การเปรียบเทียบการพัฒนาซอฟต์แวร์ภายใต้หลักวิศวกรรมความต้องการอย่างเคร่งครัดกับแบบมุ่งเน้นการพัฒนาโปรแกรม

A Comparative Study between Requirements Engineering Principles and Programming-Oriented Approach

วราพร จิระพันธุ์ทอง*
Waraporn Jirapanthong
คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยธุรกิจบัณฑิตย์
110/1-4 ถนนประชาชื่น หลักสี่ กรุงเทพมหานคร 10210

บทคัดย่อ

งานวิจัยนี้มีจุดประสงค์เพื่อศึกษาการดำเนินโครงการซอฟต์แวร์โดยเปรียบเทียบการดำเนินกิจกรรมตามหลักวิศวกรรมความต้องการอย่างเคร่งครัดและแบบมุ่งเน้นการเขียนโปรแกรมเป็นหลัก การทดลองแบ่งออกเป็น 2 ส่วน คือ การศึกษากิจกรรมวิศวกรรมความต้องการในการพัฒนาโครงงานซอฟต์แวร์ใหม่และการเปลี่ยนแปลงระบบซอฟต์แวร์ที่มีอยู่เดิมตามความต้องการใหม่ โดยจัดให้มีทีมพัฒนา 2 ทีม ส่วนที่ 1 ให้แต่ละทีมพัฒนาซอฟต์แวร์ใหม่ 3 ชุด โดยกำหนดให้ทีมพัฒนาที่หนึ่งดำเนินโครงการตามกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัดและทีมพัฒนาที่สองดำเนินโครงการโดยมุ่งเน้นการพัฒนาเขียนโปรแกรมและไม่เคร่งครัดในแง่ของวิศวกรรมความต้องการ พบว่าจำนวนเอกสารที่ทีมที่หนึ่งสร้างมีมากกว่าคิดเป็น 2.5 เท่าของทีมที่สอง และขนาดของซอฟต์แวร์ที่พิจารณาจากจำนวนบรรทัดของโปรแกรมที่ทีมที่หนึ่งพัฒนาขึ้นน้อยกว่าที่ทีมพัฒนาที่สองคิดเป็น 0.82 เท่า และค่าเฉลี่ยของความถูกต้องของโปรแกรมที่ได้จากสองทีมเป็น 4.5 และ 4.8 ตามลำดับ จากคะแนนเต็ม 5 ซึ่งแตกต่างกันเพียงเล็กน้อย 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 strictly and non-strictly. There are two steps of study: Step 1 is to study the activities of software projects development underlying requirements engineering. Step 2 is to study the activities of change management on requirements. By providing two development teams,

*E-mail address : waraporn.jir@dpu.ac.th โทรศัพท์ 029547300 ต่อ 215

each were to implement software projects with three sets of requirements. The first team was asked to strictly follow the activities of requirements engineering process while the second one was asked to focus on coding rather than strictly following the rules set forth by requirements engineering. We found the number of documents that the first team created more than 2.5 times that created by the second team. On the other hand, the number of lines of code is less than the second team's by 0.82 times. The average values of program correctness of two teams slightly differ. They are 4.5 and 4.8, respectively, of 5. Moreover, the average time that the first team spent for implementing software changes was less than the second team' s by 0.53 times.

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

1. บทนำ

วิศวกรรมความต้องการเป็นศาสตร์ความรู้ที่เสนอแนวทางปฏิบัติที่สนับสนุนกิจกรรมภายใต้โครงการซอฟต์แวร์ และเป็นปัจจัยสำคัญที่จัดการปัญหาอันเป็นสาเหตุของความล้มเหลวของโครงการ แต่อย่างไรก็ตามเนื่องจากข้อจำกัดเรื่องของเวลาในการดำเนินโครงการซอฟต์แวร์เพื่อตอบสนองความต้องการของลูกค้าที่เปลี่ยนแปลงอย่างรวดเร็วและมีความต้องการที่หลากหลายมากขึ้นในปัจจุบัน ทำให้นักพัฒนาซอฟต์แวร์ถูกเร่งรัดด้วยเงื่อนไขและปัจจัยหลายด้าน จึงเป็นไปได้ที่จะเพิกเฉยหรือไม่ให้ความสำคัญกับกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัด งานวิจัยนี้มีจุดประสงค์เพื่อศึกษาทดลองการดำเนินโครงการซอฟต์แวร์โดยเปรียบเทียบการดำเนินโครงการซอฟต์แวร์ที่เคร่งครัดตามหลักวิศวกรรมความต้องการและการดำเนินโครงการซอฟต์แวร์โดยมุ่งเน้นการเขียนโค้ดและไม่เคร่งครัดต่อกิจกรรมภายใต้วิศวกรรมความต้องการ วัตถุประสงค์ของการวิจัย คือ 1) เพื่อศึกษาข้อดีและข้อเสียของกิจกรรมภายใต้วิศวกรรมความต้องการ และ 2) เพื่อศึกษาบริบทของความต้องการที่สำคัญในการสร้างข้อกำหนดความต้องการที่สมบูรณ์

2. แนวคิดและวรรณกรรมที่เกี่ยวข้อง

วิศวกรรมความต้องการยังให้ความสำคัญกับคำนิยามสำหรับผู้มีส่วนได้ส่วนเสีย (Stakeholder) เนื่องจากผู้มีส่วนได้ส่วนเสียเป็นแหล่งที่มาของความต้องการที่สำคัญที่สุด การหลงลืมหรือขาดผู้มีส่วนได้ส่วนเสียคนหนึ่งอาจมีผลทำให้ความต้องการไม่สมบูรณ์หรือขาดตอนได้ [1]

ผู้มีส่วนได้ส่วนเสีย หมายถึง บุคคลหรือหน่วยงานที่มีผลกระทบต่อความต้องการ บุคคลที่ปฏิสัมพันธ์กับระบบ เช่น ผู้ใช้งานระบบ หรือ เจ้าหน้าที่บำรุงรักษาระบบ เป็นต้น บุคคลบางกลุ่มที่แทบจะไม่มีความสนใจหรือเกี่ยวข้องกับระบบโดยตรง เช่น ผู้จัดการระบบ ผู้ที่แอบขโมยข้อมูล ผู้มีส่วนได้ส่วนเสียของระบบคู่แข่ง เป็นต้น บุคคลเหล่านี้อาจจะถูกพิจารณาให้เป็นแหล่งที่มาหรือนิยามความต้องการของระบบได้ ผู้มีส่วนได้ส่วนเสีย เช่น ผู้ใช้งานระบบ นักเขียนโปรแกรม นักวิเคราะห์ระบบ ผู้จัดการโครงงาน

วิศวกรซอฟต์แวร์ ผู้ดูแลฐานข้อมูล เป็นต้น [2] ได้นิยามไว้ว่า ผู้มีส่วนได้ส่วนเสียของระบบ (Stakeholder of a system) คือ บุคคลหรือหน่วยงานที่มีอิทธิพลต่อความต้องการของระบบทั้งทางตรงและทางอ้อม

เป้าหมายของวิศวกรรมความต้องการ คือ เพื่อทำให้กิจกรรมในกระบวนการพัฒนาระบบบรรลุล่วงได้ด้วยดี กิจกรรมเหล่านั้น ได้แก่ การดึงความต้องการของผู้มีส่วนได้ส่วนเสียออกมา การบันทึกความต้องการเหล่านั้นในรูปแบบที่เหมาะสม การทวนสอบและการตรวจสอบความสมเหตุสมผลของความต้องการ และการบริหารจัดการความต้องการตลอดวงจรชีวิตของระบบ [3]

กิจกรรมหลักในกระบวนการวิศวกรรมความต้องการประกอบด้วย 4 กิจกรรม ดังนี้ 1) *การทำให้ได้มาซึ่งความต้องการ (Elicitation)* เริ่มตั้งแต่การศึกษาความเป็นไปได้โดยรวบรวมข้อมูลจากผู้มีส่วนได้ส่วนเสียและแหล่งข้อมูลต่างๆที่เกี่ยวข้อง การวิเคราะห์ขั้นตอนการทำงานของผู้ใช้ การตรวจสอบรายงานปัญหาที่เกิดจากระบบต่างๆที่มีอยู่และเกี่ยวข้อง การพิจารณาความต้องการของระบบอื่น การพิจารณาข้อมูลต่างๆที่เกี่ยวข้อง 2) *การบันทึกและจัดทำเอกสาร* เป็นขั้นตอนของการระบุความต้องการในรายละเอียด 3) *การตรวจสอบความต้องการ* เป็นการตรวจสอบผ่านเอกสารข้อกำหนดความต้องการ เพื่อตรวจสอบความถูกต้อง ประสิทธิภาพ ความพึงพอใจของผู้ใช้ และ 4) *การบริหารจัดการความต้องการ* เป็นบริหารจัดการการเปลี่ยนแปลงของความต้องการที่เกิดขึ้นและประเมินผลกระทบที่อาจเกิดขึ้นจากความต้องการที่เปลี่ยนแปลงไป การบริหารจัดการความต้องการที่ดีจะต้องสามารถทำการทดสอบและสืบค้นสถานะของความต้องการที่เปลี่ยนแปลงไปได้

แบบจำลองกระบวนการอย่างเช่น แบบจำลองน้ำตก [4] หรือแบบจำลองวี [5] ได้วางแบบให้การได้มาซึ่งความต้องการและการบันทึกและจัดทำเอกสารความต้องการทั้งหมดเสร็จสิ้นก่อนการออกแบบหรือการพัฒนาระบบ ซึ่งวิธีการนี้สามารถทำให้วิศวกรรมความต้องการชัดเจนและครอบคลุมตั้งแต่ช่วงเริ่มต้นของการพัฒนาระบบ ขณะที่แนวทางการพัฒนาซอฟต์แวร์ที่เกิดขึ้นใหม่อย่างเอจาย (Agile) อย่างเช่น eXtreme Programming [6] จะทำกิจกรรมการได้มาซึ่งความต้องการเพียงส่วนที่ต้องการจะพัฒนา ดังนั้นวิศวกรรมความต้องการจำเป็นต้องดำเนินการอย่างต่อเนื่องและร่วมไปพร้อมกับกิจกรรมอื่นๆในการพัฒนาระบบ

การใช้ Case Tool และการควบคุมที่ดีจะสามารถช่วยกระบวนการพัฒนาซอฟต์แวร์ได้ดีขึ้นกว่าเดิม ในทางกลับกันเครื่องมือและวิธีการเหล่านี้อาจทำให้เกิด Overhead อย่างมากแก่โครงการพัฒนาซอฟต์แวร์ขนาดกลางและขนาดเล็ก แนวความคิดแบบเอจายจึงถูกพัฒนาขึ้นโดยมีลักษณะที่สำคัญ คือ ก) ให้ความสำคัญกับซอฟต์แวร์มากกว่าการทำเอกสาร ข) ดำเนินการพัฒนาโดยทำวนซ้ำ ค) เน้นกระบวนการส่งมอบที่รวดเร็วและเพื่อรองรับการพัฒนาซอฟต์แวร์ที่ความต้องการเปลี่ยนแปลงอย่างรวดเร็วระหว่างกระบวนการพัฒนา เป้าหมายของเอจาย คือการส่งมอบซอฟต์แวร์ที่ทำงานได้จริงอย่างเร็วให้กับลูกค้าและต่อมาลูกค้าสามารถนำเสนอความต้องการใหม่หรือการเปลี่ยนแปลงได้ เหมาะกับระบบธุรกิจขนาดเล็กหรือกลาง [7]

สครัม (Scrum) [8-9] เป็นกระบวนการพัฒนาที่อยู่บนพื้นฐานของสปรินท์ (Sprint) ที่พัฒนาผลลัพธ์ออกเป็นช่วงๆ ช่วงละ 2-4 สัปดาห์ หลังสิ้นสุดแต่ละสปรินท์อาจหยุดกิจกรรม 3-5 วันให้ หลักการของสครัมประกอบไปด้วย 3 หัวข้อหลักคือ

1) *ทีมงาน (Role)* มีประมาณ 5-9 คน แต่ละคนสามารถทดแทนกันได้เสมอ มีตัวแทนของลูกค้าทำหน้าที่จัดการเรื่องรับส่งมอบผลลัพธ์ (Product Backlog) และมีหัวหน้าทีมสครัมเป็นคนรับผิดชอบคุณภาพของผลงาน

2) *วิธีการทำงาน (Process)* ตัวแทนลูกค้าจะทำหน้าที่รวบรวม feature ที่ต้องทำและจัดลำดับตามความสำคัญเพื่อเข้าสปรินท์ การทำสปรินท์เป็นการทำแบบวนลูป แต่ละช่วงมีกำหนดไม่เกิน 30 วัน แต่ละวันจะมีการประชุมกันเพื่อปรึกษาแก้ไขปัญหาที่พบจากวันก่อนหน้านี้

3) *การประเมินและติดตามงาน (Demonstration and Evaluation)* หลักการคือพิจารณาจากจำนวนงานที่เหลือและเวลาที่ใช้ไป โดยปกติพิจารณาเป็นแต่ละวัน

โดยทั่วไปประเภทของความต้องการสามารถแบ่งได้ 3 ประเภท คือ 1) *ความต้องการทางฟังก์ชัน (Functional Requirements)* ได้แก่ ความต้องการที่ระบุว่าระบบทำอะไร มีหน้าที่อะไร มีบริการอะไร ระบบมีปฏิกิริยาตอบสนองอย่างไรต่อข้อมูลเข้าและส่งข้อมูลออกอย่างไร ระบบดำเนินการอย่างไรและไม่ทำอะไรในภาวะปกติ 2) *ความต้องการที่ไม่เป็นฟังก์ชัน (Non-Functional Requirements)* เป็นความต้องการที่มักไม่เกี่ยวข้องกับหน้าที่โดยตรงของระบบย่อยระบบใดแต่เป็นการระบุคุณสมบัติบางอย่างของระบบ คุณสมบัติที่สามารถวัดความต้องการเชิงคุณภาพ ได้แก่ ความเร็ว (Speed) ผลลัพธ์ (Throughput) ขนาด (Size) ขนาดของข้อมูล (Volume) ความง่ายต่อการใช้งาน (Easiness) เวลาที่ใช้ในการฝึก (Training time) ความน่าเชื่อถือ (Reliability) ความทนทาน (Durability) เคลื่อนย้ายได้ (Portability) และ 3) ข้อจำกัด (Constraint) เป็นความต้องการที่จำกัดพื้นที่ของการแก้ปัญหาไม่ให้นอกเหนือไปจากสิ่งที่เป็นความต้องการทางฟังก์ชันและความต้องการเชิงคุณภาพ ได้แก่ ข้อจำกัดในการให้บริการหรือข้อจำกัดในหน้าที่ของระบบ ข้อจำกัดด้านเวลา ข้อจำกัดในการพัฒนาและมาตรฐาน เป็นต้น

3. วิธีการดำเนินการวิจัย

ในงานศึกษาวิจัยนี้ได้ดำเนินการสร้างแนวปฏิบัติงานวิศวกรรมความต้องการที่ดีและเหมาะสมเพื่อนำไปสู่ความสำเร็จของโครงการซอฟต์แวร์ การศึกษามีการทดลองดำเนินโครงการซอฟต์แวร์โดยเริ่มจากการตรวจสอบความรู้ของบุคลากรเพื่อจัดสรรเป็นทีมพัฒนาในโครงการซอฟต์แวร์ มีการจัดสรรทรัพยากรและความต้องการที่จำเป็นในกระบวนการวิศวกรรม ขั้นตอนในการดำเนินการวิจัยรวม 2 ขั้นตอน ดังนี้ ขั้นตอนที่ 1 การศึกษากิจกรรมวิศวกรรมความต้องการในการพัฒนาโครงงานซอฟต์แวร์ และ ขั้นตอนที่ 2 การศึกษากิจกรรมวิศวกรรมความต้องการในการเปลี่ยนแปลงความต้องการระบบซอฟต์แวร์ที่มีอยู่เดิม

ด้วยข้อจำกัดทางด้านเวลาและความเป็นไปได้ในการดำเนินงานวิจัย การวิจัยนี้ได้ถูกออกแบบให้มีการจัดตั้งทีมพัฒนาในโครงการซอฟต์แวร์โดยคัดสรรบุคลากรที่มีประสบการณ์ในการพัฒนาซอฟต์แวร์ขึ้น 2 ทีม โดยกำหนดให้ทีมพัฒนาที่หนึ่งดำเนินโครงการตามกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัดและทีมพัฒนาที่สองดำเนินโครงการโดยมุ่งเน้นการพัฒนาเขียนโปรแกรมและไม่เคร่งครัดในแง่ของวิศวกรรมความต้องการ แต่ละทีมพัฒนาจะประกอบด้วยนักวิเคราะห์ระบบ ผู้จัดการโครงการ และนักพัฒนาซอฟต์แวร์ 2 คน โดยบุคลากรที่ร่วมในโครงการทุกคนมีประสบการณ์ในการพัฒนาโครงการซอฟต์แวร์แบบดั้งเดิมและแบบสครัม นักพัฒนาซอฟต์แวร์และนักวิเคราะห์ระบบทุกคนมีประสบการณ์ในการดำเนินกิจกรรมวิศวกรรมความต้องการ ได้แก่ การรวบรวมความต้องการ สร้างข้อกำหนด วิเคราะห์และ

ออกแบบระบบเชิงวัตถุ เป็นต้น ประมาณ 4-5 ปี การวิจัยครั้งนี้ดำเนินการทดลองให้มีการสร้างโครงการพัฒนาซอฟต์แวร์ 3 ชุด เพื่อสร้างซอฟต์แวร์ที่ตอบสนองความต้องการของผู้ใช้ที่หลากหลายในแง่ประโยชน์เชิงธุรกิจจึงได้มีรูปแบบออกมาเป็น 3 แบบ โดยกำหนดระยะเวลาของการดำเนินโครงการให้เสร็จสิ้นภายใน 6 เดือน โดยกำหนดให้ทีมพัฒนาที่หนึ่งดำเนินโครงการตามกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัดและทีมพัฒนาที่สองดำเนินโครงการโดยมุ่งเน้นการพัฒนาเขียนโค้ดและไม่เคร่งครัดในแง่ของวิศวกรรมความต้องการ โดยกำหนดภาษาสำหรับพัฒนาซอฟต์แวร์คือ ภาษาจาวาเช่นเดียวกัน สำหรับเครื่องมือที่ใช้สนับสนุนกระบวนการซอฟต์แวร์ คือ 1) Eclipse เป็น IDE สำหรับพัฒนาซอฟต์แวร์ 2) Microsoft Excel เพื่อแสดงรายงานผลข้อผิดพลาด การขอเปลี่ยนแปลง การขอแก้ไข ข้อกำหนดงานในการที่จะส่งแต่ละครั้ง 3) Google Documents เพื่อแสดงข้อมูลแบบออนไลน์ระหว่างทีมผู้พัฒนา 3 และ 4) Microsoft Project เพื่อแสดงแผนงาน การดำเนินงาน การประเมินผลงานของโครงการ

3.1 การศึกษากิจกรรมวิศวกรรมความต้องการในการพัฒนาโครงงานซอฟต์แวร์

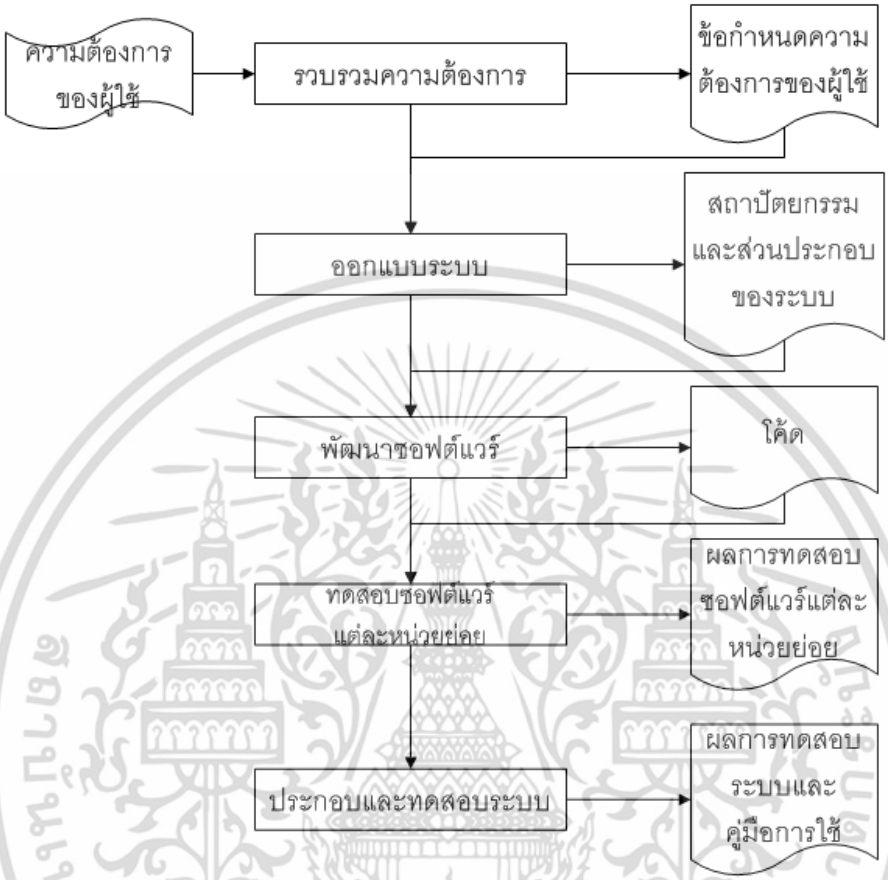
ในการดำเนินโครงการผู้พัฒนาในทีมจะได้รับบทบาทที่แตกต่างกัน ดังแสดงในภาพที่ 1 ดังนี้ ก) เริ่มต้นจากผู้พัฒนารวบรวมความต้องการทั้งหมดจากลูกค้าและสร้างเป็นข้อกำหนดความต้องการของผู้ใช้ ข้อกำหนดความต้องการจะเก็บรายละเอียดและใช้เป็นเครื่องมือในการอธิบายคุณสมบัติต่างๆของซอฟต์แวร์ ข) ผู้พัฒนาทำการออกแบบสถาปัตยกรรมระบบและ ส่วนประกอบของระบบ โดยแสดงออกมาในรูปแบบจำลองข้อมูล แผนภาพคลาสไดอะแกรม แผนภาพลำดับ และแผนภาพกิจกรรม ซึ่งการออกแบบระบบใช้ข้อกำหนดความต้องการเป็นเครื่องมือสำคัญ ค) การพัฒนาโค้ดโดยมีเอกสารการออกแบบเป็นเครื่องมือสำคัญ ง) การทดสอบซอฟต์แวร์โดยทำตามเอกสารทดสอบและใช้การทดสอบหน่วยแต่ละหน่วยเพื่อหาความผิดพลาดในแต่ละจุด จ) นำทุกหน่วยองค์ประกอบมาประกอบเข้าด้วยกันและเริ่มการทดสอบแบบรวม และ ฉ) เมื่อซอฟต์แวร์ผ่านการทดสอบจะถือเป็นผลิตภัณฑ์ซอฟต์แวร์ที่เสร็จสมบูรณ์และสามารถส่งต่อให้กับลูกค้า

การศึกษาวิจัยนี้ได้นำเสนอการประยุกต์ใช้เอกสารเพื่อสนับสนุนกิจกรรมต่างๆภายใต้กระบวนการซอฟต์แวร์ข้างต้น (ประเภทของเอกสารที่ใช้ดูรายละเอียดในส่วนสรุปผลการวิจัย) และในการศึกษานี้ได้เสนอให้มีการรวบรวมบริบทของความต้องการตามหลักการของวิศวกรรมความต้องการ เริ่มจากการที่นักพัฒนาซอฟต์แวร์จะได้รับรายละเอียดความต้องการและสร้างเป็นข้อกำหนดความต้องการพร้อมทั้งเอกสารประเภทอื่นๆ ซึ่งในทางปฏิบัติความต้องการอยู่ในรูปแบบที่หลากหลายในหลายบริบท เป็นสิ่งที่สำคัญและจำเป็นต้องรวบรวมระหว่างการดำเนินกิจกรรมวิศวกรรมซอฟต์แวร์ ซึ่งบริบทแต่ละประเภทใช้แสดงหรืออธิบายความต้องการในแง่มุมที่แตกต่างกันดังแสดงในตารางที่ 1 และได้เสนอการประยุกต์ใช้เทคนิคการรวบรวมข้อมูลความต้องการโดยได้วิเคราะห์จุดแข็งและจุดอ่อนของแต่ละเทคนิคที่นำเสนอ ดังแสดงตามตารางที่ 2 โดย

FR หมายถึง ความต้องการทางฟังก์ชัน

QR หมายถึง ความต้องการเชิงคุณภาพ

CS หมายถึง ข้อจำกัด



ภาพที่ 1 แผนภาพแสดงกิจกรรมการพัฒนาโครงงานซอฟต์แวร์

ตารางที่ 1. ประเภทของบริบทความต้องการ

บริบทของความต้องการ	**ประเภทความต้องการ**	**ลักษณะของบริบทเพื่อใช้แสดงหรืออธิบาย**
การทดสอบการยอมรับระบบ	FR QR CS	ลักษณะของระบบที่ผู้มีส่วนได้ส่วนเสียสนใจ
นิยามเงื่อนไขทางธุรกิจ	FR	เงื่อนไขทางธุรกิจเป็นหลักการหรือนโยบายที่ระบบซอฟต์แวร์จำเป็นต้องสอดคล้อง
ข้อจำกัด	CS	ข้อจำกัดในเรื่องต่างๆเพื่อนักพัฒนาระบบสามารถหาแนวทางเพื่อตอบสนองความต้องการได้ภายใต้ข้อจำกัดดังกล่าว

ตารางที่ 1. (ต่อ) ประเภทของบริบทความต้องการ

บริบทของความต้องการ	ประเภทความต้องการ	ลักษณะของบริบทเพื่อใช้แสดงหรืออธิบาย
แผนภาพการไหลของข้อมูล	FR	การเคลื่อนไหวของข้อมูลภายในระบบระหว่างส่วนต่างๆในระบบ ได้แก่ กระบวนการ แหล่งที่เก็บข้อมูล บุคคล
ระบบต้นแบบ	FR QR CS	ส่วนติดต่อประสานงานกับผู้ใช้หลักๆ เพื่อแสดงให้เห็นแนวรูปแบบของระบบที่จะพัฒนา
ยูสเคส	FR	การปฏิสัมพันธ์ระหว่างผู้ที่ติดต่อกับระบบและระบบ โดยมีรายละเอียดลำดับของขั้นตอนที่ดำเนินงานในแต่ละฟังก์ชัน
คุณลักษณะ	FR QR	ลักษณะความสามารถของระบบที่มาจากมุมมองของผู้ใช้งานระบบ
ข้อกำหนดทางเทคนิค	FR CS	แง่มุมของระบบทางด้านเทคนิค
สถานการณ์ของการใช้งานระบบ	FR	สถานการณ์หรือเงื่อนไขเหตุการณ์ที่มีผลต่อการดำเนินงานของระบบ
แผนภาพยูสเคส	FR	ภาพรวมของยูสเคส ผู้ที่ติดต่อกับระบบ ความสัมพันธ์ระหว่างผู้ที่ติดต่อกับระบบและแต่ละยูสเคส และแสดงการจำลองบริบทของระบบที่ปฏิสัมพันธ์กับระบบภายนอก
เรื่องราวของผู้ใช้	FR QR	ข้อมูลการสนทนาระหว่างผู้มีส่วนได้เสียในโครงการเก็บข้อมูลความต้องการในระดับสูงรวมทั้งความต้องการเชิงพฤติกรรม เงื่อนไขและกฎเกณฑ์ทางธุรกิจ ข้อจำกัด และข้อกำหนดทางเทคนิค

ตารางที่ 2. เทคนิคที่ใช้เพื่อรวบรวมข้อมูล

เทคนิค	คำอธิบาย	จุดแข็ง	จุดอ่อน
การให้ผู้มีส่วนได้ส่วนเสียมีส่วนร่วมในการสร้างข้อกำหนดและแบบจำลองความต้องการ	จัดให้ผู้มีส่วนได้ส่วนเสียที่มีบทบาทต่าง ๆ ในระบบมีการแสดงความคิดเห็นและให้ข้อมูลความต้องการระบบร่วมกันเพื่อสร้างเป็นข้อกำหนดและแบบจำลองความต้องการ	- สนับสนุนให้มีการทำงานร่วมกันสูง - ผู้เชี่ยวชาญโดเมนสามารถมีส่วนร่วมในการกำหนดความต้องการของระบบ - ดำเนินการภายใต้ระยะเวลาที่กำหนดและเหมาะสม - การตัดสินใจทำได้ในเวลาที่เหมาะสม	- ผู้มีส่วนได้ส่วนเสียจำนวนมากที่จำเป็นต้องเรียนรู้ทักษะการสร้างข้อกำหนดและแบบจำลองความต้องการ - ผู้มีส่วนได้ส่วนเสียบางคนไม่สามารถมีส่วนร่วมได้เต็มเวลาหรือมีเวลาที่จำกัดในการมีส่วนร่วม
สัมภาษณ์โดยตรง	เป็นการพบปะและหารือเกี่ยวกับความต้องการของผู้มีส่วนได้ส่วนเสียโดยตรง ผู้มีส่วนได้ส่วนเสียในบทบาทต่างๆที่มีต่อระบบจะถูกสัมภาษณ์เพื่อให้ข้อมูลความต้องการ	- สนับสนุนให้มีการทำงานร่วมกัน - ในการสัมภาษณ์บางครั้ง วิศวกรความต้องการสามารถเก็บรายละเอียดข้อมูลจำนวนมากได้อย่างรวดเร็วจากผู้มีส่วนได้ส่วนเสียเพียงคนเดียว	- การสัมภาษณ์จะต้องมีตารางเวลานัดหมายล่วงหน้า - ต้องอาศัยทักษะการสัมภาษณ์ซึ่งเป็นเรื่องยากที่จะเรียนรู้
ศึกษาจากเอกสารข้อมูลที่ได้บันทึกไว้	การศึกษาเอกสารข้อมูลเพื่อระบุความต้องการของระบบที่ควรครอบคลุม	- มีโอกาสที่จะได้เรียนรู้พื้นฐานของโดเมนระบบก่อนที่ติดต่อกับผู้มีส่วนได้ส่วนเสียโดยตรง	- ขาดการทำงานร่วมกัน - การปฏิบัติมักจะแตกต่างจากสิ่งที่บันทึกไว้ - จำกัดเฉพาะสำหรับนักพัฒนาที่สามารถอ่านและเข้าใจข้อมูลที่ได้บันทึกไว้

3.2 การศึกษากิจกรรมวิศวกรรมความต้องการในการเปลี่ยนแปลงความต้องการระบบซอฟต์แวร์ที่มีอยู่เดิม

การศึกษาครั้งนี้ดำเนินการต่อจากขั้นตอนที่ 1 จากการศึกษาได้ทดลองดำเนินโครงการซอฟต์แวร์ที่มีการพัฒนาซอฟต์แวร์ 3 ซอฟต์แวร์ และในขั้นตอนนี้ได้ทดลองเพื่อจัดการกับสถานการณ์ที่มีการเปลี่ยนแปลงความต้องการ ซึ่งส่งผลให้มีการเปลี่ยนแปลงผลิตภัณฑ์ซอฟต์แวร์ที่ได้พัฒนาไปแล้ว

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

4. สรุปผลการวิจัย

จากการทดลองศึกษานี้ขั้นตอนที่ 1 พบว่ากิจกรรมต่างๆภายใต้โครงการซอฟต์แวร์ที่แต่ละทีมต้องมีการพัฒนาซอฟต์แวร์ 3 ชุด โดยซอฟต์แวร์ชุดที่หนึ่ง (M1) เป็นการพัฒนาเว็บแอพพลิเคชั่นที่สนับสนุนงานธุรกิจในส่วนของการจัดการข้อมูลสินค้าคงคลังของบริษัทแห่งหนึ่ง ซอฟต์แวร์ชุดที่สอง (M2) เป็นการพัฒนาเว็บแอพพลิเคชั่นที่สนับสนุนงานธุรกิจในส่วนของการจัดการการจองและเช่าร้านในศูนย์การค้าแห่งหนึ่ง และซอฟต์แวร์ชุดที่สาม (M3) เป็นการพัฒนาเว็บแอพพลิเคชั่นที่สนับสนุนงานธุรกิจในส่วนของการขายสินค้าออนไลน์ของบริษัทแห่งหนึ่ง มีการสร้างเอกสารซอฟต์แวร์ขึ้นเป็นจำนวนมากและหลากหลาย เอกสารซอฟต์แวร์ที่ได้ถูกประยุกต์ใช้เพื่อวิศวกรรมความต้องการและวิศวกรรมซอฟต์แวร์ส่วนอื่นๆ ซึ่งเอกสารเหล่านี้ได้ถูกสร้างขึ้นเพื่อเก็บรวบรวมข้อมูลความต้องการในบริบทและแง่มุมที่แตกต่างกันจากแหล่งที่มาและผู้มีส่วนได้ส่วนเสียต่างๆ ข้อมูลเหล่านั้น ได้แก่ การทดสอบการยอมรับระบบ นิยามเงื่อนไขทางธุรกิจ ข้อจำกัด แผนภาพการไหลของข้อมูล ระบบต้นแบบ ยูสเคส คุณลักษณะ ข้อกำหนดทางเทคนิค สถานการณ์ของการใช้งานระบบ แผนภาพยูสเคส และ เรื่องราวของผู้ใช้ รายละเอียดของจำนวนของเอกสารที่ถูกสร้างขึ้นจากขั้นตอนที่ 1 แสดงในตารางที่ 3 โดยในคอลัมน์แรกแสดงประเภทของเอกสารที่ถูกสร้างขึ้นและคอลัมน์ต่อมาแสดงจำนวนของเอกสารที่ถูกสร้าง จากตารางประเภทเอกสารแต่ละประเภทมีดังนี้

ก: เอกสารวิชัน (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)
ซ: แผนภาพลำดับ (Sequence Diagram)

ตารางที่ 3. จำนวนของเอกสารที่ถูกสร้างขึ้น

ประเภท	ทีม: 1			ทีม: 2		
	M1	M2	M3	M1	M2	M3
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

เวลาที่ใช้ในการพัฒนาเอกสารสำหรับทีมที่หนึ่งและทีมที่สองเป็น 320 และ 80 ชั่วโมงตามลำดับ โดยทีมที่หนึ่งได้สร้างประเภทและจำนวนเอกสารตามประเภทของบริบทความต้องการตามตารางที่ 1 และประยุกต์ใช้เทคนิคตามความเหมาะสมดังแสดงในตารางที่ 2 และเวลาที่ใช้สำหรับการพัฒนาโค้ดของแต่ละทีมแสดงในตารางที่ 4 นอกจากนี้ได้ประยุกต์ใช้ตัววัดขนาดของโปรแกรม 2 แบบ คือ จำนวนบรรทัด (Line of Code) และหน่วยฟังก์ชัน (Function Point) รายละเอียดจำนวนบรรทัดและหน่วยฟังก์ชันดังแสดงในตารางที่ 4 พบว่าหน่วยฟังก์ชันที่ได้จากการพัฒนาจากทั้งสองทีมมีขนาดเท่ากัน แต่จำนวนบรรทัดโค้ดรวมของทีมที่หนึ่งและสองเป็น 17,249 และ 21,095 บรรทัด ตามลำดับ

ตารางที่ 4. จำนวนบรรทัดและหน่วยฟังก์ชัน

ทีม	ซอฟต์แวร์	จำนวนบรรรทัดโค้ด	หน่วยฟังก์ชัน	เวลาที่ใช้ในการพัฒนา (ชม.)
1	M1	7,689	10	400
	M2	2,280	5	150
	M3	7,280	4	320
2	M1	6,830	10	280
	M2	5,420	5	200
	M3	8,845	4	260

ทั้งสองทีมได้ดำเนินการทดสอบโปรแกรมแบบ Functional Test เพื่อตรวจสอบความถูกต้องของ Feature ต่างๆ ว่าทำงานถูกต้องหรือไม่ โดยกำหนดการทดสอบตามจำนวนยูสเคสของแต่ละซอฟต์แวร์ ผู้ทำการทดสอบทำการทดสอบความถูกต้องของแต่ละฟังก์ชั่นโดยระบุคะแนนเป็น 0-5 (0 – 1.00 หมายถึง โปรแกรมไม่สามารถนำไปใช้งานได้ 1.01 – 2.00 หมายถึง โปรแกรมต้องปรับปรุงแก้ไข 2.01 – 3.00 หมายถึง โปรแกรมมีประสิทธิภาพพอใช้ 3.01 – 4.00 หมายถึง โปรแกรมมีประสิทธิภาพดี 4.01 – 5.00 หมายถึง โปรแกรมมีประสิทธิภาพดีมาก) ทั้งนี้ผลที่ได้จากการทดสอบคำนวณจากค่าเฉลี่ยของผู้ทำการทดสอบ 4 คน โดยแต่ละคนทำการทดสอบโปรแกรมทุกโปรแกรมของทั้งสองทีม พบว่าค่าเฉลี่ยของความถูกต้องของโปรแกรมที่ได้จากสองทีมเป็น 4.5 และ 4.8 ตามลำดับ โดยคำนวณจากผลรวมของคะแนนความถูกต้องของการทดสอบแต่ละฟังก์ชั่นในแต่ละซอฟต์แวร์หารด้วยจำนวนหน่วยฟังก์ชั่นในแต่ละซอฟต์แวร์ ผลค่าเฉลี่ยของความถูกต้องของซอฟต์แวร์ที่แต่ละทีมได้พัฒนาได้ถูกนำมารวมและหารด้วยจำนวนซอฟต์แวร์ที่พัฒนาเพื่อคิดเป็นค่าเฉลี่ยของความถูกต้องของโปรแกรมที่แต่ละทีมได้พัฒนาขึ้นจากทั้งสามซอฟต์แวร์ ดังนี้

$$E_j = (x_j + y_j + z_j) / 3$$

A_j = ค่าเฉลี่ยความถูกต้องของการทดสอบโปรแกรมของของทีมที่ j
x_j = ค่าเฉลี่ยความถูกต้องของการทดสอบโปรแกรมซอฟต์แวร์ M1 ของทีมที่ j
y_j = ค่าเฉลี่ยความถูกต้องของการทดสอบโปรแกรมซอฟต์แวร์ M2 ของทีมที่ j
z_j = ค่าเฉลี่ยความถูกต้องของการทดสอบโปรแกรมซอฟต์แวร์ M3 ของทีมที่ j

ในการศึกษาทดลองขั้นตอนที่ 2 กรณีที่มีการเปลี่ยนแปลงความต้องการ ตัวชี้วัดคุณภาพของซอฟต์แวร์ที่สำคัญเกี่ยวข้องกับการจัดการการเปลี่ยนแปลง คือ ความพยายามที่ใช้ในการแก้ไขโค้ด โดยในการศึกษาใช้ร้อยละของการเปลี่ยนแปลงโมดูลหรือหน่วยย่อยของโค้ด และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงเป็นตัววัด ซึ่งพบว่าร้อยละของการเปลี่ยนแปลงโค้ดในซอฟต์แวร์ตัวที่หนึ่ง (M1) ตัวที่สอง (M2) และตัวที่สาม (M3) คิดเป็น 25 0 และ 15 ตามลำดับ และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่หนึ่ง (M1) จากทีมที่ 1 ใช้เวลาเป็น 15.5 วัน เวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่สอง (M2) ใช้เวลาเป็น 0 วันเนื่องจากไม่มีการเปลี่ยนแปลงใดๆ และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่สาม (M3) ใช้เวลาเป็น 8 วัน และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่หนึ่ง (M1) จากทีมที่ 2 ใช้เวลาเป็น 32 วัน เวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่สอง (M2) ใช้เวลาเป็น 0 วันเนื่องจากไม่มีการเปลี่ยนแปลงใดๆ และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ตัวที่สาม (M3) ใช้เวลาเป็น 12 วัน ดังแสดงในตารางที่ 5

จากการสำรวจเบื้องต้นโดยการใช้แบบสอบถามความพึงพอใจของทีมพัฒนาพบว่า นักพัฒนามีความพึงพอใจในกระบวนการที่เน้นการเขียนโค้ดมากกว่าการสร้างและบันทึกเอกสาร และนอกจากนี้นักพัฒนาบางส่วนยังไม่เห็นด้วยกับกิจกรรมติดตามที่ต้องปรับปรุงแก้ไขเอกสารที่เกี่ยวข้องทุกครั้งเมื่อมีการเปลี่ยนแปลงความต้องการเกิดขึ้น จากการสำรวจพบว่าร้อยละ 33 ของนักพัฒนามีแนวโน้มที่จะต่อต้านการปฏิบัติตามแนวทางวิศวกรรมความต้องการในขณะที่ร้อยละ 70 ของนักพัฒนามีความคิดเชิงบวกต่อแนวทางปฏิบัติที่ดีของวิศวกรรมความต้องการ โดยเฉพาะอย่างยิ่งร้อยละ 82 ของนักพัฒนามีความพึงพอใจ

กับขั้นตอนการบำรุงรักษาและการจัดการการเปลี่ยนแปลงที่ได้ดำเนินกิจกรรมภายใต้วิศวกรรมความต้องการที่ดีและเหมาะสม

ตารางที่ 5. ร้อยละของการเปลี่ยนแปลงโมดูลหรือหน่วยย่อยของโค้ด และเวลาเฉลี่ยในการดำเนินการเปลี่ยนแปลง

ทีม	ซอฟต์แวร์	ร้อยละของการเปลี่ยนแปลงโมดูลหรือหน่วยย่อยของโค้ด	เวลาในการเปลี่ยนแปลง (ชม.)
1	M1	25	15.5
	M2	0	0
	M3	15	8
2	M1	25	32
	M2	0	0
	M3	15	12

จากตารางที่ 5 เวลาที่แต่ละทีมใช้สามารถเปรียบเทียบได้โดยประยุกต์ใช้วิธี Euclidean Distance เพื่อพิจารณาความแตกต่างด้านเวลาที่ใช้ของทั้งสองทีม ดังนี้

$$E_i = \sqrt{\delta x_{ij}^{\ 2} + \delta y_{ij}^{\ 2} + \delta z_{ij}^{\ 2}}$$

E_j = หน่วยเวลาของแต่ละทีม

x_{ij} = เวลาที่ใช้ในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ M1 ของทีมที่ j

y_{ij} = เวลาที่ใช้ในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ M2 ของทีมที่ j

z_{ij} = เวลาที่ใช้ในการดำเนินการเปลี่ยนแปลงซอฟต์แวร์ M3 ของทีมที่ j

พบว่า E_1 มีค่าเท่ากับ 17.44 และ E_2 มีค่าเท่ากับ 34.18 ชั่วโมง ตามลำดับ นั่นคือทีมที่หนึ่งใช้หน่วยเวลาของการดำเนินการเปลี่ยนแปลงน้อยกว่าทีมที่สอง ในขณะที่ร้อยละของการหน่วยย่อยของโค้ดที่เปลี่ยนแปลงมีขนาดเท่ากัน

5. อภิปรายผลการวิจัยและข้อเสนอแนะ

งานวิจัยนี้ได้นำเสนอการทดลองดำเนินโครงการซอฟต์แวร์ สำหรับขั้นตอนที่ 1 ให้มีการสร้างโครงการพัฒนาซอฟต์แวร์ 3 ชุด โดยกำหนดให้ทีมพัฒนาที่หนึ่งดำเนินโครงการตามกิจกรรมภายใต้วิศวกรรมความต้องการอย่างเคร่งครัดและทีมพัฒนาที่สองดำเนินโครงการโดยมุ่งเน้นการพัฒนาเขียนโค้ดและไม่เคร่งครัดในแง่ของวิศวกรรมความต้องการ จากการทดลองพบว่าจำนวนเอกสารที่ถูกสร้างขึ้นจากทีมพัฒนาที่หนึ่งมีเอกสาร 48 เอกสารและจากทีมพัฒนาที่สองมี 19 เอกสาร เนื่องจากทีมที่หนึ่งดำเนินกระบวนการซอฟต์แวร์โดยมีการปฏิบัติตามหลักการวิศวกรรมความต้องการอย่างเคร่งครัด มีการสร้างเอกสารซอฟต์แวร์และข้อกำหนดความต้องการอย่างละเอียดและครบถ้วน ในขณะที่ทีมที่สองดำเนินกระบวนการซอฟต์แวร์โดยมุ่งเน้นการเขียนโค้ดมากกว่าทำให้ละเลยการสร้างเอกสารซอฟต์แวร์บางอย่างไป

กล่าวคือ จำนวนเอกสารที่ทีมพัฒนาที่หนึ่งสร้างมีมากกว่าเอกสารที่ทีมพัฒนาที่สองสร้างโดยคิดเป็น 2.5 เท่าของทีมที่สอง และขนาดของซอฟต์แวร์ที่ผลิตขึ้นจากทั้งสองทีมโดยวัดจากหน่วยฟังก์ชันมีเท่ากัน แต่หากพิจารณาจำนวนบรรทัดของโค้ดโดยรวมที่ทีมที่หนึ่งพัฒนาขึ้นมี 17,249 บรรทัดขณะที่ทีมที่สองมี 21,095 บรรทัด กล่าวคือ จำนวนบรรทัดของโปรแกรมที่ทีมพัฒนาที่หนึ่งสร้างน้อยกว่าที่ทีมพัฒนาที่สองสร้างคิดเป็น 0.82 เท่าของทีมที่สอง ซึ่งไม่ถือว่าแตกต่างกันมาก นอกจากนี้จากผลการทดสอบ functional test ค่าเฉลี่ยของความถูกต้องของโปรแกรมที่ได้จากสองทีมเป็น 4.5 และ 4.8 ตามลำดับ จากคะแนนเต็ม 5 ซึ่งแตกต่างกันเพียงเล็กน้อย ทั้งนี้ทีมผู้พัฒนาทั้งสองทีมมีประสบการณ์และความถนัดในโครงการซอฟต์แวร์ไม่แตกต่างกัน แต่อย่างไรก็ตามจากการสำรวจความพึงพอใจภายหลังพบว่านักพัฒนาซอฟต์แวร์จากทีมที่หนึ่งได้ประยุกต์ใช้เอกสารซอฟต์แวร์ที่ได้สร้างไว้อย่างครอบคลุมและสมบูรณ์เป็นเครื่องมือในการดำเนินกิจกรรมต่างๆภายใต้กระบวนการซอฟต์แวร์จึงทำให้สามารถเข้าใจและระบุความต้องการที่ถูกกำหนดไว้ได้อย่างรวดเร็วและถูกต้อง และสามารถนำองค์ประกอบซอฟต์แวร์บางส่วนกลับมาใช้ใหม่ทำให้ลดปริมาณบรรทัดของโปรแกรมที่ต้องพัฒนาขึ้นใหม่

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

การดำเนินกิจกรรมภายใต้กระบวนการซอฟต์แวร์ตามหลักของวิศวกรรมความต้องการทำให้เกิดการสร้างเอกสารซอฟต์แวร์มากขึ้นและสำหรับผู้ที่ไม่คุ้นเคยอาจไม่เคยชินกับการดำเนินตาม ทั้งนี้พิจารณาจากการสำรวจความพึงพอใจของนักพัฒนาซอฟต์แวร์มีความพึงพอใจในกระบวนการที่เน้นการเขียนโค้ดมากกว่าการสร้างและบันทึกเอกสารและบางส่วนยังไม่เห็นด้วยกับกิจกรรมติดตามที่ต้องปรับปรุงแก้ไขเอกสารที่เกี่ยวข้องทุกครั้งเมื่อมีการเปลี่ยนแปลงความต้องการเกิดขึ้น ร้อยละ 33 ของนักพัฒนามีแนวโน้มที่จะต่อต้านการปฏิบัติตามแนวทางวิศวกรรมความต้องการ

แต่อย่างไรก็ตามเมื่อมีการเปลี่ยนแปลงแก้ไขความต้องการภายหลังเอกสารซอฟต์แวร์ต่างๆรวมถึงข้อกำหนดความต้องการได้กลายเป็นเครื่องมือที่สำคัญและเป็นประโยชน์ทำให้สามารถดำเนินการแก้ไขได้อย่างรวดเร็วและมีประสิทธิภาพ จากการสำรวจความพึงพอใจพบว่าร้อยละ 70 ของนักพัฒนาซอฟต์แวร์มีความคิดเชิงบวกที่จะดำเนินการตามแนวทางปฏิบัติที่ดีของวิศวกรรมความต้องการ และพบว่าร้อยละ 82 ของนักพัฒนามีความพึงพอใจกับขั้นตอนการบำรุงรักษาและการจัดการการเปลี่ยนแปลงที่ได้ดำเนินกิจกรรมภายใต้วิศวกรรมความต้องการที่ดีและเหมาะสม

กิจกรรมต่างๆภายใต้วิศวกรรมความต้องการเป็นงานสำคัญของการพัฒนาระบบซอฟต์แวร์จำเป็นต้องมีความสามารถในการจัดการกับปัจจัยและตัวแปรต่างๆที่อาจเกิดขึ้นในการพัฒนา การได้มาซึ่งความต้องการ การรวบรวมความต้องการ การสร้างข้อกำหนดความต้องการ และการทวนสอบความต้องการ กิจกรรมเหล่านี้เกิดขึ้นภายใต้โครงสร้างที่ซับซ้อนและมีผู้มีส่วนได้ส่วนเสียแตกต่างกัน ตารางที่แสดงข้อสรุปด้านข้อดีและข้อเสียของแนวทางการดำเนินโครงการซอฟต์แวร์ภายใต้กิจกรรมวิศวกรรมความต้องการอย่างเคร่งครัดและแบบมุ่งเน้นการพัฒนาโปรแกรม

ตารางที่ 6. ข้อดีและข้อเสียของการดำเนินโครงการซอฟต์แวร์ภายใต้กิจกรรมวิศวกรรมความต้องการอย่างเคร่งครัดและแบบมุ่งเน้นการพัฒนาโปรแกรม

แนวทางการดำเนินโครงการซอฟต์แวร์	ข้อดี	ข้อเสีย
ภายใต้กิจกรรมวิศวกรรมความต้องการอย่างเคร่งครัด	1. การสื่อสารในทีมพัฒนาสามารถทำได้อย่างคล่องตัวและชัดเจนผ่านการพิจารณาเอกสารที่เป็นผลลัพธ์จากการดำเนินกิจกรรมต่างๆ 2. การตรวจสอบความถูกต้องและสอดคล้องกันของความต้องการและระบบซอฟต์แวร์ที่พัฒนาขึ้น สามารถทำได้อย่างมีประสิทธิภาพ 3. เมื่อมีการเปลี่ยนแปลงแก้ไขความต้องการภายหลัง เอกสารซอฟต์แวร์ต่างๆรวมถึงข้อกำหนดความต้องการได้กลายเป็นเครื่องมือที่สำคัญในการดำเนินการ ทำให้สามารถดำเนินการแก้ไขได้อย่างรวดเร็วและมีประสิทธิภาพ	1. ใช้เวลามากในการดำเนินกิจกรรมตามขั้นตอน 2. มีความยุ่งยากในการสร้างเอกสารจากการดำเนินกิจกรรม 3. ทีมผู้พัฒนาต้องมีประสบการณ์และความรู้ในการสร้างเอกสารเพื่อเป็นผลลัพธ์ของการดำเนินกิจกรรม เช่น การออกแบบระบบโดยใช้เทคนิคยูเอ็มแอล
มุ่งเน้นการพัฒนาโปรแกรม	1. ลดขั้นตอนและเวลาในการพัฒนาระบบซอฟต์แวร์ 2. ลดความผิดพลาดจากการสร้างเอกสารและการอ้างอิงเอกสารที่ไม่ถูกต้อง	1. ขาดเอกสารอ้างอิงทำให้เกิดความคลาดเคลื่อนในการสื่อสารได้ 2. การตรวจสอบความถูกต้องและสอดคล้องกันของความต้องการทำได้ยุ่งยากและไม่ชัดเจน 3. เมื่อมีการเปลี่ยนแปลงแก้ไขความต้องการภายหลัง มีความยุ่งยากและมีประสิทธิภาพน้อยเนื่องจากต้องใช้เวลาเพิ่มในการทำความเข้าใจส่วนที่ได้พัฒนาไปแล้ว

ความสำเร็จของโครงการซอฟต์แวร์เกิดขึ้นได้จากหลายปัจจัยในแง่ของวิศวกรรมความต้องการมีดังนี้ 1) ประสบการณ์ของผู้ดำเนินโครงการอย่างวิศวกรความต้องการเป็นปัจจัยหนึ่งที่มีความสำคัญ กิจกรรมภายใต้วิศวกรรมและการจัดการความต้องการมักเกิดขึ้นในบริบทหรือขั้นตอนต่างๆของการพัฒนาซอฟต์แวร์ ผู้ดำเนินโครงการที่มีประสบการณ์สูงสามารถดำเนินโครงการและแก้ไขปัญหาที่เกิดขึ้นในกิจกรรมของการพัฒนาได้ดีกว่าผู้ดำเนินโครงการที่ขาดประสบการณ์ และ 2) คุณภาพของข้อกำหนดความต้องการที่ดีมีผล

เชิงบวกต่อการดำเนินโครงการ หมายถึง หากมีการรวบรวมความต้องการได้ครบถ้วนและถูกต้อง และมีการสร้างข้อกำหนดความต้องการได้ถูกต้องเหมาะสมและครอบคลุม จะสามารถทำให้ผู้ดำเนินโครงการอย่างนักพัฒนาซอฟต์แวร์สามารถนำข้อมูลไปใช้ต่อยอดในกิจกรรมการพัฒนาลำดับถัดๆไปโดยเฉพาะอย่างยิ่งหากการพัฒนาซอฟต์แวร์นั้นประยุกต์ใช้แบบจำลองน้ำตกในกิจกรรมวิศวกรรมซอฟต์แวร์ยิ่งต้องอาศัยข้อกำหนดความต้องการที่สมบูรณ์จึงจะพัฒนางานต่อไปได้ นอกจากนี้ข้อกำหนดความต้องการที่มีความถูกต้อง ครบถ้วน และสอดคล้องกันสามารถสนับสนุนกิจกรรมวิศวกรรมความต้องการอื่นๆได้ดีกว่าข้อกำหนดความต้องการที่ไม่ถูกต้อง ไม่ครบถ้วน หรือไม่สอดคล้องกัน

อย่างไรก็ตาม ในสถานการณ์บางอย่างหน่วยงานอาจต้องการเน้นความรวดเร็วโดยอาจละเลยหรือให้ความสำคัญในเอกสารน้อยลง จะจัดทำเฉพาะเอกสารบางประเภทที่สำคัญๆต่อการบำรุงรักษาในภายหลัง จึงมีเงื่อนไขในการใช้งานต่างกัน

ทั้งนี้การบำรุงรักษาข้อกำหนดความต้องการและเอกสารซอฟต์แวร์อื่นๆให้มีคุณภาพดี กล่าวคือ ให้มีความถูกต้อง สมบูรณ์ และสอดคล้องกัน เป็นสิ่งที่ต้องใช้ความพยายามซึ่งมีต้นทุนเป็นคนและเวลา ความพยายามที่ใช้เพื่อแก้ไขระบบซอฟต์แวร์ให้สอดคล้องกับความต้องการที่เปลี่ยนแปลงไปนั้นแปรผันตามความสามารถในการระบุชิ้นส่วนหรือองค์ประกอบซอฟต์แวร์ที่มีผลกระทบจากความต้องการที่เปลี่ยนแปลงไปนั้นๆ เช่น การระบุองค์ประกอบซอฟต์แวร์ที่ต้องเปลี่ยนแปลงในเอกสารการออกแบบและโค้ดจากการที่ข้อกำหนดความต้องการเปลี่ยน ทั้งนี้ต้องให้ครอบคลุมความต้องการทุกประเภทและทุกแง่มุม ตัวอย่างเช่น ความต้องการเชิงพฤติกรรมจะครอบคลุมวิธีการใช้งานและการที่ผู้ใช้จะมีปฏิสัมพันธ์กับระบบ การที่ผู้ใช้จะเกี่ยวข้องส่วนติดต่อผู้ใช้ รูปแบบและวิธีการที่ผู้ใช้จะใช้ระบบ การที่ระบบจะดำเนินงานตอบสนองเงื่อนไขหรือกฎเกณฑ์ทางธุรกิจ ส่วนความต้องการเชิงคุณภาพจะหมายถึงความต้องการทุกด้านที่ไม่ใช่ฟังก์ชัน ได้แก่ การรักษาความปลอดภัยของระบบ การทำให้ระบบมีประสิทธิภาพและมีความน่าเชื่อถือ เช่น ความเร็วที่คาดหวังในการเข้าถึงข้อมูล ซึ่งวัดได้จากเวลาตอบสนองของส่วนติดต่อผู้ใช้ การควบคุมการเข้าถึง สิ่งที่สำคัญคือการระบุและเข้าใจความต้องการให้ถูกต้อง ครอบคลุมและสมบูรณ์ เพื่อสามารถจัดการได้อย่างถูกต้องและเหมาะสมมิเช่นนั้นอาจกลายเป็นปัญหาภายหลังได้

เอกสารอ้างอิง (References)

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

[2] Pohl, K. and Rupp, C., 2011. Requirements Engineering Fundamentals. CA: Rocky Nook Inc.

[3] Sikora, E., Tenbergen, B. and Pohl, K., 2012. Industry needs and research directions in requirements engineering for embedded systems. *Requirements Engineering*, 17, 57-58.

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

[5] V-Modell., 2004. *The V-Modell.* [online] Available at: <http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.1-eng/Dokumentation/pdf/V-Modell-XT-eng-Teil1.pdf > [Accessed 23 August 2013].

[6] Beck, K., 1999. Extreme Programming Explained-Embrace Change. MA: Addison-Wesley.

[7] Beekhuizen, K. S., Davis, M. D., Kolber, M. J. and Cheng, M. S., 2009. Test-retest reliability and minimal detectable change of the hexagon agility test. J Strength Cond Res, 23(7), 2167-2171

[8] Sutherland, J., Jacobson, C. and Johnson, K., 2007. Scrum and CMMI Level 5: A Magic Potion for Code Warriors!. Agile 2007, 3(1), 1-16.

[9] Schwaber, K. and Beedle, M., 2001. Agile Software Development with Scrum. Prentice Hall.