกต่างกัน ความรู้และประสบการณ์ในการทบทวนความต้องการอาจจะสามารถทำพบปัญหาต่างๆ

2) หาก Stakeholders ที่มีประสบการณ์ที่ต่างกันมีส่วนร่วมในกระบวนการตรวจสอบและเข้าใจในความต้องการของผู้ที่มีส่วนได้ส่วนคนเสียอื่นๆ จะมีแนวโน้มที่จะเข้าใจการเปลี่ยนแปลงความต้องการ

ทีมตรวจสอบควรประกอบไปด้วย Stakeholders ที่แตกต่างกันในการรวบรวมความต้องการ นักพัฒนาระบบควรจะมีส่วนเกี่ยวข้องในระยะนี้เพราะอาจจะพบความต้องการที่ยากต่อการดำเนินงาน ความต้องการเหล่านี้ถ้าหากค้นพบหรือแก้ไขก่อนที่จะออกแบบ จะสามารถคำนวณจำนวนแรงงานและค่าใช้จ่ายได้ จำนวนทีมตรวจสอบจะมีสมาชิกมากหรือน้อยขึ้นอยู่กับขนาดของระบบและจำนวนของ Stakeholders ที่มีแนวโน้มได้รับผลกระทบจากระบบ

การตรวจสอบจะต้องมีกลุ่มคนที่มีทักษะที่ต่างกันและมีความรับผิดชอบในการอ่านเอกสาร ทีมตรวจสอบจะรวมตัวกันในสถานที่และเวลาเดียวกันเพื่อดำเนินการตรวจสอบ สมาชิกทีมทบทวนอาจจะทำงานองค์กรที่แตกต่างกันหรือคนละภาคส่วนซึ่งเป็นเรื่องยากในการทำการทบทวน ดังนั้นการทบทวนอาจจะต้องมีการประนีประนอมซึ่งกันและกันดีกว่าเถียงกันจนทำให้เกิดความล่าช้า

7.3 รายการทบทวน

การใช้รายการในการตรวจสอบซึ่งอธิบายลักษณะและข้อผิดพลาดที่เกิดขึ้นบ่อย เป็นเทคนิคที่มีประสิทธิภาพมาก เพราะโดยปกติผู้ปฏิบัติงานมักจะทำข้อผิดพลาดเดิมๆ

ในตารางที่ 7-1 จะแสดงรานการตรวจสอบ (Checklist) ลักษณะคุณภาพของความต้องการ ที่จำเป็นต้องตรวจสอบ

ตารางที่ 7.1 คุณสมบัติหรือลักษณะที่ดีของความต้องการ

#	คุณภาพที่พึงประสงค์	ความหมาย
1.	Completeness	การระบุและรวบรวมความต้องการทั้งหมดที่เกี่ยวข้องในโครงการหรือซอฟต์แวร์อย่างครอบคลุม โดยไม่ตกหล่น หรือขาดหายไป
2.	Consistency	ควรเขียนความต้องการที่มีความสอดคล้องกัน มีความหมายไปในทิศทางเดียวกัน สอดคล้องกัน
3.	Adequacy	เอกสารความต้องการควรระบุความต้องการในรายละเอียดที่เพียงพอ เพื่อให้มีความเหมาะสม และเป็นประโยชน์ต่อโครงการหรือซอฟต์แวร์ที่กำลังพัฒนา
4.	Unambiguity	จะต้องเขียนหรืออธิบายให้ชัดเจน ไม่คลุมเครือ โดยเฉพาะอย่างยิ่งคำจำกัดความ หรือการนิยามเทอมที่เกี่ยวข้อง เพื่อให้ความต้องการถูกอธิบายอย่างชัดเจนและไม่มีความกำกวมหรือความสับสนในการอ่านหรือเข้าใจ
5.	Measurability	จะต้องถูกเขียนหรือสร้างขึ้นมา ในระดับที่จะทำให้สามารถประเมินหรือวัดผล เพื่อตรวจและทวนสอบ รวมถึงทดสอบได้
6.	Pertinence	ความต้องการต้องสอดคล้องกับวัตุประสงค์ของระบบ ภายใต้บริบทของ Problem world และ System world
7.	Feasibility	จะต้องมีความเป็นไปได้ (ไม่เพ้อฝัน) ภายใต้ข้อจำกัดต่างๆ (Constraints) เช่น งบประมาณ (Budget) เวลา (Schedule) และเทคโนโลยี
9.	Good structuring	เอกสารความต้องการ (Requirements document) จะต้องมีโครงสร้างที่เชื่อมต่อกันในองค์ประกอบที่เกี่ยวข้องกัน
10.	Modifiability	เอกสารความต้องการจะต้องสามารถนำมาปรับแก้ไข เปลี่ยนแปลง และเพิ่มเติมได้ (Revise, adapt, and extend) ในระยะเวลาที่เหมาะสม และไม่ทำให้ System-to-be เสียหาย
11.	Traceability	เนื้อหาต่างๆ ในเอกสารความต้องการ ไม่ว่าจะเป็นการเขียนใหม่ หรือการปรับแก้ จะต้องสามารถตรวจสอบย้อนกลับได้

เอกสารอ้างอิง

1. A. Lamsweerde, "Requirements Engineering from System Goals to UML Models to Software Specifications (Chapter 5)", Wiley, 2009, ISBN 978-0-470-01270-3.
2. G. Kotonya and I. Sommerville, "Requirements Engineering: Processes and Techniques (Chapter 4)", John Wiley & Sons, 1997, ISBN 0-471-97208-8.
3. I. F. Alexander and R. Stevens, "Writing Better Requirements (Chapter 8)", Addison Wesley, 2002, ISBN 0-321-13163-0.

แบบฝึกหัด

คำสั่ง ให้นิสิตใช้เอกสารความต้องการจากแบบฝึดหัดในบทที่ 6 มาใช้ในการตรวจสอบความต้องการตาม Checklist ที่กำหนดให้ พร้อมระบุผลจากการตรวจสอบ

#	คุณภาพที่พึงประสงค์	ผลการตรวจสอบ
1.	Completeness	
2.	Consistency	
3.	Adequacy	
4.	Unambiguity	
5.	Measurability	
6.	Pertinence	
7.	Feasibility	
9.	Good structuring	
10.	Modifiability	
11.	Traceability	

ส่วนที่ 3 การจัดการความต้องการ
(Requirements Management)

ผลลัพธ์การเรียนรู้ของส่วนนี้

บทที่ 8 การจัดการความต้องการ (Requirements Management)

- การจัดระเบียบและเก็บรวบรวมความต้องการ
- การเปลี่ยนแปลงความต้องการ

แผนบริหารการสอน
บทที่ 8 การจัดการความต้องการ

เนื้อหา

1. การจัดระเบียบและเก็บรวบรวมความต้องการ
2. การเปลี่ยนแปลงความต้องการ

ผลลัพธ์การเรียนรู้รายบทเรียน

1. ผู้เรียนอธิบายความต้องการที่เสถียร กับความต้องการที่แปรเปลี่ยนได้
2. ผู้เรียนระบุและจัดเก็บความต้องการได้
3. ผู้เรียนจัดการกับการเปลี่ยนแปลงความต้องการได้

วิธีการสอน

1. บรรยาย
2. ยกตัวอย่างระบบ โดยใช้กรณีตัวอย่างระบบที่เป็นที่รู้จัก
3. นิสิตร่วมอภิปรายแสดงความคิดเห็น

สื่อการสอน

1. แฟ้มดิจิทัลนำเสนอ
2. ระบบมัลติมีเดีย เช่น คอมพิวเตอร์ เครื่องเสียง เครื่องฉายภาพ
3. เอกสารประกอบคำสอน

การวัดและการประเมินผล

1. คะแนนการเข้าชั้นเรียน
2. คะแนนแบบฝึกหัด
3. คะแนนการอภิปรายของนิสิต

บทที่ 8 การจัดการความต้องการ

การเปลี่ยนแปลงความต้องการด้านซอฟต์แวร์ถือว่าเป็นเรื่องปกติที่เกิดขึ้นในแทบทุกๆ โครงการ น้อยมากที่ความต้องการจะยังคงเหมือนเดิมตั้งแต่เริ่มต้น ทั้งนี้เพราะวัตถุประสงค์ในการพัฒนาซอฟต์แวร์นั้น ก็เพื่อนำมาใช้ในการแก้ไขปัญหาหรือเพิ่มความสะดวกสบายในการดำเนินงาน ซึ่งมีความเป็นไปได้น้อยมากที่ปัญหาทั้งหมดจะถูกค้นหาได้อย่างสมบูรณ์ตั้งแต่เริ่มต้น อีกทั้งยังอาจเกิดปัญหาใหม่ๆ ได้ตลอดเวลา อันเนื่องมาจากธุรกิจยังคงดำเนินกิจการอยู่ ซึ่งการมีปัญหาเกิดขึ้นแสดงว่าธุรกิจนั้นยังมีการเจริญเติบโต แต่ถ้าหากไม่มีปัญหาใหม่ แสดงว่าธุรกิจนั้นเริ่มที่จะถดถอย

ในระหว่างกระบวนการซอฟต์แวร์ผู้มีส่วนได้ส่วนเสียมีความเข้าใจในปัญหาที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา ดังนั้น ข้อกำหนดด้านซอฟต์แวร์จะต้องมีการจัดการความต้องการ เพื่อรองรับปัญหาที่เปลี่ยนแปลงนี้ ซึ่งประกอบด้วย 2 ขั้นตอน ดังนี้

1. การวางแผนการจัดการความต้องการ (Requirement Management Planning) เป็นการวางแผนการวางแผนเป็นขั้นตอนแรกที่สำคัญในกระบวนการจัดการความต้องการ โดยต้องทำการนิยามปัญหาที่เกิดขึ้นและเชื่อมโยงกับระบบที่มีอยู่ เพราะในบางครั้งการแก้ปัญหาหนึ่งอาจจะส่งผลกระทบต่อระบบอื่น ซึ่งก่อให้เกิดปัญหาแทรกซ้อนได้ เมื่อได้ปัญหาที่แท้จริงแล้ว จึงค่อยมาประมาณการงบประมาณที่จะต้องใช้ต่อไป ทั้งนี้ ความต้องการที่เปลี่ยนแปลงนั้นไม่ถือเป็นส่วนหนึ่งของการบำรุงรักษาระบบ นั่นหมายถึงว่าต้องมีกระบวนการพัฒนาซอฟต์แวร์ตามที่ได้กล่าวมาแล้วอีกครั้งหนึ่ง

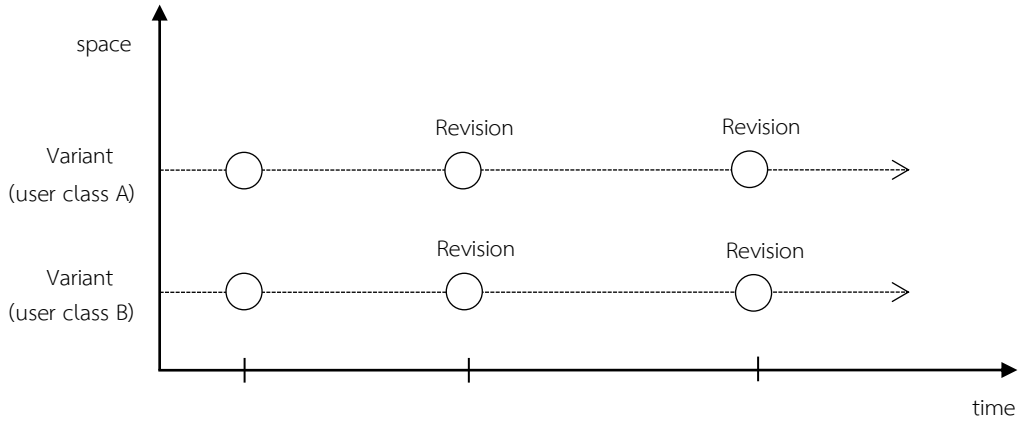
2. การจัดการความเปลี่ยนแปลงความต้องการ (Requirements Change Management) มี 3 ขั้นตอนคือ

- การวิเคราะห์ปัญหาและเปลี่ยนแปลงข้อกำหนด ขั้นตอนนี้เริ่มด้วยการระบุปัญหาเกี่ยวกับข้อกำหนดหรือบางครั้งอาจมีการเปลี่ยนแปลงที่ผู้พัฒนาระบบเสนอด้วยก็ได้ ระหว่างขั้นตอนนี้จะมีวิเคราะห์ปัญหาหรือข้อเสนอการเปลี่ยนแปลงเพื่อตรวจสอบว่าเป็นถูกต้อง ผลการวิเคราะห์นี้จะส่งกลับไปยังผู้ที่ร้องขอการเปลี่ยนแปลง เพื่อพิจารณาว่าจะยอมรับข้อกำหนดที่เปลี่ยนแปลง หรือตัดสินใจที่จะยกเลิกการเปลี่ยนแปลงก็ได้
- การวิเคราะห์การวิเคราะห์การเปลี่ยนแปลงและประมาณการค่าใช้จ่าย ในขั้นตอนนี้จะมีการประเมินการเปลี่ยนแปลงที่เสนอ รวมถึงผลกระทบกับระบบเดิม เพื่อนำมาประมาณการค่าใช้จ่ายในการเปลี่ยนแปลง ซึ่งต้องประมาณการทั้งในแง่ของการปรับเปลี่ยนเอกสารข้อกำหนด การออกแบบระบบและการนำไปใช้งานจริงเมื่อการวิเคราะห์เสร็จสิ้นจะมีการตัดสินใจว่าจะดำเนินการต่อไปหรือไม่
- ดำเนินการเปลี่ยนแปลงข้อกำหนด ขั้นตอนนี้ ผู้พัฒนาระบบจะจัดทำข้อกำหนดใหม่ที่ได้แก้ไขแล้ว เพื่อที่จะนำไปเข้าสู่กระบวนการการออกแบบและนำไปใช้จริง

8.1 การจัดระเบียบและเก็บรวบรวมความต้องการ

การจัดระเบียบและเก็บรวบรวมความต้องการ (Requirements organization and collection) เป็นขั้นตอนสำคัญในการบริหารจัดการความต้องการ (Requirements management) เป็นการนำข้อมูลการจัดระเบียบและการเก็บรวบรวมความต้องการ เพื่อให้มั่นใจมีความต้องการใดๆ ที่สำคัญตกหล่น หรือสูญหายไป

8.2 การเปลี่ยนแปลงความต้องการ



ภาพที่ 8.1 การเปลี่ยนแปลงความต้องการ

จากภาพที่ 8.1 เป็นการแสดงให้เห็นภาพของการวิวัฒนาการของการเปลี่ยนแปลง สามารถแบ่งได้เป็น 2 มิติ ซึ่งจะแสดงเป็นจุดบนเส้นเวลาของการใช้งานระบบ ประกอบด้วยการแปรผันอาจเกิดขึ้นในเวลาที่แตกต่างกัน และส่วนในการแก้ไขความต้องการ จะมีการแสดงเวอร์ชันกำกับ โดยในการแก้ไขแต่ละครั้ง จะต้องมีจุดบนกราฟที่ไม่ใช่จุดเดียวกัน

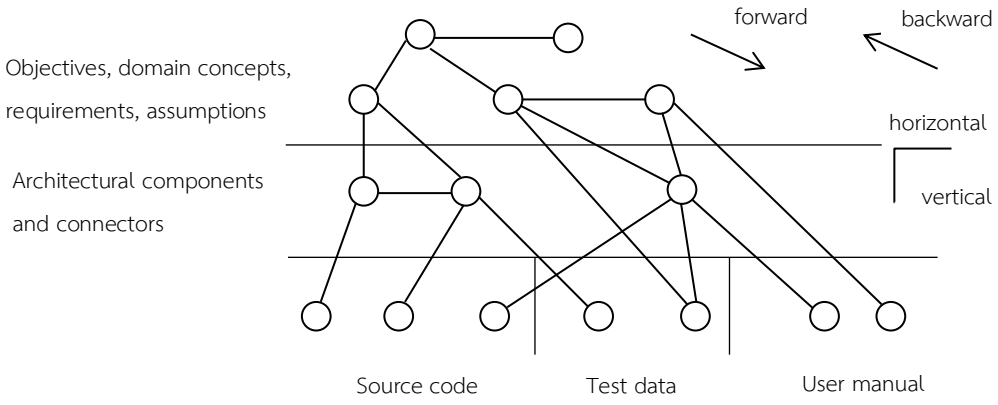
- ผลจากการปรับปรุงเวอร์ชัน (Revision) จะทำให้เวอร์ชันปัจจุบันของซอฟต์แวร์นั้นๆ มีการพัฒนาที่ดีหรือมีความเหมาะสมมากยิ่งขึ้น
- การดัดแปลง (Variants) เกิดจากความต้องการที่จะนำเอกสารต้นฉบับมาดัดแปลง จำกัดขอบเขตหรือขยาย เพื่อจำนวนผู้ใช้งานหรือเงื่อนไขเพิ่มเติม

ตารางที่ 8.1 การเปลี่ยนแปลงของเอกสารความต้องการ

No.	ปัจจัยเชิงสาเหตุ	ประเภทของการเปลี่ยนแปลง	ประเภทของเวอร์ชัน	ช่วงเวลาในการเปลี่ยนแปลง
1.	Errors and flaws in the RD	Correction	Revision	RE, design, implementation, after development
2.	Better customer understanding	Correction, extension	Revision	RE, (design), (implementation), after development
3.	New functional feature	Extension	Revision, variant	(RE), (design), (implementation), after development
4.	Improved quality feature	Amelioration	Revision	RE, design, (implementation), after development
5.	Environmental change: new class of user or new usage condition	Adaptation, contraction, extension	Variant	RE, design, (implementation), after development
6.	Environmental change: new way of doing things	Adaptation, extension	Revision	(RE), (design), (implementation), after development
7.	Environmental change: alternative way of doing thing	Adaptation	Variant	RE, (design), (implementation), after development
8.	Environmental change: new regulation	Adaptation	Revision	(RE), (design), (implementation), after development
9.	Environmental change: alternative	Adaptation	Variant	(RE), (design), (implementation), after development

No.	ปัจจัยเชิงสาเหตุ	ประเภทของการเปลี่ยนแปลง	ประเภทของเวอร์ชัน	ช่วงเวลาในการเปลี่ยนแปลง
	regulation (e.g. in another country)			
10.	Environmental change: organizational changes	Adaptation	Revision	(RE), (design), (implementation), after development
11.	Environmental change: new technology	Adaptation, extension	Revision	(RE), (design), (implementation), after development
12.	Environmental change: alternative technology	Adaptation	Variant	RE, design, implementation, after development
13.	Environmental change: new opportunities	Extension	Revision, Variant	(RE), (design), (implementation), after development
14.	Process fluctuations: change in priorities, schedules or cost constraints	Contraction, Adaptation	Revision	RE, design, implementation

8.3 การเชื่อมโยง/การตรวจสอบย้อนกลับความต้องการ



ภาพที่ 8.2 การเชื่อมโยง/การตรวจสอบย้อนกลับความต้องการ

จากภาพที่ 8.2 นั้นอธิบายถึงการตรวจสอบย้อนกลับทั้งแบบแนวตั้ง และแบบแนวนอน โดยที่การเชื่อมโยงแบบแนวนอนหรือการเชื่องโยงกันเองระหว่างวัตถุประสงค์ (Objectives) และขอบเขตของแนวคิด (Domain concepts) เรียกว่าการทวนสอบแบบแนวนอน ส่วนการเชื่อมโยงระหว่างวัตถุประสงค์และองค์ประกอบของโครงสร้าง (Architectural components) เรียกว่าการทวนสอบแบบแนวตั้ง และการทวนสอบแบบไปข้างหน้า (Forward) จะเป็นการทวนจากวัตถุประสงค์ไปสู่โค้ดของโปรแกรม ส่วนการทวนสอบไปข้างหลัง (Backward) จะเป็นการทวนสอบจากโค้ดของโปรแกรมไปสู่วัตถุประสงค์

ความหมายของการตรวจสอบย้อนกลับในทิศทาง Forward Backward Horizontal and Vertical เป็นสิ่งสำคัญ พิจารณาได้จาก

- การใช้การตรวจสอบย้อนกลับแบบ Backwards กู้คืนข้อมูลรายละเอียดหรือแหล่งที่มา (Source) เพื่อการสร้างหรือการแก้ไขอย่างง่ายดาย และจะสามารถตอบคำถามเช่น “รายการนี้มาจากไหน” โดยที่รายการเป้าหมาย (Target item) จะระบุถึงรายการแหล่งที่มา (Source item) เพื่อใช้อธิบาย และสามารถนำข้อมูลกลับมาได้

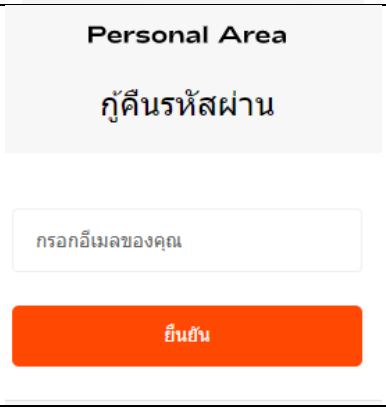
- การติดตามความเชื่อมโยงการตรวจสอบย้อนกลับแบบ Forwards จะทำให้รู้ว่ารายการที่ถูกติดตามจะส่งผลกับรายการอะไรบ้าง ทำให้สามารถระบุรายการอื่นๆ ที่มีอยู่ได้จาก Source item หรือรายการที่ถูกติดตาม

- การตรวจสอบย้อนกลับแบบ Horizontal และ Vertical traceability สามารถจำกัดผลกระทบของการสร้าง การปรับปรุง หรือการลบรายการ เพื่อใช้ในการประเมิน และทำการแก้ไขข้อมูลต่าง ๆ ที่ได้รับผลกระทบ

8.4 การตรวจสอบย้อนกลับของผลลัพธ์กับความต้องการ

การตรวจสอบย้อนกลับของผลลัพธ์กับความต้องการ (Requirement Traceability Matrix : RTM) เป็นเอกสารอีกฉบับหนึ่งที่ติดตามความต้องการของผู้ใช้ โดยจะพิจารณาข้อกำหนดทั้งหมดที่ลูกค้าเสนอและความสามารถในการตรวจสอบย้อนกลับความต้องการในเอกสารฉบับเดียว และส่งมอบเมื่อสิ้นสุดวงจรการพัฒนาซอฟต์แวร์ จุดประสงค์หลักของ RTM คือการตรวจสอบว่าข้อกำหนดทั้งหมดได้รับการดำเนินการจนเสร็จสิ้น ดังแสดงในตารางที่ 8.1

ตาราง 8.1 Requirement Traceability Matrix

SRS No.	Use Case Reference	Design Document Reference	Module Reference	Test Case Reference	Display
REQ-01	UC. 3 เข้าสู่ระบบ	page 3, page 10 ,page 14	การเข้าสู่ระบบ (Login : L)	TC04-001 - TC04-007	
REQ-01.1	UC. 3.1 เปลี่ยนรหัสผ่าน	Page 4	การเข้าสู่ระบบ (Login : L)	TC05-001 - TC05-002	

ภาคผนวก ก

กรณีศึกษาสำหรับจัดทำแบบฝึกหัดที่ 6

ผู้แต่งใช้กรณีศึกษาจากหนังสือ

Requirements Engineering From System Goals to UML Models to Software Specifications

ผู้แต่ง Axel van Lamsweerde

Wiley, ISBN 978-0-470-01270-3, 2009.

(For Educational Purposes Only)

Case study 1: Library Management

The University of Wonderland (UWON) wants to convert its library system into a new system to ensure more effective access to state-of-the-art books, periodicals and proceedings while reducing operational costs. The current system consists of multiple unconnected library subsystems, one for each UWON department. Each department subsystem is responsible for its own library according to department-specific procedures for book acquisition user registration, loan management, bibliographical search and access to library resources. Such services are essentially manual in most UWON libraries. They rely on card indexes maintained by library staff according to some keyword-based classification scheme. Such schemes are specific to each department. A few departments are using rudimentary file-based software written by their members.

Some of the complaints about the current system as reported by university authorities, library staff, department members or students include the following:

- Unnecessary duplicate acquisition, by several departments, of infrequently accessed copies of books or proceedings that are relevant to more than one department.
- Unnecessary subscription, by several departments, to expensive journals that are relevant to more than one department.
- Acquisition of books or proceedings of marginal interest to the university, which could be borrowed from other universities with which UWON has an agreement.
- Subscription to journals of marginal interest to the university, which could be accessed in other universities with which UWON has an agreement.
- Unavailability of requested books, for a variety of reasons such as department budget restrictions, excessive borrowing by the same user, lack of enforcement of rules limiting loan periods, loss or stealing of book copies and so on.
- Unavailability of journal issues while they are being bound into yearly volumes.
- Lack of traceability to previous borrowers when books, proceedings or journal volumes are found to be damaged.
- Inaccuracy of card indexes, e.g. a book is stated as being available whereas it is not found at the appropriate place in the shelves.
- Bibliographical search restricted to library opening hours.
- Slow, tedious bibliographical search due to manipulation of card indexes.
- Inaccurate search results, due to poor classification of books, journals or proceedings within departments.

- Incomplete or ineffective search results, due to relevant books, journals or proceedings being indexed in other UWON department libraries, or unavailable at UWON.

The new UWON library system should address such problems through a software-based solution integrating all department libraries. The new system should interoperate with library systems from partner universities. It should provide interactive online facilities for book acquisition, user registration, loan management, bibliographical search and book reservation. Access to such facilities should be restricted to specific user categories, according to authorization rules specific to each facility.

The new system should take advantage of opportunities provided by new technologies. In particular, it should support subscriptions to e-journals, provide access to foreign digital libraries (under specific conditions), support e-mail communication between staff and users, enable bibliographical search from anywhere at any time, and provide a Web-based interface for book e-seller comparison, selection, and order submission.

Case study 2: Train Control

Traffic at Wonderland airport (WAX) has increased drastically over the past few years. The increase in the number of companies and flights calls for the building of new terminals. The increase in the number of passengers calls for a new transportation system between all terminals, and between the main terminal and Wonderland City. The current bus transportation system has reached its limits in terms of transportation capacity and quality of service. Buses are slow and often late due to traffic jams; passengers need to stand in long queues, sometimes for an unacceptably long time, which may cause them to miss flight connections, and so on.

The government of Wonderland has decided to replace bus transportation by a train-based system. The envisaged system is aimed at increasing transportation capacity, speed and quality of service. The decision is also motivated by recent regulations for reducing greenhouse gas production

Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality, higher frequency and better information to passengers.

A consortium has been set up to undertake this project. It brings together government representatives, airport authorities, Wonderland Railways and the engineering company selected to implement the project. The latter is subcontracting the software part of it to a software house.

In the new system, all terminals will be interconnected through an underground circular, one-track railway. The main terminal and city terminal will be interconnected by a two-track line (one for each direction). The main terminal also has parking tracks for inactive trains, servicing and so on. Each track is divided into track segments of a fixed size called blocks. Each terminal holds one block called a station block. Each block is equipped with an entry signal (or 'virtual gate') and multiple sensors to detect the presence of trains, identify trains and their speed and so on.

The envisioned software is expected to control the acceleration of trains, the opening of train doors, the block signals and the display of information about the current/next station on information panels inside trains. The railway company would also like to reduce operational costs. A fully automated, driverless option is envisaged as an alternative to the standard option. In this standard option, train drivers have to follow recommendations issued by the software and respond to regular stimuli issued by the software to check driver responsiveness. The driverless option is currently being discussed with the unions.

Various concerns about the new system have already emerged at this preliminary stage:

- In order to ensure rapid transportation of passengers, trains should run fast, without unnecessary delays, and at high frequency, during rush hours at least.
- In order to guarantee safe transportation of passengers, the probability of accidents must fall below the threshold imposed by safety regulations. In particular, the distance between two trains following each other must always be sufficient to prevent the back train from hitting the front train in case the latter stops suddenly. The speed of a train on a particular block may never exceed the

limit associated with that block. Trains may never enter a block whose entry signal is set to 'stop'. Train doors must always be kept closed while a train is moving.

- In order to ensure comfortable transportation, trains should accelerate/decelerate smoothly. Passengers at a station should be informed in time about trains arriving. Passengers inside a train should be informed in time that the train is departing, which companies are being served by the next stop and so on.

Case study 3: Meeting Scheduling

With the advent of globalization, companies and organizations are increasingly distributed over multiple sites and countries. Wonderland Software Services (WSS) has identified a large potential market for meeting-scheduling software that would exploit Internet-based communication technologies. Scheduling meetings with busy people is generally a nightmare. It is hard to find a date and a place that suit everyone's constraints; meeting organizers need to pester people to get their availability; other people are unnecessarily inconvenienced by messages that do not concern them; when the meeting is scheduled some constraints have changed in the meantime; new scheduling cycles need to be repeated when no date/location is found in a reasonably short period; and so forth. As a result, meetings tend to be organized poorly and late; important people sometimes do not show up; and there is a significant, unnecessary overhead in the scheduling process.

Meetings are typically scheduled as follows. A meeting initiator informs potential participants about the need for a meeting and specifies a date range within which the meeting should take place, asking people to return their availability constraints within that time interval. Constraints are typically expressed as two sets: an exclusion set specifying dates within the date range when the participant could not attend, and an optional preference set specifying dates within the date range on which the participant would prefer the meeting to take place (a date may refer to a full day or a period in a day). In some cases, the initiator may also ask participants who will play an active role in the meeting for specific requirements regarding the meeting room (e.g. projector, laptop, network connection, videoconferencing facilities etc.). 'Important participants may optionally be asked to state preferences for meeting locations.

The scheduled meeting date should belong to the stated date range and to none of the exclusion sets; it should ideally belong to as many preferences sets as possible. The meeting venue should ideally fit the preferences of important participants. A date conflict occurs when no date can be found outside all exclusion sets. A room conflict occurs when no room can be found, at any date outside all exclusion sets, which meets the room requirements. Conflicts can be resolved in several ways: the initiator may extend the date range, some participants may remove dates from their exclusion set, or some participants may decline the invitation to attend. A new scheduling cycle may thus be required in case of conflict.

The envisioned meeting scheduler software should reflect as closely as possible the way meetings are typically managed. It should be useable by administrative staff and provide major improvements in several respects:

- Average participant attendance should increase thanks to the selection of meeting dates and locations that are the most convenient to potential participants.
- Meetings should be scheduled as quickly as possible once they are initiated.
- Meeting dates and locations should be notified as quickly as possible to all potential participants once they are scheduled. In all cases, there should be sufficient time between notification and the meeting date.

- The organizational overhead should be kept as low as possible on the initiator's side. In particular, the meeting scheduler should support all required interactions with participants, for example to communicate requests, get replies (even from participants not reacting promptly), assist in negotiation and conflict-resolution processes, and inform participants on request about the state of the scheduling process.
- The amount of interaction with potential participants for meeting scheduling, in number and length of messages, should be kept as small as possible.

The new meeting scheduler must be able to handle multiple meeting requests in parallel. Meeting requests can be competing by overlapping in time or space. Concurrency must thus be managed under physical constraints; a person may not be at two different places at the same time, and a meeting room may not be allocated to more than one meeting at a time.

To allow as much flexibility as possible, dynamic replanning of meetings should be supported. On the one hand, participants should be allowed to modify their exclusion set, preference set and/or preferred location until the meeting is scheduled. On the other hand, exceptional constraints should be accommodated after a meeting is scheduled, such as the need to schedule an urgent, more important meeting. The original meeting date or location may then need to be changed; sometimes it may even be cancelled. In all cases some way of replanning should be set up.

The system should be flexible enough to accommodate different data formats (e.g. date or address formats) and evolving data (e.g. the set of concerned participants may vary during the scheduling process, and the address at which a participant can be reached may change).

There are also security concerns to be taken into account, such as the following:

- Meeting initiation should be restricted to authorized personnel.
- Confidentiality rules should be enforced, for instance a non-privileged participant should not be aware of constraints stated by other participants, or of other meetings to which the latter are invited.

Rather than a single product, WSS is thinking of a product family. The customization space should cover the following variations:

- Professional meetings, private meetings.
- Single-site meetings, meetings where the target site needs to be determined as well, electronic meetings.
- Regular meetings (e.g. for a university course), occasional meetings.
- Single-level meetings or multi-level meetings where the importance of a attending a specific meeting is higher or lower with respect to other meetings.
- Single-level participation or multi-level participation where the importance of a meeting getting a specific attendee is higher or lower with respect to other attendees.

- Single-level participants or multi-level participants where some participants are hierarchically more important than others (regardless of a specific meeting) or have less flexibility in changing their constraints.
- Variations on what participating in a meeting means, e.g. full attendance, partial attendance, participation through delegation.
- Variations on what constraints are about, e.g. no preference set, unordered preferences, ordered preferences, date availability dependent on meeting location.
- Parameterizability on explicit conflict-resolution rules that are tunable by the client, e.g. 'best meeting dates and locations should be determined by considering participants with higher importance first', in case of date conflict the scheduler will propose a person of lower importance to withdraw from the meeting', 'in case of date conflict the meeting scheduler will propose a participant to withdraw from another meeting of lower importance', a date within some exclusion set will be considered if the corresponding participant has high flexibility'.
- Mono-lingual, multi-lingual communication with participants.
- Variations on additional features such as support for elaborating the meeting agenda or the meeting minutes.

ภาคผนวก ข

IEEE 830-1998

IEEE Recommended Practice for Software Requirements Specifications

(For Educational Purposes Only)

IEEE Std 830-1998
(Revision of
IEEE Std 830-1993)

IEEE Std 830-1998

IEEE Recommended Practice for Software Requirements Specifications

IEEE Computer Society

Sponsored by the
Software Engineering Standards Committee

20 October 1998 SH94654

IEEE Std 830™-1998(R2009)
(Revision of
IEEE Std 830-1993)

IEEE Recommended Practice for Software Requirements Specifications

Sponsor

Software Engineering Standards Committee of the IEEE Computer Society

Reaffirmed 9 December 2009
Approved 25 June 1998

IEEE-SA Standards Board

Abstract: The content and qualities of a good software requirements specification (SRS) are described and several sample SRS outlines are presented. This recommended practice is aimed at specifying requirements of software to be developed but also can be applied to assist in the selection of in-house and commercial software products. Guidelines for compliance with IEEE/EIA 12207.1-1997 are also provided.
Keywords: contract, customer, prototyping, software requirements specification, supplier, system requirements specifications

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

ISBN 0-7381-0332-2

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Introduction

(This introduction is not a part of IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.)

This recommended practice describes recommended approaches for the specification of software requirements. It is based on a model in which the result of the software requirements specification process is an unambiguous and complete specification document. It should help

a) Software customers to accurately describe what they wish to obtain;
b) Software suppliers to understand exactly what the customer wants;
c) Individuals to accomplish the following goals:
    1) Develop a standard software requirements specification (SRS) outline for their own organizations;
    2) Define the format and content of their specific software requirements specifications;
    3) Develop additional local supporting items such as an SRS quality checklist, or an SRS writer's handbook.

To the customers, suppliers, and other individuals, a good SRS should provide several specific benefits, such as the following:

— *Establish the basis for agreement between the customers and the suppliers on what the software product is to do*. The complete description of the functions to be performed by the software specified in the SRS will assist the potential users to determine if the software specified meets their needs or how the software must be modified to meet their needs.

— *Reduce the development effort*. The preparation of the SRS forces the various concerned groups in the customer's organization to consider rigorously all of the requirements before design begins and reduces later redesign, recoding, and retesting. Careful review of the requirements in the SRS can reveal omissions, misunderstandings, and inconsistencies early in the development cycle when these problems are easier to correct.

— *Provide a basis for estimating costs and schedules*. The description of the product to be developed as given in the SRS is a realistic basis for estimating project costs and can be used to obtain approval for bids or price estimates.

— *Provide a baseline for validation and verification*. Organizations can develop their validation and verification plans much more productively from a good SRS. As a part of the development contract, the SRS provides a baseline against which compliance can be measured.

— *Facilitate transfer.* The SRS makes it easier to transfer the software product to new users or new machines. Customers thus find it easier to transfer the software to other parts of their organization, and suppliers find it easier to transfer it to new customers.

— *Serve as a basis for enhancement*. Because the SRS discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued production evaluation.

The readers of this document are referred to Annex B for guidelines for using this recommended practice to meet the requirements of IEEE/EIA 12207.1-1997, IEEE/EIA Guide—Industry Implementation of ISO/IEC 12207: 1995, Standard for Information Technology—Software life cycle processes—Life cycle data.

Participants

This recommended practice was prepared by the Life Cycle Data Harmonization Working Group of the Software Engineering Standards Committee of the IEEE Computer Society. At the time this recommended practice was approved, the working group consisted of the following members:

Leonard L. Tripp, *Chair*

Edward Byrne
Paul R. Croll
Perry DeWeese
Robin Fralick
Marilyn Ginsberg-Finner
John Harauz
Mark Henley
Dennis Lawrence
David Maibor
Ray Milovanovic
James Moore
Timothy Niesen
Dennis Rilling
Terry Rout
Richard Schmidt
Norman F. Schneidewind
David Schultz
Basil Sherlund
Peter Voldner
Ronald Wade

The following persons were on the balloting committee:

Syed Ali
Theodore K. Atchinson
Mikhail Auguston
Robert E. Barry
Leo Beltracchi
H. Ronald Berlack
Richard E. Biehl
Michael A. Blackledge
Sandro Bologna
Juris Borzovs
Kathleen L. Briggs
M. Scott Buck
Michael Caldwell
James E. Cardow
Enrico A. Carrara
Lawrence Catchpole
Keith Chan
Antonio M. Cicu
Theo Clarke
Sylvain Clermont
Rosemary Coleman
Virgil Lee Cooper
W. W. Geoff Cozens
Paul R. Croll
Gregory T. Daich
Geoffrey Darnton
Taz Daughtrey
Bostjan K. Derganc
Perry R. DeWeese
James Do
Evelyn S. Dow
Carl Einar Dragstedt
Sherman Eagles
Christof Ebert
Leo Egan
Richard E. Fairley
John W. Fendrich
Jay Forster
Kirby Fortenberry
Eva Freund
Richard C. Fries
Roger U. Fujii
Adel N. Ghannam
Marilyn Ginsberg-Finner
John Garth Glynn
Julio Gonzalez-Sanz
L. M. Gunther
David A. Gustafson
Jon D. Hagar
John Harauz
Robert T. Harley
Herbert Hecht
William Hefley
Manfred Hein
Mark Heinrich
Mark Henley
Debra Herrmann
John W. Horch
Jerry Huller
Peter L. Hung
George Jackelen
Frank V. Jorgensen
William S. Junk
George X. Kambic
Richard Karcich
Ron S. Kenett
Judith S. Kerner
Robert J. Kierzyk
Dwayne L. Knirk
Shaye Koenig
Thomas M. Kurihara
John B. Lane
J. Dennis Lawrence
Fang Ching Lim
William M. Lively
James J. Longbucco
Dieter Look
John Lord
Stan Magee
David Maibor
Harold Mains
Robert A. Martin
Tomoo Matsubara
Mike McAndrew
Patrick D. McCray
Christopher McMacken
Jerome W. Mersky
Bret Michael
Alan Miller
Celia H. Modell
James W. Moore
Pavol Navrat
Myrna L. Olson
Indradeb P. Pal
Alex Polack
Peter T. Poon
Lawrence S. Przybylski
Kenneth R. Ptack
Annette D. Reilly
Dennis Rilling
Andrew P. Sage
Helmut Sandmayr
Stephen R. Schach
Hans Schaefer
Norman Schneidewind
David J. Schultz
Lisa A. Selmon
Robert W. Shillato
David M. Siefert
Carl A. Singer
James M. Sivak
Richard S. Sky
Nancy M. Smith
Melford E. Smyre
Harry M. Sneed
Alfred R. Sorkowitz
Donald W. Sova
Luca Spotorno
Julia Stesney
Fred J. Strauss
Christine Brown Strysik
Toru Takeshita
Richard H. Thayer
Booker Thomas
Patricia Trellue
Theodore J. Urbanowicz
Glenn D. Venables
Udo Voges
David D. Walden
Dolores Wallace
William M. Walsh
John W. Walz
Camille SWhite-Partain
Scott A. Whitmire
P. A. Wolfgang
Paul R. Work
Natalie C. Yopconka
Janusz Zalewski
Geraldine Zimmerman
Peter F. Zoll

When the IEEE-SA Standards Board approved this recommended practice on 25 June 1998, it had the following membership:

Richard J. Holleman, *Chair* **Donald N. Heirman,** *Vice Chair*
Judith Gorman, *Secretary*

Satish K. Aggarwal
Clyde R. Camp
James T. Carlo
Gary R. Engmann
Harold E. Epstein
Jay Forster*
Thomas F. Garrity
Ruben D. Garzon
James H. Gurney
Jim D. Isaak
Lowell G. Johnson
Robert Kennelly
E. G. "Al" Kiener
Joseph L. Koepfinger*
Stephen R. Lambert
Jim Logothetis
Donald C. Loughry
L. Bruce McClung
Louis-François Pau
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Hans E. Weinrich
Donald W. Zipse

*Member Emeritus

Valerie E. Zelenty
IEEE Standards Project Editor

Contents

IEEE Recommended Practice for Software Requirements Specifications

1. Overview

This recommended practice describes recommended approaches for the specification of software requirements. It is divided into five clauses. Clause 1 explains the scope of this recommended practice. Clause 2 lists the references made to other standards. Clause 3 provides definitions of specific terms used. Clause 4 provides background information for writing a good SRS. Clause 5 discusses each of the essential parts of an SRS. This recommended practice also has two annexes, one which provides alternate format templates, and one which provides guidelines for compliance with IEEE/EIA 12207.1-1997.

1.1 Scope

This is a recommended practice for writing software requirements specifications. It describes the content and qualities of a good software requirements specification (SRS) and presents several sample SRS outlines.

This recommended practice is aimed at specifying requirements of software to be developed but also can be applied to assist in the selection of in-house and commercial software products. However, application to already-developed software could be counterproductive.

When software is embedded in some larger system, such as medical equipment, then issues beyond those identified in this recommended practice may have to be addressed.

This recommended practice describes the process of creating a product and the content of the product. The product is an SRS. This recommended practice can be used to create such an SRS directly or can be used as a model for a more specific standard.

This recommended practice does not identify any specific method, nomenclature, or tool for preparing an SRS.

2. References

This recommended practice shall be used in conjunction with the following publications.

ASTM E1340-96, Standard Guide for Rapid Prototyping of Computerized Systems.[1]

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.[2]

IEEE Std 730-1998, IEEE Standard for Software Quality Assurance Plans.

IEEE Std 730.1-1995, IEEE Guide for Software Quality Assurance Planning.

IEEE Std 828-1998, IEEE Standard for Software Configuration Management Plans.[3]

IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software.

IEEE Std 982.2-1988, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software.

IEEE Std 1002-1987 (Reaff 1992), IEEE Standard Taxonomy for Software Engineering Standards.

IEEE Std 1012-1998, IEEE Standard for Software Verification and Validation.

IEEE Std 1012a-1998, IEEE Standard for Software Verification and Validation: Content Map to IEEE/EIA 12207.1-1997.[4]

IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions.[5]

IEEE Std 1028-1997, IEEE Standard for Software Reviews.

IEEE Std 1042-1987 (Reaff 1993), IEEE Guide to Software Configuration Management.

IEEE P1058/D2.1, Draft Standard for Software Project Management Plans, dated 5 August 1998.[6]

IEEE Std 1058a-1998, IEEE Standard for Software Project Management Plans: Content Map to IEEE/EIA 12207.1-1997.[7]

IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes.

IEEE Std 1233, 1998 Edition, IEEE Guide for Developing System Requirements Specifications.[8]

---

[1]ASTM publications are available from the American Society for Testing and Materials, 100 Barr Harbor Drive, West Conshohocken, PA 19428-2959, USA.

[2]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

[3]As this standard goes to press, IEEE Std 828-1998; IEEE Std 1012a-1998; IEEE Std 1016-1998; and IEEE Std 1233, 1998 Edition are approved but not yet published. The draft standards are, however, available from the IEEE. Anticipated publication date is Fall 1998. Contact the IEEE Standards Department at 1 (732) 562-3800 for status information.

[4]See Footnote 3.

[5]See Footnote 3.

[6]Upon approval of IEEE P1058 by the IEEE-SA Standards Board, this standard will be integrated with IEEE Std 1058a-1998 and published as IEEE Std 1058, 1998 Edition. Approval is expected 8 December 1998.

[7]As this standard goes to press, IEEE Std 1058a-1998 is approved but not yet published. The draft standard is, however, available from the IEEE. Anticipated publication date is December 1998. Contact the IEEE Standards Department at 1 (732) 562-3800 for status information. See Footnote 6.

[8]See Footnote 3.

3. Definitions

In general the definitions of terms used in this recommended practice conform to the definitions provided in IEEE Std 610.12-1990. The definitions below are key terms as they are used in this recommended practice.

3.1 contract: A legally binding document agreed upon by the customer and supplier. This includes the technical and organizational requirements, cost, and schedule for a product. A contract may also contain informal but useful information such as the commitments or expectations of the parties involved.

3.2 customer: The person, or persons, who pay for the product and usually (but not necessarily) decide the requirements. In the context of this recommended practice the customer and the supplier may be members of the same organization.

3.3 supplier: The person, or persons, who produce a product for a customer. In the context of this recommended practice, the customer and the supplier may be members of the same organization.

3.4 user: The person, or persons, who operate or interact directly with the product. The user(s) and the customer(s) are often not the same person(s).

4. Considerations for producing a good SRS

This clause provides background information that should be considered when writing an SRS. This includes the following:

a) Nature of the SRS;
b) Environment of the SRS;
c) Characteristics of a good SRS;
d) Joint preparation of the SRS;
e) SRS evolution;
f) Prototyping;
g) Embedding design in the SRS;
h) Embedding project requirements in the SRS.

4.1 Nature of the SRS

The SRS is a specification for a particular software product, program, or set of programs that performs certain functions in a specific environment. The SRS may be written by one or more representatives of the supplier, one or more representatives of the customer, or by both. Subclause 4.4 recommends both.

The basic issues that the SRS writer(s) shall address are the following:

a) *Functionality.* What is the software supposed to do?
b) *External interfaces*. How does the software interact with people, the system's hardware, other hardware, and other software?
c) *Performance*. What is the speed, availability, response time, recovery time of various software functions, etc.?
d) *Attributes*. What are the portability, correctness, maintainability, security, etc. considerations?
e) *Design constraints imposed on an implementation*. Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

The SRS writer(s) should avoid placing either design or project requirements in the SRS.

For recommended contents of an SRS see Clause 5.

4.2 Environment of the SRS

It is important to consider the part that the SRS plays in the total project plan, which is defined in IEEE Std 610.12-1990. The software may contain essentially all the functionality of the project or it may be part of a larger system. In the latter case typically there will be an SRS that will state the interfaces between the system and its software portion, and will place external performance and functionality requirements upon the software portion. Of course the SRS should then agree with and expand upon these system requirements.

IEEE Std 1074-1997 describes the steps in the software life cycle and the applicable inputs for each step. Other standards, such as those listed in Clause 2, relate to other parts of the software life cycle and so may complement software requirements.

Since the SRS has a specific role to play in the software development process, the SRS writer(s) should be careful not to go beyond the bounds of that role. This means the SRS

a) Should correctly define all of the software requirements. A software requirement may exist because of the nature of the task to be solved or because of a special characteristic of the project.
b) Should not describe any design or implementation details. These should be described in the design stage of the project.
c) Should not impose additional constraints on the software. These are properly specified in other documents such as a software quality assurance plan.

Therefore, a properly written SRS limits the range of valid designs, but does not specify any particular design.

4.3 Characteristics of a good SRS

An SRS should be

a) Correct;
b) Unambiguous;
c) Complete;
d) Consistent;
e) Ranked for importance and/or stability;
f) Verifiable;
g) Modifiable;
h) Traceable.

4.3.1 Correct

An SRS is correct if, and only if, every requirement stated therein is one that the software shall meet.

There is no tool or procedure that ensures correctness. The SRS should be compared with any applicable superior specification, such as a system requirements specification, with other project documentation, and with other applicable standards, to ensure that it agrees. Alternatively the customer or user can determine if the SRS correctly reflects the actual needs. Traceability makes this procedure easier and less prone to error (see 4.3.8).

4.3.2 Unambiguous

An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. As a minimum, this requires that each characteristic of the final product be described using a single unique term.

In cases where a term used in a particular context could have multiple meanings, the term should be included in a glossary where its meaning is made more specific.

An SRS is an important part of the requirements process of the software life cycle and is used in design, implementation, project monitoring, verification and validation, and in training as described in IEEE Std 1074-1997. The SRS should be unambiguous both to those who create it and to those who use it. However, these groups often do not have the same background and therefore do not tend to describe software requirements the same way. Representations that improve the requirements specification for the developer may be counterproductive in that they diminish understanding to the user and vice versa.

Subclauses 4.3.2.1 through 4.3.2.3 recommend how to avoid ambiguity.

4.3.2.1 Natural language pitfalls

Requirements are often written in natural language (e.g., English). Natural language is inherently ambiguous. A natural language SRS should be reviewed by an independent party to identify ambiguous use of language so that it can be corrected.

4.3.2.2 Requirements specification languages

One way to avoid the ambiguity inherent in natural language is to write the SRS in a particular requirements specification language. Its language processors automatically detect many lexical, syntactic, and semantic errors.

One disadvantage in the use of such languages is the length of time required to learn them. Also, many nontechnical users find them unintelligible. Moreover, these languages tend to be better at expressing certain types of requirements and addressing certain types of systems. Thus, they may influence the requirements in subtle ways.

4.3.2.3 Representation tools

In general, requirements methods and languages and the tools that support them fall into three general categories—object, process, and behavioral. Object-oriented approaches organize the requirements in terms of real-world objects, their attributes, and the services performed by those objects. Process-based approaches organize the requirements into hierarchies of functions that communicate via data flows. Behavioral approaches describe external behavior of the system in terms of some abstract notion (such as predicate calculus), mathematical functions, or state machines.

The degree to which such tools and methods may be useful in preparing an SRS depends upon the size and complexity of the program. No attempt is made here to describe or endorse any particular tool.

When using any of these approaches it is best to retain the natural language descriptions. That way, customers unfamiliar with the notations can still understand the SRS.

4.3.3 Complete

An SRS is complete if, and only if, it includes the following elements:

a) All significant requirements, whether relating to functionality, performance, design constraints, attributes, or external interfaces. In particular any external requirements imposed by a system specification should be acknowledged and treated.

b) Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations. Note that it is important to specify the responses to both valid and invalid input values.

c) Full labels and references to all figures, tables, and diagrams in the SRS and definition of all terms and units of measure.

4.3.3.1 Use of TBDs

Any SRS that uses the phrase "to be determined" (TBD) is not a complete SRS. The TBD is, however, occasionally necessary and should be accompanied by

a) A description of the conditions causing the TBD (e.g., why an answer is not known) so that the situation can be resolved;

b) A description of what must be done to eliminate the TBD, who is responsible for its elimination, and by when it must be eliminated.

4.3.4 Consistent

Consistency refers to internal consistency. If an SRS does not agree with some higher-level document, such as a system requirements specification, then it is not correct (see 4.3.1).

4.3.4.1 Internal consistency

An SRS is internally consistent if, and only if, no subset of individual requirements described in it conflict. The three types of likely conflicts in an SRS are as follows:

a) The specified characteristics of real-world objects may conflict. For example,
   1) The format of an output report may be described in one requirement as tabular but in another as textual.
   2) One requirement may state that all lights shall be green while another may state that all lights shall be blue.

b) There may be logical or temporal conflict between two specified actions. For example,
   1) One requirement may specify that the program will add two inputs and another may specify that the program will multiply them.
   2) One requirement may state that "A" must always follow "B," while another may require that "A and B" occur simultaneously.

c) Two or more requirements may describe the same real-world object but use different terms for that object. For example, a program's request for a user input may be called a "prompt" in one requirement and a "cue" in another. The use of standard terminology and definitions promotes consistency.

4.3.5 Ranked for importance and/or stability

An SRS is ranked for importance and/or stability if each requirement in it has an identifier to indicate either the importance or stability of that particular requirement.

Typically, all of the requirements that relate to a software product are not equally important. Some requirements may be essential, especially for life-critical applications, while others may be desirable.

Each requirement in the SRS should be identified to make these differences clear and explicit. Identifying the requirements in the following manner helps:

a) Have customers give more careful consideration to each requirement, which often clarifies any hidden assumptions they may have.
b) Have developers make correct design decisions and devote appropriate levels of effort to the different parts of the software product.

4.3.5.1 Degree of stability

One method of identifying requirements uses the dimension of stability. Stability can be expressed in terms of the number of expected changes to any requirement based on experience or knowledge of forthcoming events that affect the organization, functions, and people supported by the software system.

4.3.5.2 Degree of necessity

Another way to rank requirements is to distinguish classes of requirements as essential, conditional, and optional.

a) *Essential*. Implies that the software will not be acceptable unless these requirements are provided in an agreed manner.
b) *Conditional*. Implies that these are requirements that would enhance the software product, but would not make it unacceptable if they are absent.
c) *Optional*. Implies a class of functions that may or may not be worthwhile. This gives the supplier the opportunity to propose something that exceeds the SRS.

4.3.6 Verifiable

An SRS is verifiable if, and only if, every requirement stated therein is verifiable. A requirement is verifiable if, and only if, there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement. In general any ambiguous requirement is not verifiable.

Nonverifiable requirements include statements such as “works well,” “good human interface,” and “shall usually happen.” These requirements cannot be verified because it is impossible to define the terms “good,” “well,” or “usually.” The statement that “the program shall never enter an infinite loop” is nonverifiable because the testing of this quality is theoretically impossible.

An example of a verifiable statement is

> *Output of the program shall be produced within 20 s of event × 60% of the time; and shall be produced within 30 s of event × 100% of the time.*

This statement can be verified because it uses concrete terms and measurable quantities.

If a method cannot be devised to determine whether the software meets a particular requirement, then that requirement should be removed or revised.

4.3.7 Modifiable

An SRS is modifiable if, and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style. Modifiability generally requires an SRS to

a) Have a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing;
b) Not be redundant (i.e., the same requirement should not appear in more than one place in the SRS);
c) Express each requirement separately, rather than intermixed with other requirements.

Redundancy itself is not an error, but it can easily lead to errors. Redundancy can occasionally help to make an SRS more readable, but a problem can arise when the redundant document is updated. For instance, a requirement may be altered in only one of the places where it appears. The SRS then becomes inconsistent. Whenever redundancy is necessary, the SRS should include explicit cross-references to make it modifiable.

4.3.8 Traceable

An SRS is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation. The following two types of traceability are recommended:

a) *Backward traceability (i.e., to previous stages of development)*. This depends upon each requirement explicitly referencing its source in earlier documents.
b) *Forward traceability (i.e., to all documents spawned by the SRS)*. This depends upon each requirement in the SRS having a unique name or reference number.

The forward traceability of the SRS is especially important when the software product enters the operation and maintenance phase. As code and design documents are modified, it is essential to be able to ascertain the complete set of requirements that may be affected by those modifications.

4.4 Joint preparation of the SRS

The software development process should begin with supplier and customer agreement on what the completed software must do. This agreement, in the form of an SRS, should be jointly prepared. This is important because usually neither the customer nor the supplier is qualified to write a good SRS alone.

a) Customers usually do not understand the software design and development process well enough to write a usable SRS.
b) Suppliers usually do not understand the customer's problem and field of endeavor well enough to specify requirements for a satisfactory system.

Therefore, the customer and the supplier should work together to produce a well-written and completely understood SRS.

A special situation exists when a system and its software are both being defined concurrently. Then the functionality, interfaces, performance, and other attributes and constraints of the software are not predefined, but rather are jointly defined and subject to negotiation and change. This makes it more difficult, but no less important, to meet the characteristics stated in 4.3. In particular, an SRS that does not comply with the requirements of its parent system specification is incorrect.

This recommended practice does not specifically discuss style, language usage, or techniques of good writing. It is quite important, however, that an SRS be well written. General technical writing books can be used for guidance.

4.5 SRS evolution

The SRS may need to evolve as the development of the software product progresses. It may be impossible to specify some details at the time the project is initiated (e.g., it may be impossible to define all of the screen formats for an interactive program during the requirements phase). Additional changes may ensue as deficiencies, shortcomings, and inaccuracies are discovered in the SRS.

Two major considerations in this process are the following:

a) Requirements should be specified as completely and thoroughly as is known at the time, even if evolutionary revisions can be foreseen as inevitable. The fact that they are incomplete should be noted.

b) A formal change process should be initiated to identify, control, track, and report projected changes. Approved changes in requirements should be incorporated in the SRS in such a way as to

   1) Provide an accurate and complete audit trail of changes;
   2) Permit the review of current and superseded portions of the SRS.

4.6 Prototyping

Prototyping is used frequently during the requirements portion of a project. Many tools exist that allow a prototype, exhibiting some characteristics of a system, to be created very quickly and easily. See also ASTM E1340-96.

Prototypes are useful for the following reasons:

a) The customer may be more likely to view the prototype and react to it than to read the SRS and react to it. Thus, the prototype provides quick feedback.

b) The prototype displays unanticipated aspects of the systems behavior. Thus, it produces not only answers but also new questions. This helps reach closure on the SRS.

c) An SRS based on a prototype tends to undergo less change during development, thus shortening development time.

A prototype should be used as a way to elicit software requirements. Some characteristics such as screen or report formats can be extracted directly from the prototype. Other requirements can be inferred by running experiments with the prototype.

4.7 Embedding design in the SRS

A requirement specifies an externally visible function or attribute of a system. A design describes a particular subcomponent of a system and/or its interfaces with other subcomponents. The SRS writer(s) should clearly distinguish between identifying required design constraints and projecting a specific design. Note that every requirement in the SRS limits design alternatives. This does not mean, though, that every requirement is design.

The SRS should specify what functions are to be performed on what data to produce what results at what location for whom. The SRS should focus on the services to be performed. The SRS should not normally specify design items such as the following:

a) Partitioning the software into modules;
b) Allocating functions to the modules;
c) Describing the flow of information or control between modules;
d) Choosing data structures.

4.7.1 Necessary design requirements

In special cases some requirements may severely restrict the design. For example, security or safety requirements may reflect directly into design such as the need to

a) Keep certain functions in separate modules;
b) Permit only limited communication between some areas of the program;
c) Check data integrity for critical variables.

Examples of valid design constraints are physical requirements, performance requirements, software development standards, and software quality assurance standards.

Therefore, the requirements should be stated from a purely external viewpoint. When using models to illustrate the requirements, remember that the model only indicates the external behavior, and does not specify a design.

4.8 Embedding project requirements in the SRS

The SRS should address the software product, not the process of producing the software product.

Project requirements represent an understanding between the customer and the supplier about contractual matters pertaining to production of software and thus should not be included in the SRS. These normally include items such as

a) Cost;
b) Delivery schedules;
c) Reporting procedures;
d) Software development methods;
e) Quality assurance;
f) Validation and verification criteria;
g) Acceptance procedures.

Project requirements are specified in other documents, typically in a software development plan, a software quality assurance plan, or a statement of work.

5. The parts of an SRS

This clause discusses each of the essential parts of the SRS. These parts are arranged in Figure 1 in an outline that can serve as an example for writing an SRS.

While an SRS does not have to follow this outline or use the names given here for its parts, a good SRS should include all the information discussed here.

Table of Contents

Figure 1—Prototype SRS outline

5.1 Introduction (Section 1 of the SRS)

The introduction of the SRS should provide an overview of the entire SRS. It should contain the following subsections:

a) Purpose;
b) Scope;
c) Definitions, acronyms, and abbreviations;
d) References;
e) Overview.

5.1.1 Purpose (1.1 of the SRS)

This subsection should

a) Delineate the purpose of the SRS;
b) Specify the intended audience for the SRS.

5.1.2 Scope (1.2 of the SRS)

This subsection should

a) Identify the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);
b) Explain what the software product(s) will, and, if necessary, will not do;
c) Describe the application of the software being specified, including relevant benefits, objectives, and goals;
d) Be consistent with similar statements in higher-level specifications (e.g., the system requirements specification), if they exist.

5.1.3 Definitions, acronyms, and abbreviations (1.3 of the SRS)

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

5.1.4 References (1.4 of the SRS)

This subsection should

a) Provide a complete list of all documents referenced elsewhere in the SRS;
b) Identify each document by title, report number (if applicable), date, and publishing organization;
c) Specify the sources from which the references can be obtained.

This information may be provided by reference to an appendix or to another document.

5.1.5 Overview (1.5 of the SRS)

This subsection should

a) Describe what the rest of the SRS contains;
b) Explain how the SRS is organized.

5.2 Overall description (Section 2 of the SRS)

This section of the SRS should describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 3 of the SRS, and makes them easier to understand.

This section usually consists of six subsections, as follows:

a) Product perspective;
b) Product functions;
c) User characteristics;
d) Constraints;
e) Assumptions and dependencies;
f) Apportioning of requirements.

5.2.1 Product perspective (2.1 of the SRS)

This subsection of the SRS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful.

This subsection should also describe how the software operates inside various constraints. For example, these constraints could include

a) System interfaces;
b) User interfaces;
c) Hardware interfaces;
d) Software interfaces;
e) Communications interfaces;
f) Memory;
g) Operations;
h) Site adaptation requirements.

5.2.1.1 System interfaces

This should list each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.

5.2.1.2 User interfaces

This should specify the following:

a) *The logical characteristics of each interface between the software product and its users*. This includes those configuration characteristics (e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys) necessary to accomplish the software requirements.

b) *All the aspects of optimizing the interface with the person who must use the system*. This may simply comprise a list of do's and don'ts on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. Like all others, these requirements should be verifiable, e.g., "a clerk typist grade 4 can do function *X* in *Z* min after 1 h of training" rather than "a typist can do function *X*." (This may also be specified in the Software System Attributes under a section titled Ease of Use.)

5.2.1.3 Hardware interfaces

This should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.). It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. For example, terminal support may specify full-screen support as opposed to line-by-line support.

5.2.1.4 Software interfaces

This should specify the use of other required software products (e.g., a data management system, an operating system, or a mathematical package), and interfaces with other application systems (e.g., the linkage between an accounts receivable system and a general ledger system). For each required software product, the following should be provided:

— Name;
— Mnemonic;
— Specification number;
— Version number;
— Source.

For each interface, the following should be provided:

— Discussion of the purpose of the interfacing software as related to this software product.
— Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

5.2.1.5 Communications interfaces

This should specify the various interfaces to communications such as local network protocols, etc.

5.2.1.6 Memory constraints

This should specify any applicable characteristics and limits on primary and secondary memory.

5.2.1.7 Operations

This should specify the normal and special operations required by the user such as

a) The various modes of operations in the user organization (e.g., user-initiated operations);
b) Periods of interactive operations and periods of unattended operations;
c) Data processing support functions;
d) Backup and recovery operations.

NOTE—This is sometimes specified as part of the User Interfaces section.

5.2.1.8 Site adaptation requirements

This should

a) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode (e.g., grid values, safety limits, etc.);
b) Specify the site or mission-related features that should be modified to adapt the software to a particular installation.

5.2.2 Product functions (2.2 of the SRS)

This subsection of the SRS should provide a summary of the major functions that the software will perform. For example, an SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires.

Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product. Note that for the sake of clarity

a) The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time.
b) Textual or graphical methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product, but simply shows the logical relationships among variables.

5.2.3 User characteristics (2.3 of the SRS)

This subsection of the SRS should describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. It should not be used to state specific requirements, but rather should provide the reasons why certain specific requirements are later specified in Section 3 of the SRS.

5.2.4 Constraints (2.4 of the SRS)

This subsection of the SRS should provide a general description of any other items that will limit the developer's options. These include

a) Regulatory policies;
b) Hardware limitations (e.g., signal timing requirements);
c) Interfaces to other applications;
d) Parallel operation;
e) Audit functions;
f) Control functions;
g) Higher-order language requirements;
h) Signal handshake protocols (e.g., XON-XOFF, ACK-NACK);
i) Reliability requirements;
j) Criticality of the application;
k) Safety and security considerations.

5.2.5 Assumptions and dependencies (2.5 of the SRS)

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

5.2.6 Apportioning of requirements (2.6 of the SRS)

This subsection of the SRS should identify requirements that may be delayed until future versions of the system.

5.3 Specific requirements (Section 3 of the SRS)

This section of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input or in support of an output. As this is often the largest and most important part of the SRS, the following principles apply:

a) Specific requirements should be stated in conformance with all the characteristics described in 4.3.
b) Specific requirements should be cross-referenced to earlier documents that relate.
c) All requirements should be uniquely identifiable.
d) Careful attention should be given to organizing the requirements to maximize readability.

Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in 5.3.1 through 5.3.7.

5.3.1 External interfaces

This should be a detailed description of all inputs into and outputs from the software system. It should complement the interface descriptions in 5.2 and should not repeat information there.

It should include both content and format as follows:

a) Name of item;
b) Description of purpose;
c) Source of input or destination of output;
d) Valid range, accuracy, and/or tolerance;
e) Units of measure;
f) Timing;
g) Relationships to other inputs/outputs;
h) Screen formats/organization;
i) Window formats/organization;
j) Data formats;
k) Command formats;
l) End messages.

5.3.2 Functions

Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as “shall” statements starting with “The system shall…”

These include

a) Validity checks on the inputs
b) Exact sequence of operations
c) Responses to abnormal situations, including
    1) Overflow
    2) Communication facilities
    3) Error handling and recovery
d) Effect of parameters
e) Relationship of outputs to inputs, including
    1) Input/output sequences
    2) Formulas for input to output conversion

It may be appropriate to partition the functional requirements into subfunctions or subprocesses. This does not imply that the software design will also be partitioned that way.

5.3.3 Performance requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following:

a) The number of terminals to be supported;
b) The number of simultaneous users to be supported;
c) Amount and type of information to be handled.

Static numerical requirements are sometimes identified under a separate section entitled Capacity.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms.

For example,

95% of the transactions shall be processed in less than 1 s.

rather than,

An operator shall not have to wait for the transaction to complete.

NOTE—Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.

5.3.4 Logical database requirements

This should specify the logical requirements for any information that is to be placed into a database. This may include the following:

a) Types of information used by various functions;
b) Frequency of use;
c) Accessing capabilities;
d) Data entities and their relationships;
e) Integrity constraints;
f) Data retention requirements.

5.3.5 Design constraints

This should specify design constraints that can be imposed by other standards, hardware limitations, etc.

5.3.5.1 Standards compliance

This subsection should specify the requirements derived from existing standards or regulations. They may include the following:

a) Report format;
b) Data naming;
c) Accounting procedures;
d) Audit tracing.

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

5.3.6 Software system attributes

There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. Subclauses 5.3.6.1 through 5.3.6.5 provide a partial list of examples.

5.3.6.1 Reliability

This should specify the factors required to establish the required reliability of the software system at time of delivery.

5.3.6.2 Availability

This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

5.3.6.3 Security

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to

a) Utilize certain cryptographical techniques;
b) Keep specific log or history data sets;
c) Assign certain functions to different modules;
d) Restrict communications between some areas of the program;
e) Check data integrity for critical variables.

5.3.6.4 Maintainability

This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.

5.3.6.5 Portability

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

a) Percentage of components with host-dependent code;
b) Percentage of code that is host dependent;
c) Use of a proven portable language;
d) Use of a particular compiler or language subset;
e) Use of a particular operating system.

5.3.7 Organizing the specific requirements

For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in Section 3 of the SRS. Some of these organizations are described in 5.3.7.1 through 5.3.7.7.

5.3.7.1 System mode

Some systems behave quite differently depending on the mode of operation. For example, a control system may have different sets of functions depending on its mode: training, normal, or emergency. When organizing this section by mode, the outline in A.1 or A.2 should be used. The choice depends on whether interfaces and performance are dependent on mode.

5.3.7.2 User class

Some systems provide different sets of functions to different classes of users. For example, an elevator control system presents different capabilities to passengers, maintenance workers, and fire fighters. When organizing this section by user class, the outline in A.3 should be used.

5.3.7.3 Objects

Objects are real-world entities that have a counterpart within the system. For example, in a patient monitoring system, objects include patients, sensors, nurses, rooms, physicians, medicines, etc. Associated with each object is a set of attributes (of that object) and functions (performed by that object). These functions are also called services, methods, or processes. When organizing this section by object, the outline in A.4 should be used. Note that sets of objects may share attributes and services. These are grouped together as classes.

5.3.7.4 Feature

A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. For example, in a telephone system, features include local call, call forwarding, and conference call. Each feature is generally described in a sequence of stimulus-response pairs. When organizing this section by feature, the outline in A.5 should be used.

5.3.7.5 Stimulus

Some systems can be best organized by describing their functions in terms of stimuli. For example, the functions of an automatic aircraft landing system may be organized into sections for loss of power, wind shear, sudden change in roll, vertical velocity excessive, etc. When organizing this section by stimulus, the outline in A.6 should be used.

5.3.7.6 Response

Some systems can be best organized by describing all the functions in support of the generation of a response. For example, the functions of a personnel system may be organized into sections corresponding to all functions associated with generating paychecks, all functions associated with generating a current list of employees, etc. The outline in A.6 (with all occurrences of stimulus replaced with response) should be used.

5.3.7.7 Functional hierarchy

When none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data. When organizing this section by functional hierarchy, the outline in A.7 should be used.

5.3.8 Additional comments

Whenever a new SRS is contemplated, more than one of the organizational techniques given in 5.3.7.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification. For example, see A.8 for an organization combining user class and feature. Any additional requirements may be put in a separate section at the end of the SRS.

There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented

analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; and when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.

In any of the outlines given in A.1 through A.8, those sections called "Functional Requirement *i*" may be described in native language (e.g., English), in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, and Outputs.

5.4 Supporting information

The supporting information makes the SRS easier to use. It includes the following:

a) Table of contents;
b) Index;
c) Appendixes.

5.4.1 Table of contents and index

The table of contents and index are quite important and should follow general compositional practices.

5.4.2 Appendixes

The appendixes are not always considered part of the actual SRS and are not always necessary. They may include

a) Sample input/output formats, descriptions of cost analysis studies, or results of user surveys;
b) Supporting or background information that can help the readers of the SRS;
c) A description of the problems to be solved by the software;
d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements.

When appendixes are included, the SRS should explicitly state whether or not the appendixes are to be considered part of the requirements.

Annex A

(informative)

SRS templates

A.1 Template of SRS Section 3 organized by mode: Version 1

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
    .
    .
    .
    3.2.1.*n* Functional requirement 1.*n*
  3.2.2 Mode 2
  .
  .
  .
  3.2.*m* Mode *m*
    3.2.*m*.1 Functional requirement *m*.1
    .
    .
    .
    3.2.*m*.*n* Functional requirement *m*.*n*
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

A.2 Template of SRS Section 3 organized by mode: Version 2

3. Specific requirements
3.1. Functional requirements
  3.1.1 Mode 1
    3.1.1.1 External interfaces
      3.1.1.1.1 User interfaces
      3.1.1.1.2 Hardware interfaces
      3.1.1.1.3 Software interfaces
      3.1.1.1.4 Communications interfaces
    3.1.1.2 Functional requirements
      3.1.1.2.1 Functional requirement 1
      .
      .
      .

3.1.1.2.*n* Functional requirement *n*
3.1.1.3 Performance
3.1.2 Mode 2
.
.
.
3.1.*m* Mode *m*
3.2 Design constraints
3.3 Software system attributes
3.4 Other requirements

A.3 Template of SRS Section 3 organized by user class

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 User class 1
3.2.1.1 Functional requirement 1.1
.
.
.
3.2.1.*n* Functional requirement 1.*n*
3.2.2 User class 2
.
.
.
3.2.*m* User class *m*
3.2.*m*.1 Functional requirement *m*.1
.
.
.
3.2.*m*.*n* Functional requirement *m*.*n*
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

A.4 Template of SRS Section 3 organized by object

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 Classes/Objects
3.2.1 Class/Object 1

3.2.1.1 Attributes (direct or inherited)
3.2.1.1.1 Attribute 1
.
.
.
3.2.1.1.*n* Attribute *n*
3.2.1.2 Functions (services, methods, direct or inherited)
3.2.1.2.1 Functional requirement 1.1
.
.
.
3.2.1.2.*m* Functional requirement 1.*m*
3.2.1.3 Messages (communications received or sent)
3.2.2 Class/Object 2
.
.
.
3.2.*p* Class/Object *p*
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

A.5 Template of SRS Section 3 organized by feature

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 System features
3.2.1 System Feature 1
3.2.1.1 Introduction/Purpose of feature
3.2.1.2 Stimulus/Response sequence
3.2.1.3 Associated functional requirements
3.2.1.3.1 Functional requirement 1
.
.
.
3.2.1.3.*n* Functional requirement *n*
3.2.2 System feature 2
.
.
.
3.2.*m* System feature *m*
.
.
.
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

A.6 Template of SRS Section 3 organized by stimulus

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 Stimulus 1
    3.2.1.1 Functional requirement 1.1
    .
    .
    .
    3.2.1.*n* Functional requirement 1.*n*
  3.2.2 Stimulus 2
  .
  .
  .
  3.2.*m* Stimulus m
    3.2.*m*.1 Functional requirement *m*.1
    .
    .
    .
    3.2.*m*.*n* Functional requirement *m*.*n*
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

A.7 Template of SRS Section 3 organized by functional hierarchy

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 Information flows
    3.2.1.1 Data flow diagram 1
      3.2.1.1.1 Data entities
      3.2.1.1.2 Pertinent processes
      3.2.1.1.3 Topology
    3.2.1.2 Data flow diagram 2
      3.2.1.2.1 Data entities
      3.2.1.2.2 Pertinent processes
      3.2.1.2.3 Topology
    .
    .
    .
    3.2.1.*n* Data flow diagram *n*

3.2.1.*n*.1 Data entities
3.2.1.*n*.2 Pertinent processes
3.2.1.*n*.3 Topology

3.2.2 Process descriptions
3.2.2.1 Process 1
3.2.2.1.1 Input data entities
3.2.2.1.2 Algorithm or formula of process
3.2.2.1.3 Affected data entities
3.2.2.2 Process 2
3.2.2.2.1 Input data entities
3.2.2.2.2 Algorithm or formula of process
3.2.2.2.3 Affected data entities
.
.
.
3.2.2.*m* Process *m*
3.2.2.*m*.1 Input data entities
3.2.2.*m*.2 Algorithm or formula of process
3.2.2.*m*.3 Affected data entities

3.2.3 Data construct specifications
3.2.3.1 Construct 1
3.2.3.1.1 Record type
3.2.3.1.2 Constituent fields
3.2.3.2 Construct 2
3.2.3.2.1 Record type
3.2.3.2.2 Constituent fields
.
.
.
3.2.3.*p* Construct *p*
3.2.3.*p*.1 Record type
3.2.3.*p*.2 Constituent fields

3.2.4 Data dictionary
3.2.4.1 Data element 1
3.2.4.1.1 Name
3.2.4.1.2 Representation
3.2.4.1.3 Units/Format
3.2.4.1.4 Precision/Accuracy
3.2.4.1.5 Range
3.2.4.2 Data element 2
3.2.4.2.1 Name
3.2.4.2.2 Representation
3.2.4.2.3 Units/Format
3.2.4.2.4 Precision/Accuracy
3.2.4.2.5 Range
.
.
.
3.2.4.*q* Data element *q*
3.2.4.*q*.1 Name
3.2.4.*q*.2 Representation
3.2.4.*q*.3 Units/Format
3.2.4.*q*.4 Precision/Accuracy
3.2.4.*q*.5 Range

3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

A.8 Template of SRS Section 3 showing multiple organizations

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 User class 1
    3.2.1.1 Feature 1.1
      3.2.1.1.1 Introduction/Purpose of feature
      3.2.1.1.2 Stimulus/Response sequence
      3.2.1.1.3 Associated functional requirements
    3.2.1.2 Feature 1.2
      3.2.1.2.1 Introduction/Purpose of feature
      3.2.1.2.2 Stimulus/Response sequence
      3.2.1.2.3 Associated functional requirements
    .
    .
    .
    3.2.1.*m* Feature 1.*m*
      3.2.1.*m*.1 Introduction/Purpose of feature
      3.2.1.*m*.2 Stimulus/Response sequence
      3.2.1.*m*.3 Associated functional requirements
  3.2.2 User class 2
  .
  .
  .
  3.2.*n* User class *n*
  .
  .
  .
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

Annex B

(informative)

Guidelines for compliance with IEEE/EIA 12207.1-1997

B.1 Overview

The Software Engineering Standards Committee (SESC) of the IEEE Computer Society has endorsed the policy of adopting international standards. In 1995, the international standard, ISO/IEC 12207, Information technology—Software life cycle processes, was completed. The standard establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry.

In 1995 the SESC evaluated ISO/IEC 12207 and decided that the standard should be adopted and serve as the basis for life cycle processes within the IEEE Software Engineering Collection. The IEEE adaptation of ISO/IEC 12207 is IEEE/EIA 12207.0-1996. It contains ISO/IEC 12207 and the following additions: improved compliance approach, life cycle process objectives, life cycle data objectives, and errata.

The implementation of ISO/IEC 12207 within the IEEE also includes the following:

— IEEE/EIA 12207.1-1997, IEEE/EIA Guide for Information Technology—Software life cycle processes—Life cycle data;

— IEEE/EIA 12207.2-1997, IEEE/EIA Guide for Information Technology—Software life cycle processes—Implementation considerations; and

— Additions to 11 SESC standards (i.e., IEEE Stds 730, 828, 829, 830, 1012, 1016, 1058, 1062, 1219, 1233, 1362) to define the correlation between the data produced by existing SESC standards and the data produced by the application of IEEE/EIA 12207.1-1997.

NOTE—Although IEEE/EIA 12207.1-1997 is a guide, it also contains provisions for application as a standard with specific compliance requirements. This annex treats 12207.1-1997 as a standard.

B.1.1 Scope and purpose

Both IEEE Std 830-1998 and IEEE/EIA 12207.1-1997 place requirements on a Software Requirements Description Document. The purpose of this annex is to explain the relationship between the two sets of requirements so that users producing documents intended to comply with both standards may do so.

B.2 Correlation

This clause explains the relationship between IEEE Std 830-1998 and IEEE/EIA 12207.0-1996 and IEEE/EIA 12207.1-1997 in the following areas: terminology, process, and life cycle data.

B.2.1 Terminology correlation

Both this recommended practice and IEEE/EIA 12207.0-1996 have similar semantics for the key terms of software, requirements, specification, supplier, developer, and maintainer. This recommended practice uses

the term “customer” where IEEE/EIA 12207.0-1996 uses “acquirer,” and this recommended practice uses “user” where IEEE/EIA 12207.0-1996 uses “operator.”

B.2.2 Process correlation

IEEE/EIA 12207.0-1996 uses a process-oriented approach for describing the definition of a set of requirements for software. This recommended practice uses a product-oriented approach, where the product is a Software Requirements Description (SRD). There are natural process steps, namely the steps to create each portion of the SRD. These may be correlated with the process requirements of IEEE/EIA 12207.0-1996. The difference is that this recommended practice is focused on the development of software requirements whereas IEEE/EIA 12207.0-1996 provides an overall life cycle view and mentions Software Requirements Analysis as part of its Development Process. This recommended practice provides a greater level of detail on what is involved in the preparation of an SRD.

B.2.3 Life cycle data correlation

IEEE/EIA 12207.0-1996 takes the viewpoint that the software requirements are derived from the system requirements. Therefore, it uses the term, “description” rather that “specification” to describe the software requirements. In a system in which software is a component, each requiring its own specification, there would be a System Requirements Specification (SRS) and one or more SRDs. If the term Software Requirements Specification had been used, there would be a confusion between an SRS referring to the system or software requirements. In the case where there is a stand-alone software system, IEEE/EIA 12207.1-1997 states “If the software is a stand-alone system, then this document should be a specification.”

B.3 Content mapping

This clause provides details bearing on a claim that an SRS complying with this recommended practice would also achieve “document compliance” with the SRD described in IEEE/EIA 12207.1-1997. The requirements for document compliance are summarized in a single row of Table 1 of IEEE/EIA 12207.1-1997. That row is reproduced in Table B.1 of this recommended practice.

Table B.1—Summary of requirements for an SRD excerpted from Table 1 of IEEE/EIA 12207.1-1997

Information item	IEEE/EIA 12207.0-1996 Clause	Kind	IEEE/EIA 12207.1-1997 Clause	References
Software Requirements Description	5.1.1.4, 5.3.4.1, 5.3.4.2	Description (See note for 6.22.1 of IEEE/EIA 12207.1-1997.)	6.22	IEEE Std 830-1998; EIA/IEEE J-STD-016, F.2.3, F.2.4; MIL-STD 961D. Also see ISO/IEC 5806, 5807, 6593, 8631, 8790, and 11411 for guidance on use of notations.

The requirements for document compliance are discussed in the following subclauses:

— B.3.1 discusses compliance with the information requirements noted in column 2 of Table B.1 as prescribed by 5.1.1.4, 5.3.4.1, and 5.3.4.2 of IEEE/EIA 12207.0-1996.

— B.3.2 discusses compliance with the generic content guideline (the “kind” of document) noted in column 3 of Table B.1 as a “description”. The generic content guidelines for a “description” appear in 5.1 of IEEE/EIA 12207.1-1997.

— B.3.3 discusses compliance with the specific requirements for a Software Requirements Description noted in column 4 of Table B.1 as prescribed by 6.22 of IEEE/EIA 12207.1-1997.

— B.3.4 discusses compliance with the life cycle data objectives of Annex H of IEEE/EIA 12207.0-1996 as described in 4.2 of IEEE/EIA 12207.1-1997.

B.3.1 Compliance with information requirements of IEEE/EIA 12207.0-1996

The information requirements for an SRD are those prescribed by 5.1.1.4, 5.3.4.1, and 5.3.4.2 of IEEE/EIA 12207.0-1996. The requirements are substantively identical to those considered in B.3.3 of this recommended practice.

B.3.2 Compliance with generic content guidelines of IEEE/EIA 12207.1-1997

According to IEEE/EIA 12207.1-1997, the generic content guideline for an SRD is generally a description, as prescribed by 5.1 of IEEE/EIA 12207.1-1997. A complying description shall achieve the purpose stated in 5.1.1 and include the information listed in 5.1.2 of IEEE/EIA 12207.1-1997.

The purpose of a description is:

> IEEE/EIA 12207.1-1997, subclause 5.1.1: Purpose: Describe a planned or actual function, design, performance, or process.

An SRD complying with this recommended practice would achieve the stated purpose.

Any description or specification complying with IEEE/EIA 12207.1-1997 shall satisfy the generic content requirements provided in 5.1.2 of that standard. Table B.2 of this recommended practice lists the generic content items and, where appropriate, references the clause of this recommended practice that requires the same information.

Table B.2—Coverage of generic description requirements by IEEE Std 830-1998

IEEE/EIA 12207.1-1997 generic content	Corresponding clauses of IEEE Std 830-1998	Additions to requirements of IEEE Std 830-1998
a) Date of issue and status	—	Date of issue and status shall be provided.
b) Scope	5.1.1 Scope	—
c) Issuing organization	—	Issuing organization shall be identified.
d) References	5.1.4 References	—
e) Context	5.1.2 Scope	—
f) Notation for description	4.3 Characteristics of a good SRS	—
g) Body	5. The parts of an SRS	—
h) Summary	5.1.1. Overview	—
i) Glossary	5.1.3 Definitions	—
j) Change history	—	Change history for the SRD shall be provided or referenced.

B.3.3 Compliance with specific content requirements of IEEE/EIA 12207.1-1997

The specific content requirements for an SRD in IEEE/EIA 12207.1-1997 are prescribed by 6.22 of IEEE/EIA 12207.1-1997. A compliant SRD shall achieve the purpose stated in 6.22.1 of IEEE/EIA 12207.1-1997.

The purpose of the SRD is:

> IEEE/EIA 12207.1-1997, subclause 6.22.1: Purpose: Specify the requirements for a software item and the methods to be used to ensure that each requirement has been met. Used as the basis for design and qualification testing of a software item.

An SRS complying with this recommended practice and meeting the additional requirements of Table B.3 of this recommended practice would achieve the stated purpose.

An SRD compliant with IEEE/EIA 12207.1-1997 shall satisfy the specific content requirements provided in 6.22.3 and 6.22.4 of that standard. Table B.3 of this recommended practice lists the specific content items and, where appropriate, references the clause of this recommended practice that requires the same information.

An SRD specified according the requirements stated or referenced in Table B.3 of this recommended practice shall be evaluated considering the criteria provided in 5.3.4.2 of IEEE/EIA 12207.0-1996.

Table B.3—Coverage of specific SRD requirements by IEEE Std 830-1998

IEEE/EIA 12207.1-1997 specific content	Corresponding clauses of IEEE Std 830-1998	Additions to requirements of IEEE Std 830-1998
a) Generic description information	See Table B.2	—
b) System identification and overview	5.1.1 Scope	—
c) Functionality of the software item including: – Performance requirements – Physical characteristics – Environmental conditions	5.3.2 Functions 5.3.3 Performance requirements	Physical characteristics and environmental conditions should be provided.
d) Requirements for interfaces external to software item	5.3.1 External interfaces	—
e) Qualification requirements	—	The requirements to be used for qualification testing should be provided (or referenced).
f) Safety specifications	5.2.4 Constraints	—
g) Security and privacy specifications	5.3.6.3 Security	—
h) Human-factors engineering requirements	5.2.3 User characteristics 5.2.1.2 User interfaces	—
i) Data definition and database requirements	5.3.4 Logical data base requirements	—
j) Installation and acceptance requirements at operation site	5.2.1.8 Site adaptation requirements	Installation and acceptance requirements at operation site
k) Installation and acceptance requirements at maintenance site	—	Installation and acceptance requirements at maintenance site
l) User documentation requirements	—	User documentation requirements
m) User operation and execution requirements	5.2.1.7 Operations	User execution requirements

Table B.3—Coverage of specific SRD requirements by IEEE Std 830-1998 (continued)

IEEE/EIA 12207.1-1997 specific content	Corresponding clauses of IEEE Std 830-1998	Additions to requirements of IEEE Std 830-1998
n) User maintenance requirements	5.3.6.4 Maintainability	—
o) Software quality characteristics	5.3.6 Software system attributes	—
p) Design and implementation constraints	5.2.4 Constraints	—
q) Computer resource requirements	5.3.3 Performance requirements	Computer resource requirements
r) Packaging requirements	—	Packaging requirements
s) Precedence and criticality of requirements	5.2.6 Apportioning of requirements	—
t) Requirements traceability	4.3.8 Traceable	—
u) Rationale	5.2.5 Assumptions and dependencies	—
Items a) through f) below are from 6.22.4 a) Support the life cycle data objectives of Annex H of IEEE/EIA 12207.0-1996	—	Support the life cycle data objectives of Annex H of IEEE/EIA 12207.0-1996
b) Describe any function using well-defined notation	4.3 Characteristics of a good SRS	—
c) Define no requirements that are in conflict	4.3 Characteristics of a good SRS	—
d) User standard terminology and definitions	5.1.3 Definition	—
e) Define each unique requirement one to prevent inconsistency	4.3 Characteristics of a good SRS	—
f) Uniquely identify each requirement	4.3 Characteristics of a good SRS	—

B.3.4 Compliance with life cycle data objectives

In addition to the content requirements, life cycle data shall be managed in accordance with the objectives provided in Annex H of IEEE/EIA 12207.0-1996.

B.4 Conclusion

The analysis suggests that any SRS complying with this recommended practice and the additions shown in Table B.2 and Table B.3 also complies with the requirements of an SRD in IEEE/EIA 12207.1-1997. In addition, to comply with IEEE/EIA 12207.1-1997, an SRS shall support the life cycle data objectives of Annex H of IEEE/EIA 12207.0-1996.

To order IEEE standards…

Call 1. 800. 678. IEEE (4333) in the US and Canada.

Outside of the US and Canada:
1. 732. 981. 0600

To order by fax:
1. 732. 981. 9667

IEEE business hours: 8 a.m.–4:30 p.m. (EST)

For on-line access to IEEE standards information…

Via the World Wide Web:
http://standards.ieee.org/

Via ftp:
stdsbbs.ieee.org

ISBN 0-7381-0332-2

ภาคผนวก ค

SOFTWARE REQUIREMENT GATHERING FORM

*** ตัวอย่างต่อไปนี้เป็นเพียงแนวคิดเพื่อใช้ในการศึกษาเกี่ยวกับเรื่องความต้องการด้านซอฟต์แวร์ ***

หากนำไปใช้จริง จำเป็นจะต้องวิเคราะห์ความต้องการให้เหมาะสมกับบริบท

และสามารถลด-เพิ่ม หัวข้อต่างๆ ได้

ทั้งนี้ ผู้เขียนปรับปรุงและประยุกต์แบบฟอร์มดังกล่าวจากที่มีอยู่เดิม ของบริษัท TTT BROTHER CO., LTD. และขอขอบคุณไว้ ณ ที่นี้

(For Educational Purposes Only)

[Logo]

SOFTWARE REQUIREMENT GATHERING FORM

วันที่จัดทำ DD/MM/YYYY

วันที่ปรับปรุงล่าสุด DD/MM/YYYY

สารบัญ

ประวัติการแก้ไข

เวอร์ชัน	วันที่	ผู้ดำเนินการแก้ไขโดย	ผู้ตรวจสอบ	รายละเอียด/เหตุผล
1.0	dd/mm/yyyy	Mr.....	Mr.....	n/a

Section 1: ข้อมูลเบื้องต้นเกี่ยวกับลูกค้า

ชื่อบริษัท	
สถานที่ตั้ง	
เบอร์โทรศัพท์	
Website	
ประเภทธุรกิจ	
จำนวนสาขา	
จำนวนหน่วยงาน/แผนก	
จำนวนพนักงานทั้งหมดในหน่วยงาน / แผนก	
ตัวอย่างผัง/โครงสร้างองค์กร	❑ มี ❑ ไม่มี

Section 2: ชื่อผู้ประสานงาน

ชื่อ-สกุลผู้ติดต่อหลัก	(โปรดระบุคำนำหน้าชื่อ)
ตำแหน่ง	
สังกัดหน่วยงาน	
ที่อยู่สำหรับการติดต่อ	
เบอร์โทรศัพท์	
Email address	
รูปแบบการติดต่อที่อนุญาต (เลือกได้มากกว่า 1 ข้อ)	❑ ติดต่อตามที่อยู่ ❑ โทรศัพท์ ❑ Email address
ช่วงเวลาที่สะดวกในการติดต่อ	

จัดทำโดย (SA)	Mr./Ms.	**ผู้อนุมัติ (ลูกค้า)**	Mr./Ms. .....
วันที่จัดทำ	04/12/2566		*Customer Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566		

❑คนเดียวกับผู้ติดต่อหลัก ❑ไม่ใช่คนเดียวกับผู้ติดต่อหลัก

ชื่อ-สกุลผู้รับผิดชอบโครงการ/ผู้มีอำนาจในการตัดสินใจ (Owner)	(โปรดระบุคำนำหน้าชื่อ)
ตำแหน่ง	
สังกัดหน่วยงาน	
ที่อยู่สำหรับการติดต่อ	
เบอร์โทรศัพท์	
Email address	
รูปแบบการติดต่อที่อนุญาต (เลือกได้มากกว่า 1 ข้อ)	❑ ติดต่อตามที่อยู่ ❑ โทรศัพท์ ❑ Email address ❑ ติดต่อผ่านผู้ติดต่อหลักเท่านั้น
ช่วงเวลาที่สะดวกในการติดต่อ	

❑คนเดียวกับผู้ติดต่อหลัก/ผู้รับผิดชอบโครงการ ❑ไม่ใช่คนเดียวกับผู้ติดต่อหลัก/ผู้รับผิดชอบโครงการ

ชื่อ-สกุลผู้ให้ความต้องการ	(โปรดระบุคำนำหน้าชื่อ)
ตำแหน่ง	
สังกัดหน่วยงาน	
ที่อยู่สำหรับการติดต่อ	
เบอร์โทรศัพท์	
Email address	
รูปแบบการติดต่อที่อนุญาต (เลือกได้มากกว่า 1 ข้อ)	❑ ติดต่อตามที่อยู่ ❑ โทรศัพท์ ❑ Email address ❑ ติดต่อผ่านผู้ติดต่อหลักเท่านั้น
ช่วงเวลาที่สะดวกในการติดต่อ	

จัดทำโดย (SA)	Mr./Ms.	**ผู้อนุมัติ (ลูกค้า)**	Mr./Ms. .....
วันที่จัดทำ	04/12/2566		*Customer Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566		

Section 3: ข้อมูลผู้ใช้

หน่วยงานที่ใช้งานระบบ	
ระบุกลุ่มผู้ใช้งานระบบ (ตัวอย่าง เพื่อระดับปฏิบัติการ, เพื่อผู้บังคับบัญชา)	
จำนวนผู้ใช้งานระบบ (คน)	
ทักษะการใช้งานระบบคอมพิวเตอร์โดยรวมอยู่ในเกณฑ์ใด	❑ ไม่มีพื้นฐานด้านคอมพิวเตอร์ใดๆ ❑ ขั้นพื้นฐาน ❑ ขั้นปานกลาง ❑ ขั้นสูง
ข้อจำกัดของผู้ใช้งาน	

Section 4: As-Is Process Background

ด้านกระบวนการทำงาน		
1	ชื่อระบบงานที่ต้องการ (ถ้ามี)	
2	วัตถุประสงค์ของระบบงาน	
3	ระบบงานปัจจุบัน	❑ เป็นแบบ Manual ❑ มีระบบคอมพิวเตอร์ใช้
4	กรณีระบบงานปัจจุบัน เป็นระบบคอมพิวเตอร์อยู่แล้ว กรุณาระบุชื่อ Software/Application พร้อมทั้ง Version	
5	กรุณาระบุชื่อระบบงานหลัก ถ้าระบบงานใหม่ที่ต้องการ เป็นระบบงานย่อยภายใต้ระบบงานหลัก	
6	กรุณาอธิบายลักษณะ และขั้นตอนของระบบงานปัจจุบันโดยสังเขป	
7	ตัวอย่าง Flow การทำงานปัจจุบัน	❑ มี ❑ ไม่มี

จัดทำโดย (SA)	Mr./Ms.	**ผู้อนุมัติ (ลูกค้า)**	Mr./Ms. .....
วันที่จัดทำ	04/12/2566		*Customer Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566		

8	ระบบงานปัจจุบัน มีการใช้รายงานหรือแบบฟอร์มเอกสารต่าง ๆ อะไรบ้าง	
9	ตัวอย่างรายงาน และแบบฟอร์มเอกสารทั้งหมด	❑ มี ❑ ไม่มี
10	ระบบงานปัจจุบัน มีการเชื่อมต่อกับระบบงานอื่น หรือระบบคอมพิวเตอร์อื่น ๆ หรือไม่อย่างไร	
11	โดยระบบงาน จำเป็นต้องคำนึงถึงกฎหมาย หรือต้องสอดคล้องตามมาตรฐานสากล หรือต้องส่งข้อมูล ให้หน่วยงานราชการใด ๆ หรือไม่	
12	ระบบปัจจุบัน มีความไม่สะดวก หรือติดปัญหาใด ๆ หรือไม่ อย่างไร	
13	กรณีที่พบปัญหาในระบบงานปัจจุบัน ปัญหาใดบ้าง ที่ต้องการได้รับการแก้ไขเป็นพิเศษ หรือเร่งด่วน	
14	ปริมาณเอกสารหลักต่าง ๆ ที่เกิดขึ้นต่อวัน ในระบบงานปัจจุบัน เช่น ปริมาณใบกำกับภาษีต่อวัน, ปริมาณใบเสร็จรับเงินต่อวัน ฯลฯ	
15	จำนวนข้อมูลหลักในระบบปัจจุบัน เช่น จำนวนรายการสินค้าทั้งหมด, จำนวนสาขาทั้งหมด ฯลฯ	
ด้านระบบคอมพิวเตอร์และเครือข่าย (Infrastructure)		
16	ประเภทของ Hardware, Server และระบบ Network ที่ใช้อยู่ในปัจจุบันมีอะไรบ้าง	
17	ปัจจุบัน ใช้ระบบการจัดการฐานข้อมูล (Database) ใดอยู่	
18	มีการใช้งานระบบคอมพิวเตอร์มาเป็นระยะเวลาเท่าใด	

จัดทำโดย (SA)	Mr./Ms.	**ผู้อนุมัติ (ลูกค้า)**	Mr./Ms. .....
วันที่จัดทำ	04/12/2566		Customer Sign Your Name
วันที่ปรับปรุงล่าสุด	04/12/2566		

19	มีระบบงานใดบ้างที่ใช้ระบบคอมพิวเตอร์แล้ว และระบบใดที่ยังไม่ได้ใช้ระบบคอมพิวเตอร์	
20	ระบบการป้องกันไวรัสทั้งระบบ	❑ มี ❑ ไม่มี
21	ปัจจุบันมีการจัดการสำรองข้อมูล (Backup) อย่างไร	
22	ปัจจุบันมีผู้ดูแลระบบคอมพิวเตอร์ในบริษัทเอง หรือใช้บริการจากบริษัทอื่นเป็นผู้ดูแล และชื่อบริษัทอะไร	
23	ปัจจุบันมีผู้พัฒนาระบบคอมพิวเตอร์ในบริษัทเอง หรือใช้บริการจากบริษัทอื่นเป็นผู้ดูแล และชื่อบริษัทอะไร	

Section 5: To-Be Software Requirements

ด้านการใช้งานระบบ		
1	ประเภทความต้องการระบบงาน	❑ ต้องการเพิ่มเติมแก้ไขระบบงานเดิม ❑ ระบบงานใหม่
2	ลักษณะการทำงาน หรือฟังก์ชั่นการทำงานใดบ้างที่ต้องการให้มีในระบบงานใหม่	
3	ช่วงวันและเวลาใดที่ต้องการใช้งานระบบ หรือ ระบบจำเป็นต้องใช้ตลอดเวลา (24 ชม. ทุกวัน)	
4	สถานที่ที่ต้องการติดตั้งระบบ และกำหนดการที่ต้องการใช้งาน	
5	มีความต้องการรายงานใหม่เพิ่มเติม จากที่ไม่เคยมีอยู่ในปัจจุบันหรือไม่ ถ้ามี คือรายงานอะไรบ้าง	
6	ตัวอย่างรายงาน หรือเอกสารที่ต้องการ	❑ มี ❑ ไม่มี

จัดทำโดย (SA)	Mr./Ms.	**ผู้อนุมัติ (ลูกค้า)**	Mr./Ms. .....
วันที่จัดทำ	04/12/2566		*Customer Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566		

7	หน่วยงานใดบ้าง ที่คาดว่าจะนำระบบงานใหม่ไปใช้ รวมถึงจำนวนผู้ใช้งานระบบใหม่ ของแต่ละหน่วยงาน (โดยประมาณ)	
8	นอกเหนือจากหน่วยงานในข้อ 7 มีหน่วยงานใดบ้าง ที่อาจไม่ได้เป็นผู้ใช้งานระบบใหม่ แต่เป็นผู้ใช้ข้อมูลหรือใช้เอกสารรายงานต่าง ๆ ที่ได้จากระบบ ถ้ามี ใช้เอกสารใด และใช้ในกรณีใดบ้าง	
9	ในระบบงานใหม่ ต้องการให้มีการอนุมัติเอกสารใด ๆ ผ่านระบบหรือไม่ ถ้าต้องการ มีเอกสารเรื่องใดบ้าง	
10	เรื่องการส่ง email ต่าง ๆ ผ่านระบบ มีความจำเป็นหรือไม่	
ด้านความปลอดภัยของข้อมูลและระบบ		
11	ข้อมูลในระบบเรื่องใดบ้าง ที่มีเพียงบางหน่วยงานสามารถเห็นข้อมูลได้ เช่น เฉพาะฝ่ายจัดซื้อเท่านั้น ที่สามารถเห็นต้นทุนสินค้าได้	
12	มีความเป็นไปได้หรือไม่ ที่ท่านจำเป็นต้องใช้งานระบบ ผ่านจากระบบเครือข่ายอื่น เช่น ต้องการใช้งานระบบผ่านจากระบบ internet เมื่ออยู่นอกที่ทำงาน	
13	ท่านต้องการให้ระบบบังคับเปลี่ยนรหัสผ่าน ตามระยะเวลาที่กำหนดหรือไม่ เช่น ระบบจะแจ้งให้ท่านเปลี่ยนรหัสผ่านทุก 3 เดือน และห้ามซ้ำกับรหัสผ่านเดิม	

จัดทำโดย (SA)	Mr./Ms.	**ผู้อนุมัติ (ลูกค้า)**	Mr./Ms. .....
วันที่จัดทำ	04/12/2566		*Customer Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566		

14	ท่านต้องการจัดกลุ่มผู้ใช้งานระบบ ให้มีสิทธิในการเข้าใช้งานและเข้าถึงข้อมูลแตกต่างกันหรือไม่ อย่างไร เช่น ฝ่ายขายสามารถอ่านข้อมูลสินค้าคงเหลือ ของฝ่ายคลังสินค้าได้ แต่ไม่สามารถแก้ไขได้	
15	ระบบใหม่ จำเป็นต้องมีการ Encrypt ข้อมูลในเรื่องใดบ้าง	
16	ข้อมูลที่ใช้ในการทดสอบระบบ มีข้อมูลเรื่องใดบ้าง ที่เป็นความลับ และไม่สามารถ บันทึกในระบบทดสอบได้ (จำเป็นต้อง มีการทำ Data Masking ในการทดสอบระบบ)	
ด้านเทคนิค		
17	ท่านต้องการจำกัดภาษาหรือเทคโนโลยีที่ใช้เขียนโปรแกรมหรือไม่ ถ้าต้องการโปรดระบุ ภาษาหรือเทคโนโลยีที่ต้องการ	
18	ท่านต้องการ database แบบเฉพาะเจาะจงหรือไม่ ถ้าต้องการโปรดระบุ	
19	ท่านต้องการจำกัด Platform หรือ OS ที่ต้องการหรือไม่	
20	ท่านต้องการ Source Code หรือไม่	
21	ระยะเวลาที่ต้องการเก็บข้อมูลเก่าไว้ในระบบ (ระยะเวลาในการ Purge Data)	
22	จำเป็นต้องมีการ Migrate Data จากระบบใด หรือไม่ ถ้ามี เป็นข้อมูลเรื่องใดบ้างปริมาณเท่าไร	
23	เครื่อง Server และ Environment ต่าง ๆในการติดตั้งระบบ สำหรับระบบทดสอบ และระบบจริง จะใช้ที่มีอยู่เดิมหรือต้องจัดหามาใหม่	

จัดทำโดย (SA)	Mr./Ms.	**ผู้อนุมัติ (ลูกค้า)**	Mr./Ms. .....
วันที่จัดทำ	04/12/2566		*Customer Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566		

24	ถ้า ใช้เครื่อง Server หรือ Environment เดิมในการติดตั้งระบบ Environment ดังกล่าว มี Capacity ที่เหลืออยู่เท่าไหร่บ้าง	
25	คุณลักษณะของเครื่อง Server และ Environment ต่าง ๆในการติดตั้งระบบสำหรับระบบทดสอบ และระบบจริง	

Environment Hardware

Type Machine	
Brand Machine	
CPU	
RAM	
OS	
Other	

Environment Software

.Net Framework	
Font	
Database	
Other	

Environment Client

Browser	
Type Machine	
Brand Machine	
CPU	
RAM	
OS	
Other	

26	ข้อจำกัดของเครื่อง Server และ Environment ต่าง ๆในการติดตั้งระบบสำหรับระบบทดสอบ และระบบจริง(ถ้ามี)	

Section 6: ข้อกำหนด/นโยบายการจัดการความต้องการ

1	มีการเปลี่ยนแปลงหรือมีนโยบายอะไรเป็นพิเศษหรือไม่ ที่เกี่ยวข้องกับระบบงานใหม่	
2	กรณีที่มีความต้องการระบบงานใหม่หลายระบบ ต้องการเริ่มใช้งานระบบใดก่อน (เรียงลำดับตามความต้องการ)	
3	ความต้องการในรายงานพิเศษเพิ่มเติม	
4	ตัวอย่างรูปแบบรายงาน	❑ มี ❑ ไม่มี
5	ระบุความต้องการพิเศษอื่น ๆ	
6	ตัวอย่างความต้องการพิเศษอื่น ๆ	❑ มี ❑ ไม่มี
7	แนวโน้มความต้องการระบบคอมพิวเตอร์ในอนาคตเป็นอย่างไร	

Section 7: ข้อมูลเพิ่มเติม

1		
2		
3		
4		
5		

จัดทำโดย (SA)	Mr./Ms.	**ผู้อนุมัติ (ลูกค้า)**	Mr./Ms. .....
วันที่จัดทำ	04/12/2566		*Customer Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566		

ผู้จัดทำเอกสาร	ผู้ตรวจทาน	ลูกค้าผู้ให้ข้อมูล
ลายเซ็น	ลายเซ็น	ลายเซ็น
(_____)	(_____)	(_____)
ตำแหน่ง: Business Analyst	ตำแหน่ง: Project Manager	ตำแหน่ง:_____
วัน/เดือน/ปี : วว/ดด/ปปปป	วัน/เดือน/ปี : วว/ดด/ปปปป	วัน/เดือน/ปี : วว/ดด/ปปปป

****หมายเหตุ

ข้อกำหนดในการใช้ Requirement Survey Form

1. ในส่วนของผู้ให้ความต้องการใน Section ที่ 2 Contracts Information หากมีจำนวนมากกว่า 1 ขึ้นไป ให้ผู้จัดทำเอกสาร ทำการสร้างบล็อกในการกรอกข้อมูล เพิ่มขึ้นมาเท่าจำนวนผู้ให้ความต้องการ ที่จะต้องเก็บข้อมูล
2. กรณีเปิดโครงการเป็น SDLC แบบ Waterfall การเก็บข้อมูล Requirement Survey Form จะเก็บเพียงครั้งเดียวและเกิดเอกสารเพียง 1 เอกสารเท่านั้น และเก็บไว้ใน Phase Requirement ซึ่งหากมีการเปลี่ยนแปลงให้เกิดเอกสารใหม่พร้อมกับการเปลี่ยนแปลง Version ของเอกสาร
3. กรณีเปิดโครงการเป็น SDLC แบบ Iteration การเก็บข้อมูล Requirement Survey Form อาจจะเก็บเพียงครั้งเดียว หรือหลายครั้งตามการเปลี่ยนแปลงของงานในโครงการ ตามการวางแผน และเก็บเอกสารไว้ใน Phase Requirement ของทุกๆ Iterationซึ่งหากมีการเปลี่ยนแปลงให้เกิดเอกสารใหม่ที่แก้ไขเรียบร้อยแล้วพร้อมกับการเปลี่ยนแปลง Version ของเอกสาร
4. กำหนดให้ Requirement Survey Form ต้องมีลายเซ็นลูกค้ากำกับทุกหน้า เพื่อยืนยันความถูกต้องครบถ้วน สมบูรณ์ของข้อมูล

ภาคผนวก ง

REQUIREMENTS PROTOTYPE

*** ตัวอย่างต่อไปนี้เป็นเพียงแนวคิดเพื่อใช้ในการศึกษาเกี่ยวกับเรื่องความต้องการด้านซอฟต์แวร์ ***

หากนำไปใช้จริง จำเป็นจะต้องวิเคราะห์ความต้องการให้เหมาะสมกับบริบท

และสามารถลด-เพิ่ม หัวข้อต่างๆ ได้

ทั้งนี้ ผู้เขียนปรับปรุงและประยุกต์แบบฟอร์มดังกล่าวจากที่มีอยู่เดิม ของบริษัท TTT BROTHER CO., LTD. และขอขอบคุณไว้ ณ ที่นี้

(For Educational Purposes Only)

[Logo]

Requirements Prototype

เวอร์ชัน 1.0

วันที่จัดทำ DD/MM/YYYY

วันที่ปรับปรุงล่าสุด DD/MM/YYYY

สารบัญ

จัดทำโดย (SA)	Mr./Ms.		Mr./Ms. .....
วันที่จัดทำ	04/12/2566	**ผู้ตรวจสอบ/**	*Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566	**ผู้อนุมัติ (PM)**	

สารบัญรูป

จัดทำโดย (SA)	Mr./Ms.		Mr./Ms. .....
วันที่จัดทำ	04/12/2566	**ผู้ตรวจสอบ/**	*Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566	**ผู้อนุมัติ (PM)**	

ปรับวัติการแก้ไข

#	วันที่	เวอร์ชัน	ประเภท (Create/Edit/Approve)	รายละเอียด/เหตุผล	ผู้รับผิดชอบ
1	04/12/2566	1.0	Create/Approve		Mr./Ms. .....

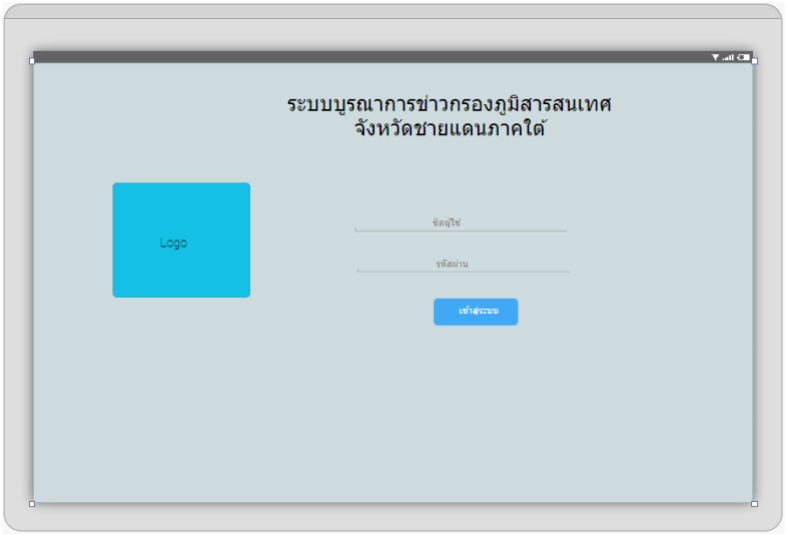
จัดทำโดย (SA)	Mr./Ms.		Mr./Ms. .....
วันที่จัดทำ	04/12/2566	ผู้ตรวจสอบ/	*Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566	ผู้อนุมัติ (PM)	

1. Screen & Interfaces

การออกแบบหน้าจอการใช้งานของระบบข่าวกรองภูมิสารสนเทศ จังหวัดชายแดนใต้ ประกอบด้วยส่วนต่างๆ ดังต่อไปนี้

1.1. หน้าจอล็อกอิน

การเข้าใช้งานโปรแกรมในครั้งแรก ผู้ใช้งานจำเป็นจะต้องกรอกชื่อผู้ใช้งานและรหัสผ่านก่อน จึงจะสามารถเข้าใช้งานในส่วนอื่นๆ ได้



รูปที่ 1 หน้าจอล็อกอิน

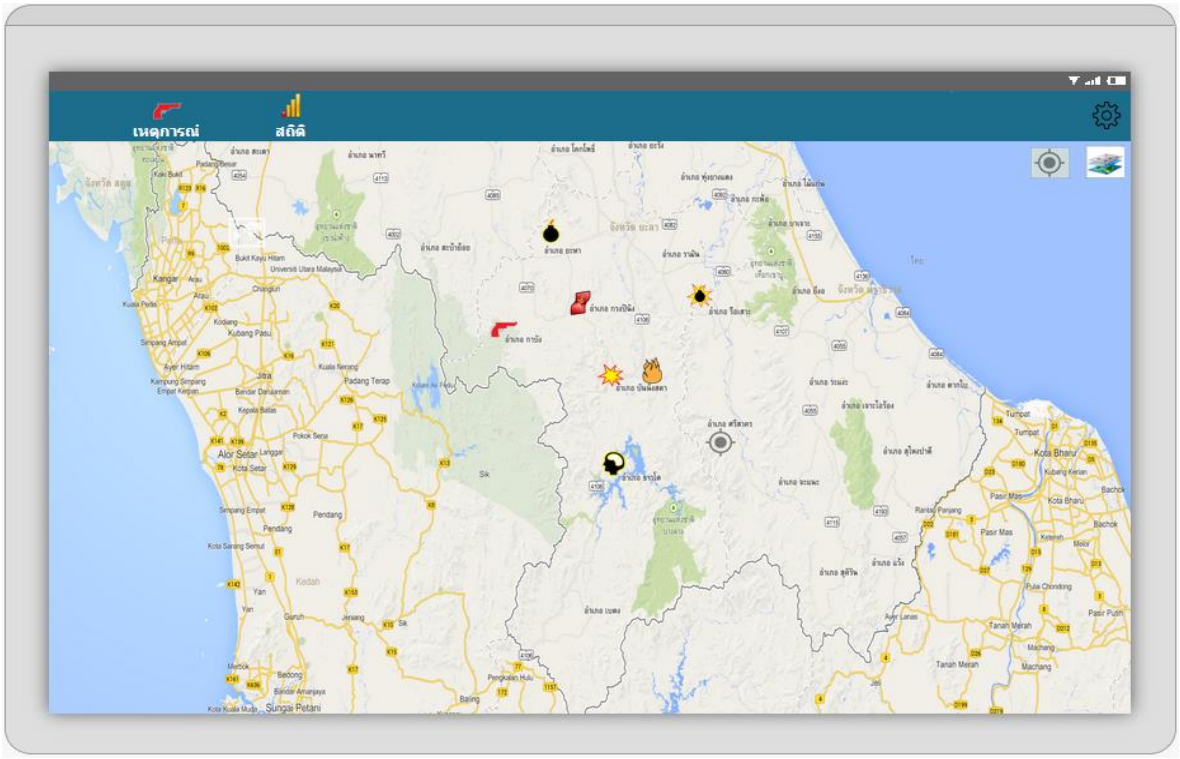
ตารางแสดงรายละเอียด

Screen Name	หน้าจอล็อกอิน
Objectives	เพื่อตรวจสอบสิทธิ์ในการเข้าใช้งานโปรแกรม
Description	สำหรับการเข้าใช้งานระบบครั้งแรก ผู้ใช้งานจำเป็นต้องกรอกชื่อผู้ใช้งานและรหัสผ่าน จึงจะสามารถเข้าสู่โปรแกรมได้ ซึ่งหน้าจอล็อกอิน ประกอบด้วย • โลโก้, ชื่อระบบงาน, ชื่อผู้ใช้งาน, รหัสผ่าน และปุ่มล็อกอิน
User	ผู้ใช้งานทุกคน
Other	-

จัดทำโดย (SA)	Mr./Ms.		Mr./Ms. .....
วันที่จัดทำ	04/12/2566	**ผู้ตรวจสอบ/**	*Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566	**ผู้อนุมัติ (PM)**	

1.2. หน้าจอแผนที่สถานการณ์

เมื่อผู้ใช้งานได้เข้าสู่โปรแกรมเรียบร้อยแล้ว โปรแกรมจะแสดงหน้าจอแผนที่สถานการณ์



รูปที่ 2 หน้าจอแสดงแผนที่สถานการณ์

ตารางแสดงรายละเอียด

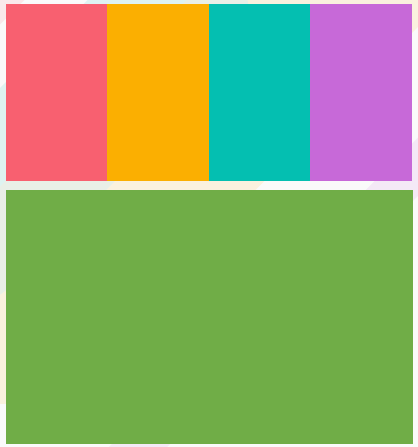
Screen Name	หน้าจอแสดงแผนที่สถานการณ์
Objectives	แสดงหน้าจอการทำงาน หลังจากที่ได้เข้าสู่โปรแกรมแล้ว
Description	หน้าจอแสดงแผนที่เหตุการณ์ ประกอบด้วย • แท็บเมนู (Action Bar) ได้แก่ เหตุการณ์, สถิติ และการตั้งค่า • แผนที่ • ไอคอน จีพีเอส แสดงตำแหน่งตนเอง • ไอคอน Base Map สำหรับเลือกรูปแบบแผนที่
User	ผู้ใช้งานทุกคน
Other	-

จัดทำโดย (SA)	Mr./Ms.		Mr./Ms. .....
วันที่จัดทำ	04/12/2566	**ผู้ตรวจสอบ/**	*Sign Your Name*
วันที่ปรับปรุงล่าสุด	04/12/2566	**ผู้อนุมัติ (PM)**	

ภาคผนวก จ

ตัวอย่าง Format การจัดทำเอกสารความต้องการ

(For Educational Purposes Only)

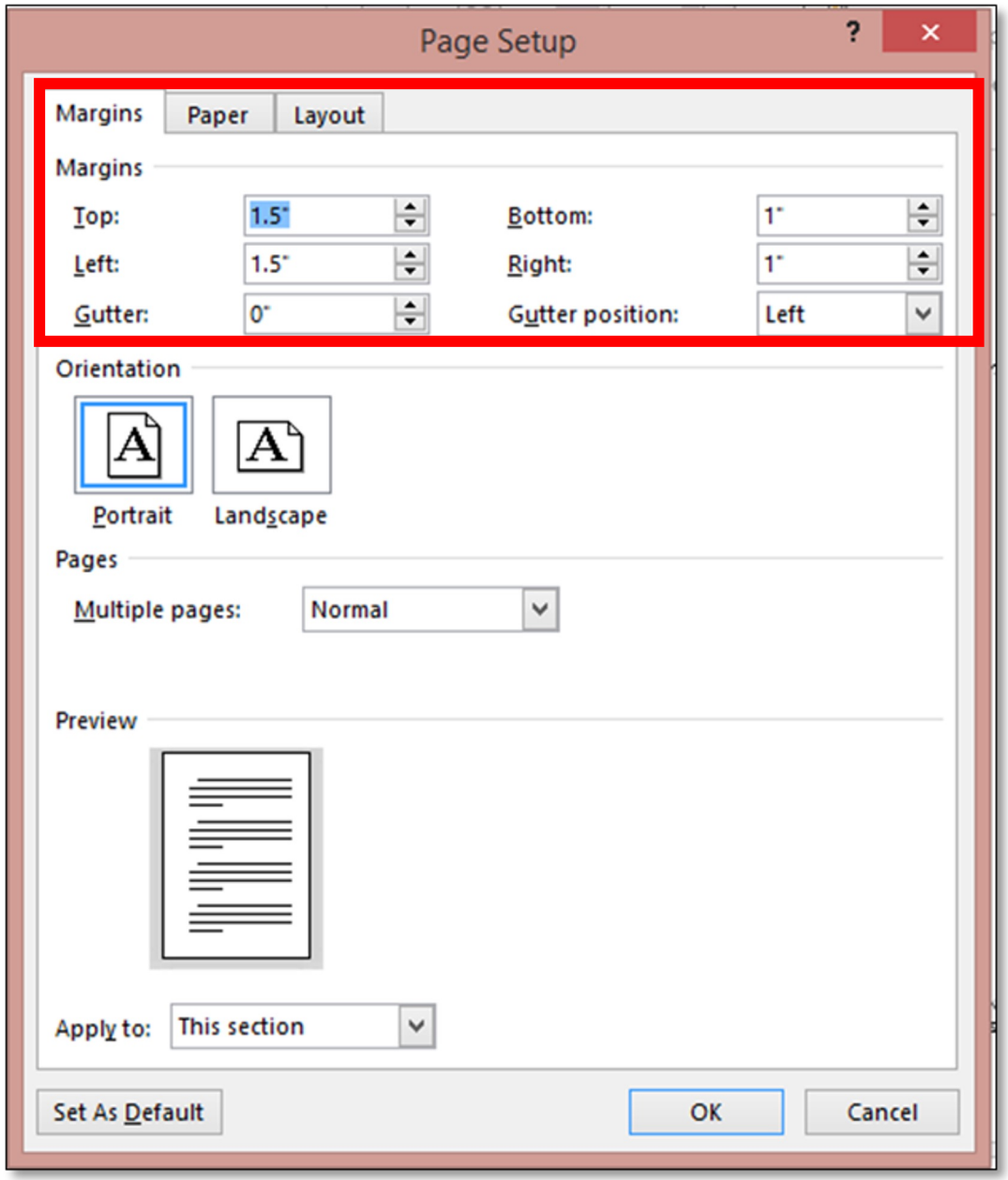


การตั้งค่า Format เอกสาร

หน้าปก

การตั้งค่าหน้ากระดาษ หน้าทั่วไป

- ขอบบน (Top) ให้เว้นระยะ 1.5 นิ้ว
- ขอบล่าง (Bottom) ให้เว้นระยะ 1 นิ้ว
- ขอบซ้าย (Left) ให้เว้นระยะ 1.5 นิ้ว
- ขอบขวา (Right) ให้เว้นระยะ 1 นิ้ว



สารบัญ

เกิน 1 หน้า ไม่ต้องใช้ สารบัญ (ต่อ)

เลข 1 ต้องตรงกับตัวอักษรแรกของคำว่าบทที่
คำว่า บทนำ ต้องตรงกับคำว่าที่
ลำดับ 1.1 ตรงกับตัวอักษรแรกของคำว่าบทนำ

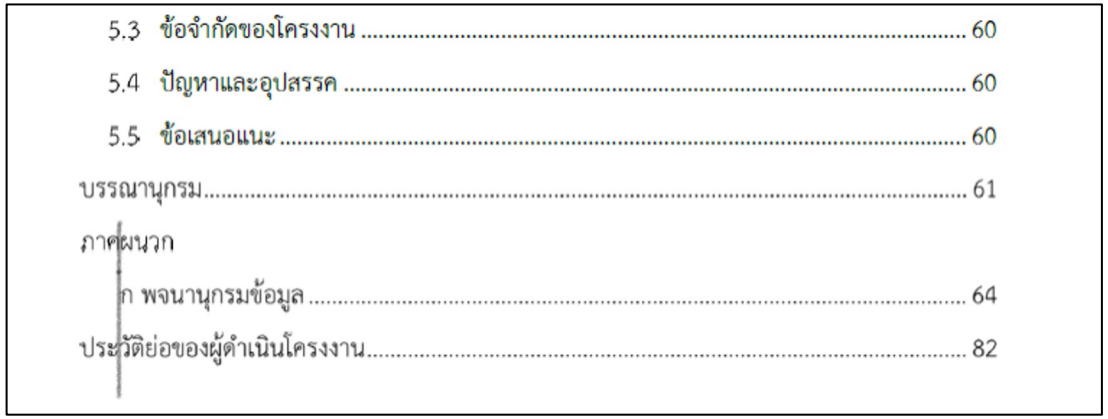
สารบัญ

สารบัญ

เกิน 1 หน้า ไม่ต้องใช้ สารบัญ (ต่อ)

ภาคผนวก

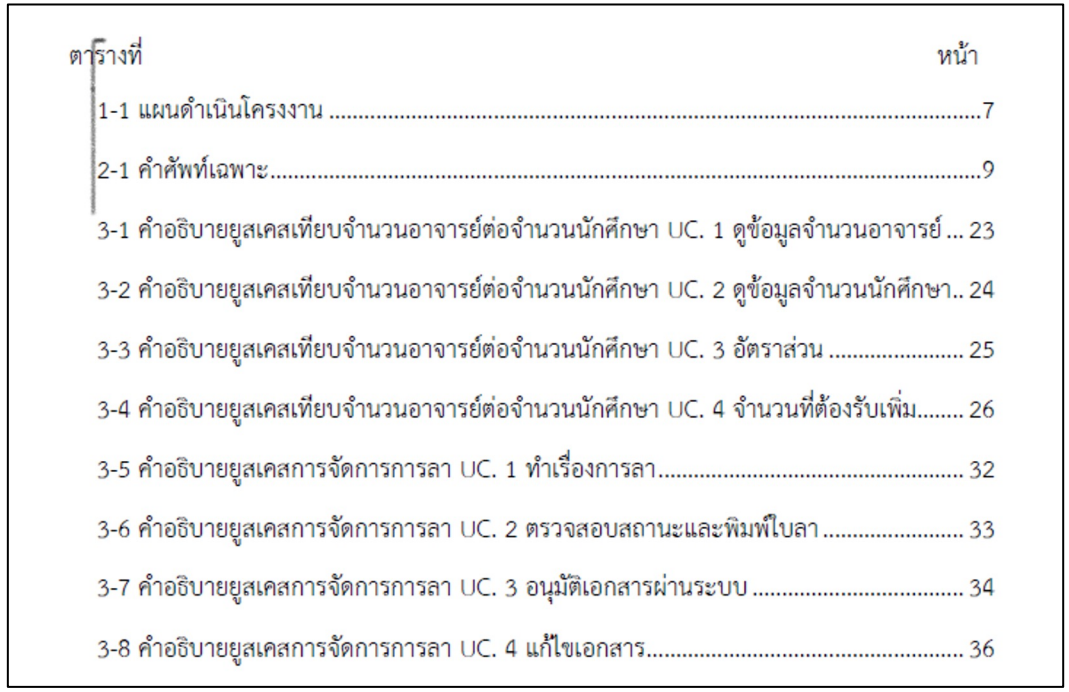
ก (ตรงกับตัว ผ ของคำว่า “ภาคผนวก”)



สารบัญตาราง

เกิน 1 หน้า ไม่ต้องใช้ สารบัญ (ต่อ)

เลข 1-1 ต้องตรงกับ ร ของคำว่าตาราง
ไม่ต้องมีคำว่า ตารางที่

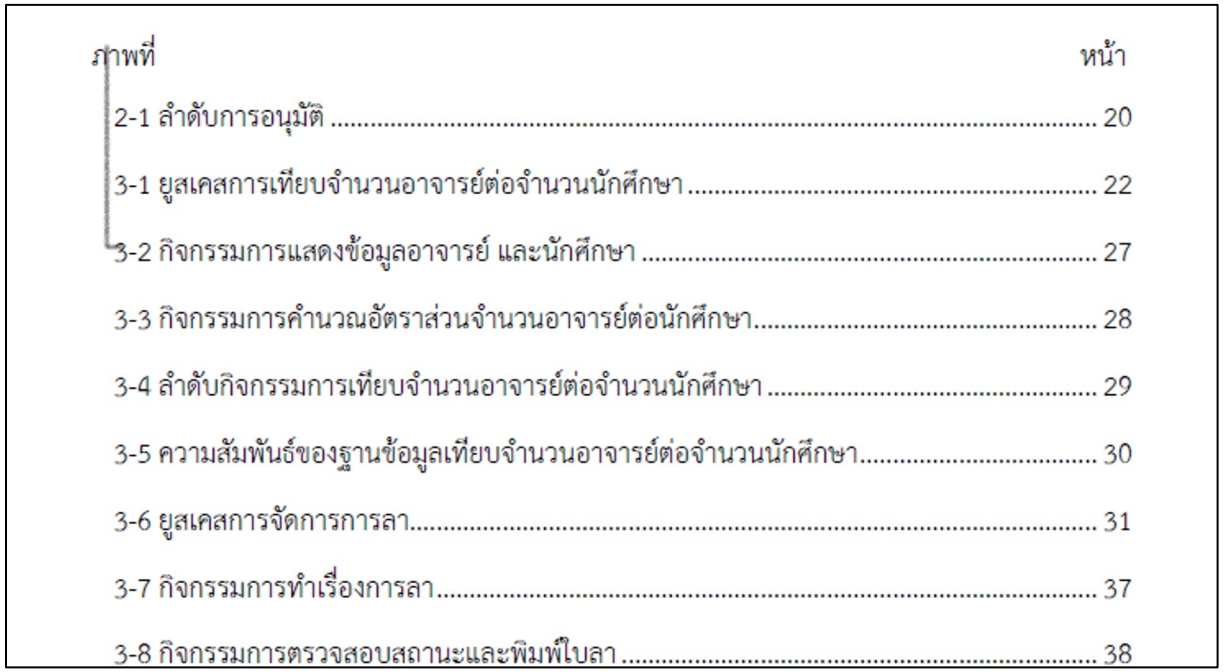


ตารางที่ หน้า

สารบัญรูปภาพ

เกิน 1 หน้า ไม่ต้องใช้ สารบัญ (ต่อ)

- เลข 1-1 ต้องตรงกับสระ อา ของคำว่าภาพที่
- ไม่ต้องมีคำว่า ภาพที่



ภาพที่ หน้า

เกริ่นนำหัวข้อ

เกริ่นนำจะต้องไม่ขึ้นต้นประโยคซ้ำกัน

บทที่ 1

บทนำ

วิชาสหกิจศึกษา (Cooperative Education) ถือเป็นรายวิชาหนึ่งที่ได้มีการจัดการเรียนการสอนขึ้น ซึ่งจะเกี่ยวข้องกับการฝึกปฏิบัติงานภายใต้สภาพแวดล้อมในการปฏิบัติงานจริง จากสถานประกอบการต่างๆ ซึ่งนิสิตสหกิจศึกษาจะมีระยะเวลาในการฝึกปฏิบัติงาน เป็นระยะเวลา 1 ภาคการศึกษารวมทั้งสิ้น 600 ชั่วโมง ระหว่างการฝึกปฏิบัติงานนั้นนิสิตสหกิจศึกษาจะมีบทบาทหน้าที่เปรียบเสมือนพนักงานในสถานประกอบการนั้นๆ ทุกประการ ซึ่งผู้ปฏิบัติงานสหกิจศึกษาศึกษาจะต้องมีการวางแผนในการดำเนินงาน เพื่อเสริมสร้างกระบวนการคิดวิเคราะห์ รวมไปถึงการแก้ไขอุปสรรค และปัญหาที่อาจเกิดขึ้นในระหว่างการปฏิบัติงาน เพื่อให้การปฏิบัติสหกิจศึกษานั้นสำเร็จบรรลุเป้าหมายตามวัตถุประสงค์ที่ได้กำหนดไว้อย่างมีประสิทธิภาพ

*****เกริ่นนำเข้าแต่ละบทที่ต้องไม่ต่ำกว่า 7 บรรทัด**

1.5 ขอบเขตของงานสหกิจศึกษาและข้อจำกัดของปัญหา

ในส่วนนี้จะเป็นการอธิบายถึงขอบเขตในการทำงานของผู้ปฏิบัติงานสหกิจศึกษาศึกษาประกอบไปด้วย ข้อมูลทั่วไป กลุ่มประชากร ตัวอย่างกลุ่มประชากร รวมไปถึงตัวอย่างข้อมูลที่ผู้ปฏิบัติงานสหกิจศึกษาศึกษาได้รับผิดชอบในการเก็บรวบรวมข้อมูล โดยมีรายละเอียดของแต่ละหัวข้อดังนี้

1.5.1 ข้อมูลทั่วไป

การปฏิบัติงานสหกิจศึกษาศึกษาในครั้งนี้ ผู้ปฏิบัติงานสหกิจศึกษาศึกษาได้ทำการเก็บรวบรวมข้อมูลจากการใช้บริการซอฟต์แวร์ของระบบต่างๆ ในแต่ละระบบจะมีทีมผู้พัฒนาเป็นผู้รับผิดชอบแตกต่างกันออกไปตามลักษณะการใช้งาน โดยผู้ปฏิบัติงานสหกิจศึกษาศึกษาเลือกทำการเก็บข้อมูลจาก 6 ระบบ ดังนี้

*****เกริ่นนำของแต่ละหัวข้อต้องไม่ต่ำกว่า 3 บรรทัด**

การใช้หัวข้อย่อย

แบบที่ถูกต้อง

5.4 ข้อเสนอแนะ

1) ผู้ใช้งานระบบติดตามและตรวจสอบบัณฑิตศึกษาต้องมีเข้าใจขั้นตอนในการทำวิทยานิพนธ์/ดุษฎีนิพนธ์ คณะวิทยาศาสตร์ มหาวิทยาลัยบูรพา

ไม่ต้องมี 1) ให้เขียนบรรยายเลย

5.4 ปัญหาและอุปสรรค

ในระหว่างการดำเนินโครงงานผู้ดำเนินโครงงานจะพบปัญหาอยู่บ่อยครั้งเนื่องจากผู้ดำเนินโครงงานไม่เชี่ยวชาญในการพัฒนาโปรแกรมจึงทำให้ เมื่อทำการแก้ไข Source code ในฟังก์ชันแล้วทำให้การทำงานของระบบผิดพลาด ซึ่งจะเป็นการเพิ่มระยะเวลาในการทำงานมากขึ้น และการแก้ไขไฟล์ในมอดูลที่ได้รับของตนเองแล้วทำให้การทำงานในมอดูลต่าง ๆ ของระบบเกิดความผิดพลาดไปด้วย เนื่องจากไม่ได้สื่อสารกันของภายในทีมของผู้ดำเนินโครงงาน รวมทั้งปัญหาทางด้านทรัพยากรของห้องปฏิบัติงานวิจัยวิศวกรรมระบบสารสนเทศ เนื่องจากมีสายแลนที่ไม่เพียงพอ หรือมีบางสายที่ไม่สามารถใช้งานได้ ทำให้ต้องเชื่อมต่อไวไฟ ส่งผลให้อินเทอร์เน็ตในการทำงานนั้นหลุดอยู่บ่อยครั้ง

5.4 ข้อเสนอแนะ

1) ผู้ใช้งานระบบติดตามและตรวจสอบบัณฑิตศึกษาต้องมีเข้าใจขั้นตอนในการทำวิทยานิพนธ์/ดุษฎีนิพนธ์ คณะวิทยาศาสตร์ มหาวิทยาลัยบูรพา

2) ขั้นตอนในการทำวิทยานิพนธ์/ดุษฎีนิพนธ์ในแต่ละสถาบันอาจมีความแตกต่างกัน อาจจะต้องมีการแก้ไขการทำงานเพื่อให้รองรับกับสถาบันหรือคณะที่ต้องการ

แบบที่ถูกต้อง

การใส่เลขหน้า

- หน้าที่เป็น บทที่ 1 2 3 4 5 ไม่ต้องมีเลขหน้า
- หน้าคั่นภาคผนวก ก ข และ ค ไม่ใส่เลขหน้า

กั้นหน้า-กั้นหลัง

กดใช้เพื่อกั้นหน้า และกั้นหลัง

วิชาสหกิจศึกษา (Cooperative Education) ถือเป็นรายวิชาหนึ่งที่ได้มีการจัดการเรียนการสอนขึ้น ซึ่งจะเกี่ยวข้องกับการฝึกปฏิบัติงานภายใต้สภาพแวดล้อมในการปฏิบัติงานจริง จากสถานประกอบการต่างๆ ซึ่งนิสิตสหกิจศึกษาจะมีระยะเวลาในการฝึกปฏิบัติงาน เป็นระยะเวลา 1 ภาคการศึกษารวมทั้งสิ้น 600 ชั่วโมง ระหว่างการฝึกปฏิบัติงานนั้นนิสิตสหกิจศึกษาจะมีบทบาทหน้าที่เปรียบเสมือนพนักงานในสถานประกอบการนั้นๆ ทุกประการ ซึ่งผู้ปฏิบัติงานสหกิจศึกษาศึกษาจะต้องมีการวางแผนในการดำเนินงาน เพื่อเสริมสร้างกระบวนการคิดวิเคราะห์ รวมไปถึงการแก้ไขอุปสรรค และปัญหาที่อาจเกิดขึ้นในระหว่างการปฏิบัติงาน เพื่อให้การปฏิบัติสหกิจศึกษานั้นสำเร็จบรรลุเป้าหมายตามวัตถุประสงค์ที่ได้กำหนดไว้อย่างมีประสิทธิภาพ

การเขียน

- กรณีเป็นประโยค วลี อักษรตัวแรกขึ้นต้นด้วยตัวอักษรตัวใหญ่
  - วิศวกรซอฟต์แวร์ (Software engineer)
  - แอปพลิเคชันสำหรับอุปกรณ์เคลื่อนที่ (Mobile application)
  - รายงานสรุปผล (Summary report)
- กรณีเป็นคำศัพท์ตัวย่อ ทุกคำขึ้นต้นด้วยตัวอักษรตัวใหญ่
  - แผนภาพแสดงการทำงานของระบบที่มีรูปแบบ OOP (Object Oriented Programming)
  - โดยใช้สัญลักษณ์ UML (Unified Modeling Language)
  - ภาษา SQL (Structured Query Language)
  - การพัฒนาซอฟต์แวร์ (Software Development Life Cycle: SDLC)

การเขียน

ภาษาอังกฤษ

- แยกของที่อยู่ใน Series ออกจากกัน
  - เช่น I can speak American English, French, German, and Mandarin Chinese
- แยกส่วนขยายหรือข้อความที่สามารถละได้ออกจากประโยค
  - เช่น *Mary walked to the party, but she was unable to walk home.*

การใช้เครื่องหมาย

วรรคหน้า วรรคหลัง

- ระบบสารสนเทศเพื่อสนับสนุนการผลิตบัณฑิต ประกอบไปด้วยระบบต่าง ๆ ได้แก่ ระบบการรับสมัคร (Admission) ระบบประเมินการศึกษา (Open AMS) ระบบทะเบียนและประมวลผลการศึกษา (E-regist) และระบบกิจการนักศึกษา (Open SA)

วรรคหลัง

- ได้รับมอบหมายให้ดาเนินการพัฒนาระบบสารสนเทศ ระบบติดตามและตรวจสอบนิสิตระดับบัณฑิตศึกษา ในส่วนของการจัดการข้อมูลนิสิต ข้อมูลบุคลากรภายนอก ข้อมูลพื้นฐานระบบ ซึ่งจะเป็นส่วนที่นาไปใช้ต่อในการพัฒนาส่วนการทางานต่างๆ ของระบบ

การใช้เครื่องหมาย

- ไปยาลน้อย (ฯ) วรรคหลัง
  - เช่น การปฏิบัติงานสหกิจศึกษา ณ ห้องปฏิบัติการวิจัยวิศวกรรมระบบสารสนเทศ คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา ห้องปฏิบัติการวิจัยฯ ได้มอบหมายให้เจ้าหน้าที่ที่ปรึกษา คือ นายมาโนชญ์ ใจกว้าง เป็นผู้ดูแลให้คำปรึกษา
- ไปยาลใหญ่ (ฯลฯ) วรรคหน้า วรรคหลัง
  - เช่น ปัจจุบันนี้มีภาษาที่ใช้เขียนโปรแกรมจำนวนมาก เช่น ภาษาซี ภาษารูบี้ ภาษาจาวา ฯลฯ

การใช้เครื่องหมาย

- วรรคหน้า วรรคหลัง
  - เช่น แผ่นตารางทำการ (Spreadsheet) หมายถึง แผ่นตารางที่ประกอบด้วยแนวตั้ง และแนวนอน ตัดกันเป็นช่องสี่เหลี่ยม แนวตั้งเรียกว่า สดมภ์ (Column) แนวนอนเรียกว่า แถว (Row)

การใช้สรรพนาม

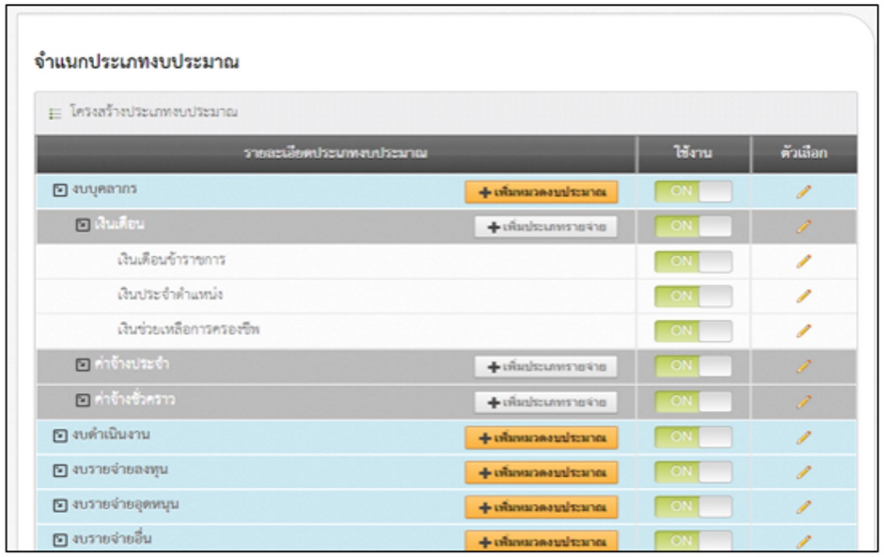
- **ไม่ควร**ใช้คำสรรพนาม ฉัน ผม เรา เขา เธอ มัน ท่าน คุณ เป็นต้น
  - เช่น การปฏิบัติงานสหกิจศึกษาในระยะเวลา 16 สัปดาห์ที่ผ่านมา **ผม** ได้รับมอบหมายให้พัฒนาระบบจัดการโครงการซอฟต์แวร์
- ให้ใช้คำว่า **ผู้ปฏิบัติงานสหกิจศึกษา** เท่านั้น

การใช้คำว่า “และ”

- ใช้เมื่อต้องการอธิบายว่ามีรายการอะไรบ้าง เช่น
  - ฉันสามารถพูดภาษาอังกฤษ ไทย จีน และลาวได้
  - รูปแบบการประเมินมี 2 ลักษณะ คือ การประเมินทั่วไป และการประเมินบุคลากร
  - ในการสำเร็จการศึกษา สาขาวิชาวิศวกรรมซอฟต์แวร์ จะต้องผ่านการสหกิจศึกษา และจัดทำโครงงาน

ชื่อภาพ/ชื่อตาราง

- ชื่อรูปภาพ จะอยู่ตำแหน่ง**กึ่งกลาง ใต้ภาพ**



ภาพที่ 4-32 หน้าจอเมนูจำแนกประเภทงบประมาณ

- ชื่อตาราง จะอยู่ตำแหน่ง**ชิดซ้าย บนหัวตาราง**

ตารางที่ 2-1 คำศัพท์เฉพาะ

ลำดับ	คำศัพท์	ความหมาย
1.	Test plan	แผนการทดสอบ
2.	Test case	กรณีทดสอบ
3.	System test	การทดสอบการทำงานร่วมกันทั้งหมดของระบบ
4.	Integration test	การทดสอบกระบวนการทำงานในแต่ละระบบย่อย
5.	Acceptance test	การทดสอบร่วมกันระหว่างผู้ใช้งานกับผู้พัฒนาระบบ
6.	Defect log	รายการข้อบกพร่อง
7.	Test summary report	รายงานสรุปผลการทดสอบ
8.	Error	ความผิดพลาดที่พบในการทดสอบ
9.	Defects	ข้อบกพร่องที่พบในการทดสอบ ระบบระบบบริหารจัดการกรอบมาตรฐานคุณวุฒิระดับอุดมศึกษา

การใช้ชื่อภาพ/ตาราง

กรณีมีรูปภาพ หรือ ตาราง ไม่มากกว่า 2

- ตารางที่ 1-1 ตารางแผนปฏิบัติงานสหกิจศึกษา ครั้งที่ 1
- ตารางที่ 1-1 ตารางแผนปฏิบัติงานสหกิจศึกษา ครั้งที่ 1 (ต่อ)

กรณีมีรูปภาพ หรือ ตาราง มากกว่า 2 ขึ้นไป

- ตารางที่ 1-1 ตารางแผนปฏิบัติงานสหกิจศึกษา ครั้งที่ 1
- ตารางที่ 1-1 ตารางแผนปฏิบัติงานสหกิจศึกษา ครั้งที่ 1 (2)
- ตารางที่ 1-1 ตารางแผนปฏิบัติงานสหกิจศึกษา ครั้งที่ 1 (3)

การตัดตาราง

กรณีที่ต้องตัดตารางเพื่อขึ้นหน้าใหม่ ต้องเอาหัวตารางมาต่อด้วย

ตารางที่ 3-2 คำอธิบายยูสเคสการตรวจสอบสินค้า

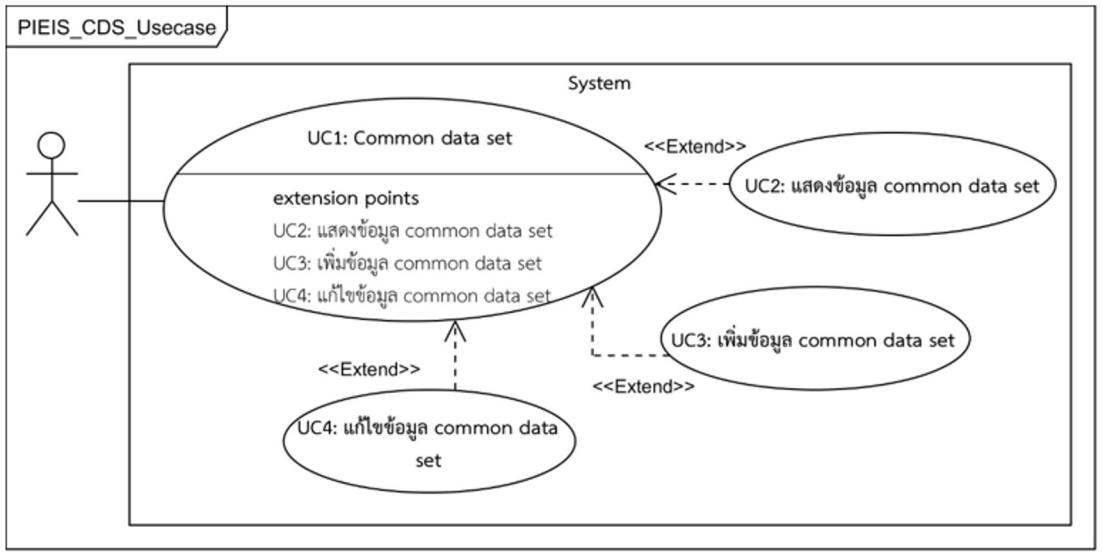
ชื่อยูสเคส : การตรวจสอบสินค้า	รหัส UC02	ระดับความสำคัญ : กลาง
ผู้กระทำหลัก : พนักงานคลังสินค้า	ระดับความซับซ้อน : ปานกลาง	
ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ : พนักงานคลังสินค้า		

ตารางที่ 3-2 คำอธิบายยูสเคสการตรวจสอบสินค้า (ต่อ)

ชื่อยูสเคส : การตรวจสอบสินค้า	รหัส UC02	ระดับความสำคัญ : กลาง
คำอธิบาย : เมื่อสินค้าถูกรับเข้ามาในคลังสินค้าแล้ว พนักงานคลังสินค้าจะสามารถตรวจสอบสถานะของสินค้าได้โดยการสแกนบาร์โค้ดของสินค้า เพื่อตรวจสอบสถานะของสินค้า หรือเพื่อให้ทราบถึงสถานที่จัดเก็บของสินค้า หรือถ้าพนักงานคลังสินค้าสแกนบาร์โค้ดของสถานที่จัดเก็บสินค้าเพื่อให้ทราบว่ามีสินค้าอะไรถูกจัดเก็บอยู่ในสถานที่จัดเก็บนี้		
สิ่งกระตุ้น : พนักงานคลังสินค้าต้องการตรวจสอบสินค้า		
ประเภทสิ่งกระตุ้น : ภายนอก	ต้องทำการรับสินค้าก่อนจึงจะสามารถตรวจสอบได้	
เงื่อนไขหลังการทำงาน	ทราบถึงสถานะของสินค้า	

การอ้างอิงรูปภาพ

แผนภาพยูสเคสแสดงความสัมพันธ์ระหว่างผู้ใช้กับระบบสบับสนุนการประกันคุณภาพการศึกษา ในส่วน Common data set โดยการทำงานที่เกี่ยวข้องในส่วนนี้คือ การแสดงข้อมูล Common data set การเพิ่มข้อมูล Common data set และการแก้ไขข้อมูล Common data set โดยที่ทั้ง 3 ส่วนเป็นส่วนหนึ่งของการทำงานในโมดูล Common data set ดังที่แสดงในแผนภาพยูสเคส ภาพที่ 3-4



ภาพที่ 3-4 แผนภาพยูสเคส โมดูล Common data set

****อ้างอิงได้ไม่เกิน 3 ภาพ ต่อการอธิบายภาพ 1 ครั้ง**

การอ้างอิงรูปภาพ

ภาพที่ 4-4 หน้าหลักการจัดการ Common data set

หน้าหลักการจัดการ Common data set ที่ปรากฏในภาพที่ 4-4 ประกอบไปด้วยเมนูเลือกหลักสูตร และปีประเมินเพื่อใช้เป็นข้อมูลตั้งต้นในการคนหาข้อมูลที่เกี่ยวข้องกับการประเมิน ส่วนการอัปเดตข้อมูล แสดงขึ้นเมื่อเลือกหลักสูตรและปีประเมินแล้วเหนือข้อมูล Common data set

การอ้างอิงตาราง

2.1 นิยามศัพท์เฉพาะ

การปฏิบัติงานสหกิจศึกษาในการพัฒนาระบบบันทึกข้อมูลตัวชี้วัด มีคำศัพท์เฉพาะที่ใช้ในการทำงานของส่วนต่าง ๆ ภายในระบบ โดยทั้งหมดในหัวข้อนี้จะเป็นการอธิบายถึงความหมายของคำศัพท์ที่ใช้ภายในระบบบันทึกข้อมูลตัวชี้วัด ดังตารางที่ 2-1

ตารางที่ 2-1 คำศัพท์เฉพาะระบบบันทึกข้อมูลตัวชี้วัด

ลำดับ	คำศัพท์	ความหมาย
1.	ตัวชี้วัด (KPI)	ดัชนีวัดความสำเร็จ หรือหน่วยวัดความสำเร็จของงานที่กำหนดขึ้น โดยเป็นหน่วยวัดที่มีผลเป็นตัวเลขที่เปรียบเทียบได้จริง เพื่อสร้างความชัดเจนในการติดตามและประเมินผลของงานที่ถูกกำหนด
2.	ปีงบประมาณ	ระยะเวลาตั้งแต่วันที่ 1 ตุลาคมของปีนั้นถึงวันที่ 30 กันยายนของปีถัดไป โดยปี พ.ศ. ถัดไปจะใช้เป็นชื่อสำหรับเรียกปีงบประมาณนั้น
3.	ผู้รับผิดชอบร่วม	ผู้ที่รับผิดชอบตัวชี้วัดร่วมกับผู้รับผิดชอบตัวชี้วัดหลัก

****อ้างอิงได้ไม่เกิน 3 ตาราง ต่อการอธิบายตาราง 1 ครั้ง**

การอ้างอิงตาราง

ตารางที่ 2-4 รายละเอียดมาตรฐานและตัวบ่งชี้ตามประเภทและลักษณะการประเมิน

องค์ประกอบ	ตัวบ่งชี้ที่	ชื่อตัวบ่งชี้	ลักษณะการประเมิน
ระดับหลักสูตร			
1. การกำกับมาตรฐาน	1.1	การบริหารจัดการหลักสูตรตามเกณฑ์มาตรฐานหลักสูตรที่กำหนดโดย สกอ.	ประเมินตามเกณฑ์
2. หลักสูตร การเรียนการสอน และการประเมินผู้เรียน	2.1	สาระของรายวิชาในหลักสูตร	ประเมินตามเกณฑ์
	2.2	การวางระบบผู้สอนและกระบวนการจัดการเรียนการสอน	ประเมินตามเกณฑ์
	2.3	การประเมินผู้เรียน	ประเมินตามเกณฑ์
	2.4	ผลการดำเนินงานหลักสูตรตามกรอบมาตรฐานคุณวุฒิ ระดับอุดมศึกษาแห่งชาติ	ประเมินตามเกณฑ์

จากตารางที่ 2-4 รายละเอียดมาตรฐานและตัวบ่งชี้ตามประเภทและลักษณะการประเมิน เป็นเกณฑ์การประเมินเพื่อประกันคุณภาพการศึกษาของวิทยาลัยภายในกำกับดูแลของสถาบันพระบรมราชชนก เมื่อถึงระยะเวลาต้องได้รับการประเมินเพื่อประกันคุณภาพการศึกษาจากองค์กรที่เกี่ยวข้อง ประกอบด้วย สภาพยาบาล สำนักงานคณะกรรมการอุดมศึกษา และสำนักงานคณะกรรมการสำนักงานรับรองมาตรฐานและประเมินคุณภาพการศึกษา

****อ้างอิงได้ไม่เกิน 3 ตาราง ต่อการอธิบายตาราง 1 ครั้ง**

ระดับเขียนหัวข้อย่อย

1.1.1 xxxxxxxxxxxxxxxxxx

1) xxxxxx

1.1) xxxxxx

1.1.1) xxxxxxxxx

● xxxxxxx

- xxxxxx

1.5.1 ขอบเขตของระบบหรือส่วนของระบบสนับสนุนการประกับคุณภาพการศึกษา

งานที่ได้รับมอบหมายในระบบสนับสนุนการประกันคุณภาพการศึกษา คือส่วนการจัดการ Common data set การจัดรูปแบบส่วนส่งออกเอกสารสรุปผลการดำเนินงาน และไอคอนตัวบ่งชี้ โดยมีรายละเอียดการวิเคราะห์ดังนี้

1) โมดูลการจัดการ Common data set เกณฑ์สำนักงานคณะกรรมการอุดมศึกษา ระดับหลักสูตร

1.1) ข้อมูลในส่วน Common data set เกณฑ์สำนักงานคณะกรรมการอุดมศึกษา ระดับหลักสูตร จะแบ่งออกเป็น 2 ส่วนคือ

- ข้อมูลที่ดึงมาจากระบบที่เกี่ยวข้อง
  - จำนวนอาจารย์ประจำหลักสูตรทั้งหมด
  - จำนวนอาจารย์ประจำหลักสูตรที่มีคุณวุฒิปริญญาโท
  - จำนวนอาจารย์ประจำหลักสูตรที่มีคุณวุฒิปริญญาเอก
  - จำนวนอาจารย์ประจำหลักสูตรที่มีตำแหน่งทางวิชาการ
  - จำนวน KPI ทั้งหมด
  - จำนวน KPI ที่ผ่านเกณฑ์

บรรณานุกรม

- **การเขียนบรรณานุกรมการหนังสือ**
  - [หมายเลขสารบัญ] ชื่อผู้แต่ง. ชื่อเรื่อง. ปีที่ตีพิมพ์.
- **การเขียนบรรณานุกรมการเว็บไซต์**
  - [หมายเลขสารบัญ] ชื่อผู้แต่ง. ชื่อเรื่อง. [ออนไลน์]. เข้าถึงจาก: URL Address. (วันที่สืบค้น)

บรรณานุกรม

ตัวอย่าง บรรณานุกรม

[1] กองวิชาการ และแผนงาน. (2558). งบประมาณจากกองวิชาการ และแผนงานกำกับงบประมาณ. [ออนไลน์]. เข้าถึงจาก: https://multi.dopa.go.th/tspd/ /web_index/official_letters.th (วันที่สืบค้น : 1 มกราคม 2563)

[2] ธนาคารแห่งประเทศไทย. (2558). ระบบรวบรวมข้อมูลกำกับเงินงบประมาณ. [ออนไลน์]. เข้าถึงจาก : https://coinmarketingcap.com (วันที่สืบค้น : 1 มีนาคม 2563)

[3] BINANCE. (2560). Binance Coin (BNB) คืออะไร ? ทำงานอะไร ?. [ออนไลน์]. เข้าถึงได้จาก : https://coinmant.com/2019/(วันที่สืบค้น : 1 มีนาคม 2563)

[4] Cypress. (2562). วิธีการใช้ Cypress.io ทดสอบระบบแบบอัตโนมัติ. [ออนไลน์]. เข้าถึงได้จาก : https://www.cypress.io.t (วันที่สืบค้น : 5 มีนาคม 2563)

[5] EGA. จัดตั้งตัวอย่างเอกสารขั้นตอนการทดสอบระบบออนไลน์ทั้งหมด. [ออนไลน์]. เข้าถึงได้จาก : http://www.ictc.ago.go.th/table_con/ (วันที่สืบค้น : 18 มีนาคม 2563)

[6] Grappik. (2563). รู้จัก Mock up Design สิ่งที่ช่วยเพิ่มมูลค่าให้กับงานภายในระบบ. [ออนไลน์]. เข้าถึงได้จาก :https://www.grappiki.com/ (วันที่สืบค้น : 1 มีนาคม 2563)

คำศัพท์ที่ควรระวัง

คำที่ควรระวัง

คำที่ถูกต้อง	คำผิด
แอปพลิเคชัน	แอปพลิเคชั่น แอพพลิเคชั่น แอพพลิเคชัน แอพพลิคเคชัน
อีเมล	อีเมล์ อีเมลล์
คำนวณ	คำนวน
ซอฟต์แวร์	ซอฟแวร์ ซอฟต์แวร์
โอเพนซอร์ส	โอเพ่นซอส โอเพนซอส โอเพ่นซอร์ส
อัปโหลด	อัพโหลด
อัปเดต	อัพเด็ต อัพเดต อัพเดท อัปเด็ต
แพ็กเกจ	แพ็คเกจ
เบราว์เซอร์	บราวเซอร์ เบราเซอร์
ซอร์สโค้ด	ซอสโค้ด ซอร์สโค๊ด

คำที่ควรระวัง (2)

คำที่ถูกต้อง	คำผิด
เซิร์ฟเวอร์	เซิฟเวอร์ เซิพเวอร์
ลายเซ็น	ลายเซ็น
สคริปต์	สคริปท์ สคริป
แท็ก	แทก
ฟังก์ชัน	ฟังก์ชั่น
เวอร์ชัน	เวอร์ชั่น
อินเทอร์เน็ต	อินเตอร์เน็ต อินเตอร์เนต อินเทอร์เนต
อิเล็กทรอนิกส์	อิเล็กโทรนิกส์ อิเล็คทรอนิกส์
ดาวน์โหลด	ดาวโหลด
บุคคล	บุคล

คำที่ควรระวัง (3)

คำที่ถูกต้อง	คำผิด
บุคลากร	บุคคลากร
ประเมิน	ประเมิณ
ผลลัพธ์	ผลลัพท์
คอลัมน์	คอลัมม์ คอลัม
ปรากฏ	ปรากฎ
โพรโทคอล	โปรโตตอล โปรโทคอล
วินโดวส์	วินโดว วินโดว์ วินโด้ วินโดน์
ฮาร์ดแวร์	ฮาร์ทแวร์ ฮาทแวร์ ฮาดแวร์
ลินุกซ์	ลีนุกซ์ ลีนักซ์ ไลนักซ์ ลินิกซ์ ลีนิกซ์
แอนดรอยด์	แอนดรอย แอนดรอยน์ แอนดรอยท์

คำที่ควรระวัง (4)

คำที่ถูกต้อง	คำผิด
ฟิลด์	ฟิวส์ ฟิลท์ ฟิล
เฟรมเวิร์ก	เฟรมเวิร์ค เฟรมเวิก
ลิขสิทธิ์	ลิขสิทธ์
อัตโนมัติ	อัติโนมัต อัตโนมัต
jQuery/JavaScript	Jquery, jquery/javascript, Java Script
สถานการณ์	สถานะการณ์ สถานการ
พจนานุกรม	พจนาณุกรม พจณานุกรม
บูรณาการ	บูรณาการณ์ บูรนาการ
สถานประกอบการ	สถานประกอบการณ์
ประยุกต์	ประยุกท์ ประยุก

คำที่ควรระวัง (5)

คำที่ถูกต้อง	คำผิด
โปรแกรมบรรณาธิกร	โปรแกรมบรรณาธิกรณ์
บรรณานุกรม	บรรณาณุกรม บรรนาณุกรม
อุปสรรค	อุปสรรค์
อนุญาต	อนุญาติ
อนุมัติ	อนุมัต
ปรากฏ	ปรากฎ
กฎหมาย	กฏหมาย
เทคนิค	เทคนิก เทกนิค
สิทธิ์การใช้งาน	สิทธิการใช้งาน
วิทยาศาสตรบัณฑิต	วิทยาศาสตร์บัณฑิต วิทยาศาสบัณฑิต

คำที่ควรระวัง (6)

คำที่ถูกต้อง	คำผิด
ไลบรารี	ไลบราลี ไลบรารี่
เว็บไซต์	เวบไซท์ เวบไซต์ เว็บไซท์ เวปไซด์
ลิงก์	ลิงค์ ลิ้งค์ ลิ๊งค์ ลิ้งก์ ลิ๊งก์
แพลตฟอร์ม	แพลทฟอร์ม
มาร์กอัป	มาร์คอัพ
แท็ก	แทก
ดิจิทัล	ดิจิตอล
บล็อก	บล็อค บล๊อค บล๊อก
ไดเรกทอรี	ไดเรคทอรี่
คลิก	คลิ๊ก

REQ SOFTWARE

REQUIREMENTS ENGINEERING AND DOCUMENTATION

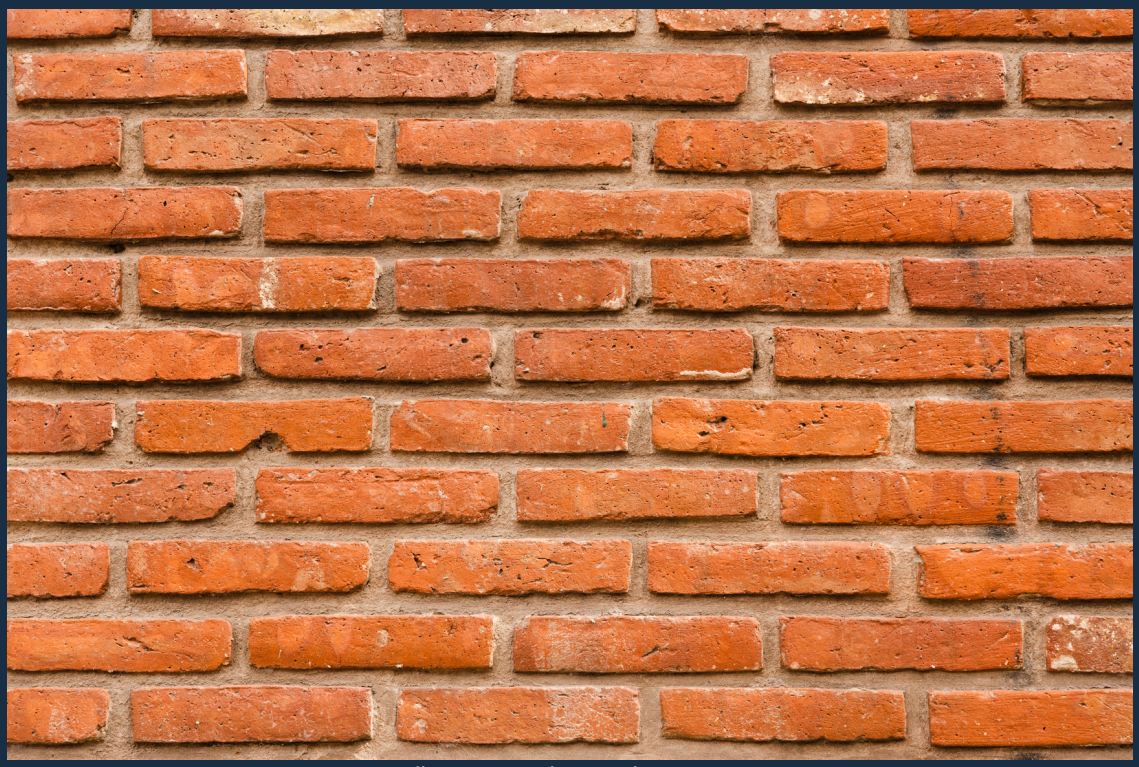


IMAGE BY FREEPIK : HTTPS://WWW.FREEPIK.COM/FREE-PHOTO/BRICKS-WALL-URBAN-TEXTURE-DESIGN_5458695

Bachelor of Science Program in
Software Engineering Faculty of Informatics,
Burapha University, Thailand