บทที่ 2

ทฤษฎีที่เกี่ยวข้องและงานวิจัย

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 แนวคิดเกี่ยวกับความต้องการและการจัดการความต้องการ (Requirements management) [1,2]

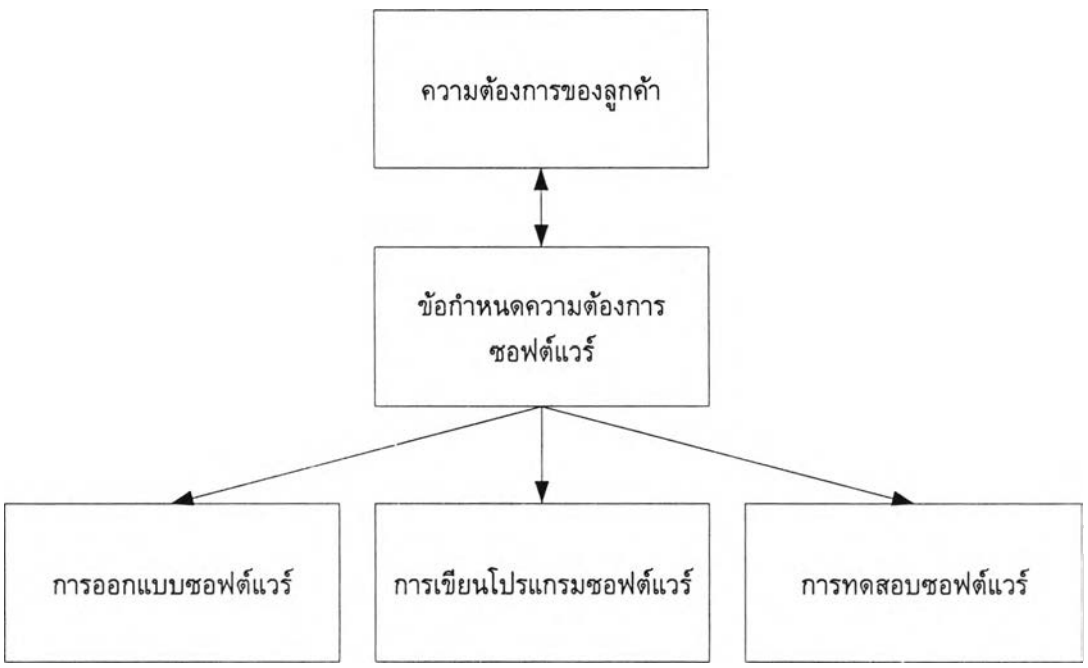
ความต้องการซอฟต์แวร์คือ คุณลักษณะหรือความสามารถของซอฟต์แวร์ที่ถูกกำหนดเพื่อใช้สำหรับแก้ปัญหาหรือดำเนินงานให้บรรลุเป้าหมายตามความต้องการของผู้ใช้ โดยที่ซอฟต์แวร์ที่พัฒนาขึ้นต้องมีความสามารถดำเนินการได้ตามความสามารถที่กำหนดได้ โดยที่ความสามารถเหล่านั้นจะถูกกำหนดในรูปของสัญญา ข้อกำหนด มาตรฐานหรือเอกสารมาตรฐานอื่นที่กำหนด โดยที่ความต้องการนั้นสามารถแบ่งได้เป็น 2 ประเภทคือ

- ความต้องการเชิงหน้าที่ คือ ความสามารถที่ถูกกำหนดโดยผู้ใช้ของระบบว่าซอฟต์แวร์ที่สร้างขึ้นจะสามารถดำเนินการได้
- ความต้องการที่ไม่ใช่ความต้องการเชิงหน้าที่ (Non-functional requirements) คือ สิ่งที่กำหนดถึงประสิทธิภาพของซอฟต์แวร์ ข้อบังคับและมาตรฐานที่ใช้ในระหว่างการพัฒนาซอฟต์แวร์

โครงการพัฒนาซอฟต์แวร์ที่ประสบความสำเร็จคือ การที่โครงการพัฒนาซอฟต์แวร์ที่สามารถสร้างซอฟต์แวร์ได้ถูกต้องและตรงกับความต้องการของผู้ใช้ภายในระยะเวลาและค่าใช้จ่ายที่กำหนดไว้ ซึ่งเห็นได้ว่าความต้องการเป็นส่วนหนึ่งในการกำหนดว่าซอฟต์แวร์ที่พัฒนานั้นประสบความสำเร็จหรือล้มเหลว ดังนั้นจึงเป็นสิ่งที่สำคัญในการกำหนดและทราบถึงความต้องการที่แท้จริงในปัจจุบัน รวมถึงมีการบันทึกถึงการเปลี่ยนแปลงเกิดขึ้นที่กับความต้องการซอฟต์แวร์ โดยที่การจัดการความต้องการคือ กระบวนการที่ใช้สำหรับจัดการกับการเปลี่ยนแปลงความต้องการของซอฟต์แวร์อย่างมีระบบ โดยแนวคิดที่เกี่ยวกับความต้องการที่ใช้นำมาใช้เพื่อช่วยในการจัดการความต้องการคือ

ก. ความสามารถในการตามรอยความต้องการ (Requirements traceability) [1] คือการเชื่อมโยงหรือการระบุความสัมพันธ์ระหว่างเอนทิตี 2 เอนทิตี ซึ่งการตามรอยความต้องการนี้จะช่วยในการตรวจสอบซอฟต์แวร์ว่าได้ถูกพัฒนาตามสิ่งที่ควรจะเป็นหรือไม่ ดังตัวอย่างในรูปที่ 2.1 เห็นได้ว่ามีการเชื่อมโยงระหว่างความต้องการของลูกค้าก็จะมีความสัมพันธ์กับข้อกำหนดความต้องการซอฟต์แวร์ เพื่อตรวจสอบว่าไม่มีหน้าที่การทำงานของข้อกำหนดความต้องการซอฟต์แวร์ที่เกินความต้องการของลูกค้าและหน้าที่การทำงานของข้อกำหนดความต้องการซอฟต์แวร์ครบถ้วน

ตามความต้องการของลูกค้า และมีการเชื่อมโยงความสัมพันธ์ระหว่างข้อกำหนดความต้องการซอฟต์แวร์กับส่วนการออกแบบ ส่วนการเขียนโปรแกรมและส่วนการทดสอบซอฟต์แวร์ เพื่อตรวจสอบว่าข้อกำหนดความต้องการทั้งหมดได้มีการออกแบบ การเขียนโปรแกรม และการออกแบบกรณีทดสอบเพื่อทดสอบข้อกำหนดความต้องการของระบบครบถ้วนทั้งหมด นอกจากนั้นความสามารถในการตามรอยความต้องการสามารถนำมาใช้ประโยชน์ในการตรวจสอบผลกระทบของการเปลี่ยนแปลงความต้องการที่เกิดขึ้น โดยเมื่อมีการเปลี่ยนแปลงส่วนใดส่วนหนึ่งเกิดขึ้น ก็จำเป็นต้องมีการตรวจสอบว่าส่วนอื่นที่มีความสัมพันธ์กับส่วนที่เปลี่ยนแปลงนั้นว่าจะได้รับผลกระทบจากการเปลี่ยนแปลงด้วยหรือไม่ เช่น เมื่อความต้องการของลูกค้าเปลี่ยนแปลง ข้อกำหนดความต้องการซอฟต์แวร์ก็อาจต้องเปลี่ยนแปลงตามความต้องการของลูกค้าที่เปลี่ยนแปลงไป ซึ่งทำให้ต้องมีการตรวจสอบว่ากรณีทดสอบที่ได้ออกแบบสำหรับทดสอบความต้องการซอฟต์แวร์นั้นจำเป็นต้องมีการเปลี่ยนแปลงตามไปด้วยหรือไม่

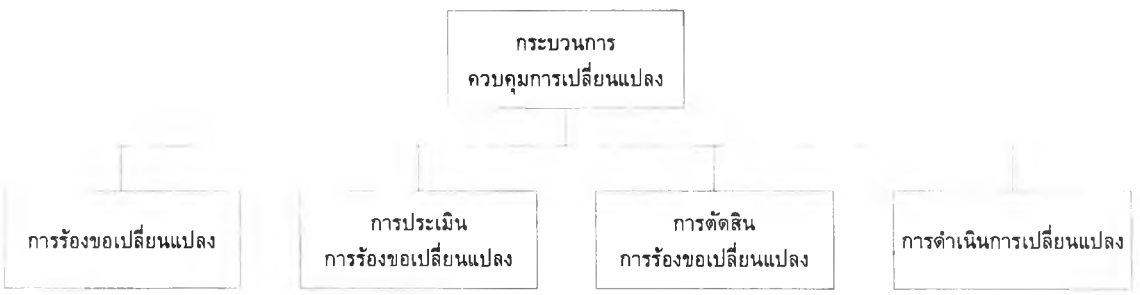


รูปที่ 2.1 ความสัมพันธ์ของความต้องการกับส่วนอื่น

ข. การเก็บข้อมูลการเปลี่ยนแปลงที่เกิดขึ้น (Change history) คือการเก็บข้อมูลต่าง ๆ ที่เกี่ยวกับการเปลี่ยนแปลงที่เกิดขึ้นกับความต้องการภายในโครงการ โดยในโครงการพัฒนาซอฟต์แวร์ควรมีการบันทึกว่าการเปลี่ยนแปลงเกิดขึ้นด้วยสาเหตุใด เกิดการเปลี่ยนแปลงเมื่อใด และใครเป็นผู้ทำการอนุมัติการเปลี่ยนแปลงที่เกิดขึ้น

2.1.2 กระบวนการควบคุมการเปลี่ยนแปลงของซอฟต์แวร์ (Change control procedure) [3]

กระบวนการควบคุมการเปลี่ยนแปลงเป็นกระบวนการที่เกี่ยวข้องกับการทบทวนและอนุมัติเกี่ยวกับการร้องขอเปลี่ยนแปลงที่เกิดขึ้น โดยประกอบด้วยกระบวนการร้องขอเปลี่ยนแปลง กระบวนการประเมินการร้องขอเปลี่ยนแปลง กระบวนการตัดสินการร้องขอเปลี่ยนแปลง กระบวนการดำเนินการเปลี่ยนแปลงอย่างมีระบบ โดยกระบวนการควบคุมการเปลี่ยนแปลงใช้สำหรับการเปลี่ยนแปลงที่เกิดขึ้นหลังจากที่ส่วนที่ต้องการทำการเปลี่ยนแปลงได้กำหนดเป็นเบสไลน์ (Baselines) แล้ว โดยที่เบสไลน์ คือ ข้อกำหนดที่ได้รับการตรวจสอบและได้รับการเห็นชอบอย่างเป็นทางการ เพื่อใช้สำหรับเป็นพื้นฐานในการพัฒนาต่อไป โดยกระบวนการเปลี่ยนแปลงประกอบด้วยกิจกรรมต่างๆ ดังรูปที่ 2.2 โดยสามารถอธิบายสำหรับรายละเอียดสำหรับแต่ละกิจกรรมได้ดังนี้คือ



รูปที่ 2.2 กิจกรรมของกระบวนการควบคุมการเปลี่ยนแปลง

การร้องขอเปลี่ยนแปลง คือ การที่ผู้ร้องขอเปลี่ยนแปลงกำหนดรายละเอียดเกี่ยวกับการเปลี่ยนแปลงในเอกสารร้องขอเปลี่ยนแปลง (Change Request Form) ที่เป็นทางการ โดยเอกสารร้องขอเปลี่ยนแปลงมีลักษณะดังรูปที่ 2.3 โดยที่จะประกอบด้วยข้อมูลสิ่งที่ต้องการให้มีการเปลี่ยนแปลง ชื่อและหน่วยงานของผู้ร้องขอเปลี่ยนแปลง วันที่ทำการร้องขอเปลี่ยนแปลง ความเร่งด่วนของการเปลี่ยนแปลงในความคิดเห็นของผู้ร้องขอ ความจำเป็นสำหรับการเปลี่ยนแปลง คำอธิบายของการร้องขอเปลี่ยนแปลง โดยในการร้องขอเปลี่ยนแปลงจะมีกระบวนการตรวจสอบความถูกต้องของการร้องขอเปลี่ยนแปลงซึ่งเป็นกระบวนการในการตรวจสอบในความชัดเจน ความครบถ้วน ความถูกต้องของการร้องขอเปลี่ยนแปลงที่เสนอเข้ามา รวมถึงมีการตรวจสอบว่าเป็นการร้องขอเปลี่ยนแปลงที่เกิดขึ้นนั้นเกิดจากความเข้าใจผิดของผู้ร้องขอเปลี่ยนแปลงหรือเป็นการร้องเปลี่ยนแปลงที่ซ้ำซ้อนกับการร้องขอเปลี่ยนแปลงที่เคยมีอยู่ก่อนหน้านี้หรือไม่ ซึ่งถ้าผลการตรวจสอบพบว่าการร้องขอเปลี่ยนแปลงไม่สมบูรณ์ ไม่ครบถ้วน การร้องขอเปลี่ยนแปลงที่เกิดขึ้นจะถูกปฏิเสธ ซึ่งเหตุผลสำหรับการปฏิเสธการร้องขอเปลี่ยนแปลงนั้นจะถูกส่งไปยังผู้ร้องขอเปลี่ยนแปลง

Change Request Form

CRF No:	Request Date :	Urgent :
Title:		
Description of Change:		
Reason of Change:		
Originator:		

Software Requirement Specification Change Proposed:
Requirement:
Description:
Comment:
Action:

Comment:	
Action:	Defer:
Action Date time:	
Board:	

รูปที่ 2.3 เอกสารร้องขอเปลี่ยนแปลง

การประเมินและตัดสินการร้องขอเปลี่ยนแปลง คือ การประเมินการร้องขอเปลี่ยนแปลงที่ได้รับการตรวจสอบแล้วว่าถ้าการร้องขอเปลี่ยนแปลงได้รับการอนุมัติการเปลี่ยนแปลงจะมีผลกระทบกับสิ่งใดบ้าง โดยการประเมินและการตัดสินใจว่าการร้องขอเปลี่ยนแปลงที่เกิดขึ้นนั้นสมควรที่จะอนุมัติให้มีการเปลี่ยนแปลงหรือไม่เป็นหน้าที่ของผู้ควบคุมการเปลี่ยนแปลง ซึ่งอาจเป็นบุคคลเพียงคนเดียวหรือคณะกรรมการที่เป็นบุคคลจากส่วนต่างๆ โดยผลการตัดสินการร้องขอเปลี่ยนแปลงจะมีผลการร้องขอเปลี่ยนแปลงได้ 3 รูปแบบคือ การอนุมัติให้การดำเนินการเปลี่ยนแปลงสามารถดำเนินงานได้ การปฏิเสธการเปลี่ยนแปลงหรือการไม่อนุมัติให้การเปลี่ยนแปลงเกิดขึ้น การเลื่อนการตัดสินการร้องขอเปลี่ยนแปลงออกไปก่อน เพื่อรอข้อมูลอื่นๆ ที่จำเป็นสำหรับการตัดสินใจ

การดำเนินการเปลี่ยนแปลง คือ กระบวนดำเนินการเปลี่ยนแปลงตามที่ได้รับการอนุมัติให้เปลี่ยนแปลงโดยประกอบด้วยการนำสิ่งที่ต้องการเปลี่ยนแปลงออกจากเบสไลน์ (Check out) โดยสิ่งที่ถูกนำออกจากเบสไลน์นั้นจะถูกป้องกันไม่ให้มีการเปลี่ยนแปลงเกิดขึ้น เพื่อป้องกันไม่มีการขัดแย้งของการเปลี่ยนแปลงเกิดขึ้น ซึ่งบุคคลที่ได้รับกำหนดให้ดำเนินการเปลี่ยนแปลงก็จะ

ดำเนินการเปลี่ยนแปลงตามคำแนะนำของผู้ควบคุมการเปลี่ยนแปลงและเมื่อการเปลี่ยนแปลงได้รับตรวจสอบว่าการเปลี่ยนแปลงที่เกิดขึ้นนั้นถูกต้อง สิ่งที่ได้ทำการเปลี่ยนแปลงอย่างถูกต้องก็จะถูกนำเข้ากลับเป็นเบสไลน์

2.1.3 เอกสารแนะนำวิธีการสร้างเอกสารข้อกำหนดความต้องการซอฟต์แวร์ (IEEE Std. 830-1998) [4]

เอกสารแนะนำวิธีการสร้างเอกสารข้อกำหนดความต้องการซอฟต์แวร์เป็นเอกสารที่แนะนำถึงกระบวนการในการสร้างเอกสารข้อกำหนดความต้องการและเนื้อหาของเอกสารข้อกำหนดความต้องการซอฟต์แวร์ โดยเอกสารแนะนำวิธีการสร้างเอกสารข้อกำหนดความต้องการซอฟต์แวร์ได้แสดงถึงหัวข้อของเนื้อหาภายในเอกสารข้อกำหนดความต้องการซอฟต์แวร์ที่ดีว่าควรจะประกอบด้วยส่วนประกอบของเอกสารความต้องการซอฟต์แวร์ดังนี้คือ

ก. บทนำ (Introduction) เป็นส่วนอธิบายภาพโดยรวมของเอกสารข้อกำหนดความต้องการซอฟต์แวร์ โดยส่วนนี้ประกอบด้วยส่วนย่อยดังนี้คือ

(1) จุดประสงค์ (Purpose) เป็นส่วนอธิบายจุดประสงค์และบุคคลที่เป็นกลุ่มบุคคลที่ใช้งานเอกสารข้อกำหนดความต้องการซอฟต์แวร์

(2) ขอบเขต (Scope) เป็นส่วนอธิบายขอบเขตของซอฟต์แวร์ว่าซอฟต์แวร์สามารถทำสิ่งใดบ้าง รวมถึงอธิบายถึงความสัมพันธ์กับเอกสารข้อกำหนดความต้องการที่อยู่ในระดับสูงกว่าว่าสัมพันธ์กันอย่างไร

(3) คำนิยาม ตัวย่อ อักษรย่อ (Definitions Acronyms and Abbreviations) เป็นส่วนอธิบายถึงคำนิยามของศัพท์ คำย่อต่างๆ ที่ใช้ในเอกสารข้อกำหนดความต้องการเพื่อผู้ใช้มีความเข้าใจถึงความหมายที่ถูกต้อง

(4) เอกสารอ้างอิง (References) เป็นส่วนแสดงถึงเอกสารทั้งหมดที่ถูกอ้างอิงถึงในเอกสารข้อกำหนดความต้องการซอฟต์แวร์นี้ โดยมีการกำหนดเอกสารโดยระบุถึงชื่อเอกสาร หมายเลขเอกสาร วันที่พิมพ์ รวมถึงสำนักพิมพ์ที่พิมพ์ของเอกสารที่ถูกอ้างอิงถึง

(5) รายละเอียดโดยรวม (Overview) เป็นส่วนอธิบายถึงเนื้อหาและรูปแบบของเอกสารข้อกำหนดความต้องการซอฟต์แวร์ในส่วนต่อไปของเอกสารข้อกำหนดความต้องการซอฟต์แวร์ว่ามีการจัดรูปแบบเอกสารอย่างไร

ข. ส่วนอธิบายภาพรวมทั้งหมด (Overall description) เป็นส่วนอธิบายถึงปัจจัยที่มีผลกระทบต่อซอฟต์แวร์และความต้องการของซอฟต์แวร์ รวมถึงอธิบายความต้องการโดยรวมของซอฟต์แวร์เพื่อเป็นพื้นฐานในการทำความเข้าใจถึงความต้องการซอฟต์แวร์ซึ่งจะระบุในส่วนความต้องการซอฟต์แวร์ทั้งหมดในระดับละเอียดได้ง่าย

(1) ภาพลักษณ์ของผลิตภัณฑ์ (Product perspective) เป็นส่วนอธิบายถึงความสัมพันธ์ของซอฟต์แวร์ที่สร้างขึ้นว่ามีความสัมพันธ์กับส่วนอื่นอย่างไรบ้าง

(2) หน้าที่ของผลิตภัณฑ์ (Product functions) เป็นส่วนอธิบายถึงหน้าที่หลักๆ ที่ซอฟต์แวร์จะสามารถปฏิบัติได้

(3) คุณสมบัติของผู้ใช้ซอฟต์แวร์ (User characteristics) เป็นส่วนอธิบายถึงคุณลักษณะของผู้ที่ใช้งานซอฟต์แวร์ เช่น ระดับการศึกษา ประสบการณ์ ความชำนาญด้านเทคนิค

(4) ข้อบังคับ (Constraints) เป็นส่วนอธิบายถึงสิ่งที่เป็นข้อบังคับทั้งหมดสำหรับการพัฒนาซอฟต์แวร์

(5) ส่วนแสดงปัจจัยที่มีความเกี่ยวข้องและผลกระทบกับข้อกำหนดความต้องการ (Assumptions and dependencies) เป็นส่วนที่แสดงปัจจัยต่างๆ ที่มีผลกระทบต่อความต้องการที่ได้กำหนดไว้ในเอกสารข้อกำหนดความต้องการซอฟต์แวร์

ค. ความต้องการของซอฟต์แวร์ทั้งหมดในระดับละเอียด (Specific requirement) เป็นส่วนอธิบายถึงความต้องการของซอฟต์แวร์ในระดับที่ละเอียด เพื่อให้ผู้ออกแบบระบบสามารถทำการออกแบบซอฟต์แวร์ รวมถึงผู้ทดสอบสามารถออกแบบกรณีทดสอบและทดสอบซอฟต์แวร์ได้ตามข้อกำหนดความต้องการซอฟต์แวร์ที่ได้กำหนด โดยแต่ละข้อกำหนดความต้องการซอฟต์แวร์ต้องกำหนดถึงสิ่งที่นำเข้าสำหรับซอฟต์แวร์ สิ่งที่เป็นผลลัพธ์ที่ได้จากซอฟต์แวร์ และหน้าที่ทั้งหมดที่ถูกปฏิบัติโดยซอฟต์แวร์

ง. เอกสารแนบท้าย (Appendixes) แสดงข้อมูลซึ่งถูกอ้างอิงภายในส่วนหลักของเอกสารข้อกำหนดความต้องการซอฟต์แวร์

2.1.4 แนวคิดเกี่ยวกับการทดสอบ [5]

การทดสอบที่ใช้สำหรับการทดสอบซอฟต์แวร์นั้นสามารถแบ่งระดับของการทดสอบออกได้เป็น 3 ระดับ คือ การทดสอบระดับหน่วย การทดสอบการรวม และการทดสอบระบบ โดยที่รายละเอียดเกี่ยวกับการทดสอบแต่ละระดับสามารถเขียนเป็นรายละเอียดได้ดังนี้คือ

ก. การทดสอบระดับหน่วย เป็นการทดสอบในการตรวจสอบระดับโมดูลของซอฟต์แวร์ โดยแต่ละโมดูลถูกทดสอบแยกจากกันทำให้ โดยแต่ละโมดูลถูกทดสอบแยกจากกัน

ข. การทดสอบการรวม เป็นการทดสอบโดยการรวมโมดูลเข้าด้วยกัน เพื่อทำการตรวจสอบว่าจะไม่เกิดความผิดพลาดเมื่อการรวมโมดูล โดยมีวิธีการรวมจากบนลงล่าง (Top-down approach) การรวมจากล่างขึ้นบน (Bottom-up approach) การรวมแบบแซนด์วิช (Sandwich approach) การรวมแบบบิ๊กแบง (Big-bang approach)

ค. การทดสอบระบบ (System testing) เป็นการทดสอบที่เกิดจากการที่ได้นำเอาทุกๆ โมดูลมารวมเข้าด้วยกันแล้ว โดยการทดสอบระดับนี้จะเน้นไปที่การทดสอบข้อกำหนดและความต้องการที่กำหนดไว้ ดังนั้นการออกแบบกรณีทดสอบเป็นการออกแบบเพื่อใช้กรณีทดสอบที่ได้ทำการออกแบบมาตรวจสอบว่าระบบที่พัฒนาขึ้นทำตรงตามข้อกำหนดและความต้องการที่กำหนดไว้หรือไม่ โดยการทดสอบระบบสามารถแบ่งออกได้ดังนี้คือ

(1) การทดสอบตามหน้าที่ (Functional Testing) เป็นการทดสอบระบบโดยไม่สนใจโครงสร้างหรือกลไกภายในของระบบ โดยจะเป็นการเน้นพิจารณาว่าผลลัพธ์ที่ได้จากระบบนั้นตรงกับความต้องการและข้อกำหนดที่ได้กำหนดไว้หรือไม่

(2) การทดสอบอื่นๆ (Non-Functional Testing) ซึ่งจะประกอบด้วย

(ก) การทดสอบความกดดัน (Stress Testing) โดยทดสอบระบบภายใต้สภาพการใช้งานที่สูง เพื่อให้ทราบถึงความสามารถสูงสุดของระบบภายใต้ทรัพยากรที่กำหนด

(ข) การทดสอบประสิทธิภาพ (Performance Testing) เพื่อทดสอบว่าระบบมีประสิทธิภาพตามที่ได้ทำการระบุได้ โดยตรวจสอบได้จากเวลาตอบสนอง (Response Time) ปริมาณงานที่ได้ (Throughput) การใช้งานหน่วยประมวลผลกลาง (CPU Utilization) เป็นต้น

(ค) การทดสอบส่วนหลัง (Background Testing) เพื่อแสดงความสามารถของระบบในกรณีที่รองรับรายการมากกว่า 1 รายการในเวลาเดียวกัน

(ง) การทดสอบโครงแบบ (Configuration Testing) เพื่อต้องการทราบข้อกำหนดของฮาร์ดแวร์ที่เหมาะสม

(จ) การทดสอบการกู้ระบบ (Recovery Testing) เพื่อแสดงว่าระบบสามารถกู้ข้อมูลกลับคืนมาได้หากระบบเกิดความเสียหาย

(ฉ) การทดสอบความปลอดภัย (Security Testing) เพื่อแสดงว่ากลไกในการรักษาความปลอดภัยของระบบมีประสิทธิภาพเพียงพอที่จะป้องกันระบบจากการลักลอบใช้งาน

2.1.5 ตารางที่ใช้เพื่อการตรวจสอบความต้องการ (Requirements Validation Matrix) [6]

ตารางที่ใช้เพื่อการตรวจสอบความต้องการเป็นวิธีการเพื่อช่วยผู้ออกแบบกรณีทดสอบ (Test Case Designer) ตรวจสอบว่าได้มีการออกแบบกรณีทดสอบที่ใช้สำหรับทดสอบข้อกำหนดความต้องการแต่ละข้อที่ได้กำหนดไว้อย่างครบถ้วนหรือไม่ โดยตารางที่ใช้เพื่อการตรวจสอบความต้องการนั้นจะแสดงความสัมพันธ์ระหว่างข้อกำหนดความต้องการซึ่งถูกแสดงทางด้านซ้ายกับกรณีทดสอบที่

ใช้สำหรับทดสอบข้อกำหนดความต้องการซอฟต์แวร์แต่ละข้อซึ่งถูกแสดงทางด้านขวา โดยที่การแสดงความสัมพันธ์ระหว่างข้อกำหนดความต้องการและกรณีทดสอบสามารถแสดงได้ 2 วิธีคือ การแสดงหมายเลขของกรณีทดสอบที่ใช้สำหรับทดสอบข้อกำหนดความต้องการซอฟต์แวร์ดังตารางที่ 2.1 กับการทำเครื่องหมายเพื่อแสดงความสัมพันธ์ระหว่างข้อกำหนดความต้องการกับกรณีทดสอบดังตารางที่ 2.2 ซึ่งตารางทั้ง 2 ตารางแสดงถึงความสัมพันธ์ระหว่างข้อกำหนดความต้องการและกรณีทดสอบที่เหมือนกัน ซึ่งสามารถอธิบายถึงความสัมพันธ์ได้ว่ากรณีทดสอบหมายเลขที่ 87 88 และ 102 ต้องได้รับการทดสอบว่าผ่านเกณฑ์เพื่อแสดงว่าการทดสอบข้อกำหนดความต้องการซอฟต์แวร์ที่ 1 นั้นผ่านเกณฑ์ และกรณีทดสอบที่ 86 ถึง 88 และ 102 ต้องได้รับการทดสอบว่าผ่านเกณฑ์เพื่อ แสดงว่าการทดสอบข้อกำหนดความต้องการซอฟต์แวร์ที่ 2 นั้นผ่านเกณฑ์

ตารางที่ 2.1 ตัวอย่างของตารางที่ใช้เพื่อการตรวจสอบความต้องการรูปแบบที่ 1

ความต้องการ	กรณีทดสอบ
1. Provide the capability to submit a sales order for a single item	87 88 102
2. Provide the capability to submit sales order with multiple items and multiple quantities.	86-88 102

ตารางที่ 2.2 ตัวอย่างของตารางที่ใช้เพื่อการตรวจสอบความต้องการรูปแบบที่ 2

ความต้องการ	กรณีทดสอบ				
	86	87	88	...	102
1. Provide the capability to submit a sales order for a single item		✓	✓	...	✓
2. Provide the capability to submit sales order with multiple items and multiple quantities	✓	✓	✓	...	✓

นอกจากนั้นตารางที่ใช้เพื่อการตรวจสอบความต้องการยังสามารถนำมาใช้ประโยชน์ได้เมื่อมีการเปลี่ยนแปลงเกิดขึ้น เช่น จากตารางที่ใช้เพื่อการตรวจสอบความต้องการในตารางที่ 2.1 และ 2.2 เมื่อมีการเปลี่ยนแปลงเกิดขึ้นกับความต้องการที่ 1 ผู้ออกแบบกรณีทดสอบก็ควรมีการตรวจสอบถึงกรณีทดสอบที่มีความสัมพันธ์กับความต้องการที่ 1 ซึ่งได้แก่กรณีทดสอบหมายเลข 87 88 และ 102 ว่ากรณีทดสอบนั้นต้องมีการเปลี่ยนแปลงเพื่อใช้ทดสอบข้อกำหนดความต้องการซอฟต์แวร์ที่ถูกเปลี่ยนแปลงไปหรือไม่

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 โปรแกรม RequisitePro[7]

โปรแกรมนี้ได้รับการพัฒนาโดยบริษัท Rational Software Corporation เป็นเครื่องมือใช้สำหรับการเก็บรวบรวมความต้องการต่างๆ โดยการเพิ่มความต้องการนั้นสามารถทำได้โดยวิธีการนำเข้าข้อความที่เป็นความต้องการที่อยู่ในรูปของเอกสารเวิร์ดหรือเพิ่มความต้องการภายในโปรแกรมโดยตรงได้ นอกจากนั้นผู้ใช้สามารถระบุการตามรอยความต้องการโดยระบุถึงความสัมพันธ์ระหว่างความต้องการต่างๆ ได้ ซึ่งการตามรอยความต้องการจะถูกนำไปใช้ประโยชน์เมื่อมีการเปลี่ยนแปลงขึ้น โดยเครื่องมือนี้จะช่วยในการแสดงความสัมพันธ์ระหว่างความต้องการว่ามีความสัมพันธ์กันอย่างไร ซึ่งทำให้ทราบถึงความต้องการที่จะอาจได้รับผลกระทบจากการเปลี่ยนแปลง เพื่อให้ผู้ใช้ตรวจสอบว่าการเปลี่ยนแปลงความต้องการจะมีผลกระทบกับความต้องการอื่นความต้องการใด

เครื่องมือนี้ไม่มีส่วนรับคำร้องเปลี่ยนแปลง และส่วนการปรับเปลี่ยนกรณีทดสอบเพื่อให้ตรงกับความต้องการที่เปลี่ยนแปลงไป

2.2.2 โปรแกรม TestDirector[8]

โปรแกรมนี้ได้รับการพัฒนาโดยบริษัท Mercury Interactive Corporation เป็นเครื่องมือช่วยในการทำการทดสอบโดยจะแยกการทำงานออกเป็น 4 ส่วนคือ

ก. ส่วนจัดการด้านความต้องการ (Requirements Manager) เป็นส่วนที่กำหนดว่ามีความต้องการใดบ้างที่จะทำการทดสอบ และสามารถที่จะทำการเชื่อมกรณีทดสอบกับความต้องการเหล่านั้นได้

ข. ส่วนจัดการแผนการทดสอบ (Test Plan Manager) เป็นส่วนที่ทำการสร้างแผนการทดสอบโดยอ้างอิงมาจากความต้องการที่จะทดสอบ แล้วทำการสร้างการทดสอบว่าความต้องการแต่ละอันจะทำการทดสอบอย่างไร โดยอาจเลือกวิธีการทดสอบแบบปกติหรืออัตโนมัติได้

ค. ส่วนจัดการทดสอบ (Test Lab Manager) เป็นส่วนที่ทำหน้าที่ในการรวมกลุ่มกรณีทดสอบที่จะใช้ทำการทดสอบร่วมกัน โดยจะเป็นส่วนที่ใส่ผลการทดสอบและซึ่งถ้าเป็นการทดสอบแบบอัตโนมัติสามารถทำการกำหนดวันเวลาที่ทดสอบได้ที่ส่วนนี้ด้วย

ง. ส่วนจัดการความผิดพลาด (Defect Manager) โดยใช้ในการเก็บ วิเคราะห์กำหนดความผิดพลาด รวมถึงติดตามความผิดพลาดที่เกิดขึ้นระหว่างการทดสอบ โดยสามารถรายงานความผิดพลาดไปยังใครก็ตามที่เกี่ยวข้องกับการทดสอบได้

เครื่องมือนี้ไม่มีส่วนรับคำร้องเปลี่ยนแปลง รวมถึงขาดการช่วยเหลือในการเปลี่ยนแปลงกรณีทดสอบ รวมถึงการบันทึกและออกเอกสารข้อกำหนดความต้องการให้ตรงกับมาตรฐาน IEEE Std. 830-1998 ได้

2.2.3 เครื่องมือสำหรับจัดการกระบวนการวางแผนการทดสอบซอฟต์แวร์[9]

เครื่องมือนี้เป็นวิทยานิพนธ์ของภาควิชาวิศวกรรมคอมพิวเตอร์ สาขาวิทยาศาสตร์คอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ประจำปีการศึกษา 2543 ซึ่งถูกพัฒนาโดยนายสันต์ทศน์ สุริยันต์ ซึ่งเป็นเครื่องมือสำหรับจัดการกระบวนการวางแผนการทดสอบซอฟต์แวร์ใน 3 ระดับ ได้แก่ การทดสอบระดับหน่วย การทดสอบการรวม และการทดสอบระบบ เครื่องมือดังกล่าวสามารถเก็บรวบรวมข้อมูลที่เกิดจากการทดสอบโดยอ้างอิงกับมาตรฐานของเอกสารที่ใช้ในการทดสอบ IEEE Std. 829-1998 รวมทั้งผู้ใช้สามารถปรับเปลี่ยนแม่แบบของเอกสารได้ นอกจากนั้นเครื่องมือนี้ยังช่วยจัดกำหนดการโดยใช้แผนภูมิแกนต์ จัดสรรทรัพยากร กำหนดความรับผิดชอบ และออกรายงานเกี่ยวกับการทดสอบ และช่วยติดตามสถานะของการทดสอบได้อีกด้วย

แต่เครื่องมือนี้ยังขาดความสามารถในการที่จะช่วยจัดการแก้ไขกับกรณีทดสอบ ในกรณีเมื่อมีการเปลี่ยนแปลงความต้องการของผู้ใช้เกิดขึ้นได้