# ระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทดสอบ

# DEFECT TRACKING MANAGEMENT SYSTEM


โดย

ศราวุธ โสพสกุลางกูร

SARAWUT SOROTKULANGKOON

อาจารย์ที่ปรึกษา

ดร.สิงหะ ฉวีสุข



รายงานนี้เป็นส่วนหนึ่งของวิชาการศึกษาอิสระ 2
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ภาคเรียนที่ 2 ปีการศึกษา 2554

# DEFECT TRACKING MANAGEMENT SYSTEM

SARAWUT SOROTKULANGKOON

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS OF THE COURSE

INDEPENDENT STUDY 2

MASTER OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/ 2012

| | |
|---|---|
| **ห้อข้อ** | ระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทดสอบ |
| **นักศึกษา** | นายศราวุธ โสพสกุลางกูร |
| **รหัสนักศึกษา** | 53660754 |
| **ปริญญา** | วิทยาศาสตรมหาบัณฑิต |
| **สาขาวิชา** | เทคโนโลยีสารสนเทศ |
| **แขนงวิชา** | เทคโนโลยีสารสนเทศและการจัดการ |
| **ปีการศึกษา** | 2554 |
| **อาจารย์ที่ปรึกษา** | ดร.สิงหะ ฉวีสุข |

## บทคัดย่อ


ในการดำเนินการจัดการความผิดพลาดของโปรแกรมระหว่างการทดสอบของ บริษัทที่ให้บริการด้านโทรคมนาคมการสื่อสารแห่งหนึ่ง (True Corporation Ltd,.) มีขั้นตอนการปฏิบัติงานโดยบันทึกข้อความผิดผลาดของโปรแกรมไว้ในเอกสาร Excel ซึ่งมีข้อมูลรายละเอียดไม่มากนัก และทำการส่งรายงานให้กับหัวหน้า หรือทุกคนที่เกี่ยวข้องกับโปรเจคทุก ๆ วัน ตลอดระยะเวลาที่ดำเนินโปรเจค ผู้ศึกษาได้สังเกตเห็นว่าการบันทึกข้อผิดผลาดระหว่างการทดสอบ และการส่งเอกสารในรูปแบบของไฟล์เอกสาร รวมถึงการติดตามตรวจสอบขั้นตอนในการปฏิบัติงาน อาจจะทำให้เกิดความผิดผลาดในข้อมูลนั้นได้

ดังนั้นจึงมีแนวคิดที่จะนำระบบสารสนเทศมาแก้ปัญหาที่เกิดขึ้นโดยนำยูเอ็มแอลช่วยในการวิเคราะห์และออกแบบ มีการจัดเก็บข้อมูลแบบ อาร์ดีบีเอ็มเอสไว้ที่ศูนย์กลาง มีการติดต่อสื่อสารโดยใช้เทคโนโลยีเว็บแอพพลิเคชั่นผ่านเครือข่ายอินทราเน็ทภายในองค์กร

ทั้งนี้ระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทดสอบจะช่วยให้การทำงานเป็นมาตรฐาน เพิ่มประสิทธิภาพการติดตามตรวจสอบสถานะของความผิดพลาดที่เกิดขึ้นระหว่างการทดสอบ มีการทำงานที่สะดวกขึ้น รวดเร็วขึ้น จึงส่งผลดีให้กับผู้ใช้งานระบบและองค์กรต่อไป

| | |
|---|---|
| **Title** | Defect Tracking Management System |
| **Student** | Mr. Sarawut Sorotkulangkoon |
| **Student ID.** | 53660754 |
| **Degree** | Master of Science |
| **Program** | Information Technology |
| **Mojor** | Information Technology and Management |
| **Academic Year** | 2011 |
| **Advisor** | Dr.Singha Chaveesuk |


## ABSTRACT

In a traditional of management the program between test of telecommunications company, which a service provider communication (True Corporation Ltd,.). Operating procedures by the gradient of the error message is logged in an Excel document which detailed data are available. And submit a report to the head. Or anyone involved with the project every day for the duration of the project. The study noted that the slope of the log error during the test. And documents in the file. The monitoring procedures in operation. May have caused the error gradient in the data.

Therefore, the idea is to bring information technology to solve the problem by the UML into analysis and design. Save the data have the RDBMS in the center. Communication using web applications via the intranet within the organization.

The management system failures during the test program will run as a standard. Performance monitoring of the status of errors occurring during the test. The work easier and faster results to the user and the system further.

# กิตติกรรมประกาศ

โครงการนี้สำเร็จได้เป็นอย่างดีด้วยความกรุณาจากท่านอาจารย์ที่ปรึกษา ดร.สิงหะ ฉวีสุข ที่กรุณาเสียสละเวลาอันมีค่าให้คำปรึกษาและคำแนะนำแก่ข้าพเจ้า ช่วยตรวจสอบแก้ไขข้อบกพร่องของโครงการนี้ ตลอดจนให้ความรู้และข้อคิดเห็นที่เป็นประโยชน์อย่างยิ่งต่อโครงการ ข้าพเจ้ารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ กรรมการสอบหัวข้อโครงการที่ได้กรุณาให้คำแนะนำและชี้แนะจนทำให้โครงการนี้สำเร็จลงได้ในที่สุด

ขอขอบพระคุณคณาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้ ตลอดจนให้คำแนะนำและคำปรึกษาทั้งทางวิชาการ การดำเนินชีวิตและกรุณาถ่ายทอดประสบการณ์ที่ดีอันมีค่าให้แก่ข้าพเจ้า

ขอขอบพระคุณ หน่วยงาน Test Management บริษัท ทรู คอร์ปอเรชั่น จำกัด (มหาชน) ที่ช่วยสนับสนุนการทำโครงการ ช่วยให้คำแนะนำแนวทาง และข้อเสนอแนะที่เป็นประโยชน์ยิ่งต่อการทำโครงการ อีกทั้งให้ข้อมูลสำหรับการจัดทำโครงการด้วยดีเสมอมา

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่านที่ให้การช่วยเหลือ สนับสนุน คอยให้คำปรึกษาและคำแนะนำที่เป็นประโยชน์ยิ่ง ตลอดจนคอยให้กำลังใจ แบ่งปันน้ำใจ ร่วมทุกข์ร่วมสุขและมีกัลยาณมิตรที่ดีต่อกันตลอดมา

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ มารดา และครอบครัวอันเป็นที่รักยิ่งของข้าพเจ้าที่เป็นกำลังใจ แรงบันดาลใจ ส่งเสริมและสนับสนุนในทุกเรื่อง จนทำให้ข้าพเจ้าสามารถทำโครงการนี้ให้สำเร็จลุล่วงได้ด้วยดี

สำหรับคุณงานความดี และประโยชน์อันพึงมาจากโครงการนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

ศราวุธ โสพสกุลางกูร

# สารบัญ

# สารบัญ (ต่อ)

# สารบัญตาราง

# สารบัญรูป

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาของโครงงาน

จากหน่วยงานเดิมที่ผู้ศึกษาได้เคยปฏิบัติหน้าที่มา เป็นบริษัทเกี่ยวกับโทรคมนาคมการสื่อสาร (True Corporation Ltd,.) และปฏิบัติงานในหน่วยงาน Test Management ซึ่งมีหน้าที่คือทำการตรวจสอบโปรแกรม หรือระบบใหม่ ๆ ที่เกิดจากการร้องขอจากผู้ปฏิบัติงานภายในองค์กร ที่เรียกกันว่า ยูสเซอร์ (User) เพื่อหาความผิดผลาดของโปรแกรม หรือที่เรียกกันว่า บัก (Bug) โดยมีขั้นตอนคราว ๆ คือ หลังจากผู้พัฒนาระบบ (Developer) ได้พัฒนาระบบตามคำร้องขอจาก User แล้ว หลังจากนั้นจะส่งมอบให้ทาง Test Management เป็นผู้ตรวจสอบโปรแกรม ซึ่งมีผู้ตรวจสอบโปรแกรม หรือที่เรียกว่า เทสเตอร์ (Tester) ทำหน้าที่ตรวจสอบและหาความผิดผลาดทั้งหมดที่อาจจะเกิดขึ้นหรือไม่เกิดขึ้นก็ได้ โดยบันทึกข้อความผิดผลาดของโปรแกรมไว้ในเอกสาร Excel โดยมีข้อมูลรายละเอียดไม่มากนัก และจะทำการส่งรายงานให้กับหัวหน้า หรือทุกคนที่เกี่ยวข้องกับโปรเจค (Project) นั้น ๆ ในทุก ๆ วันตลอดระยะเวลาที่ดำเนินโปรเจคนั้น ซึ่งงานหลักยังมีอีกมากมาย อาทิเช่น การออกแบบในส่วนของการทำเทสโปรแกรมนั้น การวางแผนและกำหนดระยะเวลาตลอดโปรเจค ซึ่ง Tester ทุกคนจะต้องสามารถทำงานได้เองทั้งหมด ผู้ศึกษาได้สังเกตเห็นว่าการบันทึกข้อผิดผลาด และการส่งเอกสารในรูปแบบของไฟล์เอกสาร อาจจะทำให้เกิดความผิดผลาดในข้อมูลนั้นได้ ทางผู้ศึกษาจึงนำแนวความคิดนี้มาพัฒนาให้รู้ในรูปแบบของ Web Application เพื่อกำหนดสิทธิ์ในการเข้าใช้งาน ป้องกันความผิดผลาดที่อาจเกิดจาก Tester เอง หรือเกิดจากบุคคลอื่นที่เกี่ยวข้องกับโปรเจค สามารถติดตามตรวจสอบขั้นตอนในการปฏิบัติงาน สามารถเข้าถึงได้ทุกสถานที่ ทุกเวลา ที่มีข้อมูลที่อัพเดรต (Update) ล่าสุดตลอดเวลา โดยที่กล่าวมาทั้งหมดจะควบคุมโดยขั้นตอนการไหลของข้อมูล (Work-Flow)

## 1.2 ปัญหาของโครงงาน

ระบบที่ผู้ศึกษานำมาเสนอเป็นระบบเกี่ยวกับ การจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบระบบ (Defect Tracking Management System) ซึ่งระบบงานเดิมของหน่วยงานทำโดยการบันทึกข้อมูลความผิดพลาดที่เกิดขึ้นลงในเอกสารรูปแบบของ Excel และจัดเก็บไว้ในฐานข้อมูลของหน่วยงาน (Share driver) ขณะเดียวกันผู้ใช้งานคนอื่นก็สามารถเปิดเอกสารนั้นเพื่อดู ปรับเปลี่ยน แก้ไข หรือลบข้อมูล ที่เก็บไว้ใน Share driver ได้ และสาเหตุอีกประการคือ ตรวจสอบหรือติดตามขั้นตอนการทำงานในแต่ละขั้นตอน เป็นไปด้วยความยากลำบาก ด้วยสาเหตุดังกล่าวทำ

ให้ผู้ศึกษาเห็นข้อบกพร่องในเรื่องการบันทึก จัดเก็บเอกสาร และการตรวจสอบหรือติดตามขั้นตอนการทำงานเป็นเรื่องยาก ซึ่งความสามารถระบบมีดังนี้

1. สามารถบันทึกข้อมูลการใช้งานของผู้ใช้งาน (User) เพื่อทำการตรวจสอบหรือติดตามขั้นตอนการทำงานได้ทุกขั้นตอน (โดยใช้ Work-Flow ควบคุมการทำงานของระบบ)
2. ป้องกันข้อมูล และจัดเก็บข้อมูลไว้ในระบบฐานข้อมูลโดยต้องมีการร้องขอเพื่อเข้าใช้งาน (Login) ซึ่งสามารถกำหนดสิทธิ์ในการใช้งานได้

## 1.3 วัตถุประสงค์ของโครงงาน

วิเคราะห์และออกแบบระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบระบบโดยใช้ภาษา UML(Unified Modeling Language) และแบบจำลองความสัมพันธ์ระหว่างเอนทิตี (Entity Relationship Model) เป็นเครื่องมือสามารถออกแบบระบบงานให้สามารถตอบสนองกับความต้องการภายในองค์กร โดยมีวัตถุประสงค์ของโครงการมีดังนี้

1. ลดการปรับเปลี่ยน แก้ไขเอกสาร หรือลบข้อมูล จากผู้ที่ใช้งานร่วมกันในหน่วยงาน
2. ตรวจสอบ ขั้นตอนการทำงานในแต่ละขั้นตอนได้อย่างใกล้ชิด และทราบความก้าวหน้าของโครงการได้สะดวกและรวดเร็วยิ่งขึ้น
3. ปรับเปลี่ยนระบบการทำงานให้มีประสิทธิภาพมากขึ้น ให้เป็นตาม Workflow ที่กำหนดไว้

## 1.4 ขอบเขตและขั้นตอนการศึกษา

ระบบที่ทำการพัฒนานี้ จะเป็นเครื่องมือที่ช่วยให้การทำงานของบุคลากรในหน่วยงาน Test Management สายงานเทคโนโลยีสารสนเทศ สามารถดำเนินงานตาม Workflow ที่ได้ทำการออกแบบไว้ตามแผนงาน เพื่อบรรลุวัตถุประสงค์ของหน่วยงานและองค์กร ซึ่งการทำงานทั้งหมดจะเป็นการใช้ติดต่อกับผู้ใช้ผ่านทางเวบเบราเซอร์ (Web Browser) และนำเสนอในรูปแบบของเวบเพจ (Web Page) โดยจะเน้นเรื่องการตรวจสอบสถานะของความผิดผลาดตั้งแต่พบ จนกระทั้งตรวจสอบแล้วไม่พบข้อผิดผลาดแล้ว โดยระบบดังกล่าวที่พัฒนาขึ้นภายใต้การทำงานบนเครือข่ายอินทราเน็ต (Intranet) ในบริษัทเท่านั้น เพื่อให้เห็นถึงกระบวนการจัดการภายในโดยมีขั้นตอนการศึกษาดังนี้

1. ศึกษาการทำงานภายในหน่วยงาน Test Management โดยศึกษาลักษณะของการจัดการบันทึกความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบระบบ โดยใช้วิธีสอบถามจากผู้ใช้งานระบบงานเดิม รวมถึงเอกสารที่จำเป็นในการดำเนินงานนั้น

2. สอบถามปัญหา และเก็บรวมรวมความต้องการของพนักงานในหน่วยงานเกี่ยวกับขั้นตอนการจัดการบันทึกความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบระบบ
3. วิเคราะห์ปัญหาและข้อจำกัดอันเนื่องมาจากกระบวนการทำงานในปัจจุบันวิเคราะห์และออกแบบเชิงวัตถุ (Object Oriented Analysis & Design) โดยนำภาษา UML (Unified Modeling Language) ซึ่งเป็นภาษาที่ใช้อธิบายโมเดลและสร้างมุมมองต่าง ๆ ให้กับการพัฒนาระบบงานวิเคราะห์ความต้องการของระบบงานใหม่ เพื่อให้สอดคล้องกับความต้องการของผู้ใช้งาน และเพื่อแก้ไขปัญหาระบบงานในปัจจุบันให้มีประสิทธิภาพมากยิ่งขึ้น
4. ศึกษาข้อมูลของการนำเอาระบบสำนักงานอัตโนมัติมาใช้ในการสร้างระบบทั้งทางอินเทอร์เน็ต หนังสือ
5. สรุปผลที่ได้ทำการศึกษามาทั้งหมด รวมถึงกำหนดทางเลือกเพื่อการตัดสินใจที่ดีที่สุด

## 1.5 เครื่องมือที่ใช้ในการพัฒนาระบบ

ในการพัฒนาระบบนี้ใช้เทคโนโลยี Web Database Application เพื่อก่อให้เกิดการทำงานร่วมกันระหว่างหน่วยงานอย่างมีประสิทธิภาพ ตลอดจนนำระบบฐานข้อมูลเชิงสัมพันธ์เพื่อใช้ในการบริหารและจัดการงานภายในขององค์กร โดยมีเครื่องมือต่าง ๆ ที่นำมาใช้ในการพัฒนาระบบดังนี้

1. ภาษาที่ใช้ในการพัฒนา Web Database Application ในโครงการ ใช้โปรแกรมภาษา Java
2. ซอฟแวร์ที่ใช้ในการพัฒนาได้แก่ Adobe Dreamweaver
3. ระบบปฏิบัติการ Windows XP Service Pack 3 พร้อมติดตั้งโปรแกรม AppServ เป็นตัวจำลอง Web Server และ Web Database เพื่อใช้ตรวจสอบความถูกต้องและสมบูรณ์ของระบบงาน
4. ระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ MySQL
6. โปรแกรม Web Brower Internet Explorer เวอร์ชั้น 6.0 ขึ้นไป

## 1.6 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน

เมื่อพัฒนาระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบระบบขึ้นแล้ว คาดว่าจะสามารถนำระบบนี้ไปใช้ในองค์กรได้ และประโยชน์ที่คาดหวังมีดังนี้

1. เพื่อจัดการกับปัญหาเรื่อง การบันทึก ปรับเปลี่ยน แก้ไข หรือลบข้อมูล

2. เพื่อตรวจสอบ ขั้นตอนการทำงานในแต่ละขั้นตอนได้อย่างใกล้ชิด และดูแลการดำเนินงานให้เกิดประสิทธิภาพสูงสุด โดยติดตามความก้าวหน้าของงานได้ผ่านเวบไซต์ และการวัดประสิทธิภาพในการทำงานของพนักงาน
3. เพื่อลดความเสี่ยงเนื่องจากการสูญหายของเอกสาร จาการเก็บไว้ในฐานข้อมูลของหน่วยงาน (Share Drive) หากเกิดความเสียหายของ disk อันเนื่องมาจากสาเหตุต่าง ๆ

# บทที่ 2
# ทฤษฎีที่เกี่ยวข้อง

การพัฒนาระบบสารสนเทศและระบบฐานข้อมูลสำหรับหน่วยงานภายในองค์กร จึงจำเป็นต้องศึกษาจากแนวความคิดและทฤษฎี เพื่อนำมาใช้เป็นแนวทางในการพัฒนาระบบ โดยการพัฒนาระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบระบบ ได้นำทฤษฎีต่างๆ มาใช้ ดังนี้

## 2.1 ทฤษฎีการทดสอบซอฟท์แวร์(Software Testing)

การทดสอบซอฟท์แวร์หมายถึง กระบวนการในการใช้งานหรือประเมินค่าซอฟท์แวร์ ทั้งการทำด้วยมือ หรือทำอย่างอัตโนมัติเพื่อที่จะตรวจสอบว่าซอฟท์แวร์เป็นไปตามความต้องการของซอฟท์แวร์ (Software Requirements) หรือเพื่อที่จะระบุความแตกต่างระหว่างผลลัพธ์ที่คาดหวัง (Expected Result) กับผลลัพธ์ที่ได้จริง (Actual Result) จากซอฟท์แวร์

เป้าหมายของกรทดสอบซอฟท์แวร์มีดังนี้

1. สามารถป้องกันความผิดพลาด (Defect) ไม่ให้เกิดขึ้นกับซอฟท์แวร์ได้
2. ถ้าการทดสอบไม่สามารถป้องกันความผิดพลาดไม่ให้เกิดขึ้นได้ก็ควรจะสามารถบอกได้ว่าความผิดพลาดนั้นจะส่งผลกระทบต่อซอฟท์แวร์อย่างไร
3. การทดสอบควรจะบอกแนวทางการแก้ไขความผิดพลาดที่ชัดเจน เพื่อให้สามารถแก้ไขความผิดพลาดได้อย่างง่ายดาย

การทดสอบซอฟท์แวร์ถือเป็นขั้นตอนที่สำคัญที่สุดขั้นตอนหนึ่งในกระบวนการพัฒนาซอฟท์แวร์ตามหลักวิศวกรรมซอฟท์แวร์การทดสอบซอฟท์แวร์จึงเป็นกระบวนการเพื่อช่วยให้ซอฟท์แวร์ที่พัฒนาขึ้นมามีความถูกต้อง, ความสมบูรณ์, ความปลอดภัย และมีคุณภาพที่ดีที่สุดโดยการทดสอบเป็นการทดลองใช้ซอฟท์แวร์อย่างมีแนวทาง โดยใช้ความรู้ทางด้านเทคนิคเพื่อให้สามารถระบุหรือค้นหาข้อผิดพลาด (Error) ของซอฟท์แวร์ที่อาจจะซ่อนอยู่ให้ปรากฏออกมาและสามารถระบุถึงแนวทางการเกิดปัญหา พร้อมทั้งสามารถระบุถึงสมมุติฐานของความผิดพลาดที่เกิดขึ้นได้ในการทดสอบซอฟท์แวร์หรือระบบงานจะถูกแบ่งออกเป็น 5 ระดับคือ

1. **Unit Testing**

Unit Testing เป็นการทดสอบ Product-Components ที่พัฒนาขึ้นมาเป็นราย Unit โดยแต่ละ Unit จะถูกพัฒนาขึ้นมาตามเอกสารแสดงการออกแบบสำหรับ Unit นั้นๆ ซึ่งในกรณีของObject-

Oriented Testing จะเป็นการทดสอบ Operation ต่างๆของ Class ตลอดจนทดสอบการทำงานร่วมกันของ Operation ทั้งหมดของ Class นั้นๆ

การสร้าง Test Case เพื่อทดสอบซอฟท์แวร์ในระดับ Unit นี้จะแบ่งออกเป็น 3 วิธี

**วิธีที่ 1 Black-Box Testing**

วิธีนี้เป็นการใช้Black-Box Technique ในการสร้าง Test Case เพื่อทดสอบ Function ของ Software Unit นั่นคือ Test Case ที่ได้จะทดสอบว่าซอฟท์แวร์ดังกล่าว มีการทำงานเป็นไปตามที่ต้องการหรือไม่โดยไม่สนใจว่าจะถูกพัฒนาขึ้นมาด้วยโครงสร้างการเขียนโปรแกรมแบบใด โดย Method ที่ใช้ในการสร้าง Test Case ในวิธีนี้นั้นจะใช้ Equivalence Partitioning, Boundary Value Analysis และ Usage Scenario Simulation เป็นหลัก โดยอาจเลือกใช้เพียงบางวิธีหรือทุกวิธีก็ได้ส่วน Method อื่นก็อาจสามารถนำมาใช้เพิ่มเติมได้ตามความเหมาะสม การออกแบบ Test Case ด้วยวิธีนี้สามารถทำได้จาก Pseudo-Code ของโปรแกรมที่มีการออกแบบไว้ได้เลยโดยไม่ต้องรอให้ Software Unit ที่จะทดสอบนั้นถูกสร้างขึ้นมาเป็น Source Code ก่อน

**วิธีที่ 2 White-Box Testing with Statement Coverage Testing**

วิธีนี้เป็นการใช้White-Box Technique ในการสร้าง Test Case เพื่อทดสอบ Structure ของ Source Code ของ Software Unit ตามหลักเกณฑ์ของ Statement Coverage นั่นคือ Test Case ที่ได้จะทดสอบทุก Statement ในโปรแกรมว่าโปรแกรมดังกล่าว ถูกพัฒนาขึ้นมาด้วยโครงสร้างการเขียนโปรแกรมที่ถูกต้องหรือไม่ การออกแบบ Test Case ด้วยวิธีนี้จะสามารถทำได้ก็ต่อเมื่อ Software Unit ที่จะทดสอบนั้นได้ถูกสร้างขึ้นมาเป็น Source Code แล้วเท่านั้น

**วิธีที่ 3 White-Boxes Testing with Basis Path Testing**

วิธีนี้เป็นการใช้White-Box Technique ในการสร้าง Test Case เพื่อทดสอบ Structure ของ Source Code ของ Software Unit โดยใช้หลักการของ Basis Path Testing นั่นคือ Test Case ที่ได้จะทดสอบทุก Independent Path ในโปรแกรมว่าโปรแกรมดังกล่าว ถูกพัฒนาขึ้นมาด้วยโครงสร้างการเขียนโปรแกรมที่ถูกต้องหรือไม่ การออกแบบ Test Case ด้วยวิธีนี้จะสามารถทำได้ก็ต่อเมื่อ Software Unit ที่จะทดสอบนั้นได้ถูกสร้างขึ้นมาเป็น Source Code แล้วเท่านั้น

**2. Integration Testing**

การทำ Integration Testing เป็นการทดสอบ Product-Component Builds ที่ประกอบกันขึ้นจาก Product-Components ที่ผ่านการทดสอบ Unit Testing มาแล้ว ซึ่งในกรณีของ Object-Oriented Testing จะเป็นการทดสอบกลุ่มของ Class ที่ทำงานร่วมกัน

ก่อนที่จะ Design Test Case และเริ่มการทดสอบสำหรับ Integration Test นั้น Integration Test Strategy จะต้องถูกกำหนดขึ้นมาก่อนสำหรับเป็นแนวทางในการ Integrate Software Unit ทั้งหมดเข้าด้วยกันเพื่อ Test โดย Integration Test Strategy ที่เลือกใช้ควรจะแสดงให้เห็นกลุ่มของ Software Units ที่ถูกนำมา Integrate กันขึ้นเป็น Integration Test Item เพื่อทำIntegration Test รวมถึงลำดับในการเรียกใช้Software Units ภายใน Integration Test Item ในระหว่างการทำ Integration Test ด้วย โดยการทดสอบโปรแกรมในระดับ Integration นี้มีหลักการว่าแต่ละ Software Unit จะต้องถูกเรียกใช้งานอย่างน้อย 1 ครั้งในระหว่างการ Execute Software เพื่อทำIntegration Testing

Integration Test Strategy ที่เลือกใช้อาจเป็นแบบ Big Bang Integration, Top-Down Incremental Integration แบบ Depth-first integration หรือ Breadth-first integration, Bottom-Up Incremental, Sandwich Incremental Integration, Critical Module-Based Integration ผู้ทำหน้าที่ในการกำหนดควรเลือก Strategy ที่ทำให้Software Unit ที่มีความสำคัญหรือมีความพร้อมที่จะได้รับการทดสอบ ได้รับการทดสอบเป็นลำดับแรกๆ ของการ Integration Test ทั้งนี้เพื่อเพิ่มโอกาสที่ Software Unit ดังกล่าวจะได้รับการทดสอบมากกว่า Software Unit อื่นๆการเลือก Integration Test Strategies รวมถึงลำดับก่อนหลังในการทดสอบ SoftwareUnit ต่างๆเพื่อทำการทดสอบในระดับ Integration Test สามารถพิจารณาได้จากปัจจัยหลายประการ เช่น

1. ความเร่งด่วนของการส่งมอบระบบงาน เช่น ถ้าเป็นงานที่เร่งด่วน จะเลือกใช้ Strategy แบบ Big Bang Integration เพื่อประหยัดเวลา หรือในกรณีที่เป็นระบบงานที่มีเวลาในการทำ Incremental Integration Testing ก็จะเลือกใช้ Strategy แบบ Incremental Integration เป็นต้น
2. ความสำคัญของ Software Unit กล่าวคือ Software Unit ที่มีการกำหนดระดับความสำคัญ (Priority) ไว้สูงมากๆ จะถูกนำมา Integrate เข้าใน Integration Test Item ก่อน
3. ลำดับการ Develop Software Unit ต่างๆ กล่าวคือ Software Unit ที่จะถูก Code ขึ้นมาก่อน จะถูกนำมา Integrate เข้าใน Integration Test Item ก่อน Software Unit ที่จะถูก Code ขึ้นมาทีหลัง
4. ลำดับการ Deliver Software Unit ต่างๆ กล่าวคือ Software Unit ที่จะถูก Deliver ก่อน จะถูกนำมา Integrate เข้าใน Integration Test Item ก่อน Software Unit ที่จะถูก Deliver ทีหลัง

Integration Test จะนิยมใช้Black-Box Technique ในการสร้าง Test Case เพื่อทดสอบ Interface ระหว่าง Software Unit ที่ประกอบกันขึ้นเป็น Integration Test Item และ Function การ

ทำงานโดยรวมของ Integration Test Item ด้วย โดย Method ที่ใช้ในการสร้าง Test Case นี้จะใช้ Equivalence Partitioning, Boundary Value Analysis และ Usage Scenario Simulation เป็นหลัก โดยอาจเลือกใช้เพียงบางวิธีหรือทุกวิธีก็ได้และในระหว่างการทำ Integration Test ของแต่ละ Integration Test Item นั้น Test Stubs หรือ Test Drivers ที่ถูกสร้างขึ้นระหว่างการทำUnit Test สามารถถูกนำมา Reuse อีกครั้งได้ตามความเหมาะสม

**3. System Testing**

การทำSystem Testing เป็นการทดสอบ Product ที่ประกอบกันขึ้นเป็นระบบทั้งระบบจาก Product-Components ที่ผ่านการทดสอบ Unit Testing และ Integration Testing มาแล้ว

System Test จะนิยมใช้ Black-Box Technique ในการสร้าง Test Case เพื่อทดสอบ Function การทำงานโดยรวมของระบบทั้งหมดโดยมีหลักในการทดสอบว่าแต่ละ Functional จะต้องได้รับการทดสอบอย่างน้อย 1 ครั้งในระหว่างการ Execute Software เพื่อทำ System Testing โดย Method ที่ใช้ในการสร้าง Test Case นี้จะใช้Equivalence Partitioning, Boundary Value Analysis และ Usage Scenario Simulation เป็นหลัก โดยอาจเลือกใช้เพียงบางวิธีหรือทุกวิธีก็ได้ ส่วน Method อื่นก็อาจสามารถนำมาใช้เพิ่มเติมได้ตามความเหมาะสม

**4. User Acceptance Testing**

การทำ User Acceptance Testing เป็นการทดสอบ Product ที่ประกอบกันขึ้นเป็นระบบทั้งระบบจาก Product-Components ที่ผ่านการทดสอบ Unit Testing, Integration Testing และ System Testing มาแล้ว

การทดสอบโปรแกรมในระดับ User Acceptance Test นี้มีหลักการคล้ายกับหลักการของ System Testing กล่าวคือใช้Black-Box Technique ในการสร้าง Test Case เพื่อทดสอบFunction การทำงานโดยรวมของระบบทั้งหมด ส่วนเรื่องของ Test Data นั้นจะใช้Operational Test Data ซึ่งเป็น Data จากการใช้งานจริงของผู้ใช้แทน Test Data ที่ Simulate ขึ้นมาเองในการทดสอบด้วย

Method ที่ใช้ในการสร้าง Test Case นี้จะใช้Usage Scenario Simulation กับ Usage Profile เป็นหลัก โดยอาจเลือกใช้เพียงบางวิธีหรือทุกวิธีก็ได้กล่าวคือสร้าง Test Case ที่มี Step การทำงานเลียนแบบการทำงานตามปกติของผู้ใช้โดยอาจใช้ความถี่หรือโอกาสของการถูกเรียกใช้งานของ Software Unit ต่างๆในระบบมาช่วยในการกำหนดจำนวน Test Case ที่จะสร้างขึ้นสำหรับแต่ละ Software Unit หรือสามารถใช้Test Case ชุดเดียวกับที่ใช้ตอนทำ System Testing เลยก็ได้โดยอาจจะเลือกมาเพียงบางส่วนที่ User ให้ความสำคัญ เช่น เลือกใช้เฉพาะ Test Case สำหรับ Valid

Data เพื่อเน้นการทดสอบ Function การทำงานของระบบโดยรวม โดยอาจจะใช้Test Case สำหรับ Invalid Data บ้างเพื่อทดสอบการแสดง Error Message เท่านั้น ส่วน Method อื่นก็อาจสามารถนำมาใช้เพิ่มเติมได้ตามความเหมาะสม

**5. Operation Readiness Testing**

การทำOperation Readiness Testing เป็นการทดสอบ Product ที่ประกอบกันขึ้นเป็นระบบทั้งระบบจาก Product-Components ที่ผ่านการทดสอบ Unit Testing, Integration Testing, System และ User Acceptance Testing มาแล้ว

การทดสอบโปรแกรมในระดับ Operation Readiness Test นี้มีหลักการคล้ายกับหลักการของ User Acceptance Testing กล่าวคือใช้Black-Box Technique ในการสร้าง Test Case เพื่อทดสอบ Function การทำงานโดยรวมของระบบทั้งหมด ส่วนเรื่องของ Test Data นั้นจะใช้ Operational Test Data ซึ่งเป็น Data จากการใช้งานจริงของผู้ใช้แทน Test Data ที่ Simulate ขึ้นมาเองในการทดสอบด้วย

Method ที่ใช้ในการสร้าง Test Case นี้จะใช้Usage Scenario Simulation กับ Usage Profile เป็นหลัก โดยอาจเลือกใช้เพียงบางวิธีหรือทุกวิธีก็ได้หรือสามารถใช้Test Case ชุดเดียวกับที่ใช้ตอนทำ User Acceptance Testing ก็ได้ส่วน Method อื่นก็อาจสามารถนำมาใช้เพิ่มเติมได้ตามความเหมาะสม

**นิยามศัพท์ของการทดสอบซอฟท์แวร์ (Software Testing)**

**1. White-Box Testing Technique** เรียกอีกอย่างหนึ่งว่า Glass-Box Testing เป็นการออกแบบวิธีการทดสอบโดยมีการใช้โครงสร้างควบคุมของโปรแกรมมาช่วยในการออกแบบการทดสอบ ดังนั้นเมื่อจะใช้วิธีการทดสอบแบบ White-Box จะต้องรู้การทำงานโครงสร้างของโปรแกรมที่จะทดสอบด้วย

**2. Black-Box Testing Technique** เรียกอีกอย่างหนึ่งว่า Behavioral Testing โดยมีเป้าหมายของการทดสอบที่ความต้องการของโปรแกรมนั้นๆ ซึ่งจะไม่พิจารณาถึงโครงสร้างของโปรแกรม

**3. Big Bang Integration** เป็นการบูรณาการ Module ต่างๆ ที่สร้างขึ้นมาทั้งหมดพร้อมๆ กันในทีเดียวเพื่อให้ได้ระบบที่สมบูรณ์และพร้อมที่จะนำไปทำการทดสอบหรือนำไปใช้งาน

**4. Top-Down Incremental Integration** เป็นการบูรณาการโมดูลต่างๆ จากโมดูลใหญ่มาสู่โมดูลย่อยๆ ตามเส้นทางการควบคุม (Control Hierarchy) โดยเริ่มการบูรณาการจาก Main Control Module แล้วเลือกเส้นทางเพื่อลงไปยัง Module ย่อยๆ ที่เกี่ยวข้อง

**5. Bottom-Up Incremental Integration** เป็นการบูรณาการโมดูลต่างๆ เริ่มจากโมดูลย่อยๆ ซึ่งอาจจะต้องมีการสร้าง Test Driver เพื่อเรียกใช้งานโมดูลย่อยๆ แต่ละกลุ่ม และกลุ่มที่ถูกทดสอบเรียบร้อยแล้วจะถูกส่งไปรวมกับโมดูลที่อยู่สูงขึ้นไป แล้วทำการทดสอบไล่ขึ้นไปจนถึง Main Control Module

**6. Sandwich Incremental Integration** เป็นการทดสอบโดยการนำการทดสอบทั้งแบบ Top-Down Integration และ Bottom-Up Integration มาใช้ร่วมกัน โดยจะมีการกำหนดระดับ (level) ของ Module ที่อ้างอิงแล้วจากระดับอ้างอิงให้มีการแบ่งการทำงาน ดังนี้

- Module ที่อยู่เหนือระดับอ้างอิง (upper level) ของโครงสร้างโปรแกรม ให้ใช้การทดสอบแบบ Top-Down Integration

- Module ที่อยู่ใต้ระดับอ้างอิง (subordinate Level) ของโครงสร้างโปรแกรม ให้ใช้การทดสอบแบบ Bottom-Up Integration

**7. Boundary Value Analysis** เป็นเทคนิคการออกแบบกรณีทดสอบโดยเลือกใช้ค่าขอบของความเป็นไปได้ในการทดสอบ

**8. Usage Scenario Simulation** เป็นเทคนิคการออกแบบกรณีทดสอบโดยการสุ่มข้อมูลโดยในการสุ่มข้อมูลเพื่อใช้ในการทดสอบระบบจะต้องมีทั้ง Valid Data และ Invalid Data

**9. Bug/ Issue/ Defect** ความผิดพลาด/ ข้อบกพร่องของการทำงานของซอฟท์แวร์หรือระบบที่ทำการทดสอบ อาจรวมถึงคำแนะนำจากผู้ทำการทดสอบระบบด้วย

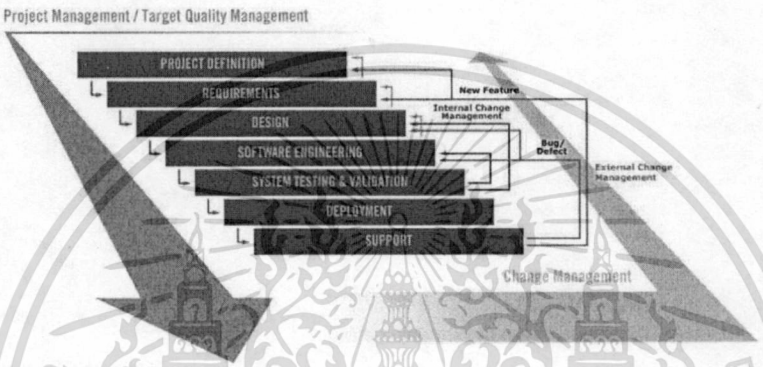
**10. Test Case** เป็นการกำหนดกรณีในการทดสอบ โดยขั้นตอนในการทดสอบแต่ละกรณีจะถูกเรียกว่า Test Script

**11. Test Data** เป็นข้อมูลที่ใช้สำหรับทำการทดสอบตามกรณีทดสอบ (Test Case) ที่ได้มีการออกแบบได้โดยข้อมูลที่ใช้ในการทดสอบนี้จะถูกแบ่งเป็นสองประเภทคือ ข้อมูลที่ถูกต้องสำหรับใช้ในการทดสอบ (Valid Data) และข้อมูลที่ไม่ถูกต้อง (Invalid Data) เมื่อนำมาใช้ในการทดสอบโปรแกรมจะต้องไม่สามารถทำงานได้

## 2.2 ทฤษฎีวงจรการพัฒนาระบบ

นอกจากการสร้างระบบสารสนเทศขึ้นมาใหม่แล้วการวิเคราะห์และออกแบบระบบจะช่วยให้เกิดขั้นตอนในการพัฒนาระบบที่ดีขึ้น โดยแบ่งการท างานเป็นระยะต่างๆ โดยการพัฒนาระบบตามปกติแล้วจะประกอบไปด้วยกลุ่มกิจกรรม 3 ส่วนหลักๆ คือ การวิเคราะห์ การออกแบบ และการนำไปใช้ ซึ่ง 3 กิจกรรมนี้จะสามารถใช้ได้กับระบบขนาดเล็ก หากเป็นระบบที่มีความซับซ้อนจะใช้การพัฒนาโปรแกรมตามหลักของ SDLC (System Development Life Cycle)

ขั้นตอนในวงจรพัฒนาระบบ ช่วยให้นักวิเคราะห์ระบบสามารถดำเนินการได้อย่างมีแนวทางและเป็นขั้นตอน ทำให้สามารถควบคุมระยะเวลาและงบประมาณในการปฏิบัติงานของโครงการพัฒนาระบบได้ ขั้นตอนต่างๆ นั้นมีลักษณะคล้ายกับการตัดสินใจแก้ปัญหาตามแนวทางวิทยาศาสตร์ (Scientific Management) ได้แก่ การค้นหาปัญหา การค้นหาแนวทางแก้ไขปัญหา การประเมินผลแนวทางแก้ไขปัญหาที่ค้นพบเลือกแนวทางที่ดีที่สุด และพัฒนาทางเลือกนั้นให้ใช้งานได้ สำหรับวงจรการพัฒนาระบบ (System Development Life Cycle: SDLC) ประกอบด้วย 7 ขั้นตอนดังนี้




**รูปที่ 2.1** วงจรการพัฒนาระบบ(System Development Life Cycle: SDLC)

**ขั้นตอนที่ 1 การกำหนดปัญหา จุดมุ่งหมายและเป้าหมาย (Identifying Problems Opportunities and Objective)**

ระบบสารสนเทศจะเกิดขึ้นได้ก็ต่อเมื่อผู้บริหารหรือผู้ใช้ตระหนักถึงความต้องการใช้ระบบสารสนเทศหรือระบบจัดการเดิม ได้แก่ ระบบจัดเก็บเอกสารในตู้เอกสารไม่มีประสิทธิภาพเพียงพอที่จะตอบสนองความต้องการในปัจจุบัน ดังนั้นควรต้องมีการปรับปรุงแก้ไข ซึ่งเป็นหน้าที่ของนักวิเคราะห์ระบบที่ทำการแก้ไขปรับปรุง การแก้ไขระบบเดิมที่มีอยู่แล้วหรือการสร้างระบบใหม่นั้นเป็นเรื่องยาก ดังนั้น นักวิเคราะห์ระบบ ควรกำหนดจุดประสงค์ในการนำระบบคอมพิวเตอร์ไปใช้งานในด้านต่างๆ ซึ่งจะต้องมองปัญหาให้ถูกต้องและมีเป้าหมายที่ชัดเจนจะได้รู้ทิศทางในการพัฒนาระบบเพื่อให้เป็นไปตามเป้าหมายที่วางไว้

**ขั้นตอนที่ 2 การกำหนดความต้องการสารสนเทศของผู้ใช้ (Determining Information Requirement)**

เป็นการเก็บรวบรวมข้อมูลที่เป็นความต้องการสารสนเทศของผู้ใช้ระบบ โดยนักวิเคราะห์ระบบจะต้องใช้เทคนิคในการเก็บข้อมูล (Fact Gathering Techniques) ได้แก่ การสุ่มตัวอย่าง ศึกษาเอกสารที่มีอยู่ตรวจสอบวิธีการทำงานในปัจจุบัน สัมภาษณ์ผู้ใช้และผู้ที่มีส่วนเกี่ยวข้องกับระบบ

การศึกษาเอกสารที่มีอยู่ ได้แก่ คู่มือการทำงานแผนผังสายงานขององค์กรรายงานต่างๆ ที่หมุนเวียนอยู่ในระบบการศึกษา วิธีการทำงานในปัจจุบันจะทำให้นักวิเคราะห์ระบบ ทราบว่าระบบงานจริงๆ ทำงานอย่างไร ซึ่งบางครั้งอาจค้นพบข้อมูลผิดพลาด และจุดที่สำคัญของระบบได้สัมภาษณ์ผู้ใช้ระบบและผู้บริหาร ทำให้นักวิเคราะห์ระบบทราบว่าระบบทำงานอย่างไร เนื่องจากผู้ใช้ระบบหรือผู้บริหารเป็นบุคคลที่เชี่ยวชาญในหน้าที่ที่ทำอยู่ ทำให้สามารถบอกได้ว่า สิ่งที่ขาดหายไปในระบบคืออะไรบ้าง เพื่อจะได้กำหนดความต้องการสารสนเทศของผู้ใช้ได้อย่างครบถ้วนสมบูรณ์

**ขั้นตอนที่ 3 การวิเคราะห์ระบบ (Analyzing System Needs)**

เป็นการนำข้อมูลที่รวบรวมได้จากการกำหนดความต้องการสารสนเทศของผู้ใช้มาเขียนเป็นไดอะแกรมการไหลของข้อมูล (Data Flow Diagram) พจนานุกรมข้อมูล (Data Dictionary) และโครงสร้างการตัดสินใจ (Structure Decision) มาช่วยในการวิเคราะห์ระบบ

**ขั้นตอนที่ 4 การออกแบบระบบ (Designing the Recommended System)**

นักวิเคราะห์ระบบ จะนำแผนภาพที่เขียนขึ้นในขั้นตอนการวิเคราะห์มาแปลงเป็นแผนภาพลำดับชั้น (แบบต้นไม้) เพื่อให้เห็นภาพลักษณ์ที่แน่นอนของโปรแกรมว่ามีความสัมพันธ์กันอย่างไร และโปรแกรมอะไรบ้างที่จะต้องเขียนในระบบ หลังจากนั้นทำการตัดสินใจว่าควรจัดโครงสร้างของโปรแกรมอย่างไร การเชื่อมโยงระหว่างโปรแกรมจะต้องทำอย่างไร ในขั้นตอนการวิเคราะห์ระบบ นักวิเคราะห์ระบบจะต้องหาว่า "จะต้องทำอะไร (What)" แต่ในขั้นตอนการออกแบบจะต้องรู้ว่า "จะต้องทำอย่างไร (How)"

**ขั้นตอนที่ 5 การพัฒนาซอฟต์แวร์และการจัดทำเอกสาร (Developing Documenting Software)**

เป็นขั้นตอนการท างานร่วมกันระหว่างโปรแกรมเมอร์และนักวิเคราะห์ระบบเพื่อพัฒนาซอฟต์แวร์และต้องเตรียมคู่มือการใช้งานควบคู่ไปด้วย โดยโปรแกรมเมอร์จะเขียนโปรแกรมตามข้อมูลที่ได้จากเอกสารข้อมูลเฉพาะที่ได้จากการออกแบบระบบ หากมีการแก้ไขเปลี่ยนแปลงในระหว่างการเขียนโปรแกรมจะต้องปรึกษากับนักวิเคราะห์ระบบด้วย

**ขั้นตอนที่ 6 การทดสอบและการบำรุงรักษาระบบ (Testing and Maintaining the System)**

ก่อนที่จะนำระบบที่สร้างขึ้นไปใช้จะต้องมีการทดสอบระบบ ทั้งนี้ผู้ทดสอบอาจเป็นโปรแกรมเมอร์เองหรืออาจให้ผู้ใช้ระบบและนักวิเคราะห์ระบบเป็นผู้ทดสอบ การทดสอบระบบควรใช้ข้อมูลจริงมาใช้ในการทดสอบ หากเมื่อมีข้อผิดพลาดจะได้ทำการแก้ไขปรับปรุง ซึ่งก็คือการบำรุงรักษาระบบ

**ขั้นตอนที่ 7 การดำเนินงานและประมวลผล (Implementing and Evaluating the System)**

เป็นขั้นตอนสุดท้ายในการดำเนินงานของระบบจะต้องมีการจัดอบรมผู้ใช้ระบบก่อน นำไปใช้งานจริง และมีการประเมินผล เพื่อให้ทราบถึงความพอใจของผู้ใช้ระบบหรือสิ่งที่ต้องแก้ไข เมื่อมีการพัฒนาระบบต่อไป

## 2.4 การวิเคราะห์และออกแบบระบบเชิงวัตถุ

### 2.4.1 แนวทางความคิดพื้นฐานเชิงวัตถุ

หลักแนวความคิดเชิงวัตถุ (Object Oriented: OO) เป็นเทคนิคอย่างหนึ่งที่นำมาใช้ในการอธิบายระบบ โดยจะมองสิ่งต่างๆ ในระบบเป็นวัตถุหรืออ็อบเจกต์ ซึ่งอ็อบเจกต์หมายถึงสิ่งที่เราสนใจ อาจจะใช้แทนคน สถานที่ เหตุการณ์ หรือรายการที่เกิดขึ้นก็ได้ ซึ่งแต่ละอ็อบเจกต์จะมีคุณสมบัติและการทำงานเฉพาะตัวแตกต่างกันออกไป บางอ็อบเจกต์ก็มีจะมีความสัมพันธ์กับอ็อบเจกต์อื่นๆ ในระบบได้ และถ้าอ็อบเจกต์ใดมีคุณลักษณะที่คล้ายๆกัน เราก็จะจะจัดกลุ่มของอ็อบเจกต์เหล่านั้นให้อยู่ด้วยกัน แต่ละอ็อบเจกต์จะประกอบไปด้วยแอตทริบิวต์ คือคุณลักษณะหรือคุณสมบัติของอ็อบเจกต์หนึ่งๆ และเมธอด คือฟังก์ชั่นของพฤติกรรม หรือบริการที่อ็อบเจกต์นั้นสามารถกระทำได้

### 2.4.2 ยูเอ็มแอล

ยูเอ็มแอล (Unified Modeling Language: UML) เป็นภาษาเพื่อใช้อธิบายตัวแบบ หรือ โมเดล (Model) ต่างๆ โดยที่มีการใช้ภาษาหรือตัวแทนที่เป็นภาพสัญลักษณ์ หรือ กราฟิก (Graphics) และเป็นภาษามาตรฐานที่นิยมใช้ในการพัฒนาระบบ เปรียบเสมือนแบบพิมพ์เขียว (Blueprint) ให้กับระบบงาน ในการสร้างมุมมอง การกำหนดรายละเอียด สร้างระบบงานและจัดทำเอกสารอ้างอิงให้กับงานที่ต้องการจะพัฒนา การนำมาภาษาที่เป็นการสร้างโมเดล (Modeling language) มาใช้ในการพัฒนานั้นทำให้สามารถสร้างโปรแกรมอย่างเป็นระบบ เนื่องจากมีการกำหนดมาตรฐานและขั้นตอนอย่างชัดเจน เหมือนแบบแปลนบ้านที่ให้ผู้รับเหมารายใดมาอ่านก็จะทำได้เหมือนๆกัน ดังนั้นผู้พัฒนาโปแกรมจึงสามารถที่จะเขียนโปรแกรมตามแบบพิมพ์เขียวของระบบได้เช่นกัน ทำให้เกิดการทำงานที่เป็นระบบ สามารถแบ่งงานและความรับผิดชอบไปให้กับโปแกรมเมอร์แต่ละคนได้ และเมื่อนำงานมารวมกันก็สามารถที่จะทำงานร่วมกันได้อย่างมีประสิทธิภาพ ซึ่งภาษา UML นั้นจะครอบคลุมทุกขั้นตอนของการพัฒนาตั้งแต่ต้นจนจบ และสำหรับการพัฒนาระบบงานครั้งนี้จะใช้ไดอะแกรมต่างๆ ซึ่งประกอบด้วย

**1. ยูสเคสไดอะแกรม** (Use Case Diagram)

ยูสเคสไดอะแกรม เป็นแผนภาพที่แสดงการทำงานของผู้ใช้ระบบ (User) และความสัมพันธ์กับระบบย่อย (Sub systems) ภายในระบบใหญ่ ในการเขียน Use Case Diagram ผู้ใช้ระบบ (User) จะถูกกำหนดว่าให้เป็น Actor และ ระบบย่อย (Sub systems) คือ Use Case จุดประสงค์หลักของการเขียน Use Case Diagram ก็เพื่อเล่าเรื่องราวทั้งหมดของระบบว่ามีการทำงานอะไรบ้าง เป็นการดึง Requirement หรือเรื่องราวต่าง ๆ ของระบบจากผู้ใช้งาน ซึ่งถือว่าเป็นจุดเริ่มต้นในการวิเคราะห์และออกแบบระบบ สัญลักษณ์ที่ใช้ใน Use Case Diagram จะประกอบด้วย

1.1 แอกเตอร์ จะใช้สัญลักษณ์เป็นรูปคน โดยแอกเตอร์นั้นจะหมายถึงคนหรือระบบก็ได้ที่ใช้งานยูสเคส

1.2 ยูสเคส จะใช้สัญลักษณ์เป็นรูปวงรี โดยยูสเคสนั้นจะหมายถึงกิจกรรมหลักๆ ที่เกิดขึ้นในระบบนั้นๆ

1.3 ความสัมพันธ์ จะใช้สัญลักษณ์เส้นตรง เป็นความเกี่ยวข้องหรือความสัพันธ์เชื่อมโยงกันระหว่างเอกเตอร์กับยูสเคส หรือระหว่างยูสเคสกับยูสเคสด้วยกัน โดยความสัมพันธ์ของ Use Case นั้น สามารถแบ่งออกได้ 2 แบบ คือ

1.3.1 ความสัมพันธ์แบบ Include หมายถึง การที่ Use Case หนึ่ง เรียกใช้งาน Use Case อีกอันหนึ่ง คล้าย ๆ กับการเรียกใช้งาน Program ย่อยโดย Program หลัก การเขียนสัญลักษณ์แทนการ Include ของ Use Case นั้น ใช้สัญลักษณ์เส้นประพร้อมหัวลูกศรชี้ไปยัง Use Case ที่ถูกเรียกใช้งาน และมีคำว่า <<include>> กำกับอยู่บนเส้นลูกศร

1.3.2 ความสัมพันธ์แบบ Extend หมายถึง การที่ Use Case หนึ่งไปมีผลต่อการทำงานตามปกติของอีก Use Case หนึ่ง นั่นหมายถึงว่า Use Case ที่มา Extend นั้นจะมีผลทำให้การทำงานของUse Case ที่ถูก Extend ถูกรบกวนหรือมีการสะดุด หรือมีกิจกรรมที่เปลี่ยนแปลงไป สัญลักษณ์ที่ใช้แทน Extend ใน Use Case Diagram ก็คือ ใช้สัญลักษณ์ลูกศร โดยเริ่มจาก Use Case ที่ Extend ไปยัง Use Case ที่ถูก Extend และมีคำว่า << extend >>กำกับ

**2. แอกทิวิตีไดอะแกรม (Activity Diagram)**

**แอกทิวิตีไดอะแกรม** แสดงลำดับ กิจกรรมของการทำงาน (Work Flow) สามารถแสดงทางเลือกที่เกิดขึ้นได้

Activity Diagram จะแสดงขั้นตอนการทำงานในการปฏิบัติการ โดยประกอบไปด้วยสถานะต่างๆ ที่เกิดขึ้นระหว่างการทำงาน และผลจากการทำงานในขั้นตอนต่าง ๆ

- วงกลมสีดำ คือ จุดเริ่มต้น
- วงกลมสีดำ มีวงล้อมอีกชั้น คือ จุดสิ้นสุด
- สี่เหลี่ยมคล้ายแคปซูล คือ กิจกรรม
- Swim lanes เป็นการแบ่งกลุ่ม activity เป็นช่องในแนวดิ่ง และกำหนดแต่ละช่องด้วยชื่อ object ไว้ด้านบน การแบ่งเป็น Swim lanes ช่วยให้แยกแยะผู้รับผิดชอบ แต่ละงานได้ว่าใครควรจะเป็นคนทำงานในหมวดหมู่ใด

**3. คลาสไดอะแกรม (Class Diagram)**

Class Diagram คือ แผนภาพที่ใช้แสดง Class และความสัมพันธ์ในแง่ต่างๆ (Relation) ระหว่าง Class เหล่านั้น ซึ่งความสัมพันธ์ที่กล่าวถึงใน Class Diagram นี่ถือเป็นความสัมพันธ์เชิงสถิตย์ (Static Relationship) หมายถึง ความสัมพันธ์ที่มีอยู่แล้วเป็นปกติในระหว่าง Class ต่างๆ ไม่ใช่ความสัมพันธ์ที่เกิดขึ้นเนื่องจากกิจกรรมต่างๆ ซึ่งเรียกว่า ความสัมพันธ์เชิงกิจกรรม (Dynamic Relationship) สิ่งที่ปรากฏใน Class Diagram นั้นประกอบด้วยกลุ่มของ Class และกลุ่มของ Relationship โดยสัญลักษณ์ที่ใช้ในการแสดง Class นั้นจะแทนด้วยสี่เหลี่ยมที่แบ่งออกเป็น 3 ส่วน โดยแต่ละส่วนนั้น (จากบนลงล่าง) จะใช้ในการแสดง ชื่อของ Class, Attribute, และฟังก์ชันต่างๆ ตามลำดับ

ในการเขียนสัญลักษณ์แทน Class สิ่งที่ต้องคำนึงถึงอีกสิ่งหนึ่งคือ ระดับการเข้าถึงเรียกสัญลักษณ์ที่ใช้แทนการเข้าถึงนี้ว่า Visibility แบ่งออกได้เป็น 3 ประเภท ได้แก่

1. Private เขียนแทนด้วยสัญลักษณ์ - หมายถึง Attribute หรือ ฟังก์ชัน ที่ไม่สามารถมองเห็นได้จากภายนอก แต่สามารถมองเห็นได้จากภายในตัวของ Class เองเท่านั้น

2. Protect เขียนแทนด้วยสัญลักษณ์ # หมายถึง Attribute หรือ ฟังก์ชัน ที่สงวนไว้สำหรับการทำ Inheritance โดยเฉพาะ Attribute หรือ ฟังก์ชันเหล่านี้ จะเป็นของ Super class เมื่อทำการ Inheritance แล้ว Attribute หรือ ฟังก์ชัน ที่มี Visibility แบบ Protect จะกลายไปเป็น Private Attribute/ฟังก์ชัน หรือ Protected ขึ้นอยู่กับภาษา Programming ที่นำไปใช้

3. Public เขียนแทนด้วยสัญลักษณ์ + หมายถึง Attribute หรือ ฟังก์ชัน ที่สามารถมองเห็นได้จากภายนอก และสามารถเข้าไปเปลี่ยนค่า อ่านค่าหรือเรียกใช้งาน Attribute หรือ ฟังก์ชัน นั้นได้ทันทีโดยอิสระจากภายนอก (โดยทั่วไปแล้ว Visibility แบบ Public มักจะใช้กับฟังก์ชันมากกว่า Attribute)

## 2.5 เทคโนโลยีที่ใช้ในการพัฒนาระบบ

### 2.5.1 HTML

HTML ย่อมาจาก Hyper Text Markup Language ซึ่งใช้แสดงผลของข้อมูลต่างๆ ไม่ว่าจะเป็นข้อความ, รูปภาพ, เสียง, หรือแม้กระทั่งภาพเคลื่อนไหว ในรูปแบบของเวบเพจ ผ่านการติดต่อกัน

ในโปรโตคอล HTTP ซึ่งมีรูปแบบของโครงสร้างภาษาที่เข้าใจง่าย และทำงานได้รวดเร็ว อีกทั้งยังจะมีเครื่องสำหรับใช้ในการพัฒนาร่วมกับ HTML อยู่อีกมากมายในปัจจุบัน

### 2.5.2 Java 2 Platform, Enterprise Edition (J2EE™)

Java 2 Platform, Enterprise Edition (J2EE™) คือ กลุ่มของเทคโนโลยีภาษาจาวาที่จำเป็นสำหรับการพัฒนาโปรแกรมประยุกต์ด้วยภาษาจาวาให้ใช้งานได้ในระดับองค์กร

แต่เดิมนั้นภาษาจาวาถูกใช้ในการสร้างโปรแกรมประยุกต์ส่วนบุคคลเป็นหลัก ต่อมาเมื่อจาวาถูกนำไปใช้งานภายในองค์กรมากขึ้น ผู้พัฒนาจาวาจำเป็นต้องเพิ่มชุดคำสั่งใหม่ๆ เข้าไปเพื่อตอบสนองความต้องการใช้งานระดับองค์กร เมื่อชุดคำสั่งที่เพิ่มเติมเข้าไปมีจำนวนมากขึ้นเรื่อยๆ ผู้พัฒนาจาวาจึงได้ทำจัดระเบียบชุดคำสั่งในภาษาจาวาเสียใหม่โดยแบ่งเทคโนโลยีจาวาออกเป็น 3 ระดับ ดังต่อไปนี้

1. J2SE (Java 2 Standard Edition) หรือ จาวาระดับมาตรฐาน คือ จาวาดั้งเดิมที่มีแต่ชุดคำสั่งปกติสำหรับการสร้างโปรแกรมประยุกต์ส่วนบุคคล

2. J2EE (Java 2 Enterprise Edition) หรือ จาวาระดับองค์กร คือ จาวาที่มีชุดคำสั่งเพื่อการสร้างโปรแกรมประยุกต์ที่ใช้งานระดับองค์กร

3. J2ME (Java 2 Micro Edition) หรือ จาวาระดับจิ๋ว คือ จาวาที่มีชุดคำสั่งเพื่อการสร้างโปรแกรมประยุกต์สำหรับอุปกรณ์อิเล็กทรอนิกส์ขนาดเล็ก

J2EE ถือได้ว่าเป็นส่วนขยายของ J2SE ด้วย กล่าวคือ J2EE รวมเอาทั้งหมดของ J2SE เข้าไปอยู่ในตัวมันแล้วเพิ่มชุดคำสั่งที่จำเป็นสำหรับการพัฒนาโปรแกรมระดับองค์กรเข้าไป

**ตารางที่ 2.1** ชุดคำสั่งในชุดของ J2EE

| ชื่อชุดคำสั่ง | หน้าที่และประโยชน์ใช้สอย |
|---|---|
| RMI-IIOP | ใช้สำหรับการสื่อสารหรือเรียกใช้เมธธอสระยะไกลผ่านระบบเครือข่าย |
| Java IDL | ภาษากลางสำหรับสื่อสารกับโปรแกรมประยุกต์ระดับองค์กรที่พัฒนาด้วยเทคโนโลยีอื่น |
| JNDI | ระบบการเรียกชื่อของบริการหรือวัตถุที่อยู่บนระบบเครือข่าย |
| JDBC | ใช้เชื่อมติดต่อกับฐานข้อมูล |
| JMS | ใช้ประสานงานหรือสื่อสารระหว่างโปรแกรมประยุกต์ด้วยกัน |
| JavaMail | ใช้ติดต่อกับเมล์เซิร์ฟเวอร์ในองค์กร |
| Java Servlets | บริการเวบบนฝั่งเซิร์ฟเวอร์ |

**ตารางที่ 2.1** ชุดคำสั่งในชุดของ J2EE(ต่อ)

| ชื่อชุดคำสั่ง | หน้าที่และประโยชน์ใช้สอย |
|---|---|
| JSP | บริการเวบบนฝั่งเซิร์ฟเวอร์ที่ดัดแปลงมาจาก Java Servlets อีกทีหนึ่งโดยมีความง่ายในการพัฒนามากกว่า |
| JTA | ชุดคำสั่งสำหรับการบริหารจัดการธุรกรรม |
| JTS | บริการธุรกรรม |
| EJB | โครงสร้างโปรแกรมประยุกต์แบบคอมโพเนนท์ |
| JAXP | ชุดคำสั่งสำหรับการจัดการไฟล์ XML |
| JAAS | ชุดคำสั่งสำหรับระบบรักษาความปลอดภัย |
| JCA | ใช้ติดต่อกับระบบอื่นๆ ที่มีอยู่เดิมในเครือข่ายขององค์กร |

### 2.5.3 Java Application Server

มีการสร้างแนวคิดเรื่อง Application Server ขึ้นมา Application Server เป็นการรวมเทคโนโลยีในชุดของ J2EE ไว้ในที่เดียวกัน เพื่อให้เกิดความง่ายในการพัฒนาโปรแกรม แนวคิดเรื่อง Application Server ทำให้การพัฒนาโปรแกรมประยุกต์สำหรับองค์กรทำได้ง่ายขึ้น เพราะ Application Server จะรับผิดชอบเกี่ยวกับการจัดการเครือข่ายทั้งหมด

**1. โครงสร้างของ Java Application Server**

Java Application Server มีลักษณะคล้ายกับ Java Virtual Machine ในจาวาระดับมาตรฐาน กล่าวคือ เป็นสิ่งที่โปรแกรมภาษาจาวาที่เราเขียนขึ้นทำงานบนตัวมันอีกที ซึ่งทำหน้าที่เป็นตัวกลางในการติดต่อสื่อสารกับภายนอกแทนโปรแกรมของเราแบบเดียวกับ Java Virtual Machine ที่คั่นกลางระหว่างโปรแกรมภาษาจาวาของเรากับระบบปฏิบัติการของเครื่อง

Java Application Server มีความแตกต่างจาก Java Virtual Machine เพราะในระดับองค์กรนอกจากโปรแกรมประยุกต์จะต้องติดต่อกับระบบปฏิบัติการแล้วยังต้องติดต่อกับสิ่งอื่นอีกมาก ซึ่งส่วนใหญ่เป็นการติดต่อผ่านทางระบบเครือข่ายขององค์กร เช่น บางครั้งโปรแกรมต้องดึงข้อมูลจากฐานข้อมูลที่อยู่บนเครื่องคอมพิวเตอร์อีกเครื่องหนึ่งที่ห่างออกไปบนเครือข่ายองค์กร เป็นต้น

**2. คุณสมบัติเด่นของ Java Application Server**

2.1 สนับสนุนมาตรฐาน COBRA (Common Object Request Broker Architecture) ซึ่งเป็นการคุยกันผ่านโปรโตคอลมาตรฐานที่เรียกว่า IIOP และใช้ภาษากลางที่เป็นมาตรฐานชื่อว่า IDL (Interface Definition Language)

2.2 สนับสนุนแนวคิดของ Web Application ใน Java Application Server จะมีคอนเทนเนอร์สำหรับ Java Servlets และ Java Server Pages (JSP) พ่วงมาด้วย เราจึงสามารถใช้เบราเซอร์เป็นตัวติดต่อกับโปรแกรมที่พัฒนาบน Java Application Server โดยใช้คอนเทนเนอร์เหล่านี้เป็นตัวกลางได้

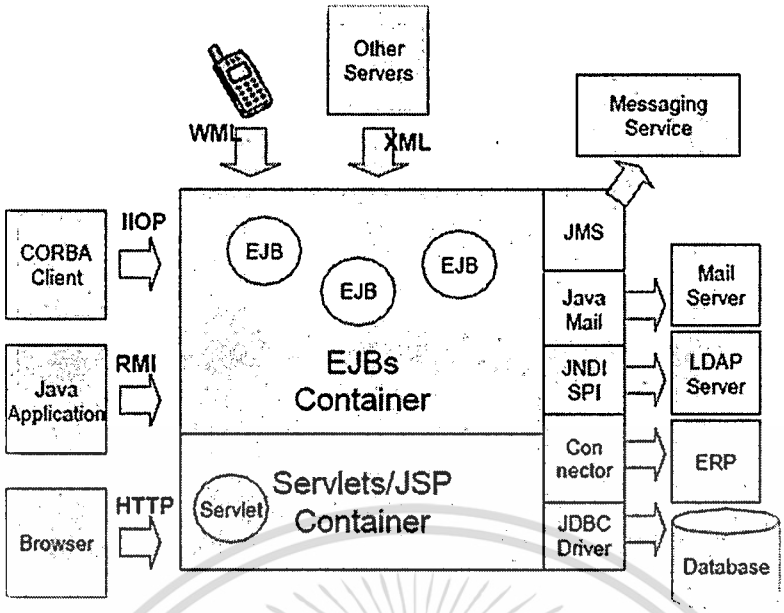
2.3 ใช้หลักการของคอมโพเนนท์ คือ หน่วยของโปรแกรมที่ทำงานเฉพาะอย่างหนึ่ง มีอิสระในตัวเอง สามารถทำงานร่วมกับคอมโพเนนท์อื่นเพื่อทำงานที่มีความซับซ้อนมากขึ้นได้ การใช้สถาปัตยกรรมแบบคอมโพเนนท์แทนการเขียนโปรแกรมขนาดใหญ่โปรแกรมเดียวเพื่อรองรับงานทั้งหมด ทำให้โปรแกรมที่พัฒนาขึ้นมีความยืดหยุ่นสูง สามารถขยายตัวเองเพื่อรองรับงานจากผู้ใช้จำนวนมากขึ้นได้ง่าย และสามารถใช้คอมพิวเตอร์หลายเครื่องช่วยกันทำงานอย่างเดียวกันให้เสร็จเร็วขึ้นด้วยการแบ่งคอมโพเนนท์ออกไปรันบนเครื่องคอมพิวเตอร์หลายเครื่องที่เชื่อมโยงกันด้วยระบบเครือข่ายได้ ทั้งหมดนี้เป็นไปได้โดยไม่ต้องมีการแก้ไขคำสั่งในโปรแกรมใหม่ คอมโพเนนท์บน Java Application Server มีชื่อเรียกว่า EJB (Enterprise Java Bean)

2.4 สนับสนุนแนวคิดของบริการผ่านเวบ บริการผ่านเวบ (Web Services) เป็นแนวคิดใหม่ที่กำลังมาแรงในแวดวงคอมพิวเตอร์ระดับองค์กร เป็นวิธีการติดต่อสื่อสารข้ามองค์กรผ่านเครือข่ายอินเตอร์เนตโดยอาศัยมาตรฐานเดียวกันได้แก่ XML (eXtensible Markup Language) โดยชุดคำสั่งภาษาจาวาที่เกี่ยวกับการจัดการ XML เช่น JAXP, JAXM, JAX-RPC ได้ถูกนำมารวมไว้ใน Java Application Server ทำให้ Java Application Server เป็นแพลตฟอร์มสำหรับการพัฒนาบริการผ่านเวบในองค์กรที่สะดวกที่สุด

2.5 มีการบริหารหน่วยความจำที่มีประสิทธิภาพ Java Application Server มีการบริหารหน่วยความจำของเครื่องให้สามารถรองรับผู้ใช้จำนวนมากในเวลาเดียวกันได้มากกว่าปกติ เพราะสามารถเพิ่มลดตัวโปรแกรมที่อยู่ในหน่วยความจำได้ตามจำนวนผู้ใช้ที่เข้ามา โดยสลับเอาโปรแกรมตัวที่ถูกใช้งานไม่บ่อยออกไปพักไว้ในหน่วยความจำสำรองเพื่อให้เกิดพื้นที่ว่างสำหรับโปรแกรมที่ทำงานหนัก โดยกลไกเหล่านี้เป็นกลไกที่เกิดขึ้นโดยอัตโนมัติโดยการจัดการของเซิร์ฟเวอร์

2.6 มีการจัดการด้านธุรกรรมข้อมูล Java Application Server มีระบบจัดการด้านธุรกรรมข้อมูลมาด้วยในตัว ทำให้การใช้งานข้อมูลสำคัญที่ต้องใช้ฐานข้อมูลเป็นไปอย่างมีเสถียรภาพ Java Application Server สนับสนุนชุดคำสั่ง JTA (Java Transaction API) และ JTS (Java Transaction Service)

2.7 มีระบบรักษาความปลอดภัยและระบบจัดการผู้ใช้ Java Application Server มีระบบรักษา ความปลอดภัยและระบบจัดการผู้ใช้ที่สนับสนุน JAAS (Java Authentication and Authorization Service) ในตัว




รูปที่ 2.2 โครงสร้างของ Java Application Server

**2.5.4 Java Servlet**

Java Servlet คือ โปรแกรมที่เขียนขึ้นมาเป็นคลาสของภาษาจาวาเพื่อทำงานทางฝั่งเซิร์ฟเวอร์ ใช้สำหรับเขียนโปรแกรมในลักษณะCGI เพื่อทำหน้าที่อ่านข้อมูลที่รับมาจากผู้ชมเวบไซต์ แล้วเอาข้อมูลนั้นมาประมวลผล จากนั้นจึงส่งผลลัพธ์กลับไปให้ผู้ชมเวบนั่นเอง

เนื่องจาก Java Servlet มีจุดเด่นที่สำคัญมาก เช่น มีประสิทธิภาพและความเร็วสูงในการทำงาน สามารถปรับปรุงแก้ไขและพัฒนาได้ง่ายเพราะใช้ภาษาจาวาซึ่งเป็นภาษาเชิงวัตถุในการพัฒนา เป็นต้น จึงเป็นอีกทางเลือกหนึ่งในการพัฒนา CGI และได้รับความนิยมนำมาพัฒนา Web Application

**2.5.5 Java Server Pages**

Java Server Pages หรือเรียกย่อๆ ว่า JSP คือ การใช้ภาษาจาวาในการสร้างเวบเพจแบบที่มีเนื้อหาไม่ตายตัว เป็นเทคโนโลยีที่ใช้สคริปต์ในการพัฒนา Web Application เพื่อทำงานทางฝั่งเซิร์ฟเวอร์ และส่งผลลัพธ์กลับมายังเวบเบราเซอร์เป็นภาษา HTML เหมือนกับเทคโนโลยีอื่น ๆ เช่น ASP , PHP เป็นต้น

การเขียนสคริปต์ JSP จะใช้ภาษาจาวาเป็นหลัก ข้อแตกต่างระหว่าง JSP และJava Servlet คือ JSP เป็นสคริปต์ ฉะนั้นเวลาพัฒนา Web Application เราสามารถเขียนแท็กคำสั่งของ JSP แทรกลงไปในบริเวณที่ต้องการ ภายในไฟล์เอกสาร HTML ได้ทันที โดยการเขียนแท็กเปิด แล้วตามด้วย Source Code JSP และปิดท้ายด้วยแท็กปิด

**ข้อดีของ JavaServer Pages**

1. ทำงานโดยไม่ยึดติดแพลตฟอร์มใด ๆ
2. ใช้งาน Java API ได้หลากหลาย
3. นำคอมโพเนนต์กลับมาใช้ได้อีก ไม่ต้องเสียเวลาสร้างใหม่
4. มีความยืดหยุ่นในการใช้งาน
5. มีความปลอดภัยสูง

### 2.5.6 JavaScript

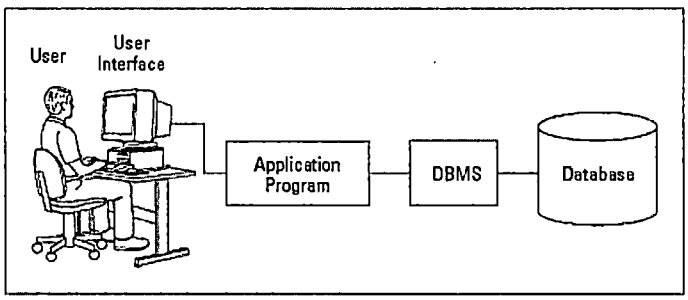
JavaScript เป็นภาษายุคใหม่สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ตที่กำลังได้รับความนิยมอย่างสูง เราสามารถเขียนโปรแกรม JavaScript เพิ่มเข้าไปในเวบเพจเพื่อใช้ประโยชน์สำหรับงานด้านต่าง ๆ ทั้งการคำนวณ การแสดงผล การรับ – ส่งข้อมูล และที่สำคัญคือสามารถโต้ตอบกับผู้ใช้ได้อย่างทันทีทันใด นอกจากนี้ยังมีความสามารถด้านอื่นๆ อีกหลายประการที่ช่วยสร้างความน่าสนใจให้กับเวบเพจของเราได้เป็นอย่างมาก

JavaScript เป็นภาษาสคริปต์เชิงวัตถุ ที่ช่วยให้เราสามารถควบคุมเวบเพจได้อย่างง่ายดาย สามารถทำงานข้ามแพลตฟอร์มได้ ทำหน้าที่เป็นตัวประสานระหว่างเว็บเพจ HTML , Java applet และเวบเบราเซอร์ ทั้งทางฝั่งไคลเอนต์ (Client) และฝั่งเซิร์ฟเวอร์ (Server)

## 2.6 ระบบฐานข้อมูล

ระบบฐานข้อมูล (Database System) หมายถึง การรวมตัวกันของฐานข้อมูลตั้งแต่ 2ฐานข้อมูลเป็นต้นไปที่มีความสัมพันธ์กัน โดยมีวัตถุประสงค์เพื่อเป็นการลดความซ้ำซ้อนของข้อมูล และทำให้การบำรุงรักษาตัวโปรแกรมง่ายมากขึ้น โดยผ่านระบบการจัดการฐานข้อมูล หรือ เรียกย่อ ๆ ว่า DBMS

ระบบการจัดการฐานข้อมูล หมายถึง โปรแกรม หรือ ซอฟต์แวร์ที่ทำหน้าที่ในการบริหารและจัดการฐานข้อมูลในการสร้าง การเรียกใช้ การปรับปรุงฐานข้อมูล เป็นเสมือนตัวกลางระหว่างผู้ใช้งานกับระบบฐานข้อมูล โปรแกรมที่ใช้ในการจัดการฐานข้อมูล เช่น Microsoft Access, Oracle, My SQL หรือ SQL Sever เป็นต้น โดยมีส่วนประกอบต่างๆ ดังนี้




**รูปที่ 2.3** การทำงานของ DBMS เชื่อมโยงระหว่างผู้ใช้งานกับระบบฐานข้อมูล

**DBMS หรือ ระบบจัดการฐานข้อมูล** จะทำหน้าที่เป็นตัวกลางระหว่างฐานข้อมูลกับโปรแกรมที่มาใช้งานฐานข้อมูลและผู้ใช้งานฐานข้อมูล ที่ติดต่อไปยังฐานข้อมูลเพื่อทำงานที่ผู้ใช้ต้องการให้สำเร็จ เช่น การจัดเก็บข้อมูลลงในฐานข้อมูล,การค้นหาข้อมูลที่ต้องการออกมาแสดงหรือการลบข้อมูล เป็นต้น

**หน้าที่ของระบบจัดการฐานข้อมูล (DBMS)**

1. จัดการพจนานุกรมของข้อมูล (Data dictionary management)
2. จัดการการจัดเก็บข้อมูล (Data storage management)
3. การแปลงข้อมูลและการนำเสนอข้อมูล (Data transformation and presentation)
4. การจัดการด้านความปลอดภัย (Security management)
5. ควบคุมการเข้าใช้งานของผู้ใช้พร้อมกัน(Multiuser access control)
6. การจัดการเรื่องการสำรองและกู้คืนข้อมูล (Backup and recovery management)
7. การจัดการความคงสภาพของข้อมูล (Data integrity management)
8. ภาษาในการเข้าถึงข้อมูลและส่วนประสานผู้ใช้ในโปรแกรมประยุกต์ (Database access languages and application programming interfaces)

# บทที่ 3

# การวิเคราะห์และออกแบบระบบ

## 3.1 วิเคราะห์ระบบงานปัจจุบัน

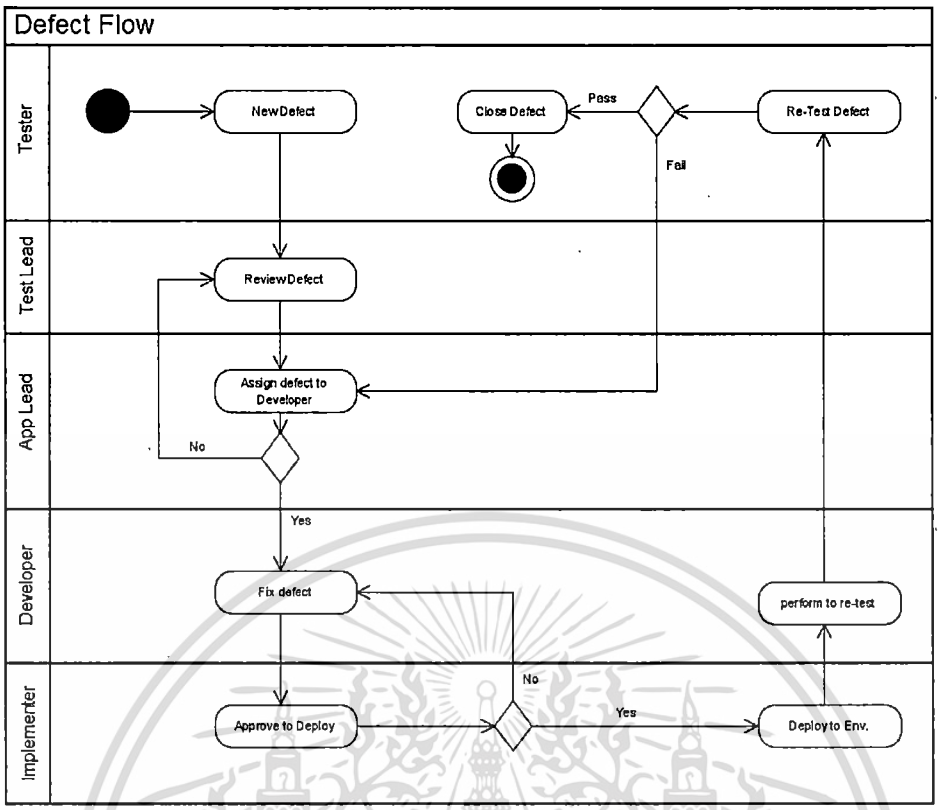
จากการศึกษาระบบงานปัจจุบันในหน่วยงาน Test Management มีขั้นตอนการทำงานโดยเริ่มต้นจากวงจรชีวิตของการพัฒนาซอฟต์แวร์ (System Development Life Cycle : SDLC) ซึ่งประกอบไปด้วยขั้นตอนต่างๆ หลายๆ ส่วนมาประกอบกัน แต่ละโครงการจะมีรายละเอียดปลีกย่อยแตกต่างกันไปตามขนาด หรือความซับซ้อนของโครงการ วงจรการพัฒนาระบบ หรือ SDLC ประกอบไปด้วย

1. การกำหนดปัญหา(Problem Definition)
2. การวิเคราะห์ปัญหา(Analysis)
3. การออกแบบ(Design)
4. การพัฒนาระบบงาน(Development)
5. การทดสอบ(Testing)
6. การติดตั้ง (Deployment)
7. การบำรุงรักษา(Maintenance)

สำหรับระบบการทำงานปัจจุบัน ของหน่วยงาน Test Management ในส่วนงานของผู้ทดสอบระบบ ที่ทำการบันทึกข้อผิดพลาดลงในเอกสาร Excel (ชื่อเอกสาร : Problem Log Report-Template-V01.00-20071105) โดยมีรูปแบบดังรูปที่ 3.1 ซึ่งจะมีการเก็บข้อมูลไม่ครบถ้วนเท่าไรนัก และขั้นตอนการทำงานของผู้ที่เกี่ยวข้องในระบบงานปัจจุบันดังรูปที่ 3.2

| Problem Log Report (Detail) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Project Name/UCR# | | | | | | | | | | | | |
| Test Level | | | | | | | | | | | | |
| Test Environment | | | | | | | | | | | | |
| Test Period | | | | | | | | | | | | |
| Log# | Log Title | Description | Serverity | Reported By | Assign To. | Status | Problem Solving | | | No. of Occurrence | Remark |
| | | | | | | | Detected | Solved | Duration | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |

**รูปที่ 3.1** เอกสารสำหรับบันทึกข้อผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบ




**รูปที่ 3.2** แสดงแผนภาพขั้นตอนการทำงานของผู้ที่เกี่ยวข้องในระบบงานปัจจุบัน

## 3.2 ปัญหาที่พบได้จากการวิเคราะห์ระบบงานปัจจุบัน

1. ความยากลำบากในการติดตามความก้าวหน้าของแต่ละข้อผิดพลาดที่ตรวจพบ เนื่องจากปัจจุบันจะทำการส่งเอกสาร Problem Log Report ทาง E-Mail ให้ผู้ที่เกี่ยวข้องให้โครงการทราบ ทำให้ผู้ที่ต้องการทราบถึงความก้าวหน้าต้องรอรับการรายงานผลการทำสอบจากทาง E-Mail เท่านั้น

2. การจัดเก็บข้อมูลของระบบงานปัจจุบันไม่สามารถนำมาใช้ประโยชน์ได้อย่างรวดเร็ว เนื่องจากไม่ได้เป็นฐานข้อมูล ซึ่งจะนำมาวิเคราะห์และใช้ประโยชน์โดยรวมได้

3. ไม่สามารถตรวจสอบได้ว่าข้อผิดพลาดที่ตรวจพบนั้นจะได้รับการดำเนินการแก้ไขหรือมีผู้รับผิดชอบในข้อผิดพลาดนั้นแล้ว

4. หัวหน้าหน่วยงานไม่มีระบบที่ช่วยจัดเก็บข้อมูลการจัดสรรโครงการให้กับพนักงาน

5. ผู้ทดสอบระบบจัดทำเอกสารที่ใช้ในแต่ละวัน โดยนำเอกสารเก่ามาทำการแก้ไข หรือเพิ่มข้อผิดพลาดที่ตรวจพบ

6. สิ้นเปลืองเวลาในการค้นหาเอกสารหากต้องการจะนำมาตรวจสอบภายหลังหรืออ้างอิง

## 3.3 การออกแบบการทำงานของระบบใหม่

การวิเคราะห์และออกแบบระบบงานใหม่นี้ ได้ดำเนินการตามหลักการวิเคราะห์และออกแบบเชิงวัตถุโดยใช้ยูเอ็มแอล โดยแผนภาพที่ใช้อธิบายการทำงานของระบบนั้นประกอบด้วยยูสเคสไดอะแกรม แอกทิวิตีไดอะแกรม แลคลาสไดอะแกรม โดยมีรายละเอียดการวิเคราะห์ และออกแบบระบบงานใหม่ดังต่อไปนี้

### 3.3.1 ยูสเคสไดอะแกรม (Use Case Diagram)

ยูสเคสไดอะแกรมใช้สำหรับอธิบายความต้องการของระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบระบบให้มีความชัดเจนขึ้น แสดงให้เห็นภาพรวมว่าผู้ใช้จะนำระบบไปใช้อะไรบ้าง ซึ่งเป็นบอกถึงเป้าหมายของผู้ใช้งาน

ยูสเคสไดอะแกรม ประกอบด้วย

1. แอกเตอร์ (Actor) แทนสัญลักษณ์รูปคน แสดงถึง ผู้มีความสัมพันธ์กับระบบในที่นี้ไม่ได้หมายถึงบุคคลเพียงอย่างเดียว อาจจะหมายถึงระบบจากภายนอกอื่นก็ได้ โดยแอกเตอร์ ที่เกี่ยวข้องกับระบบมี 6 แอกเตอร์ ได้แก่

   1.1 Tester (ผู้ทดสอบโปรแกรม) คือ พนักงานที่มีหน้าที่ทำการทดสอบโปรแกรมของบริษัททั้งหมด โดยมีหน้าที่การทำงานดังนี้

   - เมื่อมีการพัฒนาโปรแกรมใหม่ขึ้นมาใช้งานในทุก ๆ ครั้ง ผู้ทดสอบโปรแกรม จะทำการทดสอบโปรแกรม ทุกฟังก์ชั่น
   - หากมีข้อผิดผลาดของโปรแกรมเกิดขึ้น ผู้ทดสอบโปรแกรมจะต้องทำการบันทึกข้อผิดผลาดนั้นลงในระบบ
   - หลังจากผู้พัฒนาระบบได้ทำการแก้ไขระบบแล้ว ผู้ทดสอบโปรแกรมจะต้องทำจะทดสอบโปรแกรมอีกครั้ง
   - หากผู้ทดสอบโปรแกรมยังตรวจพบข้อผิดผลาด สามารถทำการบันทึก หรือแก้ไขการบึกทึกนั้นและทำการส่งให้นักพัฒนาระบบต่อไป

   1.2 Test Lead (หัวหน้าสายงานของผู้ทดสอบโปรแกรม) คือ พนักงานในส่วนงานของฝ่าย Test Management โดยมีหน้าที่การทำงานดังนี้

   - ทำการตรวจสอบข้อผิดผลาดที่ผู้ทดสอบโปรแกรมได้ลงบันทึกอีกครั้งก่อนจะทำการส่งต่อให้กับหัวสายงายของผู้พัฒนาระบบ

- หากพบว่ารายละเอียดจากผู้ทดสอบโปรแกรม ทำการบันทึกรายละเอียดของโปรแกรมไม่ครบถ้วน หัวหน้าสายงานของผู้ทดสอบโปรแกรมต้องเพิ่มเติม หรือแก้ไขรายละเอียดนั้นก่อนส่งต่อให้กับหัวสายงายของผู้พัฒนาระบบ
- ส่งข้อผิดผลาดที่ได้ทำการตรวจสอบแล้วให้กับหัวสายงายของผู้พัฒนาระบบ

1.3 App Lead (หัวหน้าสายงานของผู้พัฒนาระบบ) คือ พนักงานในส่วนแอพพริชั่น โดยมีหน้าที่การทำงานดังนี้

- ทำการตรวจสอบข้อผิดผลาดในระบบจากส่วนงานของการทำการทดสอบระบบ และทำการระบุผู้รับผิดชอบเกี่ยวกับข้อผิดผลาดในส่วนนี้

1.4 Developer (ผู้พัฒนาระบบในส่วนเขียนโปรแกรม) คือ พนักงานที่ทำการพัฒนาระบบที่ได้รับความต้องการมาจากผู้ใช้งานระบบ โดยมีหน้าที่การทำงานดังนี้

- ทำการแก้ไขข้อผิดผลาดที่ถูกตรวจพบจาก ผู้ทดสอบโปรแกรม และได้มีการตรวจสอบเบื้องต้นแล้ว จากหัวหน้าในสายงานของตนเอง ว่าเป็นหน้าที่รับผิดชอบในส่วนนี้จริง
- เมื่อทำการแก้ไขเรียบร้อยแล้ว ต้องส่งต่อให้ผู้พัฒนาระบบในส่วนปฏิบัติการเพื่อดำเนินการในขั้นตอนต่อไป

1.5 Implementer (ผู้พัฒนาระบบในส่วนปฏิบัติการ) มีหน้าที่ทำให้ผู้ใช้งานสามารถใช้งานได้จริง

1.6 Admin คือ ผู้ดูแลระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทดสอบ โดยมีหน้าที่การทำงานดังนี้

- ทำการเพิ่ม, ลบ หรือแก้ไข รายละเอียดต่าง ๆ ของผู้ใช้งานที่เกี่ยวข้องกับโปรเจคเข้าสู่ระบบ
- กำหนดสิทธิ์ของผู้ใช้งาน
- ทำการเพิ่มโปรเจคให้สอดคล้องกับโปรเจคที่เกิดขึ้นในบริษัท

2 ยูสเคส (Use Case) แทนด้วยสัญลักษณ์ วงรี แสดงถึง ฟังก์ชันการทำงานของระบบ บอกได้ว่าระบบสามารถทำอะไรได้บ้าง ยูสเคสจะได้มาจากความต้องการของระบบเป็นหลัก โดยยูสเคสที่เกี่ยวข้องกับการทำงานของระบบ มี 9 ยูสเคส ดังนี้

2.1 New Defect คือ ยูสเคสสำหรับการลงบันทึกความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างทำการทดสอบ ซึ่งในขั้นตอนนี้จะถูกกระทำมาจากผู้ทดสอบโปรแกรมเท่านั้น

โดยผู้ทดสอบโปรแกรมเป็นผู้ทำการบันทึกข้อผิดผลาดลงในระบบ ซึ่งผู้ทดสอบโปรแกรมจะต้องทำการกรอกข้อมูลให้ครบถ้วนตามความต้องการของโปรแกรมกำหนดไว้ หลังจากนั้นระบบจะทำการบันทึกไว้ในฐานข้อมูล

2.2 Search คือ ยูสเคสสำหรับค้นหาข้อมูลความผิดพลาดของโปรแกรม เพื่อทำการสืบค้นข้อมูลในระบบให้มีความรวดเร็วขึ้น ทางผู้ศึกษาจะทำเป็นแบบการคัดกรองข้อมูลเพื่อใส่ค่าที่ต้องการค้นหา ทำให้การทำงานมีประสิทธิภาพยิ่งขึ้น

2.3 View Defect คือ ยูสเคสสำหรับดูรายละเอียดต่าง ๆ ของข้อมูลที่อยู่ในระบบ เนื่องจากการส่งต่อของมูลจะต้องมีรายละเอียดให้กับผู้ที่เข้ามาใช้งานระบบดู เพื่อทำความเข้าใจและปรับแก้ในขั้นตอนต่อไป

2.4 Update Defect คือ ยูสเคสสำหรับทำการปรับปรุงสถานะของข้อมูล ที่อยู่ในระบบ ซึ่งสถานะของข้อมูลที่อยู่ในระบบจะเปลี่ยนไปในทุกครั้งที่มีการอัพเดรตจากผู้ที่เกี่ยวข้องกับโปรเจคนั้น

2.5 Assign Defect คือ ยูสเคสสำหรับกำหนดผู้รับผิดชอบในความผิดพลาดที่เกิดในกับผู้พัฒนาระบบ ซึ่งในส่วนนี้ผู้ที่มีสิทธิ์ในการกำหนดผู้รับผิดชอบจะต้องเป็นหัวหน้าในสายงานเท่านั้น

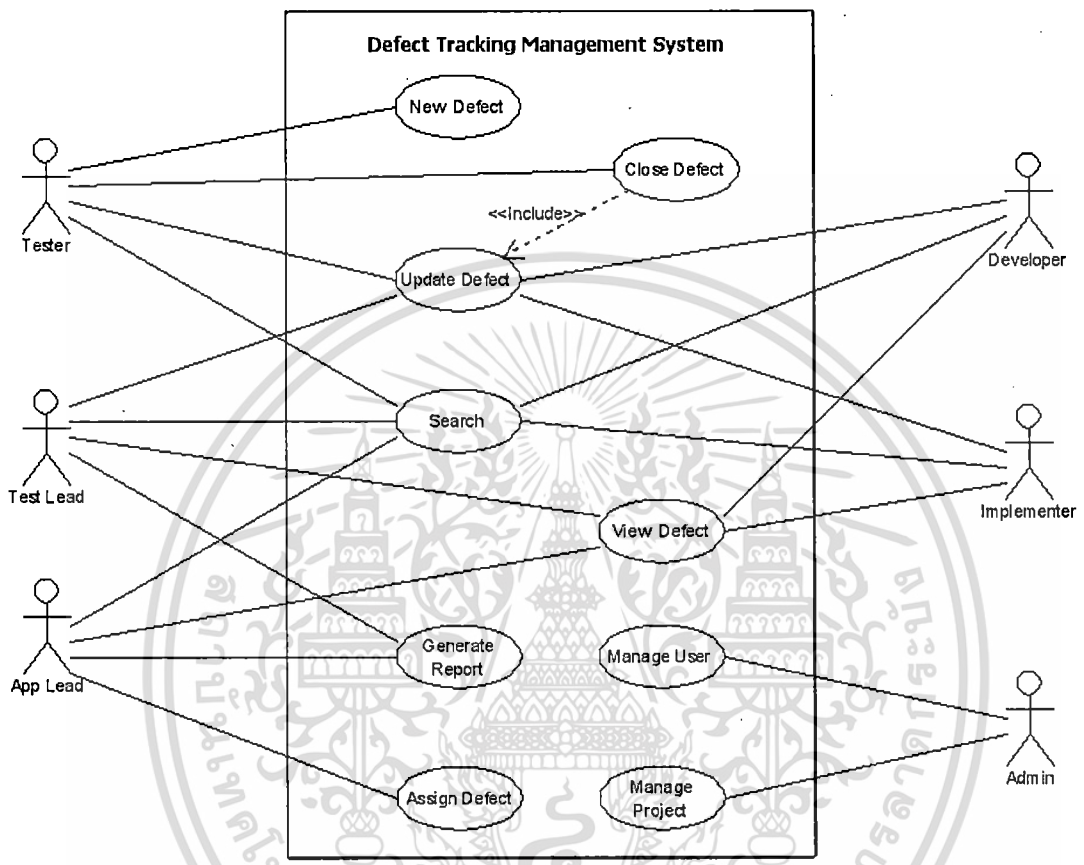
2.6 Close Defect คือ ยูสเคสสำหรับปิดบันทึกความผิดพลาดของโปรแกรม หลังจากผู้พัฒนาได้มีการแก้ไขโปรแกรมแล้ว ทางผู้ทำการทดสอบโปรแกรมทำการทดสอบอีกครั้ง แล้วทำการอัพเดรตสถานะ ซึ่งยูสเคสนี้ผู้ทดสอบโปรแกรมจะมีสิทธิ์อัพเดรตสถานเป็นปิดบันทึกความผิดพลาดของโปรแกรม ได้เพียงผู้เดียว

2.7 Generate Report คือ ยูสเคสสำหรับการจัดการรายงาน และแสดงรายงาน โดยแสดงรายงานได้ตามรายละเอียดที่ต้องการออกทางหน้าจอ

2.8 Manage User คือ ยูสเคสสำหรับการจัดการผู้ใช้งานทั้งหมดของระบบ การทำงานในยูสเคสนี้กล่าวถึง การเพิ่ม ลบ หรือแก้ไข รายละเอียดของผู้ใช้งานทั้งหมดที่เกี่ยวข้อง

2.9 Manage Project คือ ยูสเคสสำหรับการจัดการโปรเจคทั้งหมดของระบบ การทำงานในยูสเคสนี้กล่าวถึง การเพิ่ม ลบ หรือแก้ไข รายละเอียดของโปรเจคทั้งหมดจากความต้องการจากผู้ใช้

3 เส้นแสดงความสัมพันธ์ (Relationship) เป็นการแสดงความสัมพันธ์ระหว่างแอกเตอร์ กับยูสเคส โดยลากเส้นจากแอกเตอร์ไปยังยูสเคส โดยมีความสัมพันธ์กำกับว่าทั้งสองส่วนสัมพันธ์กันอย่างไร




**รูปที่ 3.2** ยูสเคสไดอะแกรมของระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบระบบ

จากแอคเตอร์แลยูสเคสของระบบ ซึ่งประกอบด้วย 9 ยูสเคส จะใช้ตารางยูสเคสมาช่วยอธิบายลำดับการทำงานของแต่ละยูสเคส เพื่อช่วยสร้างความเข้าใจถึงการทำงานของยูสเคส และสามารถนำไปช่วยในการตรวจสอบการพัฒนาระบบงานให้สอดคล้องกับความต้องการของผู้ใช้ระบบ มีรายละเอียดของแต่ละตาราง ดังต่อไปนี้

**ตารางที่ 3.1** รายละเอียดยูสเคส New Defect

| | | |
|---|---|---|
| **Use Case Name:** New Defect | **ID** 1 | **Important Level:** High |
| **Primary Actor:** Tester | | **Use Case Type:** Essential |
| **Stakeholder and Interests:** Tester | | |
| **Brief Description:** ขั้นตอนการลงบันทึกความผิดพลาดของโปรแกรม โดยเทสเตอร์ (Tester) เท่านั้น | | |
| **Precondition:** - | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** Tester<br>**Include:**<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. ผู้ใช้งานเป็นผู้ลงบันทึกโดยกรอกข้อมูลที่พบความผิดพลาดลงในระบบ<br>2. ระบบทำการตรวจสอบข้อมูลครบถ้วน ตามข้อกำหนดของระบบ<br>**Sub flows:**<br>1.1 ระบบทำการบันทึกข้อมูลลงในฐานข้อมูล<br>1.2 ระบบส่งข้อมูลต่อไปให้กับผู้ใช้งานท่านอื่น ตามข้อกำหนดของระบบ<br>จบการทำงานยูสเคสนี้<br>**Alternate/Exceptional Flows:** - | | |

**ตารางที่ 3.2** รายละเอียดยูสเคส Search

| | | |
|---|---|---|
| **Use Case Name:** Search | **ID** 2 | **Important Level:** High |
| **Primary Actor:** Tester, TestLead, App Lead, Developer, Implementer | | **Use Case Type:** Essential |
| **Stakeholder and Interests:** Tester, TestLead, App Lead, Developer, Implementer | | |
| **Brief Description:** ค้นหาข้อมูลความผิดพลาดของโปรแกรม ที่ได้ทำการบันทึกไว้ในระบบ | | |
| **Precondition:** ผ่านขั้นตอนของการ New Defect เรียบร้อยแล้ว | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** Tester, TestLead, App Lead, Developer, Implementer<br>**Include:**<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. ผู้ใช้งานสามารถค้นหาข้อมูลในฐานข้อมูลได้ โดยกำหนดเกณฑ์การค้นหา และระบุคำค้น<br>2. ระบบทำการการค้นหาข้อมูลที่อยู่ในฐานข้อมูล<br>**Sub flows:**<br>2.1 ระบบทำการแสดงผลที่ได้จากการระบุคำค้น ตามเกณฑ์การค้นหา<br>2.2 ผู้ใช้งานดำเนินการในขั้นตอนต่อไป<br>จบการทำงานยูสเคสนี้<br>**Alternate/Exceptional Flows:** - | | |

**ตารางที่ 3.3** รายละเอียดยูสเคส View Defect

| | | |
|---|---|---|
| **Use Case Name:** View Defect | **ID** 3 | **Important Level:** High |
| **Primary Actor:** TestLead, App Lead, Developer, Implementer | | **Use Case Type:** Essential |
| **Stakeholder and Interests:** TestLead, App Lead, Developer, Implementer | | |
| **Brief Description:** ดูรายละเอียดต่าง ๆ ของข้อมูลที่อยู่ในระบบ | | |
| **Precondition:** ผ่านขั้นตอนของการ Search เรียบร้อยแล้ว | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** TestLead, App Lead, Developer, Implementer<br>**Include:**<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. ผู้ใช้งานสามารถตรวจสอบความถูกต้อง ความคบถ้วนของข้อมูลที่ถูกจัดเก็บไว้ในฐานข้อมูล<br><br>**Sub flows:**<br>3.1 ระบบทำการแสดงผลที่หน้าจอของผู้ใช้งาน<br>3.2 ผู้ใช้งานดำเนินการในขั้นตอนต่อไป<br><br>จบการทำงานยูสเคสนี้<br>**Alternate/Exceptional Flows:** - | | |

**ตารางที่ 3.4** รายละเอียดยูสเคส Update Defect

| | | |
|---|---|---|
| **Use Case Name:** Update Defect | **ID** 4 | **Important Level:** High |
| **Primary Actor:** Tester, Test Lead, Implementer, Developer | | **Use Case Type:** Essential |
| **Stakeholder and Interests:** Tester, Test Lead, Implementer, Developer | | |
| **Brief Description:** ทำการปรับปรุงสถานะของข้อมูล ที่อยู่ในระบบ | | |
| **Precondition:** ผ่านขั้นตอนของการ Search และ View Defect เรียบร้อยแล้ว | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** Tester, Test Lead, Implementer, Developer<br>**Include:**<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. ผู้ใช้งานสามารถทำการปรับปรุงสถานะของความผิดพลาดที่ตรวจสอบได้<br>2. เลือกสถานะของข้อมูล<br>3. ยืนยันการปรับปรุงข้อมูล<br>**Sub flows:**<br>4.1 ระบบทำการปรับปรุงข้อมูลที่อยู่ในฐานข้อมูล<br>จบการทำงานยูสเคสนี้<br>**Alternate/Exceptional Flows:** - | | |

**ตารางที่ 3.5** รายละเอียดยูสเคส Assign Defect

| | | |
|---|---|---|
| **Use Case Name:** Assign Defect | **ID** 5 | **Important Level:** High |
| **Primary Actor:** App Lead | | **Use Case Type:** Essential |
| **Stakeholder and Interests:** App Lead | | |
| **Brief Description:** กำหนดผู้รับผิดชอบในความผิดพลาดที่เกิดในกับผู้พัฒนาระบบ | | |
| **Precondition:** ผ่านขั้นตอนของการ Search และ View Defect เรียบร้อยแล้ว | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** App Lead<br>**Include:**<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. หัวหน้าสายงานทำการตรวจสอบความผิดพลาดที่เกิดขึ้น<br>2. ทำการ กำหนดผู้รับผิดชอบ ให้กับผู้พัฒนาระบบ<br>**Sub flows:**<br>5.1 ระบบทำการเก็บข้อมูล ผู้รับผิดชอบ ลงในฐานข้อมูล<br>5.2 ระบบทำการส่งข้อมูลและรายละเอียด ให้กับผู้พัฒนาต่อไป<br>จบการทำงานยูสเคสนี้<br>**Alternate/Exceptional Flows:** - | | |

**ตารางที่ 3.6** รายละเอียดยูสเคส Close Defect

| | | |
|---|---|---|
| **Use Case Name:** Close Defect | **ID** 6 | **Important Level:** High |
| **Primary Actor:** Tester | | **Use Case Type:** Essential |
| **Stakeholder and Interests:** Tester | | |
| **Brief Description:** ปิดบันทึกความผิดพลาดของโปรแกรม | | |
| **Precondition:** ผ่านขั้นตอนของการ Search และ View Defect เรียบร้อยแล้ว | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** Tester<br>**Include:** Update Defect<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. ผู้ตรวจสอบโปรแกรมทำการตรวจสอบความผิดพลาดที่ตรวจพบอีกครั้ง<br>2. ทำการปิดบันทึกความผิดพลาดของโปรแกรมที่ตรวจพบ โดยการเปลี่ยนแปลงสถานะ<br>**Sub flows:**<br>6.1 ระบบทำการปรับปรุงข้อมูลที่อยู่ในฐานข้อมูล<br>จบการทำงานยูสเคสนี้<br>**Alternate/Exceptional Flows:** - | | |

**ตารางที่ 3.7** รายละเอียดยูสเคส Generate Report

| | | |
|---|---|---|
| **Use Case Name:** Generate Report | **ID** 7 | **Important Level:** High |
| **Primary Actor:** Test Lead, App Lead | | **Use Case Type:** Essential |
| **Stakeholder and Interests:** Test Lead, App Lead | | |
| **Brief Description:** การเรียกแสดงรายงาน โดยแสดงรายงานได้ตามรายละเอียดที่ต้องการออกทางหน้าจอ เช่น จำนวนความผิดพลาดที่พบในโปรเจคของแต่ละวัน ชี้วัดพฤติกรรมในการทำงานของผู้ปฏิบัติงาน | | |
| **Precondition:** | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** Test Lead, App Lead<br>**Include:**<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. หัวหน้าสายงานทำการเลือกดู รายงาน ตามเงื่อนไขต่างๆ<br>2. ระบบจะทำการสร้างรายงาน และแสดงผลออกที่หน้าจอ | | |
| **Sub flows:** | | |
| จบการทำงานยูสเคสนี้ | | |
| **Alternate/Exceptional Flows:** - | | |

**ตารางที่ 3.8** รายละเอียดยูสเคส Manage User

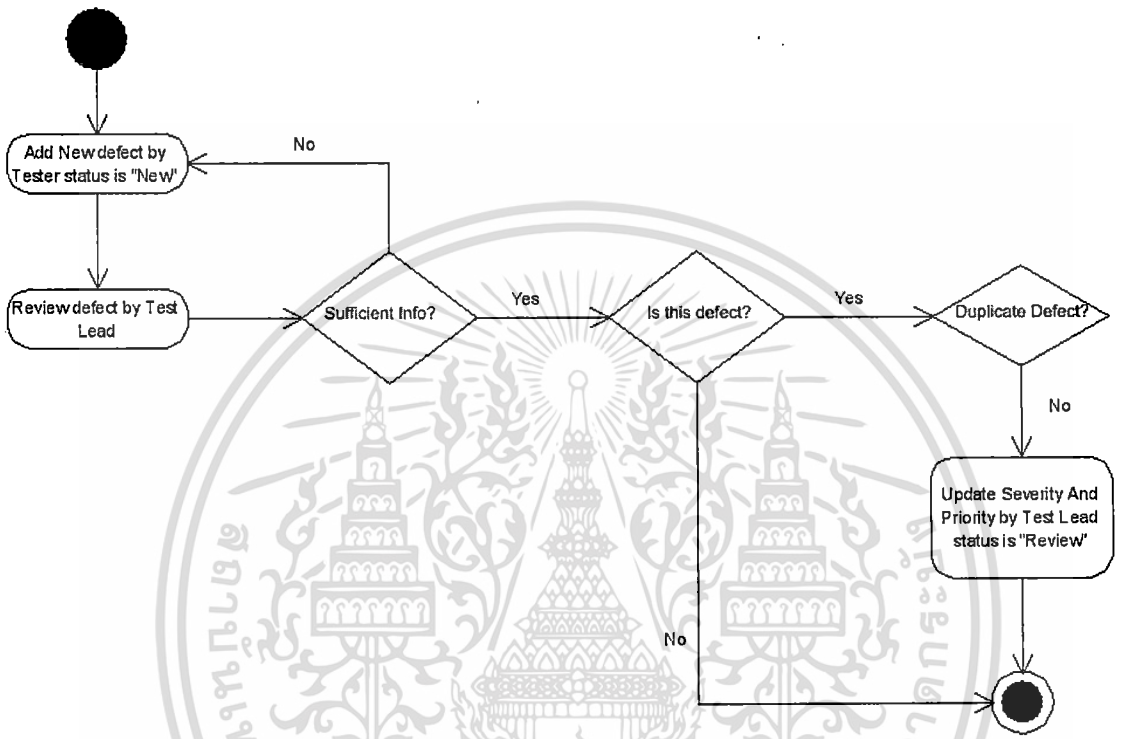
| | | |
|---|---|---|
| **Use Case Name:** Manage User | **ID** 8 | **Important Level:** High |
| **Primary Actor:** Admin | | **Use Case Type:** Essential |
| **Stakeholder and Interests** Admin | | |
| **Brief Description:** การจัดการผู้ใช้งานทั้งหมดของระบบ | | |
| **Precondition:** | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** Admin<br>**Include:**<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. จัดการข้อมูล รายละเอียดต่าง ๆ ของผู้ใช้งาน<br>2. กดบันทึก เพื่อทำการบันทึกลงในระบบ | | |
| **Sub flows:**<br>8.1 ระบบจัดเก็บข้อมูล และละเอียดต่าง ๆ ของผู้ใช้งานลงในฐานข้อมูล | | |
| จบการทำงานยูสเคสนี้ | | |
| **Alternate/Exceptional Flows:** - | | |

**ตารางที่ 3.9** รายละเอียดยูสเคส Manage Project

| | | |
|---|---|---|
| **Use Case Name:** Manage Proj | **ID** 8 | **Important Level:** High |
| **Primary Actor:** Admin | | **Use Case Type:** Essential |
| **Stakeholder and Interests** Admin | | |
| **Brief Description:** การจัดการโปรเจคทั้งหมดของระบบ | | |
| **Precondition:** | | |
| **Trigger:** - | | |
| **Relationship:**<br>**Association:** Admin<br>**Include:**<br>**Extend:**<br>**Generalization:** | | |
| **Normal Flow of Event:**<br>1. จัดการข้อมูล รายละเอียดต่าง ๆ ของโปรเจค<br>2. กดบันทึก เพื่อทำการบันทึกลงในระบบ | | |
| **Sub flows:**<br>8.1 ระบบจัดเก็บข้อมูล และละเอียดต่าง ๆ ของโปรเจคลงในฐานข้อมูล<br>จบการทำงานยูสเคสนี้ | | |
| **Alternate/Exceptional Flows:** - | | |

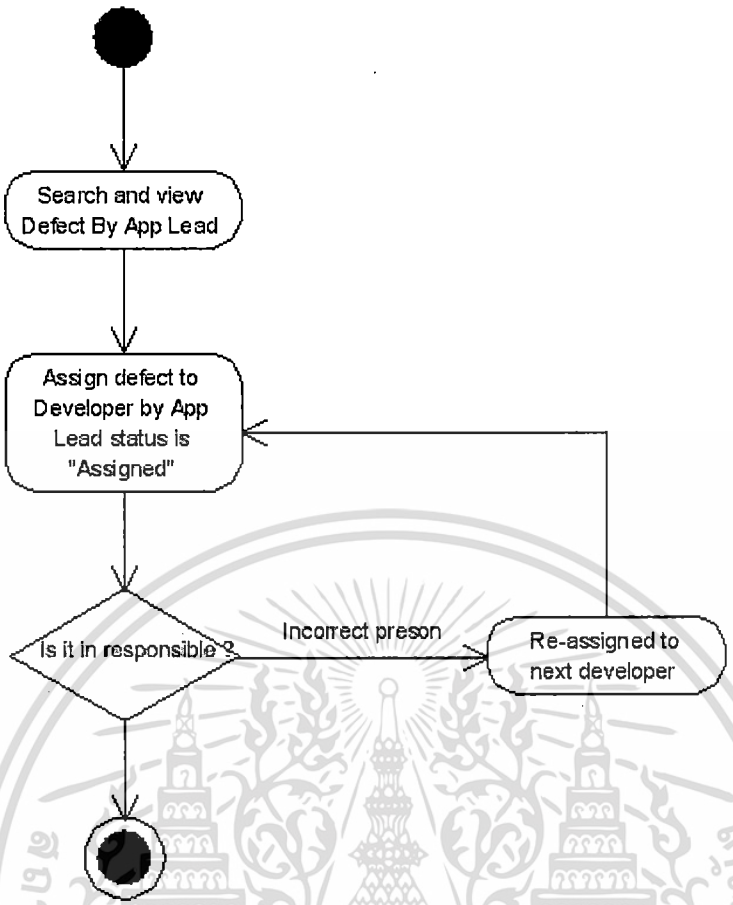
### 3.3.2 แอ็กทิวิตีไดอะแกรม (Activity Diagram)

แอ็กทิวิตีไดอะแกรม แสดงขั้นตอนการปฏิบัติงานหรือกิจกรรมในการปฏิบัติงานของระบบพัฒนาขึ้น โดยมีการแสดงถึงลำดับของกิจกรรม ในระบบรวมถึงจุดที่ต้องตัดสินใจภายในกระบวนการทำงานดังนี้



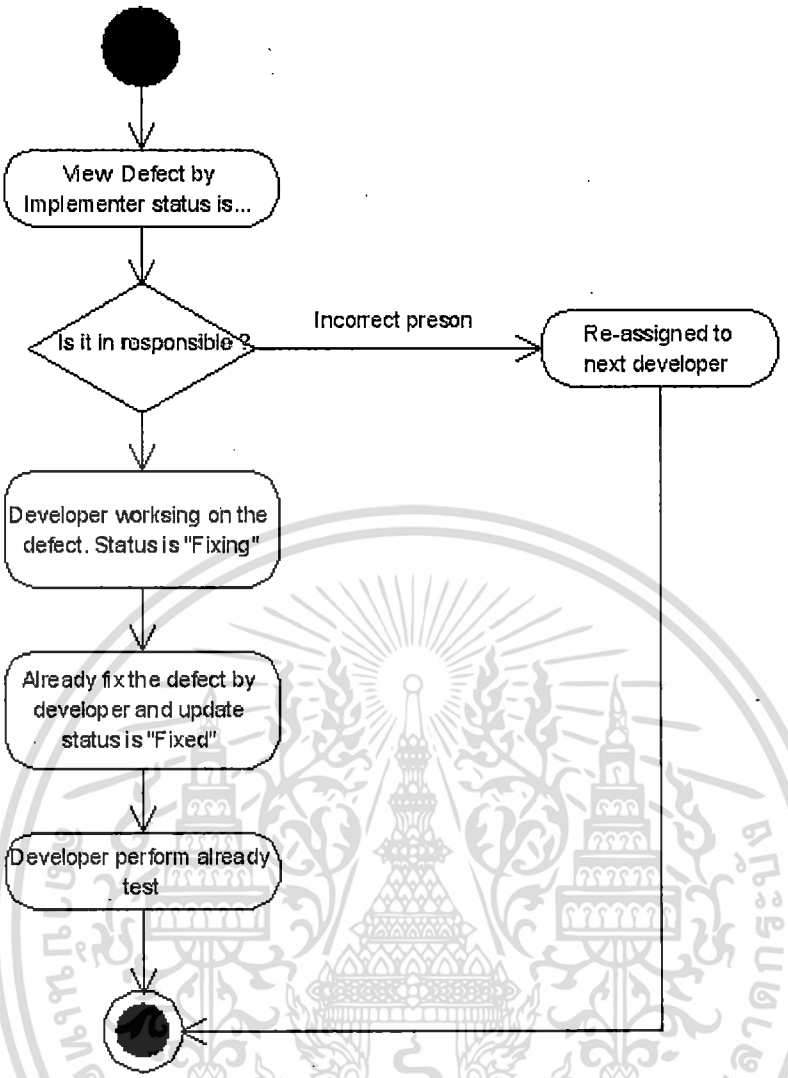

**รูปที่ 3.3** แผนภาพแอ็กทิวิตีไดอะแกรมการลงบันทึกความผิดพลาดของโปรแกรม

ระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทดสอบ เริ่มต้นจากการลงบันทึกความผิดผลาดของโปรแกรมจากผู้ทดสอบโปรแกรม โดยทำการเพิ่มความผิดผลาดลงในระบบ ซึ่งมีสถานะของข้อผิดผลาดนั้นเป็น “New” เท่านั้น หลังจากนั้นข้อผิดผลาดจะถูกตรวจสอบจากหัวหน้าในสายงานอีกครั้ง ว่ามีรายละเอียดเพียงพอหรือไม่ หากพบว่าไม่เพียงพอ หัวหน้าในสายงานสามารถเพิ่ม หรือจะส่งกลับไปให้ทาง ผู้ทดสอบโปรแกรมเป็นผู้กรอกข้อมูลให้คบถ้วนก็ได้ หลังจากทำการตรวจสอบเรียบร้อยแล้ว ทางหัวหน้าสายงานของผู้ทดสอบโปรแกรมทำการกำหนดระดับความรุนแรง และระดับความสำคุณของข้อผิดผลาดนั้น จากนั้นทำการอัพเดรตสถานะเป็น “Review” เป็นอันจบการทำงานของการบันทึกข้อผิดผลาดเข้าสู่ระบบ



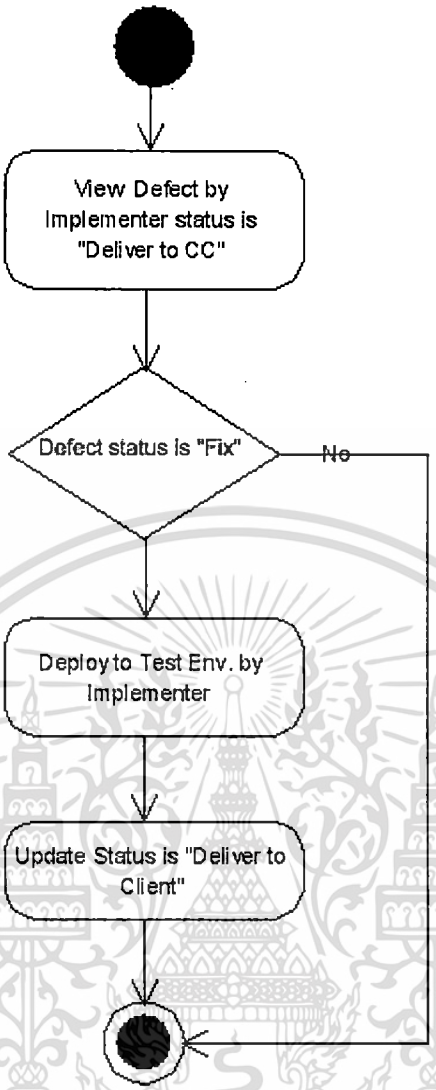

**รูปที่ 3.4** แผนภาพแอ็กทิวิตีไดอะแกรมกำหนดผู้รับผิดชอบในความผิดพลาดที่เกิดในกับผู้พัฒนาระบบ

หลังจากการบันทึกข้อผิดพลาดของโปรแกรมและการตรวจสอบจากหัวหน้าสายงานของผู้ทดสอบโปรแกรมแล้ว ระบบจะทำการส่งต่อไปให้หัวหน้าสายงานของนักพัฒนาระบบ หัวน้าสายงานของนักพัฒนาระบบจะกำหนดผู้รับผิดชอบในความผิดผลาดที่เกิดขึ้น หากกำหนดผู้รับผิดชอบผิดหรือ ข้อผิดผลาดนั้นผู้พัฒนาระบบไม่มีส่วนเกี่ยวข้องทาง หัวหน้าสายงานจะต้องทำการกำหนดผู้พัฒนาระบบอีกครั้ง โดยสถานะในส่วนนี้จะถูกเปลี่ยนเป็น “Assign”



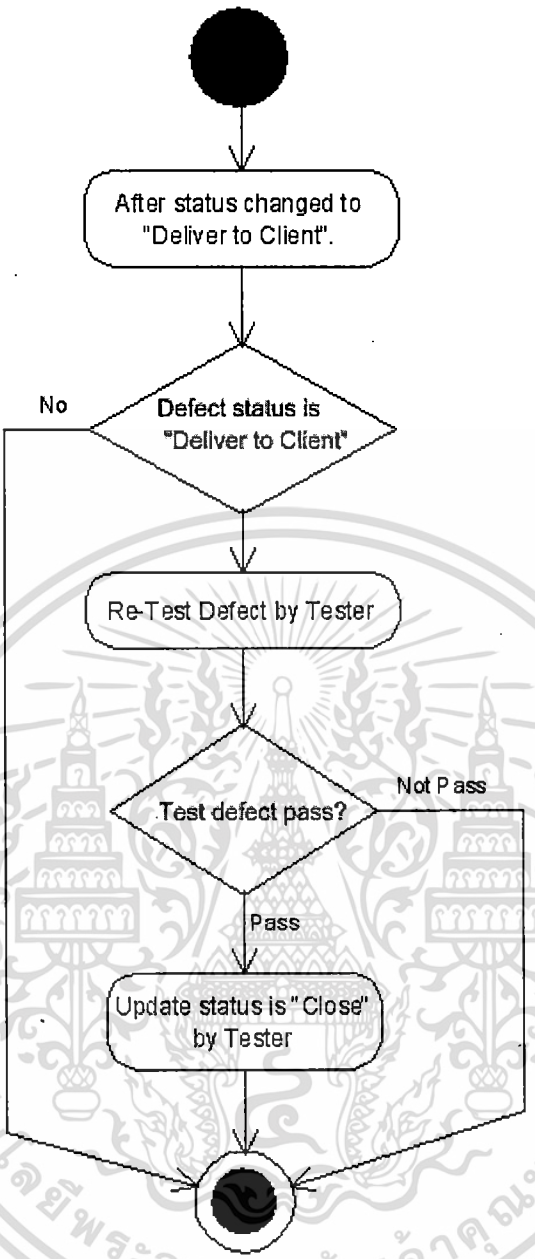

**รูปที่ 3.5** แผนภาพแอ็กทิวิตีไดอะแกรมขั้นตอนการทำงานของผู้พัฒนาโปรแกรมในส่วนเขียนโปรแกรม

ในส่วนของผู้พัฒนาโปรแกรมในส่วนเขียนโปรแกรม เมื่อหัวหน้าสายงานได้ทำการกำหนดผู้รับผิดชอบแล้ว ผู้พัฒนาระบบจะต้องทำการแก้ไขในส่วนงานที่ตนเองรับผิดชอบ และทำการเปลี่ยนสถานะของข้อผิดผลาดจาก “Assign” เป็น “Fix” หลังจากทำการอัพเดรตสถานะเป็น “Fix” แล้วระบบจะส่งต่อให้กับหน่วยงานต่อไปที่เกี่ยวข้อง




**รูปที่ 3.6** แผนภาพแอ็กทิวิตีไดอะแกรมขั้นตอนการทำงานของผู้พัฒนาโปรแกรมในส่วนปฏิบัติการ

ผู้พัฒนาโปรแกรมในส่วนปฏิบัติการ จะทำการเปลี่ยนสถานะเป็น “Deliver to CC” เพื่อตรวจการพัฒนาขึ้นระบบใช้งานจริง หลังจากพัฒนาเสร็จแล้ว ทางผู้พัฒนาโปรแกรมในส่วนปฏิบัติการ จะทำการเปลี่ยนสถานะอีกครั้งเป็น “Deliver to Client”




**รูปที่ 3.7** แผนภาพแอ็กทิวิตีไดอะแกรมขั้นตอนการปิดบันทึกความผิดพลาดของโปรแกรมของผู้ทดสอบโปรแกรม

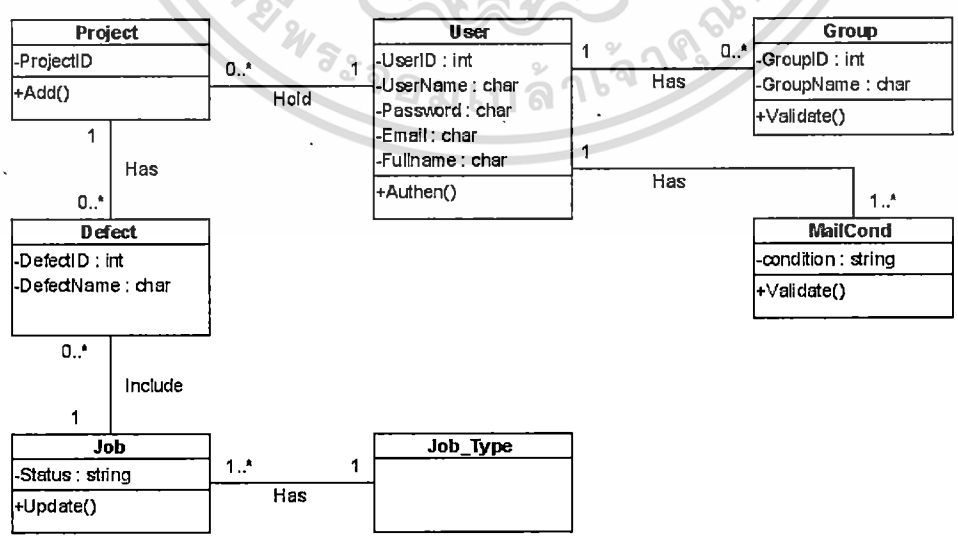
หลังจากสถานะในระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบเปลี่ยน "Deliver to Client" แล้ว หมายความว่าผู้ทดสอบโปรแกรมที่ทำการบันทึกข้อผิดผลาด จะต้องทำการทดสอบใหม่อีกครั้ง หากทำการทดสอบแล้วไม่มีข้อผิดผลาดก็ทำการเปลี่ยนสถานะเป็น "Close"ได้ แต่ถ้ายังพบข้อผิดผลาดที่เกิดขึ้นอีกให้ที่การบันทึกรายละเอียดเพิ่มเติมและเป็นสถานนะเป็น "Open" อีกครั้งระบบจะทำการส่งข้อมูลความผิดผลาดตาม Work Flow ที่กำหนดไว้

### 3.3.3 คลาสไดอะแกรม (Class Diagram)

คลาสไดอะแกรมระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบ ที่ได้จากการวิเคราะห์ความสัมพันธ์ของสิ่งที่เกี่ยวข้องกับระบบ

**ตารางที่ 3.10** คลาสระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบ

| ลำดับที่ | คลาส | คำอธิบาย |
|---|---|---|
| 1 | PROJECT | คลาสโปรเจค |
| 2 | USER | คลาสผู้ใช้งานระบบ |
| 3 | DEFECT | คลาสข้อผิดผลาดที่ถูกบันทึกโดยผู้ทดสอบโปรแกรม |
| 4 | GROUP | คลาสกลุ่มของผู้ใช้งานระบบและสิทธิ์การใช้งานระบบ |
| 5 | MAILCOND | คลาสเงื่อนไขของการส่งอีเมลของผู้ที่เกี่ยวข้องในโปรเจค |
| 6 | JOB | คลาสสถานะของระบบ |
| 7 | JOB_TYPE | คลาสชนิดของสถานะภายในระบบ |




**รูปที่ 3.8** แผนภาพคลาสไดอะแกรมระบบการจัดการความผิดพลาดของโปรแกรม

คลาสของระบบการจัดการความผิดพลาดของโปรแกรมที่เกิดขึ้นระหว่างการทดสอบมีความสัมพันธ์กัน ซึ่งอธิบายความสัมพันธ์ระหว่างคลาสต่าง ๆ ได้ดังนี้

1. คลาส PROJECT สัมพันธ์กับคลาส USER
   คือ โปรเจคที่เกิดขึ้นภายในบริษัทจำเป็นต้องมีผู้ใช้งานในระบบอย่างน้อย 1 คน หรือผู้ใช้งานระบบ อาจจะมีโปรเจคที่เกี่ยวข้องหรือไม่มีก็ได้
2. คลาส PROJECT สัมพันธ์กับคลาส DEFECT
   คือ โปรเจคที่เกิดขึ้นภายในบริษัท อาจจะมีข้อมูลข้อผิดผลาดของระบบหรือไม่มีเลยก็ได้ แต่ข้อผิดผลาดที่ถูดบันทึกโดยผู้ทดสอบโปรแกรมจำเป็นต้องมีโปรเจคเกิดขึ้นในระบบก่อนเสมอ
3. คลาส USER สัมพันธ์กับคลาส GROUP
   คือ ผู้ใช้งานระบบ อาจจะมีกลุ่มของผู้ใช้งานระบบหรือไม่ก็ได้ แต่กลุ่มของผู้ใช้งานระบบและสิทธิ์การใช้งานระบบจำเป็นต้องมีผู้ใช้งานระบบอย่างน้อย 1 คน
4. คลาส USER สัมพันธ์กับคลาส MAILCOND
   คือ ผู้ใช้งานระบบ จำเป็นต้องมีเมลของผู้ใช้งานระบบอย่างน้อย 1 อีเมล แต่อีเมลของผู้ที่เกี่ยวข้องในโปรเจคจำเป็นต้องมีผู้ใช้งานระบบอย่างน้อย 1 คนเสมอ
5. คลาส DEFECT สัมพันธ์กับคลาส JOB
   คือ ข้อผิดผลาดที่เกิดขึ้นในระบบจำเป็นต้องมีสถานะของข้อผิดผลาดนั้น แต่สถานะของข้อผิดผลาดอาจจะไม่จำเป็นต้องมีข้อผิดผลาดก็ได้
6. คลาส JOB สัมพันธ์กับคลาส JOB_TYPE
   คือ สถานะของข้อผิดผลาดจำเป็นต้องมีชนิดของสถานะภายในระบบอย่างน้อย 1 ค่า แต่ชนิดของสถานะภายในระบบอาจจะมีสถานะได้หลายค่าก็ได้

# บทที่ 4

# การออกแบบฐานข้อมูล

การออกแบบระบบฐานข้อมูลสามารถแสดงในรูปของอีอาร์ไดอะแกรม (E-R Diagram) ซึ่งถือว่าเป็นเครื่องมือที่ช่วยให้มองเห็นถึงข้อมูล และความสัมพันธ์ของข้อมูลในระบบได้ หลังจากที่ได้ทำการออกแบบ ประกอบด้วย ยูสเคสไดอะแกรม (Use Case Diagram) จะแสดงรายละเอียดของแอกเตอร์ (Actor), ยูสเคส (Use Case) ต่างๆ และคลาสไดอะแกรม (Class Diagram) ที่ทำหน้าที่แสดงคลาสของระบบว่ามีอะไรบ้าง และมีความสัมพันธ์กันอย่างไร แอ็กทิวิตีไดอะแกรม (Activity Diagram) แสดงถึงกิจกรรมที่เกิดขึ้นภายในระบบในแต่ละขั้นตอน โดยการออกแบบฐานข้อมูลที่เก็บข้อมูลนั้น เป็นระดับข้อมูลทรานแซกชันที่เกิดขึ้นรายวันในการดำเนินการต่างๆ เช่น การเพิ่มจำนวนความผิดพลาดที่ตรวจพบระหว่างการทดสอบ และทำรายการทำงานของผู้ใช้งานแต่ละคนเพื่อนำข้อมูลไปดำเนินการวิเคราะห์ และจัดสรรบุคคลากรให้เหมาะสมกับงานที่ทำอยู่โดยมีตารางดังต่อไปนี้

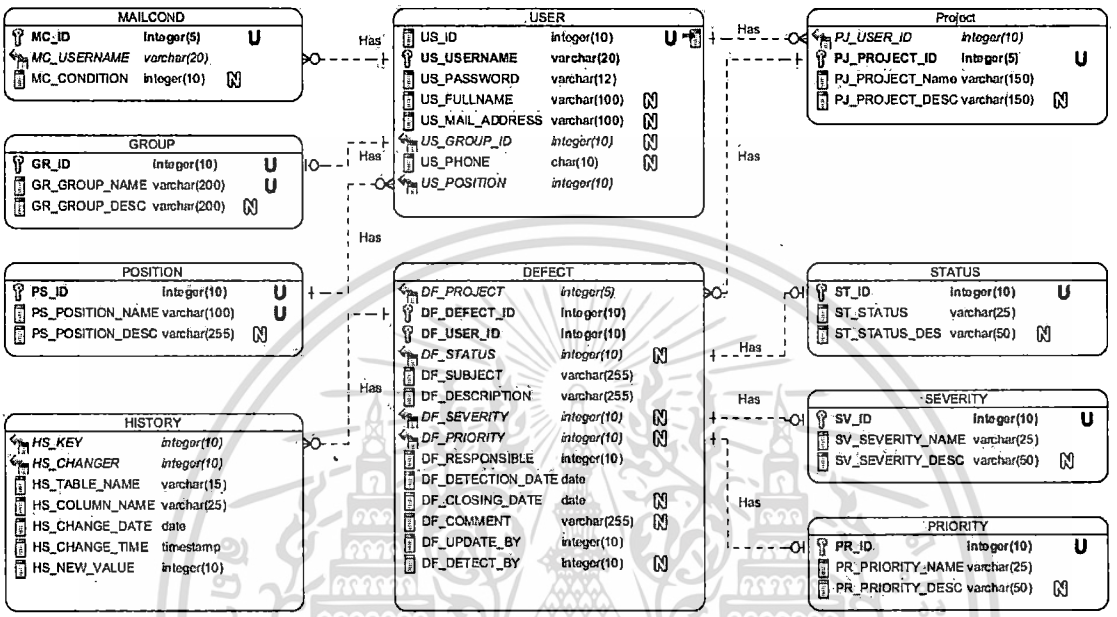
## 4.1 การออกแบบฐานข้อมูล

การออกแบบฐานข้อมูลของระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทำการทดสอบ มีการกำหนดตารางสำหรับเก็บข้อมูลทั้งหมดเป็นส่วนต่างๆ ตามที่คลาสไดอะแกรมกำหนดไว้ ดังต่อไปนี้

1. ตาราง MAILCOND ใช้สำหรับเก็บข้อมูลเงื่อนไขของการส่งอีเมล(E-Mail)
2. ตาราง USER ใช้สำหรับเก็บข้อมูลรายละเอียดของสมาชิกภายในระบบ
3. ตาราง PROJECT ใช้สำหรับเก็บข้อมูลรายละเอียดของโปรเจค
4. ตาราง GROUP ใช้สำหรับเก็บข้อมูลรายละเอียดของกลุ่มภายในระบบ
5. ตาราง POSITION ใช้สำหรับเก็บข้อมูลตำแหน่งของผู้ใช้งานภายในระบบ
6. ตาราง DEFECT ใช้สำหรับเก็บข้อมูลความผิดพลาดที่ตรวจพบจากการทำการทดสอบโปรแกรม
7. ตาราง STATUS ใช้สำหรับการเก็บข้อมูลสถานะของความผิดพลาด
8. ตาราง SEVERITY ใช้สำหรับเก็บข้อมูลระดับความรุ่นแรง
9. ตาราง PRIORITY ใช้สำหรับเก็บข้อมูลระดับความความสำคัญในการแก้ไขความผิดพลาด
10. ตาราง HISTORY ใช้สำหรับเก็บประวัติของข้อมูลที่มีการเปลี่ยนแปลง

จากการวิเคราะห์และออกแบบฐานข้อมูลดังกล่าว สามารถแสดงความสัมพันธ์ระหว่างเอนทริตีในระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทำการทดสอบแบบอีอาร์ไดอะแกรม ดังภาพที่ 4.1

### 4.1.1 อีอาร์ไดอะแกรม (E-R Diagram)




**รูปที่ 4.1** อีอาร์ไดอะแกรมของระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทำการทดสอบ

จากตารางของระบบทั้ง 10 ตารางข้างต้น เมื่อนำมากำหนดคุณสมบัติต่างๆ ของแต่ละตาราง ได้แก่ ฟิลด์ข้อมูล ชนิดของข้อมูล ขนาดของข้อมูล และการอ้างอิงข้อมูลไปยังตารางที่มีความสัมพันธ์กันเพื่อนำข้อมูลเหล่านี้ไปพัฒนาเป็นระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทำการทดสอบ โดยเราจะอธิบายรายละเอียดคุณสมบัติของตารางไว้ที่พจนานุกรมดังรายละเอียดในตารางที่ 4.1 ถึงตารางที่ 4.10 ดังนี้

**ตารางที่ 4.1** MAILCOND ข้อมูลเงื่อนไขของการส่งอีเมล(E-Mail)

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| MC_ID | ลำดับของเงื่อนไข | integer | 5 | PK | |
| MC_USERNAME | ชื่อผู้ใช้งานระบบ | varchar | 20 | FK | USER |
| MC_CONDITION | เงื่อนไขของการส่งเมล | integer | 10 | | |

**ตารางที่ 4.2** USER ข้อมูลรายละเอียดของสมาชิกภายในระบบ

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| US_ID | รหัสผู้ใช้งาน | integer | 10 | | |
| US_USERNAME | ชื่อเข้าใช้งานระบบ | varchar | 20 | PK | |
| US_PASSWORD | รหัสผ่านของระบบ | varchar | 12 | | |
| US_FULLNAME | ชื่อ-นามสกุล ของผู้ใช้งานระบบ | varchar | 100 | | |
| US_MAIL_ADDRESS | เมลของผู้ใช้งาน | varchar | 100 | | |
| US_GROUP_ID | กลุ่มของผู้ใช้งานระบบ | integer | 100 | FK | GROUP |
| US_PHONE | เบอร์โทรศัพท์ของผู้ใช้งาน | char | 10 | | |
| US_POSITION | ตำแหน่งของผู้ใช้งาน | integer | 10 | FK | POSITION |

**ตารางที่ 4.3** PROJECT ข้อมูลรายละเอียดของโปรเจค

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| PJ_USER_ID | ผู้ใช้งานที่อยู่ในโปรเจค | integer | 10 | FK | USER |
| PJ_PROJECT_ID | รหัสของโปรเจค | integer | 5 | | Categories |
| PJ_PROJECT_Name | ชื่อของโปรเจค | varchar | 150 | | |
| PJ_PROJECT_DESC | รายละเอียดของโปรเจค | varchar | 150 | | |

**ตารางที่ 4.4** GROUP ข้อมูลรายละเอียดของกลุ่มภายในระบบ

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| GR_ID | รหัสกลุ่มของระบบงาน | integer | 10 | PK | |

**ตารางที่ 4.4** GROUP ข้อมูลรายละเอียดของกลุ่มภายในระบบ(ต่อ)

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| GR_GROUP_NAME | ชื่อของกลุ่มต่างๆ ในระบบงาน | varchar | 200 | | |
| GR_GROUP_DESC | รายละเอียดต่างๆ ของกลุ่มในระบบงาน | varchar | 200 | | |

**ตารางที่ 4.5** POSITION ข้อมูลตำแหน่งของผู้ใช้งานภายในระบบ

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| PS_ID | รหัสตำแหน่งงาน | integer | 10 | PK | |
| PS_POSITION_NAME | ชื่อตำแหน่งงาน | varchar | 100 | | |
| PS_POSITION_DESC | รายละเอียดของตำแหน่งงาน | varchar | 255 | | |

**ตารางที่ 4.6** STATUS ใช้สำหรับการเก็บข้อมูลสถานะของความผิดพลาด

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| ST_ID | รหัสสถานะ | integer | 10 | PK | |
| ST_STATUS | สถานะ | varchar | 25 | | |
| ST_STATUS_DES | รายละเอียดของสถานะ | varchar | 50 | | |

**ตารางที่ 4.7** DEFECT ใช้สำหรับเก็บข้อมูลความผิดพลาดที่ตรวจพบจากการทำการทดสอบโปรแกรม

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| DF_PROJECT | ชื่อโปรเจค | integer | 5 | FK | PROJECT |
| DF_DEFECT_ID | รหัสข้องข้อผิดผลาดที่ตรวจพบ | integer | 10 | PK | |

| | | | | | |
|---|---|---|---|---|---|
| DF_USER_ID | รหัสของผู้ใช้งานระบบ | integer | 10 | PK | |
| DF_STATUS | สถานะ | integer | 10 | FK | STATUS |
| DF_SUBJECT | ชื่อเรื่องของข้อผิดผลาดที่ตรวจพบ | varchar | 255 | | |
| DF_DESCRIPTION | รายละเอียดของข้อผิดพลาดที่ตรวจพบ | varchar | 255 | | |
| DF_SEVERITY | ระดับความรุนแรง | integer | 10 | FK | SEVERITY |
| DF_PRIORITY | ระดับความสำคัญของข้อผิดผลาดที่ตรวจพบ | integer | 10 | FK | PRIORITY |
| DF_RESPONSIBLE | ผู้รับผิดชอบของแต่ละสถานะ | integer | 10 | | |
| DF_DETECTION_DATE | วันที่ลงบันทึกในระบบ | date | | | |
| DF_CLOSING_DATE | วันที่ทำการปิดข้อผิดพลาดที่ตรวจพบ | date | | | |
| DF_COMMENT | รายละเอียดเพิ่มเติมของผู้ใช้งานระบบ | varchar | 255 | | |
| DF_UPDATE_BY | ผู้ที่ทำการเปลี่ยนแปลง | integer | 10 | | |
| DF_DETECT_BY | ผู้ที่ทำการบันทึกข้อมูลลงระบบคนแรก | integer | 10 | | |

**ตารางที่ 4.8** SEVERITY ข้อมูลระดับความรุนแรง

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| SV_ID | รหัสของความรุนแรง | integer | 10 | PK | |
| SV_SEVERITY_NAME | ชื่อความรุ่นแรง | varchar | 25 | PK | |
| SV_SEVERITY_DES | รายละเอียดของความรุ่นแร | varchar | 50 | | |

**ตารางที่ 4.9** PRIORITY ข้อมูลระดับความความสำคัญในการแก้ไขความผิดพลาด

| ชื่อแอททริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| PR_ID | รหัสของความสำคัญ | integer | 10 | PK | |
| PR_PRIORITY_NAME | ชื่อของความสำคัญ | varchar | 25 | | |
| PR_PRIORITY_DESC | รายละเอียดของความสำคัญ | varchar | 50 | | |

**ตารางที่ 4.10** HISTORY ประวัติของข้อมูลที่มีการเปลี่ยนแปลง

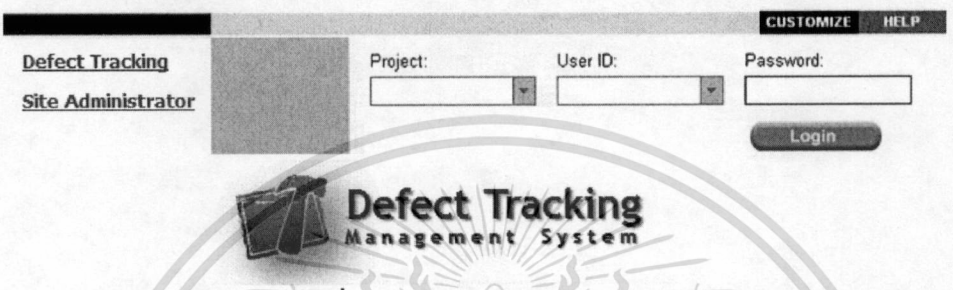
| ชื่อแอท์ทริบิวต์ | คำอธิบาย | ชนิดข้อมูล | ความยาว | คีย์ | ตารางอ้างอิง |
|---|---|---|---|---|---|
| HS_KEY | รหัสประวัติ | integer | 10 | FK | DEFECT |
| HS_CHANGER | บุคคลที่เปลี่ยนแปลง | integer | 10 | FK | DEFECT |
| HS_TABLE_NAME | ชื่อตารางที่เปลี่ยนแปลง | varchar | 15 | | |
| HS_COLUMN_NAME | ชื่อคอลัมน์ที่เปลี่ยนแปลง | varchar | 25 | | |
| HS_CHANGE_DATE | วันที่เปลี่ยนแปลง | date | | | |
| HS_CHANGE_TIME | เวลาที่ทำการเปลี่ยนแปลง | timestamp | | | |
| HS_NEW_VALUE | ค่าที่ทำการเปลี่ยนแปลง | integer | 10 | | |

# บทที่ 5

# หน้าจอการทำงานเบื้องต้น

## 5.1 หน้าจอการทำงานของการจัดการผู้ใช้และผู้ใช้ในโปรเจค

1. เปิด Web browser จากนั้นพิมพ์ URL ของระบบลงไป จะพบกับ Windows Option ของ Defect Tracking Management System จากนั้น click ที่ Defect Tracking ดังรูปที่ 5.1

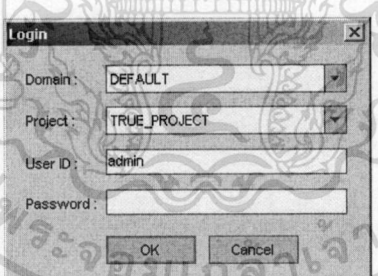



**รูปที่ 5.1** หน้าจอการใช้งานหลัก

2. หลังจาก click ลิงค์ Defect Tracking แล้วจะพบกับหน้า Login ของระบบดังรูปที่ 5.1

3. คลิกที่ CUSTOMIZE ดังรูปที่ 5.1

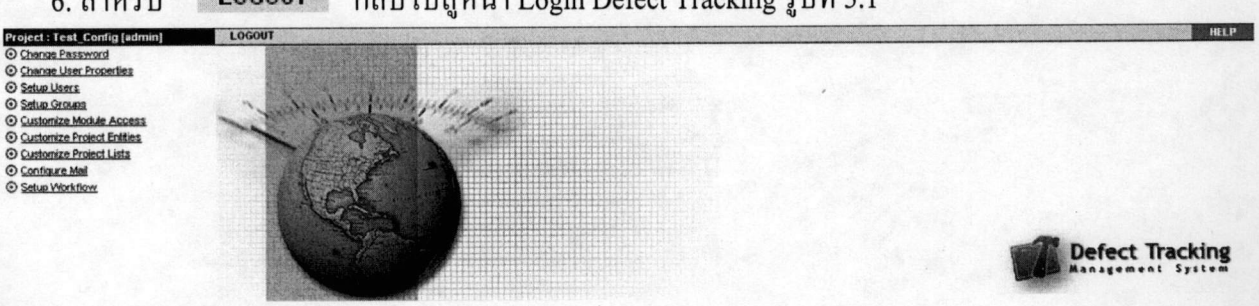
4. เลือก Domain, Project ที่เราต้องการ Setup หรือ Config ค่าต่าง ๆ แล้วใส่ User ID/Password แล้วกด ดังรูที่ 5.2




**รูปที่ 5.2** หน้าจอการลงทะเบียนเข้าใช้งาน

5. หลังจากกด OK แล้วจะพบกับหน้า Project Customization ดังรูปที่ 5.3

6. สำหรับ LOGOUT กลับไปสู้หน้า Login Defect Tracking รูปที่ 5.1




**รูปที่ 5.3** หน้าจอการทำงานทั่วไปของผู้ใช้งาน

## 5.2 เมนูหลักของการจัดการต่างๆ ในโปรเจค (Project Customization)

สำหรับ Menu ต่าง ๆ ของ Project Customization จะประกอบไปด้วย

**1. Change Password** : สำหรับเปลี่ยน password ของ User

**2. Change User Properties** : สำหรับกำหนดรายละเอียดต่าง ๆ ของ User นั้น ๆ

**3. Setup Users** : สำหรับทำการเพิ่มหรือลบ users ของ Project และสามารถ assign users ว่าอยู่ใน groups ใด

**4. Setup Groups** : สามารถ assign privilege ของ user groups โดยการกำหนดตาม permission ของ Groups ที่ทำการ setting ไว้แล้ว รวมถึงการกำหนด transition rules และ data hiding ได้อีกด้วย

**5. Customize Module Access** : การกำหนดว่าสิทธิ์ของ User Group ใด เป็น Defect Tracking หรือ Defect Module *Note : Defect Tracking สามารถใช้งานได้ทุก modules ที่อยู่ใน Defect Tracking ในขณะเดียวกัน Defects Module จะสามารถใช้งานได้แค่ Defect module เท่านั้น*

**6. Customize Project Entities** : สามารถ customize testing environment ได้ตามความต้องการ และสามารถจำกัดหรือกำหนด system fields และ เพิ่ม user fields ได้อย่างเดียวไม่สามารถแก้ไข หรือลบได้

**7. Customize Project Lists** : สำหรับแก้ไข ปรับปรุง เพิ่ม ลด หรือเปลี่ยนแปลง system หรือ user fields ได้

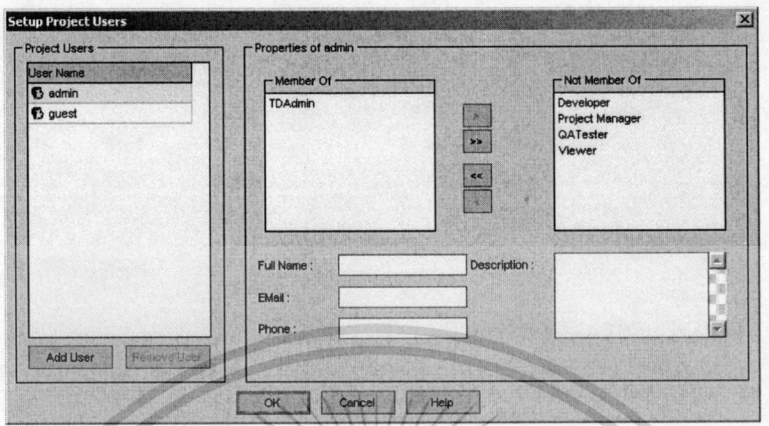
**8. Configure Mail** : สำหรับ set up mailing แจ้งรายละเอียดให้ users ทราบถึง activity ของ defect

**ขั้นตอนและวิธีการใช้งานเกี่ยวกับการ Configuration ค่าต่าง ๆ ของระบบ ด้วยเมนู customization เริ่มจาก**

1. การจัดการ User ใน Project ประกอบไปด้วย
   - Adding a User to a Project
   - Assigning Users to a User Group
   - Removing a User from a Project

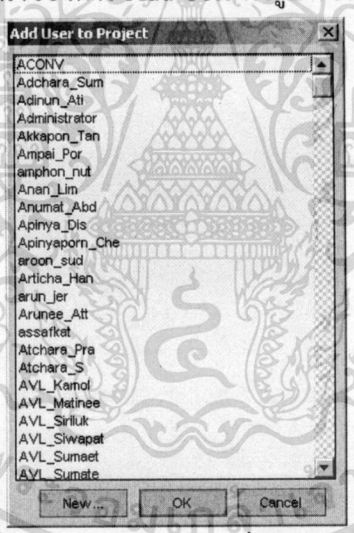
**Adding a User to Project**

1. คลิกลิงค์ Setup Users ในหน้า Project Customization จบพบหน้าของ Setup Project Users ดังรูปที่ 5.7

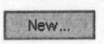



**รูปที่ 5.4** หน้าจอการจัดการผู้ใช้งานในโปรเจค

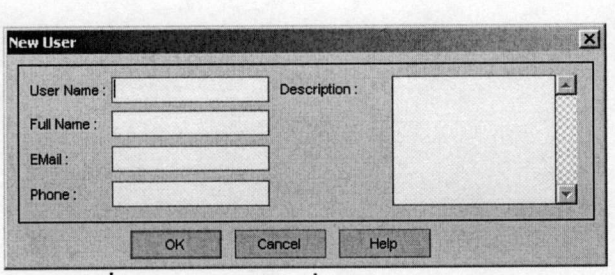
2. คลิกปุ่ม Add User จะพบหน้าของการ Add User ดังรูปที่ 5.8




**รูปที่ 5.5** หน้าจอแสดงรายชื่อของผู้ใช้งาน

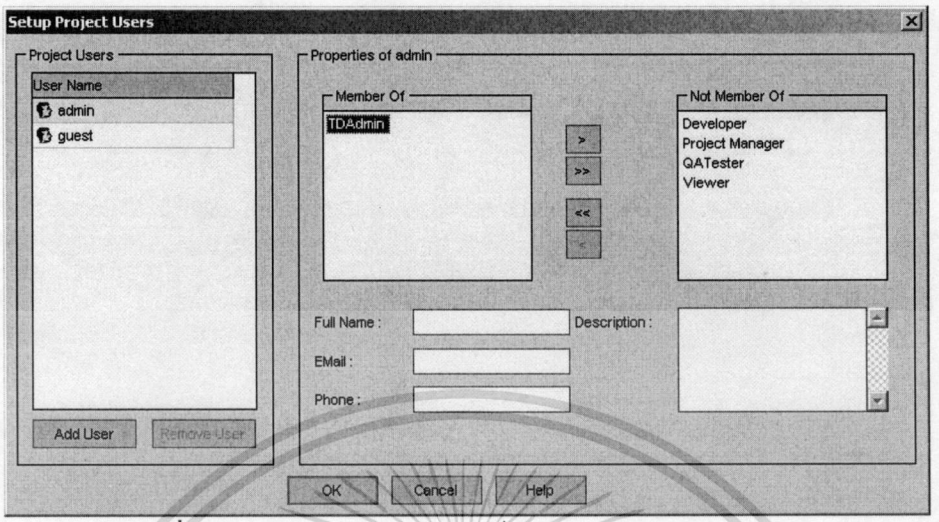
3. เลือก User ที่อยู่ใน list หรือจะทำการเพิ่ม User โดยกดปุ่ม 

จะพบกับหน้าของ New User ดังรูปที่ 5.9 จากนั้นเราสามารถเพิ่ม users และรายละเอียดได้เลย(name, e-mail, phone, and description).




**รูปที่ 5.6** หน้าจอการเพิ่มผู้ใช้งานระบบ

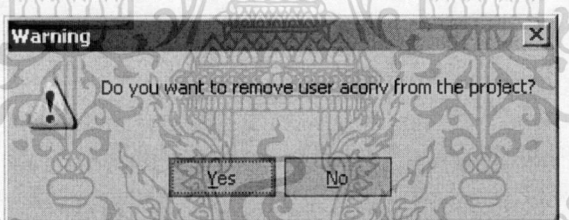
**Assigning Users to a User Group**




**รูปที่ 5.7** หน้าจอการกำหนดสิทธิ์เข้าใช้งานของผู้ใช้งาน

**Popup of Removing a User from a Project**

จะมี Popup message ว่า "Do you want to remove user accov from the project?" ในทำการกดเพื่อยืนยันอีกครั้ง ดังรูปที่ 5.11 *Note : Accov เป็นชื่อของ user ที่เราต้องการจะทำการลบ*




**รูปที่ 5.8** หน้าจอแจ้งเตือน เพื่อยืนยันการลบผู้ใช้งาน

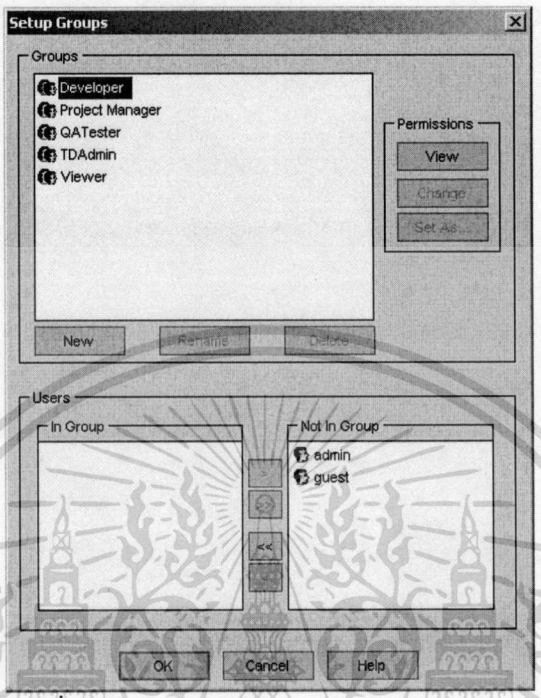
จากนั้นกดปุ่ม OK ในหน้า Setup Project Users เพื่อทำการ save และปิดหน้าต่าง

2. การจัดการกลุ่มของ User และการกำหนดสิทธิ์ของกลุมนั้น ๆ ประกอบไปด้วย

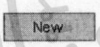
- Adding User Groups
- Assigning Existing Sets of Permissions to User Groups
- Renaming User Groups
- Deleting User Groups
- Understanding the Permission Settings Tasks

**Adding User Groups**

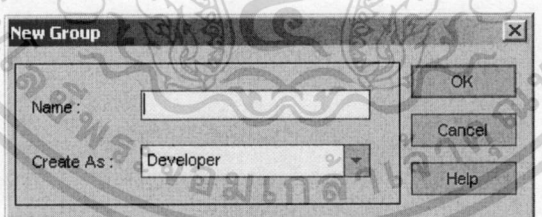
1. คลิกลิงค์ Setup Groups ในหน้า Project Customization จะพบหน้าของ Setup Groups ดังรูปที่ 5.12




**รูปที่ 5.9** หน้าการจัดการกลุ่มของผู้ใช้งาน

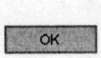
2. กด  ปุ่ม เพื่อทำการ New Group ดังรูปที่ 5.13






**รูปที่ 5.10** หน้าจอการเพิ่มกลุ่มผู้ใช้งานใหม่

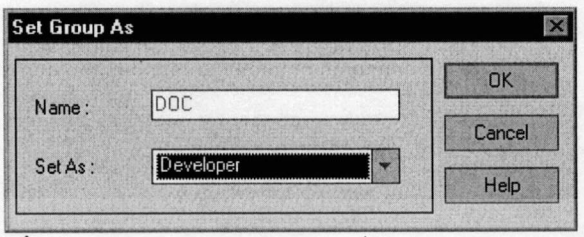
3. กำหนดและพิมพ์ชื่อ ลงในช่อง Name ส่วนช่องของ Create As ให้ทำการเลือกว่า Group ใหม่มีสิทธิ์เท่ากับ Group ใด *Note : Group ใหม่ที่เราทำการ new ขึ้นมาสามารถทำได้แก้ไขในส่วนของ Permissions ได้*

4. กด  ปุ่ม เพื่อทำการ save และปิดหน้า New Group

**Assigning Existing Sets of Permissions to User Groups**

ในขั้นตอนนี้จะอธิบายในส่วนของการ assign สิทธิ์ที่ให้กับ Group user ที่มีอยู่แล้ว

1. เลือกชื่อกลุ่มที่ต้องการ assign สิทธิ์ แล้วกด Set As… จะพบกลับหน้า Set Group As ดังรูปที่ 5.11



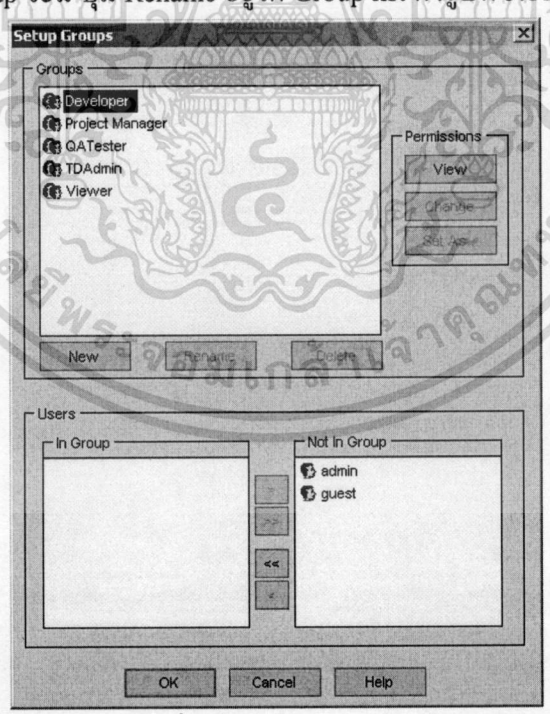

**รูปที่ 5.11** หน้าจอการกำหนดสิทธิ์ของกลุ่มผู้ใช้งาน

2. เลือกกลุ่มใน Set As list แล้ว กดปุ่ม OK เพื่อ save และปิดหน้าของการ Set Group As

**Renaming User Groups**

เราสามารถทำการเปลี่ยนชื่อของ user group ที่ทำการเพิ่มในต้อง Set Group ของ Project ได้ตามต้องการ

1. จากหน้า Setup Group จะมี ปุ่ม Rename อยู่ใต้ Group list ดังรูปที่ 5.11



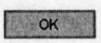

**รูปที่ 5.12** หน้าจอการเปลี่ยนแปลงรายละเอียดของกลุ่มผู้ใช้งาน

2. เลือก Group ที่ต้องการ เปลี่ยนชื่อ หรือแก้ไข คลิกที่ปุ่ม Rename

3. ทำการพิมพ์ชื่อใหม่ที่ต้องการและกด OK เพื่อ Save และจบการทำงาน

**Deleting User Groups**

ในขณะเดียวกันเมื่อเราสามารถแก้ไข เปลี่ยนแปลง และเพิ่ม Group User ได้แล้ว เราก็สามารถลบ Group ของ User ได้เหมือนกัน ทำได้โดย

1. เลือก Group name
2. คลิกปุ่ม Delete
3. กดปุ่ม  เพื่อที่การยืนยัน

# บทที่ 6

# บทสรุป

## 6.1 สรุปผลการวิเคราะห์และออกแบบระบบ

รายงานฉบับนี้ จัดทำขึ้นเพื่อใช้ในการศึกษาการพัฒนาระบบ โดยการออกแบบระบบใหม่ได้มีการนำแนวความคิดเชิงวัตถุและภาษายูเอ็มแอล มาช่วยในวิเคราะห์และออกแบบระบบงานใหม่ ให้มีประสิทธภาพมากขึ้น และจาวา ในการออกแบบหน้าจอซึ่งนับว่าเป็นเทคโนโลยีสมัยใหม่ที่รองรับเครือข่ายสื่อสารในอนาคต โดยนำความรู้ที่ได้มาพัฒนาระบบสารสนเทศ เป็นแบบเว็บแอพพลิเคชั่นผ่านเครื่อข่ายอินทราเน็ท ใช้ SQL Server เป็นระบบจัดการฐานข้อมูล จากการพัฒนาระบบทำให้สามารถทราบถึงสถานะ และสามารถตรวจสอบติดตามขั้นตอนการปฏิบัติงานของความผิดผลาดที่เกิดขึ้นระหว่างการทำการทดสอบได้ ทำให้การทำงานเป็นมาตรฐานเดียวกัน มีความสะดวกในการทำงาน สามารถจัดทำรายงานได้รวดเร็ว โดยประยุกต์เป็นการพัฒนาระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทดสอบ เพื่อช่วยให้การติดตามให้มีประสิทธิภาพมากขึ้น อีกทั้งรองรับเทคโนโลยีในอนาคตด้วย

## 6.2 ประโยชน์ที่ได้รับจากการพัฒนาระบบ

1. ได้ศึกษาเทคโนโลยีจาวา แล้วได้นำมาประยุกต์ใช้ในการพัฒนาระบบการจัดการความผิดพลาดของโปรแกรมระหว่างการทดสอบ

2. สามารถนำความรู้ในการวิเคราะห์และออกแบบระบบด้วยแนวคิดเชิงวัตถุ ภาษายูเอ็มแอล และแปลงให้เป็นตารางความสัมพันธ์ในรูปแบบเชิงสัมพันธ์

3. เพิ่มประสิทธภาพการตรวจสอบติดตามขั้นตอนการปฏิบัติงาน ของหน่วยงาน Test Management บริษัท ทรู คอร์ปอเรชั่น จำกัด (มหาชน) ซึ่งเป็นหน่วยงานที่ได้เคยสังกัดอยู่ให้มีประสิทธิภาพมากยิ่งขึ้น คือ มีฐานข้อมูลกลางเพียงแห่งเดียว ทำให้การปฏิบัติงานของพนักงานสะดวกขึ้น ทำให้การบริหารงานมีประสิทธิภาพมากขึ้น อีกทั้งลดความเสี่ยงเนื่องจากการสูญหายของเอกสาร จากการเก็บไว้ในฐานข้อมูลของหน่วยงาน (Share Drive) หากเกิดความเสียหายของ disk อันเนื่องมาจากสาเหตุต่าง ๆ

4. ได้ศึกษาแนวทางในการพัฒนาระบบสารสนเทศในรูปแบบใหม่ และนำไปประยุกต์ใช้ในการพัฒนาระบบพาณิชย์อิเล็กทรอนิกส์อื่น ๆ ได้ต่อไป

# บรรณานุกรม

นายคิง. 2009. **SQL คืออะไร.** [Online]

Available: http://www.choosak.com/page-tag/mysql-คือ/

อาจารย์นพดล ผู้มีจรรยา. **ทฤษฎีการทดสอบซอฟท์แวร์.** [Online]

Available: home.npru.ac.th/noppadon/4142502/slide/02_modelDevSoftware.ppt

CarsracBot. 2012. **ฐานข้อมูล.** [Online]

Available: http://th.wikipedia.org/wiki/ฐานข้อมูล

Denisarona. 2011. **Software testing.** [Online]

Available: http://en.wikipedia.org/wiki/Software_testing

Konmun.com. 2010. **ฐานข้อมูล คือ ฐานข้อมูลคอมพิวเตอร์ คืออะไร.** [Online]

Available: http://csc.kmitl.ac.th/faq/31-general/217-2553-09-08-03-m-s.html

Luckas-bot. 2012. **การทดสอบซอฟแวร์.** [Online]

Available: http:// th.wikipedia.org/wiki/การทดสอบซอฟแวร์

Namp. 2010. **System Development Life Cycle.** [Online]

Available: http://namp410.blogspot.com/2010/01/sdlc.html

Support. 2011. **SQL คืออะไร.** [Online]

Available: http://www.softwaresiam.com/index.php/access/11-sql

Suthida Chaichomchuen. **Verification & Validation.** [Online]

Available: http://www.na-vigator.com/software-development/software-testing/121-verification-a-validation

Seahorseruler. 2011. **System Development Life Cycle.** [Online]

Available: http://en.wikipedia.org/wiki/Systems_development_life-cycle

Y.Jaruwan. 2001. **ความรู้ทั่วไปเกี่ยวกับระบบฐานข้อมูล.** [Online]

Available: http://www.chandra.ac.th/office/ict/document/it/it04/page01.html

# ประวัติผู้แต่ง

| | |
|---|---|
| **ชื่อผู้เขียน** | นายศราวุธ โสพสกุลางกูร |
| **สถานที่เกิด** | ฉะเชิงเทรา |
| **วุฒิการศึกษาระดับปริญญาตรี** | วิทยาศาสตรบัณฑิต สาขาสถิติประยุกต์ |
| | คณะวิทยาศาสตร์ มหาวิทยาลัยราชภัฏจันทรเกษม |
| **การทำงาน** | บริษัท อินฟินิท คอมพิวติ้ง ซิสเต็มส์ (ไทยแลนด์) จำกัด |