บทที่ 1

บทนำ

การวิจัยครั้งนี้มุ่งเน้นเพื่อศึกษาการเรียงลำดับความสำคัญของความต้องการเพื่อแก้ปัญหาการเรียงลำดับความสำคัญของความต้องการขนาดใหญ่ ที่ใช้ค่าความพยายามในการเรียงลำดับความสำคัญที่น้อยลงและมีความถูกต้องของการเรียงลำดับมากยิ่งขึ้น เพื่อให้เข้าใจถึงที่มาและความสำคัญของปัญหาและวัตถุประสงค์ของการวิจัยนี้มากขึ้น ในบทนี้จึงประกอบด้วยเนื้อหาทั้งหมด 5 ส่วน ดังต่อไปนี้ (1) ที่มาและความสำคัญของปัญหา (2) วัตถุประสงค์ของการวิจัย (3) ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย (4) ขอบเขตของการวิจัย และ(5) เนื้อหาวิทยานิพนธ์ โดยมีรายละเอียดดังต่อไปนี้

1.1 ที่มาและความสำคัญของปัญหา

ความต้องการ (Requirement) ของผู้ใช้งานมักเกิดขึ้นอย่างต่อเนื่องและมีอยู่อย่างไม่จำกัด แต่ทั้งนี้จะถูกควบคุมภายใต้ข้อจำกัด (Constrains) ทางด้านงบประมาณ (Budget) เวลา (Time) และทรัพยากร (Resource) ดังนั้นในการพัฒนาซอฟต์แวร์เพื่อให้สามารถแก้ไขปัญหาของผู้ใช้งาน และนำไปสู่ความพึงพอใจสูงสุดของผู้ใช้งาน (Customer Satisfaction) จึงเป็นงานที่ท้าทายนักพัฒนาซึ่งจำเป็นต้องคำนึงถึงการจัดลำดับความสำคัญของความต้องการ (Requirements Prioritization) เหล่านั้นเพื่อการพัฒนาในขั้นตอนต่อไป

ความต้องการที่มากขึ้นเรื่อยๆทำให้จำเป็นต้องมีการทำการจัดลำดับความสำคัญของความต้องการเพื่อให้สามารถระบุความต้องการหลัก ลดข้อขัดแย้ง และช่วยสำหรับการวางแผนการพัฒนา ภายใต้ข้อกำหนด (Constrains) เรื่องงบประมาณ, คุณภาพ, ทรัพยากร, และเวลา เพื่อให้ตอบสนองความพึงพอใจสูงสุดแก่ลูกค้า (M. I. Babar et al., 2011) ซึ่งถ้าระบุความต้องการหลักผิด จะทำให้คุณภาพของซอฟต์แวร์ที่ผลิตลดลง

ในโครงการพัฒนาซอฟต์แวร์นั้นวิศวกรซอฟต์แวร์จะต้องพัฒนาซอฟต์แวร์ตามกระบวนการทางวิศวกรรมซอฟต์แวร์ โดย ISO/IEC CD 24765 ให้นิยามของคำว่าวิศวกรรมซอฟต์แวร์คือ การประยุกต์วิธีการที่เป็นระบบ มีกฎระเบียบ เชิงปริมาณ เพื่อพัฒนา ดำเนินการ และบำรุงรักษาซอฟต์แวร์ ซึ่ง

หมายถึงการประยุกต์ใช้วิศวกรรมสำหรับซอฟต์แวร์นั้นเอง ซึ่งกระบวนการทางวิศวกรรมซอฟต์แวร์ประกอบไปด้วย 5 กระบวนการ ได้แก่ (1) การเก็บและวิเคราะห์ความต้องการ (Requirements Specification or Requirements Analysis) (2) การออกแบบซอฟต์แวร์ (Software Design) (3) การสร้างซอฟต์แวร์ (Implementation and Integration) (4) การทดสอบ (Testing) รวมถึง (5) การติดตั้งและบำรุงรักษาซอฟต์แวร์ (Deployment and Maintenance) เป็นต้น

(1) การเก็บและวิเคราะห์ความต้องการ เป็นขั้นตอนที่เกี่ยวข้องกับการรวบรวมความต้องการจากผู้ที่เกี่ยวข้องกับซอฟต์แวร์ (Stakeholders) รวมถึงจัดการและต่อรองเมื่อเกิดข้อขัดแย้งระหว่างความต้องการจากผู้ที่เกี่ยวข้อง จากนั้นจึงนำความต้องการมาทำการวิเคราะห์ พร้อมทั้งจัดทำเอกสารความต้องการ (Requirements Specification) รวมไปถึงทำการตรวจสอบความถูกต้องและความครบถ้วนของความต้องการด้วย (Verification and Validation) เนื่องด้วยความต้องการเป็นส่วนประกอบพื้นฐานของขั้นตอนการพัฒนาที่เหลือต่อไป ดังนั้นหากเกิดข้อผิดพลาดในขั้นตอนนี้มากเท่าไหร่ ก็จะส่งผลกระทบต่อความสำเร็จของโครงการมากเท่านั้น ทั้งนี้เพื่อให้ได้ซอฟต์แวร์ที่มีคุณภาพตามที่กำหนด สามารถแก้ปัญหาให้แก่ผู้ใช้งานและนำไปสู่ความพึงพอใจสูงสุดของผู้ใช้งาน

(2) การออกแบบซอฟต์แวร์ คือ ขั้นตอนการนำความต้องการที่ได้จากขั้นตอนแรกมาศึกษา วิเคราะห์ออกแบบส่วนประกอบ ส่วนเชื่อมต่อและลักษณะต่างๆของซอฟต์แวร์ เพื่อช่วยให้ผู้พัฒนาและผู้ใช้งานมีความเข้าใจไปในทิศทางเดียวกัน โดยอาศัยเครื่องมือที่เป็นรูปแบบมาตรฐานช่วยในการออกแบบ เช่น ยูเอ็มเอล (UML: Unified Modeling Language) ซึ่งเป็นเครื่องมือที่ประกอบไปด้วยสัญลักษณ์ที่ช่วยในการออกแบบเพื่อนำเสนอโครงสร้าง และรายละเอียดของซอฟต์แวร์

กระบวนการออกแบบซอฟต์แวร์แบ่งออกเป็น 2 ลักษณะ คือการออกแบบเชิงรายละเอียด (Low Level Design หรือ Detailed Design) และการออกแบบภาพรวมของระบบ (High Level Design)ซึ่งการออกแบบเชิงรายละเอียด ได้แก่ การออกแบบส่วนประกอบย่อยของระบบ (Component Level Design) และ การออกแบบอัลกอลิทึม (Algorithm Design) เป็นต้น ส่วนการออกแบบภาพรวมของระบบได้แก่ การออกแบบทางสถาปัตยกรรม (Architecture Design) เป็นต้น ซึ่งสิ่งที่ได้จากขั้นตอนนี้คือรายละเอียดโครงสร้างของซอฟต์แวร์เพื่อนำไปใช้ในกระบวนสร้างและทดสอบซอฟต์แวร์ในขั้นตอนต่อไป

(3) การสร้างซอฟต์แวร์ คือขั้นตอนในการพัฒนาซอฟต์แวร์โดยอาศัยภาษาคอมพิวเตอร์ (Programming Language) ต่างๆ เช่น ภาษาจาว่า (Java Programming Language), ภาษาซี (C Programming Language), ภาษาซีชาร์ป (C# Programming Language) เป็นต้น โดยในการพัฒนา

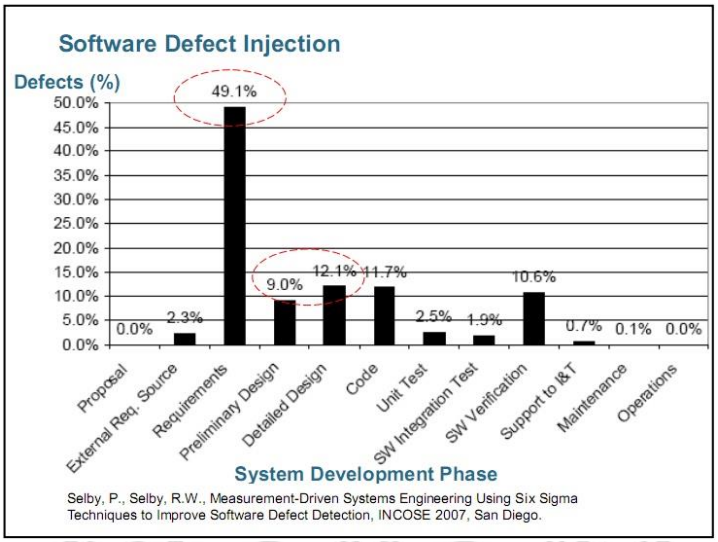
ซอฟต์แวร์นั้นนักพัฒนามักจะพัฒนาผ่านทางเครื่องมือต่างๆ ได้แก่ ไอดีอี (IDE: Integrated Development Environment) ซึ่งหมายถึงเครื่องมือที่ใช้ในการสร้างซอฟต์แวร์ และช่วยจัดเตรียมสภาพแวดล้อมให้เหมาะสมสำหรับการพัฒนาซอฟต์แวร์ ประกอบด้วย โปรแกรมแก้ไขรหัส (Code Editor) หมายถึงโปรแกรมที่ใช้สำหรับเขียน แก้ไข และเปลี่ยนแปลงโปรแกรม, คอมไพเลอร์ (Compiler) หมายถึงโปรแกรมที่ช่วยแปลภาษาคอมพิวเตอร์ให้เป็นภาษาเครื่อง เพื่อให้คอมพิวเตอร์สามารถเข้าใจได้ โดยมักจะอยู่ในรูปแบบของไบนารี (Binary Format) เช่น 0 หรือ 1, โปรแกรมดีบัก (Debugger) หมายถึงโปรแกรมที่ใช้สำหรับการทดสอบ (Test) และแก้ไขข้อพบพร่องของโปรแกรม ซึ่งการดีบักเป็นวิธีการที่ช่วยลดข้อผิดพลาดของซอฟต์แวร์ เป็นต้น ตัวอย่างไอดีอี เช่น Eclipse และ Microsoft Visual Studio เป็นต้น

(4) การทดสอบซอฟต์แวร์ เป็นขั้นตอนเพื่อหาข้อผิดพลาดของซอฟต์แวร์และวางแผนเพื่อหาวิธีการแก้ไขข้อผิดพลาดเหล่านั้น เพื่อให้ได้ซอฟต์แวร์ที่มีความถูกต้อง สมบูรณ์ ปลอดภัย และมีคุณภาพ ซึ่งในการทดสอบซอฟต์แวร์นั้นเกี่ยวข้องกับขั้นตอนหลายๆขั้นตอนเริ่มจากการวิเคราะห์ความต้องการเพื่อจัดทำแผนงานการทดสอบ จากนั้นทำการวางแผนเพื่อหาแนวทางการทดสอบก่อนนำไปทดสอบจริง เมื่อทำการทดสอบซอฟต์แวร์เสร็จสิ้น ผลการทดสอบจะถูกบันทึกและรายงาน หากพบข้อบกพร่อง ต้องทำการแก้ไขพร้อมทำการทดสอบซ้ำจนเสร็จสิ้น ทั้งนี้เพื่อให้มั่นใจว่าการแก้ไขหรือการเปลี่ยนแปลงที่เกิดขึ้นไม่ก่อให้เกิดข้อผิดพลาดใหม่ให้แก่ซอฟต์แวร์

กระบวนการการทดสอบซอฟต์แวร์มีรูปแบบการทดสอบทั้งหมด 5 รูปแบบ ได้แก่ การทดสอบระดับหน่วยย่อย (Unit Testing), การทดสอบระดับรวมหน่วย (Integration Testing), การทดสอบระบบ (System Testing), การทดสอบเพื่อยอมรับ (Acceptance Testing), และการทดสอบประสิทธิภาพการใช้งานของระบบ (Usability Testing)

(5) การติดตั้งและการบำรุงรักษาซอฟต์แวร์ เป็นขั้นตอนสุดท้ายหลังจากซอฟต์แวร์ถูกสร้างและทดสอบความถูกต้องสมบูรณ์จนได้ซอฟต์แวร์ที่มีคุณภาพตามที่กำหนดแล้ว ขั้นตอนนี้คือการส่งมอบและติดตั้งให้แก่ผู้ใช้งาน นอกเหนือจากการติดตั้งซอฟต์แวร์แล้ว กระบวนการนี้ยังครอบคลุมไปถึงการจัดฝึกอบรมการใช้งานเพื่อสร้างความรู้ความเข้าใจที่ดีเกี่ยวกับการใช้งานซอฟต์แวร์ นอกจากนี้ยังรวมไปถึงการสนับสนุนการใช้งานซอฟต์แวร์ให้แก่ผู้ใช้งานเมื่อพบข้อบกพร่อง หรือมีความต้องการเพิ่มเติมภายหลังจากการส่งมอบอีกด้วย

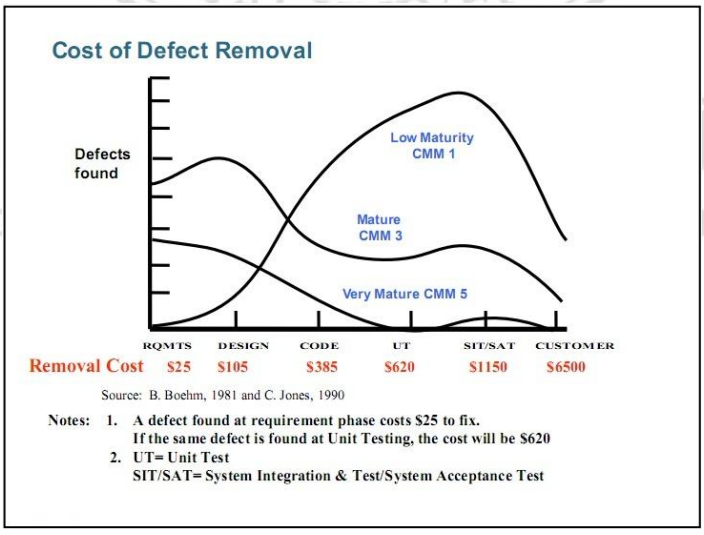
ทั้งนี้จากการศึกษาข้อผิดพลาดที่เกิดขึ้นในแต่ละขั้นตอนของการพัฒนาซอฟต์แวร์พบว่า ข้อผิดพลาดถูกค้นพบมากที่สุดอยู่ในขั้นตอนที่เกี่ยวข้องกับความต้องการดังแสดงในภาพที่ 1.1



ภาพที่ 1.1 จำนวนร้อยละการค้นพบข้อผิดพลาดในขั้นตอนการพัฒนาซอฟต์แวร์ (Selby et al., 2007)

จากภาพแสดงให้เห็นว่าร้อยละ 49.1 ของข้อผิดพลาดของซอฟต์แวร์ถูกค้นพบในขั้นตอนที่เกี่ยวข้องกับความต้องการ ซึ่งข้อผิดพลาดเหล่านี้มีมากกว่าข้อผิดพลาดที่ค้นพบในขั้นตอนการออกแบบรายละเอียดและขั้นตอนการออกแบบขั้นต้นที่ร้อยละ 12.1 และร้อยละ 9.0 ตามลำดับ

นอกจากการพบข้อผิดพลาดจากขั้นตอนที่เกี่ยวข้องกับความต้องการมากที่สุดแล้ว จากการศึกษายังพบว่าการแก้ไขข้อผิดพลาดในขั้นตอนดังกล่าวยังสูงกว่ากระบวนการอื่นๆเป็นอย่างมาก ดังแสดงในภาพที่ 1.2



ภาพที่ 1.2 ค่าใช้จ่ายในการแก้ไขข้อผิดพลาดในขั้นตอนต่างๆของกระบวนการพัฒนาซอฟต์แวร์ (Boehm, 1981; Jones, 1990)

จากภาพสามารถสรุปได้ว่าการค้นพบข้อผิดพลาดตั้งแต่ขั้นตอนที่เกี่ยวข้องกับความต้องการสามารถลดค่าใช้จ่ายในการแก้ไขข้อผิดพลาดได้เป็นอย่างมาก เมื่อเปรียบเทียบกับการค้นพบข้อผิดพลาดในกระบวนการอื่นๆ หากค้นพบข้อผิดพลาดในขั้นตอนที่เกี่ยวข้องกับความต้องการน้อย จะส่งผลให้เกิดค่าใช้จ่ายที่สูงในการแก้ไขข้อผิดพลาดนั้นๆในกระบวนการอื่นๆที่ตามมา

ตัวอย่างเช่น หากข้อผิดพลาดหนึ่งๆเกิดขึ้น ถูกค้นพบและดำเนินการแก้ไขในขั้นตอนการเก็บและวิเคราะห์ความต้องการซอฟต์แวร์มีค่าใช้จ่ายในการดำเนินการแก้ไขข้อผิดพลาดคิดเป็น 25 ดอลลาร์ต่อหนึ่งข้อผิดพลาด หากข้อผิดพลาดนี้ถูกค้นพบในขั้นตอนการสร้างซอฟต์แวร์ ค่าใช้จ่ายจะเพิ่มขึ้นเป็น 385 ดอลลาร์ต่อหนึ่งข้อผิดพลาด หรือหากข้อผิดพลาดนี้ถูกค้นพบโดยผู้ใช้งาน ค่าใช้จ่ายจะเพิ่มขึ้นสูงถึง 6,500 ดอลลาร์ต่อหนึ่งข้อผิดพลาด หรือคิดเป็นร้อยละ 26,000 ของค่าใช้จ่ายที่เพิ่มขึ้นเปรียบเทียบกับการค้นพบในขั้นตอนการเก็บและวิเคราะห์ความต้องการของซอฟต์แวร์ จากตัวอย่างดังกล่าวแสดงให้เห็นว่า ยิ่งค้นพบความผิดพลาดได้เร็ว ค่าใช้จ่ายในการแก้ไขก็จะลดน้อยลง

จากที่กล่าวมาข้างต้นสามารถสรุปได้ว่า กระบวนการพัฒนาซอฟต์แวร์ประกอบไปด้วยหลายขั้นตอน ซึ่งความต้องการเป็นหนึ่งในขั้นตอนที่มีความสำคัญและมีความจำเป็นสูง หากนักพัฒนาให้ความสำคัญและใส่ใจต่อขั้นตอนการเก็บและวิเคราะห์ความต้องการของซอฟต์แวร์ จะทำให้สามารถลดข้อผิดพลาดที่อาจจะเกิดขึ้น และนำไปสู่การลดค่าใช้จ่ายในการพัฒนาอีกด้วย

นอกจากนี้จากการศึกษาพบว่าโครงการพัฒนาซอฟต์แวร์มีอัตราความล้มเหลวของโครงการสูงอย่างมาก

ภาพที่ 1.3 รายงานผลการสำรวจโครงการพัฒนาซอฟต์แวร์ปี 2011 (CHAOS, 2011)

จากรายงานการสำรวจผลลัพธ์โครงการพัฒนาซอฟต์แวร์ประจำปี 2011 ของ CHAOS (CHAOS, 2011) พบว่ามีเพียงร้อยละ 42 ของโครงการพัฒนาซอฟต์แวร์ทั้งหมดที่สามารถดำเนินโครงการได้สำเร็จบรรลุตามเป้าหมาย (Successful Project) ซึ่งโครงการประเภทนี้สามารถพัฒนาตามความต้องการของลูกค้า และอยู่ภายใต้ข้อจำกัดด้านเวลาและงบประมาณที่ได้กำหนดไว้ อย่างไรก็ตามพบว่าร้อยละ 37 ของโครงการพัฒนาทั้งหมดเป็นโครงการที่ดำเนินการแล้วเสร็จแต่ล่าช้ากว่ากำหนดการและ/หรือใช้งบประมาณเกินกว่าที่กำหนด หรือสามารถส่งงานให้ลูกค้าได้ตามกำหนดการและตามงบประมาณที่ตั้งไว้ แต่ไม่สามารถพัฒนาความต้องการของลูกค้าได้ครบทั้งหมด (Challenged Project) นอกจากนี้ในปี 2011 พบว่าโครงการที่ล้มเหลวซึ่งไม่สามารถพัฒนาซอฟต์แวร์ได้ตามความต้องการของลูกค้า ภายใต้ระยะเวลาและงบประมาณที่กำหนดไว้ (Failed Project) มีจำนวนสูงถึงร้อยละ 21 ซึ่งเป็นตัวเลขที่สูงอย่างมาก

Stanislav ได้กล่าวว่าหนึ่งในเหตุผลที่ทำให้โครงการซอฟต์แวร์ล้มเหลว คือ การเรียงลำดับความต้องการตามความสำคัญผิด เนื่องจากการเรียงลำดับความต้องการตามความสำคัญที่ผิด จะนำไปสู่การออกแบบโครงสร้างที่ผิด ทำให้การพัฒนาเป็นไปได้ยาก และเกิดค่าใช้จ่ายสูง (Stanislav, 2012)

ในอดีตมีการนำเสนอวิธีการในการเรียงลำดับความสำคัญของความต้องการมากมาย ไม่ว่าจะเป็นวิธีการ Analytic Hierarchy Process (AHP) ซึ่งนำเสนอโดย Saaty ในช่วงปี 1970 (Saaty, 1970) โดยใช้ในการประเมินมูลค่าหรือราคาของความต้องการหนึ่งเทียบกับอีกความต้องการหนึ่ง (Pair Wise Evaluation) ตัวอย่างเช่น ค่า AHP ที่ 3 ของความต้องการที่หนึ่ง เทียบกับความต้องการที่สอง หมายถึงความต้องการที่หนึ่ง มีมูลค่ามากกว่าเป็นสามเท่าของความต้องการที่สอง และวิธีการที่สองคือ 100 Dollar methods หรือที่เรียกว่า Cumulative voting ซึ่งเป็นวิธีที่ค่อยข้างง่ายในการเรียงลำดับความสำคัญของความต้องการ โดยการสมมติให้ทุกคนมีเงินอยู่คนละ 100 ดอลลาร์ ดังนั้นทุกคนต้องตัดสินใจเพื่อจัดสรรเงินเหล่านี้เพื่อซื้อในสิ่งที่ต้องการแต่ละชิ้น ซึ่งวิธีการนี้จะได้ผลลัพธ์ในรูปแบบอัตราส่วน (Ratio Scale)

ถึงแม้ว่าวิธีการที่กล่าวมาข้างต้นจะสามารถใช้ในการแก้ปัญหาได้ดี แต่อย่างไรก็ตามในการพัฒนาซอฟต์แวร์ที่ประกอบด้วยความต้องการปริมาณมาก (Large-Scale of Requirements) ที่ประกอบด้วยความต้องการมากกว่า 100 ความต้องการขึ้นไปนั้น นักพัฒนาต้องเผชิญกับปัญหาในการเรียงลำดับความสำคัญของความต้องการเหล่านั้นอยู่เสมอ ซึ่งเป็นปัญหาที่มักจะพบเจอในการพัฒนาซอฟต์แวร์ในปัจจุบัน

ปัจจุบันเทคนิคในการเรียงลำดับความสำคัญของความต้องการมีมากมายซึ่งมีข้อดีและข้อจำกัดต่างกัน แต่ถึงอย่างไรก็ตาม ยังไม่มีหลักฐานที่สามารถแสดงได้ว่าเทคนิคเหล่านี้สามารถสนับสนุนการเรียงลำดับความสำคัญสำหรับโครงการที่มีจำนวนความต้องการมากๆ ได้ (Babar et al., 2011; Firesmith, 2004; Ma, 2009; Laurent et al., 2007; Huang et al., 2008)

มีนักวิจัยหลายกลุ่มพยายามนำเสนอวิธีการที่ช่วยในการเรียงลำดับความสำคัญของความต้องการขนาดใหญ่มากมายหลายวิธี ไม่ว่าจะเป็นวิธีการของ Beg (Beg, 2008) เรื่องการเรียงลำดับความสำคัญของความต้องการขนาดใหญ่โดยอาศัยวิธีของ Balance Search Tree ซึ่งในการทดลองของเขาได้กำหนดความต้องการทั้งหมด 121 ความต้องการ โดยแทนแต่ละโหนด (Node) ของต้นไม้ด้วยความต้องการ โหนดที่อยู่ทางซ้ายประกอบด้วยความต้องการที่มีความสำคัญน้อยกว่าโหนดที่อยู่ฝั่งขวาของต้นไม้ ซึ่งการเรียงลำดับความสำคัญจะเริ่มต้นจากโหนดแรกบนสุด หากต้องการหาความต้องการที่มีค่าความสำคัญน้อยสุด จะทำการเปรียบเทียบความต้องการตามโหนดฝั่งซ้ายไปจนกระทั่งไม่มีโหนดใดๆ ให้เปรียบเทียบอีกต่อไป ในทางกลับกัน หากต้องการค้นหาความต้องการที่มีความสำคัญสูงสุด จะเริ่มต้นทำการเปรียบเทียบตั้งแต่โหนดแรกไปทางฝั่งขวาจนกระทั่งไม่มีโหนดใดๆให้เปรียบเทียบอีกต่อไป ซึ่งผลลัพธ์ที่ได้จากการศึกษาในครั้งนี้คือจำนวนในการเปรียบเทียบความต้องการลดลงและมีความซับซ้อนน้อยกว่าเมื่อเทียบกับวิธีการ AHP ทั้งนี้การทดลองนี้ยังไม่ได้คำนึงถึงความถูกต้องของการเรียงลำดับใดๆทั้งสิ้น

เพื่อให้นักพัฒนาสามารถแก้ปัญหาในการเรียงลำดับความต้องการขนาดใหญ่ได้ง่ายขึ้น งานวิจัยนี้จึงได้ทำการศึกษาเพื่อประยุกต์ใช้เครื่องมือที่มีอยู่เหล่านี้นำมาพัฒนาแบบจำลองในการเรียงลำดับก่อนหลังของความต้องการขนาดใหญ่ โดยการศึกษาครั้งนี้จะมุ่งเน้นในการแก้ปัญหา 2 ประการต่อไปนี้

ประการแรก เนื่องจากในการเรียงลำดับความต้องการขนาดใหญ่นั้น จำเป็นต้องใช้ค่าความพยายามในการเรียงลำดับเป็นจำนวนมาก ดังนั้นในการทำวิจัยครั้งนี้เป็นการศึกษาเพื่อมุ่งเน้นการพัฒนาโมเดลที่สามารถช่วยนักพัฒนาซอฟต์แวร์ในการเรียงลำดับความสำคัญของความต้องการขนาดใหญ่โดยสามารถช่วยลดความพยายามในการเรียงลำดับความสำคัญของความต้องการได้

ประการที่สอง การส่งมอบซอฟต์แวร์ที่สามารถตอบสนองความต้องการและสร้างความพึงพอใจให้กับผู้ใช้งานคือปัจจัยหลักของความสำเร็จของโครงการ แต่เนื่องด้วยข้อจำกัดทางด้านเวลา งบประมาณและทรัพยากรในการพัฒนาซอฟต์แวร์ ทำให้นักพัฒนาจำเป็นต้องอาศัยการเรียงลำดับความสำคัญของความต้องการในการบริหารจัดการความต้องการเพื่อให้บรรลุเป้าหมายที่วางไว้ ดังนั้น

การวิจัยในครั้งนี้จึงมุ่งเน้นการพัฒนาแบบจำลองที่สามารถช่วยให้ผู้พัฒนาสามารถเรียงลำดับความสำคัญของความต้องการให้มีความถูกต้อง (Accuracy) มากที่สุด

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อนำเสนอแบบจำลองการจัดลำดับก่อนหลังสำหรับความต้องการขนาดใหญ่ โดยที่แบบจำลองที่ได้จากการวิจัยนี้จะถูกนำไปใช้ในการจัดเรียงลำดับในอุตสาหกรรมซอฟต์แวร์จริง

1.2.2 เพื่อวัดผลประสิทธิภาพจากการประยุกต์ใช้แบบจำลองการจัดลำดับก่อนหลังสำหรับความต้องการขนาดใหญ่ ทั้งนี้ในการวัดประสิทธิภาพของแบบจำลองนั้นจะพิจารณาจากประสิทธิภาพทั้ง 2 ด้าน ได้แก่ ด้านค่าความพยายามและด้านค่าความถูกต้องในการจัดเรียงลำดับก่อนหลัง

1.3 ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย

1.3.1 ได้แบบจำลองการจัดลำดับความต้องการก่อนหลังสำหรับความต้องการขนาดใหญ่ที่มีประสิทธิภาพในด้านของค่าความพยายามในการจัดเรียงลำดับที่ลดน้อยลงและมีค่าความถูกต้องในการจัดลำดับที่มากขึ้น

1.3.2 บริษัทพัฒนาซอฟต์แวร์ที่นำแบบจำลองการจัดลำดับก่อนหลังสำหรับความต้องการขนาดใหญ่ซึ่งได้จากการศึกษานี้ไปประยุกต์ใช้ มีผลประเมินด้านการจัดลำดับก่อนหลังความต้องการดีขึ้น

1.4 ขอบเขตของการวิจัย

1.4.1 แบบจำลองการจัดลำดับความต้องการก่อนหลังสำหรับความต้องการขนาดใหญ่ที่นำเสนอในงานวิจัยฉบับนี้จะออกแบบให้เกิดความถูกต้อง (Accuracy) และลดค่าความพยายาม (Effort) ในการเรียงลำดับความต้องการก่อนหลัง

1.4.2 การวัดประสิทธิภาพของแบบจำลองจะทำการวัดผลลัพธ์ออกเป็นสองประเภทใหญ่ๆ ได้แก่การวัดประสิทธิภาพจากค่าความพยายามและการวัดจากความถูกต้องของการเรียงลำดับก่อนหลังของความต้องการ ซึ่งค่าความพยายามวัดจากจำนวนความต้องการที่ถูกนำมาเรียงลำดับ และจำนวนการเปรียบเทียบในการเรียงลำดับ โดยที่ความถูกต้องของการเรียงลำดับวัดโดยการเปรียบเทียบผลลัพธ์ที่ได้จากการเรียงลำดับโดยใช้แบบจำลองที่

นำเสนอกับการเรียงลำดับที่ถูกต้องของการเรียงลำดับโดยใช้วิธีก่อนหน้านี้ที่ใช้จำนวนความต้องการเท่ากัน

1.4.3 เก็บข้อมูลจากบริษัทซอฟต์แวร์ในจังหวัดเชียงใหม่ที่ดำเนินการผลิตซอฟต์แวร์ที่ประกอบด้วยความต้องการมากกว่า 100 ความต้องการขึ้นไป โดยโครงการดำเนินการในช่วงระหว่างเดือนสิงหาคม 2556 ถึงเดือนธันวาคม 2556

1.5 เค้าโครงวิทยานิพนธ์

วิทยานิพนธ์เล่มนี้ประกอบด้วยเนื้อหา 6 บท โดยแต่ละบทมีสาระดังต่อไปนี้

บทที่ 1 เป็นบทนำ ซึ่งในบทนี้จะกล่าวถึงความเป็นมาและความสำคัญของปัญหา (Context and Problematic) ซึ่งในบทนี้จะอธิบายถึงความหมายและกระบวนการทางวิศวกรรมซอฟต์แวร์ (Software Engineering) รวมถึงปัญหาที่พบในการพัฒนาโครงการซอฟต์แวร์ นอกจากนี้มีการอธิปรายถึงความสำคัญของกระบวนการพัฒนาความต้องการ ถึงผลกระทบต่อความสำเร็จของโครงการ ผลกระทบและความรุนแรงที่เกิดขึ้นเมื่อเกิดข้อผิดพลาดจากกระบวนการนี้ รวมถึงกล่าวถึงสถานการณ์ปัจจุบันของกระบวนการวิศวกรรมความต้องการ(Requirement Engineering) เกี่ยวกับประเด็นที่ท้าทาย (Challenging Issues) พร้อมแนวทางและวิธีการในการแก้ไขปัญหา นอกจากนี้ยังอธิบายถึงความสำคัญของปัญหาในการเรียงลำดับความสำคัญของความต้องการขนาดใหญ่ไว้อีกด้วย

นอกจากนี้ยังอธิบายถึงวัตถุประสงค์ของงานวิจัย ประโยชน์ที่จะได้รับจากการศึกษา ขอบเขตงานวิจัย และเนื้อหาวิทยานิพนธ์นี้อีกด้วย

บทที่ 2 เป็นส่วนของการทบทวนวรรณกรรม เอกสาร และงานวิจัยที่เกี่ยวข้องทั้งหมด (Literature reviews) ซึ่งกล่าวถึงวิศวกรรมความต้องการ เกี่ยวกับความหมาย และกระบวนการต่างๆที่เกี่ยวข้อง เช่นกระบวนการวิเคราะห์ความต้องการ (Requirement analysis) กระบวนการบริหารจัดการความต้องการ (Requirement management) เป็นต้น เพื่อให้เข้าใจภาพรวมของกระบวนการวิศวกรรมความต้องการทั้งหมด รวมไปถึงการเรียงลำดับความสำคัญของความต้องการ โดยกล่าวถึงความหมาย เกณฑ์การตัดสินใจ (Criteria) เทคนิคในการเรียงลำดับความสำคัญ (Requirements prioritizing techniques) และกล่าวถึงภาพรวมในการเลือกเทคนิคในการเรียงลำดับความสำคัญขนาดใหญ่อีกด้วย

บทที่ 3 อธิบายวิธีการวิจัย (Research methodology) ซึ่งในบทนี้จะอธิบายการประยุกต์ใช้เครื่องมือในการเรียงลำดับความสำคัญของความต้องการเพื่อให้สามารถจัดการกับความต้องการขนาดใหญ่ได้ อีกทั้งอธิบายในส่วนของกรอบการวิจัย (Research framework) ซึ่งได้แก่ การสอบถามความต้องการและ

เกณฑ์การตัดสินใจ (Requirements and criteria elicitation), การเลือกเทคนิคในการเรียงลำดับความสำคัญของความต้องการ (Techniques selection), และการประเมินผล (Evaluation) เป็นต้น รวมถึงการอธิบายรายละเอียดในแต่ละขั้นตอนตามกรอบการวิจัยอีกด้วย

บทที่ 4 กล่าวถึงผลการศึกษาของงานวิจัยในครั้งนี้ (Analysis and results) ซึ่งอธิบายรายละเอียดของโมเดลที่นำเสนอ (Proposed model) รวมถึงอธิบายส่วนประกอบของโมเดลทั้งหมด

บทที่ 5 เป็นส่วนของการสรุปและอธิปรายผลลัพธ์จากการประยุกต์ใช้โมเดลจริง (Application) นอกจากนี้ยังอธิบายรายละเอียดของขั้นตอนการจัดเก็บข้อมูล การทดลองใช้โมเดลในโครงการที่เป็นกรณีศึกษา และการวัดผลจากการทดลองกับกรณีศึกษาอีกด้วย

บทที่ 6 เป็นส่วนของการสรุปและอธิปรายผลการศึกษาวิจัย (Conclusion and future work) ซึ่งอธิบายรายละเอียดของผลการดำเนินการวิจัย ปัญหาและอุปสรรคของการวิจัย ข้อจำกัดและข้อเสนอแนะสำหรับงานวิจัยในครั้งนี้