การศึกษาผลของการดำเนินโครงการซอฟต์แวร์ภายใต้หลักการวิศวกรรมความต้องการ

A STUDY ON RESULTS OF SOFTWARE PROJECTS DEVELOPED UNDERLYING PRINCIPLES OF REQUIREMENTS ENGINEERING

วราพร จิระพันธุ์ทอง*
Waraporn Jirapanthong*

*รองศาสตราจารย์ คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยธุรกิจบัณฑิตย์
*Associate Professor, College of Innovative Technology and Engineering, Dhurakij Pundit University
*Email: waraporn.jir@dpu.ac.th

บทคัดย่อ

งานวิจัยนี้มีจุดประสงค์เพื่อศึกษาทดลองการดำเนินโครงการซอฟต์แวร์โดยเปรียบเทียบการดำเนินกิจกรรมตามหลักวิศวกรรมความต้องการและที่ไม่เคร่งครัดต่อกิจกรรมภายใต้วิศวกรรมความต้องการ การทดลองดำเนินโครงการซอฟต์แวร์โดยแบ่งออกเป็น 2 ขั้นตอน ขั้นตอนที่ 1 คือ การศึกษากิจกรรมวิศวกรรมความต้องการในการพัฒนาโครงงานซอฟต์แวร์ และขั้นตอนที่ 2 คือ การศึกษากิจกรรมวิศวกรรมความต้องการในการเปลี่ยนแปลงความต้องการระบบซอฟต์แวร์ที่มีอยู่เดิม โดยจัดให้มีทีมพัฒนา 2 ทีม สำหรับขั้นตอนที่ 1 ให้มีการสร้างโครงการพัฒนาซอฟต์แวร์ 3 ชุดโดยที่มีความต้องการบางส่วนที่คล้ายกันและความต้องการบางส่วนที่แตกต่าง โดยกำหนดให้ทีมพัฒนาที่หนึ่งดำเนินโครงการตามกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัดและทีมพัฒนาที่สองดำเนินโครงการโดยมุ่งเน้นการพัฒนาเขียนโค้ดและไม่เคร่งครัดในแง่ของวิศวกรรมความต้องการ พบว่า จำนวนเอกสารที่ทีมพัฒนาที่หนึ่งสร้างมีมากกว่าคิดเป็น 2.5 เท่าของทีมที่สอง และขนาดของซอฟต์แวร์ที่พิจารณาจากจำนวนบรรทัดของโปรแกรมที่ทีมที่หนึ่งพัฒนาขึ้นน้อยกว่าที่ทีมพัฒนาที่สองคิดเป็น 0.82 เท่า นอกจากนี้ทีมที่หนึ่งใช้เวลาเฉลี่ยรวมในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์น้อยกว่าถึง 0.53 เท่าของทีมที่สอง

คำสำคัญ: วิศวกรรมความต้องการ วิศวกรรมซอฟต์แวร์ การสร้างข้อกำหนดความต้องการ ความต้องการ

Abstract

This research aims to study the implementation of the activities under software projects by considering applying the principles of requirements engineering. Experimental software projects are divided into two steps: Step 1 is to study the activities of software projects development underlying requirements engineering and Step 2 is to study the activities of software change management. By providing two development teams, each is to implement software projects with three sets of requirements which are some similar and some different. The first team is asked to strictly follow the activities underlying requirements engineering while the second team is asked to focus on coding and not strict in terms of requirements engineering. We found that the number of documents that the first team created are 2.5 times more than the second team's and the size of the software; the number of lines of code; is 0.82 times less than the second team's. Moreover, the first team takes average time to implement software changes 0.53 times less than the second team's.

Keywords: Requirements Engineering, Software Engineering, Requirements Specification, Requirements

บทนำ

จากการศึกษาของ Standish Group (2006) ในปีพ.ศ. 2549 พบว่า ปัญหาที่เกิดขึ้นกับโครงการซอฟต์แวร์ต่างๆได้รับการแก้ไขและมีการดำเนินการที่ดีขึ้นในช่วงสิบสองปี คือระหว่างปีพ.ศ. 2536-2549 จากที่เคยพบว่า ประมาณร้อยละ 30 ของโครงการซอฟต์แวร์ที่ได้ตรวจสอบในปี พ.ศ. 2536 ล้มเหลวและไม่ประสบความสำเร็จ ได้เหลือเพียงร้อยละ 20 ในปี พ.ศ. 2549 ทั้งนี้จากการศึกษาพบอีกว่าในปี พ.ศ. 2549 โครงการซอฟต์แวร์ที่ดำเนินการเกินระยะเวลา ข้อจำกัด หรืองบประมาณที่กำหนดไว้อย่างมีนัยสำคัญ หรือโครงการซอฟต์แวร์ที่ไม่สามารถตอบสนองความพึงพอใจของลูกค้าได้ลดลงจากร้อยละ 53 เหลือร้อยละ 46 ทั้งนี้พบว่า การสื่อสารความต้องการเป็นปัจจัยสำคัญที่ส่งผลให้การดำเนินโครงการซอฟต์แวร์ดีขึ้นมาก วิศวกรรมความต้องการจึงเป็นศาสตร์ความรู้ที่เสนอแนวทางปฏิบัติที่สนับสนุนกิจกรรมภายใต้โครงการซอฟต์แวร์ และเป็นปัจจัยสำคัญที่จัดการปัญหาอันเป็นสาเหตุของความล้มเหลวของโครงการ แต่อย่างไรก็ตามเนื่องจากข้อจำกัดเรื่องของเวลาในการดำเนินโครงการซอฟต์แวร์เพื่อตอบสนองความต้องการของลูกค้าที่เปลี่ยนแปลงอย่างรวดเร็วและมีความต้องการที่หลากหลายมากขึ้นในปัจจุบัน ทำให้นักพัฒนาซอฟต์แวร์ถูกเร่งรัดด้วยเงื่อนไขและปัจจัยหลายด้าน จึงเป็นไปได้ที่จะเพิกเฉยหรือให้ความสำคัญกับกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัด งานวิจัยนี้มีจุดประสงค์เพื่อศึกษาทดลองการดำเนินโครงการซอฟต์แวร์โดยเปรียบเทียบการดำเนินโครงการซอฟต์แวร์ที่เคร่งครัดตามหลักวิศวกรรมความต้องการและการดำเนินโครงการซอฟต์แวร์โดยมุ่งเน้นการเขียนโค้ดและไม่เคร่งครัดต่อกิจกรรมภายใต้วิศวกรรมความต้องการ

วัตถุประสงค์ของการวิจัย

1. เพื่อศึกษาข้อดีและข้อเสียของกิจกรรมภายใต้วิศวกรรมความต้องการ
2. เพื่อศึกษาบริบทของความต้องการที่สำคัญในการสร้างข้อกำหนดความต้องการที่สมบูรณ์

แนวคิดและวรรณกรรมที่เกี่ยวข้อง

กระบวนการทางวิศวกรรมความต้องการเป็นกระบวนการที่มีอิทธิพลในการผลักดันให้การดำเนินงานของโครงการประสบความสำเร็จ ความสำเร็จของโครงการจะเกิดขึ้นได้หากสามารถระบุและรวบรวมความต้องการได้อย่างครบถ้วน รวมถึงการบันทึกความต้องการเหล่านั้นได้อย่างถูกต้องและเหมาะสม

ความต้องการ (Requirements) [IEEE std. 610.12-1990] ได้ให้นิยามไว้ดังนี้

(1) ความต้องการ หมายถึง เงื่อนไขหรือความสามารถที่ผู้ใช้ต้องการให้มีเพื่อแก้ไขปัญหาหรือบรรลุเป้าหมายในการดำเนินการใดๆอย่างใดอย่างหนึ่ง

(2) ความต้องการ หมายถึง เงื่อนไขหรือความสามารถที่ระบบหรือส่วนประกอบของระบบต้องมีเพื่อบรรลุข้อตกลงมาตรฐาน ข้อกำหนด หรือเอกสารทางการอื่นๆที่อาจสามารถอ้างอิงได้

(3) ความต้องการ หมายถึง เอกสารทางการที่ระบุเงื่อนไขหรือความสามารถตามข้อที่ (1) หรือ (2)

นอกจากนี้วิศวกรรมความต้องการยังให้ความสำคัญกับคำนิยามสำหรับผู้มีส่วนได้ส่วนเสีย (Stakeholder) เนื่องจากผู้มีส่วนได้ส่วนเสียเป็นแหล่งที่มาของความต้องการที่สำคัญที่สุด การหลงลืมหรือขาดผู้มีส่วนได้ส่วนเสียคนหนึ่งอาจมีผลทำให้ความต้องการไม่สมบูรณ์หรือขาดตอนได้ (Macaulay, 1993)

ผู้มีส่วนได้ส่วนเสีย หมายถึง บุคคลหรือหน่วยงานที่มีผลกระทบต่อความต้องการ บุคคลที่ปฏิสัมพันธ์กับระบบ เช่น ผู้ใช้งานระบบ หรือ เจ้าหน้าที่บำรุงรักษาระบบ เป็นต้น บุคคลบางกลุ่มที่แทบจะไม่มีความสนใจหรือเกี่ยวข้องกับระบบโดยตรง เช่น ผู้จัดการระบบ ผู้ที่แอบขโมยข้อมูล ผู้มีส่วนได้ส่วนเสียของระบบคู่แข่งเป็นต้นบุคคลเหล่านี้อาจจะถูกพิจารณาให้เป็นแหล่งที่มาหรือนิยามความต้องการของระบบได้ผู้มีส่วนได้ส่วนเสีย เช่น ผู้ใช้งานระบบ นักเขียนโปรแกรม นักวิเคราะห์ระบบ ผู้จัดการโครงงานวิศวกรซอฟต์แวร์ ผู้ดูแลฐานข้อมูล เป็นต้น (Pohl & Rupp, 2011) ได้นิยามไว้ว่า ผู้มีส่วนได้ส่วนเสียของระบบ (Stakeholder of a system) คือ บุคคลหรือหน่วยงานที่มีอิทธิพลต่อความต้องการของระบบทั้งทางตรงและทางอ้อม

เป้าหมายของวิศวกรรมความต้องการ คือ เพื่อทำให้กิจกรรมในกระบวนการพัฒนาระบบบรรลุล่วงได้ด้วยดี กิจกรรมเหล่านั้น ได้แก่ การดึงความต้องการของผู้มีส่วนได้ส่วนเสียออกมา การบันทึกความต้องการเหล่านั้นในรูปแบบที่เหมาะสม การทวนสอบและการตรวจสอบความสมเหตุสมผลของความต้องการ และการบริหารจัดการความต้องการตลอดวงจรชีวิตของระบบ (Pohl, 1996)

วิศวกรรมความต้องการ (Requirements Engineering) หมายถึง วิธีการที่เป็นระบบและระเบียบในการสร้างข้อกำหนดและบริหารจัดการความต้องการโดยมีจุดมุ่งหมาย ดังนี้

(1) การตระหนักถึงความต้องการที่เกี่ยวข้อง การบรรลุข้อตกลงระหว่างผู้มีส่วนได้ส่วนเสียเกี่ยวกับความต้องการ การบันทึกความต้องการให้รูปแบบมาตรฐานที่กำหนดไว้ และการบริหารจัดการความต้องการอย่างเป็นระบบ

(2) การทำความเข้าใจและการบันทึกความประสงค์และความจำเป็นของผู้มีส่วนได้ส่วนเสีย และการบริหารจัดการความต้องการเพื่อลดความเสี่ยงของการพัฒนาระบบที่ไม่ตรงกับความประสงค์และความจำเป็นของผู้มีส่วนได้ส่วนเสีย

กิจกรรมหลักในกระบวนการวิศวกรรมความต้องการประกอบด้วย 4 กิจกรรม ดังนี้

(1) การทำให้ได้มาซึ่งความต้องการ (Requirements Elicitation) เริ่มตั้งแต่การศึกษาความเป็นไปได้ โดยรวบรวมข้อมูลจากผู้มีส่วนได้ส่วนเสียและแหล่งข้อมูลต่างๆที่เกี่ยวข้อง การวิเคราะห์ขั้นตอนการทำงานของผู้ใช้ การตรวจสอบรายงานปัญหาที่เกิดจากระบบต่างๆที่มีอยู่และเกี่ยวข้อง การพิจารณาความต้องการของระบบอื่น การพิจารณาข้อมูลต่างๆที่เกี่ยวข้อง ได้แก่ ความต้องการทางธุรกิจ (Business Requirement) เป็นต้น อาจใช้เครื่องมือหรือวิธีการ เช่น ใช้แผนภูมิ สร้างต้นแบบของผู้ใช้ การสัมภาษณ์

(2) การบันทึกและจัดทำเอกสาร (Documentation) เป็นขั้นตอนของการระบุความต้องการในรายละเอียด เช่น ข้อสำคัญทางธุรกิจ กฎ เงื่อนไข เป็นต้น โดยสร้างเป็นข้อกำหนดความต้องการ (Requirements Specification) เทคนิคการสร้างข้อกำหนดความต้องการที่นิยมใช้ ได้แก่ การใช้ภาษา UML

(3) การตรวจสอบความต้องการ (Requirements Validation) เป็นการตรวจสอบผ่านเอกสารข้อกำหนดความต้องการ การตรวจสอบความต้องการอาจทำได้จากการตรวจเปรียบเทียบและทดสอบจากระบบต้นแบบเพื่อตรวจสอบความถูกต้อง ประสิทธิภาพ ความพึงพอใจของผู้ใช้ ว่าข้อกำหนดความต้องการได้รวบรวมและบันทึกเป็นสิ่งที่ลูกค้าหรือผู้มีส่วนได้ส่วนเสียต้องการอย่างแท้จริงหรือไม่ หากพบความผิดพลาดในข้อกำหนดความต้องการระหว่างการพัฒนาระบบ หรือหลังจากส่งมอบระบบแล้ว ค่าใช้จ่ายที่เกิดขึ้นจะสูงมากกว่า การตรวจสอบความต้องการครอบคลุมถึงปัจจัยดังนี้ ก) ความสมเหตุสมผล (Validity Check) ตรงกับข้อกำหนดของผู้ใช้หรือไม่; ข) ความสอดคล้อง (Consistency Check) ไม่มีข้อกำหนดที่ขัดแย้งกัน; ค) ความสมบูรณ์ (Completeness Checks) มีทุกฟังก์ชันและข้อจำกัดที่ผู้ใช้ของระบบต้องการ; ง) ความเป็นไปได้จริง (Realism Checks) ระบบสามารถถูกสร้างได้จริงตามงบประมาณและเวลาที่กำหนด; และ จ) การทวนสอบ (Verifiability) ข้อกำหนดต้องพิสูจน์ยืนยันได้ เช่น มีเป้าหมายเป็นตัวเลขที่วัดได้

(4) การบริหารจัดการความต้องการ (Requirements Management) เป็นบริหารจัดการการเปลี่ยนแปลงของความต้องการที่เกิดขึ้นและประเมินผลกระทบที่อาจเกิดขึ้นจากความต้องการที่เปลี่ยนแปลงไป การบริหารจัดการความต้องการที่ดีจะต้องสามารถทำการทดสอบและสืบค้นสถานะของความต้องการที่เปลี่ยนแปลงไปได้

แบบจำลองกระบวนการอย่างเช่น แบบจำลองน้ำตก (Royce, 1987) หรือแบบจำลองวี (V-Model, 2004) ได้วางแบบให้การได้มาซึ่งความต้องการและการบันทึกและจัดทำเอกสารความต้องการทั้งหมดเสร็จสิ้นก่อนการออกแบบหรือการพัฒนาระบบ ซึ่งวิธีการนี้สามารถทำให้วิศวกรรมความต้องการชัดเจนและครอบคลุมตั้งแต่ช่วงเริ่มต้นของการพัฒนาระบบ ขณะที่แบบจำลองกระบวนการอย่างเช่น eXtreme Programming (Beck, 1999) จะทำกิจกรรมการได้มาซึ่งความต้องการเพียงส่วนที่ต้องการจะพัฒนา ดังนั้นวิศวกรรมความต้องการจำเป็นต้องดำเนินการอย่างต่อเนื่องและร่วมไปพร้อมกับกิจกรรมอื่นๆในการพัฒนาระบบ

บทบาทที่สำคัญของวิศวกรความต้องการ คือ การเป็นศูนย์กลางของความสนใจ กล่าวคือ มักเป็นบุคคลที่ติดต่อโดยตรงกับผู้มีส่วนได้ส่วนเสียทั้งหลาย ต้องมีความสามารถในการเข้าใจและคุ้นเคยกับโดเมนที่เกี่ยวข้องของระบบนั้นๆ ต้องเป็นผู้รับรู้และระบุแต่ละความต้องการของผู้มีส่วนได้ส่วนเสีย และสามารถถ่ายทอดให้กับนักพัฒนาระบบเข้าใจได้ วิศวกรความต้องการเป็นเสมือนล่ามที่เข้าใจโดเมนของระบบและลักษณะเฉพาะของระบบและมีความรู้ทางด้านเทคโนโลยีสารสนเทศเพียงพอที่จะตระหนักถึงปัญหาของนักพัฒนาระบบได้ ทั้งนี้เพื่อทำให้พวกเขาเหล่านั้นได้สื่อสารกันได้อย่างเข้าใจ

โดยทั่วไปประเภทของความต้องการสามารถแบ่งได้ 3 ประเภท คือ

(1) ความต้องการทางฟังก์ชัน (Functional Requirements) ได้แก่ ความต้องการที่ระบุว่าระบบทำอะไร มีหน้าที่อะไร มีบริการอะไร ระบบมีปฏิกิริยาตอบสนองอย่างไรต่อข้อมูลเข้าและส่งข้อมูลออกอย่างไร ระบบดำเนินการอย่างไรและไม่ทำอะไรในภาวะปกติ โดยทั่วไปแล้วสามารถมองเป็นความต้องการทางฟังก์ชันในแง่ของผู้ใช้งาน (User Functional Requirements) เป็นความต้องการที่ถูกรวบรวมขึ้นจากผู้ใช้และเป็นมุมมองที่ผู้ใช้ต้องการจากระบบ และสามารถมองเป็นความต้องการทางฟังก์ชันในแง่ของระบบ (System Functional Requirements) เป็นความต้องการที่อธิบายหน้าที่ของระบบอย่างละเอียด เช่น ข้อมูลเข้า ผลลัพธ์ ข้อมูลออก ข้อยกเว้น เป็นต้น

ลักษณะของความต้องการทางฟังก์ชัน คือ ก) มีการทำเป็นเอกสารในรูปแบบที่เรียกว่า ข้อกำหนดความต้องการซอฟต์แวร์ (Software Requirements Specification: SRS) ข) ใช้อธิบายการทำงานภายนอกที่คาดไว้ว่าระบบซอฟต์แวร์จะทำได้มากที่สุดหรือดีที่สุดอย่างไร และ ค) ถูกนำมาใช้ในการพัฒนา การทดสอบ การประกันคุณภาพ การจัดการโครงการ และการทำงานของโครงการที่สัมพันธ์กัน

(2) ความต้องการที่ไม่เป็นฟังก์ชัน (Non-Functional Requirements) หรือเรียกว่า ความต้องการเชิงคุณภาพ (Quality Requirements) ความต้องการเชิงคุณภาพเป็นความต้องการที่มักไม่เกี่ยวข้องกับหน้าที่โดยตรงของระบบย่อยระบบใด แต่เป็นการระบุคุณสมบัติบางอย่างของระบบ มักมีที่มาจาก 3 ที่ คือ ก) ความต้องการด้านผลผลิต (Products Requirement) เช่น ผลการดำเนินงานของซอฟต์แวร์ (Performance) ความน่าเชื่อถือได้ (Reliability) ข) ความต้องการด้านองค์กร (Organizational Requirement) เป็นความต้องการที่มาจากนโยบายในองค์กรของลูกค้าหรือผู้ผลิตซอฟต์แวร์ เช่น ภาษาที่ใช้ มาตรฐาน กระบวนการผลงานที่ต้องส่ง เป็นต้น ค) ความต้องการจากภายนอก (External Requirement) เป็นความต้องการที่เกิดจากปัจจัยภายนอกอื่นๆ เช่น ระบบจะต้องติดต่อกับระบบภายนอกอย่างไร ข้อกำหนดด้านกฎหมายและศีลธรรม เป็นต้น

ยกตัวอย่างความต้องการที่ไม่เป็นฟังก์ชันหรือความต้องการเชิงคุณภาพ เช่น มาตรฐาน ระเบียบและข้อตกลงร่วมกันถึงคุณภาพของซอฟต์แวร์ คำอธิบายเกี่ยวกับสิ่งรบกวนภายนอก ข้อกำหนดการปฏิบัติงานข้อบังคับในการออกแบบและการทำให้สำเร็จ คุณลักษณะทางคุณภาพ

โดยทั่วไปคุณสมบัติที่สามารถวัดความต้องการเชิงคุณภาพ ได้แก่ ความเร็ว (Speed) วัดด้วยเวลาในการตอบรับ (Responding time) ผลลัพธ์ (Throughput) วัดเป็นจำนวนงานต่อวินาที (job/sec) ขนาด (Size) วัดด้วยขนาดของโปรแกรม หน่วยวัด เช่น จำนวนบรรทัดของโปรแกรม ขนาดของข้อมูล หน่วยวัด เช่น Bytes ง่ายต่อการใช้งาน (Easiness) ตัวอย่างเช่น ดูว่าผู้ใช้เรียกให้ช่วยกี่ครั้ง เวลาที่ใช้ในการฝึก (Training time); ความน่าเชื่อถือ (Reliability) วัดเป็นเวลาเฉลี่ยในสภาวะล้มเหลว (Main time between failure) ความทนทาน (Durability) วัดด้วยการนับจำนวนครั้งที่ต้องการเริ่มระบบใหม่ (Restart) และ เคลื่อนย้ายได้ (Portability) วัดเป็นจำนวนสัดส่วนของคำสั่งที่ขึ้นอยู่กับระบบ

(3) ข้อจำกัด (Constraint) เป็นความต้องการที่จำกัดพื้นที่ของการแก้ปัญหาไม่ให้นอกเหนือไปจากสิ่งที่เป็นความต้องการทางฟังก์ชันและความต้องการเชิงคุณภาพ ได้แก่ ข้อจำกัดในการให้บริการหรือข้อจำกัดในหน้าที่ของระบบ ข้อจำกัดด้านเวลา ข้อจำกัดในการพัฒนาและมาตรฐาน เป็นต้น

วิธีการดำเนินการวิจัย

ในงานศึกษาวิจัยนี้ได้ดำเนินการสร้างแนวปฏิบัติงานวิศวกรรมความต้องการที่ดีและเหมาะสมเพื่อนำไปสู่ความสำเร็จของโครงการซอฟต์แวร์ การศึกษามีการทดลองดำเนินโครงการซอฟต์แวร์โดยเริ่มจากการตรวจสอบความรู้ของบุคลากรเพื่อจัดสรรเป็นทีมพัฒนาในโครงการซอฟต์แวร์ มีการจัดสรรทรัพยากรและความต้องการที่จำเป็นในกระบวนการวิศวกรรม

การวิจัยนี้มีขั้นตอนในการดำเนินการวิจัยรวม 2 ขั้นตอน ดังนี้

ขั้นตอนที่ 1 การศึกษากิจกรรมวิศวกรรมความต้องการในการพัฒนาโครงงานซอฟต์แวร์

ขั้นตอนที่ 2 การศึกษากิจกรรมวิศวกรรมความต้องการในการเปลี่ยนแปลงระบบซอฟต์แวร์ที่มีอยู่เดิม

ในการวิจัยได้มีการจัดตั้งทีมพัฒนาในโครงการซอฟต์แวร์โดยคัดสรรบุคลากรที่มีประสบการณ์ในการพัฒนาซอฟต์แวร์ขึ้น 2 ทีม แต่ละทีมพัฒนาจะประกอบด้วยนักวิเคราะห์ระบบ ผู้จัดการโครงการและนักพัฒนาซอฟต์แวร์ 4 คน การวิจัยครั้งนี้ดำเนินการทดลองให้มีการสร้างโครงการพัฒนาซอฟต์แวร์ 3 ชุดโดยที่มีความต้องการบางส่วนที่คล้ายกันและความต้องการบางส่วนที่แตกต่าง โครงการซอฟต์แวร์มีวัตถุประสงค์เพื่อสร้างซอฟต์แวร์ที่ตอบสนองความต้องการของผู้ใช้ที่หลากหลายในแง่ประโยชน์เชิงธุรกิจจึงได้มีรูปแบบออกมาเป็น 3 แบบ โดยกำหนดระยะเวลาของการดำเนินโครงการให้เสร็จสิ้นภายใน 6 เดือน โดยกำหนดให้

(1) ทีมพัฒนาที่หนึ่งดำเนินโครงการตามกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัด

(2) ทีมพัฒนาที่สองดำเนินโครงการโดยมุ่งเน้นการพัฒนาเขียนโค้ดและไม่เคร่งครัดในแง่ของวิศวกรรมความต้องการ

ขั้นตอนที่ 1 การศึกษากิจกรรมวิศวกรรมความต้องการในการพัฒนาโครงงานซอฟต์แวร์

ในการดำเนินโครงการผู้พัฒนาในทีมจะได้รับบทบาทที่แตกต่างกัน

- กิจกรรมแรกเริ่มต้นจากผู้พัฒนารวบรวมความต้องการทั้งหมดจากลูกค้าและสร้างเป็นข้อกำหนดความต้องการของผู้ใช้ ข้อกำหนดความต้องการจะเก็บรายละเอียดและใช้เป็นเครื่องมือในการอธิบายคุณสมบัติต่างๆของซอฟต์แวร์

- กิจกรรมต่อมาผู้พัฒนาทำการออกแบบสถาปัตยกรรมระบบและ ส่วนประกอบของระบบ โดยแสดงออกมาในรูปแบบจำลองข้อมูล แผนภาพคลาสไดอะแกรม แผนภาพลำดับ และแผนภาพกิจกรรม ซึ่งการออกแบบระบบใช้ข้อกำหนดความต้องการเป็นเครื่องมือสำคัญ

- กิจกรรมต่อมาคือ การพัฒนาโค้ดโดยมีเอกสารการออกแบบเป็นเครื่องมือสำคัญ

- กิจกรรมต่อมาคือ การทดสอบซอฟต์แวร์โดยทำตามเอกสารทดสอบและใช้การทดสอบหน่วยแต่ละหน่วยเพื่อหาความผิดพลาดในแต่ละจุด

- หลังจากนั้นนำทุกหน่วยองค์ประกอบมาประกอบเข้าด้วยกันและเริ่มการทดสอบแบบรวม

- เมื่อซอฟต์แวร์ผ่านการทดสอบจะถือเป็นผลิตภัณฑ์ซอฟต์แวร์ที่เสร็จสมบูรณ์และสามารถส่งต่อให้กับลูกค้า

ตารางที่ 1 ประเภทของบริบทความต้องการ

บริบทของความต้องการ	ประเภทความต้องการ	ลักษณะของบริบทเพื่อใช้แสดงหรืออธิบาย
การทดสอบการยอมรับระบบ	ความต้องการทางฟังก์ชัน ความต้องการเชิงคุณภาพ และข้อจำกัด	ลักษณะของระบบที่ผู้มีส่วนได้ส่วนเสียสนใจ
นิยามเงื่อนไขทางธุรกิจ	ความต้องการทางฟังก์ชัน	เงื่อนไขทางธุรกิจเป็นหลักการหรือนโยบายที่ระบบซอฟต์แวร์จำเป็นต้องสอดคล้อง
ข้อจำกัด	ข้อจำกัด	ข้อจำกัดในเรื่องต่างๆสำหรับการหาแนวทางเพื่อตอบสนองความต้องการได้ภายใต้ข้อจำกัดดังกล่าว
แผนภาพการไหลของข้อมูล	ความต้องการทางฟังก์ชัน	การเคลื่อนไหวของข้อมูลภายในระบบระหว่างส่วนต่างๆในระบบ ได้แก่ กระบวนการ แหล่งที่เก็บข้อมูล บุคคล
ระบบต้นแบบ	ความต้องการทางฟังก์ชัน ความต้องการเชิงคุณภาพ และข้อจำกัด	ส่วนติดต่อประสานงานกับผู้ใช้หลักๆ เพื่อแสดงให้เห็นแนวรูปแบบของระบบที่จะพัฒนา
ยูสเคส	ความต้องการทางฟังก์ชัน	การปฏิสัมพันธ์ระหว่างผู้ที่ติดต่อกับระบบและระบบโดยมีรายละเอียดลำดับของขั้นตอนที่ดำเนินงานในแต่ละฟังก์ชัน
คุณลักษณะ	ความต้องการทางฟังก์ชัน และความต้องการเชิงคุณภาพ	ลักษณะความสามารถของระบบที่มาจากมุมมองของผู้ใช้งานระบบ
ข้อกำหนดทางเทคนิค	ความต้องการทางฟังก์ชัน ข้อจำกัด	แง่มุมของระบบทางด้านเทคนิค
สถานการณ์ของการใช้งานระบบ	ความต้องการทางฟังก์ชัน	สถานการณ์หรือเงื่อนไขเหตุการณ์ที่มีผลต่อการดำเนินงานของระบบ
แผนภาพยูสเคส	ความต้องการทางฟังก์ชัน	ภาพรวมของยูสเคส ผู้ที่ติดต่อกับระบบ ความสัมพันธ์ระหว่างผู้ที่ติดต่อกับระบบและแต่ละยูสเคส และแสดงการจำลองบริบทของระบบที่ปฏิสัมพันธ์กับระบบภายนอก
เรื่องราวของผู้ใช้	ความต้องการทางฟังก์ชัน และความต้องการเชิงคุณภาพ	ข้อมูลการสนทนาระหว่างผู้มีส่วนได้เสียในโครงการ ข้อมูลความต้องการในระดับสูงรวมทั้งความต้องการเชิงพฤติกรรม เงื่อนไขและกฎเกณฑ์ทางธุรกิจ ข้อจำกัด และข้อกำหนด

การศึกษาวิจัยนี้ได้นำเสนอการประยุกต์ใช้เอกสารเพื่อสนับสนุนกิจกรรมต่างๆภายใต้กระบวนการซอฟต์แวร์ข้างต้น ประเภทของเอกสารที่ใช้มีดังนี้ ก) รายละเอียดยูสเคส ข) แผนภาพยูสเคส ค) แผนภาพคลาส ง) แผนภาพลำดับ จ) แผนภาพกิจกรรม ฉ) โค้ด ช) เอกสารการทดสอบ และ ซ) เอกสารมอธิบายมาตรฐานโค้ดและข้อมูลทางเทคนิค และในการศึกษานี้ได้เสนอให้มีการรวบรวมบริบทของความต้องการตามหลักการของวิศวกรรมความต้องการ เริ่มจากการที่นักพัฒนาซอฟต์แวร์จะได้รับรายละเอียดความต้องการและสร้างเป็นข้อกำหนดความต้องการพร้อมทั้งเอกสารประเภทอื่นๆ ซึ่งในทางปฏิบัติความต้องการอยู่ในรูปแบบที่หลากหลายในหลายบริบท เป็นสิ่งที่สำคัญและจำเป็นต้องรวบรวมระหว่างการดำเนินกิจกรรมวิศวกรรมซอฟต์แวร์ ซึ่งบริบทแต่ละประเภทใช้แสดงหรืออธิบายความต้องการในแง่มุมที่แตกต่างกันดังแสดงใน

ตารางที่ 1 และได้เสนอการประยุกต์ใช้เทคนิคการรวบรวมข้อมูลความต้องการโดยได้วิเคราะห์จุดแข็งและจุดอ่อนของแต่ละเทคนิคที่นำเสนอ ดังแสดงตามตารางที่ 2

ตารางที่ 2 เทคนิคที่ใช้เพื่อรวบรวมข้อมูล

เทคนิค	คำอธิบาย	จุดแข็ง	จุดอ่อน
การให้ผู้มีส่วนได้ส่วนเสียมีส่วนร่วมในการสร้างข้อกำหนดและแบบจำลองความต้องการ	จัดให้ผู้มีส่วนได้ส่วนเสียที่มีบทบาทต่างๆในระบบมีการแสดงความคิดเห็นและให้ข้อมูลความต้องการระบบร่วมกันเพื่อสร้างเป็นข้อกำหนดและแบบจำลองความต้องการ	- สนับสนุนให้ทำงานร่วมกันสูง - ผู้เชี่ยวชาญโดเมนมีส่วนร่วมในการกำหนดความต้องการของระบบ - ดำเนินการภายใต้ระยะเวลาที่กำหนดและเหมาะสม - การตัดสินใจทำได้ในเวลา	- ผู้มีส่วนได้ส่วนเสียจำนวนมากที่จำเป็นต้องเรียนรู้ทักษะการสร้างข้อกำหนดและแบบจำลองความต้องการ - ผู้มีส่วนได้ส่วนเสียบางคนไม่สามารถมีส่วนร่วมได้เต็มเวลาหรือมีเวลาที่จำกัด
สัมภาษณ์โดยตรง	เป็นการพบปะและหารือเกี่ยวกับความต้องการของผู้มีส่วนได้ส่วนเสียโดยตรงและในบทบาทต่างๆที่มีต่อระบบจะถูกสัมภาษณ์เพื่อให้ข้อมูลความต้องการ	- สนับสนุนให้มีการทำงานร่วมกัน - ในการสัมภาษณ์บางครั้งวิศวกรความต้องการสามารถเก็บรายละเอียดข้อมูลจำนวนมากได้อย่างรวดเร็วจากผู้มีส่วนได้ส่วนเสียเพียงคนเดียว	- การสัมภาษณ์จะต้องมีตารางเวลานัดหมายล่วงหน้า - ต้องอาศัยทักษะการสัมภาษณ์ซึ่งเป็นเรื่องยากที่จะเรียนรู้
ศึกษาจากเอกสารข้อมูลที่ได้บันทึกไว้	การศึกษาเอกสารข้อมูลเพื่อระบุความต้องการของระบบที่ควรครอบคลุม	- มีโอกาสที่จะได้เรียนรู้พื้นฐานของโดเมนระบบก่อนที่ติดต่อกับผู้มีส่วนได้ส่วนเสียโดยตรง	- ขาดการทำงานร่วมกัน - การปฏิบัติมักจะแตกต่างจากสิ่งที่บันทึกไว้ - จำกัดเฉพาะสำหรับนักพัฒนาที่สามารถอ่านและเข้าใจข้อมูล

ขั้นตอนที่ 2 การศึกษากิจกรรมวิศวกรรมความต้องการในการเปลี่ยนแปลงระบบซอฟต์แวร์ที่มีอยู่เดิม

การศึกษาครั้งนี้ดำเนินการต่อจากขั้นตอนที่ 1 จากการศึกษาได้ทดลองดำเนินโครงการซอฟต์แวร์ที่มีการพัฒนาซอฟต์แวร์ 3 ซอฟต์แวร์ (PM1, PM2 และ PM3) โดยซอฟต์แวร์แต่ละตัวมีความต้องการที่คล้ายกันและความต้องการบางส่วนที่แตกต่าง และในขั้นตอนนี้ได้ทดลองเพื่อจัดการกับสถานการณ์ที่มีการเปลี่ยนแปลงความต้องการ ซึ่งส่งผลให้มีการเปลี่ยนแปลงผลิตภัณฑ์ซอฟต์แวร์ที่ได้พัฒนาไปแล้ว

สิ่งที่เกิดขึ้น คือ วิศวกรความต้องการจำเป็นต้องประเมินความต้องการใหม่กับข้อกำหนดความต้องการที่มีอยู่เดิมว่าความต้องการใหม่หรือความต้องการที่เปลี่ยนแปลงไปเหล่านี้จะส่งผลกระทบต่อระบบหรือส่วนประกอบของระบบหรือไม่ อย่างไร ในการประเมินนี้วิศวกรความต้องการจำเป็นต้องให้ผู้มีส่วนได้ส่วนเสียที่มีบทบาทแตกต่างกันเป็นผู้มีส่วนร่วมในการประเมิน ตัวอย่างเช่น นักพัฒนาซอฟต์แวร์พิจารณารายละเอียดของความต้องการใหม่เปรียบเทียบกับเอกสารการออกแบบของระบบปัจจุบันเพื่อวิเคราะห์สถาปัตยกรรมระบบ ส่วนประกอบระบบ และรูปแบบข้อมูลที่สามารถนำกลับมาใช้ใหม่ได้ โดยพิจารณาจากยูสเคส แผนภาพยูสเคส แผนภาพคลาส แผนภาพลำดับ และแผนภาพกิจกรรมของระบบทั้งหมด เมื่อปรับเปลี่ยนเพิ่มเติมส่วนประกอบของระบบเพื่อให้สอดคล้องกับความต้องการใหม่ ต้องมีการทดสอบระบบตามเอกสารทดสอบโดยทดสอบแต่ละหน่วยแต่ละองค์ประกอบ เมื่อเสร็จสิ้นการทดสอบทุกองค์ประกอบชิ้นส่วนหรือองค์ประกอบทั้งหมดจะถูกรวมเข้าด้วยกัน และเริ่มการทดสอบรวมทั้งระบบ กิจกรรมเหล่านั้นทำเพื่อให้การจัดการเปลี่ยนแปลงซอฟต์แวร์ที่มีอยู่ให้สามารถสอดคล้องตามความต้องการใหม่ที่เปลี่ยนแปลงไป

ตารางที่ 3 จำนวนของเอกสารที่ถูกสร้างขึ้น

ประเภทเอกสาร	ทีม 1 ซอฟต์แวร์ PM1	ทีม 1 ซอฟต์แวร์ PM2	ทีม 1 ซอฟต์แวร์ PM3	ทีม 2 ซอฟต์แวร์ PM1	ทีม 2 ซอฟต์แวร์ PM2	ทีม 2 ซอฟต์แวร์ PM3
	จำนวนเอกสาร	จำนวนเอกสาร	จำนวนเอกสาร	จำนวนเอกสาร	จำนวนเอกสาร	จำนวนเอกสาร
ก)	1	1	1	0	0	0
ข)	1	1	1	0	0	0
ค)	1	0	0	0	0	0
ง)	1	1	1	1	1	1
จ)	4	6	3	4	6	3
ฉ)	1	1	1	1	1	1
ช)	3	5	1	0	0	0
ซ)	4	6	3	0	0	0

ตารางที่ 4 จำนวนบรรทัดและหน่วยฟังก์ชั่น และร้อยละของการเปลี่ยนแปลงโมดูลหรือหน่วยย่อยของโค้ด และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลง

ทีม	ซอฟต์แวร์	จำนวนบรรทัด (line of code)	หน่วยฟังก์ชั่น	ร้อยละของการเปลี่ยนแปลงโมดูลหรือหน่วยย่อยของโค้ด	เวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลง
1	PM1	7,689	10	25	15.5
	PM2	2,280	5	0	0
	PM3	7,280	4	15	8
	รวม	17,249	19	40	23.5
2	PM1	6,830	10	25	32
	PM2	5,420	5	0	0
	PM3	8,845	4	15	12
	รวม	21,095	19	40	44

สรุปผลการวิจัย

จากการทดลองศึกษานี้ขั้นตอนที่ 1 พบว่ากิจกรรมต่างๆภายใต้โครงการซอฟต์แวร์ที่แต่ละทีมต้องมีการพัฒนาซอฟต์แวร์ 3 ชุด มีการสร้างเอกสารซอฟต์แวร์ขึ้นเป็นจำนวนมากและหลากหลาย เอกสารซอฟต์แวร์ที่ได้ถูกประยุกต์ใช้เพื่อวิศวกรรมความต้องการและวิศวกรรมซอฟต์แวร์ส่วนอื่นๆ ดังแสดงในตารางที่ 3 มีดังนี้ ก) เอกสารวิชั่น (vision document) ข) เอกสารข้อกำหนดระบบ (system specification document) ค) เอกสารคำนิยามระบบ (glossary) ง) แผนภาพยูสเคส (use case diagram) จ) คำอธิบายยูสเคส (use case description) ฉ) แผนภาพคลาส (class diagram) ช) แผนภาพกิจกรรม (activity diagram) และ ซ) แผนภาพลำดับ (sequence diagram) ซึ่งเอกสารเหล่านี้ได้ถูกสร้างขึ้นเพื่อเก็บรวบรวมข้อมูลความต้องการในบริบทและแง่มุมที่แตกต่างกันจากแหล่งที่มาและผู้มีส่วนได้ส่วนเสียต่างๆ ข้อมูลเหล่านั้น ได้แก่ การทดสอบการยอมรับระบบ นิยามเงื่อนไขทางธุรกิจ ข้อจำกัด แผนภาพการไหลของข้อมูล ระบบต้นแบบยูสเคส คุณลักษณะ ข้อกำหนดทางเทคนิค สถานการณ์ของการใช้งานระบบ แผนภาพยูสเคส และ เรื่องราวของผู้ใช้ รายละเอียดของจำนวนของเอกสารที่ถูกสร้างขึ้นจากขั้นตอนที่ 1 ดังแสดงในตารางที่ 3

นอกจากนี้ได้ประยุกต์ใช้ตัววัดขนาดของโปรแกรม 2 แบบ คือ จำนวนบรรทัด (line of code) และหน่วยฟังก์ชั่น (function point) รายละเอียดจำนวนบรรทัดและหน่วยฟังก์ชั่นแสดงในตารางที่ 4

นอกจากนี้ในการศึกษาทดลองขั้นตอนที่ 2 กรณีที่มีการเปลี่ยนแปลงความต้องการ ตัวชี้วัดคุณภาพของซอฟต์แวร์ที่สำคัญเกี่ยวข้องกับการจัดการการเปลี่ยนแปลง คือ ความพยายามที่ใช้ในการแก้ไขโค้ด โดยในการศึกษาใช้ร้อยละของการเปลี่ยนแปลงโมดูลหรือหน่วยย่อยของโค้ด และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงเป็นตัววัด ซึ่งพบว่าร้อยละของการเปลี่ยนแปลงโค้ดในซอฟต์แวร์ตัวที่หนึ่ง (PM1) ตัวที่สอง (PM2) และตัวที่สาม (PM3) คิดเป็น 25 0 และ 15 ตามลำดับ และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่หนึ่ง (PM1) จากทีมที่ 1 ใช้เวลาเป็น 15.5 วัน เวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่สอง (PM2) ใช้เวลาเป็น 0 วันเนื่องจากไม่มีการเปลี่ยนแปลงใดๆ และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่สาม (PM3) ใช้เวลาเป็น 8 วัน และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่หนึ่ง (PM1) จากทีมที่ 2 ใช้เวลาเป็น 32 วัน เวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่สอง (PM2) ใช้เวลาเป็น 0 วันเนื่องจากไม่มีการเปลี่ยนแปลงใดๆ และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่สาม (PM3) ใช้เวลาเป็น 12 วัน ดังแสดงในตารางที่ 4

นอกจากนี้จากการสำรวจความพึงพอใจของทีมพัฒนาพบว่า นักพัฒนามีความพึงพอใจในกระบวนการที่เน้นการเขียนโค้ดมากกว่าการสร้างและบันทึกเอกสาร และนอกจากนี้นักพัฒนาบางส่วนยังไม่เห็นด้วยกับกิจกรรมติดตามที่ต้องปรับปรุงแก้ไขเอกสารที่เกี่ยวข้องทุกครั้งเมื่อมีการเปลี่ยนแปลงความต้องการเกิดขึ้น จากการสำรวจพบว่าร้อยละ 33 ของนักพัฒนามีแนวโน้มที่จะต่อต้านการปฏิบัติตามแนวทางวิศวกรรมความต้องการในขณะที่ร้อยละ 70 ของนักพัฒนามีความคิดเชิงบวกต่อแนวทางปฏิบัติที่ดีของวิศวกรรมความต้องการ โดยเฉพาะอย่างยิ่งร้อยละ 82 ของนักพัฒนามีความพึงพอใจกับขั้นตอนการบำรุงรักษาและการจัดการการเปลี่ยนแปลงที่ได้ดำเนินกิจกรรมภายใต้วิศวกรรมความต้องการที่ดีและเหมาะสม

อภิปรายผลการวิจัยและข้อเสนอแนะ

งานวิจัยนี้ได้นำเสนอการทดลองดำเนินโครงการซอฟต์แวร์โดยแบ่งออกเป็น 2 ขั้นตอน ขั้นตอนที่ 1 คือ การศึกษากิจกรรมวิศวกรรมความต้องการในการพัฒนาโครงงานซอฟต์แวร์ และขั้นตอนที่ 2 คือ การศึกษากิจกรรมวิศวกรรมความต้องการในการเปลี่ยนแปลงความต้องการระบบซอฟต์แวร์ที่มีอยู่เดิม โดยจัดให้มีทีมพัฒนา 2 ทีม แต่ละทีมประกอบด้วยนักวิเคราะห์ระบบ ผู้จัดการโครงการ และนักพัฒนาซอฟต์แวร์ 4 คน

สำหรับขั้นตอนที่ 1 ให้มีการสร้างโครงการพัฒนาซอฟต์แวร์ 3 ชุดโดยที่มีความต้องการบางส่วนที่คล้ายกันและความต้องการบางส่วนที่แตกต่าง โดยกำหนดให้ทีมพัฒนาที่หนึ่งดำเนินโครงการตามกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัด และทีมพัฒนาที่สองดำเนินโครงการโดยมุ่งเน้นการพัฒนาเขียนโค้ดและไม่เคร่งครัดในแง่ของวิศวกรรมความต้องการ จากการทดลองพบว่า จำนวนเอกสารที่ถูกสร้างขึ้นจากทีมพัฒนาที่หนึ่งมีเอกสาร 48 เอกสารและจากทีมพัฒนาที่สองมี 19 เอกสาร เนื่องจากทีมที่หนึ่งดำเนินกระบวนการซอฟต์แวร์โดยมีการปฏิบัติตามหลักการวิศวกรรมความต้องการอย่างเคร่งครัด มีการสร้างเอกสารซอฟต์แวร์และข้อกำหนดความต้องการอย่างละเอียดและครบถ้วน ในขณะที่ทีมที่สองดำเนินกระบวนการซอฟต์แวร์โดยมุ่งเน้นการเขียนโค้ดมากกว่า ทำให้ละเลยการสร้างเอกสารซอฟต์แวร์บางอย่างไป กล่าวคือ จำนวนเอกสารที่ทีมพัฒนาที่หนึ่งสร้างมีมากกว่าเอกสารที่ทีมพัฒนาที่สองสร้างโดยคิดเป็น 2.5 เท่าของทีมสอง

ขนาดของซอฟต์แวร์ที่ผลิตขึ้นจากทั้งสองทีมโดยวัดจากหน่วยฟังก์ชันมีเท่ากัน แต่หากพิจารณาจำนวนบรรทัดของโค้ดโดยรวมที่ทีมที่หนึ่งพัฒนาขึ้นมี 17,249 บรรทัดขณะที่ทีมที่สองมี 21,095 บรรทัด กล่าวคือ จำนวนบรรทัดของโปรแกรมที่ทีมพัฒนาที่หนึ่งสร้างน้อยกว่าที่ทีมพัฒนาที่สองสร้างคิดเป็น 0.82 เท่าของทีมที่สอง ซึ่งไม่ถือว่าแตกต่างกันมาก แต่อย่างไรก็ตามจากการสำรวจความพึงพอใจภายหลังพบว่า

นักพัฒนาซอฟต์แวร์จากทีมที่หนึ่งได้ประยุกต์ใช้เอกสารซอฟต์แวร์ที่ได้สร้างไว้อย่างครอบคลุมและสมบูรณ์เป็นเครื่องมือในการดำเนินกิจกรรมต่างๆภายใต้กระบวนการซอฟต์แวร์จึงทำให้สามารถเข้าใจและระบุความต้องการที่ถูกกำหนดไว้ได้อย่างรวดเร็วและถูกต้อง และสามารถนำองค์ประกอบซอฟต์แวร์บางส่วนกลับมาใช้ใหม่ทำให้ลดปริมาณบรรทัดของโปรแกรมที่ต้องพัฒนาขึ้นใหม่

สำหรับขั้นตอนที่ 2 ให้มีการเปลี่ยนแปลงความต้องการ ซึ่งพบว่า ร้อยละของการเปลี่ยนแปลงโค้ดในซอฟต์แวร์ 3 ชุดของทีมที่หนึ่งและทีมที่สองเท่ากัน คิดเป็นร้อยละเฉลี่ย 13.33 แต่อย่างไรก็ตามเวลาเฉลี่ยรวมในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ 3 ชุดจากทีมที่หนึ่งใช้เวลาเป็น 23.5 วัน ขณะที่จากทีมที่สองใช้เวลาเป็น 44 วัน ทีมที่หนึ่งใช้เวลาน้อยกว่าถึง 0.53 เท่าของทีมที่สอง

การดำเนินกิจกรรมภายใต้หลักการของวิศวกรรมความต้องการทำให้เกิดการสร้างเอกสารซอฟต์แวร์มากขึ้นและสำหรับผู้ที่ไม่คุ้นเคยอาจไม่เคยชินกับการดำเนินตาม ทั้งนี้พิจารณาจากการสำรวจความพึงพอใจของนักพัฒนาซอฟต์แวร์มีความพึงพอใจในกระบวนการที่เน้นการเขียนโค้ดมากกว่าการสร้างและบันทึกเอกสารและบางส่วนยังไม่เห็นด้วยกับกิจกรรมติดตามที่ต้องปรับปรุงแก้ไขเอกสารที่เกี่ยวข้องทุกครั้งเมื่อมีการเปลี่ยนแปลงความต้องการเกิดขึ้น ร้อยละ 33 ของนักพัฒนามีแนวโน้มที่จะต่อต้านการปฏิบัติตามแนวทางวิศวกรรมความต้องการ

แต่อย่างไรก็ตามเมื่อมีการเปลี่ยนแปลงแก้ไขความต้องการภายหลังเอกสารซอฟต์แวร์ต่างๆรวมถึงข้อกำหนดความต้องการได้กลายเป็นเครื่องมือที่สำคัญและเป็นประโยชน์ทำให้สามารถดำเนินการแก้ไขได้อย่างรวดเร็วและมีประสิทธิภาพ จากการสำรวจความพึงพอใจพบว่าร้อยละ 70 ของนักพัฒนาซอฟต์แวร์มีความคิดเชิงบวกที่จะดำเนินการตามแนวทางปฏิบัติที่ดีของวิศวกรรมความต้องการ และพบว่าร้อยละ 82 ของนักพัฒนามีความพึงพอใจกับขั้นตอนการบำรุงรักษาและการจัดการการเปลี่ยนแปลงที่ได้ดำเนินกิจกรรมภายใต้วิศวกรรมความต้องการที่ดีและเหมาะสม

บทสรุป

กิจกรรมต่างๆภายใต้วิศวกรรมความต้องการเป็นงานสำคัญของการพัฒนาระบบซอฟต์แวร์จำเป็นต้องมีความสามารถในการจัดการกับปัจจัยและตัวแปรต่างๆที่อาจเกิดขึ้นในการพัฒนา การได้มาซึ่งความต้องการ การรวบรวมความต้องการ การสร้างข้อกำหนดความต้องการ และการทวนสอบความต้องการกิจกรรมเหล่านี้เกิดขึ้นภายใต้โครงสร้างที่ซับซ้อนและมีผู้มีส่วนได้ส่วนเสียหลายหลาย

ความสำเร็จของโครงการซอฟต์แวร์เกิดขึ้นได้จากหลายปัจจัยในแง่ของวิศวกรรมความต้องการมีดังนี้

(1) ประสบการณ์ของผู้ดำเนินโครงการอย่างวิศวกรความต้องการเป็นปัจจัยหนึ่งที่มีความสำคัญกิจกรรมภายใต้วิศวกรรมและการจัดการความต้องการมักเกิดขึ้นในบริบทหรือขั้นตอนต่างๆของการพัฒนาซอฟต์แวร์ ผู้ดำเนินโครงการที่มีประสบการณ์สูงสามารถดำเนินโครงการและแก้ไขปัญหาที่เกิดขึ้นในกิจกรรมของการพัฒนาได้ดีกว่าผู้ดำเนินโครงการที่ขาดประสบการณ์

(2) คุณภาพของข้อกำหนดความต้องการที่ดีมีผลเชิงบวกต่อการดำเนินโครงการ หมายถึง หากมีการรวบรวมความต้องการได้ครบถ้วนและถูกต้อง และมีการสร้างข้อกำหนดความต้องการได้ถูกต้องเหมาะสมและครอบคลุม จะสามารถทำให้ผู้ดำเนินโครงการอย่างนักพัฒนาซอฟต์แวร์สามารถนำข้อมูลไปใช้ต่อยอดในกิจกรรมการพัฒนาลำดับถัดๆไปโดยเฉพาะอย่างยิ่งหากการพัฒนาซอฟต์แวร์นั้นประยุกต์ใช้แบบจำลองน้ำตกในกิจกรรมวิศวกรรมซอฟต์แวร์ยิ่งต้องอาศัยข้อกำหนดความต้องการที่สมบูรณ์จึงจะพัฒนางานต่อไปได้ นอกจากนี้ข้อกำหนดความต้องการที่มีความถูกต้อง ครบถ้วน และสอดคล้องกันสามารถสนับสนุนกิจกรรมวิศวกรรมความต้องการอื่นๆได้ดีกว่าข้อกำหนดความต้องการที่ไม่ถูกต้อง ไม่ครบถ้วน หรือไม่สอดคล้องกัน

การบำรุงรักษาข้อกำหนดความต้องการและเอกสารซอฟต์แวร์ให้มีคุณภาพดีจำเป็นต้องให้มีความถูกต้อง สมบูรณ์ และสอดคล้องกัน เป็นสิ่งที่ต้องใช้ความพยายามซึ่งมีต้นทุนเป็นคนและเวลาความพยายามที่ใช้เพื่อแก้ไขระบบซอฟต์แวร์ให้สอดคล้องกับความต้องการที่เปลี่ยนแปลงไปนั้นแปรผันตาม

ความสามารถในการระบุชิ้นส่วนหรือองค์ประกอบซอฟต์แวร์ที่มีผลกระทบจากความต้องการที่เปลี่ยนแปลงไปนั้นๆ เช่น การระบุองค์ประกอบซอฟต์แวร์ที่ต้องเปลี่ยนแปลงในเอกสารการออกแบบและโค้ดจากการที่ข้อกำหนดความต้องการเปลี่ยน ทั้งนี้ต้องให้ครอบคลุมความต้องการทุกประเภทและทุกแง่มุม ตัวอย่างเช่น ความต้องการเชิงพฤติกรรมจะครอบคลุมวิธีการใช้งานและการที่ผู้ใช้จะมีปฏิสัมพันธ์กับระบบ การที่ผู้ใช้จะเกี่ยวข้องส่วนติดต่อผู้ใช้ รูปแบบและวิธีการที่ผู้ใช้จะใช้ระบบ การที่ระบบจะดำเนินงานตอบสนองเงื่อนไขหรือกฎเกณฑ์ทางธุรกิจ ส่วนความต้องการเชิงคุณภาพจะหมายถึงความต้องการทุกด้านที่ไม่ใช่ฟังก์ชัน ได้แก่ การรักษาความปลอดภัยของระบบ การทำให้ระบบมีประสิทธิภาพและมีความน่าเชื่อถือ เช่น ความเร็วที่คาดหวังในการเข้าถึงข้อมูล ซึ่งวัดได้จากเวลาตอบสนองของส่วนติดต่อผู้ใช้ การควบคุมการเข้าถึง สิ่งที่สำคัญคือการระบุและเข้าใจความต้องการให้ถูกต้อง ครอบคลุมและสมบูรณ์ เพื่อสามารถจัดการได้อย่างถูกต้องและเหมาะสมมิเช่นนั้นอาจกลายเป็นปัญหาภายหลังได้

บรรณานุกรม

Achimugu, P., et. al. (2014). *A Systematic Literature Review of Software Requirements Prioritization Research.* Information and Software Technology, *56*(6), p. 568-585.

Alves, V., et al. (2010). Requirements engineering for software product line: A systematic literature review. *Journal of Information and Software Technology*, *52*(8), p. 806-820.

Al-Ani, et al. (2013). *Globally distributed system developers: Their trust expectations and processes.* In Computer supported cooperative work (CSCW), pp. 563–573.

Beck, K. (1999). *Extreme Programming Explained-Embrace Change*. MA: Addison-Wesley.

Bjarnason, E., Wnuk, K. & Regnell, B. (2011b). *Requirements are slipping through the gaps—A case study on causes & effects of communication gaps in large-scale software development.* In 2011 IEEE 19th International Requirements Engineering Conference, pp. 37–46.

Cockburn, A. (2001). *Writing Effective Use Cases*. MA: Addison-Wesley.

Chen, L., et al. (2013). Characterizing Architecturally Significant Requirements. *IEEE Software, 30*(2), pp. 38-45.

Fernandez, D. & Wieringa, R. (2013). Improving Requirements Engineering by Artefact Orientation. *Product-Focused Software Process Improvement.* Lecture Notes in Computer Science, *7983*, pp. 108-122.

Fowler et al. (2011). *Refactoring: Improving the design of existing code.* MA: Addison Wesley.

Hoda et al. (2011). *The impact of inadequate customer collaboration on self-organizing agile teams.* Information and Software Technology, *53*(5), pp. 521–534.

Huang, J., Gotel, O., & A. Zisman (Eds.). (2012). *Software and Systems Traceability*. London: Springer. ISBN: 978-1-4471-2239-5.

Humphrey, W. S. (2005). *PSP: A Self-Improvement Process for Software Engineers.* MA: Addison Wesley. ISBN: 0-321-30549-3.

IEEE std. 610.12. (1990). *Institute of Electrical and Electronics Engineers: IEEE Standard Glossary of Software Engineering Terminology (IEEE std. 610.12-1990).* New York: IEEE.

Ingram, C., & Riddle, S. (2012). *Cost-Benefits of Traceability (chapter).* Software and Systems Traceability. London: Springer. ISBN: 978-1-4471-2239-5.

Kang, K., S., et. al. (1990). *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Karlsson, L., et al. (2007). Requirements engineering challenges in market-driven software development – An interview study with practitioners. *Information and Software Technology. 49*(6), pp. 588-604.

Laplante, P. (2013). *Requirements Engineering for Software and Systems, Second Edition* FL: CRC Press. ISBN: 1466560819, 9781466560819.

Lemos, R., et al. (2010). Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. *Software Engineering for Self-Adaptive Systems II*. Lecture Notes in Computer Science, *7475*, pp. 1-32.

Macaulay, L. (1993). Requirements Capture as a Cooperative Activity. In: *Proceedings of the 1st IEEE International Symposium on Requirements Engineering*, pp. 174-181. San Diego, CA: IEEE.

Pohl, L. (1996). *Process-Centered Requirements Engineering*. Somerset: Research Study Press.

Pohl, K., & Rupp, C. (2011). *Requirements Engineering Fundamentals*. CA: Rocky Nook Inc.

Royce, W. (1987). Managing the Development of Large Software Systems. In: *Proceedings of the 9th International Conference on Software Engineering (ICSE'87)*, pp. 328-338. Los Alamitos: IEEE.

Ruiz-Lopez, T. et al. (2013). REUBI: A Requirements Engineering Method for Ubiquitous Systems. *Science of Computer Programming, 78*(10), pp. 1751-2026.

Schneider, F. & Berenbach, B. (2013). *A Literature Survey on International Standards for Systems Requirements Engineering.* In: 2013 Conference on Systems Engineering Research, *16*, pp. 798-805.

V-Model. (2004). *The V-Model.* Bundesrepublik Deutschland, Vorgehensmodell. Retrieved May 3, 2014, from http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.1-eng/Dokumentation/pdf/V-Modell-XT-eng-Teil1.pdf

Wieringa, R. (2014). Empirical research methods for technology validation: Scaling up to practice. *Journal of Systems and Software*, *95*, pp. 19–31.

Winkler, S. & Pilgrim, J. (2010). A Survey of Traceability in Requirements Engineering and Model-driven Development. *Journal Software and Systems Modeling, 9*(4), pp.529-565.