|  |
| --- |
| การวัดความคล่องตัวต่อการเปลี่ยนแปลงสำหรับระบบอิงบริการ |
| นายคงสวัสดิ์ โพธิพนม |
| โครงงานมหาบัณฑิตนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต  สาขาวิชาวิศวกรรมซอฟต์แวร์ภาควิชาวิศวกรรมคอมพิวเตอร์  คณะวิศวกรรมศาสตร์จุฬาลงกรณ์มหาวิทยาลัย  ปีการศึกษา 2554 |
| MEASUREMENT OF AGILITY TO CHANGE FOR SERVICE-BASED SYSTEMS |
| Mr. KongsawasPotipanom |
| A Master Project Submitted in Partial Fulfillment of the Requirements  for the Degree of Master of Science in Software Engineering  Department of Computer Engineering  Faculty of Engineering  Chulalongkorn University  Academic Year 2011 |

|  |  |
| --- | --- |
| หัวข้อโครงงานมหาบัณฑิต | การวัดความคล่องตัวต่อการเปลี่ยนแปลงสำหรับระบบอิงบริการ |
| โดย | นายคงสวัสดิ์ โพธิพนม |
| ภาควิชา | วิศวกรรมคอมพิวเตอร์ |
| อาจารย์ที่ปรึกษา  หน่วยงานที่ร่วมโครงการ  ผู้แทนจากหน่วยงาน | รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา  -  - |
|  |  |

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับโครงงานมหาบัณฑิตฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

………………………………………….. หัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์

(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

…………………………………………... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

|  |  |
| --- | --- |
| คงสวัสดิ์ โพธิพนม : การวัดความคล่องตัวต่อการเปลี่ยนแปลงสำหรับระบบอิงบริการ (MEASUREMENT OF AGILITY TO CHANGE FOR SERVICE-BASED SYSTEMS) อาจารย์ที่ปรึกษา : รศ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา, 68 หน้า.    แนวคิดสถาปัตยกรรมเชิงบริการได้กลายมาเป็นแนวทางที่น่าสนใจในการแก้ปัญหาการพัฒนาและบูรณาการซอฟต์แวร์ ในการนำแนวคิดของสถาปัตยกรรมเชิงบริการมาใช้อย่างเต็มรูปแบบนั้น องค์กรธุรกิจต่าง ๆ จำเป็นต้องศึกษาทำความเข้าใจเกี่ยวกับระเบียบวิธีแบบใหม่สำหรับสถาปัตยกรรมเชิงบริการ และนำสถาปัตยกรรมนี้มาใช้ร่วมกับระบบซอฟต์แวร์ต่าง ๆ ที่มีอยู่แล้ว ความพยายามดังกล่าวมักจะเห็นผลในภายหลังเมื่อระบบสามารถรองรับการเปลี่ยนแปลงได้อย่างยืดหยุ่น อย่างไรก็ตามเนื่องจากประโยชน์นั้นไม่ส่งผลทันที องค์กรธุรกิจโดยเฉพาะองค์กรขนาดเล็กจึงเกิดความลังเลในการนำสถาปัตยกรรมเชิงบริการมาใช้ โครงงานมหาบัณฑิตนี้จึงได้นำเสนอการทดลองเพื่อพิจารณาถึงความคล่องตัวต่อการเปลี่ยนแปลงของระบบอิงบริการเทียบกับระบบที่ไม่อิงบริการเพื่อทดสอบว่าระบบอิงบริการมีความคล่องตัวมากกว่าระบบที่ไม่อิงบริการหรือไม่ การทดลองจะใช้ระบบบันทึกการลาของบริษัทพัฒนาซอฟต์แวร์แห่งหนึ่งเป็นหน่วยทดลอง โดยที่แบบจำลองการวัดความคล่องตัวที่เสนอนั้นได้ทำการขยายจากตัววัดความคล่องตัวของซอฟต์แวร์ที่มีอยู่เดิมโดยเพิ่มการพิจารณาในส่วนความซับซ้อนของโค้ด ผลการทดลองได้แสดงว่าระบบอิงบริการมีความคล่องตัวกว่าระบบที่ไม่อิงบริการ และในส่วนความซับซ้อนของโค้ดนั้นสามารถช่วยให้เห็นภาพที่ชัดเจนขึ้นเกี่ยวกับความซับซ้อนของการเปลี่ยนแปลงที่ส่งผลต่อความคล่องตัว นอกจากนี้จากผลการทดลองยังพบว่าหากมีการออกแบบเซอร์วิซที่ดีจะสามารถปรับปรุงให้ระบบอิงบริการมีความคล่องตัวมากขึ้นได้ | |
| ภาควิชาวิศวกรรมคอมพิวเตอร์ | ลายมือชื่อนิสิต |
| สาขาวิชาวิศวกรรมซอฟต์แวร์  signature.............................................. | ลายมือชื่ออาจารย์ที่ปรึกษา |
| ปีการศึกษา 2554  signature.......................................... |  |

|  |  |
| --- | --- |
| # # 5371404221 : MAJOR SOFTWARE ENGINEERING  KEYWORDS : AGILITY, SERVICE ORIENTATION, MEASUREMENT  KONGSAWAS POTIPANOM :  MEASUREMENT OF AGILITY TO CHANGE FOR SERVICE-BASED SYSTEMS. PROJECT ADVISOR: ASSOC. PROF. TWITTIE SENIVONGSE,Ph.D. 68 pp.  Service orientation has become an attractive solution to many software development and integration problems. To fully adopt the service orientation concept, business organisations need to learn about a new methodology for service architecture and employ it to their existing software systems. Such an effort is likely to pay off later when the service-based systems can accommodate changes in a more flexible way. However, since the benefits are not seen right away, this could deter business organisations, especially smaller enterprises, from service adoption. This master project presents an experiment to determine agility to change of a service-based system in comparison with that of a non-service-based one to check if the service-based version is more agile to change. The two experimental systems are in the context of an E-Leave application of a small software company in Thailand. The model we use to measure agility extends an existing software agility metric by bringing code complexity into the model. The experimental results show that the service-based system is more agile to change than its counterpart, while code complexity can help give a better view of change complexity that affects agility. We also note that agility of the service-based system can be improved if the services are well-designed. | |
| Department : Computer Engineering | Student’s Signature |
| Field of Study : Software Engineering  signature.............................................. | Advisor’s Signature |
| Academic Year : 2011  signature.......................................... |  |

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ รศ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาโครงงานมหาบัณฑิตเป็นอย่างยิ่งที่ได้สละเวลาให้คำปรึกษา คำแนะนำ และแนวทางสำหรับการทำโครงงานมหาบัณฑิต รวมทั้งเป็นผู้ประสานงานให้ความช่วยเหลือแก่นิสิตที่ทำโครงงานมหาบัณฑิตทุกคน

ขอกราบขอบพระคุณ รศ.ดร.วิวัฒน์ วัฒนาวุฒิและผศ.ดร.สืบสกุล พิภพมงคล คณะกรรมการคุมสอบโครงงานมหาบัณฑิตเป็นอย่างยิ่ง ที่ได้กรุณาแนะนำแนวทาง รวมถึงการตรวจสอบและแก้ไขโครงงานมหาบัณฑิตนี้

ขอขอบคุณ คณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้คำแนะนำ ความรู้และแนวทางการทำโครงงานมหาบัณฑิต

ขอขอบคุณ พี่ ๆ เพื่อน ๆ ในหน่วยงานและเพื่อน ๆ หลักสูตรวิศวกรรมซอฟต์แวร์ สำหรับกำลังใจและคำแนะนำในการจัดทำโครงงานมหาบัณฑิต

สุดท้ายนี้ ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ รวมถึงสมาชิกทุก ๆ คนในครอบครัวที่ให้การสนับสนุนและให้กำลังใจที่ดีเสมอมา

คงสวัสดิ์ โพธิพนม

สารบัญ

หน้า

บทคัดย่อภาษาไทย ...........ง

บทคัดย่อภาษาอังกฤษ ...........จ

[กิตติกรรมประกาศ ฉ](#_Toc319798037)

[สารบัญ ช](#_Toc319798038)

[สารบัญตาราง ฌ](#_Toc319798039)

[สารบัญรูป ฎ](#_Toc319798040)

[บทที่ 1 บทนำ 1](#_Toc321857667)

[1.1 ที่มาและความสำคัญของปัญหา 1](#_Toc321857668)

[1.2 วัตถุประสงค์ของโครงงาน 2](#_Toc321857669)

[1.3 ขอบเขตของโครงงาน 2](#_Toc321857670)

[1.4 ประโยชน์ที่คาดว่าจะได้รับ 3](#_Toc321857671)

[1.5 ขั้นตอนและวิธีการดำเนินโครงงาน 3](#_Toc321857672)

[1.6 ผลงานที่ตีพิมพ์จากโครงงานมหาบัณฑิต 3](#_Toc321857673)

[บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 4](#_Toc321857674)

[2.1 ทฤษฎีที่เกี่ยวข้อง 4](#_Toc321857675)

[*2.2* งานวิจัยที่เกี่ยวข้อง 7](#_Toc321857676)

[บทที่ 3 แบบจำลองการวัดความคล่องตัว 10](#_Toc321857677)

[บทที่ 4 วิธีการดำเนินงาน 13](#_Toc321857678)

[4.1 การคัดเลือกหน่วยทดลอง 13](#_Toc321857679)

[4.2 การเลือกรายการการเปลี่ยนแปลง 18](#_Toc321857680)

[4.3 การรวบรวมข้อมูลจากการทดลอง 19](#_Toc321857681)

[4.4 สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา 21](#_Toc321857682)

[บทที่ 5 การทดลองและการวิเคราะห์ผลการทดลอง 22](#_Toc321857683)

[5.1 การดำเนินการทดลอง 22](#_Toc321857684)

[5.2 ผลการทดลอง 38](#_Toc321857685)

[5.3 การคำนวณค่าความคล่องตัว 44](#_Toc321857688)

[5.4 สรุปผลการทดลอง 50](#_Toc321857691)

[บทที่ 6 บทสรุปโครงงาน อภิปรายผล และข้อเสนอแนะ 51](#_Toc321857697)

[6.1 สรุปผลโครงงานมหาบัณฑิต 51](#_Toc321857698)

[*6.2* ปัญหาและข้อจำกัดในการทำโครงงาน 51](#_Toc321857699)

[*6.3* ข้อเสนอแนะ 52](#_Toc321857700)

[รายการอ้างอิง 53](#_Toc321857701)

[ภาคผนวก 54](#_Toc321857702)

[ภาคผนวก ก แผนภาพการออกแบบระบบการเปลี่ยนแปลงรายการที่ 1 55](#_Toc321857703)

[ภาคผนวก ข แผนภาพการออกแบบระบบการเปลี่ยนแปลงรายการที่ 2 59](#_Toc321857704)

[ภาคผนวก ค ตารางบันทึกเวลาการทำงาน 64](#_Toc321857705)

[ประวัติผู้เขียนโครงงานมหาบัณฑิต 68](#_Toc321857706)

สารบัญตาราง

|  |  |  |
| --- | --- | --- |
|  |  | หน้า |

[ตารางที่ 4.1 ตัวอย่างตารางบันทึกผลกระทบจากการเปลี่ยนแปลง (Non Service) 20](#_Toc321528662)

[ตารางที่ 4.2 ตัวอย่างตารางบันทึกผลกระทบจากการเปลี่ยนแปลง (Service) 20](#_Toc321528663)

[ตารางที่ 5.1 คำอธิบายยูสเคสการยกยอดวันลา 23](#_Toc321528664)

[ตารางที่ 5.2 การทดสอบการยกยอดวันลาคงเหลือ (Non Service) 26](#_Toc321528665)

[ตารางที่ 5.3 การทดสอบการยกยอดวันลาคงเหลือ (Service) 27](#_Toc321528666)

[ตารางที่ 5.4 คำอธิบายยูสเคสการปรับปรุงข้อมูลประเภทการทำงานเป็นกะเวลาของพนักงาน 29](#_Toc321528667)

[ตารางที่ 5.5 คำอธิบายยูสเคสการบันทึกการลาของพนักงานที่ทำงานเป็นกะ 30](#_Toc321528668)

[ตารางที่ 5.6 การทดสอบการกำหนดประเภทการทำงานของพนักงานเป็นแบบกะ(Non Service) 35](#_Toc321528669)

[ตารางที่ 5.7 การทดสอบการกำหนดประเภทการทำงานของพนักงานเป็นแบบกะ(Service) 36](#_Toc321528670)

[ตารางที่ 5.8 การทดสอบการบันทึกการลา(Non Service) 37](#_Toc321528671)

[ตารางที่ 5.9 การทดสอบการบันทึกการลา(Service) 38](#_Toc321528672)

[ตารางที่ 5.10 ผลกระทบจากการเปลี่ยนแปลงรายการที่ 1 (Non Service) 39](#_Toc321528673)

[ตารางที่ 5.11 ผลกระทบจากการเปลี่ยนแปลงรายการที่ 1 (Service) 39](#_Toc321528674)

[ตารางที่ 5.12 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 1 (Non Service) 40](#_Toc321528675)

[ตารางที่ 5.13 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 1 (Service) 40](#_Toc321528676)

[ตารางที่ 5.14 ผลกระทบจากการเปลี่ยนแปลงรายการที่ 2 (Non Service) 42](#_Toc321528677)

[ตารางที่ 5.15 ผลกระทบจากการเปลี่ยนแปลงรายการที่ 2 (Service) 42](#_Toc321528678)

[ตารางที่ 5.16 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 2 (Non Service) 43](#_Toc321528679)

[ตารางที่ 5.17 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 2 (Service) 43](#_Toc321528680)

[ตารางที่ 5.18 การคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 1 (Non Service) 45](#_Toc321528681)

[ตารางที่ 5.19 การคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 1 (Service) 45](#_Toc321528682)

[ตารางที่ 5.20 ตารางสรุปการคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 1 46](#_Toc321528683)

[ตารางที่ 5.21 การคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 2 (Non Service) 48](#_Toc321528684)

[ตารางที่ 5.22 การคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 2 (Service) 48](#_Toc321528685)

[ตารางที่ 5.23 ตารางสรุปการคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 2 49](#_Toc321528686)

สารบัญรูป

|  |  |
| --- | --- |
|  | หน้า |

[รูปที่ 2.1 ระดับชั้นของสถาปัตยกรรมเชิงบริการ 5](#_Toc321531929)

[รูปที่ 2.2 ตัวอย่างการพัฒนาด้วยสถาปัตยกรรมเชิงบริการเทียบกับการพัฒนาด้วยวิธีเดิม 6](#_Toc321531930)

[รูปที่ 2.3 ตัวอย่างกราฟการไหลของโปรแกรม และการคำนวณค่าความซับซ้อนไซโคลมาติก 7](#_Toc321531931)

[รูปที่ 3.1 มิติความซับซ้อนของการเปลี่ยนแปลง 11](#_Toc321531932)

[รูปที่ 4.1 สถาปัตยกรรมของระบบบันทึกการลา 13](#_Toc321531933)

[รูปที่ 4.2 แผนภาพดีพลอยเมนต์ของระบบบันทึกการลา 15](#_Toc321531934)

[รูปที่ 4.3 แผนภาพความสัมพันธ์ของคลาสของระบบบันทึกการลา 16](#_Toc321531935)

[รูปที่ 4.4 การออกแบบเซอร์วิซของระบบบันทึกการลา 16](#_Toc321531936)

[รูปที่ 4.5 สถาปัตยกรรมของระบบบันทึกการลาแบบอิงบริการ 17](#_Toc321531937)

[รูปที่ 4.6 แผนภาพดีพลอยเมนต์ของระบบบันทึกการลาแบบอิงบริการ 17](#_Toc321531938)

[รูปที่ 4.7 ตัวอย่างตารางบันทึกเวลาการทำงาน 19](#_Toc321531939)

[รูปที่ 4.8 โปรแกรมลอคเมตริกส์ 20](#_Toc321531940)

[รูปที่ 5.1 แผนภาพยูสเคสของการเปลี่ยนแปลงรายการที่ 1 23](#_Toc321531941)

[รูปที่ 5.2 แผนภาพความสัมพันธ์ของคลาสของการเปลี่ยนแปลงรายการที่ 1 25](#_Toc321531942)

[รูปที่ 5.3 หน้าจอเมนูส่วนการยกยอดวันลา 25](#_Toc321531943)

[รูปที่ 5.4 หน้าจอการยกยอดวันลา 26](#_Toc321531944)

[รูปที่ 5.5 แผนภาพยูสเคสของการเปลี่ยนแปลงรายการที่ 2 29](#_Toc321531945)

[รูปที่ 5.6 แผนภาพคลาสของการเปลี่ยนแปลงรายการที่ 2 32](#_Toc321531946)

[รูปที่ 5.7 หน้าจอการปรับปรุงข้อมูลพนักงาน 33](#_Toc321531947)

[รูปที่ 5.8 หน้าจอการบันทึกการลา 34](#_Toc321531948)

[รูปที่ 5.9 ข้อมูลประวัติการลาของพนักงาน 34](#_Toc321531949)

[รูปที่ 5.10 ความแตกต่างของโค้ดแบบไม่เรียกใช้เซอร์วิซกับแบบเรียกใช้เซอร์วิซ 47](#_Toc321531950)

[รูปที่ ก.1 แผนภาพกิจกรรมของการเปลี่ยนแปลงรายการที่ 1 (Non Service) 55](#_Toc321531951)

[รูปที่ ก.2 แผนภาพกิจกรรมของการเปลี่ยนแปลงรายการที่ 1 (Service) 56](#_Toc321531952)

[รูปที่ ก.4 แผนภาพลำดับของการเปลี่ยนแปลงรายการที่ 1 (Service) 58](#_Toc321531953)

[รูปที่ ข.1 แผนภาพกิจกรรมการปรับปรุงประเภทการทำงานของพนักงาน (Non Service) 59](#_Toc321531954)

[รูปที่ ข.2 แผนภาพกิจกรรมการปรับปรุงประเภทการทำงานของพนักงาน (Service) 60](#_Toc321531955)

[รูปที่ ข.3 แผนภาพกิจกรรมการบันทึกการลา (Non Service) 61](#_Toc321531956)

[รูปที่ ข.4 แผนภาพกิจกรรมการบันทึกการลา (Service) 62](#_Toc321531957)

[รูปที่ ข.5 แผนภาพลำดับของการบันทึกการลา (Non Service) 62](#_Toc321531958)

[รูปที่ ข.6 แผนภาพลำดับของการบันทึกการลา (Service) 63](#_Toc321531959)

[รูปที่ ข.7 แผนภาพลำดับของการคำนวณวันลาที่ใช้ 63](#_Toc321531960)

[รูปที่ ค.1 การบันทึกเวลาการทำงานการเปลี่ยนแปลงรายการที่ 1 (Non Service) 64](#_Toc321531961)

[รูปที่ ค.2 การบันทึกเวลาการทำงานการเปลี่ยนแปลงรายการที่ 1 (Service) 65](#_Toc321531962)

[รูปที่ ค.3 การบันทึกเวลาการทำงานการเปลี่ยนแปลงรายการที่ 2 (Non Service) 66](#_Toc321531963)

[รูปที่ ค.4 การบันทึกเวลาการทำงานการเปลี่ยนแปลงรายการที่ 2 (Service) 67](#_Toc321531964)

# [บทที่ 1](#สารบัญ) [บทนำ](#สารบัญ)

## [ที่มาและความสำคัญของปัญหา](#สารบัญ)

แนวคิดสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture) เป็นแนวคิดในการออกแบบระบบซอฟต์แวร์ ที่มีส่วนประกอบพื้นฐานของระบบคือ บริการหรือเซอร์วิซ ซึ่งเป็นหน่วยของซอฟต์แวร์ (Software Unit) ที่ทำฟังก์ชันงานบางอย่าง และให้บริการแก่ส่วนอื่นของระบบผ่านส่วนต่อประสาน (Interface) ที่ประกาศไว้ การพัฒนาซอฟต์แวร์ทำได้โดยเรียกใช้เซอร์วิซที่มีอยู่แล้ว หรือนำเซอร์วิซต่าง ๆ มาประกอบกันเป็นเซอร์วิซใหม่ หรือเรียกใช้เซอร์วิซต่าง ๆ เพื่อตอบสนองกระบวนการธุรกิจ การพัฒนาจึงทำได้สะดวกรวดเร็ว เพราะสามารถนำเซอร์วิซที่มีอยู่กลับมาใช้ใหม่ได้ และสามารถเปลี่ยนหรือถอดประกอบเซอร์วิซเป็นตัวใหม่ได้ บทความและวรรณกรรมที่เกี่ยวกับการพัฒนาระบบด้วยแนวคิดสถาปัตยกรรมเชิงบริการ ได้กล่าวถึงประโยชน์ของการพัฒนาระบบด้วยแนวคิดนี้ไว้หลายประการด้วยกัน เช่น ช่วยเพิ่มความสามารถในการทำงานร่วมกันกับระบบต่าง ๆ ภายในองค์กร เพิ่มผลตอบแทนจากการลงทุน เพิ่มความคล่องตัว และลดต้นทุนที่ใช้ในการพัฒนาระบบ เป็นต้น ซึ่งเป็นสิ่งจูงใจให้องค์กรต่าง ๆ ในปัจจุบันสนใจที่จะนำแนวคิดสถาปัตยกรรมเชิงบริการไปปรับเปลี่ยน หรือพัฒนาระบบของตนให้สอดคล้องกับเป้าหมายทางธุรกิจขององค์กร

ผู้วิจัยเห็นว่า แม้ว่าบทความ และวรรณกรรมต่าง ๆ จะกล่าวถึงประโยชน์ของสถาปัตยกรรมเชิงบริการไว้มากมาย แต่ประโยชน์นั้นน่าจะเห็นผลหลังจากที่ได้มีการพัฒนาระบบไปแล้ว คือเมื่อมีความต้องการสร้างระบบใหม่อิงเซอร์วิซเดิมหรือต้องการปรับเปลี่ยนระบบเดิม แต่ประโยชน์อาจจะไม่ปรากฏในทันที องค์กรอาจประสบปัญหาในการนำแนวคิดสถาปัตยกรรมเชิงบริการมาใช้เช่นกัน เนื่องด้วยต้องใช้เวลา และความพยายามไปกับการศึกษาทำความเข้าใจแนวคิดวิธีการในการพัฒนา การศึกษาถึงผลกระทบ รวมถึงการเปลี่ยนแปลงต่าง ๆ ภายในองค์กรกว่าจะพัฒนาระบบด้วยแนวคิดสถาปัตยกรรมเชิงบริการขึ้นมาได้

ดังนั้นผู้วิจัยจึงมีความสนใจที่จะทำการทดลองเพื่อทดสอบประโยชน์ของการพัฒนาระบบด้วยแนวคิดสถาปัตยกรรมเชิงบริการว่ามีความเหมาะสมที่จะเลือกนำมาใช้ในการพัฒนาระบบ โดยเฉพาะในช่วงของการบำรุงรักษาระบบหลังจากที่ระบบได้ถูกส่งมอบไปแล้วนั้น มักจะพบกับการเปลี่ยนแปลงระบบอยู่บ่อยครั้ง เนื่องจากต้องการปรับให้เหมาะสมกับการทำงาน หรือให้สอดคล้องกับการเปลี่ยนแปลงทางธุรกิจ ด้วยเหตุนี้ผู้วิจัยจึงเลือกที่จะพิจารณาถึงประโยชน์ของการพัฒนาระบบด้วยแนวคิดสถาปัตยกรรมเชิงบริการในแง่ความคล่องตัว (Agility) ซึ่งในที่นี้จะหมายถึงความสามารถของระบบในการตอบสนองต่อการเปลี่ยนแปลง การทดลองจะทำการเปรียบเทียบความคล่องตัวของระบบอิงบริการ (Service-Based System) กับระบบที่ไม่ได้อิงบริการ (Non Service-Based System) ในการดำเนินการกับการเปลี่ยนแปลง การเปรียบเทียบจะใช้แบบจำลองตัววัดความคล่องตัว (Agility Metric) ซึ่งพิจารณาจากความซับซ้อนของการเปลี่ยนแปลง (Change Complexity) กับระยะเวลาที่ใช้ในการดำเนินการกับการเปลี่ยนแปลง (Change Duration) โดยในส่วนของความซับซ้อนของการเปลี่ยนแปลงนั้นจะพิจารณาในสามมิติด้วยกัน ได้แก่ มิติของระดับชั้นในเชิงสถาปัตยกรรมของซอฟต์แวร์ที่การเปลี่ยนแปลงนั้นไปกระทบ (Architectural Depth) มิติความกว้างของกระบวนการทำงาน (Process Breadth) และมิติความซับซ้อนของโค้ด (Code Complexity) ที่การเปลี่ยนแปลงนั้นไปกระทบ

## [วัตถุประสงค์ของโครงงาน](#สารบัญ)

เพื่อเสนอตัววัดที่สามารถใช้ในการวัดความคล่องตัวต่อการเปลี่ยนแปลงของระบบอิงบริการ และเปรียบเทียบความคล่องตัวต่อการเปลี่ยนแปลงระหว่างระบบอิงบริการและระบบที่ไม่อิงบริการ

## [ขอบเขตของโครงงาน](#สารบัญ)

* + 1. เสนอตัววัดความคล่องตัวที่พิจารณาความสัมพันธ์ของความซับซ้อนของคำขอเปลี่ยนแปลง เทียบกับเวลาที่ใช้ดำเนินการต่อคำขอเปลี่ยนแปลง และมองความซับซ้อนของคำขอเปลี่ยนในลักษณะของปริมาตรของการเปลี่ยนแปลง ซึ่งพิจารณาจาก ระดับชั้นในเชิงสถาปัตยกรรมของซอฟต์แวร์ที่คำขอเปลี่ยนแปลงนั้นไปกระทบ ความกว้างของกระบวนการทำงาน และความซับซ้อนของโค้ดที่คำขอเปลี่ยนแปลงนั้นไปกระทบ
    2. เพื่อทำการทดลองดูค่าความคล่องตัวของระบบที่พัฒนาแบบอิงบริการ เทียบกับระบบที่พัฒนาแบบไม่อิงบริการในการตอบสนองต่อการเปลี่ยนแปลงในลักษณะต่าง ๆ
    3. ใช้ระบบบันทึกการลาของบริษัทพัฒนาซอฟต์แวร์แห่งหนึ่ง เป็นหน่วยทดลองโดยใช้เป็นตัวแทนของระบบที่ไม่อิงบริการ ส่วนตัวแทนของระบบอิงบริการจะจัดทำขึ้นจากระบบบันทึกการลา โดยจะมีการพิจารณาออกแบบเซอร์วิซหรือบริการที่เป็นเซอร์วิซเอนทิตี (Entity Services) ซึ่งเป็นเซอร์วิซที่เกี่ยวข้องกับเอนทิตีของข้อมูลในระบบ และเซอร์วิซอรรถประโยชน์ (Utility Services) ที่มีแนวโน้มของการนำกลับมาใช้ใหม่สูง
    4. กำหนดให้ช่วงของการดำเนินงานแบ่งออกเป็นสามช่วงประกอบด้วย ช่วงการวิเคราะห์ออกแบบ ช่วงการพัฒนาโปรแกรม และช่วงการทดสอบโปรแกรม โดยสมมติให้ผู้วิจัยทำหน้าที่ในสามตำแหน่งได้แก่ นักวิเคราะห์ระบบ ผู้พัฒนาโปรแกรม และผู้ทดสอบระบบ
    5. ระบบที่ใช้เป็นหน่วยทดลองจะทำการพัฒนา และติดตั้งบนเครื่องแสตนด์อะโลนเท่านั้น

## [ประโยชน์ที่คาดว่าจะได้รับ](#สารบัญ)

* + 1. องค์กรสามารถนำตัววัดความคล่องตัวไปใช้ในการวัดความคล่องตัวของระบบซอฟต์แวร์ที่ตนได้ทำการพัฒนา ไม่ว่าจะเป็นระบบที่เป็นแบบอิงบริการหรือไม่ก็ตาม เพื่อนำไปปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ของตนในลำดับต่อไปได้
    2. ได้ข้อมูลเกี่ยวกับความคล่องตัวของระบบแบบอิงบริการในแง่ของการตอบสนองต่อการเปลี่ยนแปลง เพื่อเป็นประโยชน์ต่อการพิจารณาขององค์กรในการนำสถาปัตยกรรมเชิงบริการมาใช้

## ขั้นตอนและวิธีการดำเนินโครงงาน

* + 1. ศึกษาวิธีการที่ใช้ในการวัดความคล่องตัวต่อการเปลี่ยนแปลงของระบบซอฟต์แวร์
    2. ปรับปรุงตัววัดที่ใช้ในการวัดความคล่องตัวต่อการเปลี่ยนแปลงของระบบซอฟต์แวร์
    3. ทำการคัดเลือกหน่วยทดลองสำหรับใช้ในการทดลอง
    4. สร้างหน่วยทดลองที่เป็นระบบอิงบริการ
    5. ทำการคัดเลือกการเปลี่ยนแปลงที่เหมาะสมกับการทดลอง
    6. ดำเนินการทดลอง
    7. สรุปและวิเคราะห์ผลการทดลอง
    8. จัดทำบทความตีพิมพ์
    9. จัดทำรายงาน

## ผลงานที่ตีพิมพ์จากโครงงานมหาบัณฑิต

ส่วนหนึ่งของโครงงานมหาบัณฑิตนี้ได้รับการตอบรับเพื่อตีพิมพ์เป็นบทความวิจัยในหัวข้อเรื่อง “Measurement of Agility to Change for Service-Based Systems” โดย KongsawasPotipanom and TwittieSenivongse ในวารสารงานประชุมวิชาการซอฟต์แวร์และวิศวกรรมสารสนเทศนานาชาติปี 2555(The 2012 International Conference on Software and Information Engineering : ICSIE2012) ซึ่งจัดขึ้นโดยสถาบันอิเล็กทรอนิกส์สิงคโปร์ มหาวิทยาลัยพูตรามาเลเซีย และมหาวิทยาลัยเทคโนโลยีมารา ณ โรงแรมเฟิร์สโฮเตล กรุงเทพฯ ระหว่างวันที่ 12 –13พฤษภาคม2555

# [บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง](#สารบัญ)

## [ทฤษฎีที่เกี่ยวข้อง](#สารบัญ)

* + 1. **สถาปัตยกรรมเชิงบริการ**

สถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture: SOA) [1]เป็นแนวคิดที่ใช้สร้างแบบจำลองสถาปัตยกรรมสำหรับการวางโครงสร้างพื้นฐานของระบบซอฟต์แวร์ขององค์กร ที่มุ่งเน้นในการเพิ่มประสิทธิภาพการทำงาน ความคล่องตัว และผลิตภาพ บนพื้นฐานของเซอร์วิซที่สามารถใช้ร่วมกันได้และนำกลับมาใช้ใหม่ได้ ทั้งนี้เนื่องจากระบบเทคโนโลยีสารสนเทศขององค์กรต่าง ๆ มีการพัฒนาระบบด้วยเทคโนโลยีที่หลากหลาย ซึ่งทำให้ยากในการจะใช้งานระบบเหล่านั้นร่วมกัน ยากต่อการการบำรุงรักษา มีค่าใช้จ่ายที่สูง และมีความคล่องตัวน้อยในการที่จะปรับเปลี่ยนระบบให้สอดรับกับการเปลี่ยนแปลงทางธุรกิจ แนวคิดสถาปัตยกรรมเชิงบริการจึงได้ถูกนำมาใช้เพื่อแก้ปัญหาดังกล่าว

การออกแบบระบบด้วยแนวคิดสถาปัตยกรรมเชิงบริการนั้น สามารถแบ่งออกได้เป็น 4 ระดับชั้นดังนี้ (รูปที่ 2.1)

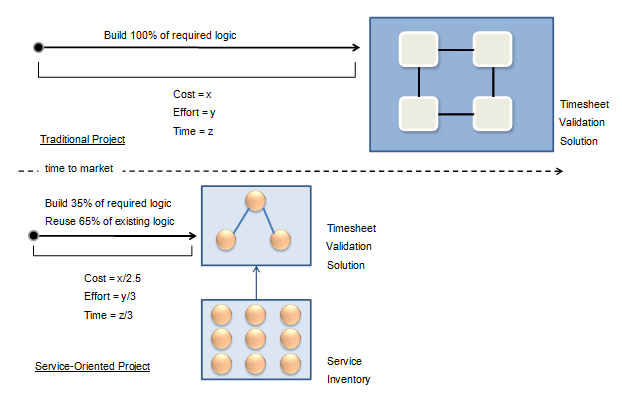
* + 1. ระดับชั้นกระบวนการทางธุรกิจ (Business Processes Layer) เป็นระดับชั้นที่ประกอบด้วยส่วนของเซอร์วิซตั้งแต่หนึ่งเซอร์วิซขึ้นไป ที่ถูกนำมาใช้ในการจัดการกระบวนทางธุรกิจ ในการแก้ไขปัญหา หรือดำเนินการอย่างใดอย่างหนึ่ง
    2. ระดับชั้นเซอร์วิซ (Services Layer) เป็นระดับชั้นที่ประกอบด้วยเซอร์วิซต่าง ๆ ที่สามารถนำกลับมาใช้ใหม่ได้ โดยเซอร์วิซเหล่านั้นถูกสร้างมาจากมอดูลต่าง ๆ ของโปรแกรมที่อยู่ระดับชั้นทรัพยากรการดำเนินงาน โดยมีส่วนต่อประสาน (Interface) ที่ประกาศไว้สำหรับให้ซอฟต์แวร์ส่วนอื่นเรียกใช้งาน
    3. ระดับชั้นส่วนประกอบวิสาหกิจ (Enterprise Components Layer) เป็นระดับชั้นของส่วนประกอบที่มีหน้าที่ในการจัดการตรรกะทางธุรกิจ และทำหน้าที่ต่อประสานกับระดับชั้นทรัพยากรการดำเนินงานในการดำเนินการของตรรกะนั้น ๆ
    4. ระดับชั้นทรัพยากรการดำเนินงาน (Operational Resources Layer) เป็นระดับชั้นที่ประกอบไปด้วยระบบสารสนเทศ โปรแกรม และฐานข้อมูล ที่ใช้ในองค์กร ซึ่งเซอร์วิซต่าง ๆ ได้ถูกสร้างจากส่วนประกอบเหล่านี้



รูปที่ 2.1 ระดับชั้นของสถาปัตยกรรมเชิงบริการ

* + 1. **ความคล่องตัว**

ความคล่องตัว (Agility) [2] เป็นประโยชน์อย่างหนึ่งที่องค์กรหรือหน่วยงานต่าง ๆ คาดหวังว่าจะได้รับจากการพัฒนาระบบด้วยแนวคิดสถาปัตยกรรมเชิงบริการ โดยความคล่องตัวจะเป็นสิ่งที่แสดงถึงความสามารถขององค์กรในการตอบสนองต่อการเปลี่ยนแปลงว่ามีประสิทธิภาพมากน้อยเพียงใด และเนื่องด้วยการพัฒนาระบบแบบสถาปัตยกรรมเชิงบริการเป็นการพัฒนาโดยการนำเซอร์วิซต่าง ๆ ที่องค์กรมีอยู่เดิมนำกลับมาใช้ใหม่ จึงสามารถช่วยลดระยะเวลา และความพยายามที่ใช้ในการพัฒนาลงได้ โดยไม่ต้องเริ่มทำใหม่ทั้งหมด ดังรูปที่ 2.2 แสดงตัวอย่างการพัฒนาระบบการลงเวลา (Timesheet) ซึ่งเห็นได้ชัดว่าการพัฒนาระบบแบบสถาปัตยกรรมเชิงบริการนั้นสามารถลดในส่วนที่จะต้องพัฒนาใหม่ลงเหลือเพียง 35 เปอร์เซ็นต์ และสามารถนำเซอร์วิซที่มีอยู่เดิมกลับมาใช้ใหม่ถึง 65 เปอร์เซ็นต์ อีกทั้งยังช่วยลดระยะเวลาในการส่งมอบลงประมาณ 50 เปอร์เซ็นต์เมื่อเทียบกับการพัฒนาระบบด้วยวิธีเดิม



รูปที่ 2.2 ตัวอย่างการพัฒนาด้วยสถาปัตยกรรมเชิงบริการเทียบกับการพัฒนาด้วยวิธีเดิม [2]

* + 1. **ค่าความซับซ้อนไซโคลมาติก**

ค่าความซับซ้อนไซโคลมาติก (Cyclomatic Complexity) [3] ถูกพัฒนาในปี คศ.1976 โดย Thomas J. McCabe ใช้ในการวัดความซับซ้อนของโปรแกรม ซึ่งเป็นตัวบ่งชี้ว่าโปรแกรมนั้นมีความยากต่อการบำรุงรักษา และการทดสอบมากน้อยเพียงใด โดยสามารถคำนวณได้จากสองวิธี

วิธีที่หนึ่ง คำนวณเส้นทางของกราฟการไหลของโปรแกรม (Flow Graph) คิดจากจำนวนเส้นเชื่อมต่อระหว่างโหนด (Edge) และจำนวนโหนด (Node) ของกราฟ ดังสมการ

**(1)**

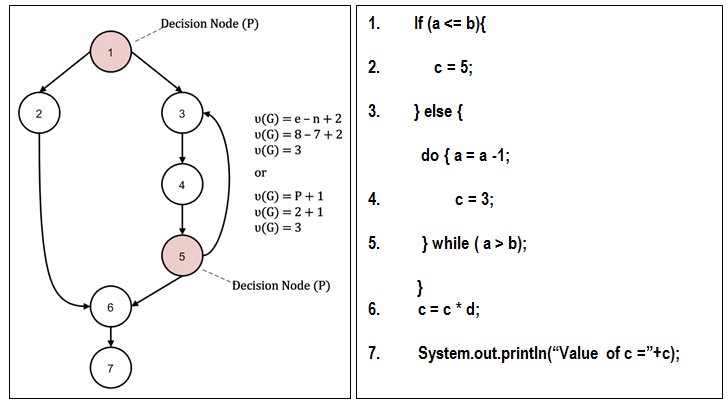
* ʋ(G) คือค่าความซับซ้อนไซโคลมาติก
* e คือจำนวนเส้นเชื่อมต่อระหว่างโหนด
* n คือจำนวนโหนด

วิธีที่สอง คำนวณจากจำนวนโหนดตัดสินใจ (Decision Node) ดังสมการ

**(2)**

* เมื่อ P คือจำนวนโหนดตัดสินใจ

ตัวอย่างกราฟการไหลของโปรแกรม และการคำนวณค่าความซับซ้อนของโปรแกรมโดยวิธีไซโคลมาติกแสดงดังรูปที่ 2.3



รูปที่ 2.3 ตัวอย่างกราฟการไหลของโปรแกรม และการคำนวณค่าความซับซ้อนไซโคลมาติก

## [งานวิจัยที่เกี่ยวข้อง](#สารบัญ)

จากการศึกษางานวิจัยที่เกี่ยวข้องกับการวัดความคล่องตัว สามารถแบ่งกลุ่มของงานวิจัยออกได้เป็น 2 กลุ่ม กลุ่มแรกเป็นกลุ่มงานวิจัยที่เกี่ยวข้องกับการวัดความคล่องตัวของซอฟต์แวร์ (Software Agility) โดยมีงานวิจัยชื่อ “Measuring the agility of a SOA approach” [4] ได้เสนอวิธีการวัดความคล่องตัวของระบบที่ออกแบบด้วยสถาปัตยกรรมเชิงบริการ ซึ่งมีการเสนอตัววัดความคล่องตัว (Agility Metric) ที่พิจารณาจากความสัมพันธ์ของความซับซ้อนของการเปลี่ยนแปลงระบบ (Change Complexity) เทียบกับเวลาที่ใช้ในการดำเนินการเปลี่ยนแปลงระบบ (Change Duration) โดยมองความซับซ้อนของการเปลี่ยนแปลงระบบเป็น 2 มิติในลักษณะของพื้นที่สี่เหลี่ยมของการเปลี่ยนแปลง ได้แก่ มิติของจำนวนระดับชั้นในเชิงสถาปัตยกรรมของซอฟต์แวร์ที่การเปลี่ยนแปลงนั้นไปกระทบ (Architectural Depth) และมิติความกว้างของกระบวนการทำงาน (Process Breadth) ที่ใช้ในการดำเนินการกับการเปลี่ยนแปลง ดังนั้นความซับซ้อนของการเปลี่ยนแปลงระบบจะเป็นผลคูณของค่าจากสองมิติดังกล่าว จากนั้นทำการวัดข้อมูลในการดำเนินการกับการเปลี่ยนแปลงของสองระบบ ระบบแรกเป็นระบบที่ออกแบบด้วยสถาปัตยกรรมเชิงบริการ ส่วนระบบที่สองเป็นระบบที่ไม่ได้ออกแบบด้วยสถาปัตยกรรมเชิงบริการ และทำการเปรียบเทียบค่าที่ได้จากการวัด โดยคาดหวังว่าค่าความคล่องตัวของระบบที่ออกแบบด้วยสถาปัตยกรรมเชิงบริการจะสูงกว่าระบบที่ไม่ได้ออกแบบด้วยสถาปัตยกรรมเชิงบริการ อีกหนึ่งงานวิจัยในกลุ่มนี้เป็นงานวิจัยชื่อว่า “A Practical Measurement for the Agility of Software Development Processes” [5] ได้เสนอวิธีการวัดความคล่องตัวของซอฟต์แวร์ด้วยการเทียบเคียงความคล่องตัวกับอัตราการเปลี่ยนแปลงของค่าเอ็นโทรปี โดยเอ็นโทรปีในงานวิจัยนี้จะหมายถึงความไม่แน่นอนของซอฟต์แวร์ ซึ่งพิจารณาจากล็อกการเปลี่ยนแปลง (Change Logs) ของซอฟต์แวร์ ณ ช่วงเวลาหนึ่ง หากค่าอัตราการเปลี่ยนแปลงของค่าเอ็นโทรปีออกมาสูง จะแสดงว่าซอฟต์แวร์นั้นรับการเปลี่ยนแปลงได้บ่อยจึงมีความคล่องตัวสูงด้วยเช่นกัน งานวิจัยดังกล่าวได้ทำการทดสอบกับโอเพนซอร์ซ (Open Source) ซอฟต์แวร์ 2 ระบบ ระบบแรกเป็นระบบที่ใช้วิธีอะไจล์ในการพัฒนา (Agile System) และระบบที่สองเป็นระบบที่ไม่ได้ใช้วิธีอะไจล์ในการพัฒนา (Non-Agile System)

กลุ่มที่สองเป็นกลุ่มงานวิจัยที่เกี่ยวข้องกับการวัดความคล่องตัวของกระบวนการทำงานและการดำเนินงาน (Process and Operation Agility) โดยมีงานวิจัยชื่อ “A measure of agility as the complexity of the enterprise system” [6] ได้เสนอแบบจำลองในการวัดความคล่องตัวของระบบวิสาหกิจ (Enterprise System) ด้วยแนวคิดที่ว่าหากระบบวิสาหกิจมีความซับซ้อนของกระบวนการทำงานต่ำ ระบบวิสาหกิจนั้นจะมีความคล่องตัวในการปรับเปลี่ยนกระบวนการทำงานที่สูงกว่าระบบวิสาหกิจที่มีความซับซ้อนของกระบวนการทำงานสูง โดยงานวิจัยได้ทำการเปรียบเทียบความซับซ้อนของกระบวนการทำงานของระบบการให้บริการด้านโทรคมนาคมที่ใช้อยู่ ณ ปัจจุบัน (As Is Process) กับกระบวนการทำงานที่จะทำการปรับรื้อใหม่ (Reengineering Process) การวัดความซับซ้อนจะพิจารณาจากอัตราการเปลี่ยนแปลงสถานะของกระบวนการทำงาน หากค่าอัตราการเปลี่ยนแปลงสถานะของกระบวนการทำงานของระบบออกมาสูงจะกล่าวได้ว่าระบบมีความซับซ้อนสูง ส่วนการวัดความคล่องตัวของกระบวนการทำงานจะพิจารณาจากต้นทุน กับเวลาที่ใช้ทั้งหมดในการเปลี่ยนสถานะของกระบวนการทำงานตั้งแต่เริ่มกระบวนการจนจบกระบวนการทำงาน หากต้นทุนและเวลาที่ใช้ทั้งหมดออกมาสูงแสดงว่าระบบนั้นมีความคล่องตัวต่ำ แนวคิดวิธีการของงานวิจัยนี้เป็นการวัดความคล่องตัวของกระบวนการทำงานของระบบซึ่งไม่สามารถนำมาใช้ในการวัดความคล่องตัวของซอฟต์แวร์ได้ แต่สำหรับแนวคิดในการพิจารณาความซับซ้อนเป็นแนวคิดที่เหมาะสมที่จะนำมาสะท้อนถึงความคล่องตัวได้ดี งานวิจัยต่อมาชื่อ “Agility Measurement Index – A Metric for the Crossroads of Software Development Methodologies” [7] ได้เสนอตัววัดที่เป็นดัชนีการวัดความคล่องตัว (Agility Measurement Index) เพื่อช่วยให้องค์กรสามารถเลือกแบบจำลองกระบวนการ (Process Model) ที่ดีที่สุด และเหมาะกับแต่ละโครงการพัฒนาซอฟต์แวร์ จากผลของงานวิจัยพบว่าหากค่าที่ได้จากการวัด มีค่าไม่สูงมากจะบ่งบอกว่าระบบที่จะทำการพัฒนานั้นค่อนข้างนิ่ง และมีความซับซ้อนในกระบวนการพัฒนาน้อย ซึ่งจะเหมาะกับแบบจำลองน้ำตก (Waterfall Model) แต่ถ้าหากค่าที่ได้ออกมาสูงจะเหมาะกับแบบยูนิไฟด์ (Unified Process) หรือแบบเอกซ์พี (Extreme Programming: XP) มากกว่า และจะต้องนำค่าน้ำหนักที่ผู้ทำการวัดกำหนดให้กับแต่ละมุมมองต่าง ๆ มาประกอบการพิจารณาด้วย ในกรณีที่ค่าน้ำหนักในมุมมองของการติดต่อกับผู้ใช้มีค่าที่สูง จะเหมาะกับแบบเอกซ์พี แต่หากค่าในมุมมองของระยะเวลาที่ใช้ และค่าของความเสี่ยงมีค่าที่สูง จะเหมาะกับแบบยูนิไฟด์มากกว่า ข้อสังเกตสำหรับงานวิจัยดังกล่าวคือ ค่าที่ใช้ในสูตรการคำนวณของตัววัด จะมาจากการให้ค่าน้ำหนักจากตัวผู้วัด ซึ่งต้องอาศัยประสบการณ์ในการพัฒนาซอฟต์แวร์มาพอสมควร การวัดทำในช่วงก่อนเริ่มโครงการ และไม่สามารถนำมาวัดความคล่องตัวของซอฟต์แวร์ได้ เนื่องจากเป็นการวัดความคล่องตัวของแบบจำลองกระบวนการทำงาน อีกงานวิจัยหนึ่งในกลุ่มนี้เป็นงานวิจัยชื่อว่า “Evaluating Agility in Extended Enterprise Systems: A Transportation Network case” [8] ซึ่งได้เสนอตัววัดที่เป็นค่าดัชนีความคล่องตัวรวม (Total Agility Index) เพื่อใช้ในการประเมินความสามารถของระบบวิสาหกิจในการปรับตัวเข้ากับการเปลี่ยนแปลงที่ไม่คาดคิดของสิ่งแวดล้อม โดยนำตัววัดไปใช้กับข้อมูลระบบการขนส่งมวลชนของเมืองนิวยอร์กในสถานการณ์ต่าง ๆ เช่น วันที่มีการแข่งขันกีฬา วันที่มีการเดินขบวนพาเหรด เป็นต้น เมื่อผลที่ได้จากการวัดออกมามีค่าต่ำ แสดงว่าระบบมีความคล่องตัวสูงในการปรับเข้ากับสถานการณ์ แต่หากผลที่ได้จากการวัดมีค่าสูงจะกล่าวได้ว่าระบบมีความคล่องตัวน้อยในการที่จะปรับตัวเข้ากับสถานการณ์นั้น ๆ ความคล่องตัวของงานวิจัยนี้จะเป็นลักษณะความคล่องตัวของการดำเนินงาน (Operation Agility) ดังนั้นจึงไม่สามารถนำมาใช้ในการวัดความคล่องตัวของซอฟต์แวร์ได้

จากงานวิจัยที่ได้กล่าวมาข้างต้น ผู้วิจัยได้เลือกให้ความสนใจกับงานวิจัยในกลุ่มที่เกี่ยวข้องกับการวัดความคล่องตัวของซอฟต์แวร์ และได้เลือกบทความ [4]ในการอ้างอิงสำหรับโครงงานนี้ เนื่องจากมีการเสนอวิธีการ และตัววัดที่สามารถทำความเข้าใจ และนำไปประยุกต์ใช้ได้ง่าย ซึ่งตัววัดที่เสนอมาสามารถประยุกต์ใช้ได้กับทุกระบบเทคโนโลยีสารสนเทศ ส่วนอีกงานวิจัยหนึ่งในกลุ่มที่ได้ทำการวัดความคล่องตัวของซอฟต์แวร์จากอัตราการเปลี่ยนแปลงของค่าเอนโทรปี มีลักษณะเป็นการพิจารณาความถี่ของการเกิดล็อกการเปลี่ยนแปลง ณ ช่วงเวลาหนึ่ง ผู้วิจัยเห็นว่าการเกิดล็อกการเปลี่ยนแปลงจำนวนมาก ไม่สามารถนำไปเป็นตัวบ่งชี้ว่าซอฟต์แวร์มีความคล่องตัวสูง เพราะการเปลี่ยนแปลงอาจจะเกิดจากการที่ซอฟต์แวร์ พัฒนาไม่ตรงตามความต้องการของผู้ใช้ หรือมีข้อบกพร่องสูง จึงจำเป็นต้องมีการร้องขอให้ทำการแก้ไข และเปลี่ยนแปลงระบบ

# 

# [บทที่ 3 แบบ](#สารบัญ)จำลองการวัดความคล่องตัว

ในบทนี้จะกล่าวถึงแบบจำลองที่ใช้ในการวัดความคล่องตัว โดยจากการศึกษางานวิจัยที่เกี่ยวข้อง ผู้วิจัยได้เลือกที่จะนำตัววัดความคล่องตัวของซอฟต์แวร์ที่เสนอในบทความ [4] มาใช้โดยพิจารณาจากความสัมพันธ์ของความซับซ้อนของการเปลี่ยนแปลงระบบ เทียบกับเวลาที่ใช้ในการดำเนินการเปลี่ยนแปลงระบบ ดังสมการที่ 3

**Agility = Change Complexity / Change Duration** **(3)**

ในส่วนความซับซ้อนของการเปลี่ยนแปลงซึ่งมีการพิจารณาพื้นที่ของการเปลี่ยนแปลง (Area of Change) ในสองมิติ ได้แก่ มิติของจำนวนระดับชั้นในเชิงสถาปัตยกรรมของซอฟต์แวร์ที่การเปลี่ยนแปลงนั้นไปกระทบ และมิติความกว้างของกระบวนการทำงานที่ใช้ในการดำเนินการกับการเปลี่ยนแปลง แสดงได้ดังรูปที่ 3.1 และดังสมการที่ 4 [4]



รูปที่ 3.1 มิติความซับซ้อนของการเปลี่ยนแปลง (ดั้งเดิม)

**Change Complexity = Architectural Depth \* Process Breadth (4)**

จากรูปที่ 3.1 และสมการที่ 4 ผู้วิจัยเห็นด้วยกับแนวคิดในการนำความซับซ้อนของการเปลี่ยนแปลงมาอธิบายถึงความคล่องตัวซึ่งคล้ายกับงานวิจัย [6] ที่นำความซับซ้อนของกระบวนการทำงานของระบบวิสาหกิจมาอธิบายความคล่องตัวในการปรับเปลี่ยนกระบวนการทำงาน แต่ถ้าหากนำมาสมการที่ 4 มาอธิบายถึงความคล่องตัวของซอฟต์แวร์นั้นอาจจะไม่ชัดเจนนักเนื่องจากสมการเป็นการอธิบายในลักษณะของกระบวนการทำงาน ผู้วิจัยเห็นว่าควรปรับการมองเป็นสามมิติในลักษณะของปริมาตรของการเปลี่ยนแปลง (Volume of Change) ด้วยการเพิ่มมิติในส่วนของการที่คำขอเปลี่ยนแปลงนั้นไปกระทบกับโค้ดของระบบ ซึ่งจะมีความซับซ้อนแตกต่างกันออกไปในแต่ละมอดูลของโปรแกรม และความซับซ้อนของโค้ดนี้มีผลต่อความยากง่ายในการดำเนินการกับการเปลี่ยนแปลงด้วยเช่นกัน ดังรูปที่ 3.2 โดยสามารถแสดงการปรับสมการของความซับซ้อนของการเปลี่ยนแปลงได้ดังสมการที่ 5



รูปที่ 3.2 มิติความซับซ้อนของการเปลี่ยนแปลง (ใหม่)

**Change Complexity =** *Architectural Depth \* Process Breadth \** **Code Complexity (5)**

* + 1. ค่าระดับชั้นในเชิงสถาปัตยกรรมของซอฟต์แวร์ที่การเปลี่ยนแปลงนั้นไปกระทบ (Architectural Depth) ได้มาจากการนับจำนวนของระดับชั้นในเชิงสถาปัตยกรรมของซอฟต์แวร์ที่การเปลี่ยนแปลงนั้นไปกระทบ หรือทำให้ต้องมีการเปลี่ยนแปลงแก้ไขในระดับชั้นนั้น ๆ ซึ่งระดับชั้นในที่นี้อาจจะแบ่งเป็น ระดับชั้นประสบการณ์ของผู้ใช้ (User Experience) ระดับชั้นกระบวนการเชิงธุรกิจ (Business Process) และระดับชั้นของการจัดเก็บข้อมูลสารสนเทศ (Information Storage) เป็นต้น
    2. ค่าความกว้างของกระบวนการทำงาน (Process Breadth) ได้มาจากการนับจำนวนตำแหน่ง (Number of Roles) ในองค์กรที่ได้มีการมอบหมายให้ดำเนินการกับการเปลี่ยนแปลง เช่น ตำแหน่งนักวิเคราะห์ระบบ ตำแหน่งผู้พัฒนาโปรแกรม และตำแหน่งผู้ทดสอบระบบ เป็นต้น รวมกับจำนวนการส่งต่องานระหว่างตำแหน่งงาน (Number of Handoffs between Roles) ยกตัวอย่าง เช่น นักวิเคราะห์ระบบส่งงานเอกสารการออกแบบระบบให้กับผู้พัฒนาโปรแกรมเพื่อใช้ในการพัฒนาระบบ หรือผู้พัฒนาโปรแกรมส่งโปรแกรมที่พัฒนาเสร็จให้กับผู้ทดสอบระบบทำการทดสอบระบบว่าเป็นไปตามข้อกำหนดความต้องการหรือไม่ เป็นต้น ซึ่งแสดงได้ดังสมการที่ 6

**Process Breadth = No.of Roles + No.of Handoffs between Roles (6)**

* + 1. ค่าความซับซ้อนของโค้ด (Code Complexity) สามารถคำนวณได้จากค่าความซับซ้อนไซโคลมาติกของโค้ดในแต่ละมอดูล ที่การเปลี่ยนแปลงนั้นไปกระทบ

# [บทที่ 4](#สารบัญ) วิธีการดำเนินงาน

ในบทนี้จะกล่าวถึงวิธีการดำเนินงานเพื่อเตรียมการทดลองซึ่งประกอบด้วย การคัดเลือกหน่วยทดลอง การเลือกรายการเปลี่ยนแปลง และการรวบรวมข้อมูลจากการทดลอง

## การคัดเลือกหน่วยทดลอง

เนื่องจากโครงงานนี้ต้องการทำการทดลองเพื่อเปรียบเทียบความคล่องตัวของระบบอิงบริการ กับระบบที่ไม่อิงบริการ ในการตอบสนองต่อการเปลี่ยนแปลง ดังนั้นจึงจำเป็นที่จะต้องเลือกหน่วยทดลองให้เหมาะสมเพื่อเป็นตัวแทนของทั้งสองระบบดังนี้

* + 1. ระบบที่ไม่อิงบริการ (Non Service-Based System)

ผู้วิจัยได้เลือกหน่วยทดลองที่เป็นระบบบันทึกการลา (E-Leave) ของบริษัทพัฒนาซอฟต์แวร์แห่งหนึ่ง เพื่อเป็นตัวแทนของระบบที่ไม่อิงบริการ โดยมีรายละเอียดของระบบดังรูปที่4.1



รูปที่ 4.1 สถาปัตยกรรมของระบบบันทึกการลา

จากรูปที่ 4.1 ซึ่งแสดงสถาปัตยกรรมของระบบบันทึกการลา แบ่งออกเป็น 4 ระดับชั้นได้แก่

1. ระดับชั้นการนำเสนอ (Presentation Layer) เป็นส่วนของการนำเสนอข้อมูลสารสนเทศแก่ผู้ใช้ และทำการแปลผลการตอบสนองของผู้ใช้ องค์ประกอบต่าง ๆ ในส่วนนี้ประกอบด้วย ฟอร์มของหน้าจอต่าง ๆ ที่เป็นเจเอสพี (Java Server Page: JSP) ซึ่งเป็นเทคโนโลยีหนึ่งของภาษาจาวา และคำสั่ง (Script)
2. ระดับชั้นแอปพลิเคชัน (Application Layer) เป็นส่วนของโปรแกรมที่ทำหน้าที่ในการควบคุมการทำงานของหน้าจอโปรแกรมต่าง ๆ ในส่วนการนำเสนอ
3. ระดับชั้นโดเมน (Domain Layer) เป็นส่วนของโปรแกรมที่เกี่ยวกับเงื่อนไขทางธุรกิจที่เกี่ยวกับการลา เช่น เงื่อนไขการคำนวณวันลาที่ใช้ไปของพนักงาน (Leave Consumed) เงื่อนไขการอนุมัติการลา (Leave Approval) เป็นต้น
4. ระดับชั้นการเข้าถึงข้อมูล (Data Access Layer) จะเป็นส่วนติดต่อระหว่างโปรแกรมกับฐานข้อมูลของระบบบันทึกการลา ซึ่งมีฐานข้อมูลชื่อ “EApps” โดยมีกรอบงานไฮเบอร์เนต (Hibernate Framework) ที่ช่วยจัดการโปรแกรมที่อยู่ในรูปของอ็อบเจกต์ ให้สามารถใช้งานกับระบบฐานข้อมูลได้อย่างมีประสิทธิภาพ

รูปที่ 4.2 แสดงแผนภาพดีพลอยเมนต์ของระบบบันทึกการลาโดยมีส่วนประกอบต่างๆได้ดังนี้

1. เครื่องลูกข่าย (Client) คือ เครื่องคอมพิวเตอร์ของผู้ใช้งาน โดยการใช้งานจะมีการติดต่อผ่านโพรโตคอลเอชทีทีพี (HTTP) และผู้ใช้สามารถติดต่อระบบผ่านทางเว็บเบราว์เซอร์(Web Browser) เพื่อเข้าใช้ระบบบันทึกการลา
2. ระบบปฏิบัติการ (OS) ที่ใช้สำหรับติดตั้งระบบบันทึกการลาใช้ระบบปฏิบัติการวินโดวส์
3. เจเอสพีเซิร์ฟเวอร์ (JSPServer) คือแอปพลิเคชันเซิร์ฟเวอร์สำหรับติดตั้งระบบบันทึกการลาซึ่งใช้โปรแกรมทอมแคต (Tomcat) รุ่น 6.0.29
4. คาตาลินาเซิร์ฟเล็ทคอนเทนเนอร์ (Catalina Servlet Container) เป็นส่วนที่ใช้ในการประมวลผลระบบบันทึกการลาที่เป็นลักษณะเว็บแอปพลิเคชันที่พัฒนาขึ้นโดยกรอบงานจาวาเซิร์ฟเวอร์เฟซ (Java Server Face Framework: JSF) และกรอบงานไฮเบอร์เนต (HibernateFramework)
5. ระบบจัดการฐานข้อมูล (Database System) ใช้เป็นมายซีเควล(MySQL) รุ่น 5.1 โดยมีฐานข้อมูลชื่อ อีแอปส์(EApps)



รูปที่ 4.2 แผนภาพดีพลอยเมนต์ของระบบบันทึกการลา

* + 1. ระบบอิงบริการ (Service-Based System)

สำหรับระบบอิงบริการที่เลือกมาเป็นหน่วยทดลอง ผู้วิจัยได้จัดทำขึ้นจากระบบบันทึกการลาที่ได้กล่าวมาในข้างต้น โดยทำการเพิ่มระดับชั้นในเชิงสถาปัตยกรรมขึ้นมาอีกหนึ่งระดับคือ ระดับชั้นเซอร์วิซ (Service Layer) การพิจารณาถึงเซอร์วิซที่จำเป็นสำหรับระบบบันทึกการลาว่าควรจะมีเซอร์วิซอะไรบ้างนั้น ผู้วิจัยได้ใช้แนวทางการออกแบบเซอร์วิซที่คำนึงถึงเอนทิตีข้อมูลเป็นหลัก (Entity-centric Business Service Design)[9] ที่เน้นการพิจารณาเซอร์วิซเอนทิตี (Entity Services) ซึ่งเป็นเซอร์วิซที่เกี่ยวข้องกับเอนทิตีของข้อมูลในระบบ รวมไปถึงเซอร์วิซอรรถประโยชน์ (Utility Services) ที่มีแนวโน้มของการนำกลับมาใช้ใหม่สูง ในส่วนของเซอร์วิซเอนทิตีนั้นได้พิจารณาจากเอนทิตีต่าง ๆ ของระบบบันทึกการลา ดังรูปที่ 4.3 โดยได้จัดทำออกมาเป็น 2 เซอร์วิซ ได้แก่ เซอร์วิซพนักงาน (EmployeeService) และเซอร์วิซการลา (EleaveService) ในส่วนของเซอร์วิซอรรถประโยชน์นั้น จะเป็นเซอร์วิซสำหรับการแจ้งเตือน (NotificationService) ที่ใช้ในการส่งอีเมลแก่พนักงานผู้ที่ทำการลา และผู้ที่ทำการอนุมัติการลา การออกแบบเซอร์วิซแสดงได้ดังรูปที่ 4.4ในส่วนของสถาปัตยกรรมของระบบบันทึกการลาแบบอิงบริการสามารถแสดงได้ดังรูปที่ 4.5โดยเซอร์วิซทั้งสามจะอยู่ในระดับชั้นเซอร์วิซ และถูกเรียกใช้โดยส่วนของโปรแกรมในระดับชั้นโดเมน และมีการเรียกใช้ส่วนของโปรแกรมในระดับชั้นการเข้าถึงข้อมูล



รูปที่ 4.3 แผนภาพความสัมพันธ์ของคลาสของระบบบันทึกการลา



รูปที่ 4.4 การออกแบบเซอร์วิซของระบบบันทึกการลา



รูปที่ 4.5 สถาปัตยกรรมของระบบบันทึกการลาแบบอิงบริการ

รูปที่ 4.6 แสดงแผนภาพดีพลอยเมนต์ของระบบบันทึกการลาที่เป็นแบบอิงบริการ โดยสิ่งที่แตกต่างจากรูปที่ 4.2 คือมีการเพิ่มคอมโพเนนต์ของอาปาเชเอซิส2 (Apache Axis2) [10] ซึ่งเป็นเว็บเซอร์วิซคอนเทนเนอร์ที่ใช้สำหรับติดตั้งเซอร์วิซเอนทิตี และเซอร์วิซอรรถประโยชน์ที่สร้างขึ้น



รูปที่ 4.6 แผนภาพดีพลอยเมนต์ของระบบบันทึกการลาแบบอิงบริการ

## การเลือกรายการการเปลี่ยนแปลง

การเปลี่ยนแปลงที่เลือกใช้ในการทดลองกับระบบอิงบริการและไม่อิงบริการ ผู้วิจัยได้เลือกมาสองรายการ ซึ่งเป็นรายการที่เกี่ยวข้องในการขอเปลี่ยนแปลงระบบบันทึกการลา โดยมีรายละเอียดของการเปลี่ยนแปลงในแต่ละรายการดังนี้

1. รายการที่ 1 เนื่องจากทุก ๆ ช่วงสิ้นปีของบริษัท ฝ่ายทรัพยากรบุคคลจะทำการร้องขอให้ผู้ดูแลระบบบันทึกการลาดำเนินการสรุปวันลาคงเหลือของพนักงานแต่ละคนเป็นแฟ้มข้อมูลไมโครซอฟท์เอกซ์เซล (Microsoft Excel) เพื่อนำไปคำนวณวันลาที่จะยกยอดไปรวมกับวันลาในปีถัดไปของพนักงานแต่ละคน จากนั้นจึงส่งแฟ้มข้อมูลนั้นกลับไปให้กับทางผู้ดูแลระบบช่วยนำข้อมูลการยกยอดวันลาของพนักงานเข้าระบบ ฝ่ายทรัพยากรบุคคลจึงมีความต้องการที่จะให้ระบบช่วยทำการยกยอดวันลาคงเหลือของพนักงานแต่ละคนให้โดยอัตโนมัติ ซึ่งเงื่อนไขในการคิดยกยอดวันลานั้น จะคิดจากวันลาคงเหลือ (Leave Balance) ของพนักงาน เทียบกับสิทธิ์การลาที่พนักงานสามารถลาได้ในปีนั้น (Leave Entitlement) โดยจะสามารถยกยอดวันลาได้สูงสุดคือครึ่งหนึ่งของสิทธิ์การลาในปีนั้น แต่ถ้าหากวันลาคงเหลือของพนักงานน้อยกว่าครึ่งหนึ่งของสิทธิ์การลาก็จะยกยอดไปเท่ากับจำนวนวันลาที่คงเหลือนั้น ยกตัวอย่าง ดังนี้

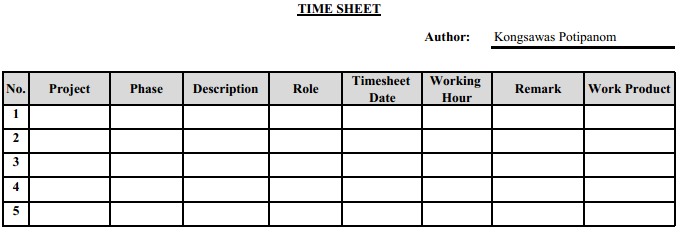
* กรณีที่ 1 พนักงานชื่อนาย ก
* มีวันลาคงเหลือเท่ากับ10 วัน
* มีสิทธิ์การลาในปีนั้นเท่ากับ 14 วัน
* ดังนั้นจำนวนวันลาที่สามารถยกยอดได้เท่ากับ 7 วัน
* กรณีที่ 2 พนักงานชื่อนาย ข
* มีวันลาคงเหลือเท่ากับ 5 วัน
* มีสิทธิ์การลาในปีนั้นเท่ากับ 14 วัน
* ดังนั้นจำนวนวันลาที่สามารถยกยอดได้เท่ากับ 5 วัน

1. รายการที่ 2 ระบบบันทึกการลาในปัจจุบันจะรองรับกับพนักงานที่ทำงานวันจันทร์ถึงวันศุกร์เท่านั้น แต่เนื่องจากมีแผนกหนึ่งซึ่งมีลักษณะการทำงานเป็นกะ (Shift) โดยพนักงานในแผนกนี้จะทำงานตามตารางที่หัวหน้าแผนกได้จัดไว้ ซึ่งจะต้องมีการเข้ามาทำงานในวันหยุดของสัปดาห์คือวันเสาร์และอาทิตย์ด้วย และถ้าพนักงานคนใดมีความจำเป็นต้องลางานในวันดังกล่าว จะไม่สามารถทำการบันทึกการลาของตนในระบบได้ เนื่องจากระบบจะไม่นับรวมวันหยุดของสัปดาห์มาคิดเป็นวันลาที่พนักงานได้ลาไป ดั้งนั้นฝ่ายทรัพยากรบุคคลจึงมีความต้องการที่จะให้ระบบทำการรองรับการทำงานของพนักงานที่ทำงานเป็นกะ

## การรวบรวมข้อมูลจากการทดลอง

ในการคำนวณหาค่าความคล่องตัวจากการทดลองนั้นต้องอาศัยข้อมูลที่ถูกรวบรวมในระหว่างทำการทดลอง โดยเครื่องมือที่ใช้ในการบันทึกข้อมูลจากการทดลองที่สำคัญมีดังนี้

1. ตารางบันทึกเวลาการทำงาน (Timesheet) นำมาใช้เพื่อบันทึกรายละเอียดการทำงานในช่วงดำเนินการกับการเปลี่ยนแปลง ข้อมูลที่บันทึกประกอบด้วย ลำดับที่ ชื่อโครงการ งานที่ทำ คำอธิบายรายละเอียดของงาน ตำแหน่งในการทำงานวันที่บันทึกการทำงาน จำนวนชั่วโมงการทำงาน หมายเหตุ และผลผลิตหรือชิ้นงานที่ได้จากการทำงาน แสดงตัวอย่างดังรูปที่ 4.7



รูปที่ 4.7 ตัวอย่างตารางบันทึกเวลาการทำงาน

1. ตารางบันทึกผลกระทบจากการเปลี่ยนแปลงใช้สำหรับบันทึกแฟ้มข้อมูล และมอดูลที่อยู่ในระดับชั้นต่าง ๆ ที่ถูกผลกระทบจากการเปลี่ยนแปลง แสดงตัวอย่างดังตารางที่ 4.1 และตารางที่ 4.2

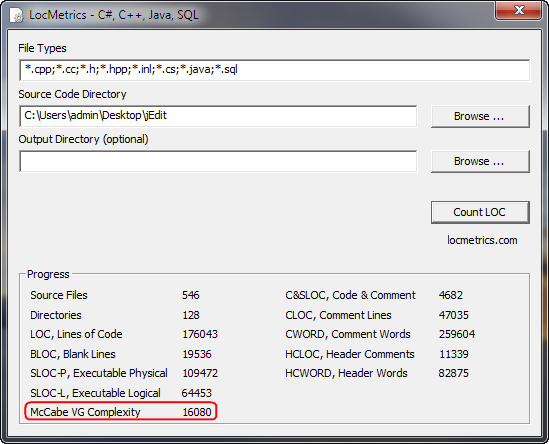
ตารางที่ 4.1 ตัวอย่างตารางบันทึกผลกระทบจากการเปลี่ยนแปลง (Non Service)

|  |  |
| --- | --- |
| **Layer**  **(Non-Service)** | **Affected JSP Files/Packages/Java Classes** |
| **Presentation** |  |
| **Application** |  |
| **Domain** |  |
| **Data Access** |  |

ตารางที่ 4.2 ตัวอย่างตารางบันทึกผลกระทบจากการเปลี่ยนแปลง (Service)

| **Layer**  **(Service)** | **Affected JSP Files/Packages/**  **Java Classes/Services** |
| --- | --- |
| **Presentation** |  |
| **Application** |  |
| **Domain** |  |
| **Service** |  |
| **Data Access** |  |

1. โปรแกรมลอคเมตริกส์(LocMetrics)[11]เป็นเครื่องมือที่ใช้ในการวัดค่าความซับซ้อนของโค้ด (McCabe VG complexity) ของมอดูลที่ได้รับผลกระทบจากการเปลี่ยนแปลงส่วนต่อประสานผู้ใช้งานของโปรแกรมลอคเมตริกส์เป็นดังรูปที่ 4.8



รูปที่ 4.8 โปรแกรมลอคเมตริกส์

## [สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา](#สารบัญ)

สภาพแวดล้อม และเครื่องมือที่ใช้ในการทดลอง มีดังต่อไปนี้

ฮาร์ดแวร์

* + 1. หน่วยประมวลผลอินเทลคอร์ 2 ดูโอ 2.00 กิกะเฮิร์ต

(CPU Intel Core 2 Duo 2.20GHz)

* + 1. หน่วยความจำ 4 กิกะไบต์ (4 GB RAM)
    2. ฮาร์ดดิสก์ความจุ 500 กิกะไบต์ (500 GB Hard Disk)

ซอฟต์แวร์

* + 1. ระบบปฏิบัติการไมโครซอฟท์วินโดวส์เซเวนโปรเฟสชันนอล (Microsoft Windows 7 Professional)
    2. เครื่องมือพัฒนาโปรแกรมเน็ตบีนส์รุ่น 6.5.1 (NetBeans 6.5.1)
    3. โปรแกรมสภาพแวดล้อมประมวลผลจาวา 6 (Java Runtime Environment 6)
    4. โปรแกรมชุดเครื่องมือพัฒนาจาวา 6 (Java Development Kit 6)
    5. ระบบจัดการฐานข้อมูลมายซีเควล รุ่น 5.1 (MySQL 5.1)
    6. โปรแกรมแอปพลิเคชันเซิร์ฟเวอร์ทอมแคต(Tomcat) รุ่น 6.0.29(Apache Tomcat 6.0.29)
    7. โปรแกรมเว็บเซอร์วิซคอนเทนเนอร์อาปาเชอาซิส2(Apache Axis 2)

# [บทที่ 5](#สารบัญ) การทดลองและการวิเคราะห์ผลการทดลอง

ในบทนี้จะกล่าวถึงการดำเนินการทดลองและการวิเคราะห์ผลการทดลองซึ่งประกอบด้วย การดำเนินการทดลอง ผลการทดลอง การคำนวณค่าความคล่องตัว และการวิเคราะห์ผลการทดลอง

## การดำเนินการทดลอง

การทดลองจะทำโดยนำการเปลี่ยนแปลงสองรายการที่เลือกมาดำเนินการกับแต่ละระบบ กำหนดให้ช่วงของการดำเนินงานแบ่งออกเป็นสามช่วงประกอบด้วย ช่วงการวิเคราะห์ออกแบบ ช่วงการพัฒนาระบบ และช่วงการทดสอบระบบ โดยสมมติให้ผู้วิจัยทำหน้าที่ในสามตำแหน่งได้แก่ นักวิเคราะห์ระบบ ผู้พัฒนาโปรแกรม และผู้ทดสอบระบบ ทุก ๆ กิจกรรมของการดำเนินงานจะมีการบันทึกเวลาการทำงาน (Timesheet) ทุกครั้ง โดยในการดำเนินการของแต่ละระบบกับการเปลี่ยนแปลงแต่ละรายการมีรายละเอียดการดำเนินงาน ดังนี้

* + 1. การเปลี่ยนแปลงรายการที่ 1

เป็นการเปลี่ยนแปลงที่ต้องการให้ระบบทำการคำนวณวันลาคงเหลือในปีที่ผ่านมาจะยกยอดไปในปีถัดไปให้โดยอัตโนมัติ กิจกรรมของการดำเนินงานมีดังนี้

* + - 1. การวิเคราะห์และออกแบบระบบ

แผนภาพที่ผู้วิจัยได้นำมาใช้ในการช่วงของการวิเคราะห์และออกแบบระบบได้แก่ แผนภาพยูสเคส แผนภาพกิจกรรม แผนภาพความสัมพันธ์ของคลาส และแผนภาพลำดับการทำงานของระบบ รายละเอียดของการออกแบบของแต่ละแผนภาพมีดังนี้

* + - * 1. แผนภาพยูสเคสของการเปลี่ยนแปลงรายการที่ 1กำหนดให้แอคเตอร์ (Actor) คือ พนักงานผู้ที่มีตำแหน่งเป็นผู้ดูแลระบบฝ่ายบุคคล (HR Admin) กำหนดยูสเคสของการเปลี่ยนแปลงนี้ คือ การยกยอดวันลา (Brought Forward) ในส่วนของแผนภาพยูสเคสนี้ของระบบที่ไม่อิงบริการ และระบบที่อิงบริการจะมีการออกแบบที่เหมือนกัน สามารถแสดงดังรูปที่ 5.1 และแสดงตารางคำอธิบายยูสเคสได้ดังตารางที่ 5.1



รูปที่ 5.1 แผนภาพยูสเคสของการเปลี่ยนแปลงรายการที่ 1

ตารางที่ 5.1 คำอธิบายยูสเคสการยกยอดวันลา

|  |  |  |
| --- | --- | --- |
| **ชื่อยูสเคส**: Brought Forward | **รหัส**: UC01 | **ระดับความสำคัญ**: สูง |
| **ผู้กระทำหลัก**: HR Admin | **ชนิดยูสเคส**: รายละเอียดและสาระสำคัญ | |
| **ผู้มีส่วนเกี่ยวข้อง:**   * HR Admin – ทำการยกยอดวันลาคงเหลือของพนักงาน | | |
| **รายละเอียดยูสเคส:**  ยูสเคสนี้อธิบายถึงขั้นตอนที่ HRAdmin ทำการยกยอดวันลาคงเหลือของพนักงานในปีที่ผ่านมาไปยังปีปัจจุบัน | | |
| **สิ่งกระตุ้น:**  HR Admin ต้องการยกยอดวันลาคงเหลือของพนักงาน | | |
| **ยูสเคสที่เกี่ยวข้อง:** - | | |
| **ภาวะก่อนทำงาน:**  พนักงานทุกคนได้บันทึกการลาของปีที่ผ่านมาครบถ้วนแล้ว | | |
| **ภาวะหลังทำงาน:**  ข้อมูลจำนวนวันลาที่ยกยอดมาถูกแสดงในหน้าบันทึกการลาของพนักงานแต่ละคน | | |
| **สายงานปกติ**:   1. ล็อกอินเข้าสู่ระบบ 2. ระบบแสดงเมนูส่วน HR Admin 3. เลือกเมนู “Year-End Brought Forward” | | |

ตารางที่ 5.1 คำอธิบายยูสเคสการยกยอดวันลา (ต่อ)

|  |  |  |
| --- | --- | --- |
| **ชื่อยูสเคส**: Brought Forward | **รหัส**: UC01 | **ระดับความสำคัญ**: สูง |
| **ผู้กระทำหลัก**: HR Admin | **ชนิดยูสเคส**: รายละเอียดและสาระสำคัญ | |
| **สายงานปกติ:**   1. ระบบแสดงข้อมูล รหัสพนักงาน ชื่อพนักงาน จำนวนวันลาทั้งหมดในปีปัจจุบัน สิทธิการลาของพนักงาน และวันลาที่ยกยอดมาจากปีที่ผ่านมาของพนักงานทั้งหมด 2. HR Admin ตรวจสอบความถูกต้องของวันลาที่ยกยอดมา 3. HR Admin กดปุ่ม “Update All” เพื่อบันทึกข้อมูลวันลาคงเหลือที่ยกยอดมาของพนักงาน 4. ระบบแจ้งการบันทึกข้อมูลเสร็จเรียบร้อย | | |
| **สายงานย่อย:**   * 1. HR Admin แก้ไขจำนวนวันลาคงเหลือหากพบว่าวันของพนักงานที่ยอดมาไม่ถูกต้องอันเนื่องมาจากที่ผ่านมาพนักงานคนนั้นมีการบันทึกการลาที่ไม่ถูกต้อง เช่น มีการกรอกวันลาเกิน หรือขาดจากจำนวนวันที่ได้ลาไปจริง ๆ | | |
| **สายงานทางเลือก / สายงานพิเศษ:** | | |

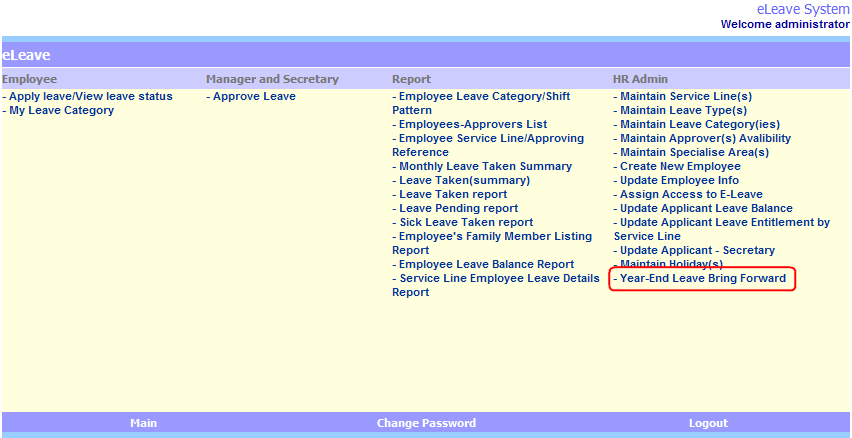
* + - * 1. แผนภาพกิจกรรมของการเปลี่ยนแปลงรายการที่ 1 เป็นแผนภาพที่แสดงกิจกรรมต่าง ๆ ในการยกยอดวันลาคงเหลือของพนักงาน สามารถแสดงแผนภาพกิจกรรมของระบบที่ไม่อิงบริการ และแผนภาพกิจกรรมของระบบอิงบริการในภาคผนวก ก
        2. แผนภาพความสัมพันธ์ของคลาสของการเปลี่ยนแปลงรายการที่ 1 ประกอบด้วยคลาสที่เกี่ยวข้องกับการเปลี่ยนแปลงนี้ 3 คลาส ได้แก่ คลาสพนักงาน (Employee) คลาสหมวดหมู่การลา (Leave Category) และคลาสการปรับยอดสิทธิ์การลา (Leave Adjustment) สามารถแสดงแผนภาพความสัมพันธ์ดังรูปที่ 5.2
        3. แผนภาพลำดับของการเปลี่ยนแปลงรายการที่ 1 จะแสดงลำดับขั้นตอนในการคำนวณวันลาคงเหลือที่จะยกยอดของพนักงาน โดยแสดงแผนภาพลำดับการทำงานของระบบไม่อิงบริการ และแผนภาพลำดับการทำงานของระบบอิงบริการในภาคผนวก ก



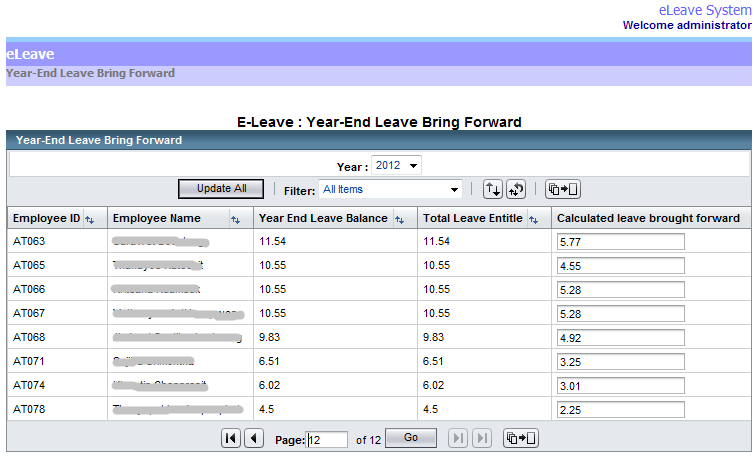
รูปที่ 5.2 แผนภาพความสัมพันธ์ของคลาสของการเปลี่ยนแปลงรายการที่ 1

* + - 1. การพัฒนาระบบ

ในส่วนนี้จะแสดงหน้าจอในการใช้งานโปรแกรมของการยกยอดวันลา โดย HR Admin เมื่อล็อกอินเข้าสู่ระบบแล้วจะทำการเลือกเมนู Year-End Leave Bring Forward ดังรูปที่ 5.3 จากนั้นระบบจะแสดงหน้าจอการยกยอดวันลาคงเหลือของพนักงานดังรูปที่ 5.4 โดยหน้าจอจะแสดงข้อมูล รหัสพนักงาน ชื่อพนักงาน สิทธิ์การลาคงเหลือของพนักงานในปีปัจจุบัน สิทธิ์การลาของพนักงานในปีนั้น และจำนวนวันลาที่ยกยอดมาจากปีที่ผ่านมา



รูปที่ 5.3 หน้าจอเมนูส่วนการยกยอดวันลา



รูปที่ 5.4 หน้าจอการยกยอดวันลา

* + - 1. การทดสอบระบบ

การทดสอบระบบจะเป็นการทดสอบตามฟังก์ชันการทำงานของโปรแกรม ซึ่งการทดสอบของการเปลี่ยนแปลงรายการที่ 1 นี้จะเป็นการทดสอบฟังก์ชันการยกยอดวันลาคงเหลือ โดยกรณีทดสอบของระบบที่ไม่อิงบริการแสดงดังตารางที่ 5.2 และกรณีทดสอบของระบบอิงบริการแสดงดังตารางที่ 5.3

ตารางที่ 5.2 การทดสอบการยกยอดวันลาคงเหลือ (Non Service)

|  |  |  |  |
| --- | --- | --- | --- |
| หมายเลขกรณีทดสอบ: | TC01 | ชื่อกรณีทดสอบ: | การยกยอดวันลาคงเหลือ |
| ระบบ: | Eleave (Non Service) | ระบบย่อย: | Leave Brought Forward |
| ผู้จัดทำ: | คงสวัสดิ์ โพธิพนม | วันที่จัดทำ: | 13 ธ.ค. 54 |
| ผู้ทำการทดสอบ: | คงสวัสดิ์ โพธิพนม | วันที่ทำการทดสอบ: | 13 ธ.ค. 54 |
| คำอธิบาย: | ทำการยกยอดวันลาคงเหลือของพนักงานในปีที่ผ่านมาไปยังปีปัจจุบัน | | |
| ภาวะก่อนการทดสอบ: | | | |
| 1) ทำการล็อกอินเข้าสู่ระบบด้วยชื่อผู้ใช้งานที่เป็นผู้ดูแลระบบ | | | |
| 2) ระบบแสดงหน้าจอเมนูหลัก | | | |

ตารางที่ 5.2 การทดสอบการยกยอดวันลาคงเหลือ (Non Service) (ต่อ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ขั้นตอน | การทดสอบ | ผลการทดสอบที่คาดหวัง | ผ่าน/ไม่ผ่าน | หมายเหตุ |
| 1 | เลือกเมนู Year-End Leave Bring Forward | ระบบแสดงข้อมูลการยกยอดวันลาของพนักงานทั้งหมดในหน้าจอการยกยอดวันลาคงเหลือ | ผ่าน |  |
| 2 | กดปุ่ม "Update All" | ระบบแสดงข้อความบันทึกข้อมูลสำเร็จ | ผ่าน |  |
| 3 | ตรวจสอบ Post-conditions |  | ผ่าน |  |
| ภาวะหลังการทดสอบ: | | | | |
| 1) ข้อมูลการยกยอดวันลาของพนักงานในปีปัจจุบันถูกเพิ่มลงในตาราง "el\_leave\_adjustment" | | | | |
|  | | | | |
|  | | | | |
|  | | | | |

ตารางที่ 5.3 การทดสอบการยกยอดวันลาคงเหลือ (Service)

|  |  |  |  |
| --- | --- | --- | --- |
| หมายเลขกรณีทดสอบ: | TC02 | ชื่อกรณีทดสอบ: | การยกยอดวันลาคงเหลือ |
| ระบบ: | Eleave (Service) | ระบบย่อย: | Leave Brought Forward |
| ผู้จัดทำ: | คงสวัสดิ์ โพธิพนม | วันที่จัดทำ: | 18 ธ.ค. 54 |
| ผู้ทำการทดสอบ: | คงสวัสดิ์ โพธิพนม | วันที่ทำการทดสอบ: | 18 ธ.ค. 54 |
| คำอธิบาย: | ทำการยกยอดวันลาคงเหลือของพนักงานในปีที่ผ่านมาไปยังปีปัจจุบัน | | |
| ภาวะก่อนการทดสอบ: | | | |
| 1) ทำการล็อกอินเข้าสู่ระบบด้วยชื่อผู้ใช้งานที่เป็นผู้ดูแลระบบ | | | |
| 2) ระบบแสดงหน้าจอเมนูหลัก | | | |

ตารางที่ 5.3 การทดสอบการยกยอดวันลาคงเหลือ (Service) (ต่อ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ขั้นตอน | การทดสอบ | ผลการทดสอบที่คาดหวัง | ผ่าน/ไม่ผ่าน | หมายเหตุ |
| 1 | เลือกเมนู Year-End Leave Bring Forward | ระบบแสดงข้อมูลการยกยอดวันลาของพนักงานทั้งหมดในหน้าจอการยกยอดวันลาคงเหลือ | ผ่าน |  |
| 2 | กดปุ่ม "Update All" | ระบบแสดงข้อความบันทึกข้อมูลสำเร็จ | ผ่าน |  |
| 3 | ตรวจสอบ Post-conditions |  | ผ่าน |  |
| ภาวะหลังการทดสอบ: | | | | |
| 1) ข้อมูลการยกยอดวันลาของพนักงานในปีปัจจุบันถูกเพิ่มลงในตาราง "el\_leave\_adjustment" | | | | |
|  | | | | |
|  | | | | |
|  | | | | |

* + 1. การเปลี่ยนแปลงรายการที่ 2

เป็นการเปลี่ยนแปลงที่ต้องการให้ระบบสามารถสนับสนุนการบันทึกการลาของพนักงานที่ทำงานเป็นกะได้ กิจกรรมของการดำเนินงานมีดังนี้

* + - 1. การวิเคราะห์และออกแบบระบบ

แผนภาพที่ผู้วิจัยได้นำมาใช้ในการช่วงของการวิเคราะห์และออกแบบระบบได้แก่ แผนภาพยูสเคส แผนภาพกิจกรรม แผนภาพความสัมพันธ์ของคลาส และแผนภาพลำดับการทำงานของระบบ รายละเอียดของการออกแบบของแต่ละแผนภาพมีดังนี้

* + - * 1. แผนภาพยูสเคสของการเปลี่ยนแปลงรายการที่ 1 กำหนดให้แอคเตอร์ (Actor) ประกอบด้วย ผู้ดูแลระบบฝ่ายบุคคล (HR Admin) และพนักงานที่ทำงานเป็นกะ กำหนดยูสเคสของการเปลี่ยนแปลงนี้ คือ การปรับปรุงข้อมูลประเภทการทำงานเป็นกะเวลาของพนักงาน (Update Shift-Working Employee Info) และการบันทึกการลาของพนักงานที่ทำงานเป็นกะ (Apply Leave) ในส่วนของแผนภาพยูสเคสนี้ของระบบที่ไม่อิงบริการ และระบบที่อิงบริการจะมีการออกแบบที่เหมือนกัน สามารถแสดงดังรูปที่ 5.5 และแสดงตารางคำอธิบายยูสเคสได้ดังตารางที่ 5.4



รูปที่ 5.5 แผนภาพยูสเคสของการเปลี่ยนแปลงรายการที่ 2

ตารางที่ 5.4 คำอธิบายยูสเคสการปรับปรุงข้อมูลประเภทการทำงานเป็นกะเวลาของพนักงาน

|  |  |  |
| --- | --- | --- |
| **ชื่อยูสเคส:** Update Shift-Working Employee Info | **รหัส:** UC02 | **ระดับความสำคัญ:**สูง |
| **ผู้กระทำหลัก:** HR Admin | **ชนิดยูสเคส:**รายละเอียดและสาระสำคัญ | |
| **ผู้มีส่วนเกี่ยวข้อง:**   * HR Admin – แก้ไขข้อมูลประเภทการทำงานของพนักงาน | | |
| **รายละเอียดยูสเคส:**  ยูสเคสนี้อธิบายถึงขั้นตอนที่ HRAdmin ทำการปรับปรุงประเภทการทำงานของพนักงานให้เป็นแบบกะเวลา (Shift Working) | | |
| **สิ่งกระตุ้น:**  HR Admin ต้องการแก้ไขประเภทการทำงานของพนักงาน | | |
| **ยูสเคสที่เกี่ยวข้อง:** - | | |
| **ภาวะก่อนทำงาน:**  ข้อมูลของพนักงานผู้ที่จะถูกกำหนดประเภทการทำงานเป็นแบบกะเวลาต้องมีข้อมูลอยู่ในระบบแล้ว | | |
| **ภาวะหลังทำงาน:**  ประเภทการทำงานของพนักงานได้ถูกกำหนดเป็นแบบกะเวลา | | |
| **สายงานปกติ:**   1. ล็อกอินเข้าสู่ระบบ 2. ระบบแสดงเมนูส่วน HR Admin | | |

ตารางที่ 5.4 คำอธิบายยูสเคสการปรับปรุงข้อมูลประเภทการทำงานเป็นกะเวลาของพนักงาน (ต่อ)

|  |  |  |
| --- | --- | --- |
| **ชื่อยูสเคส:** Update Shift-Working Employee Info | **รหัส:** UC02 | **ระดับความสำคัญ:**สูง |
| **ผู้กระทำหลัก:** HR Admin | **ชนิดยูสเคส:**รายละเอียดและสาระสำคัญ | |
| **สายงานปกติ:**   1. เลือกเมนู “Update Employee Info” 2. ระบบแสดงรายชื่อของพนักงานทั้งหมดของบริษัทในตาราง 3. ทำการกดปุ่มแก้ไขข้อมูลของพนักงานที่ต้องการแก้ไขประเภทการทำงาน 4. ระบบแสดงหน้าจอการปรับปรุงข้อมูลพนักงาน 5. กำหนดประเภทการทำงานของพนักงานให้เป็นแบบกะเวลาด้วยเลือกเช็คบ็อกซ์Shift Working 6. กดปุ่ม Submit เพื่อบันทึกข้อมูลของพนักงาน 7. ระบบบันทึกข้อมูลพนักงาน 8. ระบบแจ้งการบันทึกข้อมูลเสร็จเรียบร้อย | | |
| **สายงานย่อย:** | | |
| **สายงานทางเลือก / สายงานพิเศษ:** | | |

ตารางที่ 5.5 คำอธิบายยูสเคสการบันทึกการลาของพนักงานที่ทำงานเป็นกะ

|  |  |  |
| --- | --- | --- |
| **ชื่อยูสเคส:** Apply Leave | **รหัส:** UC03 | **ระดับความสำคัญ:**สูง |
| **ผู้กระทำหลัก:** Employee (Shift Working) | **ชนิดยูสเคส:**รายละเอียดและสาระสำคัญ | |
| **ผู้มีส่วนเกี่ยวข้อง:**   * Employee (Shift Working) – ทำการบันทึกการลา | | |
| **รายละเอียดยูสเคส:**  ยูสเคสนี้อธิบายถึงขั้นตอนที่พนักงานที่ถูกกำหนดประเภทการทำงานเป็นกะเวลาทำการบันทึกการลา | | |
| **สิ่งกระตุ้น:**  Employee (Shift Working) ต้องการบันทึกการลา | | |
| **ยูสเคสที่เกี่ยวข้อง:** - | | |
| **ภาวะก่อนทำงาน:**  พนักงานได้ถูกกำหนดให้ประเภทการทำงานเป็นแบบกะเวลาแล้ว | | |

ตารางที่ 5.5 คำอธิบายยูสเคสการบันทึกการลาของพนักงานที่ทำงานเป็นกะ (ต่อ)

|  |  |  |
| --- | --- | --- |
| **ชื่อยูสเคส:** Apply Leave | **รหัส:** UC03 | **ระดับความสำคัญ:**สูง |
| **ผู้กระทำหลัก:** Employee (Shift Working) | **ชนิดยูสเคส:**รายละเอียดและสาระสำคัญ | |
| **ภาวะหลังทำงาน:**   1. ข้อมูลการลาของพนักงานถูกเพิ่มลงในส่วนตารางประวัติการลาในหน้าจอบันทึกการลา 2. ข้อมูลการลาที่ตรงกับวันเสาร์อาทิตย์ หรือตรงกับวันหยุดที่ทางบริษัทประกาศเป็นวันหยุดจะถูกนำมาคิดเป็นวันลาที่ใช้ไปด้วย | | |
| **สายงานปกติ:**   1. พนักงานล็อกอินเข้าสู่ระบบ 2. ระบบแสดงเมนูส่วน Employee 3. เลือกเมนู “Apply Leave” 4. ระบบแสดงหน้าจอบันทึกการลา 5. พนักงานกรอกข้อมูลการลา 6. พนักงานกดปุ่ม Apply 7. ระบบทำการบันทึกข้อมูลการลา 8. ระบบส่งอีเมลแจ้งเตือนไปยังผู้อนุมัติการลาของพนักงานผู้นั้น 9. ระบบแจ้งการบันทึกข้อมูลเสร็จเรียบร้อย | | |
| **สายงานย่อย:** | | |
| **สายงานทางเลือก / สายงานพิเศษ:** | | |

* + - * 1. แผนภาพกิจกรรมของการเปลี่ยนแปลงรายการที่ 2 แสดงกิจกรรมการปรับปรุงข้อมูลประเภทการทำงานเป็นกะเวลาของพนักงาน และกิจกรรมของการบันทึกการลาของพนักงานที่ประเภทการทำงานเป็นกะเวลา ซึ่งสามารถแสดงแผนภาพกิจกรรมของระบบที่ไม่อิงบริการ และแผนภาพกิจกรรมของระบบอิงบริการในภาคผนวก ข
        2. แผนภาพความสัมพันธ์ของคลาสของการเปลี่ยนแปลงรายการที่ 2 ประกอบด้วยคลาสที่เกี่ยวข้องกับการเปลี่ยนแปลงนี้ 3 คลาส ได้แก่ คลาสข้อมูลการลา (Leave Record) คลาสประเภทการลา (Leave Type) คลาสพนักงาน (Employee) และคลาสหมวดหมู่การลา (Leave Category) สามารถแสดงแผนภาพความสัมพันธ์ดังรูปที่ 5.6

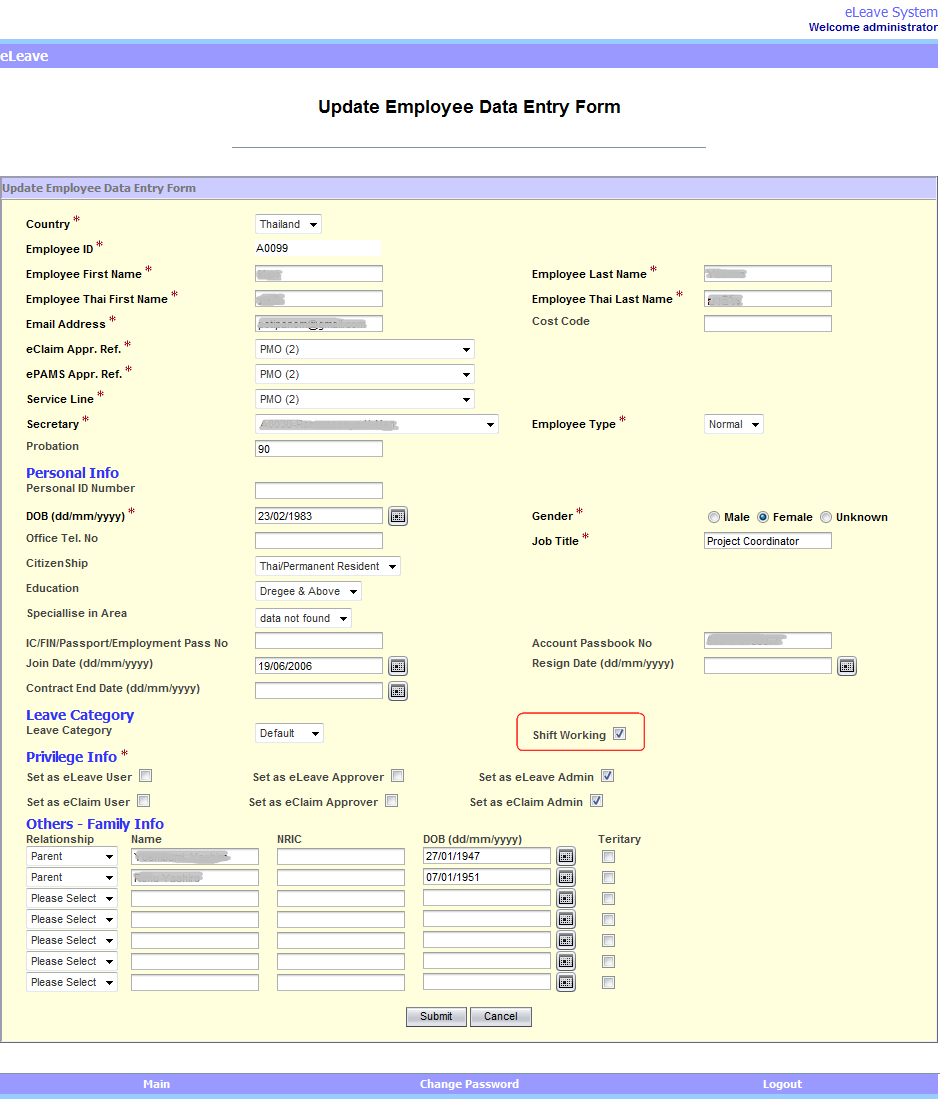


รูปที่ 5.6 แผนภาพคลาสของการเปลี่ยนแปลงรายการที่ 2

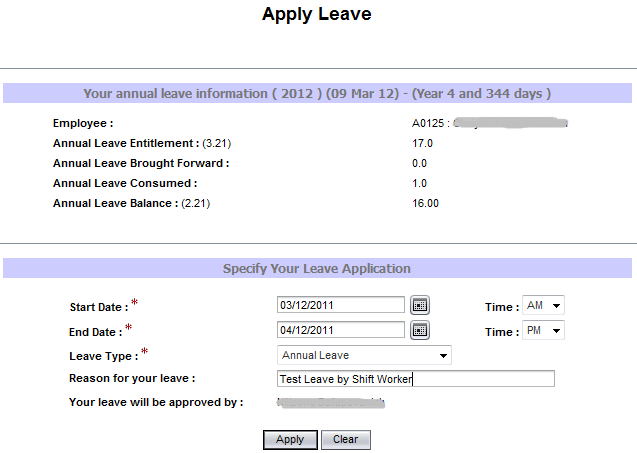
* + - * 1. แผนภาพลำดับของการเปลี่ยนแปลงรายการที่ 2 จะแสดงลำดับขั้นตอนในการบันทึกข้อมูลการลาของพนักงาน และลำดับขั้นตอนในการคำนวณจำนวนวันลาที่ใช้ โดยแสดงแผนภาพลำดับการทำงานของระบบไม่อิงบริการ และแผนภาพลำดับการทำงานของระบบอิงบริการในภาคผนวก ข
      1. การพัฒนาระบบ

ในส่วนนี้จะแสดงหน้าจอโปรแกรมที่เกี่ยวข้องกับการเปลี่ยนแปลงรายการที่ 2 ประกอบด้วยหน้าจอดังนี้

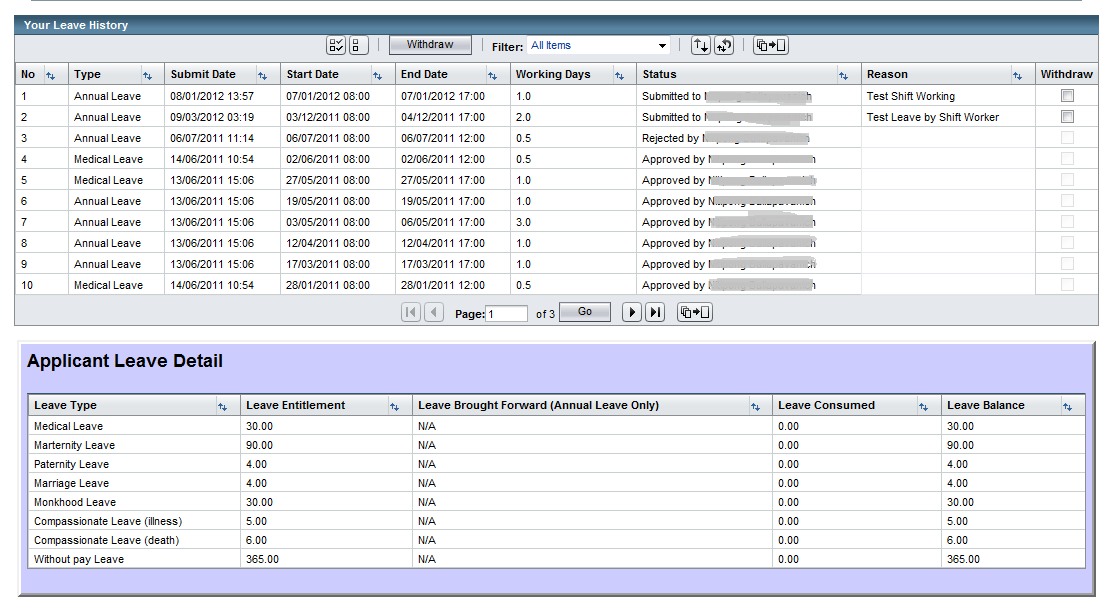
* + - * 1. หน้าจอการปรับปรุงข้อมูลพนักงาน หน้าจอนี้ HR Admin จะเข้ามาทำการกำหนดประเภทการทำงานของพนักงานให้เป็นแบบกะเวลาโดยการเลือกเช็คบ็อกซ์ดังรูปที่ 5.7
        2. หน้าจอการบันทึกการลาของพนักงาน ในหน้าจอนี้พนักงานที่ประเภทการทำงานเป็นกะเวลาเข้ามาบันทึกข้อมูลการลาแสดงดังรูปที่ 5.8 ในส่วนข้อมูลประวัติการลาของพนักงานจะแสดงดังรูปที่ 5.9 ซึ่งข้อมูลการลารายการใดที่ตรงกับวันเสาร์-อาทิตย์ หรือวันหยุดที่ทางบริษัทประกาศ จะถูกนับรวมเป็นวันลาที่ใช้ไป



รูปที่ 5.7 หน้าจอการปรับปรุงข้อมูลพนักงาน



รูปที่ 5.8 หน้าจอการบันทึกการลา



รูปที่ 5.9 ข้อมูลประวัติการลาของพนักงาน

* + - 1. การทดสอบระบบ

การทดสอบระบบจะเป็นการทดสอบตามฟังก์ชันการทำงานของโปรแกรม ซึ่งการทดสอบของการเปลี่ยนแปลงรายการที่ 2 จะเป็นการทดสอบฟังก์ชันการกำหนดประเภทการทำงานของพนักงานเป็นแบบกะฟังก์ชันการบันทึกการลา โดยที่กรณีทดสอบการกำหนดประเภทการทำงานของพนักงานเป็นแบบกะของระบบที่ไม่อิงบริการแสดงดังตารางที่ 5.6 และของระบบอิงบริการแสดงดังตารางที่ 5.7 ในส่วนกรณีทดสอบของการบันทึกการลาของระบบที่ไม่อิงบริการแสดงดังตารางที่ 5.8 และกรณีทดสอบของระบบอิงบริการแสดงดังตารางที่ 5.9

ตารางที่ 5.6 การทดสอบการกำหนดประเภทการทำงานของพนักงานเป็นแบบกะ(Non Service)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| หมายเลขกรณีทดสอบ: | | TC03 | ชื่อกรณีทดสอบ: | | กำหนดประเภทการทำงานของพนักงานเป็นแบบกะ | | |
| ระบบ: | | Eleave (Non Service) | ระบบย่อย: | | Update Employee Info. | | |
| ผู้จัดทำ: | | คงสวัสดิ์ โพธิพนม | วันที่จัดทำ: | | 5 ม.ค. 55 | | |
| ผู้ทำการทดสอบ: | | คงสวัสดิ์ โพธิพนม | วันที่ทำการทดสอบ: | | 5 ม.ค. 55 | | |
| คำอธิบาย: | | ทำการปรับปรุงข้อมูลประเภทการทำงานของพนักงานเป็นแบบกะ | | | | | |
| Pre-conditions: | | | | | | | |
| 1) ทำการล็อกอินเข้าสู่ระบบด้วยชื่อผู้ใช้งานที่เป็นผู้ดูแลระบบ | | | | | | | |
| 2) ผู้ใช้เข้าสู่หน้าจอ Update Employee Info และเลือกแก้ไขข้อมูลพนักงานผู้ที่ต้องการกำหนดประเภทการทำงานเป็นแบบกะ | | | | | | | |
| 3) ระบบแสดงหน้าจอการแก้ไขข้อมูลพนักงาน | | | | | | | |
| ขั้นตอน | การทดสอบ | | | ผลการทดสอบที่คาดหวัง | | ผ่าน/ไม่ผ่าน | หมายเหตุ |
| 1 | เลือกเช็คบ็อกซ์ "Shift Working" จากนั้นกดปุ่ม "Submit" | | | ระบบแสดงข้อความปรับปรุงข้อมูลพนักงานสำเร็จ | | ผ่าน |  |
| 2 | ตรวจสอบ Post-conditions | | |  | | ผ่าน |  |
| Post-conditions: | | | | | | | |
| 1) ฟิลด์ shift\_workingในตาราง “el\_employee” มีค่าเป็น 1 | | | | | | | |

ตารางที่ 5.7 การทดสอบการกำหนดประเภทการทำงานของพนักงานเป็นแบบกะ(Service)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| หมายเลขกรณีทดสอบ: | | TC04 | | ชื่อกรณีทดสอบ: | กำหนดประเภทการทำงานของพนักงานเป็นแบบกะ | | |
| ระบบ: | | Eleave (Service) | | ระบบย่อย: | Update Employee Info | | |
| ผู้จัดทำ: | | คงสวัสดิ์ โพธิพนม | | วันที่จัดทำ: | 10 ม.ค. 55 | | |
| ผู้ทำการทดสอบ: | | คงสวัสดิ์ โพธิพนม | | วันที่ทำการทดสอบ: | 10 ม.ค. 55 | | |
| คำอธิบาย: | | ทำการปรับปรุงข้อมูลประเภทการทำงานของพนักงานเป็นแบบกะ | | | | | |
| Pre-conditions: | | | | | | | |
| 1) ทำการล็อกอินเข้าสู่ระบบด้วยชื่อผู้ใช้งานที่เป็นผู้ดูแลระบบ | | | | | | | |
| 2) ผู้ใช้เข้าสู่หน้าจอ Update Employee Info.  และเลือกแก้ไขข้อมูลพนักงานผู้ที่ต้องการกำหนดประเภทการทำงานเป็นแบบกะ | | | | | | | |
| 3) ระบบแสดงหน้าจอการแก้ไขข้อมูลพนักงาน | | | | | | | |
| ขั้นตอน | การทดสอบ | | ผลการทดสอบที่คาดหวัง | | | ผ่าน/ไม่ผ่าน | หมายเหตุ |
| 1 | เลือกเช็คบ็อกซ์ "Shift Working" จากนั้นกดปุ่ม "Submit" | | ระบบแสดงข้อความปรับปรุงข้อมูลพนักงานสำเร็จ | | | ผ่าน |  |
| 2 | ตรวจสอบ Post-conditions | |  | | | ผ่าน |  |
| Post-conditions: | | | | | | | |
| 1) ฟิลด์ shift\_workingในตาราง “el\_employee” มีค่าเป็น 1 | | | | | | | |

ตารางที่ 5.8 การทดสอบการบันทึกการลา(Non Service)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| หมายเลขกรณีทดสอบ: | | TC05 | | ชื่อกรณีทดสอบ: | | บันทึกการลา | |
| ระบบ: | | Eleave (Non Service) | | ระบบย่อย: | | Apply Leave | |
| ผู้จัดทำ: | | คงสวัสดิ์ โพธิพนม | | วันที่จัดทำ: | | 5 ม.ค. 55 | |
| ผู้ทำการทดสอบ: | | คงสวัสดิ์ โพธิพนม | | วันที่ทำการทดสอบ: | | 5 ม.ค. 55 | |
| คำอธิบาย: | | พนักงานที่ทำงานแบบกะทำการบันทึกการลา | | | | | |
| Pre-conditions: | | | | | | | |
| 1) พนักงานถูกกำหนดให้ประเภทการทำงานเป็นแบบกะโดยผู้ดูแลระบบฝ่ายทรัพยากรบุคคล | | | | | | | |
| 2) พนักงานล็อกอินเข้าสู่ระบบ และเข้าสู่หน้าบันทึกการลา | | | | | | | |
| ขั้นตอน | การทดสอบ | | ผลการทดสอบที่คาดหวัง | | ผ่าน/ไม่ผ่าน | | หมายเหตุ |
| 1 | กรอกข้อมูลการลาโดยเลือกวันที่ทำการลาเป็นช่วงวันเสาร์และอาทิตย์จากนั้นกดปุ่ม "Apply" | | ระบบแสดงข้อความบันทึกการลาสำเร็จ | | ผ่าน | |  |
| 2 | ตรวจสอบ Post-conditions | |  | | ผ่าน | |  |
| Post-conditions: | | | | | | | |
| 1) ข้อมูลการลาถูกเพิ่มลงในตาราง "el\_leave\_record" | | | | | | | |
| 2) ข้อมูลการลาถูกเพิ่มลงในส่วนตารางประวัติการลาในหน้าจอบันทึกการลา | | | | | | | |
| 3) ข้อมูลการลาที่ตรงกับวันเสาร์อาทิตย์หรือตรงกับวันหยุดที่ทางบริษัทประกาศเป็นวันหยุด จะต้องถูกนำมาคิดเป็นวันลาที่ใช้ไปด้วย | | | | | | | |

ตารางที่ 5.9 การทดสอบการบันทึกการลา(Service)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| หมายเลขกรณีทดสอบ: | | TC06 | | ชื่อกรณีทดสอบ: | | บันทึกการลา | |
| ระบบ: | | Eleave (Service) | | ระบบย่อย: | | Apply Leave | |
| ผู้จัดทำ: | | คงสวัสดิ์ โพธิพนม | | วันที่จัดทำ: | | 10 ม.ค. 55 | |
| ผู้ทำการทดสอบ: | | คงสวัสดิ์ โพธิพนม | | วันที่ทำการทดสอบ: | | 10 ม.ค. 55 | |
| คำอธิบาย: | | พนักงานที่ทำงานแบบกะทำการบันทึกการลา | | | | | |
| Pre-conditions: | | | | | | | |
| 1) พนักงานถูกกำหนดให้ประเภทการทำงานเป็นแบบกะโดยผู้ดูแลระบบฝ่ายทรัพยากรบุคคล | | | | | | | |
| 2) พนักงานล็อกอินเข้าสู่ระบบ และเข้าสู่หน้าบันทึกการลา | | | | | | | |
| ขั้นตอน | การทดสอบ | | ผลการทดสอบที่คาดหวัง | | ผ่าน/ไม่ผ่าน | | หมายเหตุ |
| 1 | กรอกข้อมูลการลาโดยเลือกวันที่ทำการลาเป็นช่วงวันเสาร์และอาทิตย์จากนั้นกดปุ่ม "Apply" | | ระบบแสดงข้อความบันทึกการลาสำเร็จ | | ผ่าน | |  |
| 2 | ตรวจสอบ Post-conditions | |  | | ผ่าน | |  |
| Post-conditions: | | | | | | | |
| 1) ข้อมูลการลาถูกเพิ่มลงในตาราง "el\_leave\_record" | | | | | | | |
| 2) ข้อมูลการลาถูกเพิ่มลงในส่วนตารางประวัติการลาในหน้าจอบันทึกการลา | | | | | | | |
| 3) ข้อมูลการลาที่ตรงกับวันเสาร์อาทิตย์หรือตรงกับวันหยุดที่ทางบริษัทประกาศเป็นวันหยุด จะต้องถูกนำมาคิดเป็นวันลาที่ใช้ไปด้วย | | | | | | | |

## ผลการทดลอง

ผลการทดลองทั้งหมดถูกรวบรวมจากค่าที่บันทึก และวัดจากเครื่องมือต่าง ๆ ได้แก่ ตารางบันทึกเวลาการทำงาน ตารางบันทึกผลกระทบจากการเปลี่ยนแปลง และค่าความซับซ้อนของโค้ดที่วัดได้จากโปรแกรมลอคเมตริกส์ โดยผลการทดลองในการดำเนินการกับการเปลี่ยนแปลงแต่ละรายการมีรายละเอียด ดังนี้

### ผลการทดลองการเปลี่ยนแปลงรายการที่ 1

* + - 1. ตารางบันทึกผลกระทบจากการเปลี่ยนแปลง

หลังจากที่เสร็จกิจกรรมในส่วนของการวิเคราะห์และออกแบบการเปลี่ยนแปลงรายการที่ 1 จะสามารถทราบถึงผลกระทบของการเปลี่ยนแปลงที่มีต่อระบบ โดยผลกระทบที่จะเกิดกับระบบที่ไม่อิงบริการแสดงดังตารางที่ 5.10 และผลกระทบที่จะเกิดกับระบบอิงบริการแสดงดังตารางที่ 5.11

ตารางที่ 5.10 ผลกระทบจากการเปลี่ยนแปลงรายการที่ 1 (Non Service)

|  |  |
| --- | --- |
| **Layer**  **(Non-Service)** | **Affected JSP Files/Packages/Java Classes** |
| **Presentation** | ELeaveBringForward.jsp |
| **Application** | com.aoth.eapps, ELeaveBringForward.java |
| **Domain** | - |
| **Data Access** | - |

ตารางที่ 5.11 ผลกระทบจากการเปลี่ยนแปลงรายการที่ 1 (Service)

| **Layer**  **(Service)** | **Affected JSP Files/Packages/Java Classes/Services** |
| --- | --- |
| **Presentation** | ELeaveBringForward.jsp |
| **Application** | com.aoth.eapps, ELeaveBringForward.java |
| **Domain** | - |
| **Service** | EleaveService |
| **Data Access** | - |

* + - 1. ค่าความซับซ้อนของโค้ด

เมื่อทราบถึงมอดูลที่ได้รับผลกระทบจากการเปลี่ยนแปลงรายการที่ 1 การหาความซับซ้อนของโค้ดของมอดูลที่ได้รับผลกระทบทำโดยการนำแพ็คเกจของโค้ดมอดูลนั้นทำการวัดโดยโปรแกรมลอคเมตริกส์ ซึ่งได้ค่าความซับซ้อนของโค้ดของระบบที่ไม่อิงบริการดังตารางที่ 5.12 และได้ค่าความซับซ้อนของโค้ดของระบบอิงบริการดังตารางที่ 5.13

ตารางที่ 5.12 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 1 (Non Service)

| สัญลักษณ์ | จำนวน | คำอธิบาย |
| --- | --- | --- |
| Source Files | 37 | จำนวนแฟ้มข้อมูลทั้งหมด |
| Directories | 19 | จำนวนสารบบทั้งหมด |
| LOC | 25590 | จำนวนบรรทัดทั้งหมดของโค้ด |
| BLOC | 2950 | จำนวนบรรทัดว่างทั้งหมด |
| SLOC-P | 18939 | จำนวนบรรทัดทั้งหมดของโค้ดทางกายภาพที่สามารถประมวลผลได้ |
| SLOC-L | 14567 | จำนวนบรรทัดทั้งหมดของโค้ดทางตรรกะที่สามารถประมวลผลได้ |
| ***MVG*** | ***2298*** | ***ค่าความซับซ้อนทั้งหมดของโค้ด***  ***(McCabe VG Complexity)*** |
| C&SLOC | 55 | จำนวนบรรทัดทั้งหมดของโค้ดที่มีหมายเหตุ |
| CLOC | 3701 | จำนวนบรรทัดทั้งหมดที่มีเฉพาะหมายเหตุ |
| CWORD | 21079 | จำนวนคำทั้งหมดเป็นหมายเหตุ |
| HCLOC | 148 | จำนวนบรรทัดทั้งหมดของหมายเหตุที่เป็นส่วนหัวของฟังก์ชัน |
| HCWORD | 666 | จำนวนคำทั้งหมดของหมายเหตุที่เป็นส่วนหัวของฟังก์ชัน |

ตารางที่ 5.13 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 1 (Service)

| สัญลักษณ์ | จำนวน | คำอธิบาย |
| --- | --- | --- |
| Source Files | 37 | จำนวนแฟ้มข้อมูลทั้งหมด |
| Directories | 19 | จำนวนสารบบทั้งหมด |
| LOC | 25645 | จำนวนบรรทัดทั้งหมดของโค้ด |
| BLOC | 2960 | จำนวนบรรทัดว่างทั้งหมด |
| SLOC-P | 19030 | จำนวนบรรทัดทั้งหมดของโค้ดทางกายภาพที่สามารถประมวลผลได้ |
| SLOC-L | 14642 | จำนวนบรรทัดทั้งหมดของโค้ดทางตรรกะที่สามารถประมวลผลได้ |

ตารางที่5.13 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 1 (Service) (ต่อ)

| สัญลักษณ์ | จำนวน | คำอธิบาย |
| --- | --- | --- |
| ***MVG*** | ***2308*** | ***ค่าความซับซ้อนทั้งหมดของโค้ด***  ***(McCabe VG Complexity)*** |
| C&SLOC | 55 | จำนวนบรรทัดทั้งหมดของโค้ดที่มีหมายเหตุ |
| CLOC | 3655 | จำนวนบรรทัดทั้งหมดที่มีเฉพาะหมายเหตุ |
| CWORD | 20904 | จำนวนคำทั้งหมดเป็นหมายเหตุ |
| HCLOC | 148 | จำนวนบรรทัดทั้งหมดของหมายเหตุที่เป็นส่วนหัวของฟังก์ชัน |
| HCWORD | 666 | จำนวนคำทั้งหมดของหมายเหตุที่เป็นส่วนหัวของฟังก์ชัน |

* + - 1. ตารางบันทึกเวลาการทำงาน

รายละเอียดของการทำงานของการดำเนินการกับการเปลี่ยนแปลงรายการที่ 1 กับระบบที่ไม่อิงบริการ และระบบอิงบริการสามารถแสดงดังรูปในภาคผนวก ค

### ผลการทดลองการเปลี่ยนแปลงรายการที่ 2

#### ตารางบันทึกผลกระทบจากการเปลี่ยนแปลง

หลังจากที่เสร็จกิจกรรมในส่วนของการวิเคราะห์และออกแบบการเปลี่ยนแปลงรายการที่ 2 จะสามารถทราบถึงผลกระทบของการเปลี่ยนแปลงที่มีต่อระบบ โดยผลกระทบที่จะเกิดกับระบบที่ไม่อิงบริการแสดงดังตารางที่ 5.14 และผลกระทบที่จะเกิดกับระบบอิงบริการแสดงดังตารางที่ 5.15

ตารางที่ 5.14 ผลกระทบจากการเปลี่ยนแปลงรายการที่ 2 (Non Service)

|  |  |
| --- | --- |
| **Layer**  **(Non-Service)** | **Affected JSP Files/Packages/Java Classes** |
| **Presentation** | EleaveNewUpdateEmployee.jsp |
| **Application** | com.aoth.eapps, EleaveNewUpdateEmployee.java, ELeaveApplyLeave.java |
| **Domain** | com.aoth.eapps.domain, ELeaveDomain.java, |
| **Data Access** | com.aoth.eapps.dto, ElEmployee.java, ElLeaveRecord.java |

ตารางที่ 5.15 ผลกระทบจากการเปลี่ยนแปลงรายการที่ 2 (Service)

| **Layer**  **(Service)** | **Affected JSP Files/Packages/Java Classes/Services** |
| --- | --- |
| **Presentation** | EleaveNewUpdateEmployee.jsp |
| **Application** | com.aoth.eapps, EleaveNewUpdateEmployee.java, ELeaveApplyLeave.java |
| **Domain** | com.aoth.eapps.domain, ELeaveDomain.java, |
| **Service** | EleaveService, EmployeeService |
| **Data Access** | com.aoth.eapps.dto, ElEmployee.java, ElLeaveRecord.java |

#### ค่าความซับซ้อนของโค้ด

เมื่อทราบถึงมอดูลที่ได้รับผลกระทบจากการเปลี่ยนแปลงรายการที่ 2 การหาความซับซ้อนของโค้ดของมอดูลที่ได้รับผลกระทบทำโดยการนำแพ็คเกจของโค้ดมอดูลนั้นทำการวัดโดยโปรแกรมลอคเมตริกส์ ซึ่งได้ค่าความซับซ้อนของโค้ดของระบบที่ไม่อิงบริการดังตารางที่ 5.16 และได้ค่าความซับซ้อนของโค้ดของระบบอิงบริการดังตารางที่ 5.17

ตารางที่ 5.16 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 2 (Non Service)

| สัญลักษณ์ | จำนวน | คำอธิบาย |
| --- | --- | --- |
| Source Files | 68 | จำนวนแฟ้มข้อมูลทั้งหมด |
| Directories | 21 | จำนวนสารบบทั้งหมด |
| LOC | 30728 | จำนวนบรรทัดทั้งหมดของโค้ด |
| BLOC | 3847 | จำนวนบรรทัดว่างทั้งหมด |
| SLOC-P | 22804 | จำนวนบรรทัดทั้งหมดของโค้ดทางกายภาพที่สามารถประมวลผลได้ |
| SLOC-L | 17434 | จำนวนบรรทัดทั้งหมดของโค้ดทางตรรกะที่สามารถประมวลผลได้ |
| ***MVG*** | ***2746*** | ***ค่าความซับซ้อนทั้งหมดของโค้ด***  ***(McCabe VG Complexity)*** |
| C&SLOC | 113 | จำนวนบรรทัดทั้งหมดของโค้ดที่มีหมายเหตุ |
| CLOC | 4077 | จำนวนบรรทัดทั้งหมดที่มีเฉพาะหมายเหตุ |
| CWORD | 22549 | จำนวนคำทั้งหมดเป็นหมายเหตุ |
| HCLOC | 188 | จำนวนบรรทัดทั้งหมดของหมายเหตุที่เป็นส่วนหัวของฟังก์ชัน |
| HCWORD | 846 | จำนวนคำทั้งหมดของหมายเหตุที่เป็นส่วนหัวของฟังก์ชัน |

ตารางที่ 5.17 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 2 (Service)

| สัญลักษณ์ | จำนวน | คำอธิบาย |
| --- | --- | --- |
| Source Files | 68 | จำนวนแฟ้มข้อมูลทั้งหมด |
| Directories | 30 | จำนวนสารบบทั้งหมด |
| LOC | 30556 | จำนวนบรรทัดทั้งหมดของโค้ด |
| BLOC | 3841 | จำนวนบรรทัดว่างทั้งหมด |
| SLOC-P | 22712 | จำนวนบรรทัดทั้งหมดของโค้ดทางกายภาพที่สามารถประมวลผลได้ |
| SLOC-L | 17385 | จำนวนบรรทัดทั้งหมดของโค้ดทางตรรกะที่สามารถประมวลผลได้ |

ตารางที่5.17 ค่าความซับซ้อนของโค้ดของการเปลี่ยนแปลงรายการที่ 2 (Service) (ต่อ)

| สัญลักษณ์ | จำนวน | คำอธิบาย |
| --- | --- | --- |
| ***MVG*** | ***2728*** | ***ค่าความซับซ้อนทั้งหมดของโค้ด***  ***(McCabe VG Complexity)*** |
| C&SLOC | 112 | จำนวนบรรทัดทั้งหมดของโค้ดที่มีหมายเหตุ |
| CLOC | 4003 | จำนวนบรรทัดทั้งหมดที่มีเฉพาะหมายเหตุ |
| CWORD | 22267 | จำนวนคำทั้งหมดเป็นหมายเหตุ |
| HCLOC | 188 | จำนวนบรรทัดทั้งหมดของหมายเหตุที่เป็นส่วนหัวของฟังก์ชัน |
| HCWORD | 846 | จำนวนคำทั้งหมดของหมายเหตุที่เป็นส่วนหัวของฟังก์ชัน |

#### ตารางบันทึกเวลาการทำงาน

รายละเอียดของการทำงานของการดำเนินการกับการเปลี่ยนแปลงรายการที่ 2 กับระบบที่ไม่อิงบริการ และระบบอิงบริการสามารถแสดงดังรูปในภาคผนวก ค

## การคำนวณค่าความคล่องตัว

จากผลการทดลองสามารถคำนวณหาค่าความคล่องตัวของแต่ละระบบในการดำเนินการกับการเปลี่ยนแปลงแต่ละรายการดังนี้

### การคำนวณค่าความคล่องตัวจากการเปลี่ยนแปลงรายการที่ 1

การคำนวณค่าความคล่องตัวของระบบบันทึกการลาแบบไม่อิงบริการจากการเปลี่ยนแปลงรายการที่ 1 สามารถแสดงได้ดังตารางที่ 5.18 โดยที่ค่าระดับชั้นในเชิงสถาปัตยกรรมได้จากจำนวนระดับชั้นที่จะได้รับผลกระทบจากการเปลี่ยนแปลงของตารางที่ 12 ค่าความซับซ้อนของโค้ดได้จากตารางที่ 5.10 สำหรับค่าของจำนวนตำแหน่งงานจะนับจากประเภทตำแหน่งงาน (Role) ที่ใช้ดำเนินการ จำนวนการส่งต่องานระหว่างตำแหน่งงานจะนับจากผลิตภัณฑ์งาน (Work Products) ที่ได้จากการทำงานในตำแหน่งต่าง ๆ และเวลาที่ใช้ดำเนินการเปลี่ยนแปลงคำนวณจากเวลาการทำงาน (Working Hour) ทั้งหมดที่ใช้ในการดำเนินการกับการเปลี่ยนแปลงรายการที่ 1 โดยทั้งสามค่านี้ได้จากข้อมูลที่ถูกบันทึกลงในตารางการบันทึกเวลาดังรูปที่ ค.1 จากนั้นคำนวณหาค่าความกว้างของกระบวนการทำงานจากสมการ (6) คำนวณค่าความซับซ้อนของการเปลี่ยนแปลงจากสมการ (5) และคำนวณค่าความคล่องตัวจากสมการ (3)

ตารางที่ 5.18 การคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 1 (Non Service)

|  |  |
| --- | --- |
| ค่าระดับชั้นในเชิงสถาปัตยกรรม | 2 |
| จำนวนตำแหน่งงาน | 3 |
| จำนวนการส่งต่องานระหว่างตำแหน่งงาน | 2 |
| ค่าความกว้างของกระบวนการทำงาน (6) | 3 + 2 = 5 |
| ค่าความซับซ้อนของโค้ด | 2,298 |
| ค่าความซับซ้อนของการเปลี่ยนแปลง (5) | 2 \* 5 \* 2,298 = 22,980 |
| เวลาที่ใช้ดำเนินการเปลี่ยนแปลง | 29 ชั่วโมง |
| ค่าความคล่องตัว (3) | 22,980 / 29 = 792.41 |

การคำนวณค่าความคล่องตัวของระบบบันทึกการลาแบบอิงบริการจากการเปลี่ยนแปลงรายการที่ 1 สามารถแสดงได้ดังตารางที่ 5.19 โดยที่ค่าระดับชั้นในเชิงสถาปัตยกรรมได้จากจำนวนระดับชั้นที่จะได้รับผลกระทบจากการเปลี่ยนแปลงของตารางที่ 5.11 ค่าความซับซ้อนของโค้ดได้จากตารางที่ 5.13 สำหรับค่าของจำนวนตำแหน่งงานจะนับจากประเภทตำแหน่งงาน (Role) ที่ใช้ดำเนินการ จำนวนการส่งต่องานระหว่างตำแหน่งงานจะนับจากผลิตภัณฑ์งาน (Work Products) ที่ได้จากการทำงานในตำแหน่งต่าง ๆ และเวลาที่ใช้ดำเนินการเปลี่ยนแปลงคำนวณจากเวลาการทำงาน (Working Hour) ทั้งหมดที่ใช้ในการดำเนินการกับการเปลี่ยนแปลงรายการที่ 1 โดยทั้งสามค่านี้ได้จากข้อมูลที่ถูกบันทึกลงในตารางการบันทึกเวลาดังรูปที่ ค.2 จากนั้นคำนวณหาค่าความกว้างของกระบวณการทำงานจากสมการ (6) คำนวณค่าความซับซ้อนของการเปลี่ยนแปลงจากสมการ (5) และคำนวณค่าความคล่องตัวจากสมการ (3)

ตารางที่ 5.19 การคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 1 (Service)

|  |  |
| --- | --- |
| ค่าระดับชั้นในเชิงสถาปัตยกรรม | 3 |
| จำนวนตำแหน่งงาน | 3 |
| จำนวนการส่งต่องานระหว่างตำแหน่งงาน | 3 |
| ค่าความกว้างของกระบวนการทำงาน (6) | 3 + 3 = 6 |
| ค่าความซับซ้อนของโค้ด | 2,308 |
| ค่าความซับซ้อนของการเปลี่ยนแปลง (5) | 3 \* 6 \* 2,308 = 41,544 |
| เวลาที่ใช้ดำเนินการเปลี่ยนแปลง | 27 ชั่วโมง |
| ค่าความคล่องตัว (3) | 41,544 / 27 = 1,538.66 |

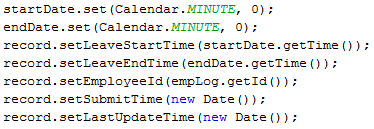
จากการคำนวณความคล่องตัวจากการเปลี่ยนแปลงรายการที่ 1 ของระบบบันทึกการลาแบบไม่อิงบริการและแบบอิงบริการสามารถสรุปและเปรียบเทียบผลการคำนวณระหว่างแบบจำลองความคล่องตัวของ [4] และของโครงงานนี้ได้ดังตารางที่ 5.20

ตารางที่ 5.20 ตารางสรุปการคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 1

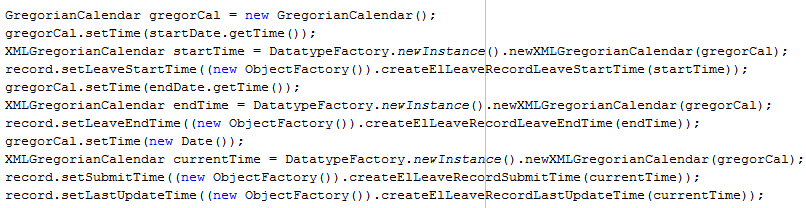
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ระบบบันทึกการลา | ระยะเวลา (ชั่วโมง) | จำนวนระดับชั้นในเชิงสถาปัตยกรรม | จำนวนตำแหน่ง | จำนวนการส่งต่องาน | ความซับซ้อนของโค้ด(MVG) | ความคล่องตัว[4] (4) | ความคล่องตัวที่เพิ่มขึ้น[4] (%) | ความคล่องตัวของโครงงานนี้ (5) | ความคล่องตัวที่เพิ่มขึ้นของโครงงานนี้(%) |
| แบบไม่อิงบริการ | 29 | 2 | 3 | 2 | 2,298 | 0.34 | 94.12 | 792.41 | 94.17 |
| แบบอิงบริการ | 27 | 3 | 3 | 3 | 2,308 | 0.66 | 1,538.66 |

เมื่อพิจารณาค่าความคล่องตัวของสองระบบในการดำเนินการกับการเปลี่ยนแปลงรายการที่ 1 ซึ่งเป็นการเปลี่ยนแปลงในลักษณะการเพิ่มฟังก์ชันงานใหม่และจากตารางที่ 5.20 พบว่า ระยะเวลาที่ใช้ดำเนินการของระบบอิงบริการน้อยกว่าระบบแบบไม่อิงบริการ เนื่องจากระบบอิงบริการมีการเรียกใช้เซอร์วิซทำให้การพัฒนามีความสะดวกมากกว่าการที่จะต้องเขียนโปรแกรมทั้งในส่วนของตรรกะโปรแกรมรวมถึงการติดต่อกับฐานข้อมูลเองทั้งหมด จำนวนระดับชั้นในเชิงสถาปัตยกรรมของระบบอิงบริการมีค่ามากกว่าเนื่องจากมีการแก้ไขตัวเซอร์วิซการลาโดยการเพิ่ม 2 เมท็อดเพื่อรองรับการทำงานของโปรแกรมการยกยอดวันลาคงเหลือที่ต้องกระทำกับข้อมูลมากกว่าหนึ่งรายการ และจากการที่ต้องเพิ่มเมท็อดในส่วนเซอร์วิซการลาทำให้ต้องมีการจัดสร้าง (Build) และติดตั้ง (Deploy) เซอร์วิซใหม่เพื่อให้โปรแกรมสามารถเรียกใช้ได้อีกครั้ง สำหรับค่าความซับซ้อนของโค้ดของระบบอิงบริการมีค่ามากกว่าเนื่องจากโค้ดบางส่วนเมื่อทำให้อยู่ในรูปแบบการเรียกใช้เซอร์วิซต้องมีการเขียนโค้ดเพิ่มเติมเพื่อทำให้สามารถติดต่อกับเว็บเซอร์วิซได้แสดงตัวอย่างโค้ดดังรูปที่ 5.10 จึงส่งผลให้มีค่าความซับซ้อนของโค้ดที่มากกว่าระบบที่ไม่ได้อิงบริการ เมื่อพิจารณาค่าความคล่องตัวที่คำนวณได้จาก [4] และจากโครงงานนี้พบว่าระบบอิงบริการมีค่าความคล่องตัวที่สูงกว่าระบบที่ไม่อิงบริการโดยมีเปอร์เซ็นต์ความคล่องตัวที่เพิ่มขึ้นใกล้เคียงกัน

(1) **Non Service**



(2) **Service**



รูปที่ 5.10 ความแตกต่างของโค้ดแบบไม่เรียกใช้เซอร์วิซกับแบบเรียกใช้เซอร์วิซ

### การคำนวณค่าความคล่องตัวจากการเปลี่ยนแปลงรายการที่ 2

การคำนวณค่าความคล่องตัวของระบบบันทึกการลาแบบไม่อิงบริการจากการเปลี่ยนแปลงรายการที่ 2 สามารถแสดงได้ดังตารางที่ 5.21 โดยที่ค่าระดับชั้นในเชิงสถาปัตยกรรมได้จากจำนวนระดับชั้นที่จะได้รับผลกระทบจากการเปลี่ยนแปลงของตารางที่ 5.14 ค่าความซับซ้อนของโค้ดได้จากตารางที่ 5.16 สำหรับค่าของจำนวนตำแหน่งงานจะนับจากประเภทตำแหน่งงาน (Role) ที่ใช้ดำเนินการ จำนวนการส่งต่องานระหว่างตำแหน่งงานจะนับจากผลิตภัณฑ์งาน (Work Products) ที่ได้จากการทำงานในตำแหน่งต่าง ๆ และเวลาที่ใช้ดำเนินการเปลี่ยนแปลงคำนวณจากเวลาการทำงาน (Working Hour) ทั้งหมดที่ใช้ในการดำเนินการกับการเปลี่ยนแปลงรายการที่ 2 โดยทั้งสามค่านี้ได้จากข้อมูลที่ถูกบันทึกลงในตารางการบันทึกเวลาดังรูปที่ ค.3 จากนั้นคำนวณหาค่าความกว้างของกระบวณการทำงานจากสมการ (6) คำนวณค่าความซับซ้อนของการเปลี่ยนแปลงจากสมการ (5) และคำนวณค่าความคล่องตัวจากสมการ (3)

ตารางที่ 5.21 การคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 2 (Non Service)

|  |  |
| --- | --- |
| ค่าระดับชั้นในเชิงสถาปัตยกรรม | 4 |
| จำนวนตำแหน่งงาน | 3 |
| จำนวนการส่งต่องานระหว่างตำแหน่งงาน | 2 |
| ค่าความกว้างของกระบวนการทำงาน (6) | 3 + 2 = 5 |
| ค่าความซับซ้อนของโค้ด | 2,746 |
| ค่าความซับซ้อนของการเปลี่ยนแปลง (5) | 4 \* 5 \* 2,746 = 54,920 |
| เวลาที่ใช้ดำเนินการเปลี่ยนแปลง | 25 ชั่วโมง |
| ค่าความคล่องตัว (3) | 54,920 / 25 = 2,196.80 |

การคำนวณค่าความคล่องตัวของระบบบันทึกการลาแบบอิงบริการจากการเปลี่ยนแปลงรายการที่ 2 สามารถแสดงได้ดังตารางที่ 5.22 โดยที่ค่าระดับชั้นในเชิงสถาปัตยกรรมได้จากจำนวนระดับชั้นที่จะได้รับผลกระทบจากการเปลี่ยนแปลงของตารางที่ 5.15 ค่าความซับซ้อนของโค้ดได้จากตารางที่ 5.17 สำหรับค่าของจำนวนตำแหน่งงานจะนับจากประเภทตำแหน่งงาน (Role) ที่ใช้ดำเนินการ จำนวนการส่งต่องานระหว่างตำแหน่งงานจะนับจากผลิตภัณฑ์งาน (Work Products) ที่ได้จากการทำงานในตำแหน่งต่าง ๆ และเวลาที่ใช้ดำเนินการเปลี่ยนแปลงคำนวณจากเวลาการทำงาน (Working Hour) ทั้งหมดที่ใช้ในการดำเนินการกับการเปลี่ยนแปลงรายการที่ 2 โดยทั้งสามค่านี้ได้จากข้อมูลที่ถูกบันทึกลงในตารางการบันทึกเวลาดังรูปที่ ค.4 จากนั้นคำนวณหาค่าความกว้างของกระบวณการทำงานจากสมการ (6) คำนวณค่าความซับซ้อนของการเปลี่ยนแปลงจากสมการ (5) และคำนวณค่าความคล่องตัวจากสมการ (3)

ตารางที่ 5.22 การคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 2 (Service)

|  |  |
| --- | --- |
| ค่าระดับชั้นในเชิงสถาปัตยกรรม | 5 |
| จำนวนตำแหน่งงาน | 3 |
| จำนวนการส่งต่องานระหว่างตำแหน่งงาน | 4 |
| ค่าความกว้างของกระบวนการทำงาน (6) | 3 + 4 = 7 |
| ค่าความซับซ้อนของโค้ด | 2,728 |
| ค่าความซับซ้อนของการเปลี่ยนแปลง (5) | 5 \* 7 \* 2,728= 95,480 |
| เวลาที่ใช้ดำเนินการเปลี่ยนแปลง | 28 ชั่วโมง |
| ค่าความคล่องตัว (3) | 95,480 / 28 = 3,410 |

จากการคำนวณความคล่องตัวจากการเปลี่ยนแปลงรายการที่ 2 ของระบบบันทึกการลาแบบไม่อิงบริการและแบบอิงบริการสามารถสรุปและเปรียบเทียบผลการคำนวณระหว่างแบบจำลองความคล่องตัวของ [4] และของโครงงานนี้ได้ดังตารางที่ 5.23

ตารางที่ 5.23 ตารางสรุปการคำนวณค่าความคล่องตัวของการเปลี่ยนแปลงรายการที่ 2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ระบบบันทึกการลา | ระยะเวลา (ชั่วโมง) | จำนวนระดับชั้นในเชิงสถาปัตยกรรม | จำนวนตำแหน่ง | จำนวนการส่งต่องาน | ความซับซ้อนของโค้ด(MVG) | ความคล่องตัว[4] (4) | ความคล่องตัวที่เพิ่มขึ้น[4] (%) | ความคล่องตัวของโครงงานนี้ (5) | ความคล่องตัวที่เพิ่มขึ้นของโครงงานนี้(%) |
| แบบไม่อิงบริการ | 25 | 4 | 3 | 2 | 2,746 | 0.8 | 56.25 | 2,196.80 | 55.23 |
| แบบอิงบริการ | 28 | 5 | 3 | 4 | 2,728 | 1.25 | 3,410.00 |

เมื่อพิจารณาค่าความคล่องตัวของสองระบบในการดำเนินการกับการเปลี่ยนแปลงรายการที่ 2 ซึ่งเป็นการเปลี่ยนแปลงในลักษณะการแก้ไขระบบเดิมให้สนับสนุนการทำงานของผู้ใช้และจากตารางที่ 5.23 พบว่าการเปลี่ยนแปลงนี้ส่งผลกระทบหลายระดับชั้นของสถาปัตยกรรม โดยเฉพาะในส่วนของระบบอิงบริการซึ่งมีการแก้ในส่วนของการพัฒนาของเซอร์วิซการลาและเซอร์วิซพนักงาน ทำให้เวลาถูกใช้ไปกับการจัดสร้าง (Build) ติดตั้ง (Deploy) และทดสอบเซอร์วิซส่งผลให้มีค่าระยะเวลาที่ใช้ ค่าของจำนวนระดับชั้นในเชิงสถาปัตยกรรมที่ถูกกระทบ และจำนวนการส่งต่องานมากกว่าระบบแบบไม่อิงบริการ สำหรับค่าความซับซ้อนของโค้ดของระบบอิงบริการมีค่าน้อยกว่าระบบไม่อิงบริการเนื่องจากการเรียกใช้เซอร์วิซของเมท็อดทำให้มีจำนวนโค้ดที่น้อยกว่าและทำให้ค่าความซับซ้อนของโค้ดน้อยลงตามไปด้วย ถึงแม้ว่าระบบแบบอิงบริการใช้เวลามากกว่าระบบแบบไม่อิงบริการแล้ว เมื่อพิจารณาค่าความคล่องตัวที่คำนวณได้จาก [4] และจากโครงงานนี้พบว่าระบบอิงบริการยังมีค่าความคล่องตัวที่สูงกว่าระบบแบบไม่อิงบริการเนื่องมาจากระยะเวลาที่ใช้ของระบบอิงบริการเป็นการดำเนินการกับค่าความซับซ้อนของการเปลี่ยนแปลงที่สูงกว่ามากเมื่อดูข้อมูลจากตารางที่ 5.22 เทียบกับตารางที่ 5.21

## [สรุปผลการทดลอง](#สารบัญ)

จากผลการทดลองของการดำเนินการกับการเปลี่ยนแปลง2รายการสามารถสรุปได้เป็นประเด็นต่าง ๆ ดังนี้

### ระดับชั้นเซอร์วิซที่เพิ่มเข้ามาในระบบทำให้ระบบมีความซับซ้อนมากขึ้น เมื่อมีการเปลี่ยนแปลงที่เกิดขึ้นกับระดับเซอร์วิซไม่ว่าจะเป็นการเพิ่มฟังก์ชันหรือมีการแก้ไขในส่วนการพัฒนาของเซอร์วิซ จะเห็นได้จากค่าความซับซ้อนของการเปลี่ยนแปลงของระบบอิงบริการในตารางที่ 5.20 และตารางที่ 5.23 มีค่าที่สูงกว่าระบบแบบไม่อิงบริการ

### การเปลี่ยนแปลงที่เกิดขึ้นกับระดับชั้นเซอร์วิซอาจบ่งบอกว่าการออกแบบเซอร์วิซของระบบนั้นยังไม่ดีพอ การออกแบบเซอร์วิซที่ดีควรจะออกแบบให้สามารถนำกลับมาใช้ใหม่และครอบคลุมการทำงานในบริบทต่าง ๆ ของโดเมนทางธุรกิจนั้น ๆ

### ค่าความคล่องตัวที่วัดจากแบบจำลองการวัดความคล่องตัวที่เสนอจากโครงงานนี้และจาก [4] ให้ผลลัพธ์ที่เป็นไปในทางเดียวกัน คือระบบอิงบริการมีความคล่องตัวกว่าระบบแบบไม่อิงบริการซึ่งจะเห็นได้จากเปอร์เซ็นต์ความคล่องตัวที่เพิ่มขึ้นในตารางที่ 5.20 และตารางที่ 5.23 มีค่าที่ใกล้เคียงกัน

### ปัจจัยความซับซ้อนของโค้ดที่เพิ่มเข้าในการพิจารณาความซับซ้อนของการเปลี่ยนแปลงช่วยทำให้เราเข้าใจระบบมากขึ้น จะเห็นได้จากค่าความซับซ้อนของโค้ดจากตารางที่ 5.20 มีค่าน้อยกว่าในตารางที่ 5.23 เนื่องมาจากการเปลี่ยนแปลงรายการที่ 1 มีมอดูลของโปรแกรมที่จะถูกกระทบน้อยกว่าการเปลี่ยนแปลงรายการที่ 2 ที่จะกระทบกับหลายมอดูลของโปรแกรม

### ระบบอิงบริการมีความเหมาะสมและคุ้มค่ากับการพัฒนา เมื่อพิจารณาจากระยะเวลาที่ใช้ในการดำเนินการจากตารางที่ 5.20 ระบบอิงบริการใช้เวลาน้อยกว่าแบบไม่อิงบริการและแม้ว่าในตารางที่ 5.23 ระยะเวลาที่ใช้ของระบบอิงบริการจะมีค่ามากกว่าแต่เมื่อพิจารณาเทียบกับค่าความซับซ้อนของการเปลี่ยนแปลงแล้วยังให้ค่าความคล่องตัวที่มากกว่าระบบแบบไม่อิงบริการ

# 

# [บทที่ 6 บทสรุปโครงงาน อภิปรายผล และข้อเสนอแนะ](#สารบัญ)

## สรุปผลโครงงานมหาบัณฑิต

โครงงานมหาบัณฑิตนี้ได้นำเสนอการวัดความคล่องตัวของระบบที่ไม่ได้อิงบริการเทียบกับระบบอิงบริการที่มีฟังก์ชันการทำงานเหมือนกัน โดยการวัดจะขึ้นกับความเร็วในการดำเนินการกับการเปลี่ยนแปลง และความซับซ้อนของการเปลี่ยนแปลง จากผลการทดลองพบว่าค่าความคล่องตัวที่ได้จากการแบบจำลองการวัดความคล่องตัวของ [4] และแบบจำลองการวัดความคล่องตัวของโครงงานนี้ที่ได้เพิ่มการพิจารณาความซับซ้อนของโค้ดเข้าไปด้วยนั้นได้ให้ผลที่เป็นไปในทิศทางเดียวกันคือระบบที่อิงบริการนั้นมีความคล่องตัวมากกว่า นอกจากนี้ยังพบว่าความคล่องตัวสามารถถูกปรับปรุงให้ดีขึ้นกว่าเดิมถ้าหากมีการออกแบบเซอร์วิซที่ดีให้สามารถนำกลับมาใช้งานใหม่ได้ และครอบคลุมการทำงานในบริบทต่าง ๆ ของโดเมนทางธุรกิจนั้น ๆ จะส่งผลให้เมื่อมีการเปลี่ยนแปลงเกิดขึ้น ระบบจะสามารถยืดหยุ่นปรับเข้ากับการเปลี่ยนแปลงนั้น ๆ ได้อย่างรวดเร็ว

## ปัญหาและข้อจำกัดในการทำโครงงาน

จากการดำเนินงานโครงงานนี้ พบปัญหาและข้อจำกัดของระบบ ดังต่อไปนี้

* + 1. เนื่องจากไม่มีหน่วยทดลองที่เป็นตัวแทนของระบบอิงบริการ ผู้วิจัยจำเป็นต้องจัดทำขึ้นใหม่โดยใช้ระบบบันทึกการลาที่นำมาเป็นหน่วยทดลองตัวแทนของระบบที่ไม่อิงบริการมาเป็นตัวตั้งต้นในการจัดทำ จึงเสียเวลาไปกับขั้นตอนนี้อยู่พอสมควรเพื่อทำให้ระบบอิงบริการที่จัดทำขึ้นสามารถใช้งานได้จริง
    2. เนื่องด้วยเวลาในการดำเนินโครงงานมีจำกัดทำให้ส่วนของการออกแบบเซอร์วิซของระบบบันทึกการลานั้นอาจจะไม่ดีพอตามหลักการออกแบบเซอร์วิซ [2]
    3. ในส่วนของการทดสอบผู้วิจัยทำหน้าที่ในสามบทบาทคือเป็นทั้ง นักวิเคราะห์ระบบ ผู้พัฒนาโปรแกรม และผู้ทดสอบระบบ ซึ่งโดยหลักปฏิบัติแล้วในการทำงานจริงผู้ทดสอบระบบจะเป็นคนละคนกับผู้พัฒนาโปรแกรม เพราะถ้าหากผู้พัฒนาเป็นผู้ทดสอบระบบเองอาจจะมองข้ามการทดสอบและการหาข้อบกพร่องของระบบในบางกรณี ทำให้เวลาที่ใช้กับการทดสอบไม่มากพอ อาจจะส่งผลต่อระยะเวลารวมที่ใช้ในการดำเนินการกับการเปลี่ยนแปลงของการทดลองคลาดเคลื่อน

## ข้อเสนอแนะ

การทดลองในโครงงานมหาบัณฑิตนี้เป็นเพียงตัวอย่างประโยชน์ของแนวคิดสถาปัตยกรรมเชิงบริการในแง่ของความคล่องตัว และปัจจัยต่าง ๆ ที่ส่งผลต่อค่าความคล่องตัวในแบบจำลองการวัดความคล่องตัว ผู้วิจัยเห็นว่าแนวทางการทดลองยังสามารถปรับปรุงได้เช่นการนำหลักการทางสถิติมาใช้ร่วมด้วย กล่าวคือการทดลองควรจะกระทำกับระบบแอปพลิเคชันหลายระบบ และมีประเภทของการเปลี่ยนแปลงที่หลากหลาย ซึ่งจะช่วยให้สามารถทดสอบความสัมพันธ์ระหว่างความคล่องตัว กับปัจจัยต่าง ๆ และการเปลี่ยนแปลงลักษณะต่าง ๆ อีกทั้งควรทำการตรวจสอบด้วยว่าค่าความซับซ้อนของโค้ดนั้นมีนัยสำคัญทางสถิติต่อแบบจำลองการวัดความคล่องตัวหรือไม่ นอกจากนี้ผู้วิจัยพบว่าในส่วนการออกแบบระบบอิงบริการยังสามารถปรับปรุงการออกแบบให้ดีขึ้นกว่าเดิมได้ และจะทำให้ความคล่องตัวของระบบมีการปรับปรุงขึ้นด้วยเช่นกัน

# [รายการอ้างอิง](#สารบัญ)

1. Gabhart, K., and Bhattacharaya, B. Service Oriented Architecture Field Guide for Executives. Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.
2. Erl, T. SOA Principles of Service Design. Prentice Hall, 2008.
3. McCabe, T.J. A Complexity Measure. Software Engineering, IEEE Transactions, pp. 308-320. IEEE, 1976.
4. Malik, N. Measuring the agility of a SOA approach [Online]. 2007. Available from: http://blogs.msdn.com/b/nickmalik/archive/2007/12/14/measuring-the-agility-of-a-soa-approach.aspx [2012, March 12].
5. Shawky, D.M., and Ali A.F. A Practical Measurement for the Agility of Software Development Processes. Computer Technology and Development (ICCTD), 2010 2nd International Conference, pp. 230-234. IEEE, 2010.
6. Arteta, B.M., and Giachetti, R.E. A measure of agility as the complexity of the enterprise system. Robotics and Computer-Integrated Manufacturing 20, pp. 495-503. Science Direct, 2004.
7. Datta, S. Agility Measurment Index – A Metric for the Crossroads of Software Development Methodologies. ACM-SE 44 Proceedings of the 44th annual Southeast regional conference, pp. 271-273. New York: ACM, 2006.
8. Mansouri, M., Ganguly, A., and Mostashari, A. Evaluating Agility in Extended Enterprise Systems: A Transportation Network case. American J. of Engineering and Applied Sciences, pp. 142-152. Science Publications, 2011.
9. Erl, T. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, 2005.
10. The Apache Software Foundation. Apache Axis2 [Online]. 2011. Available from: http://axis.apache.org/axis2/java/core/index.html [2012, March 13].
11. LocMetrics [Online]. 2007. Available from: http://www.locmetrics.com [2012, March 13].

# [ภาคผนวก](#สารบัญ)

# [ภาคผนวก ก แผนภาพการออกแบบระบบการเปลี่ยนแปลงรายการที่ 1](#สารบัญ)



รูปที่ ก.1 แผนภาพกิจกรรมของการเปลี่ยนแปลงรายการที่ 1 (Non Service)



รูปที่ ก.2 แผนภาพกิจกรรมของการเปลี่ยนแปลงรายการที่ 1 (Service)



รูปที่ ก.3 แผนภาพลำดับของการเปลี่ยนแปลงรายการที่ 1 (Non Service)



รูปที่ ก.3 แผนภาพลำดับของการเปลี่ยนแปลงรายการที่ 1 (Service)

# ภาคผนวก ข แผนภาพการออกแบบระบบการเปลี่ยนแปลงรายการที่ 2



รูปที่ ข.1 แผนภาพกิจกรรมการปรับปรุงประเภทการทำงานของพนักงาน (Non Service)



รูปที่ ข.2 แผนภาพกิจกรรมการปรับปรุงประเภทการทำงานของพนักงาน (Service)



รูปที่ ข.3 แผนภาพกิจกรรมการบันทึกการลา (Non Service)



รูปที่ ข.4 แผนภาพกิจกรรมการบันทึกการลา (Service)



รูปที่ ข.5 แผนภาพลำดับของการบันทึกการลา (Non Service)

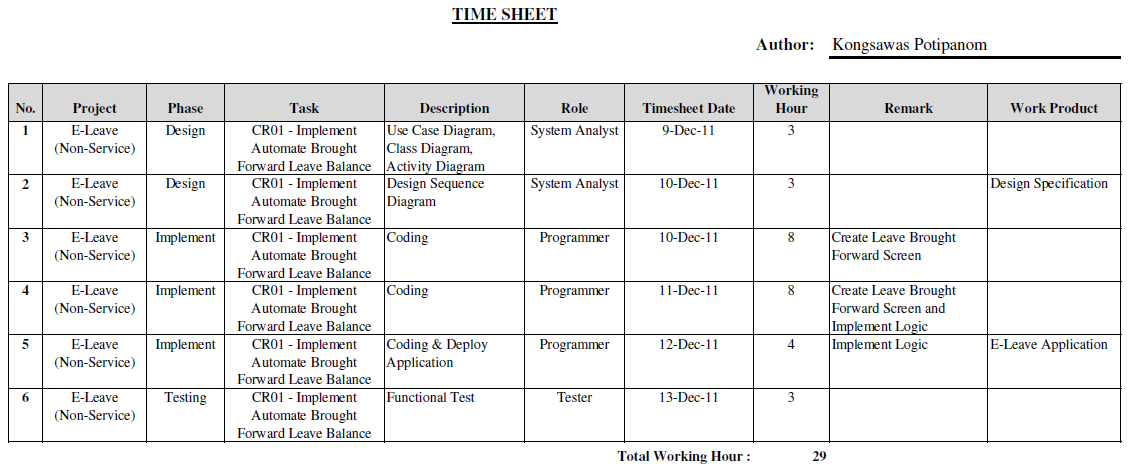


รูปที่ ข.6 แผนภาพลำดับของการบันทึกการลา (Service)

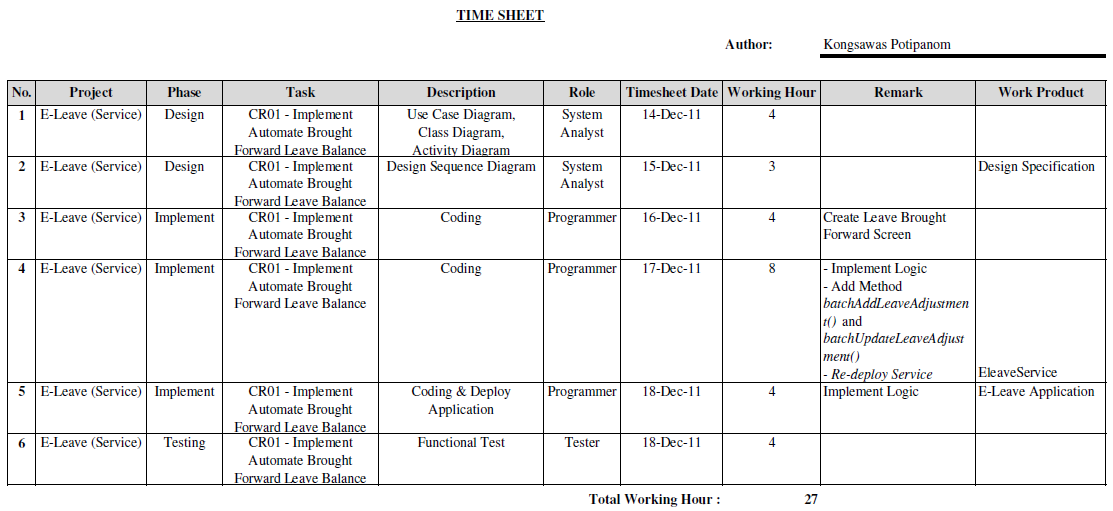


รูปที่ ข.7 แผนภาพลำดับของการคำนวณวันลาที่ใช้

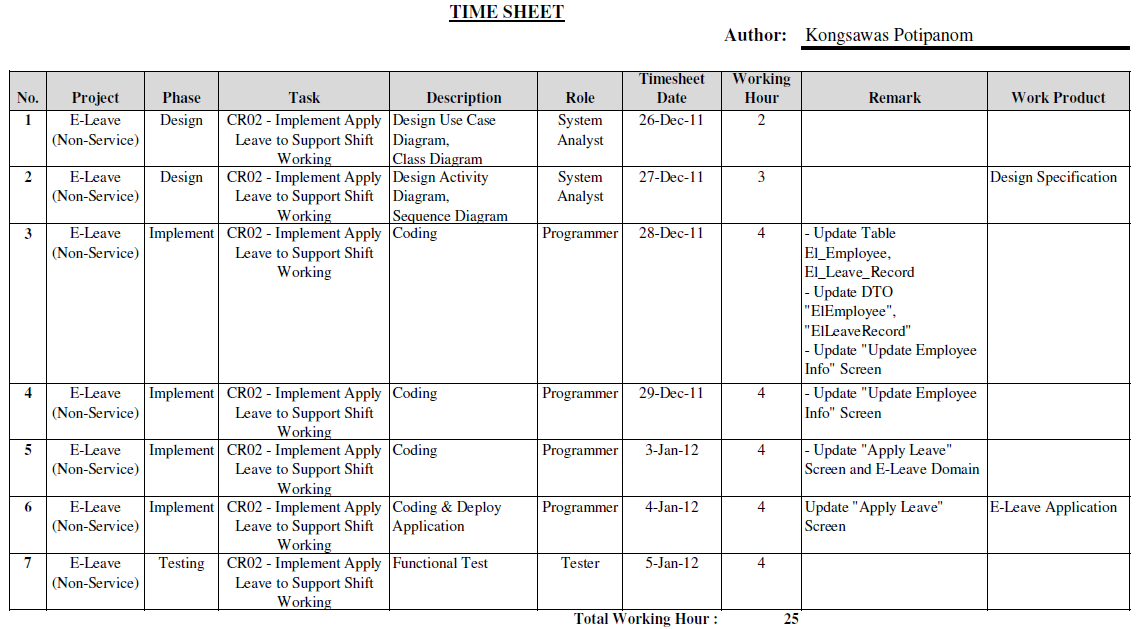
# ภาคผนวก ค ตารางบันทึกเวลาการทำงาน



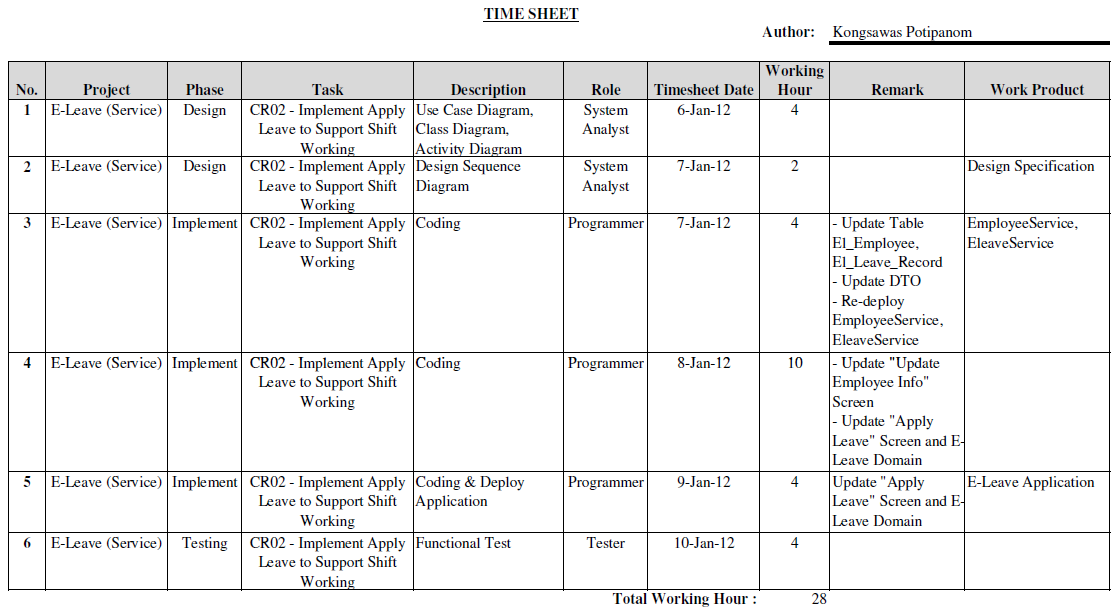
รูปที่ ค.1 การบันทึกเวลาการทำงานการเปลี่ยนแปลงรายการที่ 1 (Non Service)



รูปที่ ค.2 การบันทึกเวลาการทำงานการเปลี่ยนแปลงรายการที่ 1 (Service)



รูปที่ ค.3 การบันทึกเวลาการทำงานการเปลี่ยนแปลงรายการที่ 2 (Non Service)



รูปที่ ค.4 การบันทึกเวลาการทำงานการเปลี่ยนแปลงรายการที่ 2 (Service)

# ประวัติผู้เขียนโครงงานมหาบัณฑิต

นายคงสวัสดิ์ โพธิพนม เกิดเมื่อวันที่ 29 มีนาคม พ.ศ. 2525 ที่จังหวัดเชียงใหม่สำเร็จการศึกษาหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ ในปีการศึกษา 2547 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2553