การพัฒนาระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์
Development of Software Defect Tracking and Management System

ประพาฬ กาญจโนภาศ[1]* และ เสาวณี แซ่ตั้ง[2]

[1,2]อาจารย์ ภาควิชาวิศวกรรมซอฟต์แวร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยกรุงเทพ กรุงเทพฯ 10110

บทคัดย่อ

การวิจัยและพัฒนาระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์นี้เพื่อพัฒนาระบบสารสนเทศที่สนับสนุนการติดตามและจัดการข้อผิดพลาดให้สะดวกและง่ายยิ่งขึ้นจากการเป็นศูนย์กลางการประสานงานระหว่างบุคคลต่าง ๆ ในทีมพัฒนาซอฟต์แวร์ของแต่ละโครงการพัฒนาซอฟต์แวร์ (Software Project) ช่วยสร้างการสื่อสารและความเข้าใจที่ตรงกันเกี่ยวกับการติดตามและจัดการข้อผิดพลาดได้อย่างมีประสิทธิภาพมากสูงขึ้น จุดเด่นของตัวระบบคือ ฟังก์ชันการทำงานต่าง ๆ ใช้งานง่ายไม่ซับซ้อน โดยการวิเคราะห์และออกแบบระบบจะรวบรวมความต้องการจากทีมพัฒนาซอฟต์แวร์ในบริษัทผู้ผลิตซอฟต์แวร์ขนาดกลางแล้วทำการพัฒนาระบบติดตามและจัดการข้อผิดพลาดที่ยึดกระบวนการพัฒนาตามกรรมวิธีการพัฒนาแบบอาไจล์ (Agile) หลังจากนั้นจึงให้ทีมพัฒนาซอฟต์แวร์ในบริษัทดังกล่าวประเมินผลความพึงพอใจในการใช้งานระบบสารสนเทศ และประเมินประสิทธิภาพจากการลดระยะเวลาของการติดตามและจัดการข้อผิดพลาดเปรียบเทียบกันระหว่างการใช้และไม่ใช้ระบบการติดตามและจัดการข้อผิดพลาดซอฟต์แวร์นี้ ผลของการวิจัย พบว่า ผู้ประเมินมีความพึงพอใจโดยรวมอยู่ในระดับมาก ด้วยคะแนนเฉลี่ย 4.15 จาก 5 คะแนน และพบว่า ระบบนี้ยังสามารถช่วยลดระยะเวลาในการจัดการข้อผิดพลาดได้มากกว่า 15% โดยการเปรียบเทียบระยะเวลาตั้งแต่พบข้อผิดพลาดจนกระทั่งจัดการข้อผิดพลาดนั้นเสร็จสิ้นของโครงการที่ไม่ได้ใช้งานและใช้งานระบบการจัดการข้อผิดพลาดนี้ แต่อย่างไรก็ตาม ระบบสารสนเทศนี้ควรมีการสร้างเอกสารคู่มือการใช้งานออนไลน์เพื่อช่วยให้ผู้ใช้งานสามารถเรียนรู้และทบทวนการใช้งานได้รวดเร็วยิ่งขึ้นกว่าเดิม

Abstract

This research aims to develop software defect tracking and management system for faster performance of software development team. The proposed system will allow the team to interact with each other more effectively in each software project, which can lead to better communication and minimal misunderstanding. The advantage of this research is to simplify the software defect tracking and management system. Software development team who works in medium software company was interviewed to gather requirements of the system. Agile methodology was used in the system development process. Evaluation of the developed system was performed through test cases and users' satisfaction. Results of the research indicate that the development team was satisfied with this system in high level. Furthermore, this system could reduce the duration of defect tracking and management by more than 15 percent. However, online user manual should be provided for faster study and practice with the system.

คำสำคัญ : ข้อผิดพลาดซอฟต์แวร์ การติดตามข้อผิดพลาด กรรมวิธีการพัฒนาแบบอาไจล์ การพัฒนาระบบ

Keywords : Software Defect, Defect Tracking, Agile Methodology, Software Development

** ผู้นิพนธ์ประสานงานไปรษณีย์อิเล็กทรอนิกส์ praparl.k@bu.ac.th โทร. 0 2350 3500 ต่อ 1690*

1. บทนำ

1.1 ความเป็นมาและความสำคัญ

ในปัจจุบันตลาดการพัฒนาซอฟต์แวร์สำเร็จรูป (Package Software) หรือซอฟต์แวร์ที่ถูกว่าจ้างให้พัฒนาขึ้นมาโดยเฉพาะด้านเพื่อให้เหมาะสมกับงานหรือกิจกรรมการดำเนินงาน (Outsource Software Development) ต่าง ๆ มีการขยายธุรกิจการพัฒนาซอฟต์แวร์อย่างต่อเนื่อง (เพ็ญจิรา คันธวงศ์, 2010) แต่ในการพัฒนาซอฟต์แวร์นั้นนอกจากจะต้องตอบสนองความต้องการของธุรกิจแล้ว ซอฟต์แวร์ที่ผลิตขึ้นต้องมีคุณภาพ โดยต้องลดปริมาณข้อผิดพลาดให้เหลือน้อยที่สุดเท่าที่จะเป็นไปได้ เพื่อให้ซอฟต์แวร์ที่พัฒนาขึ้นมีความน่าเชื่อถือ (Reliability) การตรวจสอบและทดสอบซอฟต์แวร์ถือเป็นส่วนสำคัญในกระบวนการพัฒนาซอฟต์แวร์ โดยทีมพัฒนาซอฟต์แวร์จะต้องดูแลและตรวจหาข้อผิดพลาดซอฟต์แวร์ (Software Defect) ตั้งแต่เริ่มต้นไปตลอดจนครบวงจรการพัฒนาซอฟต์แวร์ (Software Development Life Cycle) ซึ่งขั้นตอนและกระบวนการตรวจสอบและทดสอบซอฟต์แวร์นั้น ผู้ที่มีหน้าที่รับผิดชอบและเกี่ยวข้องกัน ได้แก่ ผู้จัดการโครงการ (Project Manager) นักวิเคราะห์ความต้องการ (Business Analysis) นักพัฒนา (Developer) นักทดสอบ (Tester) และผู้ควบคุมคุณภาพ (Quality Assurance) ผู้รับผิดชอบแต่ละคนในทีมพัฒนาซอฟต์แวร์จะมีหน้าที่ตรวจหา ทดสอบ และแก้ไขข้อผิดพลาดตามที่ได้รับมอบหมาย โดย Lethbridge (2003) กล่าวว่าทีมพัฒนาซอฟต์แวร์จำเป็นต้องใช้ระบบติดตามข้อผิดพลาด ซึ่งถือว่าเป็นสิ่งสำคัญในการเก็บประวัติข้อมูล ซึ่งจากการทำวิจัย พบว่านักพัฒนาจะใช้ระบบในการปรับปรุงข้อมูลของข้อผิดพลาด เพื่อช่วยจัดเก็บข้อมูลของข้อผิดพลาดที่ถูกต้องและเป็นข้อมูลล่าสุด สามารถนำมาใช้ประโยชน์ในงานวิเคราะห์ และการปรับปรุงรูปแบบหรือกระบวนการในการพัฒนาซอฟต์แวร์ได้ โดยระบบดังกล่าวที่ได้รับความนิยมมากในระดับสากลและถือเป็นโอเพ่นซอร์สซอฟต์แวร์ (Open Source Software) คือ Bugzilla ที่ช่วยในการติดตามข้อผิดพลาด และการเปลี่ยนแปลงของโค้ด (Code) การส่งหรือทบทวนแพตช์ (Patch) การจัดการคุณภาพ การติดต่อสื่อสารระหว่างทีมผู้พัฒนาและอื่น ๆ (Menzies, T: 2008, 346-355) แต่ในขณะที่ Smart, J.F. (2007) และ Barnson, M.P. (2001) ให้ความคิดเห็นที่ตรงกันว่า Bugzilla เป็นระบบติดตามข้อผิดพลาดซอฟต์แวร์ที่ดี แต่มีปัญหาความยุ่งยากในการติดตั้งและการบำรุงรักษา ซึ่งไม่เหมาะสำหรับผู้ไม่เคยฝึกใช้งานมาก่อน นอกจากนี้ ยังมี Mantis ซึ่งเป็นอีกหนึ่งซอฟต์แวร์ในการจัดการข้อผิดพลาดที่ได้รับความนิยมและมีจุดต่าง คือ มี Custom Field เพื่อให้ประยุกต์ใช้งานในรูปแบบที่ผู้ใช้แต่ละคนต้องการ ซึ่ง Bugzilla ไม่มี แต่อย่างไรก็ตาม การออกแบบส่วนต่อประสานยุ่งยาก และมีการแสดงรายละเอียดต่าง ๆ เยอะเกินไปในแต่ละหน้าต่างการใช้งาน

ดังนั้น ผู้วิจัยจึงมีแนวคิดในการพัฒนาระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์ (Software Defect Tracking and Management System) เพื่อช่วยในด้านการประสานงานและเป็นศูนย์กลางระหว่างทีมผู้พัฒนาและทีมทดสอบระบบของแต่ละโครงการพัฒนาซอฟต์แวร์ ช่วยให้การควบคุมและดูแลติดตามข้อผิดพลาดนั้นเป็นไปได้ง่ายมากยิ่งขึ้น และช่วยให้เกิดความเข้าใจที่ตรงกันในทีมพัฒนาซอฟต์แวร์ (Avram, G., 2007) ในขณะที่ไม่เกิดความซ้ำซ้อนของข้อมูล

(Jalbert, 2008: 52-61) และลดระยะเวลาในการดำเนินการด้านเอกสารต่าง ๆ ที่ต้องส่งต่อกันระหว่างทีมพัฒนา และทีมทดสอบระบบ และจุดเด่นของตัวระบบ คือ ใช้งานง่ายไม่ซับซ้อน และมีฟังก์ชันงานที่สำคัญและเหมาะกับการใช้งานจริง โดยการวิเคราะห์และออกแบบระบบจะรวบรวมความต้องการขั้นพื้นฐานจากทีมพัฒนาซอฟต์แวร์ในบริษัทผู้ผลิตซอฟต์แวร์ขนาดกลาง ซึ่งการพัฒนาระบบติดตามและจัดการข้อผิดพลาดจะยึดตามกรรมวิธีการพัฒนาแบบอาไจล์ (Agile) จนได้ระบบที่สมบูรณ์ หลังจากนั้นจึงให้ทีมพัฒนาซอฟต์แวร์ประเมินผลความพึงพอใจในการใช้งานระบบ และประเมินประสิทธิภาพจากการลดระยะเวลาของการติดตามและจัดการข้อผิดพลาดเปรียบเทียบระหว่างการใช้และไม่ใช้งานระบบการติดตามและจัดการข้อผิดพลาดซอฟต์แวร์นี้

1.2 วัตถุประสงค์ของการศึกษา

1. เพื่อการศึกษาความต้องการพัฒนาติดตามและจัดการข้อผิดพลาดซอฟต์แวร์
2. เพื่อพัฒนาระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์
3. เพื่อศึกษาความพึงพอใจในการใช้งานระบบติดตามและจัดการข้อผิดพลาดในการพัฒนาระบบและหาแนวทางในการพัฒนาต่อไป

2. วิธีการศึกษา

การดำเนินงานวิจัยและพัฒนามีรายละเอียดดังนี้

2.1 วางแผนการดำเนินโครงการ โดยแบ่งระยะการพัฒนาออกเป็น 9 รอบ (Sprint) ตามหลักกรรมวิธีการพัฒนาแบบอาไจล์ (Agile) มีการกำหนดจำนวนวันในการพัฒนาในแต่ละรอบเป็น 7 วัน และในแต่ละรอบการพัฒนาจะถูกแบ่งออกเป็น 3 ส่วน คือ แผนงาน (Plan) สิ่งส่งมอบที่ได้ (Deliverables) และกิจกรรมที่ต้องกระทำ (Task Name)

2.2 ศึกษาความต้องการขั้นต้นของระบบโดยเข้าไปเก็บความต้องการจากทีมผู้พัฒนาซอฟต์แวร์ในบริษัทขนาดกลาง วิธีการที่ใช้เก็บความต้องการ คือ การสัมภาษณ์และใช้เทคนิคการทำต้นแบบ (Prototyping) โดยกิจกรรมนี้กระทำในทุก ๆ รอบการพัฒนาจนกระทั่งได้ระบบที่พัฒนาเสร็จสมบูรณ์ สิ่งที่ได้ดำเนินการวิเคราะห์ ระหว่างขั้นตอนนี้ ทำให้ได้แผนภาพความสัมพันธ์ระหว่างเอนทิตีเพื่อนำเสนอถึงโครงสร้างฐานข้อมูล ผลการออกแบบส่วนต่อประสานผู้ใช้ การออกแบบโครงสร้างเชิงสถาปัตยกรรม

2.3 พัฒนาระบบต้นแบบโดยใช้ Microsoft Visual C# .Net และเชื่อมต่อกับระบบการจัดการฐานข้อมูล SQL Server โดยพัฒนาระบบบน Microsoft Windows XP ให้สำเร็จตามแผนที่วางไว้ในแต่ละรอบการพัฒนา และสอดคล้องกับงานวิเคราะห์ที่ได้มาอย่างสมบูรณ์

2.4 ทดสอบระบบด้วยการทดลองใช้งานจริงในบริษัท โดยดำเนินการสาธิตการใช้งานระบบก่อน หลังจากนั้นให้ผู้ใช้งานประเมินความพึงพอใจในการใช้งานระบบ พร้อมทั้งประเมินประสิทธิภาพด้านการใช้เวลาในการติดตามและจัดการข้อผิดพลาดซอฟต์แวร์ก่อนและหลังการใช้งานระบบนี้เพื่อนำผลการประเมินมาใช้ปรับปรุงต่อไป

3. ผลการศึกษาและอภิปรายผล

3.1 ผลการศึกษาความต้องการพัฒนาติดตามและจัดการข้อพิดพลาดซอฟต์แวร์

การศึกษาความต้องการของระบบขั้นต้นโดยเข้าไปเก็บความต้องการจากทีมผู้พัฒนาซอฟต์แวร์จริงในบริษัท กำหนดขอบเขตในการพัฒนาโดยแบ่งฟังก์ชันการทำงานที่จำเป็น ออกเป็น 5 ส่วน ที่สามารถสนับสนุนกระบวนการพัฒนาซอฟต์แวร์ใด ๆ ได้แก่

3.1.1 การกำหนดสิทธิ์ผู้ใช้งานระบบ (System Authorization and Authentication) ซึ่งระบบสามารถกำหนดและจำกัดสิทธิ์การใช้งานของผู้ใช้งานกลุ่มต่าง ๆ อันได้แก่ ผู้ดูแลระบบ (System Admin) ผู้ดูแลผู้ใช้งาน (User Admin) ผู้บริหารโครงการ (Project Manager) นักวิเคราะห์ความต้องการของระบบ (Business Analysis) นักพัฒนา (Developer) ผู้ควบคุมคุณภาพในการพัฒนาระบบหรือนักทดสอบระบบ (Quality Assurance, Tester) เพื่อให้มีสิทธิ์การทำงานตามขั้นตอนต่าง ๆ ที่เกี่ยวข้องในการพัฒนาซอฟต์แวร์และมีอยู่ภายใต้กลุ่มโครงงานที่ตนเองรับผิดชอบเท่านั้น

3.1.2 การกำหนดโครงการ (Project Initiate) สามารถบริหารจัดการข้อมูลพื้นฐานโครงการสำหรับการพัฒนาซอฟต์แวร์ ประกอบด้วย 1) การกำหนดสิทธิ์และบัญชีผู้ใช้สำหรับโครงการ 2) การกำหนดกลุ่มบริษัทภายใต้โครงการพัฒนา 3) การสร้างโครงการใหม่และปรับปรุงข้อมูลโครงการ 4) การกำหนดข้อมูลผลิตภัณฑ์ซอฟต์แวร์ที่จะพัฒนา และ 5) การบริหารจัดการข้อมูลรุ่นของผลิตภัณฑ์

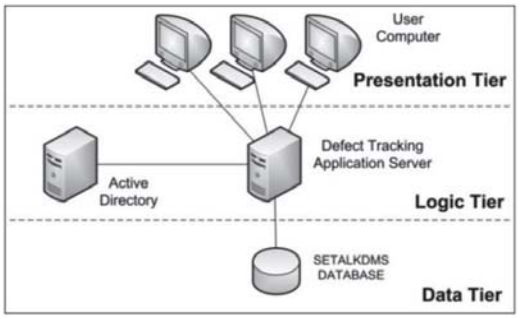
3.1.3 การบริหารจัดการข้อผิดพลาด (Defect Management) โดยฟังก์ชันการทำงานส่วนนี้ยอมให้ผู้พัฒนา อันได้แก่ นักวิเคราะห์ความต้องการ นักพัฒนา นักทดสอบระบบ สามารถจัดเก็บข้อมูลข้อผิดพลาดที่ค้นพบอยู่ระหว่างขั้นตอนของการทดสอบระบบ แบ่งออกเป็น 1) การเริ่มจับข้อผิดพลาด (Raise Defect) และจัดเก็บข้อมูลข้อผิดพลาด 2) การปรับปรุงสถานะของข้อผิดพลาดให้สอดคล้องกับกระแสงาน 3) การแสดงสถานะของข้อผิดพลาดทั้งหมด และ 4) การปรับปรุงรายละเอียดให้สมบูรณ์เมื่อข้อผิดพลาดนั้นถูกแก้ไขเรียบร้อยแล้ว

3.1.4 การค้นหารายละเอียดข้อมูลข้อผิดพลาด (Searching) สามารถค้นหาข้อมูลข้อผิดพลาดต่าง ๆ ได้ตามคำค้น (Keyword) เช่น ชื่อข้อผิดพลาด สถานะของข้อผิดพลาด และเลขที่ เป็นต้น

3.1.5 การออกรายงาน (Reporting) เป็นส่วนที่นำเสนอถึงผลการสรุปข้อผิดพลาดทั้งหมดที่ระบบได้จัดเก็บ และสามารถแบ่งแยกประเภทแสดงผลรายงานข้อผิดพลาดในรูปแบบต่าง ๆ

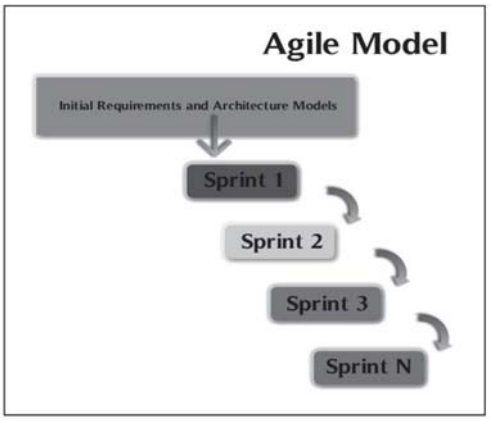
3.2 พลการพัฒนาระบบติดตามและจัดการข้อพิดพลาดซอฟต์แวร์

3.2.1 การพัฒนาระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์นั้น ผู้พัฒนาทำการออกแบบให้ระบบมีโครงสร้างของสถาปัตยกรรมการใช้งานบนเครือข่ายแบบเครื่องแม่ข่าย-ลูกข่าย 3 ชั้น (3-tier Client-Server) ดังแสดงในรูปที่ 1 โดยส่วนที่เป็นชั้นการนำเสนอ (Presentation Tier) นั้น เป็นส่วนที่ผู้ใช้ระบบเข้าใช้งานผ่านโปรแกรมเบราว์เซอร์ (Browser) และผลลัพธ์ในการทำงานต่าง ๆ ถูกแสดงออกมาให้ผู้ใช้งานได้รับทราบ ส่วนที่เป็นชั้นตรรกะ (Logic Tier) เป็นส่วนที่ดำเนินการด้านการประมวลผลข้อมูลและตรรกะทั้งหมด เช่น การรับข้อมูลนำเข้าที่ได้จากชั้นการนำเสนอเพื่อมาประมวลผลและส่งผลลัพธ์กลับไปยังชั้นการนำเสนอ รวมทั้งการส่งข้อมูลเข้าไปจัดเก็บยังชั้นข้อมูล (Data Tier) หรือการร้องขอข้อมูลที่ต้องการจากชั้นข้อมูลมาประมวลผลแล้วส่งไปแสดงผลยังชั้นนำเสนอ



รูปที่ 1 โครงสร้างสถาปัตยกรรมการใช้งานระบบ

3.2.2 การออกแบบกระบวนการในการพัฒนา ใช้หลักการออกแบบกระบวนการหรือวัฏจักรการพัฒนาซอฟต์แวร์ตามกรรมวิธีแบบอาไจล์ แสดงดังรูปที่ 2 โดยผู้พัฒนาให้ความสำคัญในการพัฒนาระบบเป็นอันดับแรก ส่วนเรื่องเอกสารเป็นอันดับรองลงมา เน้นใช้ระยะเวลาการพัฒนาซอฟต์แวร์ให้น้อยที่สุด และครอบคลุมตามความต้องการของผู้ใช้ได้ครบถ้วนมากที่สุด แบ่งออกเป็น 2 ระยะ คือ 1) ระยะเริ่มต้น (Initial Phase) คือ ระยะที่ผู้พัฒนาสื่อสารกับผู้ใช้งานมีการวางแผนงาน การระบุความต้องการขั้นต้น (Backlog) การวิเคราะห์และออกแบบระบบเพื่อกำหนดขอบเขตของโครงสร้างสถาปัตยกรรม มีการสร้างแบบจำลองและเอกสารงานออกแบบต่าง ๆ เท่าที่จำเป็น และ 2) ระยะการพัฒนา (Development Phase) โดยผู้พัฒนาได้แบ่งระยะการพัฒนาออกเป็น 9 รอบพัฒนา (Sprint) แบ่งกิจกรรมในแต่ละรอบการพัฒนาออกเป็น 3 ส่วนย่อย คือ การออกแบบส่วนต่อประสาน (Interface Design) การเขียนโค้ด (Coding) และทดสอบ (Testing) และการส่งมอบ (Deliver) ซอฟต์แวร์ให้ผู้ใช้ได้ทดลองใช้



รูปที่ 2 กระบวนการพัฒนาตามกรรมวิธีแบบอาไจล์

3.2.3 แบบจำลองระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์ ได้แก่ 1) การออกแบบส่วนต่อประสานผู้ใช้ นำเสนอเป็นแผนผังส่วนต่อประสาน ผู้ใช้ (User Interface Map) ดังแสดงในรูปที่ 3 และแผนภาพการแตกโครงสร้างของงาน (Work Breakdown Structure) ดังแสดงในรูปที่ 4 2) การออกแบบระบบฐานข้อมูลเชิงสัมพันธ์ นำเสนอด้วยแผนภาพความสัมพันธ์ระหว่างเอนทิตีของระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์ ดังรูปที่ 5 3) ขอบเขตการพัฒนานำเสนอโดยใช้แผนภาพยูสเคส ดังรูปที่ 6 4) และตัวอย่างจอภาพของระบบติดตามและจัดการข้อผิดพลาดที่พัฒนาขึ้นจริง แสดงดังรูปที่ 7-10

3.3 ผลการประเมินความพึงพอใจในการใช้งานระบบติดตามและจัดการข้อผิดพลาดในการพัฒนาระบบ

ผู้วิจัยได้ติดตั้งระบบซึ่งพัฒนาถ่ายทอดการใช้งานระบบด้วยการบรรยาย และสาธิตวิธีการใช้งานแบบรายกลุ่ม และให้ผู้ใช้ได้นำไปทดลองใช้งานจริง โดยมีกลุ่มที่เข้าร่วมการทดสอบจำนวน 4 กลุ่ม ได้แก่ Project Manager, Business Analysis,

Developer และ Tester โดยให้ทุกกลุ่มได้ทดลองใช้กับโครงการจำนวน 2 โครงการ จากนั้นให้ผู้ใช้ตอบแบบสอบถามประเมินความพึงพอใจในการใช้งาน โดยใช้คำถามแบบเลือกตอบตามระดับความพึงพอใจ โดยแบ่งความพึงพอใจเป็น 5 ระดับ (Rating Scale) ดังนี้

5 หมายถึง มีระดับความพึงพอใจในระดับมากที่สุด

4 หมายถึง มีระดับความพึงพอใจในระดับมาก

3 หมายถึง มีระดับความพึงพอใจในระดับพอใช้

2 หมายถึง มีระดับความพึงพอใจในระดับน้อย

1 หมายถึง มีระดับความพึงพอใจในระดับน้อยที่สุด

เมื่อได้ผลการประเมินความพึงพอใจในแต่ละด้าน ผู้วิจัยได้นำผลมาคำนวณหาค่าทางสถิติด้วยโปรแกรมสำเร็จรูป เพื่อวิเคราะห์หาค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐาน และสรุปผลการประเมินความพึงพอใจ ดังตารางที่ 1 โดยมีเกณฑ์การวัดระดับความพึงพอใจในการใช้งาน

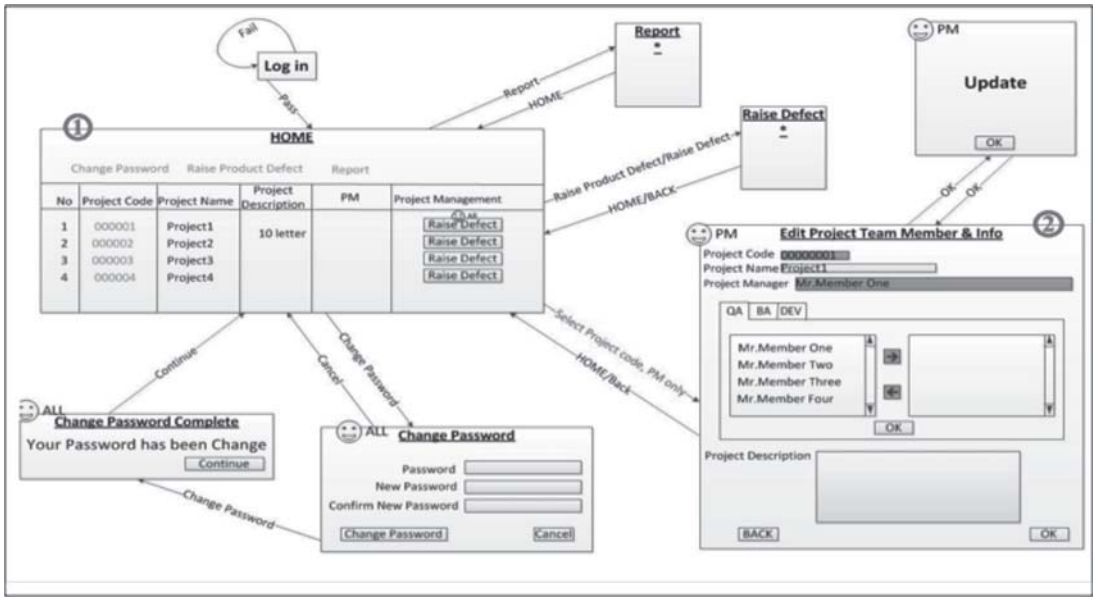
4.51-5.00 หมายถึง มีความพึงพอใจในระดับมากที่สุด

3.51-4.50 หมายถึง มีความพึงพอใจในระดับมาก

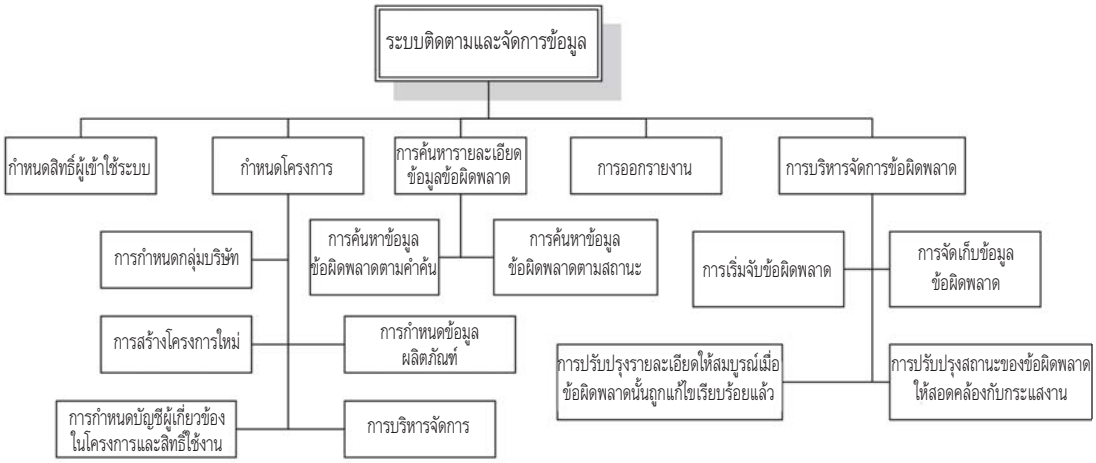
2.51-3.50 หมายถึง มีความพึงพอใจในระดับพอใช้

1.51-2.50 หมายถึง มีความพึงพอใจในระดับน้อย

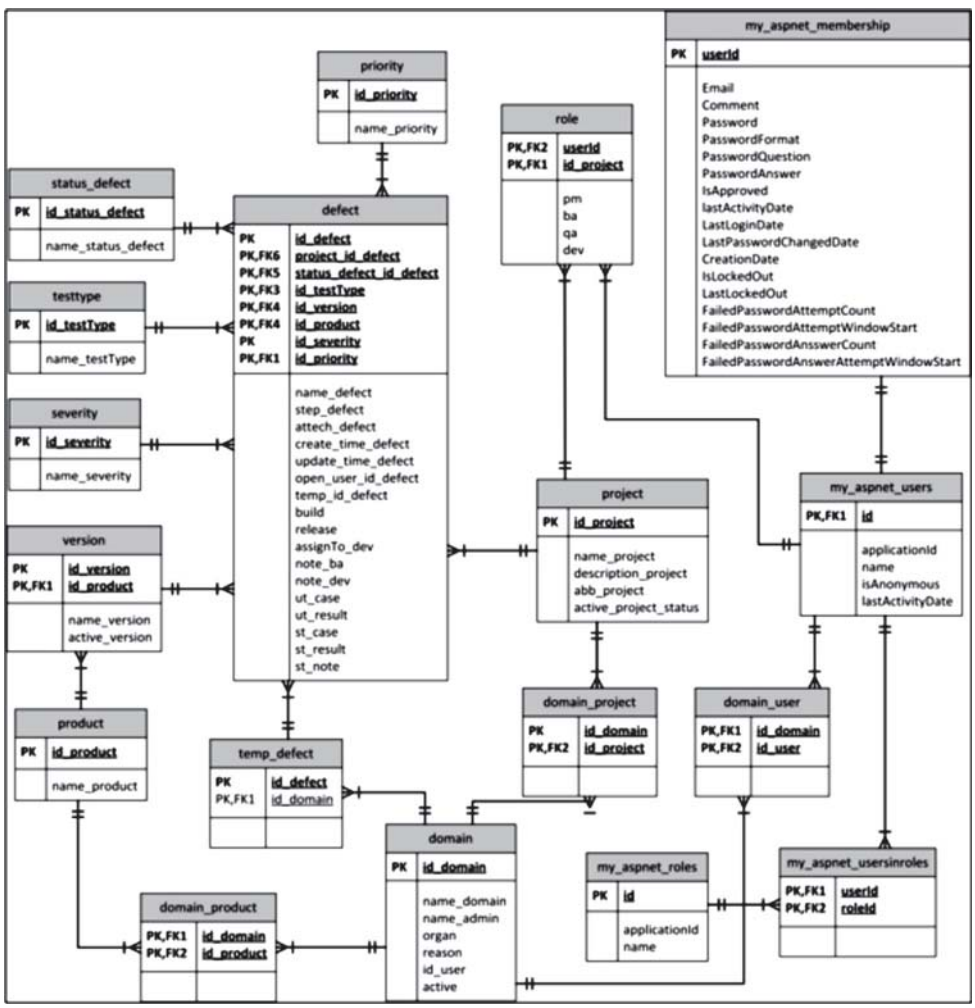
1.00-1.50 หมายถึง มีความพึงพอใจในระดับน้อยที่สุด



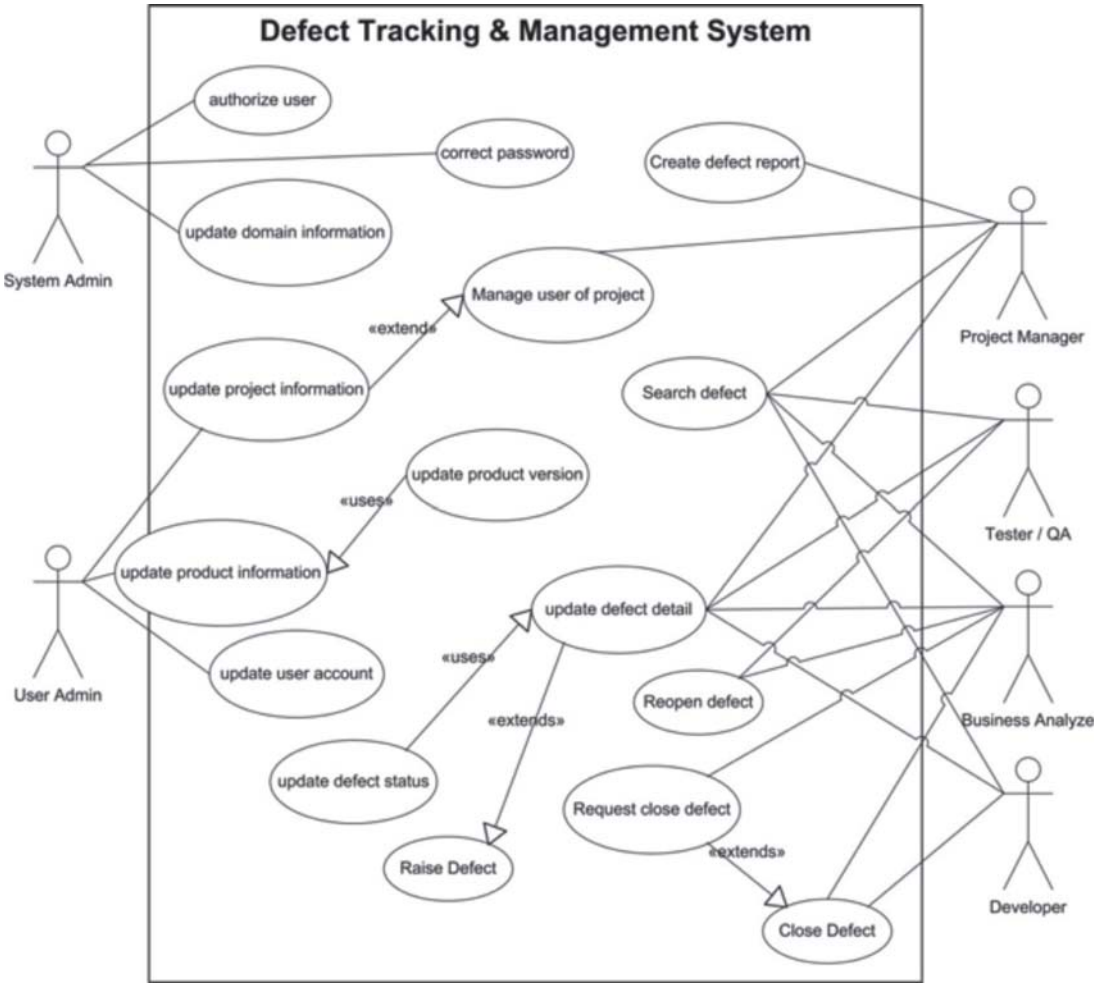
รูปที่ 3 การออกแบบแผนผังส่วนต่อประสานผู้ใช้ส่วนหนึ่งของระบบ



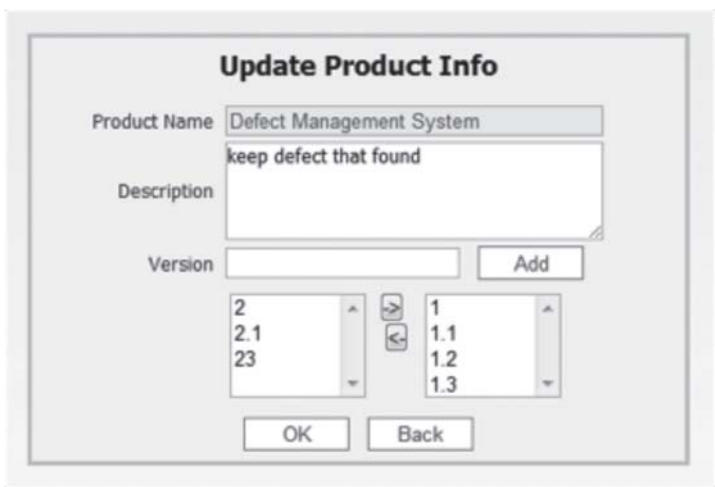
รูปที่ 4 แผนภาพการแตกโครงสร้างของงาน



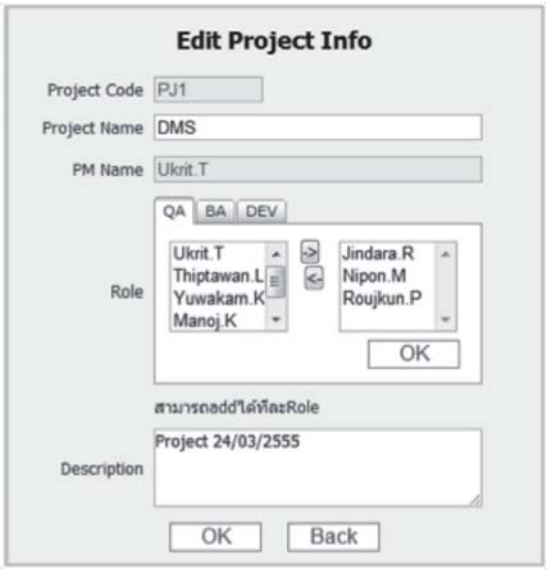
รูปที่ 5 แผนภาพความสัมพันธ์ระหว่างเอนทิตีของระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์



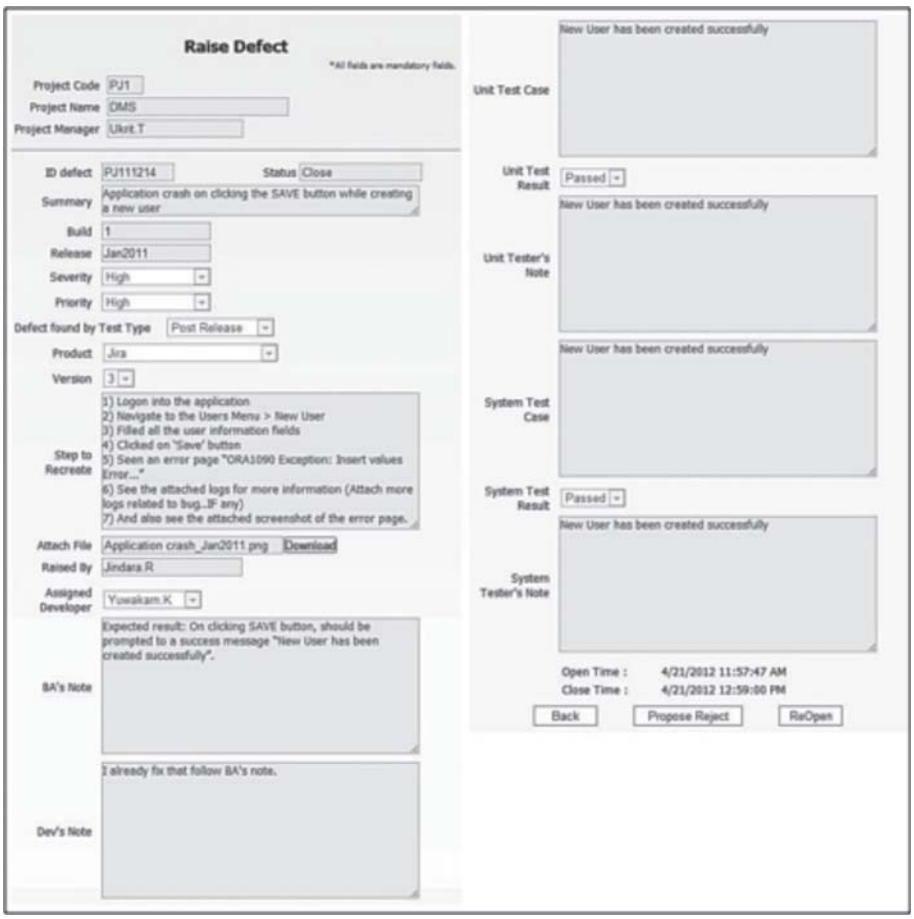
รูปที่ 6 แผนภาพยูสเคสโดยย่อของระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์



รูปที่ 7 ตัวอย่างจอภาพการจัดการ Version ของ Product



รูปที่ 8 ตัวอย่างจอภาพในการกำหนดทีม ผู้พัฒนาในแต่ละบทบาท



รูปที่ 9 ตัวอย่างจอภาพในการติดตามและจัดการข้อผิดพลาดของซอฟต์แวร์

Report

No.	Defect ID	Defect Name	Assigned To	Test Type	Defect Status	Priority	Severity	Product Name	Product Version	Project Name	Project Code	Project Status	Open Time	Close Time
1	JR111212	Submit button dissapper	Thiptawan.L	ST	Coding	High	High	Jira	4	Jira	JR1	Active	4/20/2012 5:55:24 AM	
2	PJ111207	Status not change follow step of defect.	Roujkun.P	ST	Close	High	High	Defect Management System	1.3	DMS	PJ1	Active	3/24/2555 8:14:27 AM	4/19/2012 6:58:14 AM
3	PJ111210	Submit button dissapper	Yuwakarn.K	PR	Close	High	Low	Defect Management System	1.1	DMS	PJ1	Active	4/11/2012 4:18:34 AM	4/11/2012 4:47:18 AM
4	PJ111211	Report Page not response		UT	Open	Low	High	Defect Management System	1.1	DMS	PJ1	Active	4/19/2012 7:25:50 AM	

รูปที่ 10 ตัวอย่างจอภาพในการแสดงผลรายละเอียดข้อผิดพลาดที่เกิดขึ้น

ตารางที่ 1 ผลการประเมินการยอมรับระบบติดตามและจัดการข้อผิดพลาดในการพัฒนาซอฟต์แวร์

รายการประเมิน	ค่าเฉลี่ย	ส่วนเบี่ยงเบนมาตรฐาน	ระดับความพึงพอใจ
ด้านสมรรถนะและความถูกต้องของระบบ	3.92	0.14	มาก
ด้านระยะเวลาในการเรียนรู้และจดจำคำสั่งในการใช้งาน	3.75	0.25	มาก
ด้านการนำไปใช้ประโยชน์	4.75	0.25	มากที่สุด
ด้านความพึงพอใจโดยรวมต่อระบบ	4.17	0.29	มาก
ค่าเฉลี่ยโดยรวม	**4.15**	**0.23**	**มาก**

จากตารางที่ 1 ผลการประเมินการยอมรับของผู้ใช้งานระบบติดตามและจัดการข้อผิดพลาดในการพัฒนาซอฟต์แวร์ทั้ง 4 ด้าน พบว่า ผู้ใช้งานมีความพึงพอใจในระดับมาก โดยทุกด้านมีระดับการกระจายที่ไม่แตกต่างกัน และผู้ใช้มีระดับความพึงพอใจในการนำไปใช้ประโยชน์อยู่ในระดับมากที่สุด

4. สรุป

4.1 สรุปและอภิปรายผล

4.1.1 ผู้วิจัยได้ศึกษาความต้องการของทีมพัฒนาซอฟต์แวร์ซึ่งต้องการระบบที่ไม่ซับซ้อน ฟังก์ชันงานมีเฉพาะในส่วนที่ต้องการใช้งาน ซึ่งสอดคล้องกับ Opperthauser, D. (2003) ที่กล่าวว่าระบบการจัดการข้อผิดพลาดต้องสามารถระบุคำอธิบายเกี่ยวกับข้อผิดพลาดที่เกิดขึ้นอย่างชัดเจน และแสดงขั้นตอนรายละเอียดของการเกิดข้อผิดพลาดนั้น พร้อมทั้งแสดงสภาพแวดล้อมในขณะที่พบข้อผิดพลาด นอกจากนี้ ต้องเก็บประวัติของข้อผิดพลาด ซึ่งประกอบด้วย ใครเป็นผู้พบข้อผิดพลาด และใครเป็นผู้รับผิดชอบแก้ไขข้อผิดพลาดถูกแก้ไขไปเมื่อใด และใครเป็นผู้ตรวจสอบการแก้ไขข้อผิดพลาดนั้น ข้อผิดพลาดนั้นเกิดขึ้นจากส่วนใด หรือเหตุใดข้อผิดพลาดนั้นจึงเกิดขึ้น ณ ตำแหน่งนี้ เป็นความผิดพลาดจากการเก็บความต้องการหรือไม่ การออกแบบ การเขียนโปรแกรม หรือความผิดพลาดของเครื่องมือ เพื่อที่จะได้ทราบและจัดการกับข้อผิดพลาดที่เกิด

ขึ้น และสามารถพัฒนากระบวนการให้ดีและมีประสิทธิภาพมากยิ่งขึ้น

4.1.2 การประเมินความพึงพอใจการใช้งานระบบสารสนเทศโดยทีมผู้พัฒนาซอฟต์แวร์ในบริษัทขนาดกลางอยู่ในระดับมาก เนื่องจากผู้ประเมินได้ทดลองใช้และพบว่า ระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์นั้นสามารถใช้ประโยชน์ได้ตรงตามความต้องการและมีประสิทธิภาพ ระบบเป็นศูนย์กลางในการรวบรวมข้อมูลข้อผิดพลาดที่เกิดขึ้น สำหรับการบันทึกข้อมูลเกี่ยวกับปัญหาข้อผิดพลาดที่เกิดขึ้นในการพัฒนาซอฟต์แวร์นั้นทำให้สามารถวิเคราะห์ข้อมูลได้ง่าย เพื่อที่จะได้ทราบแหล่งปัญหาอื่น ๆ ที่อาจทำให้เกิดการทำงานซ้ำซ้อน ซึ่งทำให้ต้นทุนการผลิตซอฟต์แวร์สูงขึ้นอีกด้วย (Boehm and Basili, 2001: 135-137) และทำให้การสื่อสารระหว่างทีมผู้พัฒนาในการจัดการข้อผิดพลาดเป็นไปได้ง่ายมากยิ่งขึ้น นอกจากนี้ ทำให้กระบวนการติดตามและจัดการข้อผิดพลาดมีประสิทธิภาพในการดำเนินงาน ช่วยให้ลดระยะเวลาในการจัดการข้อผิดพลาดได้มากกว่า 15% โดยการเปรียบเทียบระยะเวลาตั้งแต่พบข้อผิดพลาดจนกระทั่งจัดการข้อผิดพลาดนั้นเสร็จสิ้นของโครงการที่ไม่ได้ใช้งานจำนวน 2 โครงการ และใช้งานระบบการจัดการข้อผิดพลาดนี้อีกจำนวน 2 โครงการที่มีขนาดและระยะเวลาในการพัฒนา 4-8 เดือนในแต่ละโครงการ

4.1.3 จากการดำเนินการพัฒนาระบบติดตามและจัดการข้อผิดพลาดซอฟต์แวร์ พบว่า จะมีปัญหาของความต้องการที่เพิ่มมากยิ่งขึ้น เนื่องจากทีมผู้พัฒนาซอฟต์แวร์มีความต้องการที่หลากหลายและจากการศึกษาวิจัย พบว่า มีคำแนะนำและวิธีการในการติดตามและจัดการข้อผิดพลาดซอฟต์แวร์ที่หลากหลายเช่นกัน จึงทำให้ขอบเขต (Scope) ของการพัฒนาระบบจะมีขนาดที่แตกต่างกัน (Shaffiei, Z.A., 2010) ดังนั้น ผู้วิจัยจึงเลือกพัฒนาโดยเน้นการใช้งานที่ง่ายไม่ซับซ้อน และสามารถนำไปใช้ให้เกิดประโยชน์ได้ก่อนในเบื้องต้น ถึงแม้ว่าคะแนนความพึงพอใจในการใช้งานระบบในด้านของระยะเวลาการเรียนและจดจำซึ่งทีมผู้ประเมินให้ความพึงพอใจอยู่ในระดับมาก แต่คะแนนค่าเฉลี่ยมีค่าเพียง 3.75 ดังนั้น สำหรับผู้วิจัยแล้วนอกจากคำนึงถึงเรื่องของประโยชน์การใช้งานแล้วยังต้องคำนึงถึงการออกแบบการใช้งานให้ง่ายมากยิ่งขึ้นอีกด้วย

4.2 ข้อเสนอแนะ

ข้อเสนอแนะในการนำผลการวิจัยไปใช้งานมีดังนี้

4.2.1 ควรมีการทดสอบกับกลุ่มพัฒนาที่มีขนาดใหญ่ เพื่อนำผลการนำระบบไปใช้ประโยชน์มาเปรียบเทียบความแตกต่างของระยะเวลาที่ลดลงในการติดตามข้อผิดพลาดควรมีเอกสารคู่มือการใช้งานออนไลน์เพื่อช่วยให้ผู้ใช้เรียนรูปการใช้งานระบบได้ง่ายมากยิ่งขึ้น (Shaffiei, Z.A., 2010)

4.2.2 ควรมีการพัฒนาในส่วนของการวิเคราะห์ข้อผิดพลาดที่เกิดขึ้นในรูปแบบของกราฟเพื่อนำข้อมูลมาพัฒนากระบวนการติดตามและจัดการข้อพลาดให้มีประสิทธิภาพมากยิ่งขึ้น

4.2.3 ควรมีการพัฒนาฟังก์ชันงานเพิ่มเติมเพื่อขยายขอบเขตให้ครอบคลุมการติดตามและจัดการข้อผิดพลาดตั้งแต่การสร้างและกำหนดกรณีทดสอบ (Test Case) ลงในระบบ โดยกำหนดชื่อผู้ที่รับผิดชอบในการพัฒนา และผู้ดำเนินการทดสอบให้ชัดเจน และหากเมื่อพบข้อผิดพลาดที่

เกิดขึ้นระหว่างการทดสอบระบบ ผู้ทดสอบสามารถส่งต่อรายละเอียดข้อผิดพลาดดังกล่าวไปยังผู้พัฒนาได้ทันที ซึ่งจะช่วยให้กระบวนการทดสอบและติดตามข้อผิดพลาดของระบบมีประสิทธิภาพมากยิ่งขึ้น

5. กิตติกรรมประกาศ

ขอขอบพระคุณมหาวิทยาลัยกรุงเทพ และคณะวิทยาศาสตร์และเทคโนโลยี ที่ให้การส่งเสริมการทำวิจัยในครั้งนี้จนสำเร็จลุล่วง

6. เอกสารอ้างอิง

Avram, G., Sheehan, A., and Sullivan, D.K. 2007. **Defect Tracking Systems in Global Software Development – a work practice study.** The Challenges of Collaborative Work in Global Software Development Workshop 2007, Ireland.

Boehe, B. and Basili, V.R. 2001. **Software Defect Reduction Top 10 List.** Software Management, USA: 135-137.

Barnson, M.P. 2001. **The Bugzilla Guide.** Retrieved February 12, 2012 from http://db.glugboom.org/Documentation/Bugzillar-Guide

Jalbert, N. 2008. **Automated Duplicate Detection for Bug Tracking Systems.** Dependable Systems and Networks with FTCS and DCC, IEEE International Conference 2008: 52-61.

Lethbridge, T.C., J. Singer, et al. 2003. How Software Engineering use Documentation: The State of The Practice. **IEEE Software.** Volume 20 (Issue 6).

Opperthauser, D. 2003. Defect Management in an Agile Development Environment. **Crosstalk the Journal of Defense Software Engineering.** www.stsc.hill.af.mil

Shaffiei, Z.A., Mokhsin, M., and Hamidi, S.R. 2010. Change and Bug Tracking System: Anjung Penchala Sdn. Bhd. **International Journal of Computer Applications,** Volume 10, No. 3, 2010: 28-34.

Singh, L., 2004. **Advanced Verification Techniques: A System C Based Approach for Successful Tapeout.** English: Springer.

Smart, J.F. 2007. **Javaworld.com: What issue tracking system is best for you?** Retrieved September 14, 2012 from http://www.javaworld.com/javaworld/jw-03-2007/jw-03-bugs.html

Time, M., and Marcus, A. 2008. **Automated Severity Assessment of Software Defect Reports.** Software Maintenance, 2008. ICSM2008: 346-355.

เพ็ญจิรา คันธวงศ์. 2010. แผนการตลาดของผู้ผลิตซอฟต์แวร์สำหรับองค์กรปี 2553 (Marketing Plan of Business Software Company in 2010). **Executive Journal,** Vol. 30, No. 2, April-June 2010: 71-76.