การพัฒนาระบบวางแผนการทดสอบซอฟต์แวร์ตามวงจรการทดสอบซอฟต์แวร์และมาตรฐาน ISO 29119, IEEE 829

The Development of Planning Testing System Follow by Software Testing Life Cycle and ISO 29119, IEEE 829 Standard

ชฎาธาร รักสวัสดิ์ (Chadatarn Raksawat)[1] ปัทมา เจริญพร (Pattama Charoenporn)[2]

บทคัดย่อ

ในกระบวนการพัฒนาซอฟต์แวร์ควรมีการทดสอบซอฟต์แวร์ เพื่อตรวจสอบและปรับปรุงให้ซอฟต์แวร์มีความถูกต้องสมบูรณ์ คุณภาพดี และสามารถระบุ ค้นหาหรือแก้ไขข้อผิดพลาดของซอฟต์แวร์ได้ การทดสอบซอฟต์แวร์ให้มีประสิทธิภาพถูกต้อง และเชื่อถือได้ จำเป็นต้องอ้างอิงมาตรฐานในการทดสอบซอฟต์แวร์ โดยศึกษาและทำความเข้าใจในมาตรฐานการทดสอบให้ครบถ้วนและละเอียด รอบคอบ ซึ่งเป็นสิ่งที่ยุ่งยาก เสียเวลา และยังเพิ่มต้นทุนในการพัฒนาซอฟต์แวร์ที่มีภาระงานด้านการทดสอบซอฟต์แวร์เพิ่ม โดยงานวิจัยนี้จัดทำเพื่อใช้ในกระบวนการทดสอบซอฟต์แวร์ ซึ่งเริ่มพัฒนาระบบโดยหาความสัมพันธ์ระหว่างวงจรการทดสอบซอฟต์แวร์ และมาตรฐานการทดสอบซอฟต์แวร์ทั้ง ISO 29119 และ IEEE 829 ที่เป็นมาตรฐานด้านเอกสารการทดสอบ เพื่อศึกษาความเกี่ยวข้อง สอดคล้อง ระหว่างวงจรการทดสอบและเอกสารการทดสอบแล้วนำมาสร้างฟังก์ชันในระบบวางแผนการทดสอบซอฟต์แวร์ให้สามารถปรับใช้ได้ในซอฟต์แวร์ทุกประเภท และมีการวัดประสิทธิภาพทั้งในด้านข้อกำหนดเชิงบรรทัดฐานและประสิทธิภาพในการทดสอบซอฟต์แวร์ เพื่อควบคุม ตรวจสอบ และปรับปรุงคุณภาพของซอฟต์แวร์ โดยระบบช่วยลดต้นทุนและความยุ่งยากในการทดสอบซอฟต์แวร์ ฟังก์ชันภายในงานวิจัยประกอบด้วย test plan, test procedure specification, test data requirement, test environment requirement, test status, test result, measurement software performance และ documentation ซึ่งเอกสารการทดสอบที่ผู้ใช้งานจะได้รับมีความถูกต้อง ครอบคลุม ครบถ้วนทุกขั้นตอนในการทดสอบ และมีเนื้อหาในเอกสารการทดสอบที่เป็นไปตามมาตรฐานสากล

คำสำคัญ การทดสอบซอฟต์แวร์ การพัฒนาซอฟต์แวร์ ประสิทธิภาพของซอฟต์แวร์ มาตรฐานสากลในการทดสอบซอฟต์แวร์ เอกสารการทดสอบ

[1]นักศึกษาหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง e-mail address : 60605076@kmitl.ac.th

[2] ดร. ประจำสาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง e-mail address : pattama.ch@kmitl.ac.th

Abstract

In software development process there should be software testing to inspection and improve the software to be correctness, completely, good quality and able to identify, finding or fix software errors. Software testing to be efficient, correctness and reliability that are necessary to refer to the software testing standard. By studying and understanding the testing standards to completely and thoroughly which is complicated, time-consuming, and increases the cost of software development with additional software testing workloads. In this research, develops for use in the software testing process. That developed by finding a relationship between software testing life cycle and software testing standards in ISO 29119 and IEEE 829 that are software test documentation standard. To study the related between the testing life cycle and the test documentation and then use to create functions in the software testing planning system that can be used in all types of software and measure the performance of a normative requirement and software testing performance to control, verify and improve software quality. Including the system able to reduce the cost and difficulty of software testing. So, function in the research consists of a test plan, test procedure specification, test data requirement, test environment requirement, test status, test result, measurement software performance and documentation. The user will receive the test documents that are correctness, cover and completeness in every step of the test, and the content of the test document that meets international standards.

Keywords: Software testing, Software development, Software performance, Software testing standard, Software documentation

บทนำ

ในปัจจุบันนี้เทคโนโลยีมีการพัฒนาอย่างก้าวหน้าและรวดเร็ว เมื่อโลกในปัจจุบันขับเคลื่อนด้วยเทคโนโลยี (25 ก.ย. 2561) จึงเรียกว่า ยุค Digital 4.0 หรือ “ดิจิทัล 4.0” หรือ “ดิจิทัลไทยแลนด์” เป็นวลีที่คนไทยเริ่มจะได้ยินบ่อยขึ้นในช่วงหลายปีมานี้ (สำนักงานปลัดกระทรวงศึกษาธิการ, 2561) ซึ่งเทคโนโลยีเป็นที่สนใจของคนทุกมุมโลก ทุกเพศ ทุกวัย ทุกสาขาอาชีพ อย่างแพร่หลายและมีการนำมาใช้ในการทำงาน ชีวิตประจำวัน และการเรียนการศึกษา ส่งผลให้ในปัจจุบันนี้มีหลักสูตรที่เกี่ยวกับเทคโนโลยีเพิ่มเข้าไปด้วย เพื่อให้เด็ก ๆ รู้เท่าทันถึงภัยที่มาพร้อมกับโลกยุคดิจิทัล ซึ่งเทคโนโลยีและโลกดิจิทัลเปลี่ยนไปอย่างรวดเร็ว และเคลื่อนที่ไม่มีหยุด องค์กรธุรกิจต่าง ๆ จึงจำเป็นต้องปรับตัวให้ทันตามเทรนด์ พัฒนานวัตกรรมเพื่อต่อยอดธุรกิจบนการแข่งขันที่รวดเร็วและรอบด้านจาก SME (Small and Medium Enterprises) ที่แปลว่า วิสาหกิจขนาดกลางและขนาดย่อมให้กลายเป็น Smart Enterprise ที่มีศักยภาพสูงขึ้น จากบริการธรรมดาให้กลายเป็น High Value Service (P. Eira, R. J. Machado and A. Silva, 2018) เพื่อความมั่นคง มั่งคั่ง และยั่งยืนของธุรกิจ (สำนักงานปลัดกระทรวงศึกษาธิการ, 2561) โดยหลายธุรกิจมีการนำซอฟต์แวร์มาปรับใช้ในการทำงานมากขึ้นเพื่อเพิ่มประสิทธิภาพการทำงาน ลดต้นทุน ลดระยะเวลา ควบคุมคุณภาพของงาน เพิ่มโอกาสทางการตลาดและโอกาสในการทำงาน ส่งผลให้ในปัจจุบันนี้มีซอฟต์แวร์ออกสู่ท้องตลาดเพิ่มขึ้นมากในรูปแบบที่หลากหลายเพื่อให้ตอบสนองต่อกลุ่มธุรกิจที่มีหลายประเภท โดยส่วนใหญ่ที่ออกสู่ท้องตลาดจะอยู่ในรูปแบบซอฟต์แวร์ประยุกต์ที่ประกอบด้วยซอฟต์แวร์สำเร็จและซอฟต์แวร์ใช้งานเฉพาะ ซึ่งซอฟต์แวร์สำเร็จ (software package) เป็นซอฟต์แวร์ที่มีความนิยมใช้กันสูงมาก ดังนั้นบริษัทที่จะผลิตซอฟต์แวร์ขายจะต้องมีคุณสมบัติที่เป็นจุดเด่นเฉพาะตัว เพื่อดึงดูดลูกค้าและต้องสร้างความน่าเชื่อถือให้แก่ลูกค้า สร้างความมั่นใจให้แก่ลูกค้าเมื่อใช้ซอฟต์แวร์ของทางบริษัทว่าจะไม่ส่งผลให้ธุรกิจของลูกค้าเกิดความเสียหายเป็นอันขาด ดังนั้นการทดสอบซอฟต์แวร์ก่อนที่จะทำการส่งมอบให้ลูกค้าถือเป็นสิ่งที่สำคัญ หากขาดการทดสอบซอฟต์แวร์ก่อนส่งมอบให้ลูกค้าทำให้เกิด ค่าใช้จ่ายในการแก้ไขซอฟต์แวร์สูง ซอฟต์แวร์มีคุณภาพต่ำ ส่งผลต่อซอฟต์แวร์ทั้งระบบทำให้เกิด failure ระบบไม่สามารถใช้งานได้ ส่งผลให้ลูกค้าขาดความเชื่อมั่นในซอฟต์แวร์ของบริษัท

การทดสอบซอฟต์แวร์จะต้องทำการศึกษาค้นคว้าหาข้อมูลก่อนที่จะทดสอบซอฟต์แวร์ทั้งขั้นตอนในการทดสอบซอฟต์แวร์ ความเหมาะสมที่จะนำมาปรับใช้กับซอฟต์แวร์ที่ได้พัฒนาขึ้น หรือแผนการทดสอบซอฟต์แวร์ที่บริษัทมีอยู่แล้ว ด้วยซอฟต์แวร์มีหลายประเภทและหลายขนาด อีกทั้งยังมีคุณลักษณะความต้องการเฉพาะตัว การที่จะหาข้อสรุปมาวัดว่าการทดสอบที่ได้ทำอยู่นั้นดีหรือไม่นั้นทำได้ยาก การนำมาตรฐานการทดสอบมาใช้ในการทดสอบซอฟต์แวร์เป็นสิ่งที่ยุ่งยาก เสียเวลา แต่เพื่อป้องกันข้อผิดพลาดที่จะเกิดขึ้นในกระบวนการทดสอบ การทดสอบที่ครอบคลุมครบถ้วน สามารถนำไปใช้งานได้จริง (E. Veenendaal, 2016) อีกทั้งยังสามารถสร้างความน่าเชื่อถือให้แก่ผู้ใช้งานระบบ ความสำคัญของการทดสอบตามมาตรฐานสากลจริงเป็นสิ่งที่ควรทำ (The Home of International Standards for Software Testing, 2018)

ทางผู้วิจัยเห็นถึงความจำเป็นพร้อมทั้งหาแนวทางในการแก้ไขปัญหาข้างต้น จึงได้พัฒนาระบบที่ช่วยในการทดสอบซอฟต์แวร์ โดยเริ่มจากการหาความสัมพันธ์ระหว่างวงจรการทดสอบและมาตรฐานในการทดสอบ ทางผู้วิจัยได้อ้างอิงมาตรฐาน ISO 29119 และมาตรฐาน IEEE 829 ที่เป็นมาตรฐานสากลในการทดสอบซอฟต์แวร์ (A. Rana, A. Singh Rawat and A. Bijalwan, 2017) ซึ่งเป็นที่แน่นอนว่ามาตรฐานที่ได้ประกาศใช้นั้นมีการตรวจสอบความถูกต้องทั้งกระบวนการทำงานและความเหมาะสมที่สามารถนำมาปรับใช้กับงานในปัจจุบันนี้ (ISO/IEC/IEEE 29119-2, 2013), (ISO/IEC/IEEE 29119-3, 2013) การที่นำมาตรฐานการทดสอบ ISO 29119 และ IEEE 829 หาความสัมพันธ์เพื่อต้องการให้ระบบที่สร้างขึ้นมา

ครอบคลุมทุกขั้นตอนในการดำเนินงาน ครอบคลุมเอกสารที่ในการทดสอบ อีกทั้งนำไปจับคู่กับวงจรการพัฒนาซอฟต์แวร์ที่เป็นขั้นตอนพื้นฐานในการทดสอบเพื่อให้สามารถนำมาปรับใช้ในซอฟต์แวร์หลายประเภท ส่งผลให้ระบบสามารถสร้างเอกสารการทดสอบที่ครอบคลุมทั้งวงจรการทดสอบซอฟต์แวร์ที่เป็นไปตามมาตรฐานสากลและสามารถนำไปใช้งานได้จริง ซึ่งภายในระบบนี้ได้มีการวัดประสิทธิภาพของซอฟต์แวร์ที่ทำการทดสอบ โดยมีการวัดประสิทธิภาพ 2 ประเภท คือด้านข้อกำหนดเชิงบรรทัดฐานและด้านความสามารถในการทดสอบซอฟต์แวร์ เพื่อให้ทราบถึงคุณภาพ ประสิทธิภาพ และความสมบูรณ์ของซอฟต์แวร์ก่อนที่จะส่งมอบให้ลูกค้าหรือนำไปใช้งานจริง

วัตถุประสงค์

1. เพื่อตรวจสอบ และปรับปรุงให้ซอฟต์แวร์มีความถูกต้อง สมบูรณ์ คุณภาพที่ดี
2. เพื่อวัดประสิทธิภาพของซอฟต์แวร์ที่นำมาทดสอบ
3. เพื่อสามารถสร้างและนำเอกสารการทดสอบซอฟต์แวร์ไปใช้งาน โดยอ้างอิงตามมาตรฐานการทดสอบซอฟต์แวร์

ระเบียบวิธีวิจัย

ระบบวางแผนการทดสอบซอฟต์แวร์ ผู้วิจัยได้ทำการหาความสัมพันธ์ระหว่างวงจรการทดสอบซอฟต์แวร์กับมาตรฐานการทดสอบซอฟต์แวร์ทั้ง ISO 29119 และ IEEE 829 เพื่อสร้างระบบและฟังก์ชันการทำงานให้ครอบคลุมทุกขั้นตอนของการทดสอบและให้ได้เอกสารประกอบการทดสอบซอฟต์แวร์ที่ครบถ้วนสมบูรณ์ตามขั้นตอนการทดสอบและมาตรฐานในการทดสอบซอฟต์แวร์ (N. Chen, Ethan W. Chen and Ian S. Chen, 2018) ซึ่งความสัมพันธ์ระหว่างขั้นตอนในการทดสอบซอฟต์แวร์และมาตรฐานในการทดสอบซอฟต์แวร์แสดงดังภาพที่ 1 โดยผู้วิจัยได้เริ่มจากศึกษาวงจรในการทดสอบซอฟต์แวร์ 6 ขั้นตอน คือ (1) กำหนดความต้องการในการทดสอบ (2) วางแผนในการทดสอบ (3) กำหนดกรณีทดสอบต่าง ๆ ที่ต้องการทดสอบ (4) ตั้งค่าสภาพแวดล้อมที่ต้องการทดสอบ (5) ทดสอบการทำงานของซอฟต์แวร์ และ (6) รายงานผลการทดสอบซอฟต์แวร์ (U. Kumar Tiwari and S. Kumar, 2017) จากนั้นศึกษามาตรฐานในการทดสอบซอฟต์แวร์และหาความสัมพันธ์กับขั้นตอนการทดสอบซอฟต์แวร์ที่มีอยู่โดยทั่วไป เพื่อให้ได้กระบวนการในการทดสอบซอฟต์แวร์เป็นไปตามมาตรฐานสากล ครอบคลุมทุกขั้นตอนของการทดสอบและสามารถนำไปใช้งานได้จริง ซึ่งในงานวิจัยนี้อ้างอิงมาตรฐานสากลในการทดสอบซอฟต์แวร์ คือ ISO 29119 ประกอบด้วย 5 ส่วนดังนี้ (1) แนวความคิดและการนิยาม (2) กระบวนการทดสอบ (3) เอกสารการทดสอบ (4) เทคนิคในการทดสอบ และ (5) การทดสอบด้วยคำหลัก (N. Chen, Ethan W. Chen and Ian S. Chen, 2018) และ มาตรฐาน IEEE 829 ประกอบด้วย 9 ส่วน ดังนี้ (1) master test plan (2) level test plan (3) level test design (4) level test case (5) level test procedure (6) level test log (7) anomaly report และ (8) test report โดยที่ในงานวิจัยจะเริ่มจาก ISO 29119-2 คือ ส่วนกระบวนการทดสอบ เป็นส่วนหลักของมาตรฐานที่ใช้ในการกำหนดขั้นตอนของการทดสอบซอฟต์แวร์ แบ่งออกเป็น 3 ส่วน คือ (1) กระบวนการขององค์กร (2) กระบวนการจัดการทดสอบ และ (3) กระบวนการทดสอบแบบไดนามิก (ISO/IEC/IEEE 29119-2, 2013) และต่อมาศึกษามาตรฐาน ISO 29119-3 และ IEEE 829 ที่เป็นส่วนมาตรฐานด้านเอกสารการทดสอบซอฟต์แวร์มีการทำงานซึ่งแบ่งออกเป็น 3 ส่วน เช่นเดียวกันกับมาตรฐาน ISO 29119-2 (ISO/IEC/IEEE 29119-3, 2013) เอกสารการทดสอบประกอบด้วย (1) แผนการทดสอบ (2) ข้อกำหนดการทดสอบ (3) กำหนดความต้องการข้อมูลในการทดสอบ (4) กำหนดความต้องการสภาพแวดล้อมในการทดสอบ (5) รายงานสถานะการทดสอบ (6) เปรียบเทียบพร้อมวัดประสิทธิภาพการทดสอบ (7) รายงาน

เหตุการณ์ และ (8) เอกสารประกอบการทดสอบทั้งหมดที่พร้อมในการพิมพ์ ซึ่งโปรแกรมที่ใช้ในการพัฒนาระบบวางแผนการทดสอบซอฟต์แวร์ คือ โปรแกรม visual studio พัฒนาโดยภาษา java ในรูปแบบ windows application

จากการหาความสัมพันธ์ระหว่างมาตรฐาน ISO 29119-2, ISO 29119-3, IEEE 829 และ วงจรการทดสอบในข้างต้น สามารถสร้างฟังก์ชันสำหรับการเตรียมเอกสารการทดสอบให้ครอบคลุมทั้งหมด ซึ่งในระบบประกอบด้วยฟังก์ชัน test plan, test procedure specification, test data requirement, test environment requirement, test status, test result, measurement software performance and documentation โดยสามารถแบ่งความสัมพันธ์ออกได้เป็น 4 ส่วน ดังนี้

ส่วนที่ 1 : test management documentation ในส่วนนี้เกี่ยวข้องกับการสร้างแผนการทดสอบซอฟต์แวร์ คือ ฟังก์ชัน test plan

ส่วนที่ 2 : dynamic test documentation เกี่ยวข้องกับกระบวนการจัดการทดสอบซอฟต์แวร์ ที่ประกอบด้วย 5 ฟังก์ชันดังนี้ (1) ฟังก์ชัน test procedure specification คือ ข้อกำหนดการทดสอบ (2) ฟังก์ชัน test data requirement คือ กำหนดความต้องการของข้อมูลในการทดสอบ (3) ฟังก์ชัน test environment requirement คือ กำหนดความต้องการของสภาพแวดล้อมที่ใช้ในการทดสอบ (4) ฟังก์ชัน test status คือ รายงานสถานการณ์ทดสอบ (5) ฟังก์ชัน test result คือ เปรียบเทียบผลการทดสอบ

ส่วนที่ 3 : measurement เป็นขั้นตอนในการวัดประสิทธิภาพซอฟต์แวร์ ในระบบแบ่งการวัดประสิทธิภาพออกเป็น 2 ด้าน คือ ข้อกำหนดเชิงบรรทัดฐาน และการวัดประสิทธิภาพในการทดสอบซอฟต์แวร์

3.1 การวัดประสิทธิภาพในด้านข้อกำหนดเชิงบรรทัดฐาน จะทำการศึกษาความสัมพันธ์ระหว่างข้อกำหนดเชิงบรรทัดฐานใน ISO 29119-2 กับรายการข้อมูลของ ISO 29119-3 แบ่งส่วนในการวัดประสิทธิภาพออกเป็น 3 ส่วน คือ (1) กระบวนการขององค์กร (2) กระบวนการจัดการทดสอบ (3) กระบวนการทดสอบแบบไดนามิก ซึ่งการวัดประสิทธิภาพข้อกำหนดเชิงบรรทัดฐานจะดำเนินในฟังก์ชัน test status ไปยัง Measurement โดยคลิกปุ่ม Normative requirement โดยมีค่าระดับความจำเป็นของข้อมูล 3 ระดับ คือ may, should, shall งานวิจัยได้กำหนดค่าดังนี้

may มีค่าความจำเป็นของข้อมูลอยู่ที่ร้อยละ 40

should มีค่าความจำเป็นของข้อมูลอยู่ที่ร้อยละ 60

shall มีค่าความจำเป็นของข้อมูลอยู่ที่ร้อยละ 80

3.2 การวัดประสิทธิภาพในการทดสอบซอฟต์แวร์ จะทำการวัดทั้งหมด 5 ข้อ คือ (1) จำนวนที่ทดสอบตามกรณีทดสอบที่กำหนดไว้ (number of executed test cases) (2) จำนวนที่นำกรณีทดสอบมาทดสอบซ้ำ (number of re-executed test cases) (3) จำนวนเหตุการณ์ที่เกิดขึ้น (number of incidents per category) (4) จำนวนปัญหาที่ได้รับการแก้ไขอย่างสมบูรณ์ (number of solved incidents per category) และ (5) จำนวนเวลา (ชั่วโมง) ที่ใช้ในการทดสอบ (number of hours spent) โดยการวัดประสิทธิภาพจะดำเนินการในฟังก์ชัน test result โดยคลิกปุ่ม Select Metric ซึ่งสูตรที่ใช้ในการคำนวณประสิทธิภาพแสดงดังนี้

จำนวนที่ทดสอบตามกรณีทดสอบที่กำหนดไว้ = (จำนวนกรณีทดสอบที่ถูกทดสอบ/จำนวนกรณีทดสอบทั้งหมด)

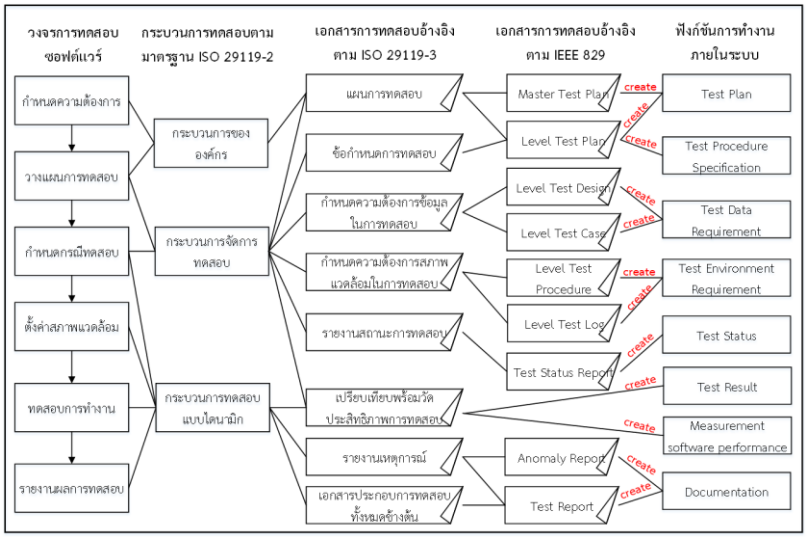
จำนวนที่นำกรณีทดสอบมาทดสอบซ้ำ = (จำนวนกรณีทดสอบที่มีจำนวนครั้งในการทดสอบมากกว่า 1 ครั้ง/จำนวนกรณีทดสอบทั้งหมด)

จำนวนเหตุการณ์ที่เกิดขึ้น = (จำนวนเหตุการณ์ที่พบในกรณีทดสอบ/จำนวนกรณีทดสอบที่ถูกทดสอบ)

จำนวนปัญหาที่ได้รับการแก้ไขอย่างสมบูรณ์ = (จำนวนกรณีทดสอบที่แก้ไข/จำนวนกรณีทดสอบที่พบปัญหา)

จำนวนเวลา (ชั่วโมง) ที่ใช้ในการทดสอบ = (เวลาที่ใช้ในการทดสอบ/จำนวนกรณีทดสอบที่ถูกทดสอบ)

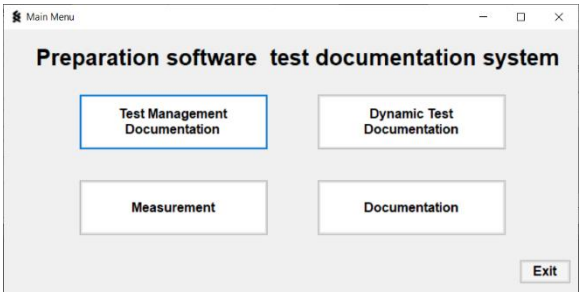
ส่วนที่ 4 : documentation เป็นขั้นตอนสุดท้ายของระบบที่เกี่ยวกับเอกสารการทดสอบซอฟต์แวร์ ซึ่งประกอบด้วยเอกสารที่รายงานเหตุการณ์ที่เกิดขึ้นระหว่างการทดสอบ คือ ฟังก์ชัน test incident report และฟังก์ชัน documentation ที่ใช้ในการพิมพ์เอกสารการทดสอบ



ภาพที่ 1 แผนภาพความสัมพันธ์ระหว่างวงจรการทดสอบซอฟต์แวร์ มาตรฐานการทดสอบ และฟังก์ชันการทำงานในระบบ

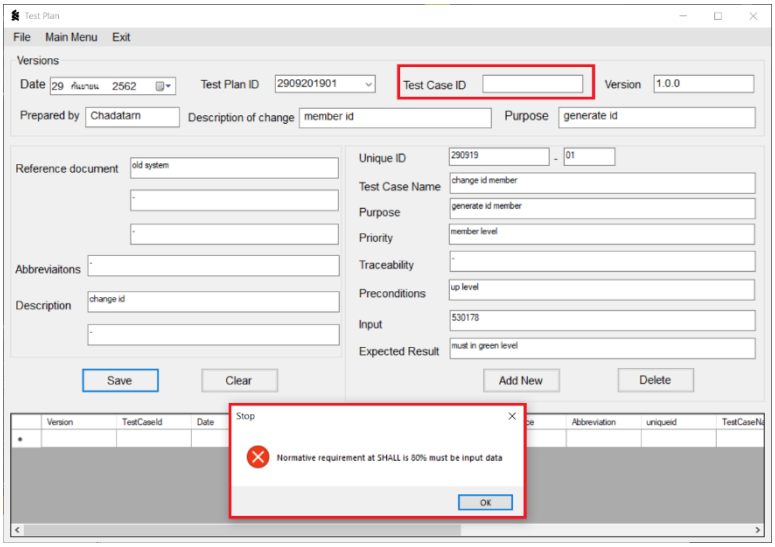
ผลการวิจัย

ระบบวางแผนการทดสอบซอฟต์แวร์สามารถตรวจสอบ และปรับปรุงให้ซอฟต์แวร์มีความถูกต้อง สมบูรณ์ คุณภาพดี สามารถวัดประสิทธิภาพของซอฟต์แวร์ได้ โดยแบ่งออกเป็น 2 ส่วน โดย ส่วนแรกวัดประสิทธิภาพในด้านข้อกำหนดเชิงบรรทัดฐาน โดยดำเนินการในรูปของฟังก์ชัน Normative Requirement ซึ่งเป็นส่วนที่บอกระดับความจำเป็นของข้อมูลซึ่งจะแจ้งเตือนในทุกฟังก์ชัน เพื่อเป็นการเตือนผู้ใช้งานให้ทราบถึงระดับความจำเป็นของข้อมูล ส่วนที่สองวัดประสิทธิภาพในการทดสอบซอฟต์แวร์ โดยดำเนินการในรูปของฟังก์ชัน Measurement เพื่อเป็นการบอกถึงคุณภาพและประสิทธิภาพของซอฟต์แวร์ ซึ่งจะแสดงผลให้ผู้ใช้งานทราบถึงส่วนที่ต้องปรับปรุงแก้ไข และตรวจสอบความถูกต้องสมบูรณ์ และสามารถสร้างและนำเอกสารการทดสอบไปใช้ได้จริง โดยแสดงในรูปฟังก์ชัน documentation โดยจากระเบียบวิธีวิจัยแบ่งระบบออกเป็น 4 ส่วน คือ test management documentation, dynamic test documentation, measurement และ documentation แสดงดังภาพที่ 2 ซึ่งเป็นหน้าแรกของระบบวางแผนการทดสอบซอฟต์แวร์ ซึ่งภายในระบบวางแผนการทดสอบซอฟต์แวร์ประกอบด้วยฟังก์ชันดังนี้



ภาพที่ 2 หน้าหลักของระบบวางแผนการทดสอบซอฟต์แวร์

1. **ฟังก์ชัน test plan** เป็นฟังก์ชันในการสร้างแผนการทดสอบ แสดงดังภาพที่ 3 ผู้ใช้งานจำเป็นต้องกรอกข้อมูลให้ครบถ้วน เนื่องจากทุกช่องต้องมีการเก็บค่าประสิทธิภาพของข้อกำหนดเชิงบรรทัดฐานเพื่อนำไปคำนวณค่าประสิทธิภาพ หากผู้ใช้งานไม่กรอกข้อมูลในช่องนั้นระบบทำการแจ้งเตือน ดังภาพที่ 3 เพื่อเป็นการควบคุมคุณภาพ และตรวจสอบความถูกต้องความสมบูรณ์ของซอฟต์แวร์ เมื่อผู้ใช้งานทำการทดสอบระบบเรียบร้อยแล้วคลิก “Save” เพื่อบันทึกข้อมูลลงตารางด้านล่าง จากนั้นคลิกไปที่ช่องเมนูบาร์เพื่อไปยังฟังก์ชัน Test Procedure Specification ต่อไป

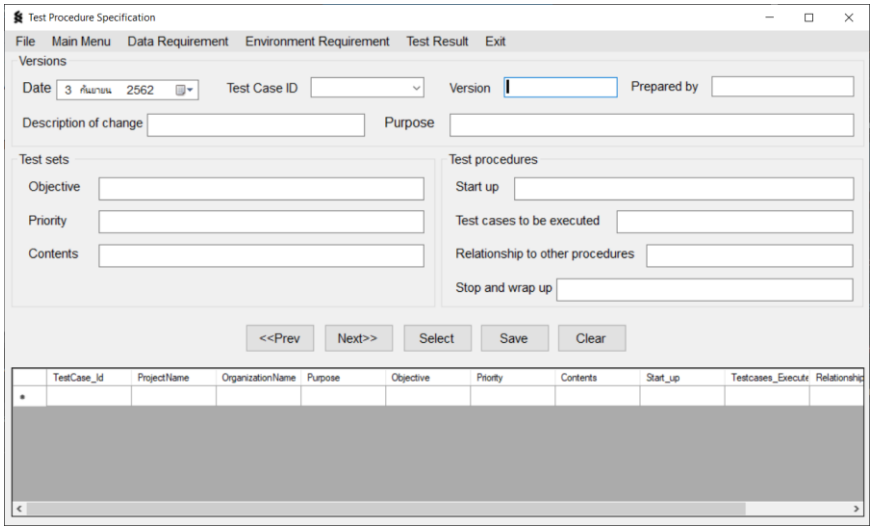


ภาพที่ 3 ฟังก์ชันในการสร้างแผนการทดสอบ

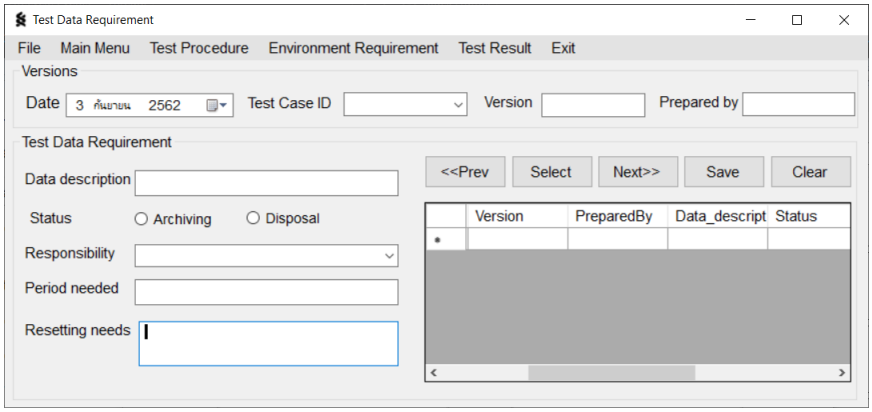
2. **ฟังก์ชัน test procedure specification** เป็นฟังก์ชันในการกำหนดขั้นตอนการทดสอบ แสดงดังภาพที่ 4 ผู้ใช้งานจำเป็นต้องเลือก Test Case ID ขึ้นมาเพื่อทำการรายงานผลที่เกิดขึ้นหลังจากการทดสอบซอฟต์แวร์ โดยในส่วนที่ 1 ผู้ใช้งานทำการกรอกข้อมูลรายละเอียดการทดสอบที่ประกอบด้วย วัตถุประสงค์ ลำดับความสำคัญ และเนื้อหาการทดสอบ ส่วนที่ 2 ผู้ใช้งานกรอกข้อมูลเกี่ยวกับขั้นตอนในการทดสอบที่ประกอบด้วย จุดเริ่มต้น กรณีทดสอบที่ใช้ ความสัมพันธ์ในแต่ละขั้นตอน จุดหยุดหรือเปลี่ยนการทดสอบ เมื่อผู้ใช้งานทำการกรอกข้อมูลเรียบร้อยแล้วคลิก “Save” เพื่อบันทึกข้อมูลลงตารางด้านล่าง จากนั้นคลิกไปที่ช่องเมนูบาร์เพื่อไปยังฟังก์ชัน test data requirement ต่อไป

3. **ฟังก์ชัน test data requirement** เป็นฟังก์ชันในการกำหนดความต้องการของข้อมูลที่จะนำมาใช้ในการทดสอบ แสดงดังภาพที่ 5 ประกอบด้วย ให้คำจำกัดความข้อมูล กำหนดสถานะให้ข้อมูล กำหนดหน้าที่ความรับผิดชอบ ระยะเวลาที่ต้องการ การตั้งค่าข้อมูล เมื่อผู้ใช้งานทำการกรอกข้อมูลเรียบร้อยแล้วคลิก “Save” เพื่อบันทึกข้อมูลลงตารางด้านล่าง จากนั้นคลิกไปที่ช่องเมนูบาร์เพื่อไปยังฟังก์ชัน Environment requirement ต่อไป

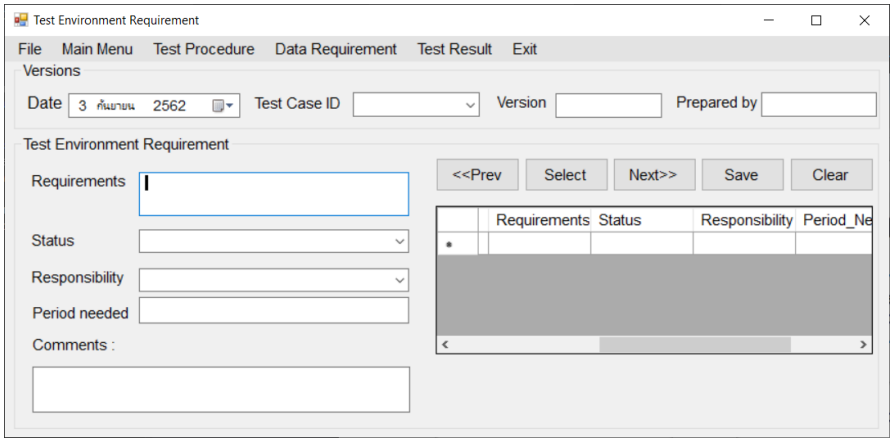
4. **ฟังก์ชัน test environment requirement** เป็นฟังก์ชันในการกำหนดสภาพแวดล้อมที่ใช้ในการทดสอบ แสดงดังภาพที่ 6 ซึ่งประกอบด้วย การระบุความต้องการ สถานะของสภาพแวดล้อม ช่วงเวลาที่ต้องการใช้งาน ส่วนแสดงความคิดเห็น เมื่อผู้ใช้งานทำการกรอกข้อมูลเรียบร้อยแล้วคลิก “Save” เพื่อบันทึกข้อมูลลงตารางด้านล่าง จากนั้นคลิกไปที่ช่องเมนูบาร์เพื่อไปยังฟังก์ชัน Test result ต่อไป



ภาพที่ 4 ข้อกำหนดของการทดสอบ

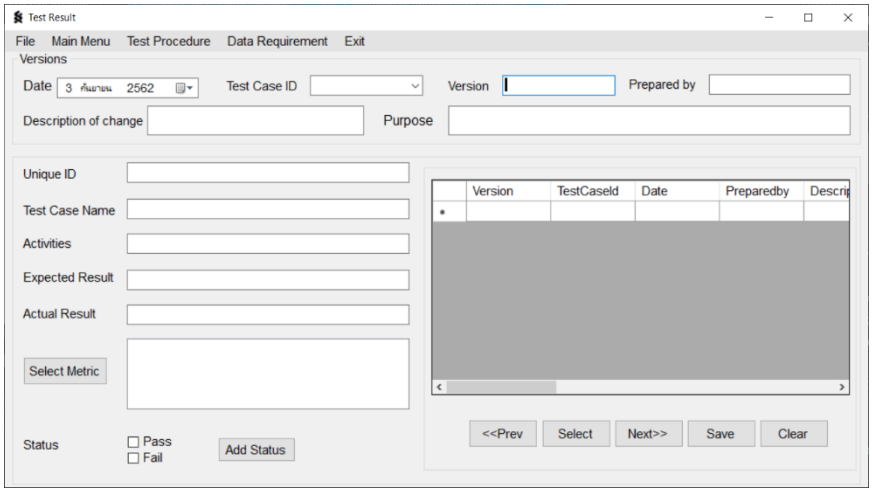


ภาพที่ 5 กำหนดความต้องการของข้อมูลในการทดสอบ



ภาพที่ 6 ข้อกำหนดความต้องการสภาพแวดล้อมในการทดสอบ

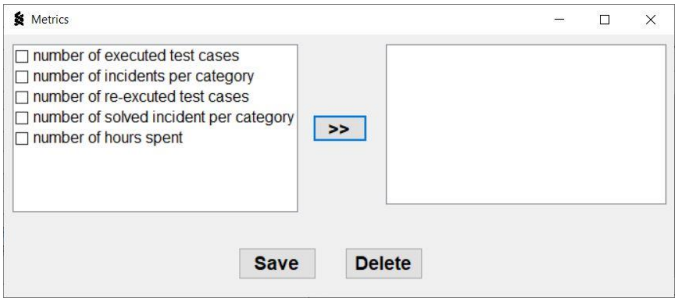
5. **ฟังก์ชัน test result** เป็นฟังก์ชันที่ใช้ในการกำหนดผลลัพธ์ที่ต้องการ พร้อมทั้งเลือกเมทริกซ์ที่จะนำไปใช้ในการวัดประสิทธิภาพของซอฟต์แวร์ แสดงดังภาพที่ 7 ซึ่งในฟังก์ชันนี้ประกอบด้วย กำหนดหมายเลข ระบุชื่อกรณีทดสอบ กิจกรรมที่ทดสอบ ผลลัพธ์ที่คาดหวัง ผลลัพธ์ที่เกิดขึ้นจริง เลือกเมทริกซ์โดยคลิกปุ่ม “Select Metric” เมื่อคลิกแล้วแสดงดังภาพที่ 8 ผู้ใช้งานเลือกเมทริกซ์ที่ต้องการนำมาทดสอบจากนั้นทำการบันทึกจะกลับมาสู่หน้าฟังก์ชันเดิม และเลือกสถานะของการทดสอบที่ทดสอบเสร็จแล้วโดยคลิกปุ่ม “Add status” ระบบจะไปยังฟังก์ชัน test status เป็นฟังก์ชันที่ใช้ในการรายงานสถานะกรณีทดสอบ (test case) ที่ได้ทำการทดสอบเสร็จแล้ว แสดงดังภาพที่ 9 ซึ่งประกอบด้วย ระบุวันที่ที่กำหนดสถานการณ์ทดสอบ ช่วงเวลาความคืบหน้าแผนการทดสอบ



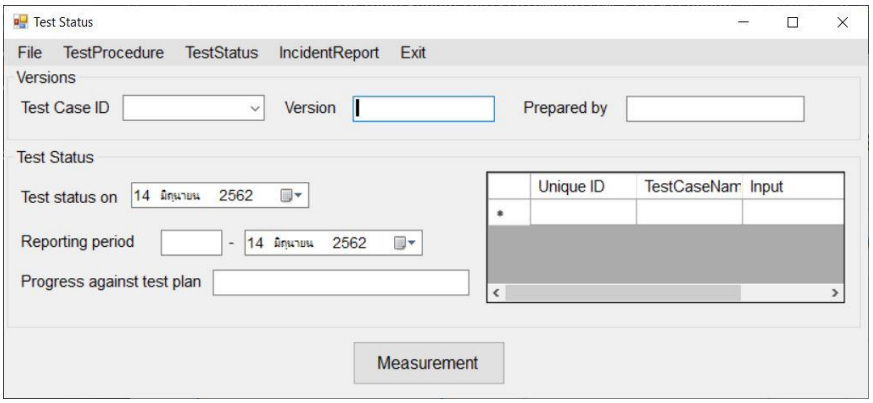
ภาพที่ 7 ข้อมูลผลการทดสอบ

เมื่อกรอกรายละเอียดสมบูรณ์แล้ว ต่อไปทำการคำนวณประสิทธิภาพของซอฟต์แวร์โดยคลิกปุ่ม “Measurement” เพื่อที่จะคำนวณประสิทธิภาพในด้านข้อกำหนดเชิงบรรทัดฐาน แสดงดังภาพที่ 10 แสดงระบบสามารถแสดงข้อมูลเป็นกราฟได้โดยคลิกปุ่ม

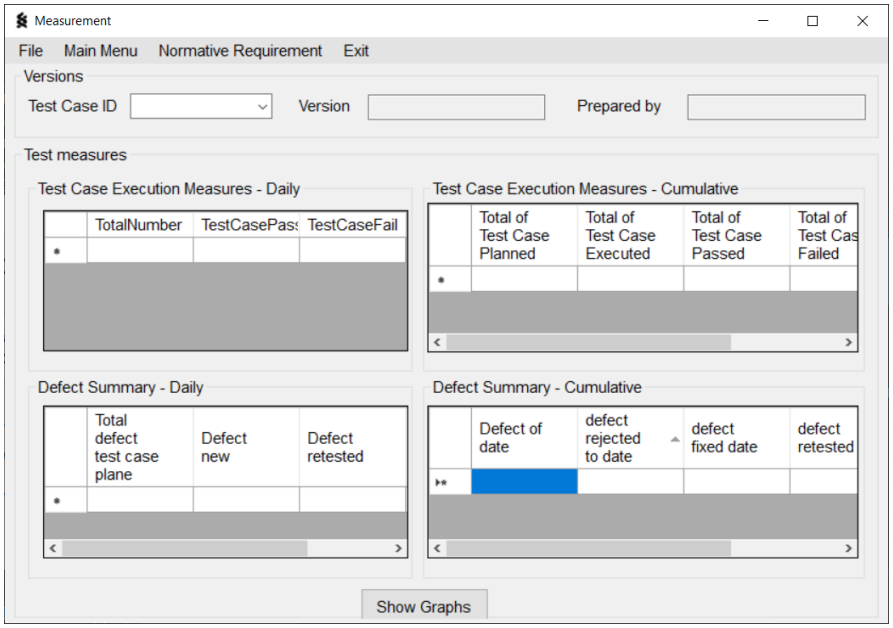
“Show graphs” แสดงดังภาพที่ 11 และเมื่อต้องการคำนวณประสิทธิภาพในการทดสอบซอฟต์แวร์ ไปที่แถบเมนูบาร์คลิกปุ่ม “Normative requirement” เพื่อคำนวณประสิทธิภาพแสดงดังภาพที่ 12 เมื่อผู้ใช้งานทำการกรอกข้อมูลเรียบร้อยแล้วคลิก “Save” เพื่อบันทึกข้อมูลลงตารางด้านล่าง หากการทดสอบมีสถานะไม่ผ่านจำเป็นต้องไปยังช่องเมนูบาร์ที่ฟังก์ชัน Incident report เพื่อสร้างรายงานข้อผิดพลาดที่เกิดขึ้น



ภาพที่ 8 เมทริกซ์ที่ใช้ในการวัดประสิทธิภาพ

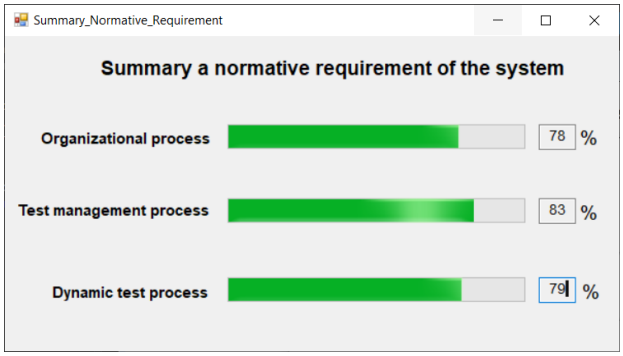


ภาพที่ 9 สถานการณ์ทดสอบ



ภาพที่ 10 วัดประสิทธิภาพการทดสอบ

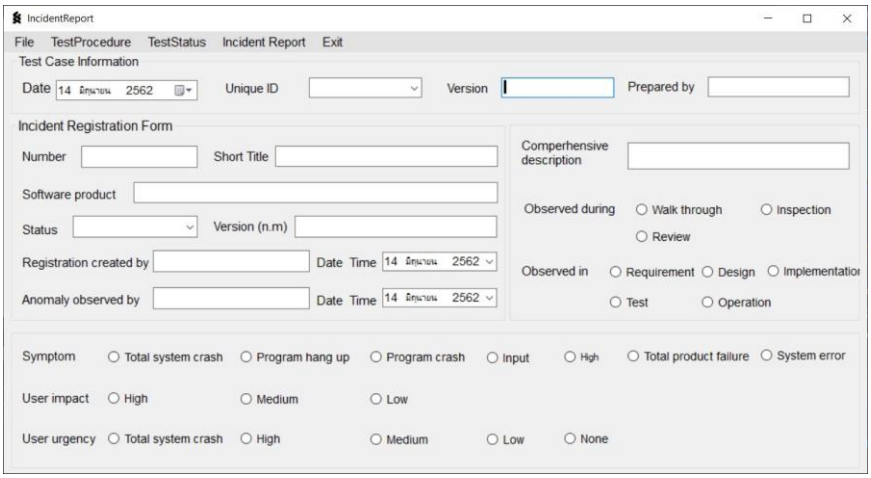
ภาพที่ 11 แสดงผลประสิทธิภาพการทดสอบ



ภาพที่ 12 ประสิทธิภาพในด้านข้อกำหนดเชิงบรรทัดฐาน

เมื่อผู้ใช้งานทำการกรอกข้อมูลเรียบร้อยแล้วคลิก “Save” เพื่อบันทึกข้อมูลลงตารางด้านล่าง หากการทดสอบมีสถานะไม่ผ่าน จำเป็นต้องไปยังช่องเมนูบาร์ที่ฟังก์ชัน Incident report เพื่อสร้างรายงานข้อผิดพลาดที่เกิดขึ้น

6. **ฟังก์ชัน incident report** เป็นฟังก์ชันในการรายงานเหตุการณ์ที่เกิดขึ้นระหว่างการทดสอบซอฟต์แวร์ แสดงดังภาพที่ 13 ประกอบด้วย หมายเลขรายงาน หัวข้อรายงาน ชนิดของซอฟต์แวร์ สถานะของการทดสอบ เวอร์ชันที่ทดสอบ ชื่อผู้สร้างรายงาน ชื่อผู้รายงานข้อผิดพลาด พร้อมระบุเวลารายงาน คำอธิบายรายงาน (comprehensive description) สถานการณ์รายงาน (observed during) ข้อผิดพลาดเกิดขึ้นในขั้นตอนไหน (observed in) ข้อผิดพลาดมีอาการแบบไหน (symptom) ผลกระทบต่อผู้ใช้งานในระดับไหน (user impact) ผู้ใช้งานมีความต้องการใช้งานในระดับไหน (user urgency)



ภาพที่ 13 รายงานข้อผิดพลาด

7. **ฟังก์ชัน documentation** เป็นฟังก์ชันในการพิมพ์เอกสารการทดสอบซอฟต์แวร์ออกมาใช้งาน ผู้ใช้งานสามารถสร้างและนำเอกสารการทดสอบซอฟต์แวร์ไปใช้ได้จากฟังก์ชันนี้ ซึ่งพิมพ์ออกมาได้ในรูปแบบสกุลไฟล์ pdf โดยสามารถคลิกปุ่ม Documentation ในภาพที่ 2 แล้วทำการเลือกเอกสารที่ต้องการ โดยภาพที่ 14 เป็นตัวอย่างเอกสารแผนการทดสอบที่สมบูรณ์แสดงอยู่ในไฟล์สกุล pdf

Test Plan Report 7/6/2562

Organization Name : KMITL	Project Name ทดสอบ UI	Test Plan Id : 1
Project Manager Name : -	Project Id : 1	Test Case Id : 1
Project Executor Name : ชฎาธาร	Created By :-	Created Date : 2562-07-06

Test Project Detail

TestPlan Name : ทดสอบ UI	Scope : ทดสอบให้สมัครสมาชิกได้
Module Name : สมัครสมาชิก	Module Test : สมัครสมาชิก
Test Cost Planning ฿ 0.00	TestCost Actual : ฿ 0.00
Overview of System : ทดสอบ UI	Standard : ISO 29119
Context Testing : ทดสอบการใช้งานในส่วน UI	Test Delivery : ระบบสามารถทำงานได้รับถ้วยสมบูรณ์
Risk Contingency : การกดซ้ำ มีผลต่อคำสั่งในการทำงาน	Object Test : ทดสอบการใช้งานได้สมบูรณ์
Criteria Pass : ระบบทดสอบสมบูรณ์	Criteria Fail : ระบบทดสอบไม่สมบูรณ์
Requirement : ทดสอบชื่อผู้ใช้งานและรหัสผ่านต่าง ๆ	Requirement : ทดสอบสิทธิในการใช้งาน
Requirement :	Requirement :
Reference : -	Glossary : -

Staff Name	Staff Role	Staff Activities	Staff Responsibility	Staff Hiring
ชฎาธาร	ทดสอบ	ทดสอบ	ทดสอบ	฿ 8,000.00

Test Actitivies	Date Activities
ทดสอบการทำงานของระบบการสมัครสมาชิก	2562-07-06

ภาพที่ 14 ตัวอย่างเอกสารประกอบการทดสอบ

อภิปรายผลการวิจัย

ระบบวางแผนการทดสอบซอฟต์แวร์อ้างอิงตามมาตรฐานการทดสอบซอฟต์แวร์ และวงจรการทดสอบซอฟต์แวร์ สามารถนำไปใช้ในส่วนการทดสอบของซอฟต์แวร์ โดยใช้ได้กับซอฟต์แวร์หลายประเภท ซึ่งช่วยลดระยะเวลาในการศึกษาข้อมูล ขั้นตอนและกระบวนการในการทดสอบซอฟต์แวร์ พร้อมทั้งยังลดต้นทุนในด้านการทดสอบซอฟต์แวร์ โดยระบบภายในงานวิจัยนี้มีการรวบรวมข้อมูล และรายละเอียดทั้งหมดของมาตรฐานการทดสอบซอฟต์แวร์ตามมาตรฐานสากลไว้เรียบร้อยแล้ว ซึ่งระบบสามารถตรวจสอบ และชี้แนวทางในการปรับปรุงซอฟต์แวร์ให้มีความถูกต้อง สมบูรณ์ มีคุณภาพ สามารถบอกระดับความจำเป็นของข้อมูล ทำให้การทดสอบซอฟต์แวร์สามารถทำได้ง่าย อีกทั้งสามารถวัดประสิทธิภาพในการทดสอบซอฟต์แวร์ เพื่อให้ทราบถึงคุณภาพของซอฟต์แวร์ และส่วนที่ต้องปรับปรุงก่อนส่งมอบหรือนำไปใช้งานซึ่งป้องกันการเกิด failure ของระบบได้อย่างดี พร้อมทั้งสามารถสร้างเอกสารทดสอบซอฟต์แวร์ที่ครอบคลุมทุกขั้นตอนการทดสอบและถูกต้องตามมาตรฐานในการทดสอบซอฟต์แวร์ ผู้ใช้งานสามารถนำไปใช้งานได้อย่างถูกต้อง ครบถ้วน

ข้อเสนอแนะ

สามารถนำงานวิจัยนี้ไปปรับใช้ในกระบวนการทดสอบซอฟต์แวร์ได้ในทุกประเภทของซอฟต์แวร์ ซึ่งนำไปใช้ในส่วนการทดสอบ และเอกสารการทดสอบซอฟต์แวร์ เพื่อให้ซอฟต์แวร์ที่ทดสอบนั้นมีความถูกต้อง ครบถ้วน ตามหลักมาตรฐานการทดสอบซอฟต์แวร์สากล ซึ่งสามารถใช้งานได้ง่ายขึ้น ครอบคลุมเนื้อหามาตรฐานการทดสอบซอฟต์แวร์ทั้งหมด

เอกสารอ้างอิง

The Home of International Standards for Software Testing. (2018). IEEE Std 29119, Part 1, Part 2, Part 3, Part 4, Part 5. สืบค้นเมื่อ 6 เมษายน 2562, จาก https://www.softwaretestingstandard.org/index.php.

สำนักงานปลัดกระทรวงศึกษาธิการ. (2561). ยุค Digital 4.0 เมื่อโลกขับเคลื่อนด้วยเทคโนโลยี. สืบค้นเมื่อ 29 สิงหาคม 2562, จาก http://www.ops.moe.go.th/ops2017/สาระน่ารู้/2876-ยุค-digital-4-0-เมื่อโลกขับเคลื่อนด้วยเทคโนโลยี-25-ก-ย-2561.

P. Eira, R. J. Machado and A. Silva. (2018). Tailoring ISO/IEC/IEEE 29119-3 Standard for Small and Medium-sized Enterprises. *IEEE Trans, On computer society,* 380-389.

N. Chen, Ethan W. Chen and Ian S. Chen. (2018). Integration Software Testing Standard ISO/IEC/IEEE 29119 to Agile Development. *SERP'18*, 89-95.

A. Rana, A. Singh Rawat and A. Bijalwan. (2017). Process of finding defects in Software testing. *Intelligent and Computing in Engineering*. 10, 297-300.

U. Kumar Tiwari, S. Kumar. (2017). Component Level Testing in Component-Based Software Development. *International Journal of Innovations & Advancement in Computer Science*. 6, 69-73.

E. Veenendaal. (2016). TMMi and ISO/IEC 29119: Friends or Foes?. *TMMi Foundation*, 1-12.

International Standard ISO/IEC/IEEE 29119-3. (2013). Software and systems engineering – Software Testing-Part 3: Test documentation. *IEEE Xplore*.

International Standard ISO/IEC/IEEE 29119-2. (2013). Software and systems engineering – Software Testing–Part 2: Test processes. *IEEE Xplore*.

จักรินทร์ สันติรัตนภักดี และจิติมนต์ อั่งสกุล. (2012). การประเมินความสามารถในการใช้งานซอฟต์แวร์วางแผนมื้ออาหารเพื่อการควบคุมน้ำหนัก. *วารสารเทคโนโลยีสุรนารี,* 6(2), 73-98.

KM Tank. (2553). การรักษาความมั่นคงปลอดภัยของระบบสารสนเทศ. สืบค้นเมื่อ 2 กันยายน 2562, จาก https://www.tistr.or.th/tistrblog/?p=703.