ความรู้เกี่ยวกับความต้องการและการจัดการความต้องการทางด้านซอฟต์แวร์

Knowledge and Management of Software Requirements

บุญประเสริฐ สุรักษ์รัตนสกุล[1*]

[1]ผู้ช่วยศาสตราจารย์ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ 10520

บทคัดย่อ

ในกระบวนการพัฒนาซอฟต์แวร์ ความต้องการจากผู้ใช้งานหรือผู้ที่มีส่วนเกี่ยวข้องจะเป็นข้อมูลเริ่มต้นในกระบวนการทางวิศวกรรมซอฟต์แวร์ ความต้องการที่ค้นหา ค้นพบและเก็บเกี่ยวมาได้นั้น จะถูกนำมากำหนด จำแนกและกลั่นกรอง ทำให้มีคุณภาพและจัดเก็บอย่างเป็นระบบตามหลักของการจัดการความต้องการ เพื่อให้ความต้องการมีความถูกต้องและมีความพร้อมใช้งานในกระบวนการพัฒนาขั้นตอนถัดไป ซึ่งเรียกกระบวนการนี้ว่า วิศวกรรมความต้องการทางด้านซอฟต์แวร์ หากการได้มาของความต้องการขาดคุณภาพ มีข้อผิดพลาดหรือถูกจัดการอย่างไม่เป็นระบบ จะส่งผลกระทบให้กระบวนการพัฒนาในขั้นตอนถัดไปล้มเหลว เป็นผลให้ผลิตภัณฑ์ทางซอฟต์แวร์ที่ได้ด้อยคุณภาพ ไม่ตรงตามความต้องการของผู้ใช้งาน ผู้ว่าจ้างและผู้ที่มีส่วนเกี่ยวข้อง ด้วยเหตุนี้ บทความจึงได้นำเสนอภาพรวมและนิยามเบื้องต้นเกี่ยวกับความต้องการและการจัดการความต้องการทางด้านซอฟต์แวร์ ระดับและความแตกต่างของความต้องการ คุณภาพของความต้องการและการทำความเข้าใจในมิติคุณภาพ ปัญหาที่มักเกิดขึ้นเกี่ยวกับการเก็บเกี่ยวความต้องการและการตระหนักถึงผลกระทบและความเสียหายเมื่อความต้องการเกิดข้อผิดพลาดหรือด้อยคุณภาพ ปัจจัยที่ส่งผลต่อการจัดการความต้องการให้บรรลุตรงตามเป้าหมาย กฎ1-10-100 และท้ายสุดกล่าวถึงบทบาทและเป้าหมายของทีมค้นหา เก็บเกี่ยวและจัดการความต้องการทางด้านซอฟต์แวร์

Abstract

In software engineering, requirements from customers, users, and stakeholder are initial information for software development process. The requirements elicited and gained are defined, classified, refined, and stored systematically according to needs management principles. Therefore, the requirements would be correct and ready to be used in the next step of development process. This process is called Software Requirements Engineering. If the gained software requirements are incorrect, lack quality, or are managed unsystematically, they could lead to project failure and low-quality software product which cannot meet the needs of users and stakeholders. For these reasons, this article proposed an overview of software requirements and an approach for software requirements management, level of software requirements, quality of software requirements and understanding of quality aspect, common problems of requirements elicitation, and awareness of impact and damage due to incorrect or low quality requirements. Success factors of software requirements management and 1-10-100 rules are also discussed. Finally, the article addresses roles and goals of software requirements management team.

คำสำคัญ : วิศวกรรมความต้องการ ความต้องการทางด้านซอฟต์แวร์ การจัดการความต้องการทางด้านซอฟต์แวร์ คุณภาพของความต้องการทางด้านซอฟต์แวร์ มาตรฐานในการจัดทำความต้องการทางซอฟท์แวร์

Keywords : Requirements Engineering, Software Requirements, Software Requirements Management, Software Requirements Quality, Standard of Software Requirement Development

**ผู้นิพนธ์ประสานงานไปรษณีย์อิเล็กทรอนิกส์ boonprasert@it.kmitl.ac.th โทร. 0 2723 4976*

1. บทนำ

1.1 นิยามของความต้องการทางด้านซอฟต์แวร์

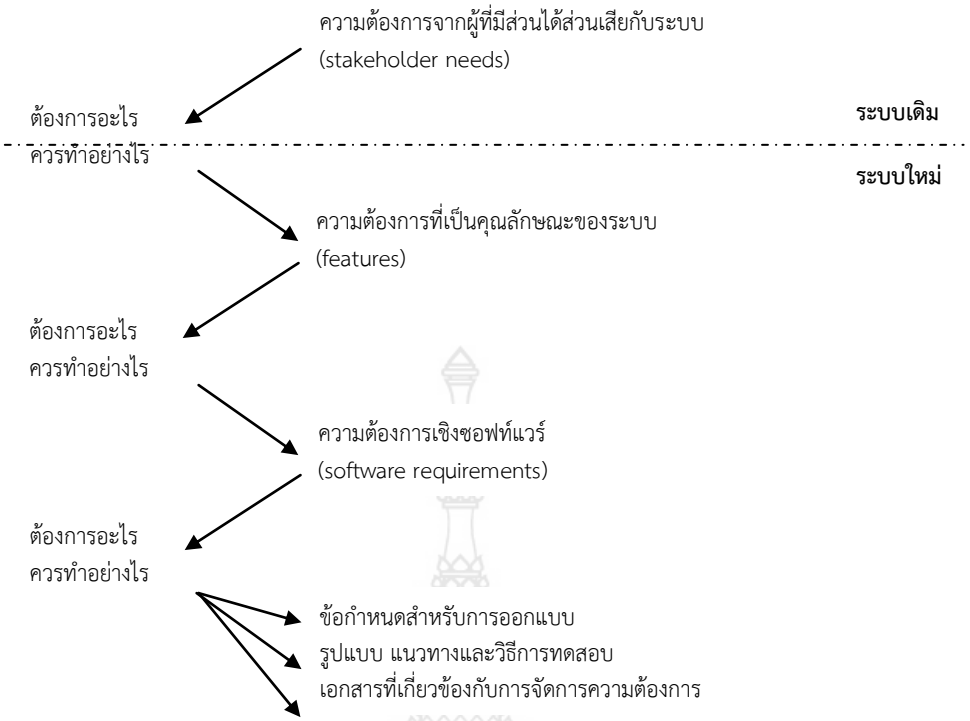
ในบริบททางด้านวิศวกรรมความต้องการ ความต้องการ(requirements) หมายถึง เงื่อนไขหรือข้อกำหนดที่ระบบจำเป็นต้องปฏิบัติตามหรือพึงกระทำ เพื่อตอบสนองให้ผู้ร้องขอหรือผู้กำหนดเงื่อนไขมีความพึงพอใจสูงสุด (Kotonya & Sommerville, 1998: 68) โดยทั่วไปแล้ว เมื่อมีการพัฒนาหรือปรับปรุงระบบซอฟต์แวร์ ความต้องการในระบบใหม่มักมีจำนวนมากและหลากหลาย อาจมีแหล่งที่มาจากบุคคลหลากหลายบทบาท รวมทั้งอาจมีเงื่อนไขหรือข้อจำกัดต่างๆ เช่น ระยะเวลา งบประมาณ และการใช้ทรัพยากรที่มีอยู่อย่างจำกัด ด้วยเหตุนี้ ความต้องการจึงจำเป็นต้องมี การบริหารจัดการความต้องการ(requirements management) ซึ่งเป็นกระบวนการหรือวิธีการ อันประกอบด้วย การค้นหาความต้องการที่แท้จริง การเก็บเกี่ยวความต้องการจากผู้ที่มีส่วนเกี่ยวข้อง การกำหนดชนิดและจัดลำดับชั้นของความต้องการ การจัดทำเอกสารที่เกี่ยวข้องกับความต้องการเพื่อใช้ประกอบกระบวนการพัฒนาระบบ การจัดการความเปลี่ยนแปลงของความต้องการ และอาจรวมถึงการจัดทำและรักษาข้อตกลงร่วมระหว่างผู้ว่าจ้าง ผู้ใช้งานและผู้พัฒนาระบบ (Leffingwell & Widrig, 2004: 98) นอกจากคำนิยามข้างต้นเกี่ยวกับความต้องการและการจัดการความต้องการโดยภาพรวมแล้ว ความต้องการยังสามารถนิยามได้หลายความหมาย ทั้งนี้ขึ้นอยู่กับบริบทตามสภาวะแวดล้อมของผู้นิยามเอง ตัวอย่างเช่น กระบวนการพัฒนาแบบยูนิฟายโปรเซส(Unified Process: UP) ได้นิยามความหมายของความต้องการไว้ว่า ความต้องการ คือ การอธิบายปัจจัยหรือความสามารถที่ทำให้ระบบมีการทำงานที่สอดคล้องในแนวทางเดียวกันกับความพึงประสงค์ของผู้ร้องขอ ซึ่งเป็นแนวทางที่ได้ทำข้อตกลงร่วมกันแบบเป็นทางการที่เป็นลายลักษณ์อักษร (Sommerville Ian, 2010: 128) ขณะที่ในส่วนของยูเอ็มแอล(Unified Modeling Language: UML) ได้นิยามความต้องการไว้ว่า ความต้องการ คือ การกำหนดคุณลักษณะของระบบ ให้ระบบพึงมีหรือมีพฤติกรรมที่ได้กำหนดตามผู้ที่มีส่วนได้ส่วนเสียกับระบบร้องขอ (Booch, Rumbaugh & Jacobson, 1998: 223)

อย่างไรก็ตาม แม้ว่าคำนิยามของความต้องการจะมีหลายความหมายขึ้นกับบริบทแวดล้อมที่กล่าวถึง แต่โดยสรุปแล้วมีความหมายที่ไปในแนวทางเดียวกัน คือ “การอธิบายเกี่ยวกับระบบที่กำลังจะพัฒนาหรือปรับปรุงขึ้น ให้สามารถปฏิบัติงานเพื่อให้ได้ผลลัพธ์ตามที่ต้องการ มากกว่าสนใจว่าระบบจะทำงานอย่างไร หรือกล่าวสั้นๆได้ว่า สนใจในผลลัพธ์ที่ได้ว่าตรงตามสิ่งที่ต้องการหรืออยากได้หรือไม่เท่านั้น” (บุญประเสริฐ สุรักษ์รัตนสกุล, 2551: 2)

ในบริบททางด้านวิศวกรรมความต้องการเชิงซอฟต์แวร์ ก็มีคำนิยามในลักษณะแนวทางเดียวกัน แต่ความต้องการที่นำมาใช้ในกระบวนการพัฒนาซอฟต์แวร์นั้น จะมีการจำแนกเป็นหมวดหมู่ ซึ่งแต่ละรูปแบบ ก็จะมีบทบาทที่แตกต่างกันตามช่วงระยะของการพัฒนาระบบ ตัวอย่างเช่น ความต้องการที่เป็นคุณลักษณะของระบบ(feature) หมายถึง ความต้องการที่ผู้ว่าจ้าง ผู้ใช้งาน และผู้ที่มีส่วนได้ส่วนเสียกับระบบต้องการอย่างแท้จริง ซึ่งมีที่มาหรือสาเหตุจากปัญหาที่พบในระบบเดิมหรือปัญหาที่เกิดจากกระบวนการทางธุรกิจ ความต้องการเชิงซอฟต์แวร์(software requirement) หมายถึง รายละเอียดความต้องการสำหรับการพัฒนาซอฟต์แวร์เพื่อให้ได้ผลลัพธ์ที่ต้องการโดยกระบวนการทางวิศวกรรมซอฟต์แวร์ จะเห็นได้ว่า ความต้องการและการจัดการความต้องการเป็นปัจจัยหนึ่ง ที่ส่งผลต่อความสำเร็จในกระบวนการพัฒนาระบบ ซึ่งความสำเร็จนี้จะบรรลุวัตถุประสงค์ได้ ปัจจัยหนึ่งที่สำคัญ คือ ความต้องการจำเป็นต้องเกิดขึ้นอย่างอิสระ รวมถึงการปฏิบัติที่ตอบสนองผู้ที่มีส่วนเกี่ยวข้องให้ได้มากที่สุด เช่น ผู้ให้ความต้องการมีความเป็นอิสระในการให้ความต้องการของตนเองโดยปราศจากการควบคุมหรือแนะนำ ผู้ที่นำไปปฏิบัติหรือพัฒนาให้เกิดขึ้นมีความเป็นอิสระในวิธีการหรือกระบวนการที่สร้างสรรค์ที่สามารถตอบสนองความต้องการของผู้ให้ความต้องการได้ หากความต้องการใดเกิดขึ้นแบบไม่เป็นอิสระหรือมีการควบคุมแนวทางปฏิบัติ เราเรียกความต้องการนั้นว่า ข้อจำกัด (constraint) ในส่วนของการจัดการความต้องการผู้ที่ทำหน้าที่จัดการความต้องการต้องสร้างความมั่นใจว่า ความต้องการที่ได้มานั้น สามารถทำการกำหนดเป้าหมายหรือจุดสัมฤทธิ์ผลของระบบที่จะพัฒนาให้บรรลุได้อย่างชัดเจน

1.2 ระดับของความต้องการทางด้านซอฟต์แวร์

ความต้องการทางด้านซอฟต์แวร์มีหลายระดับและไม่มีกฎเกณฑ์ในการแบ่งระดับที่ชัดเจน ทั้งนี้ขึ้นอยู่กับแนวคิดการประยุกต์ใช้และความเหมาะสมตามดุลยพินิจของผู้พัฒนา รวมถึงประสบการณ์ของทีมผู้พัฒนาโครงการ และกระบวนการพัฒนาที่นำมาประยุกต์ แต่โดยทั่วไป ความต้องการทางด้านซอฟต์แวร์จะถูกแบ่งออกเป็น 3 ระดับอย่างกว้างๆ ได้แก่ ความต้องการจากผู้ที่มีส่วนได้ส่วนเสียกับระบบ (stakeholder needs) ความต้องการที่เป็นคุณลักษณะของระบบ(features) และความต้องการเชิงซอฟต์แวร์(software requirements) ซึ่งการแบ่งระดับความต้องการนี้ จะใช้แนวคิดแบบ What คือต้องการอะไร และ How แล้วควรทำอย่างไรหรือจะตอบสนองอย่างไร (Rational Software, 2005: 288) ดังแสดงในรูปที่ 1



รูปที่ 1 ระดับของความต้องการทางด้านซอฟต์แวร์

การแบ่งระดับของความต้องการ จะเริ่มจากการที่ผู้ที่มีส่วนเกี่ยวข้องหรือผู้ที่มีส่วนได้ส่วนเสียทำการกำหนดความต้องการ ซึ่งได้มาจากการวิเคราะห์ปัญหาหรือสิ่งที่ต้องการให้เกิดขึ้นตามแผนการทางธุรกิจ ข้อมูลของผู้แข่งขันทางการค้า รายงานข้อผิดพลาดของระบบเดิม หรือข้อเสนอแนะจากผู้เชี่ยวชาญ ซึ่งความต้องการเหล่านี้จะอยู่ในรูปแบบของความต้องการจากผู้ที่มีส่วนได้ส่วนเสียกับระบบ หลังจากที่ได้ความต้องการจากผู้ที่มีส่วนได้ส่วนเสียกับระบบแล้ว ความต้องการจะถูกนำมาใช้ในการกำหนดคุณลักษณะที่พึงมีของระบบเป็นความต้องการที่เป็นคุณลักษณะของระบบใหม่ ซึ่งคุณลักษณะของระบบดังกล่าว ต้องสามารถตอบสนองความต้องการจากผู้ที่มีส่วนได้ส่วนเสียกับระบบได้ โดยคุณลักษณะที่รวบรวมมานั้น จะถูกนำมาวิเคราะห์และคัดกรอง เนื่องจากบางคุณลักษณะสามารถตอบสนองได้โดยไม่ต้องใช้กระบวนการทางซอฟต์แวร์ เช่น การปรับโครงสร้างองค์กร การแก้ไขขั้นตอนการปฏิบัติงาน เป็นต้น ในส่วนของคุณลักษณะที่ตอบสนองโดยกระบวนการทางซอฟต์แวร์ จะถูกกำหนดให้เป็นความต้องการเชิงซอฟต์แวร์ เพื่อใช้ในกระบวนการพัฒนาระบบ ซึ่งอาจอยู่ในรูปแบบของฟังก์ชันการทำงาน(functional requirements) หรือรูปแบบสนับสนุนประกอบในการทำงาน (supplementary requirements) โดยในส่วนความต้องการเชิงซอฟต์แวร์ที่เป็นฟังก์ชันการทำงานนั้น นักพัฒนาระบบสามารถนำความต้องการเหล่านี้มาเป็นข้อกำหนดในการจัดสร้างยูสเคส(use case) สำหรับกระบวนการพัฒนาระบบในระยะถัดไปได้ ซึ่งจำนวนของความต้องการในแต่ละระดับที่กล่าวมาจะอยู่ในรูปแบบที่เป็นทรงพีระมิด เนื่องจากความต้องการจะถูกแจกแจงและลงรายละเอียดมากขึ้นเรื่อยๆเมื่อเข้าสู่ระดับขั้นถัดไป

2. คุณภาพและมิติการจัดการของความต้องการเชิงซอฟต์แวร์

2.1 คุณภาพของความต้องการ

นอกจากการค้นหาและการได้มาของความต้องการแล้ว คุณภาพของความต้องการเชิงซอฟต์แวร์ที่ได้ก็เป็นสิ่งสำคัญ เพราะความต้องการที่มีคุณภาพจะช่วยพัฒนาวิธีการแก้ไขปัญหาของระบบอย่างถูกต้อง และยังมีส่วนช่วยในเรื่องของการเขียนรายละเอียดของความต้องการเชิงซอฟต์แวร์ที่เกิดขึ้นในอนาคตด้วย ซึ่งถูกนำมาใช้เป็นตัวกำหนดจุดบรรลุความสำเร็จของโครงการการพัฒนาระบบ หรืออาจนำมาเป็นข้อตกลงร่วมกันระหว่างผู้ว่าจ้างและผู้พัฒนาระบบ ดังนั้นความต้องการที่นำมาใช้ในการพัฒนาซอฟต์แวร์จึงจำเป็นต้องมีคุณภาพ ซึ่งคุณภาพของความต้องการสามารถอธิบายได้ด้วยคุณสมบัติ ดังนี้

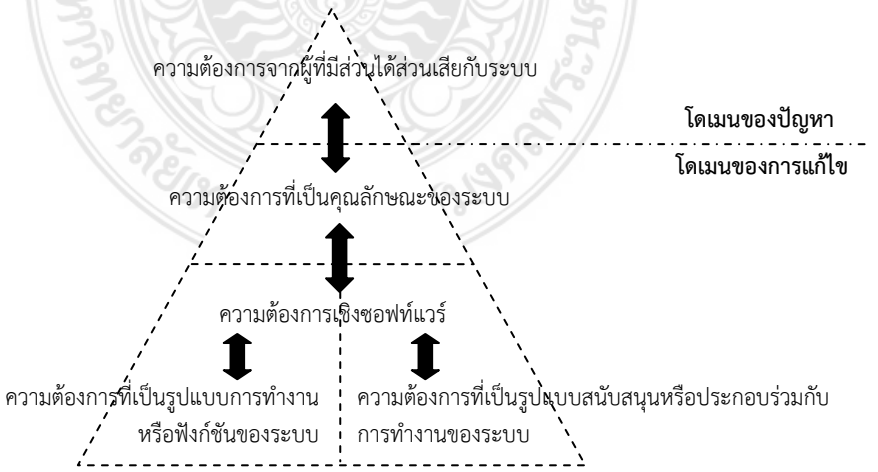
2.1.1 ต้องสามารถจัดลำดับความสำคัญ(priority) และความมีเสถียรภาพ(stability)ได้ เมื่อผู้พัฒนาระบบทำการกำหนดความต้องการ แต่ละความต้องการที่ได้กำหนดขึ้นมานั้น ต้องสามารถให้ลำดับความสำคัญและระบุความมี

เสถียรภาพได้ เพื่อใช้แสดงการวัดระดับความสำคัญหรือความจำเป็นของความต้องการที่สนใจเทียบกับความต้องการที่มีอยู่ทั้งหมด เนื่องจากโดยทั่วไปแล้ว ความต้องการที่เก็บเกี่ยวมาจากผู้ว่าจ้าง ผู้ใช้งานและผู้ที่มีส่วนได้ส่วนเสียกับระบบ มักมีจำนวนมากและไม่สมดุลกับระยะเวลาในการพัฒนาที่มีอย่างจำกัด ดังนั้น ความต้องการจึงจำเป็นต้องมีการคัดสรรและจัดลำดับ เพื่อให้ความต้องการที่มีความสำคัญที่สามารถตอบสนองเป้าหมายของการพัฒนาระบบได้สูงสุด ถูกนำมาพัฒนาได้อย่างครบถ้วนสมบูรณ์ก่อน ในส่วนของความมีเสถียรภาพ ก่อนที่ความต้องการจะถูกนำไปพัฒนาตามกระบวนการพัฒนาระบบ ผู้พัฒนาระบบต้องมั่นใจว่า ความต้องการนั้นต้องไม่เกิดการเปลี่ยนแปลงขึ้นในภายหลัง เนื่องจากผลกระทบของการเปลี่ยนแปลงความต้องการหลังจากที่ได้นำมาพัฒนาระบบไปแล้วนั้น ก่อให้เกิดความเสียหายค่อนข้างมาก เนื่องจากเป็นจุดเริ่มต้นของหลายๆสิ่งในกระบวนการพัฒนา ซึ่งเป็นไปตามกฎ 1-10-100 ที่กล่าวถึงในตอนท้ายของบทความนี้

2.1.2 ต้องสามารถตรวจสอบและพิสูจน์ได้ (verification) ทุกความต้องการเมื่อได้ทำการกำหนดให้เป็นความต้องการเชิงซอฟต์แวร์แล้ว ต้องสามารถทำการตรวจสอบและพิสูจน์ได้ โดยกระบวนการตรวจสอบและพิสูจน์ต้องมีความชัดแจ้งและชัดเจน มีกระบวนการวัดผลโดยบุคคลหรือเครื่องมือที่มีความน่าเชื่อถือและเป็นมาตรฐานสากลที่สามารถตอบและแสดงได้ว่าผลลัพธ์ที่ได้นั้น สามารถบรรลุความสำเร็จตรงกับความต้องการที่ได้กำหนดไว้หรือไม่ หรือเป็นไปตามข้อตกลงที่ระบุร่วมกันไว้หรือไม่ ซึ่งในส่วนนี้จะมีบทบาทต่อการออกแบบการทดสอบระบบ เพื่อส่งมอบงานให้ผู้ว่าจ้างในอนาคต เมื่อโครงการพัฒนาระบบเสร็จสิ้น

2.1.3 ต้องสามารถปรับเปลี่ยนให้อยู่ในรูปแบบที่เหมาะสมได้(modify) โดยทั่วไปแล้ว โครงสร้างและรูปแบบของความต้องการเชิงซอฟต์แวร์ที่ได้มา มักมีรูปแบบที่หลากหลาย ซึ่งเมื่อเข้าสู่กระบวนการจัดการความต้องการ ความต้องการเหล่านั้นจำเป็นต้องมีการเปลี่ยนแปลงให้อยู่ในรูปแบบที่เหมาะสมหรืออยู่ในรูปแบบที่สามารถนำไปใช้งานได้จริง ทั้งนี้การปรับเปลี่ยนรูปแบบจำเป็นต้องคงความสมบูรณ์ของเนื้อหาและมีความสอดคล้องไม่ผิดเพี้ยนไปจากความหมายเดิม นอกจากการปรับเปลี่ยนรูปแบบให้สามารถนำไปใช้งานได้จริงแล้ว การปรับเปลี่ยนรูปแบบยังสามารถช่วยลดปัญหาในเรื่องความซ้ำซ้อนของความต้องการได้ เนื่องจากบางความต้องการเดียวกันอาจสามารถมาได้ในหลายรูปแบบ ซึ่งการลดความซ้ำซ้อนอาจทำได้โดยการลดรูป การกำหนดรูปแบบมาตรฐาน และตรวจสอบการอ้างอิง

2.1.4 ต้องสามารถตรวจสอบแหล่งที่ได้มาและที่จะไปได้ (traceability) การตรวจสอบแหล่งที่ได้มาและที่จะไปของความต้องการ มีไว้เพื่อตรวจสอบว่าแต่ละความต้องการที่มีอยู่นั้น ได้มีการกำหนดที่มาที่ไปไว้อย่างชัดเจนหรือมีจุดเริ่มต้นที่ชัดแจ้งหรือไม่ และช่วยในเรื่องของการบำรุงรักษาการตามรอยของความต้องการโดยการใช้จุดอ้างอิงที่ชัดเจนในการตรวจสอบ ซึ่งการตรวจสอบแหล่งที่ได้มาและที่จะไปสามารถทำได้โดยการใช้ระดับชั้นของความต้องการเป็นกรอบในการตรวจสอบและติดตามรอยดังแสดงในรูปที่ 2 ช่วยได้ เพื่อให้มั่นใจได้ว่าทุกความต้องการมีที่มาจากความต้องการที่แท้จริง และทุกความต้องการจะถูกนำมาใช้เพื่อพัฒนาระบบได้ครบถ้วนตามที่ผู้ร้องขอต้องการ นอกจากนี้ยังช่วยในเรื่องของการติดตามผลกระทบเมื่อความต้องการเกิดการเปลี่ยนแปลงได้ด้วย



รูปที่ 2 การตรวจสอบความสัมพันธ์ของความต้องการในระดับชั้นต่างๆ

2.1.5 ต้องมีความถูกต้อง(validation) ทุกความต้องการเมื่อได้ทำการกำหนดให้เป็นความต้องการเชิงซอฟต์แวร์แล้วต้องมีการตรวจสอบความถูกต้อง ว่าตรงตามสิ่งที่ผู้ร้องขอความต้องการกำหนดไว้หรือไม่ อาจมีการทบทวนหรือยืนยันอีกครั้งก่อนที่จะนำไปใช้ ซึ่งผู้ที่สามารถยืนยันความถูกต้องได้ดีที่สุด คือ ผู้ที่ให้ความต้องการนั้นเอง

- ต้องละเอียดครบถ้วนสมบูรณ์ได้ใจความ(complete) ทุกความต้องการเมื่อได้ทำการกำหนดให้เป็นความต้องการเชิงซอฟต์แวร์แล้วต้องครบถ้วนสมบูรณ์ได้ใจความ เป็นความต้องการที่สอดคล้องกับเงื่อนไขและมีรายละเอียดต่างๆที่สามารถนำไปใช้งานได้จริงอย่างเหมาะสม ซึ่งรายละเอียดนั้นอาจอยู่ในรูปแบบของมิติรายละเอียดต่างๆ เช่น มีการกำหนดขอบเขตของค่าที่เป็นไปได้ มีการอ้างอิงและมีคำจำกัดความ รูปภาพ ตาราง และแผนภาพ หรือมีหน่วยมาตรฐานในการวัดเป็นต้น

2.1.6 ต้องมีความสอดคล้อง(consistent) ทุกความต้องการเมื่อได้ทำการกำหนดให้เป็นความต้องการเชิงซอฟต์แวร์แล้วต้องมีความสอดคล้อง หรือเป็นเนื้อความและความหมายเดียวกันทั่วทั้งเอกสารที่เกี่ยวข้องหรือสิ่งที่ใช้ประกอบการทำงานทั้งหมด เพื่อให้มั่นใจได้ว่าความต้องการที่ถูกกำหนดและนำมาพัฒนาระบบซอฟต์แวร์นั้น เป็นความต้องการเดียวกันและไม่มีเนื้อหาขัดแย้งกันเอง

2.1.7 ต้องไม่กำกวมหรือตีความได้หลากหลาย (unambiguous) ความต้องการต้องไม่กำกวมและมีเพียงความหมายเดียวที่สามารถให้ความเข้าใจตรงกันได้ ทีมผู้พัฒนาต้องมีการกำหนดคำศัพท์หรือคำนิยามที่เป็นมาตรฐานสำหรับใช้เฉพาะในโครงการที่กำลังพัฒนา ซึ่งการกำหนดนิยามศัพท์ ควรมีที่มาจากคำนิยามของผู้ว่าจ้าง ผู้ที่มีส่วนได้ส่วนเสีย และผู้เชี่ยวชาญที่อยู่ในโดเมน เพื่อก่อให้เกิดมาตรฐานเดียวกันในการติดต่อสื่อสาร

2.2 มิติการจัดการความต้องการให้มีคุณภาพ

การจัดการความต้องการให้มีคุณภาพ ต้องมีการกำหนดมิติคุณภาพของความต้องการที่เหมาะสม ในบทความนี้ ขอเสนอการใช้หลักองค์ประกอบ FURPS+ เป็นเกณฑ์ในการวัดตรวจสอบคุณภาพของความต้องการ (IBM Rational, 2008) โดย FURPS+ นั้น เป็นอักษรที่ย่อมาจาก

(F) = Functionality ความสามารถในการทำงานหลักของระบบ ตัวอย่างเช่น รูปแบบและความสามารถในการทำงาน(feature set capabilities) การเป็นไปตามหลักการทำงานสากลนิยม(generality)ที่ผู้ใช้งานสามารถคาดการณ์ผลลัพธ์ที่เกิดขึ้นได้(รู้ได้ว่าทำแล้วได้อะไร) และมาตรฐานความมั่นคงในการปฏิบัติงาน(security) เป็นต้น

(U) = Usability ความสามารถในการใช้งานของผู้ใช้หรือการติดต่อกับส่วนที่เกี่ยวข้องกับระบบ ตัวอย่างเช่น ด้านปัจจัยมนุษย์หรือขีดความสามารถในการปฏิบัติงานของผู้ใช้งานปกติ(human factors) ความน่าใช้งานหรือสุนทรียศาสตร์(aesthetics) ความความเข้าใจที่สอดคล้องกับการใช้งาน(consistency) รวมถึงเอกสารแนะนำหรือช่วยเหลือ(documentation) ซึ่งเป็นได้ทั้งเอกสารคู่มือโปรแกรมผู้ช่วยเหลือหรือระบบช่วยเหลือแบบออนไลน์ เป็นต้น ทั้งนี้อาจรวมถึงระยะเวลาในการอบรมและการฝึกฝนความชำนาญการของผู้ใช้งานด้วย

(R) = Reliability ความน่าเชื่อถือและความไว้วางใจในการทำงานของระบบ ตัวอย่างของเช่น ความถี่หรือระดับความรุนแรงเมื่อเกิดข้อผิดพลาด (frequency/severity of failure) ความสามารถในการกู้คืน (recovery) เช่น จำนวนขนาดของข้อมูลหรือระยะเวลาที่ใช้ในการกู้คืนระบบให้เป็นปกติ ระดับของความแม่นยำของข้อมูลหรือผลลัพธ์ที่ได้(accuracy) เช่น จำนวนตัวเลขทศนิยม ความสามารถในการคาดเดาสถานการณ์(predictability) และอัตราเฉลี่ยของความผิดพลาดต่อหนึ่งหน่วยที่กำหนด(MTBF - Mean Time between Failure) เช่น สามารถผิดพลาดได้ไม่เกินกี่ครั้งต่อการทำงาน หรือ อาจเป็นจำนวนเปอร์เซ็นต์ความน่าเชื่อถือ เช่น 99.999% เป็นต้น

(P) = Performance ประสิทธิภาพของการทำงานที่พึงพอใจหรือเกณฑ์ขั้นต่ำที่ผู้ใช้สามารถยอมรับได้ ตัวอย่างเช่น ความเร็วในการประมวลผล(speed) ซึ่งอาจแสดงได้ทั้งในสภาวะปกติและสภาวะการรองรับการปฏิบัติงานสูงสุด ประสิทธิภาพ(efficiency) ความสิ้นเปลืองทรัพยากร (resource usage)ในการปฏิบัติงานหรือเกณฑ์ทรัพยากรขั้นต่ำที่ระบบสามารถปฏิบัติงานให้บรรลุเป้าหมายได้ อัตรางานที่เกิดขึ้นต่อหนึ่งหน่วยเวลาที่กำหนด(throughput) และเวลาที่ใช้ในการตอบสนอง(response time) เป็นต้น

(S) = Supportability ความสามารถในการรองรับหรือขยายเพิ่มเติม การสนับสนุนการทำงานในสิ่งที่จะเกิดขึ้นในอนาคต ตัวอย่างเช่น สามารถตรวจสอบได้ (testability)ว่าระบบกำลังอยู่ในสภาวะปกติ ความสามารถในการขยายจากระบบเดิม(extensibility)โดยที่ไม่มีการเปลี่ยนแปลงสิ่งที่มีอยู่ ความสามารถในการปรับตัวหรือความยืดหยุ่น(adaptability) ความสามารถในการบำรุงดูแลรักษา (maintainability)ให้ปฏิบัติงานได้ปกติ ความสามารถในการเข้ากันได้หรือการรองรับกับสิ่งอื่น(compatibility) ความสามารถในการกำหนดและการจัดการ(configurability) ความสามารถในการให้บริการ(serviceability) เช่น มีการทำระบบประกันคุณภาพการให้บริการ(Quality of Service – QOS) ความสามารถในการติดตั้งในสภาพแวดล้อมที่

เฉพาะเจาะจง(installability) และความทนทานต่อสภาวะที่ไม่ปกติ(robustness) เป็นต้น

Plus (+) คุณภาพเฉพาะของความต้องการซึ่งขึ้นอยู่กับบริบทของระบบนั้นๆ ตัวอย่างของสิ่งที่เพิ่มเติมเฉพาะ เช่น ระบบด้านความมั่นคงอาจมีความต้องการเพิ่มเติมในส่วนของชั้นความลับหรือการเข้ารหัสป้องกัน หรือระบบที่สัมพันธ์กับมูลค่าชีวิต เช่น ระบบที่เกี่ยวข้องกับชีวิตมนุษย์ ระบบที่ใช้ในกรณีฉุกเฉินหรือสภาวะภัยพิบัติ ซึ่งระบบเหล่านี้อาจต้องมีความต้องการในเรื่องของการรับประกันคุณภาพอย่างสูงสุด เป็นต้น

นอกจากเกณฑ์คุณภาพของความต้องการที่เป็นรายละเอียดเฉพาะตัวของความต้องการหนึ่งๆแล้ว ภาพรวมของคุณภาพความต้องการในระดับโครงการก็เป็นสิ่งสำคัญในการผลักดันให้โครงการสำเร็จบรรลุเป้าหมายตามที่ต้องการ โดยมีนิยามสั้นๆว่า จุดมุ่งหมายของความต้องการในระดับโครงการ คือ เพื่อให้เกิดงานที่มีคุณภาพ ซึ่งอยู่ภายใต้ทรัพยากรที่กำหนด และค้นพบความต้องการที่แท้จริงจากผู้ที่มีส่วนเกี่ยวข้อง

3. ปัญหาและความเสียหายที่เกิดขึ้นกับความต้องการเชิงซอฟต์แวร์

3.1 ปัญหาที่เกิดขึ้นกับความต้องการ

ปัญหาที่มักพบกับความต้องการเชิงซอฟต์แวร์ อาจมีที่มาจากผู้ที่พัฒนาหรือผู้ที่ทำการเก็บเกี่ยวความต้องการ หรือจากตัวความต้องการเอง ในส่วนของผู้พัฒนา ปัญหาที่พบได้บ่อยครั้ง คือ ผู้พัฒนาอาจมีความเข้าใจที่ผิด ในการกำหนดความต้องการเชิงซอฟต์แวร์ ยกตัวอย่างเช่น

3.1.1 ผู้พัฒนาใช้การออกแบบระบบมาเป็นตัวกำหนดความต้องการ เช่น จะทำอย่างไรให้ความต้องการที่หาได้สามารถนำมาใช้ได้กับการออกแบบที่เตรียมไว้อยู่แล้ว หรือออกแบบโดยกำหนดในแต่ละส่วนของระบบก่อน แล้วค่อยระบุถึงส่วนติดต่อย่อยภายในส่วนประกอบนั้น

3.1.2 ใช้การตรวจสอบมากำหนดเป็นความต้องการ เช่น จะทำอย่างไรให้ความต้องการที่หาได้ สามารถตรวจสอบได้ตามที่กำหนดไว้ หรือคิดแล้วว่าจะตรวจสอบอย่างไร แล้วจึงไปกำหนดความต้องการเพื่อให้เกิดการตรวจสอบในรูปแบบที่คิดเอาไว้

3.1.3 นำข้อมูลโครงการมากำหนดเป็นความต้องการ เช่น นำกำหนดตารางเวลาในการพัฒนาระบบเป็นความต้องการ

3.1.4 บางครั้งผู้พัฒนาระบบหลงลืมตน คิดว่าตนเองเข้าใจระบบดีกว่าผู้ใช้งานจริง อาจมีการตัดสินใจโดยพลการหรือเสนอแนะโน้มน้าวความคิดของผู้ให้ความต้องการจริงซึ่งอาจเกิดขึ้นโดยไม่ได้ตั้งใจ

ซึ่งสิ่งที่กล่าวมาทั้งหมดข้างต้นนี้ ไม่ถือว่าเป็นความต้องการเชิงซอฟต์แวร์ นอกจากปัญหาที่เกิดจากผู้พัฒนาเองแล้ว ปัญหาในการค้นหาและเก็บเกี่ยวความต้องการก็เป็นอุปสรรคสำคัญหนึ่ง เนื่องจากความต้องการมักจะมีคุณสมบัติดังนี้

3.1.5 ความต้องการมักไม่ค่อยแสดงให้เห็นเด่นชัด ผู้พัฒนาอาจต้องตีความหรือทำการตรวจสอบเพิ่มเติม เพื่อให้เข้าถึงความต้องการที่แท้จริง

3.1.6 ความต้องการมีที่มาจากหลายแหล่งซึ่งแหล่งที่มาบางแหล่ง อาจอยู่ภายนอกระบบหรือองค์กรที่กำลังพัฒนา เช่น ข้อมูลคู่แข่งทางธุรกิจ เป็นต้น

3.1.7 บางความต้องการมีคุณสมบัติเฉพาะตัว ซึ่งผู้ที่ทำการเก็บเกี่ยวอาจต้องทำการศึกษาเชิงลึก หรืออาศัยผู้เชี่ยวชาญให้คำแนะนำ

3.1.8 ความต้องการมีการเปลี่ยนแปลงอยู่เสมอ ซึ่งผู้พัฒนาจำเป็นต้องรอคอยให้เกิดการเปลี่ยนแปลงจนถึงที่สุด หรือกำหนดวันที่ช้าที่สุดในการหยุดการเปลี่ยนแปลงความต้องการ(freeze) เพื่อทำการจัดเก็บ

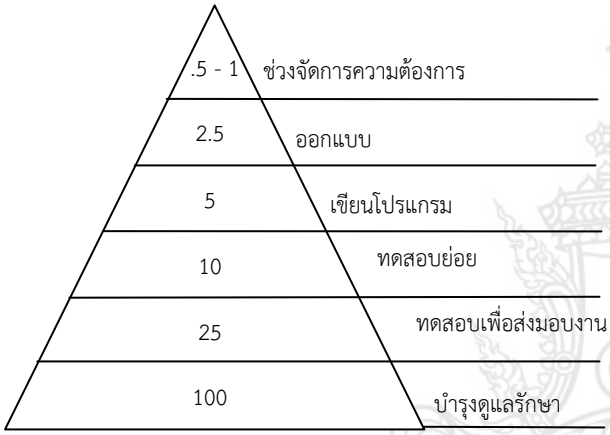
ซึ่งปัญหาที่กล่าวมาข้างต้นเหล่านี้ เป็นปัญหาปกติที่มักเกิดขึ้นในขั้นตอนการค้นหาและเก็บเกี่ยวความต้องการ ซึ่งหากถูกปล่อยปะละเลยโดยขาดการควบคุมหรือแก้ไข ผลเสียที่เกิดขึ้นอาจทำให้ผู้ที่มีส่วนเกี่ยวข้อง ทั้งผู้ว่าจ้างและทีมผู้พัฒนาเอง หมดกำลังใจหรือเกิดความท้อถอยในการทำงานให้บรรลุเป้าหมายได้

3.2 ความเสียหายเมื่อความต้องการเกิดข้อผิดพลาด

ความเสียหายที่เกิดขึ้นเมื่อความต้องการเกิดข้อผิดพลาด มักไม่หยุดนิ่งและมีการขยายตัวไปพร้อมๆกับกระบวนการทำงานที่ดำเนินงานไปเรื่อยๆ เนื่องจากความต้องการเป็นจุดเริ่มของสิ่งที่จะพัฒนาและเป็นจุดวัดความสำเร็จในขั้นตอนการส่งมอบเมื่อสิ้นสุดโครงการ ซึ่งหากความต้องการมีข้อผิดพลาดก็จะหมายถึงจุดจบที่ไม่ประสบความสำเร็จตามไปด้วย และจากการที่ความต้องการจะมีขยายความเสียหายตามกระบวนการพัฒนาที่ดำเนินการไปเรื่อยๆ นั้น การควบคุมความต้องการและลดความผิดพลาดตั้งแต่เนิ่นๆและทันทีจึงเป็นสิ่งที่ดีที่สุด ซึ่งต้องใช้กระบวนการจัดการความต้องการเข้ามาช่วยในการดำเนินงาน

กฎ 1-10-100 ในรูปที่ 3 (ที่มา: Rational Software, 2005) เป็นกฎที่แสดงให้เห็นถึงความเสียหายที่เกิดขึ้นเมื่อความต้องการมีข้อผิดพลาดแล้วถูกปล่อยปละละเลยดำเนินการต่อไป ความเสียหายที่เกิดขึ้นอาจสามารถขยายเป็นสิบเท่าเมื่อก้าวข้ามสู่กระบวนการถัดไป ตัวอย่างเช่น หากความต้องการเกิดข้อผิดพลาดในกระบวนการเก็บเกี่ยว การแก้ไขอาจทำได้โดยการลบความ

ต้องการนั้นทิ้งแล้วทำการเก็บเกี่ยวใหม่ แต่ถ้าหากปล่อยความต้องการที่ผิดพลาดนี้เข้าสู่กระบวนการวิเคราะห์และออกแบบระบบ ความเสียหายที่เกิดขึ้น อาจจำเป็นต้องทำการจ้างวิศวกรซอฟต์แวร์ให้ทำการวิเคราะห์และออกแบบระบบใหม่ หรือให้โปรแกรมเมอร์ทำการเขียนโปรแกรมใหม่ ซึ่งทำให้ต้นทุนการพัฒนาโครงการสูงขึ้น และทำให้ใช้ระยะเวลาในการพัฒนาระบบเพิ่มขึ้น ซึ่งอาจจะเกินกว่าที่ได้ตกลงกันไว้กับผู้ว่าจ้าง หรือในกรณีที่เลวร้ายสุด คือ ผู้ใช้งานนำระบบที่ผิดพลาดไปปฏิบัติงานจริงแล้วเกิดปัญหาในขั้นตอนการทำงานจริง ซึ่งบางระบบอาจไม่สามารถทำการประเมินมูลค่าความเสียหายหรือความสูญเสียได้ เช่น ระบบที่เกี่ยวข้องกับชีวิตและความปลอดภัย รูปที่ 3 แสดงอัตราส่วนค่าใช้จ่ายในการแก้ไขปัญหาต่อช่วงระยะการพัฒนาระบบซอฟต์แวร์



อัตราส่วนของต้นทุนในการซ่อมแซมข้อผิดพลาด
เปรียบเทียบมูลค่าตอนเริ่มต้นและค่าซ่อมแซม

รูปที่ 3 กฎ 1-10-100

4. ปัจจัยที่ส่งผลต่อความสำเร็จและบทบาทของทีมพัฒนาในการจัดการความต้องการ

4.1 ปัจจัยที่ส่งผลต่อความสำเร็จในการจัดการความต้องการ

ในการจัดการความต้องการ คำถามที่มักจะพบได้บ่อยในระหว่างการดำเนินการ คือ ต้องมีความต้องการจำนวนเท่าใด จึงจะเพียงพอต่อการพัฒนาระบบซอฟต์แวร์ ซึ่งเป็นคำถามที่ไม่มีคำตอบในเรื่องของปริมาณ แต่ขึ้นอยู่กับเงื่อนไขสำคัญคือ เรื่องของเงื่อนไขเวลาและทรัพยากร โดยทั่วไปแล้วความต้องการที่ได้มา มักจะมีจำนวนมากเกินกว่าเวลาและทรัพยากรที่มีอยู่ที่ผู้พัฒนาระบบสามารถพึงกระทำได้(ความต้องการของมนุษย์ไร้ขีดจำกัด) ด้วยเหตุนี้ การจัดการความต้องการจึงจำเป็นต้องมีการกำหนดการควบคุมที่ชัดเจน ซึ่งผู้พัฒนาอาจใช้เส้นกำกับโครงการ(base line) เป็นเครื่องมือในการกำหนดเวลาและขั้นตอนการดำเนินงานต่างๆ ที่เกี่ยวข้องกับการจัดการความต้องการ เช่น กำหนดการในการค้นหาและเก็บเกี่ยวความต้อง วันที่ตรวจทานและตรวจสอบแก้ไข วันที่สิ้นสุดการเก็บเกี่ยวความต้องการทั้งโครงการ เป็นต้น (Borland CaliberRM, 2012: online)

นอกจากเงื่อนไขเวลาและทรัพยากรที่มีอยู่แล้ว ปัจจัยที่ส่งผลต่อความสำเร็จในการจัดการความต้องการ ยังขึ้นอยู่กับ 3 ปัจจัย ดังนี้

4.1.1 ผู้ใช้ระบบให้ความร่วมมือและให้ความสำคัญ เป็นที่แน่นอนว่าหากโครงการใดที่ผู้ใช้งานให้ความร่วมมือและให้ความสำคัญ โครงการนั้นจะลดปัญหาและความเสี่ยงลงไปได้มาก เนื่องจากความต้องการเป็นงานที่เกี่ยวข้องกับผู้ใช้ระบบโดยตรง หากผู้ใช้ให้ความร่วมมือและสนับสนุนการดำเนินงาน จะทำให้ความต้องการที่ได้มานั้นมีคุณภาพและสามารถนำไปใช้ในการแก้ปัญหาได้จริง ซึ่งการทำความเข้าใจนั้นเป็นสิ่งที่สำคัญ นอกจากนี้ทีมผู้พัฒนาอาจต้องมีทักษะการโน้มน้าว การให้มีส่วนร่วมหรือเป็นเจ้าของร่วมกัน หรือการใช้ทฤษฎีได้ประโยชน์ทั้งสองฝ่าย(Win – Win) ช่วยในการค้นหาและเก็บเกี่ยวความต้องการ

4.1.2 แรงสนับสนุนจากผู้บริหารระดับสูงหรือผู้นำองค์กร นอกจากกลุ่มบุคคลเหล่านี้จะมีส่วนร่วมผลักดันด้านนโยบายและการจัดสรรงบประมาณแล้ว การดำเนินงานต่างๆ จะรวดเร็วขึ้นหากได้รับแรงสนับสนุนจากผู้บริหารระดับสูงหรือผู้นำองค์กร เนื่องจากเป็นผู้สั่งการในส่วนที่บางครั้ง ผู้เก็บเกี่ยวความต้องการซึ่งถือเป็นบุคคลภายนอกองค์กรไม่สามารถพึงกระทำเองได้โดยพลการ

4.1.3 ระบบมีเป้าหมายทางธุรกิจที่ชัดเจน การมีเป้าหมายที่ชัดเจนเป็นส่วนช่วยให้การดำเนินงานเกี่ยวกับการจัดการความต้องการเป็นไปตามทิศทางที่ถูกต้อง ซึ่งทำให้ความต้องการที่ได้มานั้นสามารถดำเนินตามจุดมุ่งหมาย นอกจากนี้ยังทำให้สามารถมองเห็นภาพรวมของสิ่งที่จะเกิดขึ้นและสิ่งที่อยากให้เป็นในอนาคตชัดเจนขึ้นด้วย

ซึ่งนอกจากปัจจัยที่มีผลต่อความสำเร็จในการจัดการความต้องการแล้ว ยังมีปัจจัยอื่นๆอีกซึ่งมีความสำคัญต่อการดำเนินงานในการจัดการความต้องการ เช่น ปัจจัยเรื่องขนาดของโครงการ ซึ่งจากสถิติที่ผ่านมา มีความชัดเจนว่าโครงการที่มีขนาดใหญ่ มักมีความเสี่ยงของการล้มเหลวสูงกว่าโครงการที่มีขนาดเล็กกว่า ซึ่งเป็นผลเนื่องมาจากความซับซ้อนที่มากขึ้นและการควบคุมดูแลที่เป็นไปได้ยากขึ้น ซึ่งอาจแก้ไขโดยการกระจายความเสี่ยง การแบ่งเป็นส่วนๆเพื่อลดความซับซ้อน การจ้างเหมาช่วง(subcontract) เป็นต้น

4.2 บทบาทและเป้าหมายของทีมค้นหาเก็บเกี่ยวและจัดการความต้องการ

ในทีมค้นหาเก็บเกี่ยวและจัดการความต้องการโดยทั่วไปแล้ว มักจะประกอบด้วยบุคคลที่มีบทบาทดังนี้ ผู้พัฒนา(developer) ผู้ทดสอบ(tester) และผู้เขียน(writer) ซึ่งจำนวนคนและจำนวนทีมนั้น ขึ้นอยู่กับผู้จัดการโครงการเป็นผู้ตัดสินใจ โดยมีปัจจัยทางด้านขนาดของโครงการ ระยะเวลาในการดำเนินงาน และสภาพแวดล้อมของการค้นหาและเก็บเกี่ยวความต้องการเป็นพื้นฐาน ซึ่งทั้งหมดจะต้องมีจุดมุ่งหมายและหลักการทำงานที่สัมพันธ์กันดังนี้ คือ

4.2.1 ช่วยกันพัฒนาความต้องการ และให้เป็นระบบจัดการความต้องการ

4.2.2 ยึดถือในข้อปฏิบัติร่วมกันตามที่ได้ตกลงกันไว้ เพื่อให้เกิดประสิทธิภาพในการปฏิบัติงานและการจัดการ

4.2.3 ยืนยันกระบวนการทำงานให้เป็นไปตามแผนการดำเนินงานที่ได้กำหนด

4.2.4 จัดทำเอกสารรายงานความต้องการ เพื่อใช้เป็นหลักฐานที่เป็นลายลักษณ์อักษร คำกล่าวลอยๆ หรือข้อตกลงปากเปล่า จะไม่นับเป็นความต้องการ จนกว่าจะมีการจดบันทึกและเห็นชอบทั้งผู้ให้และผู้เก็บเกี่ยวความต้องการนั้นๆ

4.2.5 มีส่วนร่วมในการตรวจสอบความต้องการ เพื่อให้มั่นใจว่า สามารถทำการส่งมอบงานได้ เมื่อโครงการเสร็จสิ้นและเป็นไปตามข้อตกลงร่วมกับผู้ว่าจ้าง

4.2.6 เข้าร่วมในตำแหน่งคณะกรรมการควบคุมการเปลี่ยนแปลง CCB (Change Control Board) ซึ่งมีหน้าที่ควบคุม ตรวจสอบและตัดสินใจอนุมัติในการเปลี่ยนแปลงความต้องการ โดยจะพิจารณาว่า หากความต้องการเกิดเปลี่ยนแปลง จะมีผลกระทบใดบ้างทั้งในภาพรวมและส่วนย่อย และหากเปลี่ยนแปลงแล้วจะทำให้โครงการที่ดำเนินอยู่บรรลุเป้าหมายหรือผิดวัตถุประสงค์หรือไม่

4.2.7 ตรวจสอบผลลัพธ์ที่ได้จากการค้นหาและเก็บเกี่ยวความต้องการ

4.2.8 ยืนยันในมิติคุณภาพ การทบทวน ความพร้อมมูลของความต้องการที่ได้มา เพื่อเตรียมความพร้อมในกระบวนการพัฒนาขั้นตอนถัดไป

5. สรุป

จะเห็นได้ว่า ในกระบวนการพัฒนาระบบซอฟต์แวร์ ความต้องการจากผู้ใช้ ผู้ว่าจ้างและผู้ที่มีส่วนได้ส่วนเสียกับระบบ จะเป็นข้อมูลเริ่มต้นและขับเคลื่อนในกระบวนการทางวิศวกรรมซอฟต์แวร์ ดังนั้นความต้องการที่ค้นหาและได้เก็บเกี่ยวมานั้น จึงจำเป็นต้องมีคุณภาพ ผ่านการกลั่นกรองและต้องมีการจัดเก็บและจัดการอย่างเป็นระบบ เพื่อให้ผลิตภัณฑ์ทางซอฟต์แวร์ที่ได้ ตอบสนองเป้าหมายและบรรลุวัตถุประสงค์ของผู้ที่ให้ความต้องการ แต่อย่างไรก็ตาม การจัดการความต้องการไม่ใช่เรื่องง่าย เนื่องจากมีปัจจัยที่เกี่ยวข้องในหลายๆด้าน ทั้งที่มาจากบุคคล เช่น ผู้พัฒนา ผู้ใช้งาน ผู้ว่าจ้างหรือผู้ที่มีส่วนเกี่ยวข้องอื่นๆ หรือปัญหาที่มาจากธรรมชาติของความต้องการเอง เช่น มีความไม่ชัดเจน ต้องมีการตีความและเข้าใจถึงวัตถุประสงค์แอบแฝง ซึ่งหากปัญหาเหล่านี้มีการปล่อยปละละเลยไป ก็จะส่งผลกระทบให้กระบวนการพัฒนาในขั้นตอนต่อไปล้มเหลวหรือโครงการพัฒนาซอฟต์แวร์นั้นไม่ประสบความสำเร็จได้

แต่อย่างไรก็ตาม คำตอบของการจัดการความต้องการให้ประสบความสำเร็จนั้น มีปัจจัยหลายด้านเป็นองค์ประกอบและไม่มีสูตรสำเร็จที่ตายตัว (บุญประเสริฐ และคณะ, 2552: 2) บางวิธีสามารถใช้ได้กับระบบหนึ่งแต่ไม่สามารถใช้ได้กับระบบอื่น ซึ่งประสบการณ์จริงในการทดลองปฏิบัติงาน ความเข้าใจในระบบอย่างถ่องแท้ การเผชิญและแก้ไขปัญหา และการฝึกฝนอย่างสม่ำเสมอเป็นสิ่งที่มีความสำคัญ ที่จะผลักดันให้การเก็บเกี่ยวและการจัดการความต้องการทางด้านซอฟต์แวร์ประสบความสำเร็จและสามารถบรรลุเป้าหมายตามวัตถุประสงค์ที่ต้องการ ขอขอบคุณ Professor Dr. Kazuhiko HAMAMOTO, Tokai University ประเทศญี่ปุ่น และคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในการอำนวยความสะดวกเพื่อจัดทำบทความวิชาการนี้

6. กิตติกรรมประกาศ

ขอขอบคุณ Professor Dr. Kazuhiko HAMAMOTO, Tokai University ประเทศญี่ปุ่น และคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในการอำนวยความสะดวกเพื่อจัดทำบทความวิชาการนี้

7. เอกสารอ้างอิง

บุญประเสริฐ สุรักษ์รัตนสกุล.2551. **สถาปัตยกรรมเครื่องมือจัดการความต้องการโดยใช้แนวคิดเชิงบริการ** ในการประชุมวิชาการ ม.อ.ภูเก็ตวิจัย ครั้งที่ 1 สหวิทยาการเพื่อการพัฒนาอย่างยั่งยืน.ภูเก็ต : มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตภูเก็ต.

บุญประเสริฐ สุรักษ์รัตนสกุล สุรเชษฐ์ สูรย์ส่องธานี ลิสา และสิมะสาธิตกุล.2552.**ระบบบริหารจัดการความต้องการทางซอฟต์แวร์แบบเว็บเซอร์วิส** ในการประชุมวิชาการและนำเสนอผลงานวิจัยนานาชาติ ครั้งที่ 2 ความร่วมมือเพื่อการพัฒนาบนเส้นทางระเบียงเศรษฐกิจ ตะวันออก-ตะวันตก. สกลนคร : มหาวิทยาลัยราชภัฏสกลนคร.

Kotonya G. and Sommerville I. 1998. **Requirements Engineering Processes and Techniques.** Chichester : John Wiley & Sons.

Leffingwell Dean and Widrig Don. 2004. **Managing Software Requirements: A Use Case Approach.** 2nd Edition. Reading, Massachusetts : Addison-Wesley Professional.

Sommerville, lan. 2010. **Software Engineering.** 9th Edition. Boston, Massachusetts : Addison-Wesley.

Booch, G, Rumbaugh, J, and Jacobson, I. 1998. **The Unified Modeling Language User Guide.** Reading, Massachusetts : Addison-Wesley.

Rational Software. 2005. *Requirements Management with Use Case*. Boston, Massachusetts : Addison-Wesley.

IBM. **Rational RequisitePro: A Requirements Management tool.** 2008. [Online] Available : http://www-01.ibm.com/software/awdtools/reqpro/ (25 September 2012)

Borland. **CaliberRM Enterprise Software Requirements Management System.** (2012) [Online] Available : http://www.borland.com/us/products/caliber/index.html. (15 October 2012).