แอพพลิเคชันสำหรับการตรวจสอบสภาพแวดล้อมและการตั้งเวลาทดสอบ

APPLICATION FOR CHECKING THE TEST ENVIRONMENT AND SCHEDULING TEST RUN

โดย

จารุวรรณ อริยะพัฒนพาณิชย์

JARUWAN ARIYAPATTANAPANICH

อาจารย์ที่ปรึกษา

ผศ.ดร.พรฤดี เนติโสภากุล

รายงานนี้เป็นส่วนหนึ่งของวิชาการศึกษาอิสระ2
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ภาคเรียนที่ 2 ปีการศึกษา 2553

APPLICATION FOR CHECKING THE TEST ENVIRONMENT AND SCHEDULING TEST RUN

JARUWAN ARIYAPATTANAPANICH

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

2/2010

หัวข้อ	แอพพลิเคชันสำหรับตรวจสอบสภาพแวดล้อมและการตั้งเวลาการทดสอบ
นักศึกษา	นางสาวจารุวรรณ อริยะพัฒนพาณิชย์
รหัสนักศึกษา	52660721
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
แขนงวิชา	เทคโนโลยีสารสนเทศและการจัดการ
ปีการศึกษา	2553
อาจารย์ที่ปรึกษา	ผศ.ดร.พรฤดี เนติโสภากุล

บทคัดย่อ

การทดสอบซอฟต์แวร์แบบอัตโนมัติเป็นแนวทางหนึ่งที่ทำให้ผู้ทดสอบซอฟต์แวร์สามารถตรวจสอบซอฟต์แวร์ได้อย่างครอบคลุมภายใต้ข้อจำกัดของเวลา ก่อนการทดสอบซอฟต์แวร์แบบอัตโนมัติจะเกิดขึ้นได้นั้นจะต้องมีองค์ประกอบและการจัดเตรียมความพร้อมต่างๆ ที่สำคัญจำนวนมาก โครงการนี้จึงได้พัฒนาแอพพลิเคชันสำหรับเป็นเครื่องมือที่ช่วยในการตรวจสอบและเตรียมสภาพแวดล้อมของการทดสอบ ซึ่งจากเดิมที่ต้องใช้บุคลากรในการจัดเตรียมให้เป็นแบบอัตโมนัติมากขึ้น เพื่อช่วยลดเวลาในการจัดเตรียมสภาพแวดล้อมให้น้อยลง รวมทั้งลดความเสี่ยงจากข้อผิดพลาดที่อาจเกิดขึ้นจากตัวบุคลากรได้อีกด้วย นอกจากนี้จะช่วยให้ผู้ทดสอบมีเวลาศึกษาและวิเคราะห์ปัญหาที่เกี่ยวข้องด้านธุรกิจขององค์กรมากขึ้น อีกทั้งยังเป็นการเพิ่มประสิทธิภาพของจำนวนการทดสอบได้มากยิ่งขึ้น การดำเนินโครงการนี้ได้ศึกษาความต้องการของระบบ นำเสนอการวิเคราะห์และออกแบบระบบตามหลักการวิเคราะห์และออกแบบเชิงวัตถุด้วยยูเอ็มแอล โดยอธิบายรายละเอียดการทำงานของระบบผ่านแบบจำลองต่างๆ รวมถึงได้ดำเนินการออกแบบหน้าจอแสดงผล และพัฒนาแอพพลิเคชันด้วย Virtual Studio.net 2005

Title	Application for checking test environment and schedule runs
Student	Miss Jaruwan Ariyapattanapanich
Student ID.	52660721
Degree	Master of Science
Program	Information Technology
Major	Information Technology and Management
Academic Year	2010
Advisor	Assist. Prof. Dr. Ponrudee Netisopakul

ABSTRACT

Automated software testing is one of a testing approach to help a software tester get more system coverage to test software under the time condition. Despite automated testing can be executed, there are amounts of essential pre-requisite and preparation task. This project has developing an automated application tool for preparing and setting the testing environments by replacing a human effort consumable. Also this application tool can be reduced a risk of human error while preparing and setting the testing environments. Additionally with less time spending on a testing environment preparation, a software tester can utilize their time and pay attention to a company business oriented. Moreover, this will also increase the testing scale and performance. This project has study the system requirements by presenting the analysis and design for all system models with UML and develop a user interface with Virtual Studio.net 2005.

กิตติกรรมประกาศ

โครงการพัฒนาระบบสารสนเทศฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำและคำปรึกษาจาก ผศ.ดร. พรฤดี เนติโสภากุล อาจารย์ที่ปรึกษาโครงการที่กรุณาสละเวลาให้คำปรึกษาและช่วยตรวจสอบแก้ไขข้อบกพร่องของโครงการ ตลอดจนให้ความรู้และข้อคิดเห็นที่เป็นประโยชน์อย่างยิ่ง ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์จากอาจารย์เป็นอย่างมากและขอกราบขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคณาจารย์ภาควิชาเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้าตลอดมา

ขอขอบคุณ คุณ เจษฎา ศรีอโนมัย ที่ช่วยเหลือและให้คำแนะนำความรู้ทางด้าน Visual Studio.Net ที่ใช้ในการพัฒนาระบบด้วยดีตลอดมา

ขอขอบคุณ คุณ บุษประสิทธิ์ พงษ์พิชิต ที่ช่วยให้คำแนะนำทางด้านความต้องการของระบบ ตลอดจนช่วยทดสอบระบบกับสภาพแวดล้อมจริง

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจ และให้การสนับสนุนในทุกๆเรื่อง ทำให้ข้าพเจ้าสามารถจัดทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงเป็นอย่างดี

หากโครงการศึกษาอิสระฉบับนี้ก่อให้เกิดประโยชน์และความดีอันใด ข้าพเจ้าขอมอบให้กับบิดา มารดา และครูอาจารย์ที่เคารพ ผู้ซึ่งถ่ายทอดวิชาความรู้และประสบการณ์แก่ข้าพเจ้า

จารุวรรณ อริยะพัฒนพาณิชย์

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญตาราง(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ด้วยบริษัท ดีเอสที เวิลด์วายด์เซอร์วิส (DST Worldwide Services (Thailand) Limited) เป็นบริษัทที่พัฒนาซอฟต์แวร์ขึ้นมาจำหน่าย โดยซอฟต์แวร์ที่ผลิตออกมานั้นจะอยู่ในรูปแบบของ Release software คือแต่ละเวอร์ชันจะมีการส่งมอบให้กับลูกค้าเป็นระยะๆ ตามความต้องการของลูกค้าเป็นสำคัญ สำหรับลูกค้าของบริษัทมีอยู่ด้วยกันหลายพื้นที่ทั้งในประเทศไทยและต่างประเทศ ทำให้ซอฟต์แวร์ของบริษัทจะต้องสามารถรองรับการทำงานได้หลายสภาพแวดล้อม เช่น ซอฟต์แวร์จะต้องสามารถทำงานได้ดีทั้งบนระบบปฏิบัติการยูนิกซ์ (Unix) หรือวินโดว์ (Windows) เป็นต้น ทางบริษัทได้ให้ความสำคัญในคุณภาพของซอฟต์แวร์ก่อนที่จะทำการส่งมอบให้กับทางลูกค้าเป็นสำคัญ มีทีมงานตรวจสอบคุณภาพซอฟต์แวร์ทั้งแบบทั่วไป (Manual Testing) และแบบอัตโนมัติ (Automated Testing) ที่จะทำการทดสอบคุณภาพของซอฟต์แวร์ก่อนทำการมอบให้กับลูกค้า

ในส่วนของการทดสอบแบบอัตโนมัติทางบริษัทจะมีทีมที่ทำการทดสอบแบบถดถอยหรือ Regression Testing เพื่อตรวจสอบสิ่งที่ลูกค้าขอเพิ่มหรือเปลี่ยนแปลงนั้น ส่งผลกระทบกับซอฟต์แวร์หรือโมดูลใกล้เคียงหรือไม่ ภายในทีมจะมีการนำเครื่องมือทดสอบซอฟต์แวร์ (Software Testing Tool) ต่างๆ มาช่วยในการสร้างสคริปท์เพื่อใช้ทดสอบ การทดสอบแบบอัตโนมัติภายในทีมนั้นก่อนที่จะทำการทดสอบแต่ละครั้งจำเป็นต้องมีการจัดเตรียมสภาพแวดล้อมของการทดสอบ (Test Environment) ต่างๆ ให้เหมือนกับที่ลูกค้าใช้มากที่สุด ด้วยลูกค้าที่มีอยู่หลายพื้นที่ทำให้การทดสอบจะต้องทดสอบให้ครอบคลุมทุกสภาพแวดล้อมมากที่สุด ส่วนประกอบของสภาพแวดล้อมที่จำเป็นต้องเตรียมก่อนการทดสอบนั้นมีด้วยกันหลายอย่าง ได้แก่ ถังข้อมูล (Database Slot) ฐานข้อมูลที่ใช้ทดสอบ (Master Database) ซอฟต์แวร์ที่ใช้ทดสอบ(Build) เครื่องที่ใช้ทดสอบ (Test Machine) สคริปท์ที่จะทดสอบ (Test Script) เป็นต้น ปัจจุบันการจัดเตรียมสภาพแวดล้อมของการทดสอบยังคงต้องให้ผู้ที่ทำการทดสอบเข้ามาจัดเตรียมด้วยตัวเอง รวมถึงการสั่งงานให้ Test Script ทำงานนั้น ผู้ทดสอบจะต้องเป็นผู้สั่งให้ Test Script ทำงาน ในแต่ละวันผู้ทดสอบจะต้องทดสอบซอฟต์แวร์มากกว่าหนึ่งเวอร์ชันและในแต่ละเวอร์ชันมีจำนวน Test Script ที่จะทดสอบจำนวนมาก ทำให้ในแต่ละวันผู้ทดสอบจะเสียเวลาไปกับการเตรียม Test Environment มากกว่าที่จะวิเคราะห์หาสาเหตุของปัญหา (Analysis Issue) ที่เกิดขึ้นจากข้อบกพร่อง (Defect) ของซอฟต์แวร์ที่พบ อีกทั้งถ้าผู้ทดสอบเตรียมสภาพแวดล้อมต่างๆ ช้า หรือผิดพลาด จะทำให้การทดสอบล่าช้า ไม่สามารถส่งมอบงานให้กับลูกค้าได้ทันเวลา

ด้วยปัญหาดังกล่าว ทำให้เกิดแนวคิดที่จะพัฒนาแอพพลิเคชันสำหรับเป็นเครื่องมือที่ช่วยในการตรวจสอบและเตรียม Test Environment ต่างๆที่ต้องทำแบบ Manual ให้เป็นแบบอัตโนมัติมากขึ้น เช่น จากเดิมผู้ใช้จะต้องเข้าไปที่เครื่องทดสอบแต่ละเครื่องและสั่งให้ Test Script ทำงานด้วยระบบที่จะพัฒนาขึ้นมาจะช่วยสั่งให้ Test Script ที่ได้ตั้งไว้สามารถทำงานได้อย่างอัตโนมัติ เพื่อช่วยลดเวลาในการจัดเตรียมสภาพแวดล้อมให้น้อยลง ทำให้ผู้ทดสอบมีเวลาศึกษาและวิเคราะห์ปัญหาที่เกี่ยวข้องทางด้านธุรกิจขององค์กรมากขึ้น นอกจากนี้ระบบจะยังสามารถตั้งเวลาการทดสอบได้ เพื่ออำนวยความสะดวกให้กับผู้ทดสอบในกรณีที่ไม่สามารถทดสอบในช่วงเวลานั้นได้

1.2 วัตถุประสงค์ของการพัฒนาระบบ

โครงการศึกษาและพัฒนาแอพพลิเคชันสำหรับการเตรียมสภาพแวดล้อมและการตั้งเวลาการทดสอบซอฟต์แวร์ มีวัตถุประสงค์ดังต่อไปนี้

1. เพื่อปรับปรุงขั้นตอนการเตรียมสภาพแวดล้อมของการทดสอบให้มีความถูกต้องสอดคล้องและลดความซ้ำซ้อน
2. เพื่อลดข้อผิดพลาดที่เกิดจากการเตรียมสภาพแวดล้อมของผู้ทดสอบ (Human error) ได้
3. เพื่อให้ผู้ทดสอบสามารถทดสอบและได้ผลการทดสอบที่รวดเร็วทันต่อการส่งมอบงานให้กับลูกค้า

1.3 ขอบเขตของการพัฒนาระบบ

แอพพลิเคชันที่พัฒนาขึ้นมานั้น เป็นแอพพลิเคชันใหม่ที่พัฒนาขึ้นมาเพื่อปรับปรุงกระบวนการทดสอบเชิงถดถอย (Regression Testing) ของทีมให้มีประสิทธิภาพมากยิ่งขึ้น โดยขอบเขตของการพัฒนาระบบ ดังต่อไปนี้

1. ระบบสามารถให้ผู้ทดสอบเข้ามาจัดการข้อมูลสภาพแวดล้อม (Environment data) ที่ระบบต้องนำไปใช้ในการทดสอบได้ ได้แก่
    - ระบบสามารถให้ผู้ทดสอบจัดการข้อมูล Database Slot โดยการเพิ่ม ลบ หรือแก้ไขข้อมูลได้
    - ระบบสามารถให้ผู้ทดสอบจัดการข้อมูล CFG โดยการเพิ่ม ลบ หรือหรือแก้ไขข้อมูลได้
    - ระบบสามารถให้ผู้ทดสอบจัดการเครื่องทดสอบ (Test Machine) โดยการเพิ่ม ลบ หรือแก้ไขข้อมูลได้
    - ระบบสามารถให้ผู้ทดสอบจัดการเครื่องเซิร์ฟเวอร์ที่ใช้ทดสอบ โดยการเพิ่ม ลบ หรือแก้ไขข้อมูลได้

- ระบบสามารถให้ผู้ทดสอบจัดการข้อมูลสคริปท์ทดสอบ (Test Suite) โดยการเพิ่ม ลบ หรือแก้ไขข้อมูลได้

2. ปรับรูปแบบการวางแผนการทดสอบจากเดิมที่มีการจัดเก็บเป็น File system ให้อยู่ในรูปของฐานข้อมูล เพื่อลดความซ้ำซ้อนและเพิ่มความถูกต้องของข้อมูล
3. ระบบสามารถให้ผู้ทดสอบทำการวางแผนการทดสอบได้
4. ระบบสามารถตรวจสอบความพร้อมของ Test Environment ต่างๆที่ต้องใช้ได้ เช่น Database Slot และ ขนาดของพื้นที่บนเซิร์ฟเวอร์ที่ใช้ทดสอบ เป็นต้น ถ้ามีตั้งเวลาทดสอบโดยที่บาง Test Environment ที่ยังไม่พร้อมใช้งาน ระบบจะต้องแจ้งเตือนไปยังผู้ทดสอบ
5. ระบบสามารถทำการตั้งเวลาทดสอบซอฟต์แวร์ล่วงหน้าได้ โดยมีรูปแบบการตั้งเวลาหลายรูปแบบ ดังนี้
   5.1 เมื่อถึงเวลาระบบจะเตรียม Test Environment และเรียก Test Suite ขึ้นมาทดสอบ เนื่องจาก ServerBuild และ ClientBuild ที่ต้องรอทาง Programmer ผลิตขึ้นมานั้น มีช่วงเวลาที่ไม่แน่นอน บางวันอาจจะได้ ServerBuild และ ClientBuild ช่วงกลางวันผู้ทดสอบก็จะสามารถเตรียม Test Environment ได้ทัน แต่บางวันอาจจะได้ ServerBuild และ ClientBuild ช่วงกลางคืนทำให้ไม่สามารถเตรียม Test Environment ได้ทัน
   5.2 เมื่อถึงเวลาระบบจะเรียก Test Suite ขึ้นมาทดสอบอย่างเดียว โดยไม่มีการเตรียม Test Environment ให้ เนื่องจากเครื่องเซิร์ฟเวอร์ที่ใช้ในการทดสอบจะมีการรีสตาร์ทในช่วงเวลา 20.00 น. ถึง 20.30 น. ของทุกวัน ผู้ทดสอบสามารถเตรียม Test Environment ได้เองและตั้งเวลาทดสอบหลังช่วงเวลารีสตาร์ทของเครื่องเซิร์ฟเวอร์
6. ระบบสามารถทำการ Trigger สั่งให้แต่ละเครื่องที่ได้ทำการวางแผนทดสอบไว้ทำการทดสอบได้แบบอัตโนมัติ
7. ระบบสามารถออกรายงานสรุปผลการทดสอบได้

1.4 ขั้นตอนของการศึกษา

เพื่อให้การพัฒนาแอพพลิเคชันสำหรับการเตรียมสภาพแวดล้อมและการตั้งเวลาทดสอบเป็นไปอย่างมีระบบและดำเนินงานภายใต้แผนที่วางไว้ ได้มีขั้นตอนการพัฒนาระบบดังนี้

1. ศึกษาทฤษฎีของการทดสอบซอฟต์แวร์ เพื่อเป็นแนวทางในการศึกษาและพัฒนาระบบงาน

2. ศึกษากระบวนการทดสอบเชิงถดถอย (Regression Testing) ขององค์กรให้เข้าใจกระบวนการทำงานมากขึ้น
3. ศึกษาเครื่องมือและซอฟต์แวร์ที่ใช้ในการพัฒนาระบบ
4. ออกแบบระบบงานโดยใช้ Unified Model Language
5. ออกแบบหน้าจอในส่วนที่ติดต่อกับผู้ใช้ด้วยโปรแกรม Visual Studio.net 2005
6. ใช้ระบบที่ได้จากการพัฒนาเพื่อสรุปผลโครงการ
7. จัดทำเอกสารสรุปการทำโครงการพัฒนา

1.5 ประโยชน์ที่คาดว่าจะได้รับ

จากการพัฒนาแอพพลิเคชันสำหรับการเตรียมสภาพแวดล้อมและการตั้งเวลาทดสอบ เมื่อนำไปใช้ในการดำเนินงานแล้ว ประโยชน์ที่คาดว่าจะได้รับคือ

1. ผู้ทดสอบสามารถจัดการข้อมูลสภาพแวดล้อม (Environment Data) ที่จำเป็นในการทดสอบได้รวดเร็ว รวมถึงสามารถจัดการข้อมูลสภาพแวดล้อมได้อย่างถูกต้อง
2. ผู้ทดสอบสามารถวางแผนการทดสอบในระบบได้ ทำให้ไม่เกิดความซ้ำซ้อนของข้อมูล
3. หัวหน้างานสามารถใช้ประโยชน์จากรายงานสรุปผลการทดสอบได้
4. ผู้ทดสอบสามารถทดสอบซอฟต์แวร์ได้ในปริมาณที่เพิ่มขึ้น เนื่องจากสามารถตั้งเวลาทดสอบได้ และสามารถส่งมอบงานให้กับลูกค้าได้ตรงตามเวลาที่กำหนด

บทที่ 2
ทฤษฎีและเทคโนโลยีที่เกี่ยวข้องในการพัฒนาระบบ

2.1 การทดสอบซอฟต์แวร์

2.1.1 กลยุทธ์การทดสอบซอฟต์แวร์ (Testing Strategies)

(พรฤดี เนติโสภากุล. 2549: 254) กลยุทธ์การทดสอบซอฟต์แวร์ (Testing Strategies) หมายถึงวิธีการออกแบบกรณีทดสอบและการวางแผนการทดสอบ เพื่อให้ได้ชุดของขั้นตอนที่ปฏิบัติตามได้ เป็นเครื่องยืนยันว่าการสร้างซอฟต์แวร์ประสบผลสำเร็จ กลยุทธ์ดังกล่าวคือแผนที่เส้นทางที่อธิบายถึง ขั้นตอนที่ทำระหว่างการทดสอบ กำหนดเวลาที่จะทำตามขั้นตอนเหล่านี้ กำหนดแรงงาน เวลาและทรัพยากรที่ต้องการใช้ ดังนั้นกลยุทธ์ใดๆต้องมีแผนการทดสอบ การออกแบบกรณีทดสอบ การลงมือทดสอบ การรวบรวมและประเมินผลข้อมูลผลลัพธ์

กลยุทธ์การทดสอบควรจะมีความยืดหยุ่นพอที่จะปรับแก้แนวทางการทดสอบได้ ขณะเดียวกันก็ควรจะคงรูปพอที่จะส่งเสริมการวางแผน และการจัดการติดตามเมื่อโครงการได้ดำเนินไป

กลยุทธ์การทดสอบซอฟต์แวร์ มีลักษณะทั่วไปดังนี้

1. การปฏิบัติการทดสอบให้ได้ผล ควรตรวจทานเอกสารทางเทคนิค เพื่อลดข้อผิดพลาดก่อนเริ่มทดสอบจริง
2. การทดสอบเริ่มที่องค์ประกอบย่อยก่อนแล้วเป็นภาพรวมของระบบทั้งหมด
3. เทคนิคที่แตกต่างกันเหมาะสมกับการทดสอบในจุดต่างๆกันตามกาลเวลา
4. การทดสอบ กระทำโดยผู้พัฒนาซอฟต์แวร์และนักทดสอบอิสระ
5. การทดสอบ (Test) และการตรวจสอบข้อผิดพลาด (Debug) เป็นกิจกรรมที่ต่างกัน แต่การ Debug เป็นสิ่งที่เกิดขึ้นควบคู่ไปกับการทดสอบ

2.1.2 กลยุทธ์การทดสอบสำหรับซอฟต์แวร์แบบดั้งเดิม (Testing Strategies for Conventional Software)

(พรฤดี เนติโสภากุล. 2549: 263-270) มีกลยุทธ์หลายอย่างที่สามารถนำใช้ทดสอบซอฟต์แวร์ได้ ตั้งแต่รอจนกว่าทีมซอฟต์แวร์จะสร้างระบบเสร็จแล้วค่อยทดสอบระบบทั้งหมด โดยหวังว่าจะพบข้อผิดพลาด วิธีการนี้แม้จะดึงดูดใจแต่มักไม่ได้ผล ผลลัพธ์ที่ได้คือซอฟต์แวร์มีข้อผิดพลาดจำนวนมาก ทำให้ลูกค้าและผู้ใช้งานผิดหวัง ส่วนอีกวิธีคือวิศวกรซอฟต์แวร์จะทำการทดสอบทุกๆวัน เมื่อมีการสร้างส่วนใหม่ๆของระบบออกมา แนวทางนี้จะให้ประสิทธิผลดีกว่า แต่นักพัฒนาซอฟต์แวร์ไม่นิยมที่จะทำ

กลยุทธ์การทดสอบที่ทีมพัฒนาซอฟต์แวร์ส่วนใหญ่เลือกใช้ จะเริ่มด้วยการทดสอบหน่วยย่อย (Unit Testing) ของโปรแกรมแล้วเคลื่อนมาทำการทดสอบที่ออกแบบสำหรับเชื่อมรวมกันของหน่วยย่อย (Integration Testing) จบท้ายด้วยการทดสอบที่ใช้งานระบบที่สร้างขึ้น ดังนี้

1. การทดสอบระดับหน่วยย่อย (Unit Testing)

การทดสอบระดับหน่วยย่อย เป็นการทดสอบหน่วยที่เล็กที่สุดของงานออกแบบซอฟต์แวร์ ในระดับองค์ประกอบซอฟต์แวร์ (Component) หรือโมดูลของซอฟต์แวร์ (Module) โดยใช้คำอธิบายประกอบการออกแบบระดับองค์ประกอบ เป็นแนวทางทดสอบเส้นทางควบคุมหลักเพื่อค้นหาข้อผิดพลาดภายในขอบเขตของโมดูล ความซับซ้อนสัมพัทธ์ของการทดสอบและข้อผิดพลาดที่การทดสอบค้นหาได้ จะถูกจำกัดด้วยขอบเขตของการทดสอบระดับหน่วย

2. การทดสอบระดับบูรณาการ (Integration Testing)

การทดสอบระดับบูรณาการ เป็นการทดสอบโปรแกรมที่เกี่ยวข้องกันตั้งแต่ 2 โปรแกรมขึ้นไป เพื่อทดสอบการเชื่อมโยงการทำงานระหว่างโปรแกรม โดยมีวัตถุประสงค์คือ นำองค์ประกอบที่ผ่านการทดสอบระดับหน่วยย่อยแล้ว มาต่อเป็นโครงสร้างโปรแกรมที่ออกแบบไว้ นอกจากนี้ยังช่วยเพิ่มความมั่นใจในความถูกต้องของลำดับการทำงานของโปรแกรม

3. การทดสอบเชิงถดถอย (Regression Testing)

ในทุกๆ ครั้งที่มีโมดูลใหม่ถูกเพิ่มเข้ามาเป็นส่วนหนึ่งของ Integration Testing นั่นแสดงถึงว่าซอฟต์แวร์ได้เปลี่ยนไปแล้ว เกิดเส้นทางไหลของข้อมูลใหม่ เกิดการติดต่อ I/O ใหม่ และอาจเกิดการควบคุมเชิงตรรกะใหม่ๆ เกิดขึ้น การเปลี่ยนแปลงเหล่านี้อาจก่อให้เกิดปัญหากับหน้าที่การทำงานที่เคยทำงานได้ดี การทดสอบเชิงถดถอย คือการทดสอบแบบซ้ำโดยใช้ชุดทดสอบเดิมที่เคยทำมาแล้ว เพื่อให้มั่นใจว่าการเปลี่ยนแปลงที่เกิดขึ้นจะไม่กระทบกับการทำงานของโปรแกรม

การทดสอบเชิงถดถอยอาจทำโดยมนุษย์ ด้วยการทดสอบซ้ำด้วยชุดกรณีทดสอบชุดเดิม หรืออาจทำแบบอัตโนมัติด้วยใช้เครื่องมือบันทึกและเล่นซ้ำ (Capture/playback tools) เครื่องมือนี้จะช่วยให้ผู้ทดสอบสามารถดักจับกรณีทดสอบ และเปรียบเทียบผลของการทดสอบ เพื่อใช้ทดสอบซ้ำในกาทดสอบครั้งถัดๆไป การทดสอบเชิงถดถอย ประกอบด้วยกรณีทดสอบที่แตกต่างกัน 3 อย่าง ได้แก่

- ตัวแทนของตัวทดสอบที่จะทำงานกับทุกหน้าที่ของซอฟต์แวร์
- เน้นการทดสอบหน้าที่ของซอฟต์แวร์ที่ได้เพิ่มขึ้นมาที่อาจได้รับผลกระทบจากการเปลี่ยนแปลง
- ตัวทดสอบจะมุ่งเน้นไปที่องค์ประกอบของโปรแกรมที่เพิ่งทำการเปลี่ยนแปลงไป

เมื่อการทำ Integration Testing ได้ดำเนินไป จำนวนของการทดสอบเชิงถดถอยอาจจะโตขึ้นอย่างมาก ดังนั้นชุดของการทดสอบเชิงถดถอยควรได้รับการออกแบบให้รวมถึงการทำงานหลักๆของโปรแกรม

4. การทดสอบแบบสโมค (Smoke Testing)

เป็นวิธีการทดสอบระดับบูรณาการแบบหนึ่ง ที่นิยมทั่วไประหว่างการพัฒนาซอฟต์แวร์ เป็นกลไกที่ใช้ในโครงการที่มีข้อจำกัดด้านเวลาสูง เพื่อให้ทีมงานประเมินโครงการได้บ่อยๆ การทดสอบแบบสโมค มีกิจกรรมที่สำคัญคือ

1. องค์ประกอบ (Component) ซอฟต์แวร์ที่ถูกแปลงเป็นโค้ดแล้ว จะถูกรวมกันเข้าเป็น Build ซึ่ง Build จะประกอบไปด้วยไฟล์ข้อมูล ไลบลารี่ โมดูลที่ใช้ซ้ำ และองค์ประกอบอื่นๆ ที่จำเป็นในการทำหน้าที่หนึ่งๆ ของซอฟต์แวร์
2. ออกแบบชุดของการทดสอบที่จะทำให้เห็นความผิดพลาดที่อาจทำให้ Build ทำงานไม่ปกติ โดยมุ่งเน้นในการเปิดเผยความผิดพลาดที่ร้ายแรง ที่มีโอกาสทำให้โครงการล่าช้าได้สูง
3. รวม Build เข้ากับ Build อื่นๆ และเข้ากับผลิตภัณฑ์เท่าที่สำเร็จในปัจจุบัน เพื่อทำการทดสอบแบบสโมคเป็นประจำวัน

ประโยชน์ของการทดสอบแบบสโมค

1. ลดความเสี่ยงเชิงบูรณาการ เพราะการทดสอบประจำวันจะพบข้อผิดพลาดเร็ว จึงลดความเสี่ยงของการล่าช้า
2. ผลิตภัณฑ์สุดท้ายมีคุณภาพดีขึ้น
3. ช่วยให้หาสาเหตุและการแก้ไขข้อผิดพลาดทำได้ง่าย ข้อผิดพลาดที่พบจะสัมพันธ์กับส่วนซอฟต์แวร์ใหม่ที่เพิ่มมา
4. การประเมินความก้าวหน้าทำได้ง่าย เพราะแต่ละวันซอฟต์แวร์จะถูกนำมารวมกันได้มากขึ้น และเป็นตัวบ่งชี้ว่างานมีความก้าวหน้า

2.1.3 การทดสอบซอฟต์แวร์แบบทั่วไป (Traditional Software Testing)

(Poston. 1994a: 124-129) ในยุคแรกของการพัฒนาซอฟต์แวร์จะมีรูปแบบของการทดสอบที่เป็นการทดสอบด้วยมือ นั่นคือการที่ผู้ทดสอบทำการป้อนข้อมูล 2-3 ค่าให้กับคอมพิวเตอร์แล้วดูการตอบสนองของซอฟต์แวร์ที่จะส่งผลลัพธ์กลับมา ซึ่งแบ่งการทำงานออกเป็น 3 ขั้นตอน คือ

- *Setup* เป็นการตั้งสถานะให้อยู่ในสถานะพร้อมที่จะทำการทดสอบ โดยมีการให้ค่าข้อมูลเข้า (Input) ที่จำเป็นในการทดสอบนั้นๆ แก่ซอฟต์แวร์

- *Execute* เป็นการเรียกใช้การทำงานที่ต้องการ เพื่อให้ได้ซึ่งผลลัพธ์ที่จะนำไปพิจารณาต่อไป ว่าตรงตามที่ต้องการหรือไม่ภายหลัง
- *Cleanup* เป็นการยืนยันผลลัพธ์ที่ได้และเปลี่ยนสถานะของซอฟต์แวร์จากสถานะที่ทำการทดสอบเป็นสถานะปกติ นั่นคือหากมีการเปลี่ยนแปลงระบบใดๆ จากขั้นตอน Setup ก็ต้องทำการเปลี่ยนกลับให้อยู่ในสถานะปกติก่อนการทดสอบ

1. กรณีในการทดสอบ (Poston. 1994b: 48-58)

ปัจจุบันการทดสอบจะมีการสร้างกรณีในการทดสอบ (Test case) ซึ่งเป็นชุดข้อมูลเข้า (Input) ที่ใช้ในการทดสอบขึ้นมาก่อน จากนั้นจึงทำการประมวลผล Test case และการประเมินผลของการทดสอบ สำหรับการสร้าง Test case อาจออกแบบได้ด้วย 6 เทคนิค ได้แก่

การทดสอบการทำงาน (Functional testing) เป็นการทดสอบหน้าที่การทำงานหลักๆ ซึ่งจะต้องมีการทดสอบอย่างน้อยหนึ่งครั้ง นั่นคือจะมีอย่างน้อยหนึ่งTest case ที่มี ชื่อ ค่าข้อมูลเข้าหรืออินพุต และค่าที่คาดหวัง (Expected value)

การวิเคราะห์ค่าขอบเขต (Boundary value Analysis) เป็นการหาข้อผิดพลาดในเรื่องของขอบเขตของค่าต่างๆ หรือตรวจหาในช่วงของข้อมูล 5 ค่า คือ ค่าขอบเขตต่ำสุด ค่าที่มีค่ามากกว่าค่าขอบเขตต่ำสุดอยู่ 1 ค่า ค่าปกติ ค่าที่มีค่าน้อยกว่าค่าขอบเขตสูงสุดอยู่ 1 ค่า และค่าขอบเขตสูงสุด

การวิเคราะห์ชั้นสมมูล (Equivalence Class) เป็นการวิเคราะห์ตรวจสอบค่าที่เป็นไปได้ที่ถูกแบ่งแยกออกเป็นกลุ่มเล็กๆ ที่เรียกว่า Equivalence Classes ซึ่งเราจะสร้าง Test case ที่มีข้อมูลเข้าอย่างน้อยหนึ่งค่าสำหรับแต่ละ sub domain และ Equivalence Class ที่ระบุ

Cause-effect Graphing เป็นการทดสอบเชิงตรรกะหรือเงื่อนไข ซึ่งทุกๆ เงื่อนไขจะถูกตรวจสอบค่าความจริงที่เป็นจริง หรือค่าความจริงที่เป็นเท็จ กล่าวได้ว่าเป็นการทดสอบ Boolean Logic ซึ่ง Test case จะถูกสร้างสำหรับตรวจสอบทุกๆ เงื่อนไข ในทุกๆการกระทำในค่าที่เป็นจริงและค่าที่เป็นเท็จ

การทดสอบเหตุการณ์ (Event-directed Testing) เป็นการสร้าง test case ที่ทำทุกๆ เหตุการณ์อย่างน้อยหนึ่งครั้ง โดยเหตุการณ์หมายถึงสิ่งที่เกิดขึ้นจากภายนอกระบบแล้วมีผลทำให้เกิดการกระทำขึ้นภายในระบบโปรแกรม นั่นคือ ต้องหาเหตุการณ์หรือสิ่งภายนอกระบบที่สามารถก่อให้ระบบเกิดการทำงานหรือการกระทำขึ้น

การทดสอบสถานะ (State-directed Testing) คือ Test case ที่ทำให้มีการเปลี่ยนสถานะและจะต้องมีการสร้าง Test case อย่างน้อย 1 ชุดต่อ 1 สถานะในการทดสอบ ซึ่งสถานะคือกลุ่มของข้อมูลหรือคุณลักษณะ หรือความสัมพันธ์ระหว่างค่านั้นๆ และการเปลี่ยนสถานะคือการเปลี่ยนค่าสิ่งเหล่านั้น

2. ประเภทของการทดสอบ (Patton. 2000)

สำหรับการทดสอบซอฟต์แวร์สามารถแบ่งประเภทของการทดสอบออกได้ 2 ประเภทจากวิธีการที่ใช้ในการทดสอบ คือ การทดสอบแบบกล่องดำ (Black-Box) และการทดสอบแบบกล่องขาว (White-Box) ทั้งสองประเภทต่างกันที่ Block-box ผู้ทดสอบจะรู้เพียงสิ่งที่ซอฟต์แวร์ควรจะทำโดยไม่รู้ว่าทำได้ด้วยวิธีใด นั่นคือถ้าพิมพ์อินพุตบางค่าให้ก็ได้จะได้รับเอาท์พุตนั้นกลับมาโดยไม่รู้ว่ามันเกิดขึ้นได้อย่างไร แต่สำหรับ white-box ผู้ทดสอบจะต้องเข้าไปดูถึงโค้ดของโปรแกรมและใช้เป็นแนวทางในการทดสอบ ซึ่งขึ้นอยู่กับว่าได้อะไรจากโค้ดมาบ้าง ก็จะนำมาเป็นข้อมูลสำหรับการทดสอบต่อไป

ในอีกมุมมองหนึ่งอาจแบ่งได้เป็นการทดสอบแบบ Static และการทดสอบแบบ Dynamic ซึ่งการทดสอบแบบ Static จะใช้ในการทดสอบอะไรก็ตามที่ไม่สมารถสั่งให้ประมวลผลได้ ส่วนมากจะเป็นการตรวจสอบและวิจารณ์ เช่นการตรวจสอบเกี่ยวกับเอกสารข้อกำหนดเบื้องต้น (Specification) ของโปรแกรม เป็นต้น ส่วนการทดสอบแบบ Dynamic จะเป็นการทดสอบสิ่งที่สามารถเรียกประมวลผลได้หรือต้องมีการใช้ซอฟต์แวร์ในการทดสอบ จากทั้งสองมุมมองหากนำมาใช้ร่วมกันจะเกิดประเภทของการทดสอบ 4 ประเภทดังนี้

- ***การทดสอบแบบ Static Black-Box*** มักใช้ในการตรวจสอบเอกสารระบบ (Specification) ซึ่งเป็นเอกสารที่ไม่สามารถประมวลผลได้ ซึ่งไม่จำเป็นต้องรู้ที่มาของข้อมูลในเอกสารหรือเหตุผลของการสร้างเอกสาร
- ***การทดสอบแบบ Dynamic Black-Box*** มักใช้ในการทดสอบซอฟต์แวร์โดยที่ไม่รู้ถึงรายละเอียดของโค้ดภายในและใช้การทดสอบด้วยการเรียกประมวลผลเหมือนกับการใช้โปรแกรมตามปกติ
- ***การทดสอบแบบ Static White-Box*** มักใช้ในการตรวจสอบการออกแบบและโค้ดของโปรแกรม โดยไม่ต้องทำการเรียกประมวลผล แต่จะเป็นการวิเคราะห์โค้ดของโปรแกรมที่ต้องการทดสอบแทน
- ***การทดสอบแบบ Dynamic White-Box*** เป็นการทดสอบโปรแกรมที่ต้องรู้รายละเอียดของโค้ด เพื่อช่วยให้ตรวจสอบได้ถึงการทำงานภายใน ซึ่งยังคงมีการเรียกประมวลผลซอฟต์แวร์ตามเงื่อนไขของโค้ดที่มีโอกาสก่อให้เกิดข้อผิดพลาดต่างๆ

2.1.4 การทดสอบซอฟต์แวร์ด้วยบุคคลปฏิบัติ (Manual Software Testing)

เป็นการทดสอบซอฟต์แวร์โดยผู้ทดสอบกระทำเองทั้งหมดตามกระบวนการของการทดสอบซอฟต์แวร์ทั่วไป ดังที่ได้กล่าวไว้ข้างต้นทุกครั้งที่ทำการทดสอบ

2.1.5 การทดสอบซอฟต์แวร์แบบอัตโนมัติ (Automated Software Testing)

(Poston. 1994b: 124-129) การทดสอบแบบทั่วไปจะมีความยุ่งยากสำหรับผู้ทดสอบที่ต้องทดสอบซอฟต์แวร์หลายๆ ครั้งหรือเมื่อมีการปรับปรุงซอฟต์แวร์ให้ดีขึ้นก็จะต้องมีการทดสอบอีกครั้ง เพื่อให้มั่นใจว่าซอฟต์แวร์ดังกล่าวได้แก้ปัญหาที่พบในครั้งก่อนแล้ว การทดสอบเช่นนี้เรียกว่าการทดสอบเชิงถดถอย (Regression Testing) ในการทดสอบซ้ำกันในกรณีเดิมบ่อยครั้งก็เป็นเรื่องที่น่าเบื่อหน่ายและเสียเวลาหากทำด้วยวิธีทั่วไป

แนวทางหนึ่งในการแก้ปัญหาก็คือการพัฒนาการทดสอบแบบอัตโนมัติ ซึ่งมีการใช้เครื่องมือช่วยในการทดสอบที่เรียกว่า Test Execution Tool สำหรับหลักการทำงานโดยทั่วไปของการทดสอบแบบอัตโนมัติจะคล้ายกับเครื่องเล่นวิดีโอเทป นั่นคือมีการบันทึกและนำสิ่งที่บันทึกไว้มาเล่นซ้ำได้

การบันทึกของการทดสอบแบบอัตโนมัติ คือการบันทึกค่าต่างๆ ในการทำการทดสอบทั่วไป ทั้ง Test case หรือค่าสถานะต่างๆ ที่ต้องตั้งค่าใหม่ทุกครั้งเพื่อใช้ในการทดสอบ ซึ่งผู้ทำการทดสอบจะต้องทำสิ่งเหล่านี้ในขั้นตอน Setup ในครั้งแรกเท่านั้น ซึ่งหลังจากเริ่มบันทึกค่าดังกล่าวแล้ว ผู้ทดสอบก็จะสามารถทำขั้นตอน Execute และ Cleanup ไปจนจบ แล้วจึงเลิกการบันทึก ซึ่งข้อมูลต่างๆที่ถูกบันทึกไว้หรือ Test script ก็จะถูกเก็บไว้เพื่อนำกลับมาใช้ใหม่ในการทดสอบครั้งต่อไป หรือที่เรียกว่าการเล่นซ้ำ กล่าวคือ Test script จะถูกนำมาใช้ในการทดสอบแบบ Regression เพื่อประหยัดเวลาในการต้องตั้งค่าในขึ้นตอน Setup นั่นเอง

ประโยชน์ของการทดสอบแบบอัตโนมัติ ได้แก่

- ความสะดวกรวดเร็ว
- ประสิทธิภาพในการทำงานของผู้ทดสอบเพิ่มมากขึ้น
- ถูกต้องแม่นยำในการทดสอบแบบซ้ำๆกัน
- เครื่องมือสำหรับทดสอบแบบอัตโนมัติ สามารถทำงานได้อย่างต่อเนื่อง

2.1.6 เครื่องมือในการทดสอบซอฟต์แวร์ (Test Tools) (Patton. 2000)

เครื่องมือสำหรับช่วยในการทดสอบนั้นมีอยู่ด้วยกันมากมายหลายประเภทให้เลือกใช้ ซึ่งในการเลือกใช้ก็ต้องขึ้นอยู่กับประเภทของซอฟต์แวร์ที่จะทดสอบและวิธีการที่ใช้ในการทดสอบว่าเป็นแบบ Black-Box หรือ White-Box ในที่นี้จะกล่าวถึงประเภทของเครื่องมือในหมวดใหญ่ๆและวิธีการใช้งาน ดังนี้

1. Viewers and Monitors เป็นเครื่องมือที่ทำให้ผู้ทดสอบเห็นรายละเอียดการทำงานของซอฟต์แวร์ที่กำลังทดสอบ ซึ่งปกติจะไม่สามารถมองเห็น เช่น สามารถเห็นว่าโค้ดบรรทัดไหนที่กำลังประมวลผลอยู่ หรือฟังก์ชันใดกำลังทำงานอยู่ หรือรู้ว่ากำลังทดสอบไปตามเส้นทางการ

ทำงานของโค้ดเส้นทางไหน หรือดูข้อมูลที่ถูกส่งผ่านเครือข่าย เป็นต้น ตัวอย่างของเครื่องมือประเภทนี้ได้แก่ Code coverage analyzer และ Communications analyzer

2. Drivers เป็นเครื่องมือที่ใช้ในการควบคุมและเรียกซอฟต์แวร์ขึ้นมาทดสอบ ตัวอย่างเช่น Batch file ที่เป็นรายการของโปรแกรมหรือคำสั่งง่ายๆ ที่จะถูกเรียกประมวลผลเป็นลำดับ ซึ่งจะเรียก Batch file นี้ขึ้นทดสอบซอฟต์แวร์โดยไม่จำเป็นต้องเฝ้าดูตลอดเวลา ในเวลาต่อมา Batch file อาจถูกแทนที่ด้วยการใช้เอกสารสคริปท์มาแทนในส่วนที่มีความซับซ้อนมากขึ้น ตัวอย่างของเครื่องมือประเภทนี้อีกตัวอย่างคือการนำคอมพิวเตอร์อีกเครื่องหนึ่งที่มีโปรแกรมขนาดเล็กทำการสร้างข้อมูลเข้า (Input) สำหรับซอฟต์แวร์ที่ต้องทดสอบด้วยข้อมูลจำนวนมาก คอมพิวเตอร์ดังกล่าวจะทำงานเป็น driver ซึ่งสามารถสร้างข้อมูลได้เอง

3. Stubs เป็นเครื่องมือคล้าย Driver โดยแตกต่างกันที่ Stubs จะไม่ได้ไปควบคุมหรือเรียกซอฟต์แวร์ขึ้นมาทดสอบ แต่จะใช้การรับหรือตอบสนองต่อข้อมูลที่ซอฟต์แวร์ส่งมาแทน เช่นการทดสอลบการพิมพ์ของเครื่องพิมพ์ โดยต้องพิมพ์ข้อมูลจำนวนมาก แทนที่จะใช้การพิมพ์ออกที่เครื่องพิมพ์เพื่อดูผลการพิมพ์เหมือนการใช้งานจริงซึ่งจะเสียเวลาและไม่มีประสิทธิภาพ เราอาจใช้เครื่องคอมพิวเตอร์แทนการพิมพ์ โดยเขียนโปรแกรมพิเศษขึ้นมาที่สามารถทำการอ่านและประมวลผลข้อมูลที่ส่งมาให้เครื่องพิมพ์ ก็จะสามารถทำการทดสอบการพิมพ์ได้รวดเร็วและถูกต้องมากขึ้น

4. เครื่องมือ Stress และ Load เป็นเครื่องมือที่ใช้ในการสร้างสถานการณ์ที่ตึงเครียดและการทำงานที่หนักสำหรับซอฟต์แวร์ที่กำลังทดสอบ เพื่อใช้ทดสอบในสภาวะที่ตึงเครียดซึ่งอาจทำให้เกิดการทำงานที่ผิดพลาดได้ ตัวอย่างของเครื่องมือประเภทนี้ได้แก่ LoadRunner ซึ่งเป็นครื่องมือที่ใช้ในการทดสอบประสิทธิภาพทำงานของซอฟต์แวร์(Performance Testing) สนับสนุนแอพพลิเคชันที่เป็นแบบ Windows-Based และ Web-Based ซึ่งเป็นของบริษัท Hewlett-Packard

5. Functional Tool เป็นเครื่องมือที่ใช้ในการทดสอบฟังก์ชันการทำงานของซอฟต์แวร์ผ่านส่วนที่ติดต่อกับผู้ใช้ (User Interface) ตัวอย่างของเครื่องมือประเภทนี้ได้แก่ WinRunner เป็นซอฟต์แวร์ที่ใช้ทดสอบแอพพลิเคชันแบบ Windows-Based เท่านั้น และQuick Test Professional (QTP) เป็นซอฟต์แวร์ที่ใช้ทดสอบหน้าที่การทำงานของแอพพลิเคชันได้ทั้งแบบ Windows-Based และ Web-Based ทั้งสองเครื่องมือนี้เป็นของบริษัท Hewlett-Packard เป็นต้น โดยในโครงการนี้ได้นำเครื่องมือ WinRunner มาช่วยบันทึกและสร้างเป็น Test Script

6. Interference Injectors และ Noise Generators เป็นเครื่องมือที่คล้ายกับเครื่องมือ Stress และ Load แต่มีการสามารถสร้างสถานการณ์แบบสุ่มมากกว่า ซึ่งมักใช้ในการทดสอบที่มีการเปลี่ยนสถานการณ์อย่างฉับพลัน เช่นมีการเพิ่มหรือลดจำนวนหน่วยความจำสำหรับการใช้ซอฟต์แวร์ด้วยจำนวนที่ต่างกันอย่างมาก ในการใช้เครื่องมือประเภทนี้อาจมีการใช้ Viewer มาประยุกต์ในการเฝ้ามองข้อมูลและทำการเปลี่ยนข้อจำกัดต่างๆของระบบด้วยตัวเอง

7. เครื่องมือสำหรับการวิเคราะห์ เป็นเครื่องมือทั่วไปที่ผู้ทดสอบใช้ร่วมเพื่อให้การทดสอบง่ายขึ้น ตัวอย่างเช่น โปรแกรมเอกสาร (Word Processing Software) โปรแกรมสเปรดชีท (Spreadsheet Software) โปรแกรมฐานข้อมูล (Database Software) โปรแกรมเทียบไฟล์ (File Comparison Software) โปรแกรมจับภาพหน้าจอและเปรียบเทียบหน้าจอ (Screen Capture and Comparison Software) โปรแกรมตรวจสอบข้อผิดพลาด (Debugger) เป็นต้น

2.2 หลักการทำงานของสคริปท์อัตโนมัติภายในทีม

องค์กรที่พัฒนาซอฟต์แวร์ขึ้นมาเพื่อจำหน่ายหรือเป็นองค์กรที่รับจ้างในการผลิตซอฟต์แวร์นั้นควรจะมีทีมเฉพาะสำหรับทดสอบซอฟต์แวร์ เพื่อตรวจสอบคุณภาพของซอฟต์แวร์ให้เป็นไปตามความต้องการของผู้ใช้ ซึ่งการทดสอบจะขึ้นอยู่กับประเภทซอฟต์แวร์ของแต่ละองค์กร สำหรับทีม Automated Regression Team ภายในองค์กร DST World Wide Service นั้นมีหน้าที่ทำการทดสอบเชิงถดถอย (Regression Testing) ซึ่งจะต้องมีการเตรียมสภาพแวดล้อม การวางแผนการทดสอบ การเตรียมคอนฟิกุเรชันไฟล์ (Configuration file) ต่างๆ ที่จำเป็นให้พร้อมก่อนที่จะเริ่มการทดสอบ ดังรูปที่ 2.1

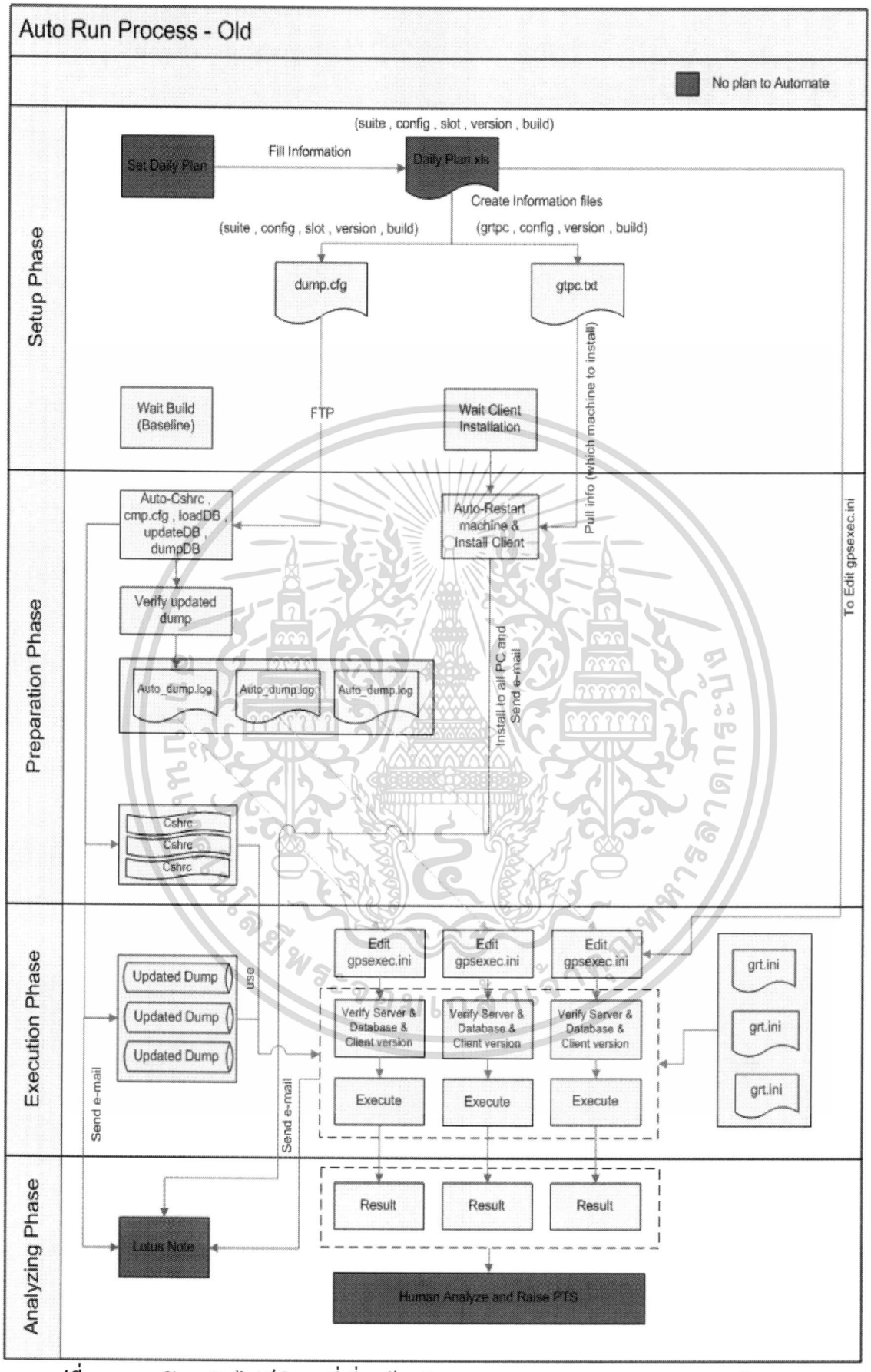
จากรูปสามารถอธิบายรายละเอียดได้ดังนี้ ในขั้นตอนของการทดสอบ Regression test ภายในทีมนั้นแบ่งการทำงานออกเป็น 4 ส่วนคือ

1. Setup Phase

เป็นส่วนของการวางแผนการทดสอบและเตรียมความพร้อมของไฟล์ต่างๆ โดยจะเริ่มตั้งแต่การวางแผนการทดสอบว่าต้องการทดสอบ Test Suite อะไรบ้าง โดยทำการวางแผนและระบุค่าต่างๆที่จำเป็นในเอกสาร DailyPlan.xls ซึ่งมีสิ่งที่ต้องระบุคือ Test Machine, Test Suite, CFG, Database Slot, ClientBuild version และ ClientBuild date จากนั้นทำการสร้างไฟล์อีก 2 ไฟล์ ได้แก่ Dump.cfg เพื่อใช้ในการโหลดข้อมูลเข้าสู่ Database Slot และ grtpc.txt เพื่อใช้ในการติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ เมื่อทำการวางแผนและเตรียมไฟล์ต่างๆพร้อมแล้วจจะต้องทำการรอเวอร์ชันและวันที่ของ ServerBuild และ ClientBuild ที่ได้ระบุไว้ใน DailyPlan การทดสอบจะเริ่มไม่ได้ถ้ายังไม่มีเวอร์ชันและวันที่ของ ServerBuild และ ClientBuildที่ต้องการ

2. Preparation Phase

เป็นส่วนของการเตรียมสภาพแวดล้อมก่อนการทดสอบ ขั้นตอนนี้จะเริ่มขึ้นได้ก็ต่อเมื่อมีเวอร์ชันและวันที่ของ ServerBuild และ ClientBuild ที่ต้องการถูกผลิตออกมาก การทำงานจะแยกเป็น 2 ส่วน คือ ส่วนแรกเป็นส่วนของ ClientServer จะถูกนำไปติดตั้งที่เครื่องทดสอบ หลังจากติดตั้งเรียบร้อยแล้วจะทำการส่งอีเมลแจ้งไปยังผู้ทดสอบ และอีกส่วนเป็นส่วนของฝั่งเซิร์ฟเวอร์ที่จะอ่านข้อมูลจาก Dump.cfg เพื่อไปทำการสร้างไฟล์ Cshrc ไว้สำหรับติดต่อระหว่างเซิร์ฟเวอร์กับกับ CFG จากนั้นจะทำการโหลดและอัพเดตข้อมูลเข้าสู่ Database Slot เพื่อใช้ในการทดสอบ



รูปที่ 2.1 คอนฟิกูเรชันไฟล์ต่างๆ ที่เกี่ยวข้องกับการทดสอบ Regression Test ของทีม ART

3. Execution Phase

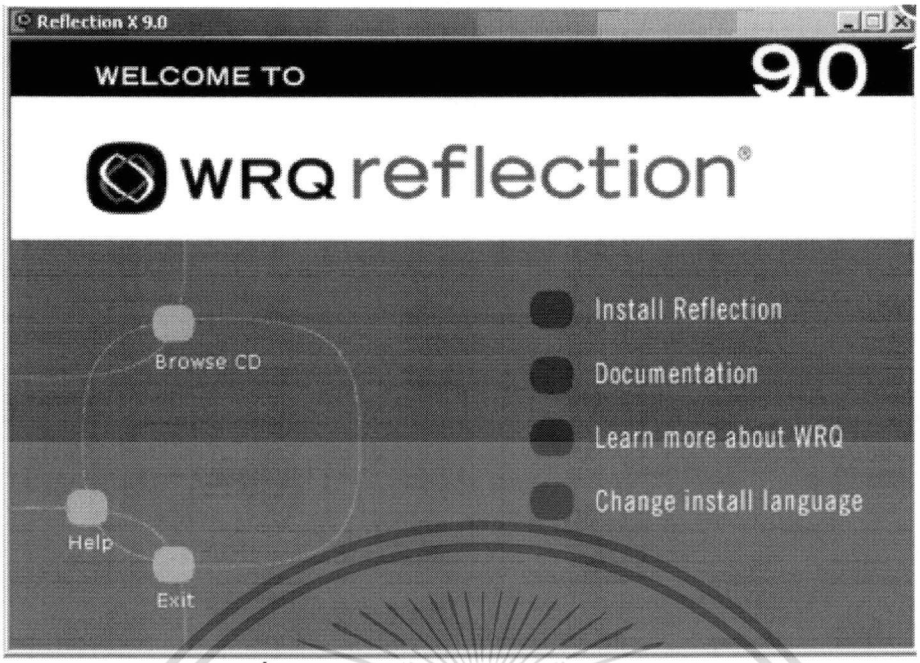
เป็นส่วนของการทดสอบ Regression test ในขั้นตอนของการทดสอบจะใช้ Test Script ที่ถูกสร้างขึ้นด้วย WinRunner tool โดยเริ่มแรกของการทดสอบจะมีการเซตค่าในซอฟต์แวร์ที่จะทดสอบเพื่อให้สามาถเปิดแอพพลิเคชันขึ้นมาทำการทดสอบได้ ซึ่งการเซตค่าจะไปทำในไฟล์ gpsexec.ini โดยไฟล์นี้จะได้มาในขั้นตอนของการติดตั้ง ClientBuild หลังจากทำการเซตค่าเรียบร้อย Test Script จะทำการตรวจสอบเวอร์ชันและวันที่ของ ServerBuild และ ClientBuild ว่าตรงกันหรือไม่ ถ้าไม่ตรงกันจะหยุดทำการทดสอบ แต่ถ้าตรงกันจะเริ่มทำการทดสอบตาม Test Script ต่อไป

4. Analyzing Phase

เป็นส่วนของการวิเคราะห์ผลการทดสอบ ซึ่งหลังจากที่การทดสอบสิ้นสุดลง ไม่ว่าผลการทดสอบจะผ่านหรือไม่ผ่าน จะทำการส่งผลแจ้งไปทางอีเมลเพื่อให้ผู้ทดสอบทำการวิเคราะห์หาสาเหตุ หากเป็นข้อบกพร่อง (Defect) จะทำการแจ้งไปยังผู้พัฒนาระบบ (Developer) เพื่อทำการแก้ไขต่อไป

2.3 WRQ Reflection

ซอฟต์แวร์ Reflection ของ WRQ เป็นโซลูชันที่ใช้ในการจำลองเทอร์มินัล เซิร์ฟเวอร์ PC และโซลูชันด้านความปลอดภัย SSH ส่วนเซิร์ฟเวอร์รวม Verastream ของ WRQ เป็นแพลตฟอร์มเดียวสำหรับการใช้ legacy logic และข้อมูลในเว็บแอพพลิเคชันและซีอาร์เอ็มแอพพลิเคชัน ซึ่ง Reflection มีขีดความสามารถด้านความปลอดภัยและการจัดการ โดยจะสนับสนุนและรับประกันความปลอดภัยสำหรับระบบ host ที่หลากหลายรวมถึง ระบบ IBM zSeries (mainframe), IBM iSeries (AS/400), HP e3000, HP 9000, OpenVMS, Unisys, UNIX, Tandem และ Windows server ตัวอย่างแสดงดังรูปที่ 2.2 โดยในส่วนของโครงการนี้ได้นำซอฟต์แวร์ Reflection มาช่วยในเรื่องของการเข้าถึงเครื่องคอมพิวเตอร์แม่ข่าย (host access) เพื่อใช้ในการโหลดและอัพเดตข้อมูลเข้าสู่ Database Slot

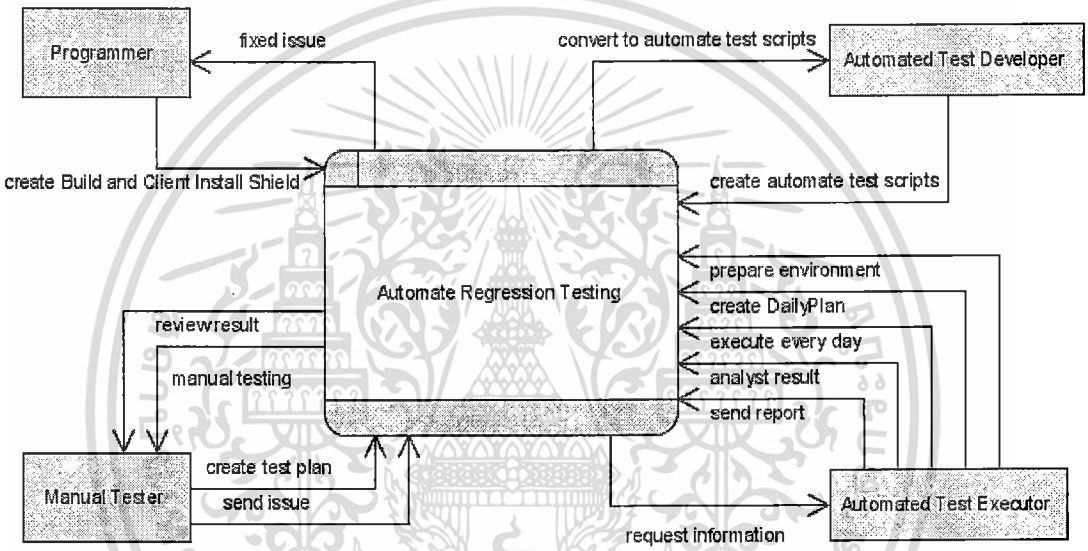


รูปที่ 2.2 ภาพตัวอย่างซอฟต์แวร์ Reflection

บทที่ 3
การศึกษาและวิเคราะห์ระบบ

3.1 การทำงานของระบบปัจจุบัน

จากการศึกษาระบบการทำงานเดิมของทีม Automated Regression Team (ART) พบว่าในการทดสอบซอฟต์แวร์ของทีมนั้นจะเกี่ยวข้องกับฝ่ายต่างๆ 4 ฝ่าย แสดงด้วยแผนผังแสดงภาพรวมของการทดสอบ Regression ดังรูปที่ 3.1



รูปที่ 3.1 แผนผังแสดงภาพรวมของการทดสอบ Regression ในมุมมองการไหลของข้อมูล

จากภาพสามารถอธิบายรายละเอียดได้ดังนี้

1. **Programmer** คือ ผู้ที่มีหน้าที่สร้าง แก้ไข และปรับปรุงซอฟต์แวร์ จากนั้นจะผลิตออกมาเป็น ServerBuild และ ClientBuild เพื่อส่งให้กับทีมอื่นๆนำไปใช้หรือทดสอบ
2. **Manual Tester** คือ ผู้ที่มีหน้าที่ทดสอบซอฟต์แวร์ด้วยวิธีแบบ Manual
3. **Automated Test Developer** คือ ผู้ที่มีหน้าที่ใช้เครื่องมือการทดสอบ (Testing tools) แปลง Test plan ที่ทาง Manual Tester ส่งมาให้เป็นสคริปท์แบบอัตโนมัติ (Automated Test Scripts)
4. **Automated Test Executor** คือ ผู้ที่มีหน้าที่ในการทำสคริปท์แบบอัตโนมัติ (Automated Test Scripts) ไปทำการทดสอบ แล้วจัดทำรายงานส่งผลลัพธ์ของการทดสอบไปยัง Manual Tester

3.1.2 ขั้นตอนการทำงาน

ลักษณะการทำงานภายในทีม ART สามารถแบ่งการทำงานออกเป็น 3 ขั้นตอน ดังนี้

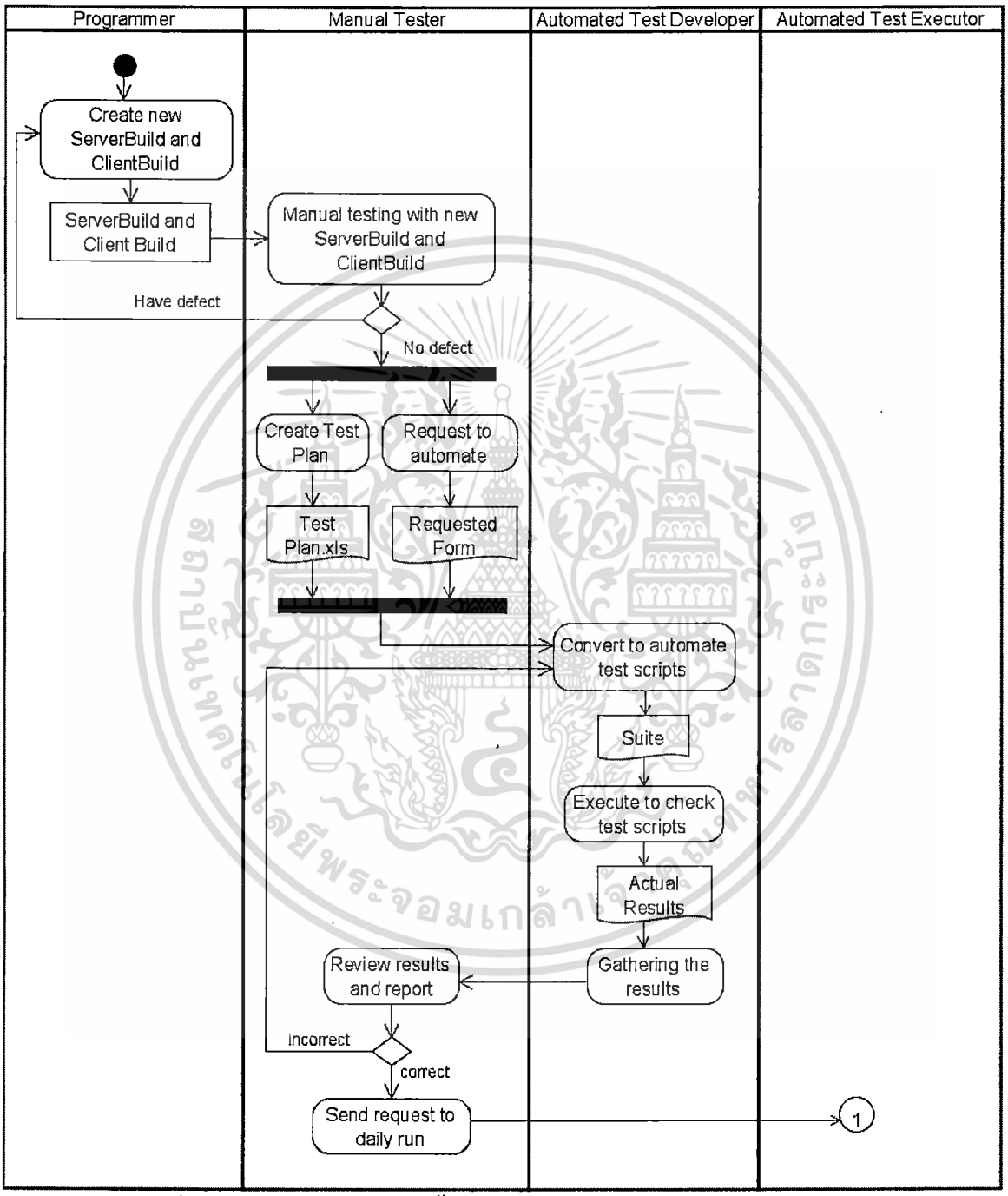
1. ขั้นตอนการร้องขอและการสร้างสคริปท์อัตโนมัติ (Request and develop automated Test Script)
2. ขั้นตอนการเตรียมสภาพแวดล้อมก่อนทำการทดสอบ (Preparation and setup environment)
3. ขั้นตอนการทดสอบและการวิเคราะห์ผลการทดสอบ (Execution and analysis result)

1. ขั้นตอนการร้องขอและการสร้างสคริปท์อัตโนมัติ (Request and develop automate test script)

ขั้นตอนการร้องขอและการสร้างสคริปท์อัตโนมัติ สามารถแสดงด้วยแผนภาพ ดังรูปที่ 3.2 จากรูปสามารถอธิบายการทำงานได้ดังนี้ ขั้นตอนการร้องขอและการสร้างสคริปท์อัตโนมัติจะเริ่มจากการที่ Programmer ทำการพัฒนาหรือปรับปรุงซอฟต์แวร์ตามที่ลูกค้าต้องการ เมื่อทำการปรับปรุงเรียบแล้วจะทำการคอมไพล์ (Compile) โปรแกรมและผลิต (Build) ออกมาได้เป็น ServerBuild สำหรับติดตั้งที่เครื่องแม่ข่าย (Server) และ ClientBuild สำหรับติดตั้งที่เครื่องทดสอบ (Test Machine) โดยทาง Programmer จะทำการผลิตออกมาทุกวัน วันละหลายเวอร์ชัน และเวลาที่จะได้ซอฟต์แวร์ออกมานั้นจะไม่แน่นอน จากนั้นทางทีม Manual Test จะทำการตรวจสอบดูว่ามี ServerBuild และ ClientBuild ของเวอร์ชันที่ต้องการหรือไม่ เพื่อนำมาทดสอบตาม Test Plan ซึ่งเป็นเอกสารขั้นตอนการทดสอบที่ได้ออกแบบไว้ หากพบข้อบกพร่อง (Defect) จะรายงานให้ทาง Programmer ทำการแก้ไข กรณีที่ทาง Manual Tester ต้องการทดสอบแบบอัตโนมัติ (Automated Testing) จะทำการคัดเลือก Test Plan และใบร้องขอส่งมาให้กับทีม ART

ภายในทีม ART จะแบ่งการทำงานออกเป็น 2 ทีมย่อย คือทีม Automated Test Developer ที่ทำหน้าที่สร้างสคริปท์ทดสอบอัตโนมัติ (Automated Test Script) และ ทีม Automated Test Executor ที่ทำหน้าที่วางแผนการทดสอบและเก็บผลการทดสอบ เมื่อทางทีม Automated Test Developer ได้รับ Test Plan จากทีม Manual Tester มาแล้ว จะนำมาแปลงให้เป็นสคริปท์ทดสอบอัตโนมัติ (Automated Test Script) โดยใช้ซอฟต์แวร์ WinRunner ซึ่งเป็นเครื่องมือการทดสอบซอฟต์แวร์ (Software Testing Tool) ที่มีคุณสมบัติในเรื่องของการบันทึกและเล่นซ้ำ (Record and Play) มาช่วยในการแปลง สำหรับ Automated Test Script ที่ได้มานั้นจะประกอบเป็นชุดตามวัตถุประสงค์ของ Test Plan ที่ได้รับมา Automated Test Script แต่ละชุดที่ได้มานั้นจะเรียกว่า Test Suite เมื่อ Automated Test Developer สร้าง Test Suite เสร็จแล้วจะส่งต่อให้กับ Automated Test Executor เพื่อนำไปทดสอบ (Execute) และส่งผลการทดสอบให้กับทางทีม Manual Test ตรวจสอบ

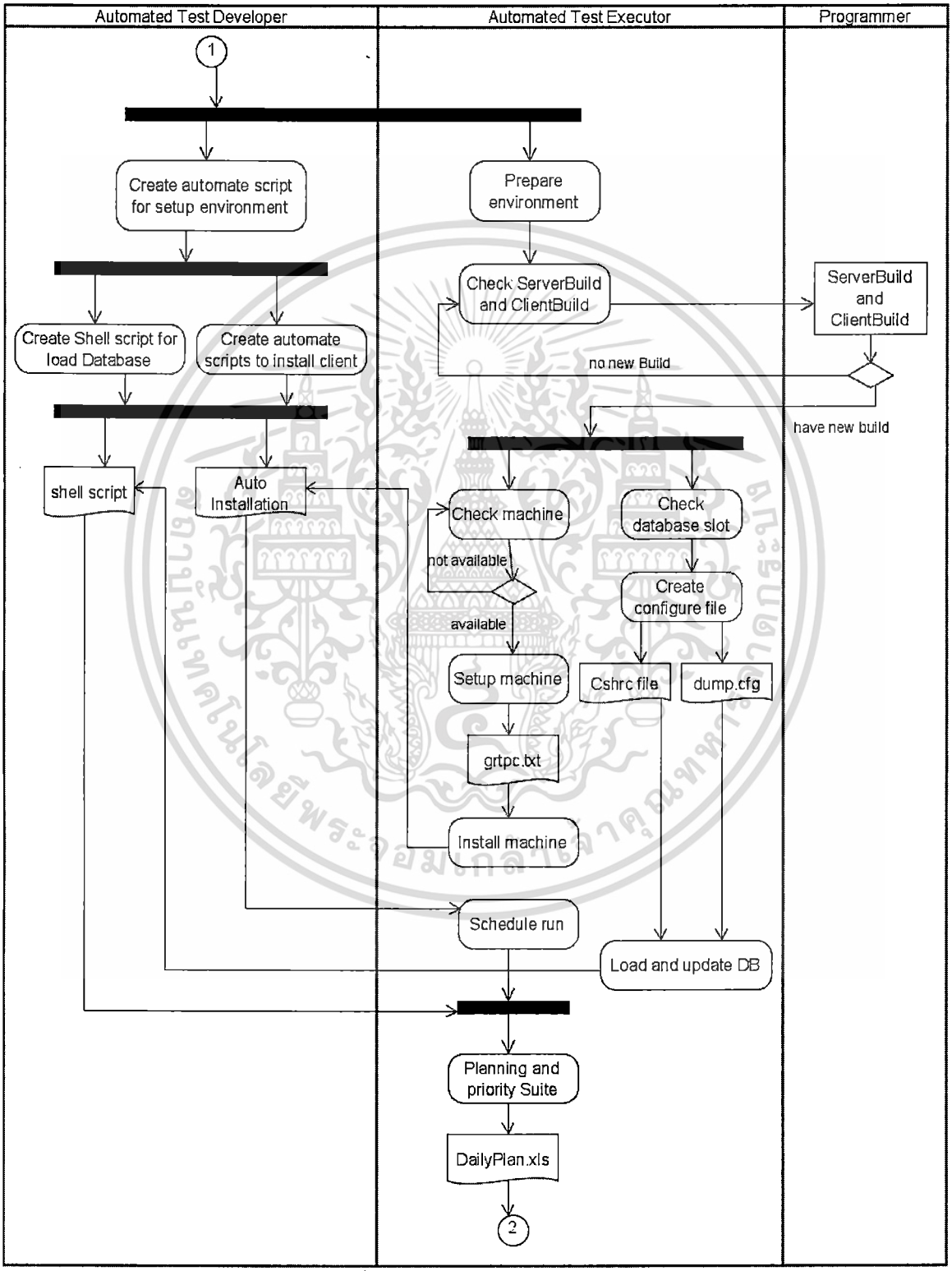
อีกครั้ง เมื่อทาง Manual Tester พิจารณาแล้วต้องการให้ทดสอบ Suite ตัวนี้กับ ServerBuild ล่าสุดทุกวัน (Daily Run) ก็จะส่งคำร้องขอไปยัง Automated Test Executor เพื่อทำการเตรียมสภาพแวดล้อมของการทดสอบ (Test Environment) ต่อไป



รูปที่ 3.2 แอกทิวิตีไดอะแกรมขั้นตอนการร้องขอและการสร้างสคริปท์อัตโนมัติ

2. ขั้นตอนการเตรียมสภาพแวดล้อมก่อนทำการทดสอบ (Preparation and setup test environment)

ก่อนจะเริ่มทำการทดสอบ Regression test ได้นั้น จะต้องมีการเตรียมสภาพแวดล้อมต่างๆ ให้พร้อม ซึ่งในเตรียมสภาพแวดล้อมของการทดสอบนั้นสามารถแบ่งงานออกเป็น 3 กลุ่มงาน ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 แอกทิวิตีไดอะแกรมขั้นตอนการเตรียมสภาพแวดล้อมก่อนทำการทดสอบ

จากรูปสามารถอธิบายการทำงานได้ดังนี้

2.1 ทาง Automated Test Developer จะทำการสร้าง Automated Test Script สำหรับช่วยให้ Automated Test Executor สามารถเตรียม Test Environment ได้สะดวกขึ้น ซึ่ง Automated Test Script ที่สร้างได้แก่

2.1.1 Shell Script สำหรับใช้โหลดฐานข้อมูลที่ของ Suite ที่จะทดสอบเข้าสู่ Database Slot

2.1.2 Auto Installation สำหรับใช้ติดตั้ง ClientBuild ของเวอร์ชันที่ต้องการบนเครื่องที่จะทดสอบ

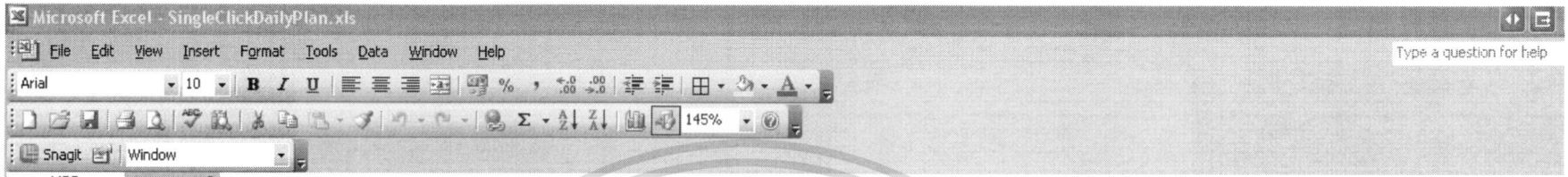
2.2 ทาง Automated Test Executor จะเริ่มทำการเตรียมสภาพแวดล้อมของการทดสอบ (Test Environment) สิ่งแรกและเป็นสิ่งสำคัญที่จะต้องเตรียมคือ ServerBuild และ ClientBuild โดยทาง Automated Test Executor จะทำการตรวจสอบว่ามีเวอร์ชันและวันที่ที่ต้องการหรือไม่ ถ้ายังไม่มีจะต้องรอจนกว่า ServerBuild และ ClientBuild ที่ต้องการถูกผลิตออกมาถึงจะเริ่มทำการทดสอบได้ แต่ช่วงเวลาที่จะได้ ServerBuild และ ClientBuild ตามที่ต้องการนั้นจะไม่แน่นอน ขึ้นอยู่กับทาง Programmer ว่าจะแก้ไขเสร็จและผลิตออกมาเมื่อไหร่ กรณีที่ได้ ServerBuild และ ClientBuild ที่มีวันที่และเวอร์ชันตรงตามที่ต้องการก็จะเริ่มดำเนินการเตรียม Test Environment ตัวอื่นๆ ได้แก่

2.2.1 Machine - ตรวจสอบเครื่องที่จะทำการทดสอบว่าว่างหรือไม่ กรณีที่ว่างจะทำการติดตั้ง ClientBuild ของเวอร์ชันที่ต้องการทดสอบ โดยในการติดตั้งสามารถทำได้สองแบบ คือ แบบติดตั้งเอง หรือใช้ Automated Test Script ที่ Automated Test Developer ได้สร้างไว้ จากนั้นทำการ setup ค่าเบื้องต้น จึงจะสามารถเปิดใช้งานแอพพลิเคชันที่จะทดสอบได้

2.2.2 Database Slot – ตรวจสอบดูว่าว่างหรือไม่ กรณีที่ว่างจะทำการสร้างคอนฟิกุเรชันไฟล์ต่างๆ ได้แก่ Cshrc file, dump.cfg เป็นต้น เพื่อใช้ในการโหลดฐานข้อมูล (Master Database) ซึ่งเป็นข้อมูลตั้งต้นสำหรับใช้ทดสอบ Test Suite เข้าสู่ Database Slot สำหรับการโหลดฐานข้อมูลสามารถสามารถทำได้สองแบบ คือ แบบติดตั้งเอง หรือใช้ Automated Test Script ที่ Automated Test Developer ได้สร้างไว้

2.2.3 Test Suite – ตรวจสอบดูว่าในเครื่องทดสอบมีสคริปท์ทดสอบ (Test Suite) ตรงตามเวอร์ชันที่ต้องการทดสอบหรือไม่

เมื่อทำการเตรียม Test Environment ต่างๆเรียบร้อยแล้ว ทาง Automated Test Executor แต่ละคนจะวางแผนและจัดลำดับ Suite ของการทดสอบ ในการวางแผนนั้นจะไปทำการระบุเครื่องและ Test Environment ต่างๆ ที่ใช้ในเอกสารที่เรียกว่า Daily Plan ต่อไป ตัวอย่างแสดงดังรูป 3.4

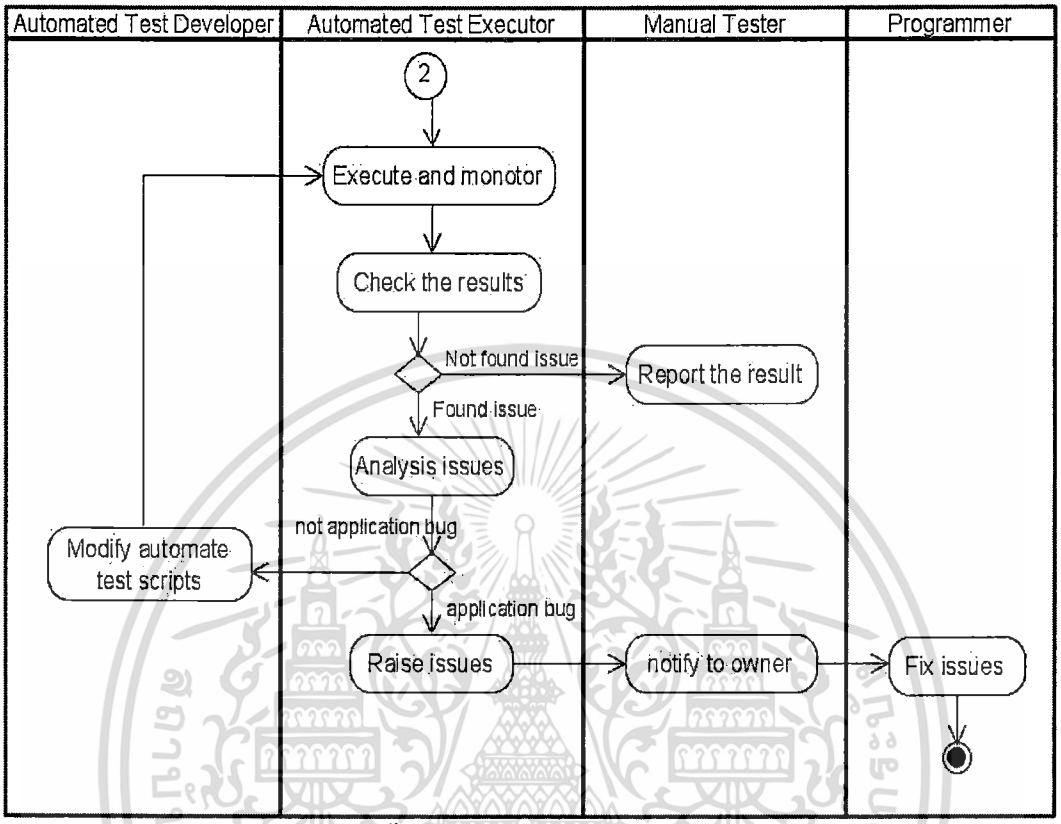


	A	B	C	D	E	F	G	H	J	K	L	M	N
1	**Machine**	**Suite**	**Config**	**Slot**	**Version**	**Build**	**Who**	**HiInvest Server**	**Application Server**	**Report Server**	**Database Server**	**Port**	**Application**
2	grt-w001	IDW	CFG1	grt6cs3	17.14.1	20100202	Joy	agamemnon	agamemnon	actuate8srv	corinsyb3	133	WebLogic
3	grt-w001	IDW2	CFG1	grt7cs3	17.14.1	20100202	Joy	agamemnon	agamemnon	actuate8srv	corinsyb3	135	WebLogic
4	grt-w002	ATB	CFG2	grt2cs1	17.11.7	20090819	Joy	percy	percy	actuate8srv	corinsyb1	537	WebSphere
5	grt-w002	ST	CFG2	grt3cs1	17.11.7	20090819	Joy	percy	percy	actuate8srv	corinsyb1	538	WebSphere
6	grt-w003	Risk	CFG1	grt1cs3	17.12.1	20100202	Ying	agamemnon	agamemnon	actuate8srv	corinsyb3	128	WebLogic
7	grt-w004	ATB	CFG3	GRT6CR	17.14.1	20100202	Eak	agamemnon	cicero	actuate8srv	corin1	265	WebSphere
8	grt-w005	MDL	CFG1	grt7cs3	17.14.1	20100202	Eak	agamemnon	agamemnon	actuate8srv	corinsyb3	135	WebLogic
9	grt-w005	STIO	CFG1	grt9cs3	17.14.1	20100202	Bee	agamemnon	agamemnon	actuate8srv	corinsyb3	137	WebLogic
10	grt-w103	IDW	CFG1	grt6cs3	17.14.1	20100202	Joy	agamemnon	agamemnon	actuate8srv	corinsyb3	133	WebLogic
11	grt-w102	IDW2	CFG1	grt7cs3	17.14.1	20100202	Joy	agamemnon	agamemnon	actuate8srv	corinsyb3	135	WebLogic
12	grt-w101	ATB	CFG2	grt2cs1	17.11.7	20090819	Joy	percy	percy	actuate8srv	corinsyb1	537	WebSphere
13	grt-w101	ST	CFG2	grt3cs1	17.11.7	20090819	Joy	percy	percy	actuate8srv	corinsyb1	538	WebSphere
14	grt-w014	Risk	CFG1	grt1cs3	17.12.1	20100202	Ying	agamemnon	agamemnon	actuate8srv	corinsyb3	128	WebLogic
15	grt-w014	ATB	CFG3	GRT6CR	17.12.1	20100202	Eak	agamemnon	cicero	actuate8srv	corin1	265	WebSphere
16	grt-w010	MDL	CFG1	grt7cs3	17.14.1	20100202	Eak	agamemnon	agamemnon	actuate8srv	corinsyb3	135	WebLogic
17	grt-w010	STIO	CFG1	grt9cs3	17.14.1	20100202	Bee	agamemnon	agamemnon	actuate8srv	corinsyb3	137	WebLogic

รูปที่ 3.4 ตัวอย่างเอกสาร Daily Plan สำหรับใช้ในการวางแผนการทดสอบ

3. ขั้นตอนการทดสอบและการวิเคราะห์ผลการทดสอบ (Execution and analysis result)

ขั้นตอนการทดสอบและการวิเคราะห์ผลการทดสอบ สามารถแสดงได้ดังรูปที่ 3.5

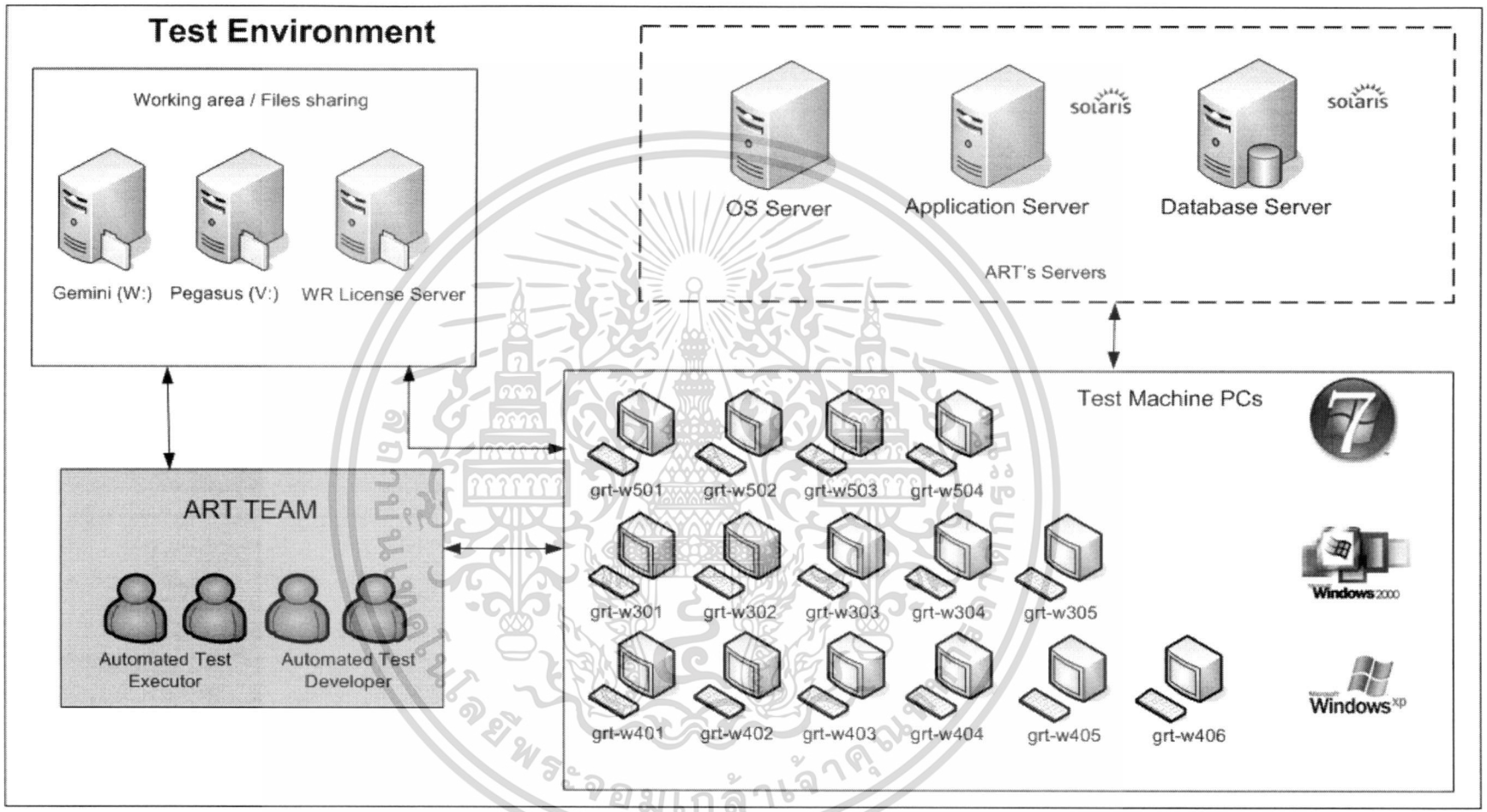


รูปที่ 3.5 แอกทิวิตีไดอะแกรมขั้นตอนการทดสอบและการวิเคราะห์ผลการทดสอบ

จากรูปสามารถอธิบายการทำงานได้ดังนี้ เมื่อได้ทำการเตรียม Test Environment ต่างๆ เรียบร้อยแล้ว ทาง Automated Test Executor จะรีโมตไปที่เครื่องที่ใช้ทดสอบเพื่อเปิดสคริปท์อัตโนมัติ (Automated Test Script) ด้วย WinRunner Tool จากนั้นกดปุ่มรัน (Run) ให้สคริปท์ทำงาน เมื่อการทดสอบเสร็จสิ้นจะส่งผลการทดสอบมายังอีเมลของ Automated Test Executor เพื่อทำการจัดเก็บผลการทดสอบและส่งรายงานให้กับ Manual Tester ในกรณีผลการทดสอบไม่ผ่าน เช่น ผลที่ได้ไม่ตรงกับผลที่ได้คาดหวัง (Expected Result) ไว้ Automated Test Executor จะทำการวิเคราะห์ถึงสาเหตุ หากพบว่าสาเหตุมาสคริปท์จะส่งไปให้ทาง Automated Test Developer ทำการแก้ไข แต่หากเป็นปัญหาเกี่ยวกับแอพพลิเคชันจะส่งปัญหา ไปให้ Manual Tester ทดสอบอีกครั้ง

3.1.2 องค์ประกอบของการทำงาน

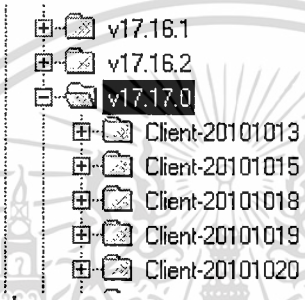
องค์ประกอบของการทำ Regression Testing ภายในทีม ART นั้น สามารถแสดงได้ดังรูปที่ 3.6



รูปที่ 3.6 แผนภาพ Test environment ที่ต้องใช้ในการทดสอบ Regression testing

จากรูป Test environment ที่ต้องใช้ในการทดสอบ Regression testing ประกอบไปด้วย

1. Working area/Files sharing คือเครื่องต่างๆที่ใช้เก็บไฟล์ที่ใช้ในการทดสอบ ได้แก่
   1.1 WinRunner Tool คือ เครื่องมือที่ช่วยในการบันทึกขั้นตอนการทำงานตาม Test Plan ที่ทาง Manual Tester ส่งมา เพื่อแปลงขั้นตอนดังกล่าวให้อยู่ในรูปของ Automated test scripts โดย Automated test scripts ที่ได้จะเรียกว่า Test Suite
   1.2 ClientBuild คือ ซอฟต์แวร์สำหรับใช้ติดตั้งบนเครื่องลูกข่ายที่จะทำการทดสอบ โดยเวอร์ชันและวันที่ของ ClientBuild ที่จะติดตั้งจะต้องตรงกับ ServerBuild โดยมีรูปแบบดังนี้ Client-YYYYMMDD ตัวอย่างเช่น Client_20100730 ซึ่งการจัดเก็บจะแบ่งแยกตามเวอร์ชัน ดังรูป 3.7



รูปที่ 3.7 แสดงตัวอย่างการเก็บ ClientBuild

2. Server คือเครื่องเซิร์ฟเวอร์ต่างๆที่ใช้ในการทดสอบ โดยจะประกอบด้วย Application Server , Database Server และ OS Server
   2.1 Application Server คือเครื่องเซิร์ฟเวอร์ที่จะมี Websphere หรือ Weblogic บรรจุอยู่
   2.2 Database Server คือเครื่องเซิร์ฟเวอร์ที่ใช้สร้าง Database Slot และโหลดข้อมูลเข้าสู่ Database Slot เพื่อใช้ในการทดสอบ โดย Database Slot ที่ใช้บรรจุข้อมูลที่อยู่ใน Master Database นั้นจะแบ่งไปตาม DBMS ดังนี้
      - Sybase 15 หรือ CFG1 จำนวน 9 slots
      - Sybase 12 หรือ CFG2 จำนวน 8 slots
      - Oracle 11g หรือ CFG3 จำนวน 13 slots
   2.3 OS Server คือ เครื่องเซิร์ฟเวอร์ที่มีระบบปฏิบัติการที่ต้องการทดสอบ เช่น Unix หรือ Linux เป็นต้น ซึ่งภายในเครื่องจะเก็บไฟล์ต่างๆ ดังนี้
      2.3.1 Master Database คือ ฐานข้อมูลตั้งต้นที่จะใช้ทำการทดสอบ โดยจะแยกตามแต่ละ Suite
      2.3.2 ServerBuild คือโปรแกรมสำหรับติดตั้งที่เครื่องแม่ข่าย (Server) ซึ่งได้จากการคอมไพล์ของ programmer โดยชื่อของ Build มีรูปแบบดังนี้ Mxxxx_date ซึ่ง

xxxx คือ เลขเวอร์ชัน ตัวอย่างเช่น M17170_20100730 หมายถึง Build ของเวอร์ชัน 17.17.0 วันที่ 30/07/2010

3. Test Machine คือ เครื่องที่ใช้ทำการทดสอบ ปัจจุบันมีทั้งหมด 15 เครื่อง แบ่งตามระบบปฏิบัติการได้ ดังนี้

- Windows 2000 จำนวน 6 เครื่อง
- Windows XP จำนวน 5 เครื่อง
- Windows 7 จำนวน 4 เครื่อง

3.2 ปัญหาที่พบของการทำงานปัจจุบัน

จากการศึกษาระบบการทำงานเดิมที่มีอยู่ พบว่าในการทดสอบ Regression Test นั้นก่อนจะทำการทดสอบได้ จะต้องเตรียมสภาพแวดล้อมของการทดสอบ (Test Environment) หลายอย่าง การจัดเตรียมและตรวจสอบยังคงต้องให้ผู้ทดสอบทำการจัดเตรียมด้วยตัวเอง ทำให้การทดสอบในแต่ละครั้งจะต้องเสียเวลาไปกับการเตรียมสภาพแวดล้อมเป็นส่วนมาก ถ้าภายในช่วงเวลานั้นมีเวอร์ชันของซอฟต์แวร์ที่จะต้องส่งมอบให้กับลูกค้าหลายเวอร์ชัน การจัดเตรียมจะยิ่งวุ่นวายและเกิดข้อผิดพลาดได้ง่าย ในกรณีที่ผู้ทดสอบเป็นพนักงานใหม่ ไม่มีความชำนาญในการจัดเตรียมด้วยแล้วอาจทำให้เกิดข้อผิดพลาดได้ ซึ่งจะส่งผลกระทบทำให้ไม่สามารถทำการทดสอบได้ เมื่อการทดสอบล่าช้า ผลที่ตามมาคือการส่งมอบงานให้ลูกค้าที่ไม่ทันเวลา

ด้วยปัญหาดังกล่าว ทำให้เกิดแนวคิดที่จะพัฒนาแอพพลิเคชันสำหรับเป็นเครื่องมือที่ช่วยในการตรวจสอบและเตรียม Test Environment ต่างๆที่ต้องทำแบบ Manual ให้เป็นแบบอัตโนมัติมากขึ้น เช่น จากเดิมผู้ใช้จะต้องเข้าไปที่เครื่องทดสอบแต่ละเครื่องและสั่งให้ Test Script ทำงาน ด้วยระบบที่จะพัฒนาขึ้นมาจะช่วยสั่งให้ Test Script ที่ได้ตั้งไว้สามารถทำงานได้อย่างอัตโนมัติ เพื่อช่วยลดเวลาในการจัดเตรียมสภาพแวดล้อมให้น้อยลง ทำให้ผู้ทดสอบมีเวลาศึกษาและวิเคราะห์ปัญหาที่เกี่ยวข้องทางด้านธุรกิจขององค์กรมากขึ้น นอกจากนี้ระบบจะยังสามารถตั้งเวลาการทดสอบได้ เพื่ออำนวยความสะดวกให้กับผู้ทดสอบในกรณีที่ไม่สามารถทดสอบในช่วงเวลานั้นได้ โดยรายละเอียดจะนำเสนอในบทที่ 4 ต่อไป

บทที่ 4
การวิเคราะห์และออกแบบระบบใหม่

จากการวิเคราะห์ระบบการทำงานในปัจจุบันและทำให้ทราบถึงขั้นตอนในการดำเนินงานในปัจจุบัน ปัญหา และข้อจำกัดต่างๆ ดังนั้น จึงได้มีการวิเคราะห์และออกแบบระบบงานใหม่ขึ้นมา เพื่อช่วยลดปัญหาในการดำเนินงานปัจจุบัน และให้ได้ระบบที่สามารถทำงานได้ตรงกับความต้องการของผู้ใช้งาน โดยใช้หลักการวิเคราะห์และออกแบบเชิงวัตถุเป็นเครื่องมือในการวิเคราะห์และออกแบบระบบงานใหม่

4.1 การวิเคราะห์ความต้องการของผู้ใช้

ระบบสำหรับตรวจสอบสภาพแวดล้อมและการตั้งเวลาการทดสอบ ถูกออกแบบให้รองรับและทำงานตามความต้องการของผู้ใช้ ซึ่งข้อกำหนดต่างๆ ได้จากการรวบรวมความต้องการของผู้ใช้ ดังนี้

4.1.1 ความต้องการทางด้านการเตรียม Test Environment

- ระบบสามารถตรวจสอบเวอร์ชันและวันที่ของ Build และ Client Install Shield ตามที่ได้วางแผนการทดสอบไว้ได้ โดยสามารถระบุระยะเวลาที่จะให้ระบบทำการตรวจสอบได้ เช่น ตรวจสอบทุกๆ 3 นาทีเป็นเวลา 1 ชั่วโมง เป็นต้น ระยะเวลาที่ระบบสามารถวนตรวจสอบได้นานสุด 24 ชั่วโมง
- ระบบสามารถตรวจสอบขนาดของพื้นที่บนเครื่องเซิร์ฟเวอร์และไดเรคเทอรี่ (Directory) ที่ได้ Map Network Drive ไว้ได้ เนื่องจากในการทดสอบจะต้องมีการแตกไฟล์ฐานข้อมูลที่อยู่ในรูปของซิปไฟล์บนเซิร์ฟเวอร์ ถ้ามีการทดสอบหลายตัวและขนาดพื้นที่ไม่เพียงพอจะไม่สามารถทำการทดสอบได้ ดังนั้นขณะที่ทำการตั้งเวลาทดสอบระบบจะต้องทำการตรวจสอบขนาดของพื้นที่ตามไดเรคเทอรี่ที่ได้ระบุไว้ และมีการแจ้งเตือนในกรณีที่พื้นที่ไม่เพียงพอได้
- ระบบสามารถตรวจสอบสถานะของ Database Slot ได้ เนื่องจากในการทดสอบซอฟต์แวร์นั้น จะต้องโหลดข้อมูลเข้าสู่ Database Slot ก่อน ถ้า Database Slot ที่ต้องการใช้มีสถานะไม่ว่างหรือเกิดการชนกันของข้อมูลจะทำให้ Database Slot นั้นเสียหายได้ ดังนั้นระบบที่พัฒนาขึ้นจะต้องสามารถตรวจสอบสถานะของ Database Slot ได้ และระบบจะต้องมีการแจ้งเตือนในกรณีผู้ทดสอบเลือกใช้ Database Slot ที่ใช้งานอยู่
- ระบบสามารถทำการโหลดข้อมูลจาก Master Database เข้าสู่ Database Slot ที่ได้เลือกไว้อย่างอัตโนมัติ

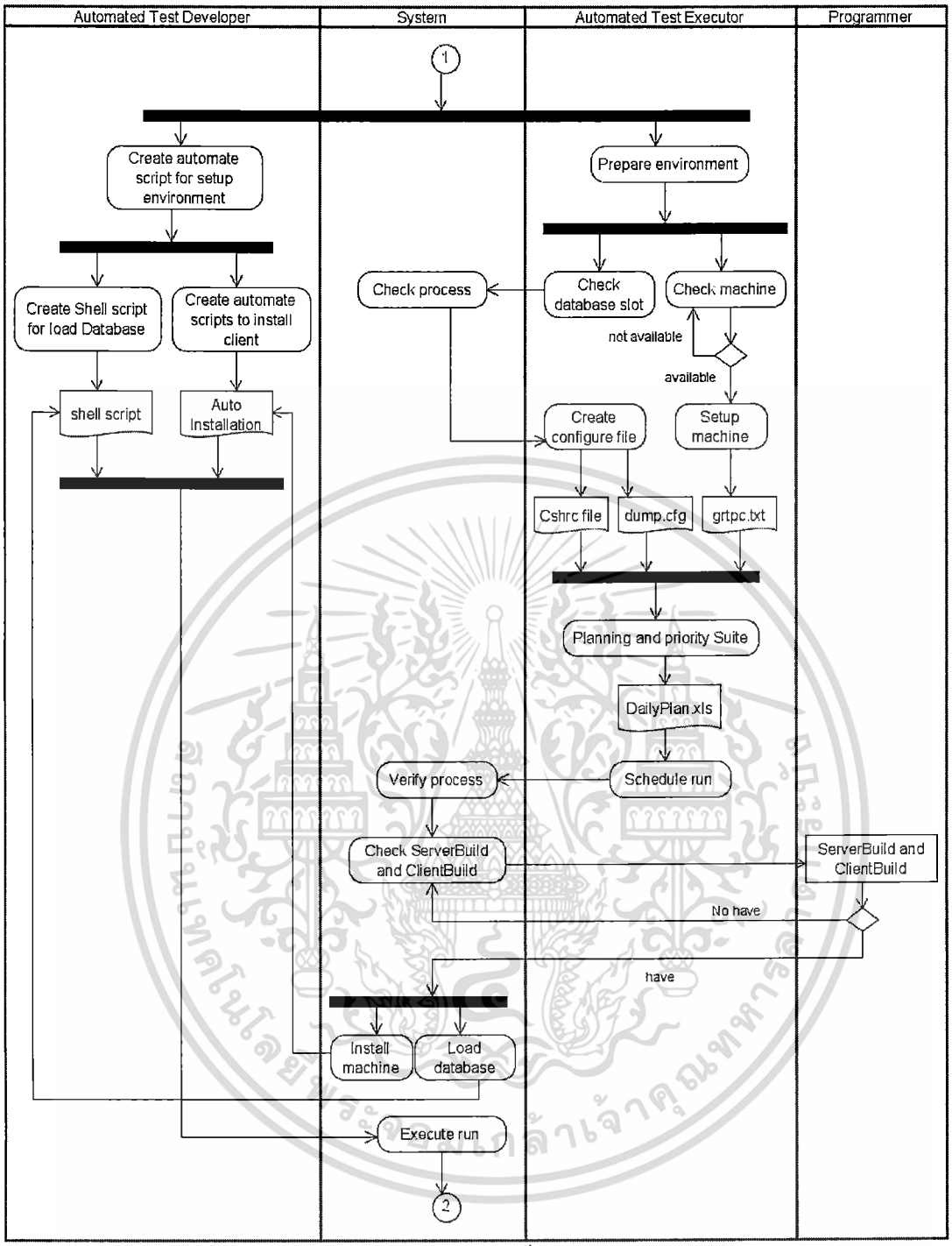
- ระบบสามารถสั่งให้ซอฟต์แวร์ที่จะทดสอบทำการติดตั้งที่เครื่องทดสอบได้อย่างอัตโนมัติตามที่ได้วางแผนทดสอบเอาไว้
- ระบบสามารถให้ผู้ทดสอบจัดการข้อมูลส่วนที่เป็นข้อมูลหลัก เพื่อนำข้อมูลดังกล่าวไปใช้ในการวางแผนการทดสอบต่อไปได้ ข้อมูลหลักได้แก่ ข้อมูลซอฟต์แวร์ที่ใช้ทดสอบ ข้อมูลเครื่องทดสอบ ข้อมูลเครื่องเซิร์ฟเวอร์ ข้อมูล Database Slot เป็นต้น
- ผู้ทดสอบสามารถวางแผนการทดสอบในระบบได้ จากเดิมจะต้องทำการวางแผนการทดสอบในเอกสารเอ็กซ์เซล ซึ่งบางครั้งไฟล์อาจเกิดปัญหาทำให้ไม่สามารถทำการทดสอบได้
- ระบบสามารถเก็บ Log สถานะการทำงานของ Database Slot ได้ เช่น การโหลดฐานข้อมูล (Load Database) การอัพเดตฐานข้อมูล (Update Database) เป็นต้น

4.1.2 ความต้องการทางด้านการตั้งเวลาการทดสอบ

- ระบบสามารถตั้งเวลาล่วงหน้าในการทดสอบได้ ในการตั้งเวลาสามารถเลือกได้ ดังนี้
  - ทำการทดสอบทันที ใช้สำหรับในกรณีที่ Test Environment ของการทดสอบพร้อมที่จะทดสอบ
  - ตั้งเวลาทดสอบและระบบเตรียม Test Environment ให้ ใช้สำหรับในกรณีที่ Test Environment ของการทดสอบบางส่วนยังไม่พร้อม
  - ตั้งเวลาทดสอบแต่ระบบไม่เตรียม Test Environment ให้ใช้สำหรับในกรณีที่ต้องการเตรียม Test Environment ด้วยตัวเอง

4.2 ขั้นตอนการทำงานของระบบงานใหม่

จากการรวมรวบความต้องการของผู้ใช้ทำให้พบว่าระบบที่ผู้ใช้ต้องการนั้นเกี่ยวข้องกับขั้นตอนการเตรียมสภาพแวดล้อมของการทดสอบ (Test Environment) ซึ่งจากเดิม ทาง Automated Test Executor จะต้องรอเวอร์ชันและวันที่ของซอฟต์แวร์ที่จะทดสอบให้พร้อมก่อน จึงจะเริ่มเตรียม Test Environment ตัวอื่นๆที่เกี่ยวข้องได้ แต่ด้วยระบบใหม่จะช่วยให้ Automated Test Executor ทำการเตรียมความพร้อมของ Test Environment ตัวอื่นๆที่เกี่ยวข้องไว้ก่อนได้ จากนั้นระบบจะทำการตรวจสอบเวอร์ชันและวันที่ของ Build และ Client Install Shield ให้ สามารถอธิบายด้วยแอกทิวิตีไดอะแกรม ดังรูปที่ 4.1



รูปที่ 4.1 แอกทิวิตีไดอะแกรมแสดงขั้นตอนการเตรียมTest Environment

จากรูปสามารถอธิบายการทำงานของการเตรียม Test Environment โดยแบ่งการทำงานได้ดังนี้

1. Automated Test Developer จะทำการสร้างสคริปท์ที่เกี่ยวกับการจัดการ Environment ซึ่งมีด้วยกัน 2 ส่วน คือ Shell script เป็นสคริปท์ไว้สำหรับช่วยในการโหลดข้อมูลและติดต่อกับ Unix Server และอีกส่วนหนึ่งคือ Auto Installation เป็นสคริปท์ไว้สำหรับช่วยในการติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ

2. Automated Test Executor จะทำการตรวจสอบเตรียม Test Environment ให้พร้อมตามที่ได้รับการร้องขอให้ทดสอบ ซึ่งจะต้องทำการตรวจสอบความพร้อมของใช้งานของ Database Slot และ เครื่องที่ใช้ทดสอบว่าว่างพร้อมใช้งานหรือไม่ รวมทั้งทำการเตรียมไฟล์คอนฟิกุเรชันไฟล์ต่างๆ จากนั้น Automated Test Executor จะทำการจัดลำดับและวางแผนการทดสอบในไฟล์ DailyPlan.xls จะตั้งเวลาการทดสอบ

3. System ในส่วนของระบบจะทำหน้าที่ตรวจสอบเงื่อนไขต่างๆ ตามที่ Automated Test Executor ได้กำหนดไว้ โดยจะเริ่มทำการตรวจสอบเงื่อนไขเวลา เมื่อถึงเวลาทดสอบระบบจะทำการตรวจสอบในเรื่องของวันที่และเวอร์ชันของซอฟต์แวร์ที่ต้องการทดสอบ ในกรณีที่ไม่พบวันที่และเวอร์ชันของซอฟต์แวร์ที่จะใช้ทดสอบ ระบบจะทำการรอและตรวจสอบไปเรื่อยๆ จนกว่าจะหมดเวลา แต่ถ้าพบจะเริ่มดำเนินการทดสอบต่อไป

4.3 การวิเคราะห์ออกแบบระบบงานใหม่

ในการวิเคราะห์และออกแบบระบบสำหรับเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบจะใช้หลักการวิเคราะห์และออกแบบเชิงวัตถุ โดยใช้ไดอะแกรมต่างๆ ที่สำคัญได้แก่ ยูสเคสไดอะแกรม แอกทิวิตีไดอะแกรม คลาสไดอะแกรม และซีเควนซ์ไดอะแกรม

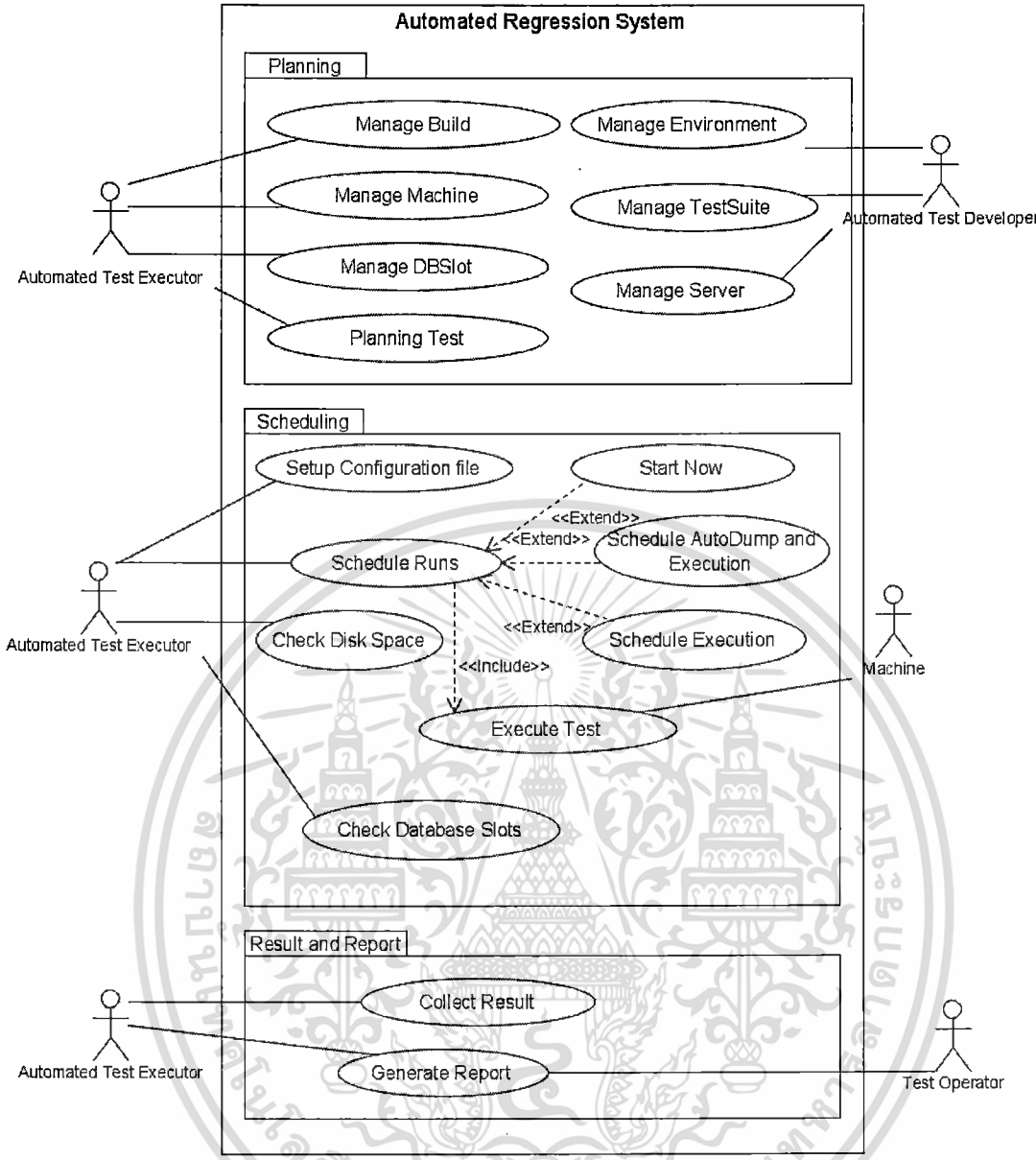
4.3.1 ยูสเคสไดอะแกรม

เพื่อแสดงภาพรวมในการทำงานของระบบ จึงได้เขียนแผนภาพที่ช่วยอธิบายส่วนประกอบต่างๆ รวมถึงขอบเขตการทำงานของระบบออกเป็นยูสเคสไดอะแกรม ได้ดังรูปที่ 4.2 จากรูปแสดงยูสเคสไดอะแกรมของระบบ Automated Regression ประกอบไปด้วย 3 ระบบย่อย ดังนี้

1. Planning คือส่วนของการวางแผนการทดสอบ
2. Scheduling คือ ส่วนของการเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบ
3. Result and Report คือ ส่วนของการรายงานผลการทดสอบ

โดยยูสเคสไดอะแกรมทั้ง 3 ระบบย่อย มีแอกเตอร์ที่เป็นการแสดงถึงบุคคลที่เกี่ยวข้องกับระบบ ซึ่งจากยูสเคสไดอะแกรมดังรูปที่ 4.2 นั้นประกอบด้วย 4 แอกเตอร์

1. Automated Test Executor คือ ผู้ทดสอบ มีหน้าที่จัดการ Test Environment ต่างๆภายในระบบ และ เตรียมสภาพแวดล้อมในการทดสอบ ตรวจสอบพื้นที่ที่ใช้ทดสอบว่าเพียงพอหรือไม่ และตั้งเวลาทดสอบ รวมถึงทำหน้าที่วิเคราะห์ผลการทดสอบ
2. Automated Test Developer คือ ผู้ทดสอบที่มีหน้าที่จัดการคอนฟิกุเรชันไฟล์และสคริปท์อัตโนมัติต่างๆ ที่ใช้ในการทดสอบ
3. Machine คือ เครื่องที่ใช้ทดสอบ
4. Test Operator คือ ผู้ที่ทำหน้าที่ออกรายงานส่งให้ผู้บริหารและผู้ที่เกี่ยวข้อง



รูปที่ 4.2 ยูสเคสไดอะแกรมของระบบ Automated Regression

ยูสเคสระบบย่อยของการวางแผนการทดสอบ (Planning) ประกอบด้วย 7 ยูสเคส ซึ่งมีรายละเอียด ดังนี้

1. Manage Build คือ การจัดการเวอร์ชันและวันที่ของซอฟต์แวร์ที่จะทำการทดสอบ
2. Manage Machine คือ การจัดการเครื่องที่ใช้ทดสอบ กรณีที่มีการเพิ่มหรือลดเครื่องที่ใช้ทดสอบ
3. Manage DBSlot คือ การจัดการ Database Slot ที่ใช้ในการทดสอบ
4. Manage Environment คือ การจัดการ Environment ที่ใช้ในการทดสอบ ว่าต้องการทดสอบบน Environment แบบใด
5. Manage Testsuite คือ การจัดการสคริปท์ที่ใช้ทดสอบ

6. Manage Server คือก การจัดการเครื่องเซิร์ฟเวอร์ต่างๆที่ใช้ทดสอบ

7. Planning Test คือ การวางแผนการทดสอบในแต่ละครั้ง

ยูสเคสระบบย่อยของการเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบ (Scheduling) ประกอบด้วย 8 ยูสเคส ซึ่งมีรายละเอียด ดังนี้

1. Check Disk Space คือ การตรวจสอบพื้นที่บน Server ที่จะทดสอบว่าเพียงพอหรือไม่

2. Check Database Slot คือ การตรวจสอบการใช้งานของ Database Slot ที่จะได้ทดสอบว่ามีตัวไหนว่างพร้อมให้ใช้งานได้

3. Setup Configuration File คือ การเตรียมไฟล์ต่างไที่เกี่ยวข้องกับการทดสอบ ไฟล์ต่างๆที่ต้องเตรียมนั้นได้แก่ Dump.cfg สำหรับทำ AutoDump และ grt.txt สำหรับทำการติดตั้งซอฟต์แวร์บนเครื่องทดสอบ เป็นต้น

4. Schedule Runs คือ การกำหนดเวลาการทดสอบ ซึ่งสามารถเลือกได้ 3 แบบ

5. Start Now คือ รูปแบบของการทดสอบ โดยจะทำการทดสอบทันที

6. Schedule AutoDump and Execute คือ รูปแบบของการกำหนดเวลาทดสอบล่วงหน้า โดยรูปแบบนี้จะมีการเตรียมสภาพแวดล้อม เช่น AutoDump ตรวจสอบ Build เป็นต้นและทำการทดสอบเมื่อถึงเวลาที่ได้กำหนดไว้

7. Schedule Execution คือ รูปแบบของการกำหนดเวลาทดสอบล่วงหน้า โดยรูปแบบนี้จะทำการทดสอบอย่างเดียว

8. Execute Test คือ การทดสอบบนเครื่องทดสอบแต่ละเครื่อง การทดสอบจะเริ่มขึ้นก็ต่อเมื่อเงื่อนไขต่างๆ ที่ได้ระบุไว้ครบถ้วน

ยูสเคสระบบย่อยของการรายงานผลการทดสอบ (Reporting) ประกอบด้วย 2 ยูสเคส ซึ่งมีรายละเอียด ดังนี้

1. Collect Result คือ การเก็บผลการทดสอบ โดยหลังจากระบบส่งอีเมลแจ้งผลการทดสอบไปยังผู้ที่เกี่ยวข้องแล้ว ผู้ที่เกี่ยวข้องต่างๆจะเข้ามาวิเคราะห์และอัพเดตสถานะของผลการทดสอบ

2. Generate Report คือ การออกรายงานผลการทดสอบ

รายละเอียดของแต่ละยูสเคส สามารถอธิบายได้ด้วยคำอธิบายยูสเคส ดังตารางที่ 4.1 ถึงตารางที่ 4.17

ตารางที่ 4.1 รายละเอียดยูสเคสของการจัดการเวอร์ชันและวันที่ของซอฟต์แวร์ (Manage Build)

Use Case Name	Manage Build	
Triggering Event	Automated Test Executor ต้องการเพิ่มหรือแก้ไขข้อมูลเวอร์ชันและวันที่ของซอฟต์แวร์ที่ใช้ทดสอบ	
Brief Description	ซอฟต์แวร์ที่ใช้ทดสอบ หรือ Build นั้นจะถูกสร้างขึ้นมาทุกวันโดยผู้พัฒนาระบบ แต่ละ Build จะมีหลายเวอร์ชัน ดังนั้นผู้ทดสอบจะต้องเลือกเวอร์ชันและวันที่ของ Build ให้ตรงกับที่ได้มีการร้องขอให้ทดสอบ ถ้าไม่มีเวอร์ชันและวันที่ของ Build นั้นอยู่ในฐานข้อมูล ผู้ทดสอบสามารถจัดการเพิ่มหรือแก้ไขเวอร์ชันและวันที่ของ Build ได้	
Actor	Automated Test Executor	
Related Use Case	ไม่มี	
Preconditions	Automated Test Executor ต้องการจัดการข้อมูลของซอฟต์แวร์ที่ใช้ทดสอบ 1.ในกรณีเพิ่มจะเป็นการเพิ่มเวอร์ชันและวันที่ของซอฟต์แวร์ที่ใช้ทดสอบเข้ามาจัดเก็บในฐานข้อมูลของระบบ 2. การแก้ไขเป็นการเรียกข้อมูล Build ที่มีอยู่เดิมขึ้นมาเพื่อทำการปรับปรุงแล้วบันทึกลงในตำแหน่งเดิม 3. การลบเป็นเรียกข้อมูล Build ที่มีอยู่ขึ้นมาและลบออกไปจากฐานข้อมูล	
Post conditions	Automated Test Executor เพิ่ม แก้ไขหรือลบข้อมูล Build ได้ ข้อมูลของซอฟต์แวร์ที่ใช้ทดสอบถูกจัดเก็บอย่างถูกต้องและสามารถนำไปใช้ในการวางแผนการทดสอบต่อไปได้	
Flow of Activities	**Actor**	**System**
	1. เพิ่มข้อมูล 1. Automated Test Executor เข้าสู่เมนู Build 2. กดปุ่ม Add เพื่อทำการเพิ่มข้อมูล	1.1 ระบบแสดงหน้าจอ Build Configuration พร้อมรายการ Build ที่มีอยู่ในระบบ 2.1 ระบบแสดงหน้าจอและฟอร์มการเพิ่มข้อมูล Build

ตารางที่ 4.1 (ต่อ)

Flow of Activities	Actor	System
	3. Automated Test Executor กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save	3.1 ระบบแสดงข้อมูล Build บนหน้าจอและจัดเก็บข้อมูล Build ลงฐานข้อมูลของระบบ
	2. แก้ไขข้อมูล	
	1. Automated Test Executor เลือกข้อมูล Build ที่ต้องการแก้ไขแล้วกดปุ่ม Update	1.1 ระบบแสดงหน้าจอและฟอร์มสำหรับแก้ไขข้อมูล Build
	2. Automated Test Executor กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save	2.1 ระบบแสดงข้อมูล Build ที่ได้ทำการแก้ไขบนหน้าจอและจัดเก็บข้อมูลลงฐานข้อมูลของระบบ
	3. ลบข้อมูล	
	1. Automated Test Executor เลือกข้อมูล Build ที่ต้องการลบแล้วกดปุ่ม Delete	1.1 ระบบแสดงข้อความยืนยันการลบ
	2. Automated Test Executor ยืนยันการลบข้อมูล	2.1 ระบบจะลบข้อมูล Build นั้นๆ ออกจากหน้าจอและฐานข้อมูลของระบบ
Exception Conditions	ไม่มี	

ตารางที่ 4.2 รายละเอียดยูสเคสของการจัดการเครื่องที่ใช้ทดสอบ (Manage Machine)

Use Case Name	Manage Machine
Triggering Event	Automated Test Executor ต้องการเพิ่ม แก้ไขหรือลบข้อมูลเครื่องที่ใช้ในการทดสอบ
Brief Description	การทดสอบแต่ละครั้งจะทำการทดสอบบนเครื่องที่เตรียมไว้สำหรับทำการทดสอบเฉพาะ ซึ่งแต่ละเครื่องจะมีระบบปฏิบัติการที่แตกต่างกันไปขึ้นอยู่การร้องขอในแต่ละครั้ง
Actor	Automated Test Executor

ตารางที่ 4.2 (ต่อ)

Related Use Case	ไม่มี	
Preconditions	Automated Test Executor ต้องการจัดการข้อมูลของเครื่องทดสอบ 1. ในกรณีเพิ่มจะเป็นการเพิ่มข้อมูลของเครื่องทดสอบเข้ามาจัดเก็บในระบบ 2. การแก้ไขเป็นการเรียกข้อมูลเครื่องที่ใช้ในการทดสอบที่มีอยู่เดิมในระบบขึ้นมาเพื่อทำการปรับปรุงแล้วบันทึกลงในตำแหน่งเดิม 3. การลบเป็นการเรียกข้อมูลเครื่องที่ใช้ในการทดสอบที่มีอยู่ออกไปจากฐานข้อมูลของระบบ	
Post conditions	Automated Test Executor เพิ่ม แก้ไขหรือลบข้อมูลเครื่องที่ใช้ในการทดสอบได้ ข้อมูลของเครื่องทดสอบถูกจัดเก็บอย่างถูกต้องและสามารถนำไปใช้ในการวางแผนการทดสอบต่อไปได้	
Flow of Activities	**Actor**	**System**
	1. เพิ่มข้อมูล 1. Automated Test Executor เข้าสู่เมนู Machine 2. Automated Test Executor กดปุ่ม Add เพื่อทำการเพิ่มข้อมูล 3. Automated Test Executor กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save **2. แก้ไขข้อมูล** 1. Automated Test Executor เลือกข้อมูล Machine ที่ต้องการแก้ไขแล้วกดปุ่ม Update 2. Automated Test Executor กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save	1.1 ระบบแสดงหน้าจอ Machine โดยจะแสดงข้อมูลของเครื่องทดสอบที่มีอยู่ในระบบ 2.1 ระบบแสดงหน้าจอและฟอร์มการเพิ่มข้อมูล Machine 3.1 ระบบแสดงข้อมูล Machine บนหน้าจอและจัดเก็บข้อมูล Machine ลงในฐานข้อมูลของระบบ 1.1 ระบบแสดงหน้าจอและฟอร์มการแก้ไขข้อมูล Machine 2.1 ระบบแสดงข้อมูล Machine บนหน้าจอและจัดเก็บข้อมูล Machine ลงฐานข้อมูลของระบบ

ตารางที่ 4.2 (ต่อ)

Flow of Activities	Actor	System
	3. ลบข้อมูล 1. Automated Test Executor เลือกข้อมูล Machine ที่ต้องการลบแล้วกดปุ่ม Delete 2. Automated Test Executor ยืนยันการลบข้อมูล Machine	1.1 ระบบแสดงข้อความยืนยันการลบ 2.1 ระบบจะลบข้อมูล Machine ออกจากหน้าจอและฐานข้อมูลของระบบ
Exception Conditions	ไม่มี	

ตารางที่ 4.3 รายละเอียดยูสเคสของการจัดการ Database Slot (Manage DBSlot)

Use Case Name	Manage DBSlot
Triggering Event	Automated Test Executor ต้องการเพิ่ม แก้ไข หรือลบข้อมูล Database Slot ที่ใช้ในการทดสอบ
Brief Description	การทดสอบแต่ละ Test Suite จะมีการโหลดข้อมูล เข้าสู่ Database Slot เพื่อใช้เป็นข้อมูลตั้งต้นในการทดสอบ โดย Database Slot จะขึ้นอยู่กับ Environment
Actor	Automated Test Executor
Related Use Case	ไม่มี
Preconditions	Automated Test Executor ต้องการจัดการข้อมูลของ Database Slot 1.ในกรณีเพิ่มจะเป็นการเพิ่มข้อมูลของ Database Slot เข้ามาจัดเก็บในระบบ 2. การแก้ไขเป็นการเรียกข้อมูล Database Slot เดิมที่มีอยู่ในระบบขึ้นมาเพื่อทำการปรับปรุงแล้วบันทึกลงในตำแหน่งเดิม 3. การลบเป็นการเรียกข้อมูล Database Slot ที่มีอยู่ออกไปจากฐานข้อมูลของระบบ
Post conditions	Automated Test Executor ทำการเพิ่ม แก้ไขหรือลบข้อมูลได้ โดยข้อมูล Database Slot ถูกจัดเก็บอย่างถูกต้องและสามารถนำไปใช้ในการวางแผนการทดสอบต่อไปได้

ตารางที่ 4.3 (ต่อ)

Flow of Activities	Actor	System
	1. เพิ่มข้อมูล 1. เข้าสู่เมนู Database Slot เพื่อจัดการข้อมูล Database Slot	1.1 ระบบแสดงหน้าจอ Database Slot โดยจะแสดงรายการข้อมูล Database Slot ที่มีอยู่ในระบบ
	2. Automated Test Executor กดปุ่ม Add เพื่อทำการเพิ่มข้อมูล Database Slot	2.1 ระบบแสดงหน้าจอและฟอร์มการเพิ่มข้อมูล Database Slot
	2. Automated Test Executor กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save	3.1 ระบบแสดงข้อมูล Database Slot บนหน้าจอและจัดเก็บข้อมูล Database Slot ลงฐานข้อมูลของระบบ
	2. แก้ไขข้อมูล 1. Automated Test Executor เลือกข้อมูล Database Slot ที่ต้องการแก้ไขแล้วกดปุ่ม Update	1.1 ระบบแสดงหน้าจอและฟอร์มการแก้ไขข้อมูล Database Slot
	2. Automated Test Executor แก้ไขข้อมูล Database Slot แล้วกด Save	2.1 ระบบแสดงข้อมูล Database Slot บนหน้าจอและจัดเก็บข้อมูลลงฐานข้อมูลของระบบ
	3. ลบข้อมูล 1. Automated Test Executor เลือกข้อมูล Database Slot ที่ต้องการลบแล้วกดปุ่ม Delete	1.1 ระบบแสดงข้อความยืนยันการลบ
	2. Automated Test Executor ยืนยันการลบข้อมูล Database Slot	2.1 ระบบจะลบข้อมูล Database Slot ออกจากหน้าจอและฐานข้อมูลของระบบ
Exception Conditions	ไม่มี	

ตารางที่ 4.4 รายละเอียดยูสเคสของการจัดการ Environment (Manage Environment)

Use Case Name	Manage Environment	
Triggering Event	Automated Test Developer ต้องการเพิ่ม แก้ไขหรือลบข้อมูล Environment ที่ใช้ในการทดสอบ	
Brief Description	การทดสอบแต่ละครั้งจะมีการร้องขอให้ทดสอบบน environment ที่แตกต่างกันไป เช่น ทดสอบบนระบบปฏิบัติการยูนิกซ์ วินโดว์ หรือ ลีนุกซ์ เป็นต้น หรือต้องการทดสอบด้วยระบบการจัดการฐานข้อมูล (DBMS) ที่เป็น Sybase หรือ Oracle เป็นต้น	
Actor	Automated Test Developer	
Related Use Case	ไม่มี	
Preconditions	Automated Test Developer ต้องการจัดการข้อมูลของ Environment ที่ใช้ทดสอบ 1.ในกรณีเพิ่มจะเป็นการเพิ่มข้อมูลของ Environment ที่ใช้ทดสอบเข้ามาจัดเก็บในระบบ 2. การแก้ไขเป็นการเรียกข้อมูล Environment ที่มีอยู่เดิมขึ้นมาเพื่อทำการปรับปรุงแล้วบันทึกลงในตำแหน่งเดิม 3. การลบเป็นการเรียกข้อมูล Environment ที่มีอยู่ออกไปจากฐานข้อมูลของระบบ	
Post conditions	Automated Test Developer เพิ่ม แก้ไขข้อมูลได้ ข้อมูลของ environment ที่ใช้ทดสอบถูกจัดเก็บอย่างถูกต้องและสามารถนำไปใช้ในการวางแผนการทดสอบต่อไปได้	
Flow of Activities	**Actor**	**System**
	1. เพิ่มข้อมูล 1. เข้าสู่เมนู CFG เพื่อจัดการข้อมูล Environment 2. Automated Test Developer กดปุ่ม Add เพื่อทำการเพิ่มข้อมูล Environment	1.1 ระบบแสดงหน้าจอ CFG โดยจะแสดงข้อมูล Environment ที่มีอยู่ในระบบ 2.1 ระบบแสดงหน้าจอและฟอร์มการเพิ่มข้อมูล Environment

ตารางที่ 4.4 (ต่อ)

Flow of Activities	Actor	System
	3. Automated Test Developer กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save	3.1 ระบบแสดงข้อมูล Environment บนหน้าจอและจัดเก็บข้อมูล Environment ลงฐานข้อมูลของระบบ
	2. แก้ไขข้อมูล	
	1. Automated Test Developer เลือกข้อมูล Environment ที่ต้องการแก้ไขแล้วกดปุ่ม Update	1.1 ระบบแสดงหน้าจอและฟอร์มการแก้ไขข้อมูล Environment
	2. Automated Test Developer แก้ไขข้อมูลแล้วกด Save	2.1 ระบบแสดงข้อมูล Environment บนหน้าจอและจัดเก็บข้อมูล Environment ลงฐานข้อมูลของระบบ
	3. ลบข้อมูล	
	1. Automated Test Developer เลือกข้อมูล Environment ที่ต้องการลบแล้วกดปุ่ม Delete	1.1 ระบบแสดงข้อความยืนยันการลบ
	2. Automated Test Developer ยืนยันการลบข้อมูล	2.1 ระบบจะลบข้อมูล Environment ออกจากหน้าจอและฐานข้อมูลของระบบ
Exception Conditions	ไม่มี	

ตารางที่ 4.5 รายละเอียดยูสเคสของการจัดการ Test Suite (Manage TestSuite)

Use Case Name	Manage TestSuite
Triggering Event	Automated Test Developer ต้องการเพิ่ม แก้ไขหรือลบข้อมูล Test Suite ที่ใช้ในการทดสอบ
Brief Description	สคริปท์อัตโนมัติที่ใช้ทดสอบ หรือเรียกว่า Test Suite นั้น จะถูกสร้างขึ้นตามวัตถุประสงค์ของการทดสอบที่ Manual Tester ร้องขอเข้ามา โดยจะแปลงมาจากขั้นตอนการทดสอบ (Test Plan) ที่ได้รับมาด้วย Automated Test Tools

ตารางที่ 4.5 (ต่อ)

Actor	Automated Test Developer	
Related Use Case	ไม่มี	
Preconditions	Automated Test Developer ต้องการจัดการข้อมูลของ Test Suite ที่สร้างขึ้นมาและต้องการทดสอบ 1.ในกรณีเพิ่มจะเป็นการเพิ่มข้อมูลของ Test Suite ที่ใช้ทดสอบเข้ามาจัดเก็บในระบบ 2. การแก้ไขเป็นการเรียกข้อมูล Test Suite ที่มีอยู่เดิมขึ้นมาเพื่อทำการปรับปรุงแล้วบันทึกลงในตำแหน่งเดิม 3. การลบเป็นการเรียกข้อมูล Test Suite ที่มีอยู่ออกไปจากฐานข้อมูลของระบบ	
Post conditions	Automated Test Developer เพิ่ม แก้ไขหรือลบข้อมูลได้ ข้อมูลของ Test Suite ที่ใช้ทดสอบถูกจัดเก็บอย่างถูกต้องและสามารถนำไปใช้ในการวางแผนการทดสอบต่อไปได้	
Flow of Activities	**Actor**	**System**
	1. เพิ่มข้อมูล 1. เข้าสู่เมนู Suite เพื่อจัดการข้อมูล Test Suite	1.1 ระบบแสดงหน้าจอ Suite โดยจะแสดงข้อมูล Test Suite ที่มีอยู่ในระบบ
	2. Automated Test Developer กดปุ่ม Add เพื่อทำการเพิ่มข้อมูล Test Suite	2.1 ระบบแสดงหน้าจอและฟอร์มการเพิ่มข้อมูล Test Suit
	3. Automated Test Developer กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save	3.1 ระบบแสดงข้อมูล Test Suite บนหน้าจอและจัดเก็บข้อมูล Test Suite ลงฐานข้อมูลของระบบ
	2. แก้ไขข้อมูล 1. Automated Test Developer เลือกข้อมูล Test Suite ที่ต้องการแก้ไขแล้วกดปุ่ม Update	1.1 ระบบแสดงหน้าจอและฟอร์มการแก้ไขข้อมูล Test Suite
	2. Automated Test Developer แก้ไขข้อมูลแล้วกด Save	2.1 ระบบแสดงข้อมูล Test Suite บนหน้าจอและจัดเก็บข้อมูล Test Suite ลงฐานข้อมูลของระบบ

ตารางที่ 4.5 (ต่อ)

Flow of Activities	Actor	System
	3. ลบข้อมูล 1. Automated Test Developer เลือกข้อมูล Test Suite ที่ต้องการลบแล้วกดปุ่ม Delete 2. Automated Test Developer ยืนยันการลบข้อมูล	1.1 ระบบแสดงข้อความยืนยันการลบ 2.1 ระบบจะลบข้อมูล Test Suite ออกจากหน้าจอและฐานข้อมูลของระบบ
Exception Conditions	ไม่มี	

ตารางที่ 4.6 รายละเอียดยูสเคสของการจัดการเครื่องเซิร์ฟเวอร์ (Manage Server)

Use Case Name	Manage Server
Triggering Event	Automated Test Developer ต้องการเพิ่ม แก้ไขหรือลบข้อมูลเซิร์ฟเวอร์ต่างๆที่เกี่ยวข้องในการทดสอบ
Brief Description	Environment ที่ใช้ในการทดสอบซอฟตแวร์แต่ละครั้งนั้น จะประกอบด้วย 2 ส่วน คือส่วนเซิร์ฟเวอร์และส่วนไคลแอนด์ โดยในส่วนของเซิร์ฟเวอร์สามารถแบ่งได้เป็น 3 ประเภทหลักๆ คือ Database Server, Engine Server และ Application Server เพื่อนำเซิร์ฟเวอร์ต่างๆ เหล่านี้ไปประกอบเป็น Environment ในการทดสอบ
Actor	Automated Test Developer
Related Use Case	ไม่มี
Preconditions	Automated Test Developer ต้องการจัดการเซิร์ฟเวอร์ต่างๆที่เกี่ยวข้องในการทดสอบ 1.ในกรณีเพิ่มจะเป็นการเพิ่มข้อมูลเซิร์ฟเวอร์ต่างๆที่เกี่ยวข้องในการทดสอบเข้ามาจัดเก็บในระบบ 2. การแก้ไขเป็นการเรียกข้อมูลเซิร์ฟเวอร์ต่างๆที่มีอยู่เดิมขึ้นมาเพื่อทำการปรับปรุงแล้วบันทึกลงในตำแหน่งเดิม 3. การลบเป็นการเรียกข้อมูลเซิร์ฟเวอร์ต่างๆที่มีอยู่ออกไปจากฐานข้อมูลของระบบ

ตารางที่ 4.6 (ต่อ)

Post conditions	Automated Test Developer เพิ่ม แก้ไขหรือลบข้อมูลได้ ข้อมูลเซิร์ฟเวอร์ต่างๆที่เกี่ยวข้องในการทดสอบถูกจัดเก็บอย่างถูกต้องและสามารถนำไปใช้ในการสร้าง Environment ของการทดสอบต่อไปได้	
Flow of Activities	**Actor**	**System**
	1. เพิ่มข้อมูล	
	1. เข้าสู่เมนู Database Server, Engine Server และ Application Server ตามลำดับ เพื่อจัดการข้อมูลเซิร์ฟเวอร์ต่างๆที่เกี่ยวข้องในการทดสอบ	1.1 ระบบแสดงหน้าจอ Server โดยจะแสดงข้อมูลเซิร์ฟเวอร์ต่างๆที่เกี่ยวข้องในการทดสอบที่มีอยู่ในระบบ
	2. Automated Test Developer กดปุ่ม Add เพื่อทำการเพิ่มข้อมูลเซิร์ฟเวอร์ต่างๆ	2.1 ระบบแสดงหน้าจอและฟอร์มการเพิ่มข้อมูลเซิร์ฟเวอร์ต่างๆ
	3. Automated Test Developer กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save	3.1 ระบบแสดงข้อมูลเซิร์ฟเวอร์ต่างๆบนหน้าจอและจัดเก็บข้อมูลเซิร์ฟเวอร์ต่างๆลงฐานข้อมูลของระบบ
	3. แก้ไขข้อมูล	
	1. Automated Test Developer เลือกข้อมูลที่ต้องการแก้ไขแล้วกดปุ่ม Update	1.1 ระบบแสดงหน้าจอและฟอร์มการแก้ไขข้อมูลเซิร์ฟเวอร์ต่างๆ
	2. Automated Test Developer แก้ไขข้อมูลแล้วกด Save	2.1 ระบบแสดงข้อมูลเซิร์ฟเวอร์ต่างๆบนหน้าจอและจัดเก็บข้อมูลเซิร์ฟเวอร์ต่างๆลงฐานข้อมูลของระบบ
	4. ลบข้อมูล	
	1. Automated Test Developer เลือกข้อมูลที่ต้องการลบแล้วกดปุ่ม Delete	1.1 ระบบแสดงข้อความเตือน
	2. Automated Test Developer ยืนยันการลบข้อมูล	2.1 ระบบจะลบข้อมูลออกจากหน้าจอและฐานข้อมูล

ตารางที่ 4.6 (ต่อ)

Exception Conditions	ไม่มี

ตารางที่ 4.7 รายละเอียดยูสเคสของการวางแผนการทดสอบ (Planning Test)

Use Case Name	Planning Test	
Triggering Event	Automated Test Executor ต้องการวางแผนการทดสอบโดยการเพิ่มหรือแก้ไขหรือลบข้อมูลการทดสอบ	
Brief Description	ในการทดสอบแต่ละครั้งจะต้องทำการวางแผนการทดสอบทุกครั้ง การวางแผนการทดสอบเสมือนเป็นการจับจอง Environment ต่างๆ เช่น เครื่องทดสอบ และDatabase Slot เป็นต้น รวมถึงเป็นการแจ้งให้ผู้ร่วมทดสอบทราบถึง Environment ที่เหลืออยู่เพื่อจะได้ไม่ใช้งานซ้อนกัน	
Actor	Automated Test Executor	
Related Use Case	ไม่มี	
Preconditions	Automated Test Executor ต้องการวางแผนการทดสอบ 1.ในกรณีเพิ่มจะเป็นการเพิ่มข้อมูลแผนการทดสอบเข้ามาจัดเก็บ 2. การแก้ไขเป็นการเรียกแผนการทดสอบเดิมที่มีอยู่ขึ้นมาเพื่อทำการปรับปรุงแล้วบันทึกลงในตำแหน่งเดิม 3. การลบเป็นการเรียกแผนการทดสอบที่มีอยู่ออกไปจากฐานข้อมูลของระบบ	
Post conditions	Automated Test Executor เพิ่ม แก้ไขหรือลบข้อมูลได้ ข้อมูลแผนการทดสอบถูกจัดเก็บอย่างถูกต้องและสามารถนำข้อมูลไปใช้ในการ Execute Test ต่อไปได้	
Flow of Activities	**Actor**	**System**
	1. เข้าสู่เมนู Tools ไปที่ Planning เพื่อวางแผนการทดสอบ **1. เพิ่มข้อมูล** 1. Automated Test Executor กดปุ่ม Add เพื่อทำการเพิ่มแผนรายการที่จะทดสอบ	1.1 ระบบแสดงหน้าจอ Planning พร้อมด้วยข้อมูลแผนการทดสอบที่มีอยู่ในฐานข้อมูลของระบบ 1.1 ระบบแสดงหน้าจอและฟอร์มการเพิ่มข้อมูลแผนรายการที่จะทดสอบ

ตารางที่ 4.7 (ต่อ)

Flow of Activities	Actor	System
	2. Automated Test Executor กรอกรายละเอียดต่างๆตามเงื่อนไขให้ครบแล้วกด Save	2.1 ระบบแสดงรายการของแผนการทดสอบบนหน้าจอและจัดเก็บข้อมูลรายการที่จะทดสอบลงฐานข้อมูลของระบบ
	2. แก้ไขข้อมูล	
	1. Automated Test Executor เลือกข้อมูลที่ต้องการแก้ไขแล้วกดปุ่ม Update	1.1 ระบบแสดงหน้าจอและฟอร์มการแก้ไขข้อมูล
	2. Automated Test Executor แก้ไขข้อมูลแล้วกด Save	2.1 ระบบแสดงข้อมูลบนหน้าจอและจัดเก็บข้อมูลลงฐานข้อมูล
	3. ลบข้อมูล	
	1. Automated Test Developer เลือกข้อมูลแผนการทดสอบที่ต้องการลบแล้วกดปุ่ม Delete	1.1 ระบบแสดงข้อความยืนยันการลบ
	2. Automated Test Developer ยืนยันการลบข้อมูล	2.1 ระบบจะลบข้อมูลแผนการทดสอบออกจากหน้าจอและฐานข้อมูลของระบบ
Exception Conditions	ไม่มี	

ตารางที่ 4.8 รายละเอียดยูสเคสของการตรวจสอบพื้นที่ที่ใช้ในการทดสอบ (Check Disk Space)

Use Case Name	Check Disk Space
Triggering Event	Automated Test Executor ทำการตรวจสอบขนาดพื้นที่ที่ใช้ในการทดสอบ
Brief Description	ในการทดสอบแต่ละครั้งจำเป็นต้องใช้พื้นที่บนเซิร์ฟเวอร์ เพื่อแตกไฟล์ของฐานข้อมูล และโหลดข้อมูลเข้าสู่ Database Slot ดังนั้น Automated Test Executor จำเป็นต้องตรวจสอบพื้นที่บนเซิร์ฟเวอร์ทุกครั้งก่อนเริ่มทำการทดสอบ
Actor	Automated Test Executor
Related Use Case	ไม่มี

ตารางที่ 4.8 (ต่อ)

Preconditions	ไม่มี	
Post conditions	ระบบจะทำการตรวจสอบพื้นที่ โดยจะแสดงขนาดและเปอร์เซ็นต์ของพื้นที่ที่เหลืออยู่	
Flow of Activities	**Actor**	**System**
	1. Automated Test Executor เข้าสู่เมนู Tools -> Setting เพื่อทำการเลือก Directory ที่ต้องการตรวจสอบและกำหนดขนาดของพื้นที่	1.1 ระบบแสดงหน้าจอ Setting พร้อมแสดง Directory ทั้งหมดที่เครื่องทดสอบนั้นๆ ได้ทำการ Mapping ไว้
	2. Automated Test Executor ทำการเลือก Directory และกำหนด % ของพื้นที่ที่ยอมรับในการทดสอบได้ แล้วทำการ save	2.1 ระบบทำการบันทึกข้อมูลใหม่และจะทำการปิดหน้าจอ Setting
	3. Automated Test Executor เข้าสู่เมนู Tools -> Disk Space Checking เพื่อทำการตรวจสอบขนาดของพื้นที่บนเซิร์ฟเวอร์ตามที่ได้ Map Drive ไว้	3.1 ระบบจะทำการตรวจสอบขนาดของพื้นที่บนเซิร์ฟเวอร์และแสดงขนาดของพื้นที่ที่เหลือบนหน้าจอ
Exception Conditions	1.1 กรณีที่ไม่มี Drive ที่ระบุไว้ ระบบจะแสดงข้อความเตือนว่าไม่มี Drive นั้นอยู่ 1.2 กรณีที่พื้นที่บนเซิร์ฟเวอร์มีขนาดไม่เพียงกับการทดสอบ ระบบจะแสดงข้อความเตือน และไม่สามารถเริ่มการทดสอบได้	

ตารางที่ 4.9 รายละเอียดยูสเคสของการตรวจสอบ Database Slot (Check Database Slot)

Use Case Name	Check Database Slot
Triggering Event	Automated Test Executor ต้องการรู้ว่า Database Slot ตัวไหนว่างหรือถูกใช้งานอยู่
Brief Description	การทดสอบ Test Suite ในแต่ละ Environment จะมีการโหลดชุดข้อมูลตั้งต้น (Master Dump) เข้าสู่ DB Slot ดังนั้น Automated Test Executor จะต้องตรวจสอบ DB Slot ก่อนทำการโหลด หากไม่ตรวจสอบและโหลดทับตัวที่ใช้งานอยู่จะทำให้ Database Slot นั้นเสียหายได้

ตารางที่ 4.9 (ต่อ)

Actor	Automated Test Executor	
Related Use Case	ไม่มี	
Preconditions	ไม่มี	
Post conditions	Automated Test Executor ทราบสถานะของ Database Slot ในระบบทั้งหมดที่มีอยู่ตามแต่ละ CFG	
Flow of Activities	**Actor**	**System**
	1. Automated Test Executor เข้าสู่เมนู Tools -> Database Checking เพื่อตรวจสอบสถานะของ Database Slot 2. Automated Test Executor เลือก CFG ของ Database Slot ที่ต้องการตรวจสอบ	1.1 ระบบแสดงหน้าจอการตรวจสอบสถานะของ Database Slot 2.1 ระบบทำการตรวจสอบและแสดงสถานะของ Database Slot ตาม CFG ที่ได้เลือกไว้
Exception Conditions	2.1 เครื่องทดสอบที่จะสามารถตรวจสอบสถานะของ Database Slot ได้นั้นจะต้องมีการกำหนดค่า ODBC ของ CFG นั้นๆไว้	

ตารางที่ 4.10 รายละเอียดยูสเคสของการเตรียมไฟล์ที่เกี่ยวข้องกับการทดสอบ (Setup Configuration File)

Use Case Name	Setup Configuration File
Triggering Event	Automated Test Executor ต้องการเริ่มทำการทดสอบ
Brief Description	เมื่อ Automated Test Executor ได้รับการร้องขอให้ทำการทดสอบแอพพลิเคชัน Automated Test Executor จะทำการตรวจสอบสภาพแวดล้อม และสร้างไฟล์ต่างๆที่จำเป็นต้องใช้ในการทดสอบ
Actor	Automated Test Executor
Related Use Case	ไม่มี
Preconditions	ได้รับการร้องขอให้ทำการทดสอบ
Post conditions	ไฟล์ต่างๆจะถูกสร้างขึ้นและจะถูกเรียกใช้โดยระบบ

ตารางที่ 4.10 (ต่อ)

Flow of Activities	Actor	System
	1. Automated Test Executor ทำการตรวจสอบ Environment ของการทดสอบว่าพร้อมหรือไม่ โดยเข้าสู่เมนู Tools -> Setting 2. Automated Test Executor เตรียมไฟล์ต่างๆที่เกี่ยวข้องแล้วกด Save	1.1 ระบบแสดงหน้าจอ Setting ค่าต่างๆ ที่จำเป็นในระบบ ซึ่งจะแสดงค่า Default ไว้สามารถเปลี่ยนแปลงค่าได้ 2.1 ระบบทำการบันทึกข้อมูล Setting ลงฐานข้อมูลของระบบ
Exception Conditions	2.1 กรณีที่ข้อมูลต่างๆไม่ครบถ้วน ระบบจะทำการแจ้งเตือนและไม่บันทึกข้อมูลจนกว่าข้อมูลจะถูกต้อง	

ตารางที่ 4.11 รายละเอียดยูสเคสการตั้งเวลาการทดสอบ (Schedule Runs)

Use Case Name	Schedule Runs
Triggering Event	Automated Test Executor ตั้งเวลาการทดสอบ
Brief Description	Automated Test Executor สามารถตั้งเวลาการทดสอบไว้ล่วงหน้าได้ ซึ่งมีให้เลือก 3 แบบ ได้แก่ 1. **Start Now** คือ รูปแบบที่ระบบจะทำการทดสอบทันที รูปแบบนี้เหมาะสำหรับมีสภาพแวดล้อมของการทดสอบทุกอย่างพร้อมแล้ว สามารถเริ่มทดสอบได้ทันที 2. **Schedule AutoDump and Execution** คือ รูปแบบที่สามารถตั้งเวลาการทดสอบล่วงหน้า เมื่อถึงเวลาที่ได้กำหนดไว้ ระบบจะทำการตรวจสอบเวอร์ชันและวันที่ของ Build และ Client Install Shield ที่ต้องการทดสอบ จากนั้นทำการจะโหลดฐานข้อมูล (Load DB) และปรับปรุงฐานข้อมูล (Update DB) รูปแบบนี้เหมาะสำหรับต้องการให้ระบบโหลดฐานข้อมูล (Load DB) ปรับปรุงฐานข้อมูล (Update DB) และทดสอบให้อย่างอัตโนมัติ 3. **Schedule Execution** คือ รูปแบบที่สามารถตั้งเวลาล่วงหน้าและเริ่มทดสอบเมื่อถึงเวลาที่กำหนดไว้ รูปแบบนี้เหมาะสำหรับมีสภาพแวดล้อมที่ต้องการบางส่วน แต่ยังต้องรอ Build และ Client Install Shield ที่ต้องการ

ตารางที่ 4.11 (ต่อ)

Brief Description	4. **Schedule AutoDump and Execution** คือ รูปแบบที่สามารถตั้งเวลาการทดสอบล่วงหน้า เมื่อถึงเวลาที่ได้กำหนดไว้ ระบบจะทำการตรวจสอบเวอร์ชันและวันที่ของ Build และ Client Install Shield ที่ต้องการทดสอบ จากนั้นทำการจะโหลดฐานข้อมูล (Load DB) และปรับปรุงฐานข้อมูล (Update DB) รูปแบบนี้เหมาะสำหรับต้องการให้ระบบโหลดฐานข้อมูล (Load DB) ปรับปรุงฐานข้อมูล (Update DB) และทดสอบให้อย่างอัตโนมัติ 5. **Schedule Execution** คือ รูปแบบที่สามารถตั้งเวลาล่วงหน้าและเริ่มทดสอบเมื่อถึงเวลาที่กำหนดไว้ รูปแบบนี้เหมาะสำหรับมีสภาพแวดล้อมที่ต้องการบางส่วน แต่ยังต้องรอ Build และ Client Install Shield ที่ต้องการ	
Actor	Automated Test Executor	
Related Use Case	Setup Environment, Run Tests, Check Disk Space	
Preconditions	ก่อนตั้งเวลาการทดสอบจะต้องมีการเตรียมไฟล์ต่างๆที่จำเป็นครบถ้วน	
Post conditions	ระบบทำงานตามเงื่อนไขที่ได้เลือกไว้	
Flow of Activities	**Actor**	**System**
	1. Automated Test Executor ทำการบันทึกแผนการทดสอบออกเป็นไฟล์ Excel โดยใช้ปุ่ม SavePlan 2. Automated Test Executor ทำการเลือกรูปแบบการตั้งเวลาของระบบ	1.1 ระบบทำการ Export รายการแผนการทดสอบที่ได้วางแผนไว้ออกมาเป็นไฟล์ Excel ในรูปแบบที่ถูกต้อง 2.1 ระบบทำงานตามเงื่อนไขและรูปแบบที่ได้ถูกเลือกไว้
Exception Conditions	1.1 กรณีที่ไม่ได้ทำการเลือกรูปแบบ ระบบจะกำหนดไว้ที่ Start Now	

ตารางที่ 4.12 รายละเอียดยูสเคส Start Now

Use Case Name	Start Now
Triggering Event	Automated Test Executor ต้องการตั้งเวลาทดสอบแบบ Start Now
Brief Description	Automated Test Executor เลือกการตั้งเวลาแบบ Start Now เพื่อต้องการให้ระบบทำงานทันที

ตารางที่ 4.12 (ต่อ)

Actor	Automated Test Executor	
Related Use Case	**Extend:** Schedule Runs	
Preconditions	Automated Test Executor เตรียมไฟล์ต่างๆที่เกี่ยวข้องให้พร้อมก่อน ทำการตั้งเวลา	
Post conditions	ระบบทำการทดสอบตามเวลาที่ได้กำหนดไว้	
Flow of Activities	**Actor**	**System**
	1. หลังจาก Automated Test Executor ตั้ง schedule run โดยเลือกการทดสอบแบบ Start Now แล้วกด SingleClick	1.1 ระบบตรวจสอบองค์ประกอบต่างๆ ที่เกี่ยวข้องกับการทดสอบว่าพร้อมหรือไม่ ดังนี้ – ตรวจสอบพื้นที่ที่ใช้ในการทดสอบว่าเพียงพอหรือไม่ – ตรวจสอบ Directory ต่างๆ ที่เกี่ยวข้อง ตรวจสอบ Database Slot ว่า in-used หรือไม่ 1.2 เมื่อองค์ประกอบต่างๆพร้อมแล้ว ระบบจะเริ่มทำงานทันทีด้วยการตรวจสอบวันที่และเวอร์ชันของซอฟต์แวร์ก่อนเป็นอันดับแรก กรณีที่มีซอฟต์แวร์ที่จะทดสอบแล้วนั้น ระบบจะแบ่งการทำงานออกเป็น 2 ส่วน คือ โหลดข้อมูลเข้าสู่ Database slot และติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ
Exception Conditions	ไม่มี	

ตารางที่ 4.13 รายละเอียดยูสเคส Schedule AutoDump and Execute

Use Case Name	Schedule AutoDump and Execute
Triggering Event	Automated Test Executor ต้องการตั้งเวลาทดสอบแบบ Schedule AutoDump and Execute

ตารางที่ 4.13 (ต่อ)

Brief Description	Automated Test Executor เลือกการตั้งเวลาแบบ Schedule AutoDump and Execute เพื่อต้องการให้ระบบทำ Dump แบบอัตโนมัติ และทดสอบซอฟต์แวร์ตามเวลาที่ได้ระบุไว้ โดยให้ระบบเริ่มทำงานตามเวลาที่ได้กำหนดไว้	
Actor	Automated Test Executor	
Related Use Case	Extend: Schedule Runs	
Preconditions	Automated Test Executor จะต้องทำการเตรียมไฟล์ต่างๆที่เกี่ยวข้องให้พร้อมก่อน ทำการตั้งเวลาทดสอบ	
Post conditions	ระบบทำการทดสอบตามเวลาที่ได้กำหนดไว้	
Flow of Activities	**Actor**	**System**
	1. หลังจาก Automated Test Executor ตั้ง schedule run โดยเลือกการทดสอบแบบ Schedule AutoDump and Execute แล้วกด SingleClick	1.1 ระบบตรวจสอบองค์ประกอบต่างๆ ที่เกี่ยวข้องกับการทดสอบว่าพร้อมหรือไม่ ดังนี้ – ตรวจสอบพื้นที่ที่ใช้ในการทดสอบว่าเพียงพอหรือไม่ – ตรวจสอบ Directory ต่างๆ ที่เกี่ยวข้อง – ตรวจสอบ Database Slot ว่า in-used หรือไม่ 1.2 เมื่อองค์ประกอบต่างๆพร้อมแล้ว ระบบจะรอให้ถึงเวลาที่ Automated Test Executor ได้กำหนดไว้ เมื่อถึงเวลาที่กำหนดไว้ ระบบจะเริ่มทำงานด้วยการตรวจสอบวันที่และเวอร์ชันของซอฟต์แวร์ก่อนเป็นอันดับแรก

ตารางที่ 4.13 (ต่อ)

Flow of Activities	Actor	System
		กรณีที่มีซอฟต์แวร์ที่จะทดสอบแล้วนั้น ระบบจะแบ่งการทำงานออกเป็น 2 ส่วน คือ โหลดข้อมูลเข้าสู่ Database slot และ ติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ
Exception Conditions	ไม่มี	

ตารางที่ 4.14 รายละเอียดยูสเคส Schedule Execution

Use Case Name	Schedule Execution	
Triggering Event	Automated Test Executor ต้องการตั้งเวลาทดสอบแบบ Schedule Execution	
Brief Description	Automated Test Executor เลือกการตั้งเวลาแบบ Schedule Execution โดยรูปแบบนี้จะไม่ทำการโหลดข้อมูลเข้า Database Slot แต่จะทำการติดตั้งซอฟต์แวร์ที่เครื่องทดสอบและทดสอบให้ตามเวลาที่ได้กำหนดไว้	
Actor	Automated Test Executor	
Related Use Case	Extend: Schedule Runs	
Preconditions	Automated Test Executor มีไฟล์ต่างๆที่เกี่ยวข้องและ Database Slot มีข้อมูลอยู่แล้ว เหลือเพียงรอเวลาทดสอบอย่างเดียว	
Post conditions	ระบบทำการทดสอบตามเวลาที่ได้กำหนดไว้	
Flow of Activities	**Actor**	**System**
	1. หลังจาก Automated Test Executor ตั้ง schedule run โดยเลือกการทดสอบแบบ Schedule Execution แล้วกด SingleClick	1.1 ระบบตรวจสอบองค์ประกอบต่างๆ ที่เกี่ยวข้องกับการทดสอบว่าพร้อมหรือไม่ ดังนี้ – ตรวจสอบพื้นที่ที่ใช้ในการทดสอบว่าเพียงพอหรือไม่ – ตรวจสอบ Directory ต่างๆ ที่เกี่ยวข้อง – ตรวจสอบ Database Slot ว่า in-used หรือไม่

ตารางที่ 4.14 (ต่อ)

Flow of Activities	Actor	System
		1.2 เมื่อองค์ประกอบต่างๆพร้อมแล้ว ระบบจะรอให้ถึงเวลาที่ Automated Test Executor ได้กำหนดไว้ เมื่อถึงเวลาที่กำหนดไว้ ระบบจะเริ่มทำงานด้วยการตรวจสอบวันที่และเวอร์ชันของซอฟต์แวร์ก่อนเป็นอันดับแรก กรณีที่มีซอฟต์แวร์ที่จะทดสอบแล้วนั้น ระบบจะทำการติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ และดำเนินการทดสอบ
Exception Conditions	ไม่มี	

ตารางที่ 4.15 รายละเอียดยูสเคสของการทดสอบ (Execute Test)

Use Case Name	Execute Tests
Triggering Event	ระบบเริ่มทำงานตามเงื่อนไขต่างๆ เมื่อถึงเวลาที่ได้กำหนดไว้
Brief Description	เมื่อถึงเวลาที่ได้กำหนดไว้ ระบบเริ่มทำงานตามเงื่อนไขต่างๆ ของรูปแบบที่ได้เลือกไว้ ดังนี้ 1. แบบ Start Now - ระบบจะทำงานทันที 2. แบบ Schedule AutoDump and Execution - ระบบจะรอจนถึงเวลาที่ได้กำหนดไว้ เมื่อถึงเวลาระบบจะตรวจสอบวันที่เวอร์ชันและBuild และ Client Install Shield จากนั้นจะทำ AutoDump ตามแผนที่ได้วางไว้ 3. แบบ Schedule Execution - ระบบจะรอจนถึงเวลาที่ได้กำหนดไว้ เมื่อถึงเวลาระบบจะตรวจสอบวันที่เวอร์ชันและวันที่ของ Build และติดตั้งซอฟต์แวร์ที่เครื่องทดสอบจากนั้นจึงเริ่มดำเนินการทดสอบ
Actor	Schedule Runs
Related Use Case	**Include:** Schedule Runs
Preconditions	Automated Test Executor เลือกรูปแบบการทำงานของระบบ

ตารางที่ 4.15 (ต่อ)

Post conditions	เมื่อถึงเวลาที่ได้กำหนดไว้ ระบบจะติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ จากนั้นจึงเริ่มดำเนินการทดสอบ	
Flow of Activities	**Actor**	**System**
	1. ระบบได้ถูก Automated Test Executor เลือกรูปแบบและระบุเงื่อนไขการทำงาน พร้อมทั้งกดปุ่ม SingleClick	1.1 ระบบทำการตรวจสอบรูปแบบการทำงานและเงื่อนไขที่ได้ระบุไว้ โดยระบบจะทำงานดังนี้ 1.2 ตรวจสอบชนิดของรูปแบบและตรวจสอบเวลา 1.3 ระบบตรวจสอบว่าวันที่และเวอร์ชันของ Build และ Client Install Shield ว่ามีที่ต้องการหรือไม่ 1.4 ระบบทำการถอนซอฟต์แวร์เวอร์ชันที่มีอยู่ก่อนหน้า 1.5 ติดตั้งซอฟต์แวร์ที่จะทดสอบที่เครื่องทดสอบที่ได้กำหนดไว้ 1.6 RunAgent ที่เครื่องทดสอบแต่ละเครื่องจะถูกปลุกให้ทำงานอยู่เบื้องหลัง (Background Mode) 1.7 เริ่มทำการทดสอบ 1.8 RunAgent แต่ละเครื่องจะทำงานตาม Suite ที่ได้ระบุไว้ 1.9 RunAgent เรียก Automated Test Scripts ของแต่ละ Suite ขึ้นมาทำงาน 1.10 ส่งเมลแจ้งไปยังเมล์กลาง เมื่อแจ้งผลการทดสอบ
Exception Conditions	1.3 กรณีที่ระบบตรวจสอบวันที่และเวอร์ชันของ Build และ Client Install Shield และไม่พบ ระบบจะทำการตรวจสอบอีกครั้งตามที่ได้ระบุไว้และจะยกเลิกการค้นหาเมื่อหมดเวลา 1.9 กรณีถึงเวลา Reboot ของเครื่องเซิร์ฟเวอร์ Test Suite ที่กำลังจะถูกทดสอบจะถูกหยุดไว้	

ตารางที่ 4.16 รายละเอียดยูสเคสของการเก็บผลการทดสอบ (Collect Result)

Use Case Name	Collect Result	
Triggering Event	Automated Test Executor ต้องการบันทึกผลการทดสอบ	
Brief Description	เมื่อการทดสอบสิ้นสุดลง Automated Test Executor จะต้องทำการเก็บผลและบันทึกผลการทดสอบ	
Actor	Automated Test Executor	
Related Use Case	ไม่มี	
Preconditions	สิ้นสุดการทดสอบและได้รับอีเมลแจ้งผลการทดสอบเบื้องต้น	
Post conditions	มีการระบุ Status และ/หรือ Cause of failure ของปัญหา (ถ้ามี)	
Flow of Activities	**Actor**	**System**
	1. Automated Test Executor เข้าเมนู Tools - Planning เพื่อทำการบันทึกผลการทดสอบและอัพเดต Status ของ Test Suite ที่แผนทดสอบ	1.1 ระบบแสดงหน้าจอวางแผนการทดสอบ พร้อมทั้งรายการทดสอบที่ได้วางแผนไว้
	2. Automated Test Executor ทำการบันทึกผลการทดสอบและกด Save	2.1 ระบบทำการบันทึกผลการทดสอบลงฐานข้อมูลของระบบ
Exception Conditions	ไม่มี	

ตารางที่ 4.17 รายละเอียดยูสเคสของการออกรายงานผลการทดสอบ (Generate Result)

Use Case Name	Generate Result
Triggering Event	Test Operator หรือ Automated Test Executor ต้องการออกรายงานผลการทดสอบ
Brief Description	เมื่อสิ้นสุดการทดสอบในแต่ละเวอร์ชัน Test Operator จะต้องทำการรายงานผลการทดสอบให้กับผู้บริหารหรือผู้ที่เกี่ยวข้องทราบถึงผลการทดสอบ
Actor	Test Operator , Automated Test Executor
Related Use Case	ไม่มี
Preconditions	Automated Test Executor ทำการอัพเดตสถานะของผลการทดสอบเรียบร้อย

ตารางที่ 4.17 (ต่อ)

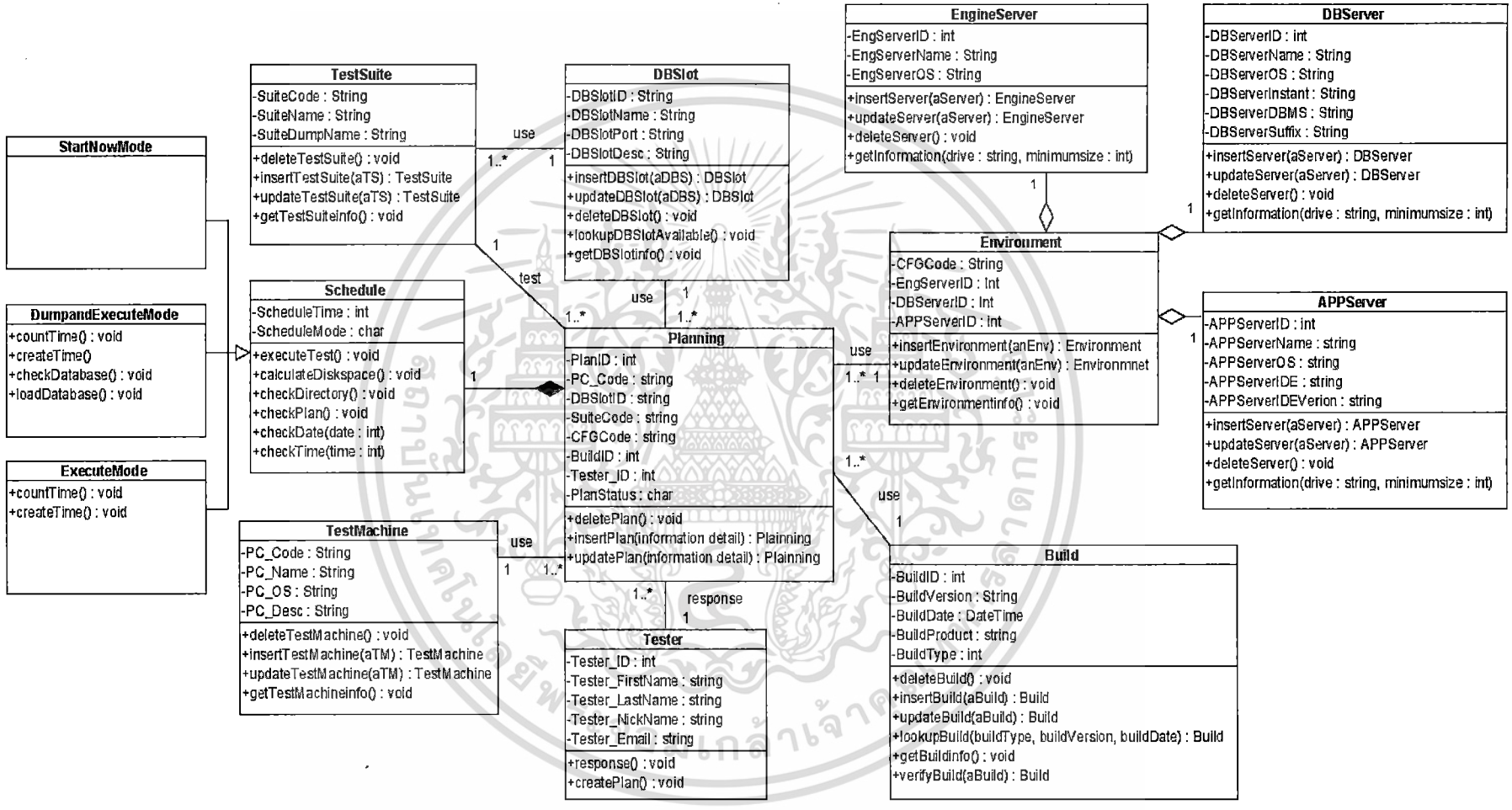
Post conditions	ได้รายงานผลการทดสอบส่งให้กับผู้บริหารและผู้ที่เกี่ยวข้องอื่นๆ	
Flow of Activities	**Actor**	**System**
	1. Test Operator เข้าเมนู Tools – Planning เพื่อทำการตรวจสอบสถานะของ Test Suite ที่ต้องการออกรายงานผลการทดสอบ 2. Test Operator หรือ Automated Test Executor ทำการออกรายงานผลการทดสอบด้วยการกดปุ่ม Save Plan	1.1 ระบบแสดงหน้าจอวางแผนการทดสอบ 2.1 ระบบทำการ Generate ไฟล์ออกมาเป็นไฟล์ Excel พร้อมทั้งข้อมูล Test Suite ที่ได้เลือกไว้
Exception Conditions	ไม่มี	

4.3.2 การออกแบบคลาสไดอะแกรม

ในขั้นตอนการวิเคราะห์การทำงานของระบบปัจจุบัน ได้ใช้ยูสเคสไดอะแกรม เพื่อแสดงฟังก์ชันการทำงานหลักของระบบ และแสดงถึงความสัมพันธ์ระหว่างแอกเตอร์กับยูสเคส ผลที่ได้จากการวิเคราะห์จะนำมาใช้ในการสร้างคลาสไดอะแกรม เพื่อใช้แสดงโครงสร้างของวัตถุที่ระบบสนใจ

การศึกษาวิเคราะห์และออกแบบแอพพลิเคชันสำหรับการเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบสามารถสร้างคลาสไดอะแกรมได้ดังรูปที่ 4.3 ซึ่งสามารถอธิบายส่วนประกอบแต่ละคลาสได้ดังนี้

1. Planning หมายถึง คลาสของการวางแผนการทดสอบ
2. TestSuite หมายถึง คลาสของสคริปท์ที่จะทดสอบ
3. Tester หมายถึง คลาสของผู้รับผิดชอบทำการทดสอบในครั้งนั้นๆ
4. DBSlot หมายถึง คลาสของ Database Slot
5. TestMachine หมายถึง คลาสของเครื่องที่ใช้ทดสอบ
6. StartNowMode หมายถึง คลาสของรูปแบบการทดสอบ โดยเป็นรูปแบบของการเริ่มทำการทดสอบทันที
7. DumpandExecutionMode หมายถึง คลาสของรูปแบบการทดสอบ โดยรอให้ถึงเวลาทดสอบและโหลดฐานข้อมูลเข้า DBSlot
8. ExecuteMode หมายถึง คลาสของรูปแบบการทดสอบ โดยสามารถตั้งเวลาทดสอบและทำการทดสอบทันทีเมื่อถึงเวลาที่กำหนด



รูปที่ 4.3 คลาสไดอะแกรมแอพพลิเคชันสำหรับการเตรียมสภาพแวดล้อมและตั้งเวลาทดสอบ

9. Schedule หมายถึง คลาสของการตั้งเวลาทดสอบ
10. Build หมายถึง คลาสของซอฟต์แวร์ที่จะใช้ทดสอบ
11. ServerBuild หมายถึง คลาสของซอฟต์แวร์ที่ไว้ติดตั้งที่เครื่องเซิร์ฟเวอร์
12. ClientBuild หมายถึง คลาสของซอฟต์แวร์ที่ไว้ติดตั้งที่เครื่องไคล์แอนด์
13. Environment หมายถึง คลาสของสภาพแวดล้อมที่จะทำการทดสอบ เช่นทดสอบบนสภาพแวดล้อมที่เป็น Unix , Sybase 15.0.4 และ WebLogic
14. EngineServer หมายถึง คลาสของเครื่องเซิร์ฟเวอร์ที่ระบุระบบปฏิบัติในการทดสอบ เช่น Unix หรือ Windows
15. APPServer หมายถึง คลาสของเครื่องเซิร์ฟเวอร์ที่ระบุว่าเป็น Weblogic หรือ Websphere
16. DBServer หมายถึง คลาสของเครื่องเซิร์ฟเวอร์ที่DBMS ต่างๆ เช่น Sybase หรือ Oracle เป็นต้น

การอธิบายคลาสเพื่อทำให้เข้าใจถึงการทำงานและหน้าที่ความรับผิดชอบของแต่ละคลาสจึงได้จัดทำตารางความรับผิดชอบและการร่วมมือของคลาสด้วย Class Responsibility Collaborator CRC คือการกำหนดความร่วมมือของคลาสหนึ่งๆกับ คลาสหนึ่งๆ เพื่อให้งานลุล่วงไปด้วยดี โดยเทคนิคคิดค้นโดยโดย CRC จะแบ่งออกเป็น 3 กลุ่มตามStandard Index ได้แก่

1. คลาส (Class)
2. ภาระหน้าที่ของคลาส (Responsibility) เป็นสิ่งต่างๆที่คลาสทราบ หรือสิ่งที่คลาสกระทำ
3. ความร่วมมือ (Collaborator) เป็นคลาสที่คอยรับข้อมูล หรือการทำงานบางอย่างของตารางที่ 4.18 ถึง 4.31

ตารางที่ 4.18 ตารางความรับผิดชอบและการร่วมมือของคลาส Planning

Planning	
Super Class: N/A	
Sub Class: N/A	
Description: คลาสของการวางแผนการทดสอบ	
Attributes:	
Name	**Collaborator**
PlanID	รหัสของแผนทดสอบ
PlanStatus	สถานะของแผนทดสอบ
PC_Code	รหัสเครื่องทดสอบ

ตารางที่ 4.18 (ต่อ)

Name	Collaborator
DBSlotID	รหัส Database Slot
SuiteCode	รหัสสคริปท์ที่ใช้ทดสอบ
CFGCode	รหัส Environment
TESTER_ID	รหัสผู้รับผิดชอบ
BuildID	รหัสซอฟต์แวร์ที่ใช้ทดสอบ
Responsibilities:	
Name	**Collaborator**
DeletePlan	TestMachine , Build , TestSuite , DBSlot , Environment ,
insertPlan	Tester
updatePlan	

ตารางที่ 4.19 ตารางความรับผิดชอบและการร่วมมือของคลาส TestSuite

TestSuite	
Super Class: N/A	
Sub Class: N/A	
Description: คลาสของสคริปท์ที่จะทดสอบ	
Attributes:	
Name	**Collaborator**
SuiteCode	รหัสของ Test Suite
SuiteName	ชื่อของ Test Suite
SuiteDumpName	ชื่อ dump ของ Test Suite
Responsibilities:	
Name	**Collaborator**
deleteTestSuite	Planning
insertTestSuite	
updateTestSuite	
getTestSuiteinfo	

ตารางที่ 4.20 ตารางความรับผิดชอบและการร่วมมือของคลาส Tester

Tester
Super Class: N/A

ตารางที่ 4.20 (ต่อ)

Sub Class: N/A	
Description: คลาสของผู้รับผิดชอบการทดสอบ	
Attributes:	
Name	**Collaborator**
Tester_ID	รหัสของผู้รับผิดชอบการทดสอบ
Tester_Name	ชื่อของผู้รับผิดชอบการทดสอบ
Tester_Email	อีเมลของผู้รับผิดชอบการทดสอบ
Responsibilities:	
Name	**Collaborator**
response	Planning
createPlan	

ตารางที่ 4.21 ตารางความรับผิดชอบและการร่วมมือของคลาส DBSlot

DBSlot	
Super Class: N/A	
Sub Class: N/A	
Description: คลาสของ Database Slot มีหน้าที่เก็บข้อมูลที่ใช้ในการทดสอบ	
Attributes:	
Name	**Collaborator**
DBSlotID	รหัสของ DB Slot
DBSlotName	ชื่อของ DB Slot
DBSlotPort	เลขพอร์ตของ DB Slot
DBSlotDesc	รายละเอียดของ DB Slot
Responsibilities:	
Name	**Collaborator**
insertDBSlot	Planning
updateDBSlot	
deleteDBSlot	
getDBSlotinfo	
lookupDBSlotAvailable	

ตารางที่ 4.22 ตารางความรับผิดชอบและการร่วมมือของคลาส StartNowMode

StartNowMode	
Super Class: Schedule	
Sub Class: N/A	
Description: คลาสของรูปแบบการทดสอบ โดยเป็นรูปแบบของการเริ่มทำการทดสอบทันที	
Attributes:	
Name	**Description**
Responsibilities:	
Name	**Collaborator**
	Schedule

ตารางที่ 4.23 ตารางความรับผิดชอบและการร่วมมือของคลาส TestMachine

TestMachine	
Super Class: N/A	
Sub Class: N/A	
Description: คลาสของเครื่องที่เกี่ยวข้องกับการทดสอบ	
Attributes:	
Name	**Collaborator**
PC_Code	รหัสของเครื่องทดสอบ
PC_Name	ชื่อของเครื่องทดสอบ
PC_OS	ระบบปฏิบัติการของเครื่องทดสอบ
PC_Desc	หน่วยประมวลผลของเครื่องทดสอบ
Responsibilities:	
Name	**Collaborator**
insertTestMachine	Planning
updateTestMachine	
deleteTestMachine	
getTestMachineinfo	

ตารางที่ 4.24 ตารางความรับผิดชอบและการร่วมมือของคลาส DumpandExecuteMode

DumpandExecuteMode	
Super Class: Schedule	
Sub Class: N/A	
Description: คลาสของรูปแบบการทดสอบ โดยรอให้ถึงเวลาทดสอบและโหลดฐานข้อมูลเข้า DBSlot	
Attributes:	
Name	**Description**
Responsibilities:	
Name	**Collaborator**
countTime createTime checkDatabase loadDatabase	Schedule

ตารางที่ 4.25 ตารางความรับผิดชอบและการร่วมมือของคลาส ExecuteMode

ExecuteMode	
Super Class: Schedule	
Sub Class: N/A	
Description: คลาสของรูปแบบการทดสอบ โดยสามารถตั้งเวลาทดสอบและทำการทดสอบทันทีเมื่อถึงเวลาที่กำหนด	
Attributes:	
Name	**Description**
Responsibilities:	
Name	**Collaborator**
countTime createTime	Schedule

ตารางที่ 4.26 ตารางความรับผิดชอบและการร่วมมือของคลาส Schedule

Schedule
Super Class: N/A
Sub Class: StartNowMode , DumpandExecuteMode , ExecutionMode

ตารางที่ 4.26 (ต่อ)

Description: คลาสของการตั้งเวลาทดสอบ	
Attributes:	
Name	**Description**
ScheduleTime	เวลาที่จะทดสอบ
ScheduleMode	รูปแบบของการตั้งเวลาทดสอบ
Responsibilities:	
Name	**Collaborator**
calculateDiskspace checkPlan checkDirectory checkTime checkDate	Planning

ตารางที่ 4.27 ตารางความรับผิดชอบและการร่วมมือของคลาส Build

Build	
Super Class: N/A	
Sub Class: N/A	
Description: คลาสของซอฟต์แวร์ที่ใช้ทดสอบ	
Attributes:	
Name	**Collaborator**
BuildID	รหัสของซอฟต์แวร์
BuildDate	วันที่ของซอฟต์แวร์
BuildVersion	เวอร์ชันของซอฟต์แวร์
BuildProduct	ผลิตภัณฑ์ของซอฟต์แวร์
BuildType	ประเภทของซอฟต์แวร์
Responsibilities:	
Name	**Collaborator**
insertBuild deleteBuild updateBuild lookupBuild	Planning

ตารางที่ 4.28 ตารางความรับผิดชอบและการร่วมมือของคลาส Environment

Environment	
Super Class: N/A	
Sub Class: N/A	
Description: คลาสของสภาพแวดล้อมที่จะทำการทดสอบ เช่นทดสอบบนสภาพแวดล้อมที่เป็น Unix , Sybase 15.0.4 และ WebLogin	
Attributes:	
Name	**Collaborator**
CFGCode	รหัสของ Environment
DBServerID	รหัสของ Database Server
EngServerID	รหัสของ Engine Server
APPServerID	รหัสของ Application Server
Responsibilities:	
Name	**Collaborator**
insertEnvironment	Planning , EngineServer , DBServer , AppServer
updateEnvironment	
deleteEnvironment	
getEnvironmentinfo	

ตารางที่ 4.29 ตารางความรับผิดชอบและการร่วมมือของคลาส EngineServer

EngineServer	
Super Class: N/A	
Sub Class: N/A	
Description: คลาสของเครื่องเซิร์ฟเวอร์ที่ระบุระบบปฏิบัติในการทดสอบ เช่น Unix หรือ Windows	
Attributes:	
Name	**Collaborator**
EngServerID	รหัสของเครื่องระบบปฏิบัติเซิร์ฟเวอร์
EngServerName	ชื่อเครื่องระบบปฏิบัติเซิร์ฟเวอร์
EngServerOS	ประเภทเครื่องระบบปฏิบัติเซิร์ฟเวอร์
Responsibilities:	
Name	**Collaborator**
insertServer	Environment

ตารางที่ 4.29 (ต่อ)

Name	Collaborator
deleteServer updateServer getInformation	

ตารางที่ 4.30 ตารางความรับผิดชอบและการร่วมมือของคลาส APPServer

APPServer	
Super Class: N/A	
Sub Class: N/A	
Description: คลาสของเครื่องเซิร์ฟเวอร์ที่ระบุว่าเป็น Weblogic หรือ Websphere	
Attributes:	
Name	**Collaborator**
APPServerID	รหัสของเครื่องเซิร์ฟเวอร์
APPServerName	ชื่อเครื่องเซิร์ฟเวอร์
APPServerOS	ระบบปฏิบัติการของเครื่องเซิร์ฟเวอร์
APPServerIDE	ประเภทของ IDE
APPServerIDEVersion	เวอร์ชันของ IDE
Responsibilities:	
Name	**Collaborator**
insertServer deleteServer updateServer getInformation	Environment

ตารางที่ 4.31 ตารางความรับผิดชอบและการร่วมมือของคลาส DBServer

DBServer
Super Class: N/A
Sub Class: N/A
Description: คลาสของเครื่องเซิร์ฟเวอร์ที่DBMS ต่างๆ เช่น Sybase , Oracle เป็นต้น
Attributes:

ตารางที่ 4.31 (ต่อ)

Name	Collaborator
DBServerID	รหัสของเครื่องดาต้าเบสเซิร์ฟเวอร์
DBServerName	ชื่อเครื่องดาต้าเบสเซิร์ฟเวอร์
DBServerOS	ระบบปฏิบัติการของเครื่องดาต้าเบสเซิร์ฟเวอร์
DBServerInstant	ชื่อตัวแทนเครื่องดาต้าเบสเซิร์ฟเวอร์
DBServerDBMS	ชื่อ DBMS บนเครื่องดาต้าเบสเซิร์ฟเวอร์
DBServerSuffix	นามสกุลของไฟล์บนเครื่องดาต้าเบสเซิร์ฟเวอร์
Responsibilities:	
Name	**Collaborator**
insertServer	Environment
deleteServer	
updateServer	
getInformation	

4.3.3 ซีเควนซ์ไดอะแกรม

จากยูสเคสไดอะแกรมและคลาสไดอะแกรมของระบบที่ได้กล่าวไปแล้ว สามารถอธิบายถึงการสื่อสารหรือการส่งอ็อบเจกต์เพื่อทำให้เกิดการทำงานขึ้นในระบบ โดยแสดงผ่านแบบจำลองซีแควนซ์ไดอะแกรม โดยจะแสดงเฉพาะซีแควนซ์ไดอะแกรมที่สำคัญ ดังรายละเอียดต่อไปนี้

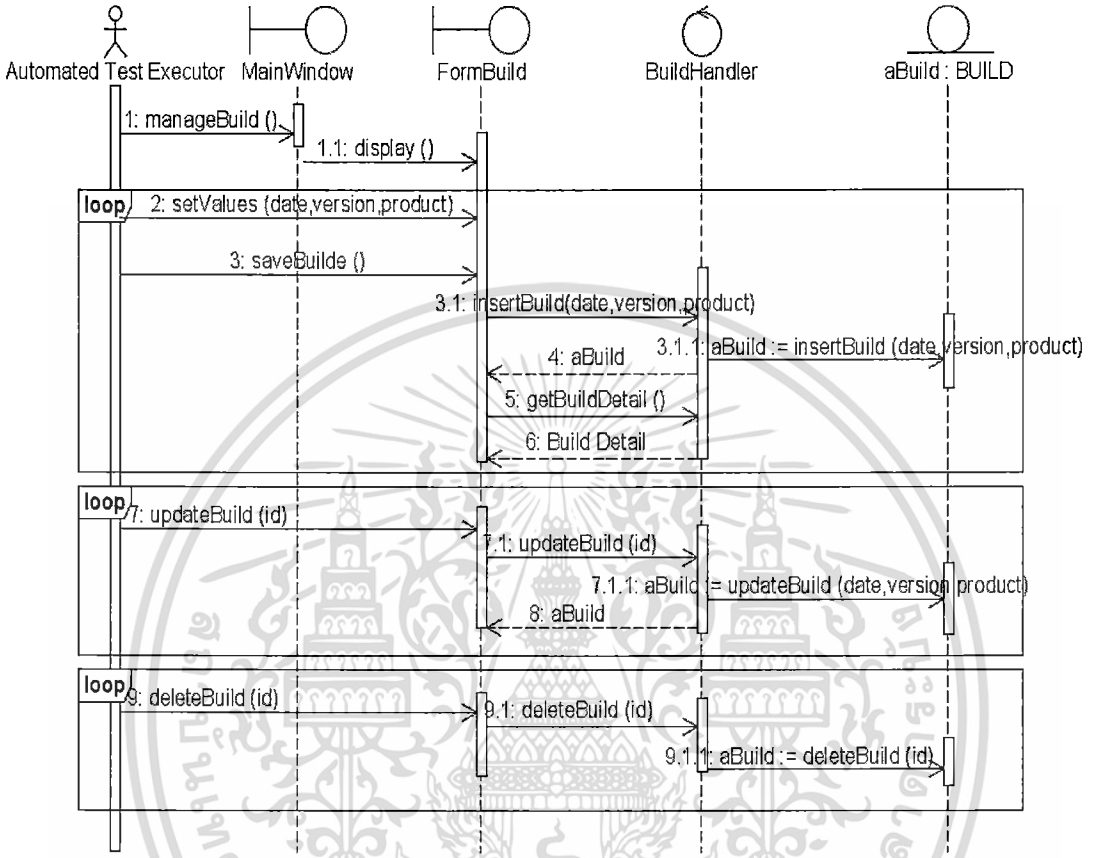
4.3.3.1 ซีเควนซ์ไดอะแกรมการจัดการซอฟต์แวร์ที่ใช้ทดสอบ (Manage Build)

ซีเควนซ์ไดอะแกรมการจัดการซอฟต์แวร์ที่ใช้ทดสอบ แสดงด้วยแผนภาพดังรูปที่ 4.4 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor ต้องการจัดการซอฟต์แวร์ที่ใช้ทดสอบ (Build) จะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Master Data -> Build ระบบจะแสดงหน้าจอรายการ Build ขึ้นมาแสดง ซึ่งในการจัดการนั้นสามารถเพิ่ม แก้ไขและลบข้อมูล Build ได้

กรณีทำการเพิ่มข้อมูล Build นั้น Automated Test Executor ทำการระบุข้อมูลวันที่ เลขเวอร์ชันและผลิตภัณฑ์ จากนั้นกดปุ่ม save ที่หน้าจอ อ็อบเจกต์ BuildHandler จะส่งข้อความไปยังอ็อบเจกต์ Build เพื่อทำการเพิ่มข้อมูล Build ในฐานข้อมูลของระบบ

กรณีทำการแก้ไขข้อมูล Build นั้น Automated Test Executor ทำการเลือก Build ที่ต้องการแก้ไข จากนั้นกดปุ่ม Update ที่หน้าจอ จากนั้นระบุค่าที่ต้องการ แล้วกด save อ็อบเจกต์ BuildHandler จะส่งข้อความไปยังอ็อบเจกต์ Build เพื่อทำการอัพเดตข้อมูล Build ในฐานข้อมูลของระบบ

กรณีทำการลบข้อมูล Build นั้น Automated Test Executor ทำการเลือก Build ที่ต้องการลบ จากนั้นกดปุ่ม Delete ที่หน้าจอ อ็อบเจกต์ BuildHandler จะส่งข้อความไปยังอ็อบเจกต์ Build เพื่อทำการลบข้อมูล Build ออกจากฐานข้อมูลของระบบ



รูปที่4.4 ซีเควนซ์ไดอะแกรมการจัดการซอฟต์แวร์ที่ใช้ทดสอบ (Manage Build)

4.3.3.2 ซีเควนซ์ไดอะแกรมการจัดการเครื่องทดสอบ (Manage Machine)

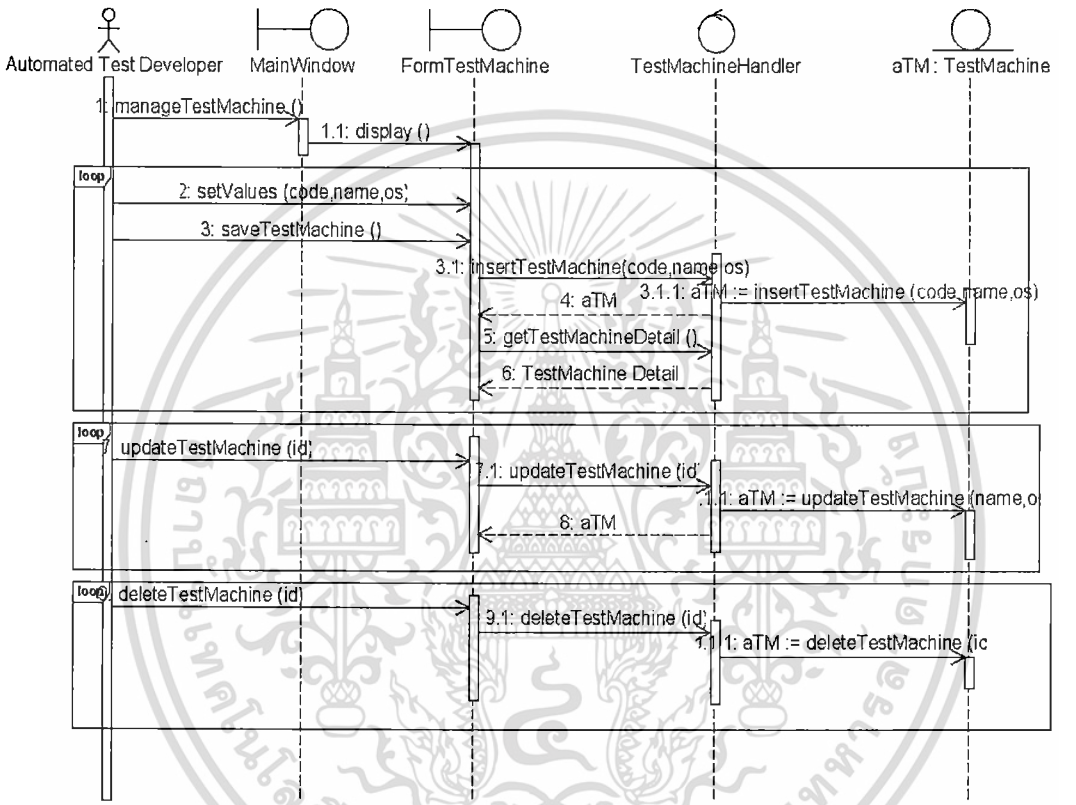
ซีเควนซ์ไดอะแกรมการจัดการเครื่องทดสอบ แสดงด้วยแผนภาพดังรูปที่ 4.5 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor ต้องการจัดการเครื่องทดสอบ จะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Master Data -> PC ระบบจะแสดงหน้าจอรายการเครื่องทดสอบขึ้นมาแสดง ซึ่งในการจัดการนั้นสามารถเพิ่ม แก้ไขและลบข้อมูลเครื่องทดสอบได้

กรณีทำการเพิ่มข้อมูลเครื่องทดสอบ Automated Test Executor ทำการระบุข้อมูลรหัสเครื่อง ชื่อเครื่องและระบบปฏิบัติการของเครื่อง จากนั้นกดปุ่ม save อ็อบเจกต์ TestMachineHandler จะส่งข้อความไปยังอ็อบเจกต์ TestMachine เพื่อทำการเพิ่มข้อมูล TestMachine ในฐานข้อมูลของระบบ

กรณีทำการแก้ไขข้อมูลเครื่องทดสอบ Automated Test Executor ทำการเลือกรหัสเครื่องทดสอบที่ต้องการแก้ไข จากนั้นกดปุ่ม Update ที่หน้าจอ จากนั้นระบุค่าที่ต้องการ แล้วกด save

อ็อบเจกต์ TestMachineHandler จะส่งข้อความไปยังอ็อบเจกต์ TestMachine เพื่อทำการอัพเดตข้อมูล TestMachine ในฐานข้อมูลของระบบ

กรณีทำการลบข้อมูลเครื่องทดสอบ Automated Test Executor ทำการเลือกรหัสของเครื่องทดสอบที่ต้องการลบ จากนั้นกดปุ่ม Delete ที่หน้าจอ อ็อบเจกต์ TestMachineHandler จะส่งข้อความไปยังอ็อบเจกต์ TestMachine เพื่อทำการลบข้อมูล TestMachine ออกจากฐานข้อมูลของระบบ



รูปที่4.5 ซีเควนซ์ไดอะแกรมการจัดการเครื่องทดสอบ (Manage Machine)

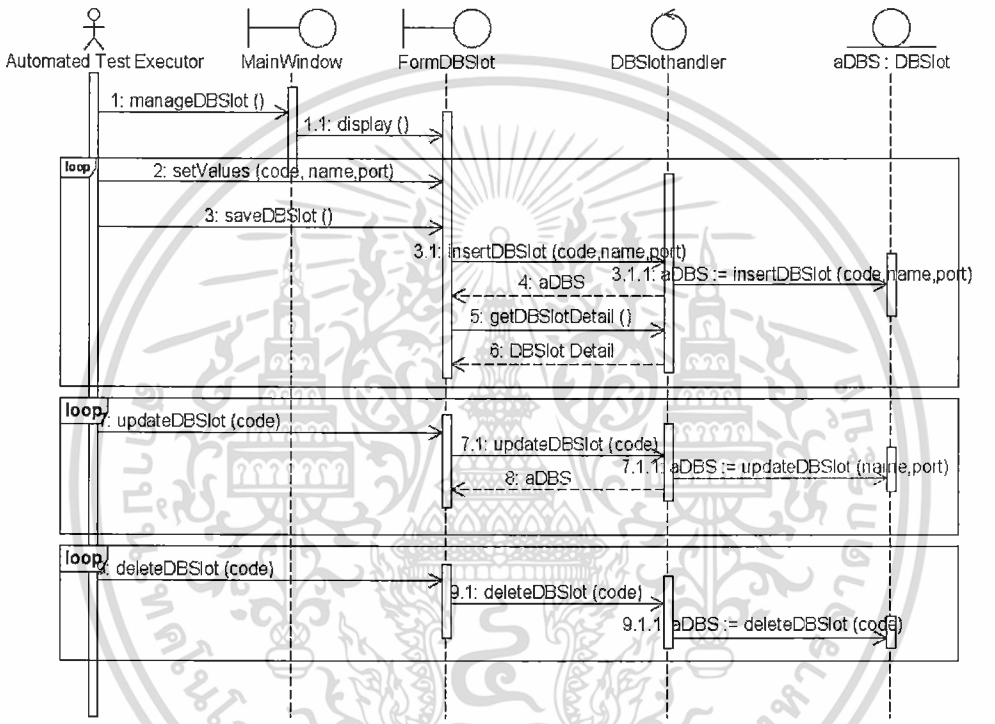
4.3.3.3 ซีเควนซ์ไดอะแกรมการจัดการ Database Slot (Manage DB Slot)

ซีเควนซ์ไดอะแกรมการจัดการ Database Slot แสดงด้วยแผนภาพดังรูปที่ 4.6 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor ต้องการจัดการ Database Slot จะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Master Data -> Database Slot ระบบจะแสดงหน้าจอรายการ Database Slot ขึ้นมาแสดง ซึ่งในการจัดการนั้นสามารถเพิ่ม แก้ไขและลบข้อมูลเครื่องทดสอบได้

กรณีทำการเพิ่มข้อมูล Database Slot นั้น Automated Test Executor ทำการระบุข้อมูลรหัส ชื่อ และหมายเลขพอร์ด จากนั้นกดปุ่ม save อ็อบเจกต์ DBSlotHandler จะส่งข้อความไปยังอ็อบเจกต์ DBSlot เพื่อทำการเพิ่มข้อมูล TestMachine ในฐานข้อมูลของระบบ

กรณีทำการแก้ไขข้อมูล Database Slot นั้น Automated Test Executor ทำการเลือกรหัสของ Database Slot ที่ต้องการแก้ไข จากนั้นกดปุ่ม Update ที่หน้าจอ จากนั้นระบุค่าที่ต้องการ แล้วกด save อ็อบเจกต์ DBSlotHandler จะส่งข้อความไปยังอ็อบเจกต์ DBSlot เพื่อทำการอัพเดตข้อมูล TestMachine ในฐานข้อมูลของระบบ

กรณีทำการลบข้อมูล Database Slot นั้น Automated Test Executor ทำการเลือก Database Slot ที่ต้องการลบ จากนั้นกดปุ่ม Delete ที่หน้าจอ อ็อบเจกต์ DBSlotHandler จะส่งข้อความไปยังอ็อบเจกต์ DBSlot เพื่อทำการลบข้อมูล TestMachine ออกจากฐานข้อมูลของระบบ

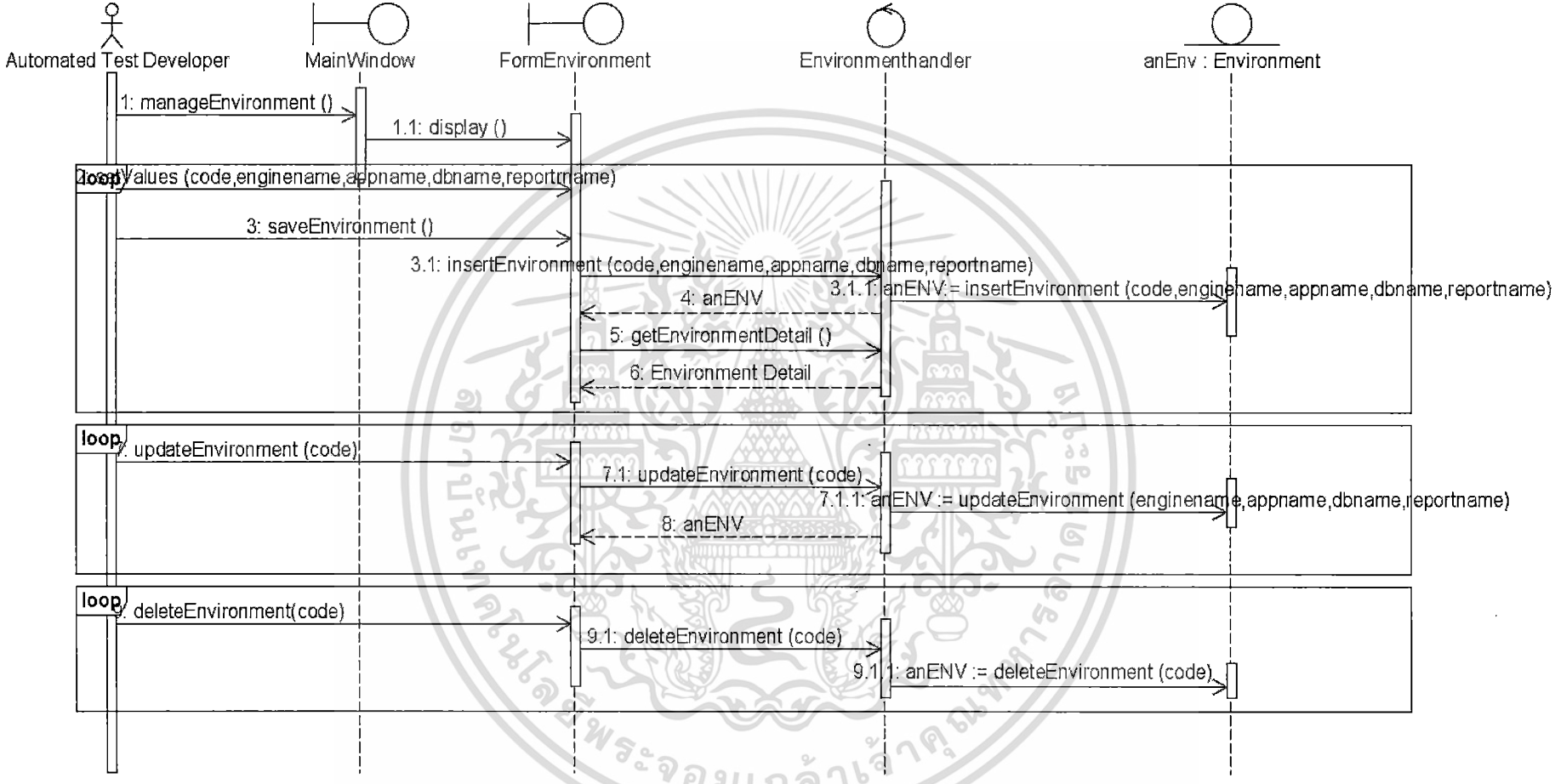


รูปที่4.6 ซีเควนซ์ไดอะแกรมการจัดการ Database Slot (Manage DB Slot)

4.3.3.4 ซีเควนซ์ไดอะแกรมการจัดการสภาพแวดล้อมที่ใช้ในการทดสอบ (Manage Environment)

ซีเควนซ์ไดอะแกรมการจัดการสภาพแวดล้อมที่ใช้ในการทดสอบ แสดงด้วยแผนภาพดังรูปที่ 4.7 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Developer ต้องการจัดการสภาพแวดล้อมที่ใช้ในการทดสอบ จะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Master Data -> Configuration Mapping ระบบจะแสดงหน้าจอรายการ Configuration ที่มีอยู่ขึ้นมาแสดง ซึ่งในการจัดการนั้นสามารถเพิ่ม แก้ไขและลบข้อมูล Configuration Mapping ได้

กรณีทำการเพิ่มข้อมูล Configuration Mapping นั้น Automated Test Developer ทำการระบุข้อมูลรหัส ชื่อเครื่องเซิร์ฟเวอร์ จากนั้นกดปุ่ม save อ็อบเจกต์ EnvironmentHandler จะส่งข้อความไปยังอ็อบเจกต์ Environment เพื่อทำการเพิ่มข้อมูล Environment ในฐานข้อมูลของระบบ



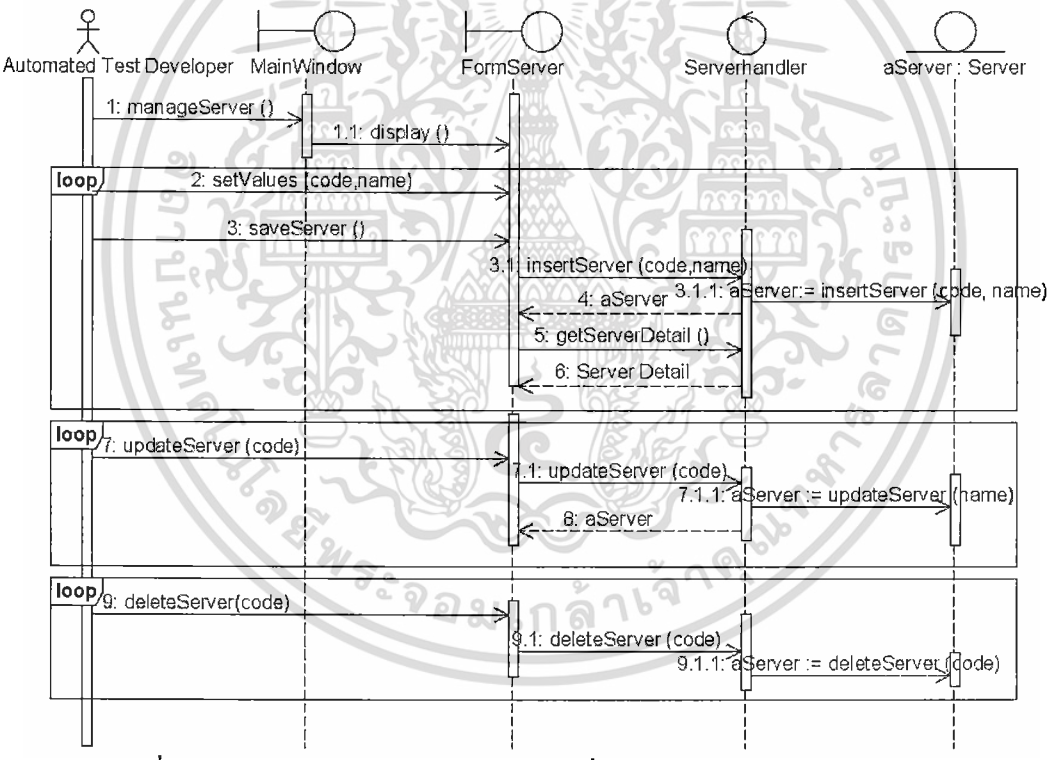
รูปที่ 4.7 ซีเควนซ์ไดอะแกรมการจัดการสภาพแวดล้อมที่ใช้ในการทดสอบ (Manage Environment)

กรณีทำการแก้ไขข้อมูล Configuration Mapping นั้น Automated Test Developer ทำการเลือกรหัสของ Configuration Mapping ที่ต้องการแก้ไข จากนั้นกดปุ่ม Update ที่หน้าจอ จากนั้นระบุค่าที่ต้องการ แล้วกด save อ็อบเจกต์ EnvironmentHandler จะส่งข้อความไปยังอ็อบเจกต์ Environment เพื่อทำการอัพเดตข้อมูล Environment ในฐานข้อมูลของระบบ

กรณีทำการลบข้อมูล Configuration Mapping นั้น Automated Test Developer ทำการเลือก Configuration Mapping ที่ต้องการลบ จากนั้นกดปุ่ม Delete ที่หน้าจอ อ็อบเจกต์ EnvironmentHandler จะส่งข้อความไปยังอ็อบเจกต์ Environment เพื่อทำการลบข้อมูล Environment ออกจากฐานข้อมูลของระบบ

4.3.3.5 ซีเควนซ์ไดอะแกรมการจัดการเครื่องเซิร์ฟเวอร์ (Manage Server)

ซีเควนซ์ไดอะแกรมการจัดการเครื่องเซิร์ฟเวอร์ แสดงแผนภาพดังรูปที่ 4.8



รูปที่ 4.8 ซีเควนซ์ไดอะแกรมการจัดการเครื่องเซิร์ฟเวอร์ (Manage Server)

จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Developer ต้องการจัดการเครื่องเซิร์ฟเวอร์ จะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Master Data -> Engine Server หรือ Application Server หรือ DB Server หรือ Report Server ระบบจะแสดงหน้าจอรายการเซิร์ฟเวอร์ตามที่ได้เลือกเข้ามาขึ้นมาแสดง ซึ่งในการจัดการนั้นสามารถเพิ่ม แก้ไขและลบข้อมูลเซิร์ฟเวอร์ได้

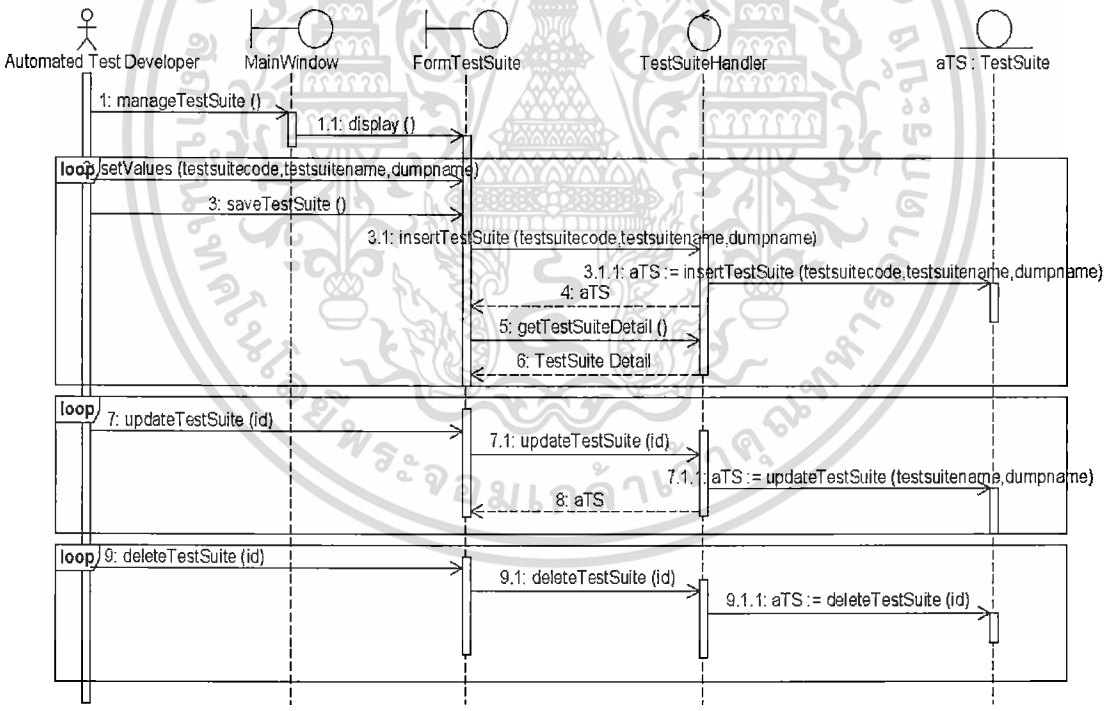
กรณีทำการเพิ่มข้อมูลเซิร์ฟเวอร์นั้น Automated Test Developer ทำการระบุข้อมูลรหัส ชื่อเครื่องเซิร์ฟเวอร์ จากนั้นกดปุ่ม save อ็อบเจกต์ ServerHandler จะส่งข้อความไปยังอ็อบเจกต์ Server เพื่อทำการเพิ่มข้อมูล Server ในฐานข้อมูลของระบบ

กรณีทำการแก้ไขเซิร์ฟเวอร์นั้น Automated Test Developer ทำการเลือกรหัสของเซิร์ฟเวอร์ที่ต้องการแก้ไข จากนั้นกดปุ่ม Update ที่หน้าจอ จากนั้นระบุค่าที่ต้องการ แล้วกด save อ็อบเจกต์ ServerHandler จะส่งข้อความไปยังอ็อบเจกต์ Server เพื่อทำการอัพเดตข้อมูล Server ในฐานข้อมูลของระบบ

กรณีทำการลบข้อมูลเซิร์ฟเวอร์นั้น Automated Test Developer ทำการเลือกรหัสของเซิร์ฟเวอร์ ที่ต้องการลบ จากนั้นกดปุ่ม Delete ที่หน้าจอ อ็อบเจกต์ ServerHandler จะส่งข้อความไปยังอ็อบเจกต์ Server เพื่อทำการลบข้อมูล Server ออกจากฐานข้อมูลของระบบ

4.3.3.6 ซีเควนซ์ไดอะแกรมการจัดการสคริปท์ที่ใช้ทดสอบ (Manage TestSuite)

ซีเควนซ์ไดอะแกรมการจัดการสคริปท์ที่ใช้ทดสอบ แสดงด้วยแผนภาพดังรูปที่ 4.9



รูปที่ 4.9 ซีเควนซ์ไดอะแกรมการจัดการสคริปท์ที่ใช้ทดสอบ (Manage TestSuite)

จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Developer ต้องการจัดการสคริปท์ที่ใช้ทดสอบ จะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Master Data -> Test Suite ระบบจะแสดงหน้าจอรายการสคริปท์ที่ใช้ทดสอบที่มีอยู่ขึ้นมาแสดง ซึ่งในการจัดการนั้นสามารถเพิ่ม แก้ไขและลบข้อมูลสคริปท์ที่ใช้ทดสอบได้

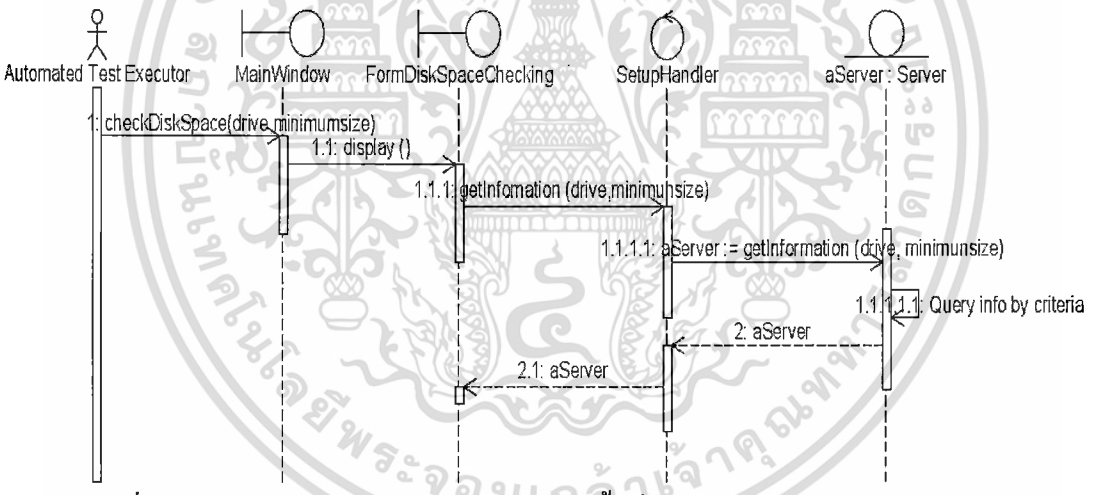
กรณีทำการเพิ่มข้อมูลสคริปท์ที่ใช้ทดสอบนั้น Automated Test Developer ทำการระบุข้อมูลรหัส ชื่อของสคริปท์ จากนั้นกดปุ่ม save อ็อบเจกต์ TestSuiteHandler จะส่งข้อความไปยังอ็อบเจกต์ TestSuite เพื่อทำการเพิ่มข้อมูล Test Suite ในฐานข้อมูลของระบบ

กรณีทำการแก้ไขข้อมูลสคริปท์ที่ใช้ทดสอบนั้น Automated Test Developer ทำการเลือกรหัสของ Test Suite ที่ต้องการแก้ไข จากนั้นกดปุ่ม Update ที่หน้าจอ จากนั้นระบุค่าที่ต้องการ แล้วกด save อ็อบเจกต์ TestSuiteHandler จะส่งข้อความไปยังอ็อบเจกต์ TestSuite เพื่อทำการอัพเดตข้อมูล Test Suite ในฐานข้อมูลของระบบ

กรณีทำการลบข้อมูลสคริปท์ที่ใช้ทดสอบนั้น Automated Test Developer ทำการเลือกรหัสของ Test Suite ที่ต้องการลบ จากนั้นกดปุ่ม Delete ที่หน้าจอ อ็อบเจกต์ TestSuiteHandler จะส่งข้อความไปยังอ็อบเจกต์ TestSuite เพื่อทำการลบข้อมูล Test Suite ออกจากฐานข้อมูลของระบบ

4.3.3.7 ซีเควนซ์ไดอะแกรมการตรวจสอบพื้นที่บน Server (Check Disk Space)

ซีเควนซ์ไดอะแกรมการตรวจสอบพื้นที่บน Server แสดงด้วยแผนภาพดังรูปที่ 4.10

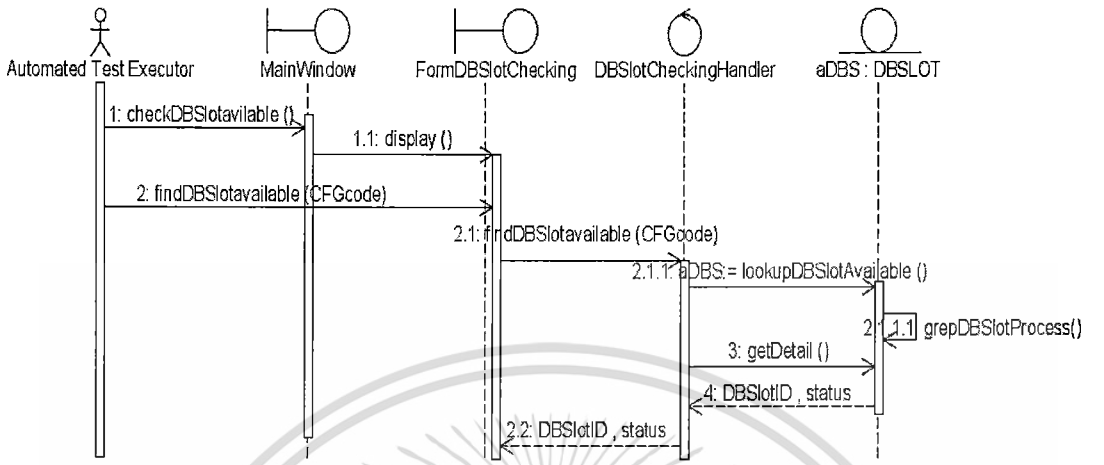


รูปที่ 4.10 ซีเควนซ์ไดอะแกรมการตรวจสอบพื้นที่บน Server (Check Disk Space)

จากรูปสามารถอธิบายได้ดังนี้ ก่อนทำการทดสอบ Automated Test Executor ควรจะตรวจสอบพื้นที่บน Server ที่ใช้ทดสอบ โดยจะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Tools -> Disk Space checking ระบบจะแสดงหน้าจอการตรวจสอบพื้นที่ของ Drive ที่ได้ระบุไว้ว่าเหลือพื้นที่เท่าไหร่เพียงพอกับที่จะใช้ทดสอบหรือไม่

4.3.3.8 ซีเควนซ์ไดอะแกรมการตรวจสอบการใช้งานของ Database Slot (Check Database Slot)

ซีเควนซ์ไดอะแกรมการตรวจสอบการใช้งานของ Database Slot แสดงด้วยแผนภาพดังรูปที่ 4.11

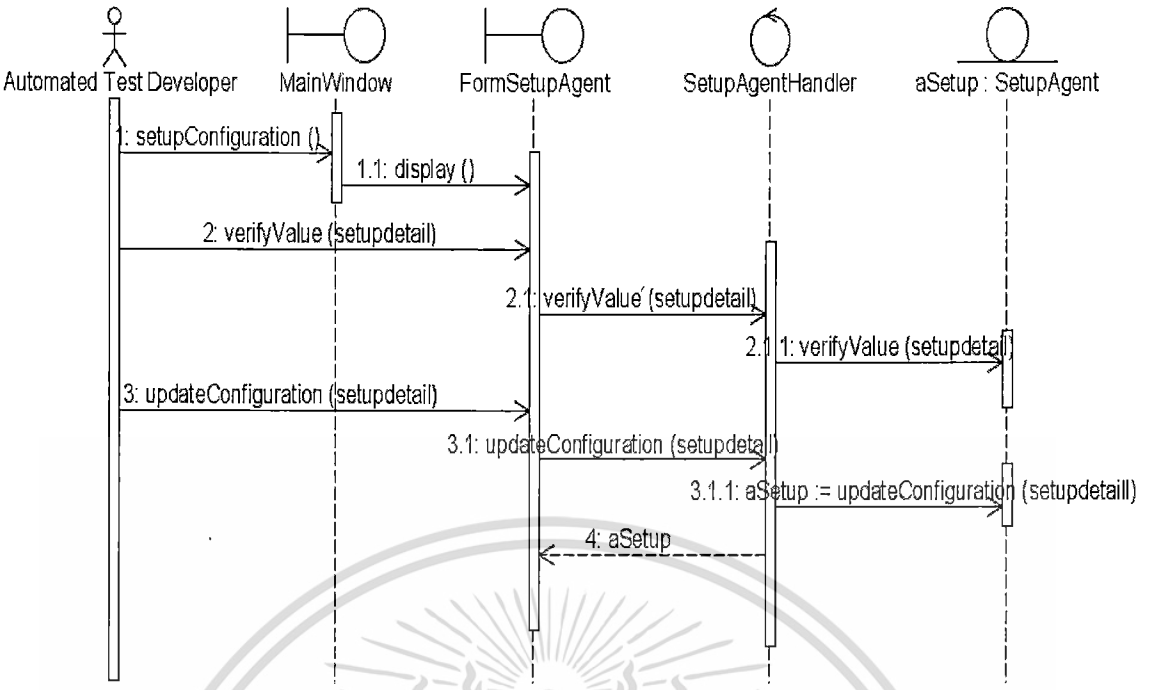


รูปที่4.11 ซีเควนซ์ไดอะแกรมการตรวจสอบ Database Slot (Check DBSlot)

จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor ต้องการตรวจสอบดูว่า Database Slot ตัวใดว่าง โดยจะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Tools -> Database Slot checking ระบบจะแสดงหน้าจอการตรวจสอบการใช้งานของ Database Slot จากนั้นAutomated Test Executor จะส่งข้อความไปยังอ็อบเจกต์ DBSlotCheckingHandler เพื่อขอข้อมูล CFGCode จากนั้นอ็อบเจกต์ DBSlot จะทำการตรวจสอบสถานะด้วย UNIX command แล้วส่งข้อความกลับมาแสดงผลให้กับ Automated Test Executor

4.3.3.9 ซีเควนซ์ไดอะแกรมการเตรียมไฟล์ต่างๆที่เกี่ยวข้องกับการทดสอบ (Setup Configuration File)

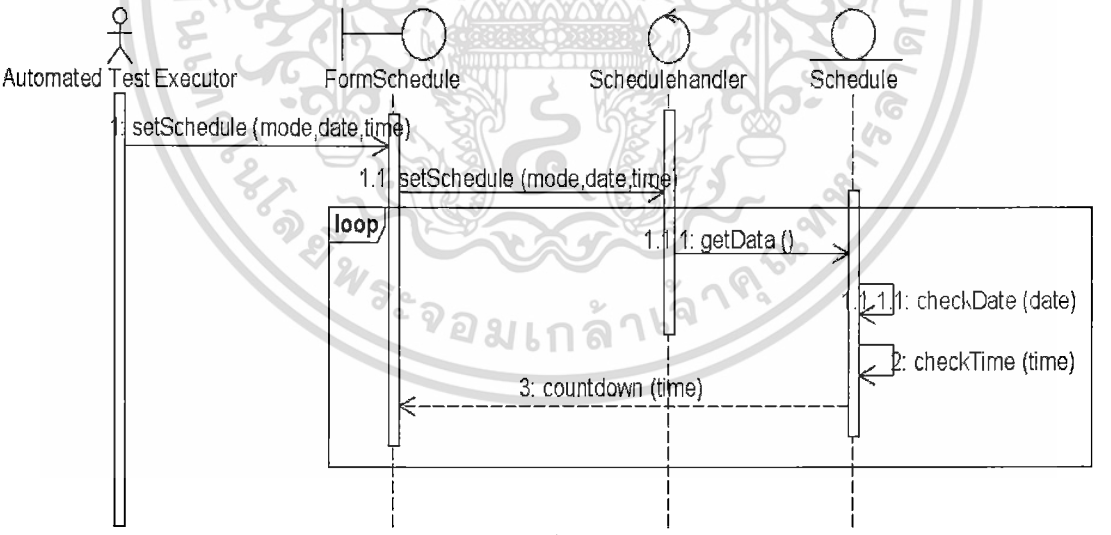
ซีเควนซ์ไดอะแกรมการเตรียมไฟล์ต่างที่เกี่ยวข้องกับการทดสอบ แสดงด้วยแผนภาพดังรูปที่ 4.12 จากรูปสามารถอธิบายได้ดังนี้ ก่อนการทดสอบ Automated Test Developer จะเข้ามาที่ระบบเพื่อทำการเตรียมไฟล์ Configuration ต่างๆที่เกี่ยวข้องกับการทดสอบให้พร้อม โดยจะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Tools -> Setting ระบบจะแสดงหน้าจอการ Setup ค่าต่างๆที่ต้องใช้ในการทดสอบ Automated Test Developer จะส่งข้อความไปยังอ็อบเจกต์ SetupAgentHandler จากนั้นจะส่งข้อความต่อไปยังอ็อบเจกต์ SetupAgent เพื่อตรวจสอบค่าการ Setup ต่างๆ หรือทำการแก้ไขข้อมูลต่างๆได้



รูปที่ 4.12 ซีเควนซ์ไดอะแกรมการเตรียมไฟล์ต่างๆที่เกี่ยวข้องกับการทดสอบ

4.3.3.10 ซีเควนซ์ไดอะแกรมการตั้งเวลาทดสอบ (Schedule Run)

ซีเควนซ์ไดอะแกรมการตั้งเวลาทดสอบ (Schedule Run) แสดงด้วยแผนภาพดังรูปที่ 4.13



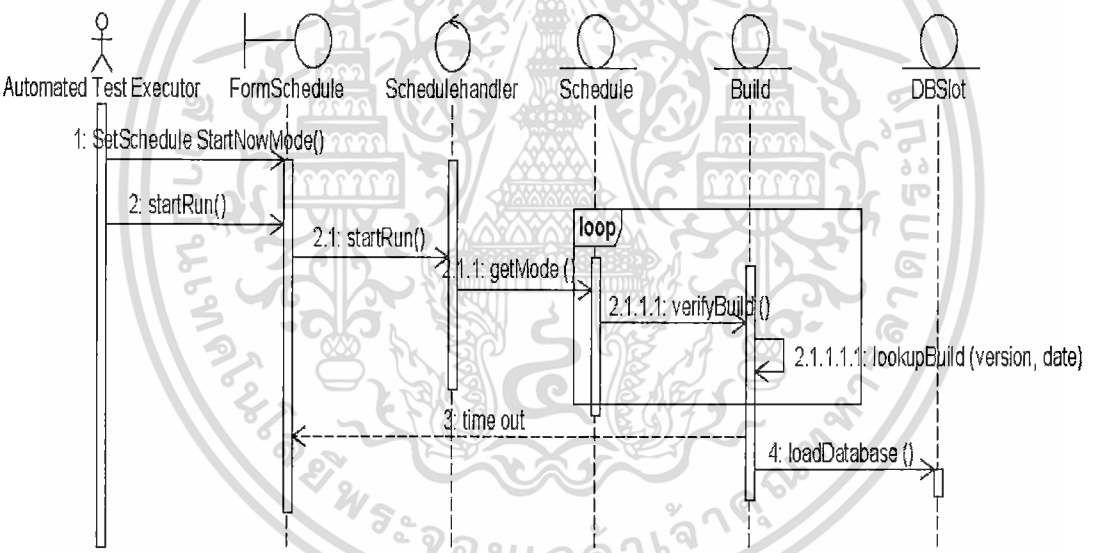
รูปที่ 4.13 ซีเควนซ์ไดอะแกรมการตั้งเวลาทดสอบแบบ Schedule Run

จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor ต้องการทดสอบ ผู้ทดสอบจะต้องทำการตั้งเวลาการทดสอบ ซึ่งมีรูปแบบให้เลือก 3 Mode ได้แก่ Start Now Mode, Schedule AutoDump and Execution Mode และ Schedule Execution Mode ซึ่ง Automated Test Executor จะต้องเลือก Mode และระบุเงื่อนไขที่ต้องการทดสอบที่หน้าจอของแอพพลิเคชัน เมื่อกำหนดค่า

เรียบร้อยแล้วจะส่งข้อความไปยังอ็อบเจกต์ ScheduleHandler เพื่อส่งข้อความไปให้อ็อบเจกต์ Schedule จากนั้นอ็อบเจกต์ Schedule จะทำการตรวจสอบเงื่อนไขได้รับ และดำเนินงานต่อไปเมื่อถึงเวลาตามเงื่อนไข

4.3.3.11 ซีเควนซ์ไดอะแกรมรูปแบบของการทดสอบแบบทันที (Start Now)

ซีเควนซ์ไดอะแกรมรูปแบบของการทดสอบแบบทันที แสดงด้วยแผนภาพดังรูปที่ 4.14 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor เลือกรูปแบบการทดสอบ แบบ Start Now และกดรันไปแล้ว ระบบจะส่งข้อความไปยังอ็อบเจกต์ ScheduleHandler เพื่อส่งข้อความไปให้อ็อบเจกต์ Schedule จากนั้นอ็อบเจกต์ Schedule จะเริ่มทำการทดสอบโดยไปตรวจสอบวันที่และเวอร์ชันของ Build ว่าตรงตามที่กำหนดไว้หรือไม่ ถ้าไม่มีก็จะทำการวนหาจนหมดเวลา (Time out) แต่ถ้ามีก็จะส่งข้อความไปยังอ็อบเจกต์ DBSlot เพื่อทำการโหลดข้อมูลเข้า Database Slot และดำเนินการทดสอบต่อไป

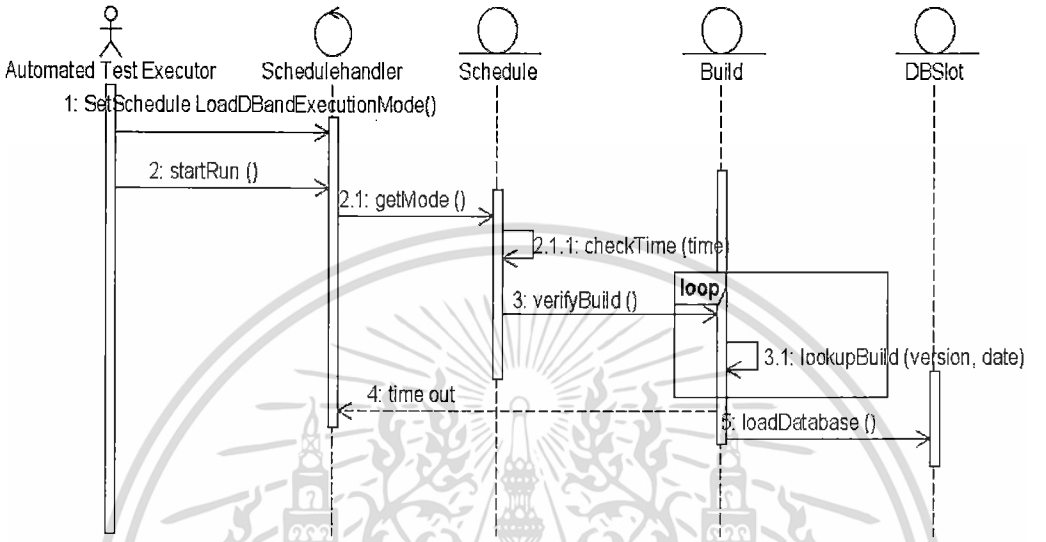


รูปที่ 4.14 ซีเควนซ์ไดอะแกรมรูปแบบของการทดสอบแบบทันที (Start Now)

4.3.3.12 ซีเควนซ์ไดอะแกรมรูปแบบของการกำหนดเวลาทดสอบล่วงหน้าแบบเตรียม Environment และทดสอบ (Schedule AutoDump and Execute)

ซีเควนซ์ไดอะแกรมรูปแบบของการกำหนดเวลาทดสอบล่วงหน้าแบบเตรียม Environment และทดสอบ แสดงด้วยแผนภาพดังรูป 4.15 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor เลือกรูปแบบการทดสอบแบบ Schedule AutoDump and Execute และกดรันไปแล้ว ระบบจะส่งข้อความไปยังอ็อบเจกต์ ScheduleHandler เพื่อส่งข้อความไปให้อ็อบเจกต์ Schedule จากนั้นอ็อบเจกต์ Schedule จะเริ่มทำตรวจสอบเงื่อนไขของเวลาตามที่ Automated Test Executor

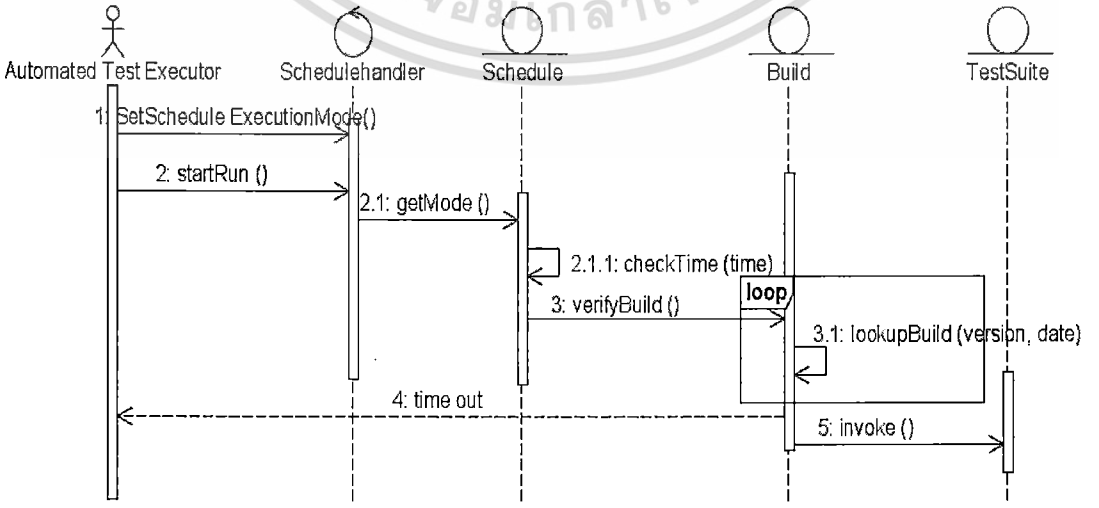
ได้ทำการตั้งเวลาทดสอบไว้ เมื่อถึงเวลาที่ได้ระบุไว้ ระบบจะส่งข้อความไปให้อ็อบเจกต์ Build เพื่อตรวจสอบวันที่และเวอร์ชันของ Build ว่าตรงตามที่กำหนดไว้หรือไม่ ถ้าไม่มีก็จะทำการวนหาจนหมดเวลา (Time out) แต่ถ้ามีก็จะส่งข้อความไปยังอ็อบเจกต์ DBSlot เพื่อทำการโหลดข้อมูลเข้า Database Slot และดำเนินการทดสอบต่อไป



รูปที่ 4.15 ซีเควนซ์ไดอะแกรมรูปแบบของการกำหนดเวลาทดสอบล่วงหน้าแบบเตรียม Environment และทดสอบ (Schedule AutoDump and Execute)

4.3.3.13 ซีเควนซ์ไดอะแกรมรูปแบบของการกำหนดเวลาทดสอบล่วงหน้าแบบทดสอบอย่างเดียว (Schedule Execution)

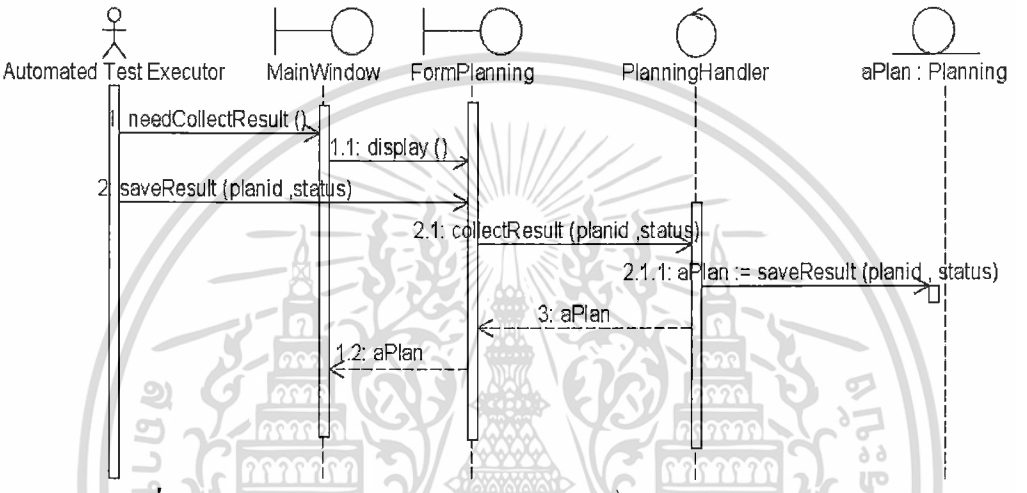
ซีเควนซ์ไดอะแกรมรูปแบบของการกำหนดเวลาทดสอบล่วงหน้าแบบทดสอบอย่างเดียว แสดงด้วยแผนภาพดังรูปที่ 4.16



รูปที่ 4.16 ซีเควนซ์ไดอะแกรมรูปแบบของการกำหนดเวลาทดสอบล่วงหน้าแบบทดสอบอย่างเดียว

4.3.3.14 ซีเควนซ์ไดอะแกรมการเก็บผลการทดสอบ (Collect Result)

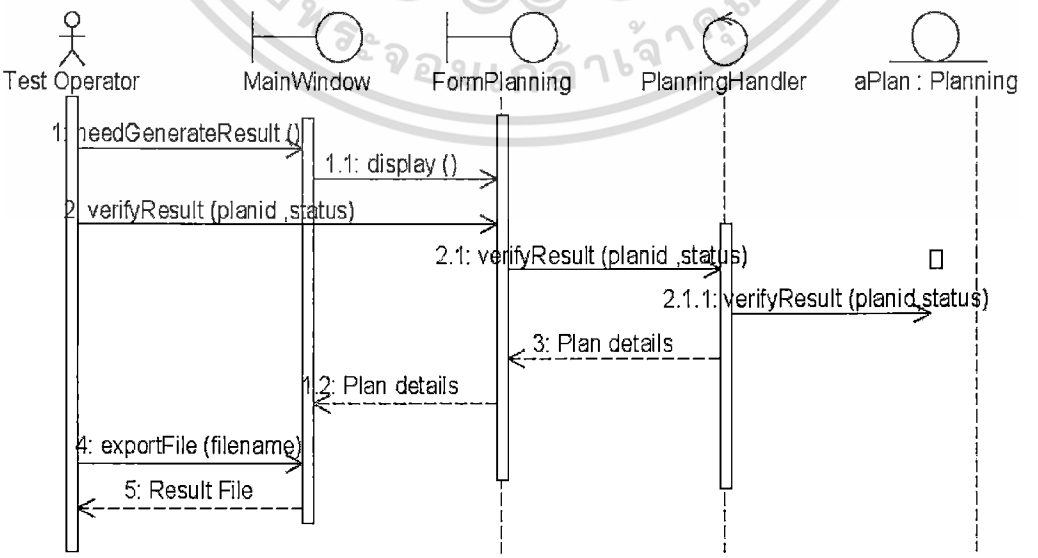
ซีเควนซ์ไดอะแกรมการเก็บผลการทดสอบ แสดงด้วยแผนภาพดังรูปที่ 4.17 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor ต้องการบันทึกผลการทดสอบ จะเข้ามาที่ระบบและไปที่ Tools -> Planning ระบบจะแสดงหน้าจอแผนการทดสอบ Automated Test Executor ทำการบันทึกผลการทดสอบ อ็อบเจกต์ PlanningHandler จะทำการส่งข้อความไปอ็อบเจกต์ Planning เพื่อทำการบันทึกข้อมูลลงในระบบ และส่งข้อความกลับมาที่หน้าพร้อมทั้งแสดงรายการที่ได้บันทึกผลการทดสอบเรียบร้อยแล้ว



รูปที่ 4.17 ซีเควนซ์ไดอะแกรมการเก็บผลการทดสอบ (Collect Result)

4.3.3.15 ซีเควนซ์ไดอะแกรมารออกรายงานผลการทดสอบ (Generate Report)

ซีเควนซ์ไดอะแกรมารออกรายงานผลการทดสอบ แสดงด้วยแผนภาพดังรูป 4.18



รูปที่ 4.18 ซีเควนซ์ไดอะแกรมารออกรายงานผลการทดสอบ (Generate Report)

จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Test Operator ต้องการออกรายงานผลการทดสอบ จะเข้ามาที่ระบบและไปที่ Tools -> Planning โดยส่งข้อความไปอ็อบเจกต์ PlanningHandler เพื่อส่งข้อความไปยังอ็อบเจกต์ Planning เพื่อทำการตรวจสอบผลการทดสอบที่ต้องการว่าได้ใส่ผลการทดสอบเรียบร้อยแล้ว อ็อบเจกต์ Planning จะส่งข้อความรายละเอียดกลับมาที่หน้าจอ จากนั้น Test Operator ทำการออกรายงานผลการทดสอบจะได้ไฟล์ผลการทดสอบออกมาจากระบบ

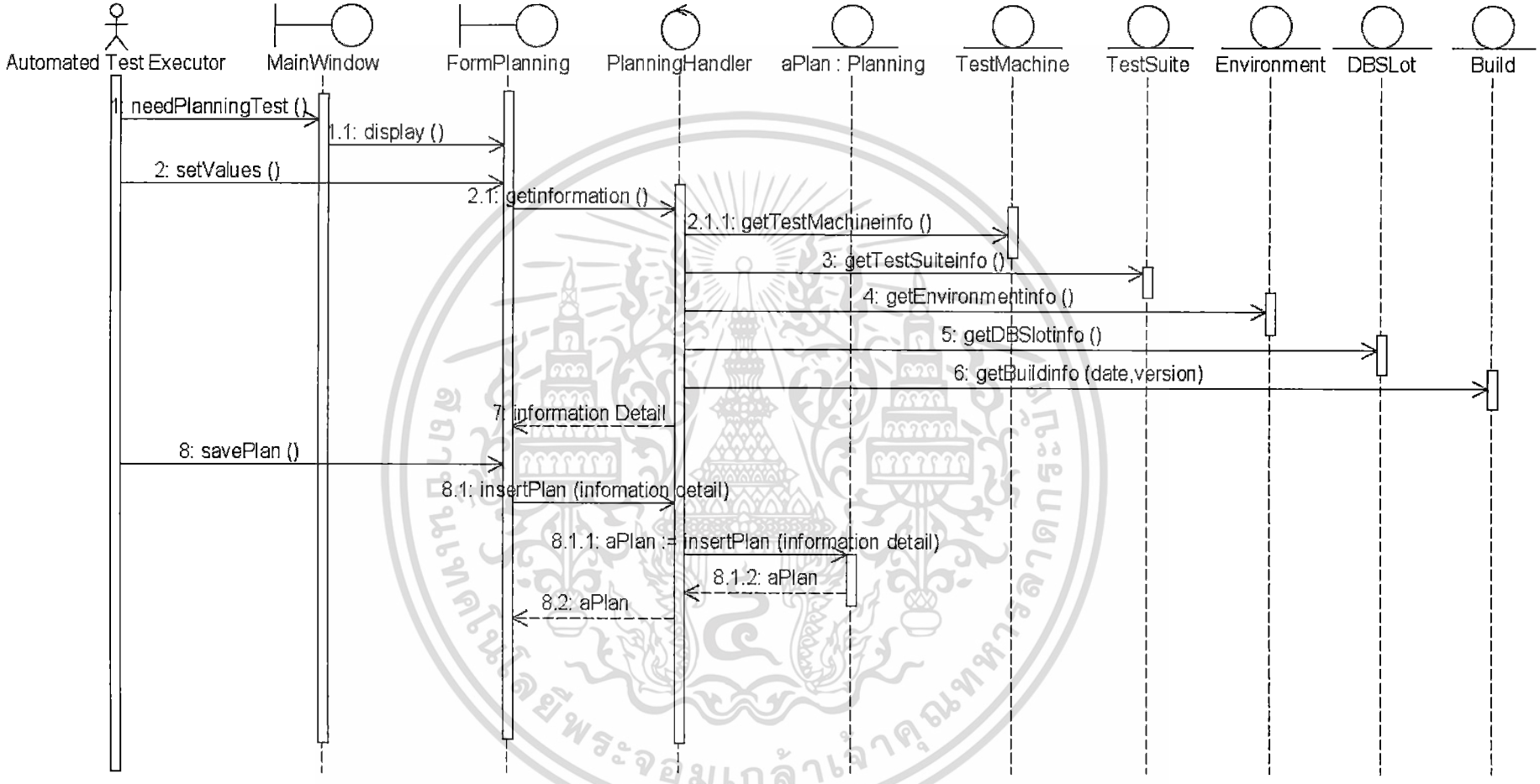
4.3.3.16 ซีเควนซ์ไดอะแกรมการวางแผนการทดสอบ (Planning Test)

ซีเควนซ์ไดอะแกรมการวางแผนการทดสอบ แสดงด้วยแผนภาพดังรูปที่ 4.19 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor ต้องการวางแผนการทดสอบ จะเข้าไปที่ระบบจากนั้นเข้าสู่เมนู Tools -> Planning ระบบจะแสดงหน้าจอการวางแผนการทดสอบ อ็อบเจกต์ PlanningHandler จะส่งข้อความไปยังอ็อบเจกต์อื่นๆที่เกี่ยวข้องเพื่อดึงข้อมูลต่างๆมาแสดงที่หน้าจอ ซึ่งในการจัดการนั้นสามารถเพิ่ม แก้ไขและลบแผนทดสอบได้

กรณีทำการเพิ่มแผนทดสอบ Automated Test Executor ทำการระบุข้อมูลที่เกี่ยวข้องที่หน้าจอ โดยต้องระบุข้อมูลให้ครบ จากนั้นกดปุ่ม save อ็อบเจกต์ PlanningHandler จะส่งข้อความไปยังอ็อบเจกต์ Planning เพื่อทำการเพิ่มข้อมูล Plan ในฐานข้อมูลของระบบ

กรณีทำการแก้ไขแผนทดสอบ Automated Test Executor ทำการเลือกรหัสแผนทดสอบที่ต้องการแก้ไข จากนั้นกดปุ่ม Update ที่หน้าจอ จากนั้นระบุค่าที่ต้องการ แล้วกด save อ็อบเจกต์ PlanningHandler จะส่งข้อความไปยังอ็อบเจกต์ Planning เพื่อทำการอัพเดตข้อมูล Plan ในฐานข้อมูลของระบบ

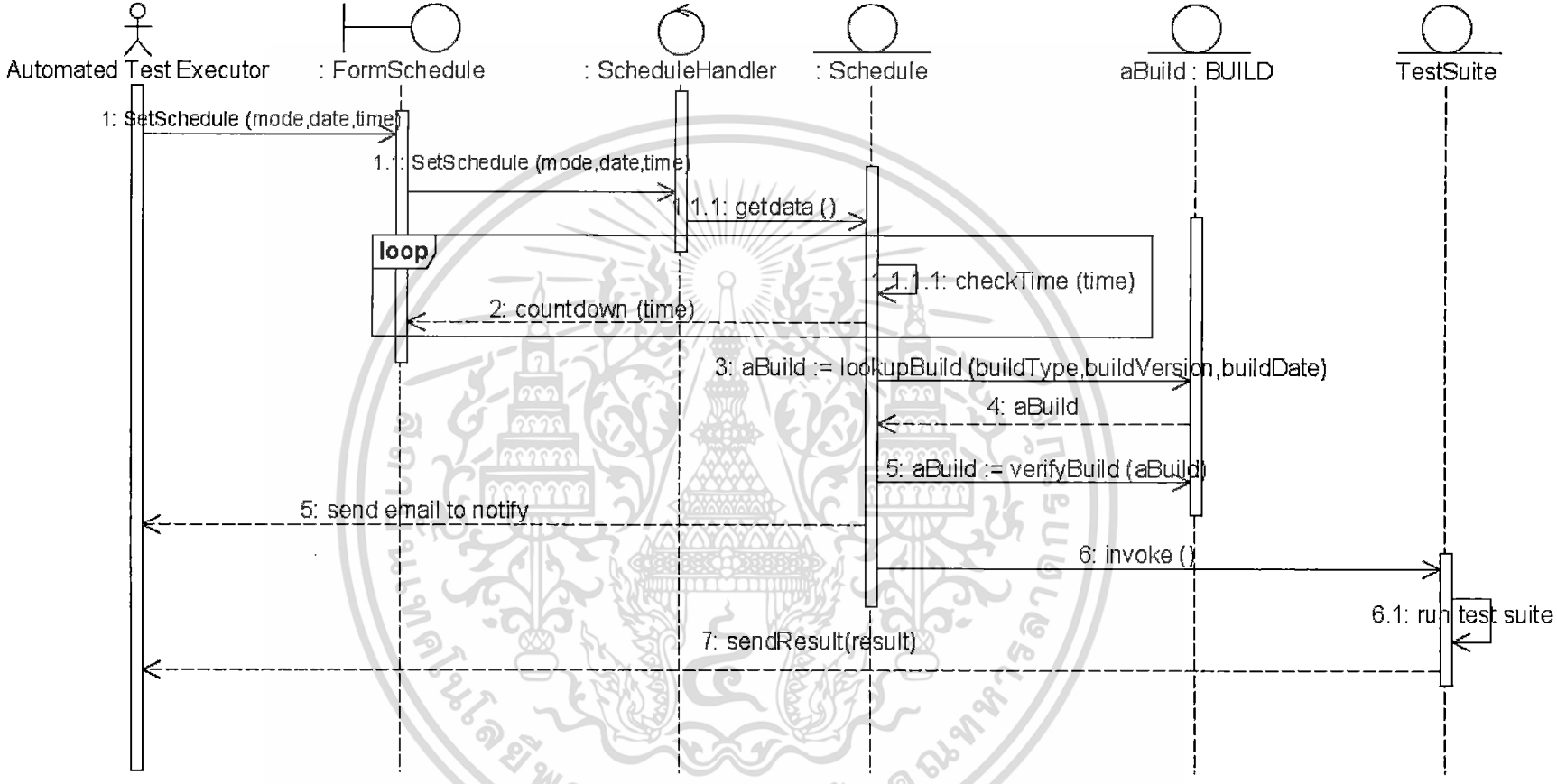
กรณีทำการลบแผนทดสอบ Automated Test Executor ทำการเลือก รหัสแผนทดสอบที่ต้องการลบ จากนั้นกดปุ่ม Delete ที่หน้าจอ อ็อบเจกต์ PlanningHandler จะส่งข้อความไปยังอ็อบเจกต์ Planning เพื่อทำการลบข้อมูล Plan ออกจากฐานข้อมูลของระบบ



รูปที่ 4.19 ซีเควนซ์ไดอะแกรมการวางแผนการทดสอบ (Planning Test)

4.3.3.17 ซีเควนซ์ไดอะแกรมการทดสอบบนเครื่องทดสอบแต่ละเครื่อง (Execute Test)

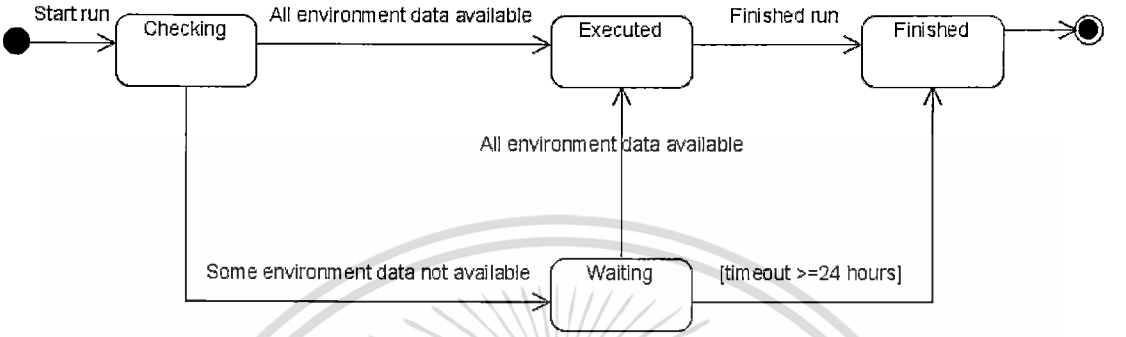
ซีเควนซ์ไดอะแกรมการทดสอบบนเครื่องทดสอบแต่ละเครื่อง แสดงด้วยแผนภาพดังรูปที่ 4.20 จากรูปสามารถอธิบายได้ดังนี้ เมื่อ Automated Test Executor ทำการตั้งเวลาด้วยรูปแบบที่ต้องการและทำการกดรัน อ็อบเจกต์ ScheduleHandler จะทำการส่งข้อความไปยังอ็อบเจกต์ Schedule เพื่อทำการขอข้อมูลการตั้งเวลา อ็อบเจกต์ Schedule ทำการดึงข้อมูลมาตรวจสอบว่าตั้งเวลาทดสอบด้วย Mode ใด จากนั้นอ็อบเจกต์ Schedule จะทำการตรวจสอบเงื่อนไขเรื่องวันที่และเวลาที่ทดสอบเมื่อถึงกำหนดจะส่งข้อความไปยังอ็อบเจกต์ Build เพื่อตรวจสอบเวอร์ชันและวันที่ของ Build อ็อบเจกต์ Build จะส่งข้อความเวอร์ชันและวันที่ที่ต้องการไปหา ServerBuild และ ClientBuild จากนั้นทั้งอ็อบเจกต์ ServerBuild และ ClientBuild จะส่งข้อความกลับมาที่อ็อบเจกต์ Build หลังจากนั้นอ็อบเจกต์ Schedule จะทำการตรวจสอบว่าเวอร์ชันของ ServerBuild และ ClientBuild ตรงกันหรือไม่ กรณีที่ตรงกันอ็อบเจกต์ Schedule จะส่งข้อความไป invoke ให้อ็อบเจกต์ TestSuite เริ่มทำการติดตั้งและทดสอบ เมื่อทดสอบเสร็จเรียบร้อย อ็อบเจกต์ TestSuite จะส่งอีเมลแจ้งผลการทดสอบไปยัง Automated Test Executor



รูปที่ 4.20 ซีเควนซ์ไดอะแกรมการทดสอบบนเครื่องทดสอบแต่ละเครื่อง (Execute Test)

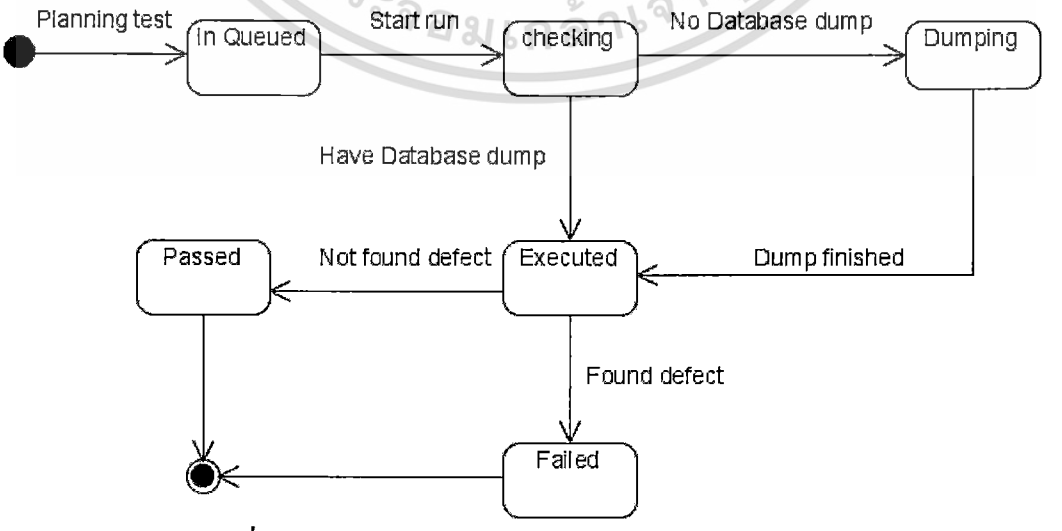
4.3.4 สเตทชาร์ตไดอะแกรม

จากการออกแบบคลาสไดอะแกรมจะเห็นได้ว่าอ็อบเจกต์ Schedule จะมีสถานะของอ็อบเจกต์ที่เปลี่ยนไปตามเหตุการณ์ที่เกิดขึ้น โดยสถานะของการทำงานสามารถแสดงได้โดยใช้สเตทชาร์ตไดอะแกรมดังรูปที่ 4.21 ถึง 4.22



รูปที่ 4.21 สเตทชาร์ตไดอะแกรมของอ็อบเจกต์ Schedule

จากรูปเมื่อผู้ทดสอบได้ตั้งเวลาการทดสอบแล้ว ระบบจะทำการตรวจสอบ (Checking) ข้อมูลสภาพแวดล้อม (Environment Data) ที่จำเป็นต้องใช้ในการทดสอบ ถ้าตรวจสอบแล้วพบว่ามีข้อมูลสภาพแวดล้อมบางตัวยังไม่พร้อมใช้งาน ระบบจะต้องรอ (Waiting) ให้ข้อมูลสภาพแวดล้อมทุกตัวพร้อมใช้งานเสียก่อน ซึ่งระบบจะมีการกำหนดระยะเวลาของการรอข้อมูลสภาพแวดล้อมเอาไว้ ถ้าครบระยะเวลาที่กำหนดไว้และข้อมูลสภาพแวดล้อมทั้งหมดยังไม่พร้อมใช้งาน ระบบจะสิ้นสุดการทำงาน (Stopped) แต่ถ้าระบบทำการตรวจสอบข้อมูลสภาพแวดล้อมและพบว่าข้อมูลสภาพแวดล้อมทั้งหมดพร้อมใช้งาน ระบบจะเริ่มทำการทดสอบ (Executing) ตามที่ได้วางแผนการทดสอบไว้และเมื่อสิ้นสุดการทดสอบระบบจะหยุดทำงาน (Finished)



รูปที่ 4.21 สเตทชาร์ตไดอะแกรมของอ็อบเจกต์ Planning

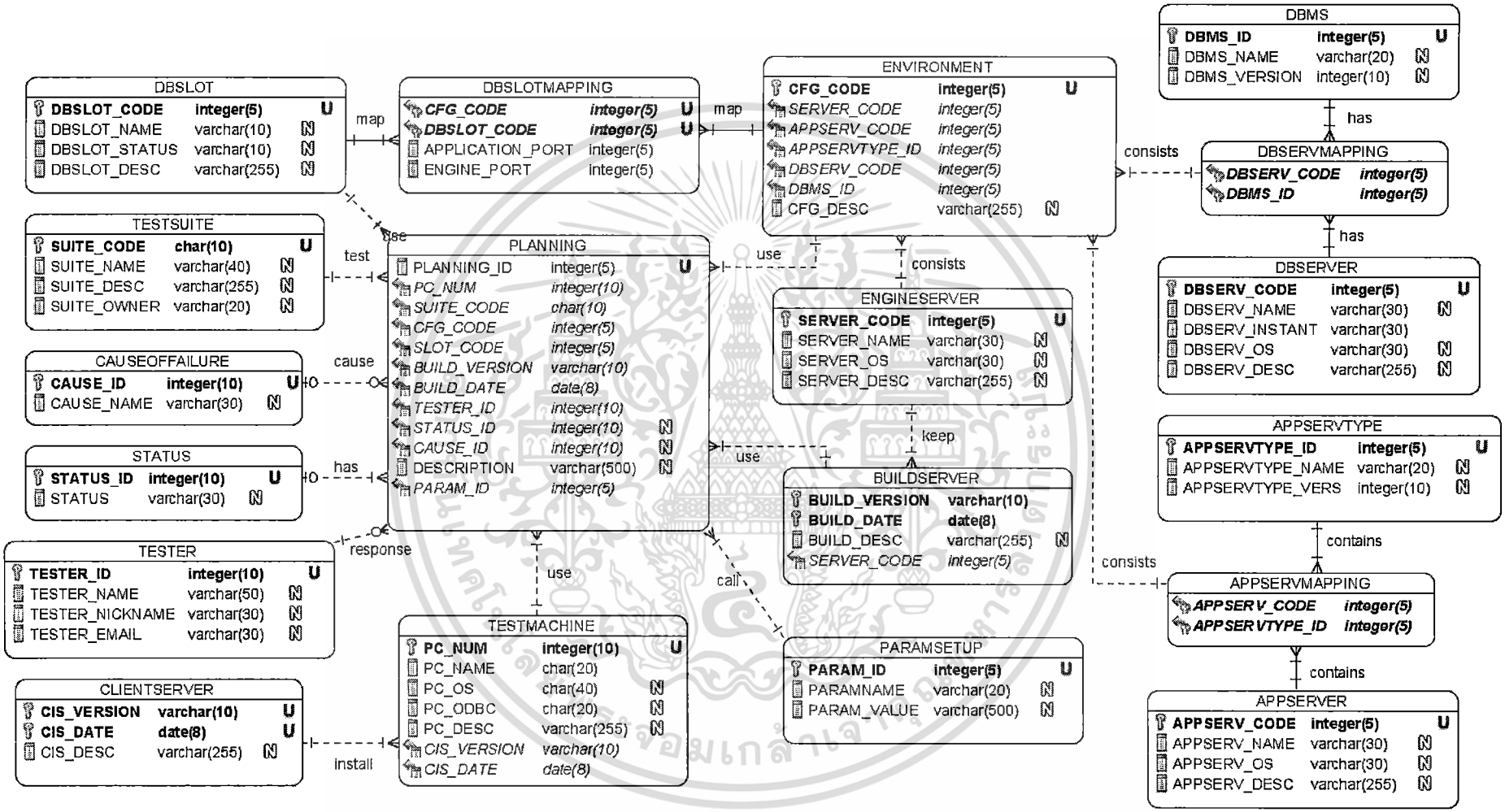
จากรูปเมื่อผู้ทดสอบได้ทำการวางแผนสคริปท์ทดสอบ (Test Suite) สถานะค่าเริ่มต้นของสคริปท์ทดสอบจะถูกกำหนดให้เป็นอยู่ในคิว (In Queued) จากนั้นเมื่อระบบได้เริ่มทำงานตามที่ได้ตั้งเวลาไว้ ระบบจะทำการตรวจสอบว่าสคริปท์ทดสอบที่ทดสอบมีตัวฐานข้อมูล Database dump) อยู่หรือไม่ ในกรณีที่ไม่มีระบบจะต้องทำการสร้างฐานข้อมูล (Dumping) ของสคริปท์ทดสอบก่อน แต่ถ้าตรวจสอบแล้วพบว่ามีอยู่แล้ว ระบบจะเริ่มดำเนินการทดสอบ (Executed) เมื่อการทดสอบเสร็จสิ้นระบบจะไปทำการอัพเดตผลการทดสอบของสคริปท์ทดสอบนั้นๆ ถ้าพบข้อบกพร่อง (Defect) ของแอพพลิเคชันที่ทดสอบ ระบบจะเปลี่ยนสถานะของสคริปท์ทดสอบเป็นไม่ผ่าน (Failed) แต่ถ้าไม่พบข้อบกพร่องของแอพพลิเคชันที่ทดสอบระบบจะเปลี่ยนสถานะของสคริปท์ทดสอบเป็นผ่าน (Passed)

บทที่ 5
การออกแบบฐานข้อมูล

การออกแบบฐานข้อมูลของแอพพลิเคชันสำหรับเตรียมสภาพแวดล้อมและตั้งเวลาทดสอบ ได้ออกแบบตามหลักการฐานข้อมูลเชิงสัมพันธ์ โดยการสร้างแบบจำลองฐานข้อมูลเชิงสัมพันธ์ หรืออีอาร์ไดอะแกรม เพื่ออธิบายถึงความสัมพันธ์ของแต่ละเอนทิตีในระบบ และได้อธิบายรายละเอียดต่างๆของข้อมูลไว้ในพจนานุกรมข้อมูล

แบบจำลองฐานข้อมูลของระบบประกอบไปด้วยเอนทิตีที่มีความสัมพันธ์กันทั้งหมด 18 เอนทิตี ดังรูปที่ 5.1 ซึ่งมีรายละเอียดดังนี้

1. TESTSUITE หมายถึง สคริปต์ที่ใช้ในการทดสอบ
2. TESTMACHINE หมายถึง เครื่องที่ใช้ในการทดสอบ
3. CLIENTSERVER หมายถึง ซอฟต์แวร์ที่ใช้ติดตั้งบนเครื่องที่ใช้ทดสอบ
4. BUILDSERVER หมายถึง ซอฟต์แวร์ที่ใช้ติดตั้งบนเครื่องเซิร์ฟเวอร์
5. ENGINESERVER หมายถึง เครื่องเซิร์ฟเวอร์ โดยจะแบ่งว่าเป็นระบบปฏิบัติการ Unix หรือ Windows Server 2003 เป็นต้น
6. DBSERVER หมายถึง เครื่องดาต้าเบสเซิร์ฟเวอร์
7. DBMS หมายถึง ระบบการจัดการฐานข้อมูล ซึ่งประกอบไปด้วย Sybase และ Oracle เวอร์ชันต่างๆ
8. DBSERVMAPPING หมายถึง เครื่องดาต้าเบสเซิร์ฟเวอร์ที่มีระบบการจัดการฐานข้อมูล (DBMS) เป็น Sybase หรือ Oracle
9. APPSERVER หมายถึง เครื่องแอพพลิเคชันเซิร์ฟเวอร์
10. APPSERVTYPE หมายถึง โดเมนของแอพพลิเคชันเซิร์ฟเวอร์ประเภทต่างๆ เช่น WebLogic หรือ WebSphere เวอร์ชันต่างๆ
11. APPSERVMAPPING หมายถึง เครื่องแอพพลิเคชันเซิร์ฟเวอร์ที่มีโดเมนชนิดต่างๆ เช่น Weblogic หรือ WebSphere เป็นต้น
12. ENVIRONMENT หมายถึง สภาพแวดล้อมของการทดสอบที่เกิดจากการรวมกันของ ENGINESERVER, DBSERVER และ APPSERVER
13. DBSLOT หมายถึง Slot ฐานข้อมูลที่เอาไว้สำหรับรองรับ MASTER DB ที่จะโหลดเข้ามา
14. DBSLOTMAPPING หมายถึง การ Map กันระหว่าง Environment กับ Database Slot
15. STATUS หมายถึง สถานะของผลการทดสอบ



รูปที่ 5.1 แบบจำลองฐานข้อมูลเชิงสัมพันธ์ของแอพพลิเคชันสำหรับเตรียมสภาพแวดล้อมและตั้งเวลาทดสอบ

16. CAUSEOFFAILURE หมายถึง สาเหตุของการทดสอบที่มีข้อบกพร่อง เนื่องจากการบกพร่องของการทดสอบนั้น สามารถเกิดได้หลายๆสาเหตุ

17. PARAMSETUP หมายถึง พารามิเตอร์และตัวแปรต่างๆที่ใช้ในการทดสอบ

18. TESTER หมายถึง ผู้ทดสอบที่มีหน้าที่รับผิดชอบการตั้งเวลาทดสอบในครั้งนั้นๆ

19. PLANNING หมายถึง ตารางการทดสอบ โดยมีรายละเอียดต่างๆ ของการทดสอบ

จากรูปที่ 5.1 เอนทิตีที่มีความสัมพันธ์กันทั้งหมด 16 เอนทิตี สามารถอธิบายความสัมพันธ์ระหว่างเอนทิตี ได้ดังต่อไปนี้

- เอนทิตี DBSLOT กับเอนทิตี DBSLOTMAPPING กับเอนทิตี ENVIRONMENT หมายความว่า ในการทดสอบครั้งหนึ่งๆ จะทดสอบบนสภาพแวดล้อมที่แตกต่างกัน เช่น ทดสอบบนระบบปฏิบัติการยูนิกซ์ ลีนุกซ์ หรือวินโดว์ เป็นต้น ซึ่ง Database Slot ที่ใช้ทดสอบจะแตกต่างกันขึ้นอยู่กับ Environment ที่ทดสอบ

- เอนทิตี DBSERVER กับเอนทิตี DBSERVMAPPING กับเอนทิตี DBMS หมายความว่า เครื่องดาต้าเบสเซิร์ฟเวอร์ที่ใช้จะมีระบบการจัดการฐานข้อมูลอยู่ภายในเครื่องดาต้าเบสเซิร์ฟเวอร์นั้น ซึ่งเครื่องดาต้าเบสเซิร์ฟเวอร์ 1 เครื่องมีระบบการจัดการฐานข้อมูลได้หลายตัว และระบบการจัดการฐานข้อมูลหนึ่งชนิด สามารถติดตั้งบนเครื่องดาต้าเบสเซิร์ฟเวอรได้หลายเครื่อง

- เอนทิตี APPSERVER กับเอนทิตี APPSERVMAPPING กับเอนทิตี APPSERVTYPE หมายความว่า เครื่องแอพพลิเคชันเซิร์ฟเวอร์ที่ใช้จะมีการติดตั้งโดเมนอยู่ภายในเครื่องแอพพลิเคชันเซิร์ฟเวอร์นั้น ซึ่งเครื่องแอพพลิเคชันเซิร์ฟเวอร์ 1 เครื่องมีโดเมนได้หลายชนิดและหลายเวอร์ชัน และโดเมนหนึ่งโดเมนสามารถติดตั้งบนเครื่องแอพพลิเคชันเซิร์ฟเวอร์ได้หลายเครื่อง

- เอนทิตี DBSERVER กับเอนทิตี ENVIRONMENT หมายความว่า สภาพแวดล้อมของการทดสอบจะประกอบไปด้วยเครื่องดาต้าเบสเซิร์ฟเวอร์ 1 เครื่อง และเครื่องดาต้าเบสเซิร์ฟเวอร์ 1 เครื่อง อาจใช้ในการทดสอบได้หลายๆสภาพแวดล้อม

- เอนทิตี APPSERVER กับเอนทิตี ENVIRONMENT หมายความว่า สภาพแวดล้อมของการทดสอบจะประกอบไปด้วยเครื่องแอพพลิเคชันเซิร์ฟเวอร์1 เครื่อง และเครื่องแอพพลิเคชันเซิร์ฟเวอร์ 1 เครื่อง อาจใช้ในการทดสอบได้หลายๆสภาพแวดล้อม

- เอนทิตี ENGINESERVER กับเอนทิตี ENVIRONMENT หมายความว่า สภาพแวดล้อมของการทดสอบจะประกอบไปด้วยเครื่องเอนจิ้นเซิร์ฟเวอร์1 เครื่อง และเครื่องเอนจิ้นเซิร์ฟเวอร์ 1 เครื่อง อาจใช้ในการทดสอบได้หลายๆสภาพแวดล้อม

- เอนทิตี ENGINESERVER กับเอนทิตี BUILDSERVER หมายความว่า ซอฟต์แวร์ที่ใช้ทดสอบที่ติดตั้งบนเครื่องเซิร์ฟเวอร์นั้นจะถูกเก็บไว้ที่เครื่องเอนจิ้นเซิร์ฟเวอร์ โดยซอฟต์แวร์ที่เก็บนั้นจะมีด้วยกันหลายเวอร์ชัน

- เอนทิตี PLANNING กับเอนทิตี TESTSUITE หมายความว่า ในตารางการวางแผนการทดสอบจะต้องมีการระบุตัวทดสอบหรือ Test Suite ที่ต้องการทดสอบ โดยในการวางแผนการทดสอบ 1 แผนสามารถทดสอบ Test Suite ได้หลายตัว

- เอนทิตี PLANNING กับเอนทิตี CAUSEOFFAILURE หมายความว่า ในตารางการวางแผนการทดสอบสามารถระบุสาเหตุของข้อบกพร่องที่พบได้ หรือจะไม่ระบุก็ได้ถ้าผลการทดสอบผ่าน

- เอนทิตี PLANNING กับเอนทิตี STATUS หมายความว่า ในตารางการวางแผนการทดสอบจะต้องมีการระบุสถานะของตัวทดสอบนั้นๆ ว่ามีสถานะหรือผลการทดสอบเป็นอย่างไร เช่น Running , In Queue , Passed หรือ Failed เป็นต้น

- เอนทิตี PLANNING กับเอนทิตี BUILDSERVER หมายความว่า ในตารางแผนการทดสอบจะต้องมีการระบุเวอร์ชันและวันที่ของซอฟต์แวร์ที่จะทดสอบ ซึ่งตารางแผนการทดสอบ สามารถทดสอบได้หลายเวอร์ชันพร้อมกัน

- เอนทิตี PLANNING กับเอนทิตี ENVIRONMENT หมายความว่า ในตารางแผนการทดสอบจะต้องมีการระบุสภาพแวดล้อมของการทดสอบ ซึ่งตารางการทดสอบสามารถทดสอบได้หลายๆสภาพแวดล้อม

- เอนทิตี PLANNING กับเอนทิตี DBSLOT หมายความว่า ในตารางการวางแผนการทดสอบจะต้องมีการระบุ Databaase Slot ที่จะใช้ในการโหลดข้อมูล โดยในการวางแผนการทดสอบ 1 แผนสามารถใช้ Database Slot ได้หลาย Slot

- เอนทิตี PLANNING กับเอนทิตี PARAMSETUP หมายความว่า ในตารางการวางแผนการทดสอบจะต้องมีการตั้งค่าในการทดสอบ โดยการตั้งค่าการทดสอบ 1 ครั้ง สามารถใช้กับแผนทดสอบได้หลายๆแผน

- เอนทิตี PLANNING กับเอนทิตี TESTER หมายความว่า ในตารางการวางแผนการทดสอบจะต้องมีการระบุผู้รับผิดชอบในการทดสอบ โดยในการวางแผนการทดสอบ 1 แผนสามารถมีผู้รับผิดชอบในการทดสอบได้มากกว่า 1 คน

- เอนทิตี TESTMACHINE กับเอนทิตี CLIENTSERVER หมายความว่า เครื่องที่ใช้รันตัวทดสอบจะต้องทำการติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ โดยเครื่องทดสอบ 1 เครื่องจะติดตั้งซอฟต์แวร์ได้ครั้งละ 1 เวอร์ชัน และซอฟต์แวร์ 1 เวอร์ชันสามารถติดตั้งบนเครื่องทดสอบได้หลายๆเครื่อง

จากแบบจำลองฐานความสัมพันธ์ระหว่างเอนทิตี สามารถอธิบายรายละเอียดของแต่ละเอนทิตีด้วยพจนานุกรมข้อมูล ดังตารางที่ 5.1 ถึงตารางที่ 5.18

ตารางที่ 5.1 พจนานุกรมข้อมูลของตาราง TESTSUITE

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
SUITE_CODE	รหัสตัวทดสอบ	char(10)	PK	
SUITE_NAME	ชื่อตัวทดสอบ	varchar(40)		
SUITE_DESC	รายละเอียดตัวทดสอบ	varchar(255)		
SUITE_OWNER	ชื่อเจ้าของตัวทดสอบ	varchar(20)		

ตารางที่ 5.2 พจนานุกรมข้อมูลของตาราง BUILDSERVER

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
BUILD_VERSION	เวอร์ชันของ Build	varchar(10)	PK	
BUILD_DATE	วันที่ของ Build	date(8)	PK	
BUILD_DESC	วันที่ของ Build	varchar(255)		
SERVER_CODE	รหัสเครื่องเซิร์ฟเวอร์	integer(5)	FK	ENGINESERVER

ตารางที่ 5.3 พจนานุกรมข้อมูลของตาราง CLIENTSERVER

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
CIS_VERSION	เวอร์ชันของ CIS	varchar(10)	PK	
CIS_DATE	วันที่ของ CIS	date(8)	PK	
CIS_DESC	รายละเอียดของ CIS	varchar(255)		

ตารางที่ 5.4 พจนานุกรมข้อมูลของตาราง TESTMACHINE

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
PC_NUM	รหัสเครื่องทดสอบ	integer(10)	PK	
PC_NAME	ชื่อเครื่องทดสอบ	char(20)		
PC_OS	ระบบปฏิบัติการของเครื่องทดสอบ	char(40)		
PC_ODBC	เวอร์ชันของ Sybase	char(20)		
PC_DESC	รายละเอียดของเครื่องทดสอบ	varchar(255)		
PC_DESC	รายละเอียดของเครื่องทดสอบ	varchar(255)		

ตารางที่ 5.4 (ต่อ)

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
CIS_DATE	วันที่ของ CIS	date(8)		

ตารางที่ 5.5 พจนานุกรมข้อมูลของตาราง PLANNING

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
PLANNING_ID	รหัสแผนการทดสอบ	integer(5)	PK	
PC_NUM	รหัสเครื่องทดสอบ	integer(10)	FK	TESTMACHINE
SUITE_CODE	รหัสตัวทดสอบ	varchar(10)	FK	TESTSUITE
CFG_CODE	รหัสคอนฟิกุเรชัน	integer(5)	FK	ENVIRONMENT
SLOT_CODE	รหัส database slot	integer(5)	FK	DBSLOT
BUILD_VERSION	เวอรชันซอฟต์แวร์	varchar(10)	FK	BUILDSERVER
BUILD_DATE	วันที่ซอฟต์แวร์	date(8)		
TESTER_ID	ผู้รับผิดชอบ	integer(10)	FK	TESTER
STATUS_ID	รหัสสถานะ	integer(10)	FK	STATUS
CAUSE_ID	รหัสสาเหตุ	integer(10)	FK	CAUSEOFFAILURE
DESCRIPTION	รายละเอียดของแผน	varchar(500)		

ตารางที่ 5.6 พจนานุกรมข้อมูลของตาราง TESTER

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
TESTER_ID	รหัสผู้รับผิดชอบ	integer(10)	PK	
TESTER_NAME	ชื่อผู้รับผิดชอบ	varchar(50)		
TESTER_NICKNAME	ชื่อเล่นผู้รับผิดชอบ	varchar(30)		
TESTER_EMAIL	อีเมลผู้รับผิดชอบ	varchar(30)		

ตารางที่ 5.7 พจนานุกรมข้อมูลของตาราง DBMS

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
DBMS_ID	รหัสระบบการจัดการฐานข้อมูล	integer(5)	PK	
DBMS_NAME	ชื่อระบบการจัดการฐานข้อมูล	varchar(20)		
DBMS_VERSION	เวอร์ชันของระบบการจัดการฐานข้อมูล	integer(10)		

ตารางที่ 5.8 พจนานุกรมข้อมูลของตาราง DBSERVER

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
DBSERV_CODE	รหัสเครื่องดาต้าเบส เซิร์ฟเวอร์	integer(5)	PK	
DBSERV_NAME	ชื่อเครื่องดาต้าเบส เซิร์ฟเวอร์	varchar(30)		
DBSERV_OS	ระบบปฏิบัติการของ เครื่องดาต้าเบส เซิร์ฟเวอร์	varchar(30)		
DBSERV_DESC	รายละเอียดเครื่องดาต้า เบสเซิร์ฟเวอร์	varchar(255)		

ตารางที่ 5.9 พจนานุกรมข้อมูลของตาราง DBSERVMAPPING

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
DBSERV_CODE	รหัสเครื่องดาต้าเบส เซิร์ฟเวอร์	integer(5)	PK/ FK	DBSERVER
DBMS_ID	รหัสระบบการจัดการ ฐานข้อมูล	integer(5)	PK/ FK	DBMS

ตารางที่ 5.10 พจนานุกรมข้อมูลของตาราง APPSERVER

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
APPSERV_CODE	รหัสเครื่องแอพพลิเคชัน เซิร์ฟเวอร์	integer(5)	PK	
APPSERV_NAME	ชื่อเครื่องแอพพลิเคชัน เซิร์ฟเวอร์	varchar(30)		
APPSERV_OS	ระบบปฏิบัติการของ แอพพลิเคชันเซิร์ฟเวอร์	varchar(30)		
APPSERV_DESC	รายละเอียดเครื่องดาต้า เบสเซิร์ฟเวอร์	varchar(255)		

ตารางที่ 5.11 พจนานุกรมข้อมูลของตาราง APPSERVTYPE

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
APPSERVTYPE_ID	รหัสชนิดแอพพลิเคชัน เซิร์ฟเวอร์	integer(5)	PK	

ตารางที่ 5.11 (ต่อ)

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
APPSERVTYPE_NAME	ชื่อชนิดแอพพลิเคชันเซิร์ฟเวอร์	varchar(20)		
APPSERVTYPE_VERS	เวอร์ชันชนิดแอพพลิเคชันเซิร์ฟเวอร์	integer(10)		

ตารางที่ 5.12 พจนานุกรมข้อมูลของตาราง APPSERVMAPPING

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
APPSERV_CODE	รหัสเครื่องแอพพลิเคชันเซิร์ฟเวอร์	integer(5)	PK/ FK	APPSERVER
APPSERVTYPE_ID	รหัสชนิดแอพพลิเคชันเซิร์ฟเวอร์	integer(5)	PK/ FK	APPSERVTYPE

ตารางที่ 5.13 พจนานุกรมข้อมูลของตาราง ENVIRONMENT

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
CFG_CODE	รหัสคอนฟิกุเรชัน	integer(5)	PK	
SERVER_CODE	รหัสเครื่องเซิร์ฟเวอร์	integer(5)	FK	ENGINESERVER
APPSERV_CODE	รหัสเครื่องแอพพลิเคชันเซิร์ฟเวอร์	integer(5)	FK	APPSERVMAPPING
APPSERVTYPE_ID	รหัสประเภทแอพพลิเคชันเซิร์ฟเวอร์	integer(5)		
DBSERV_CODE	รหัสเครื่องดาต้าเบสเซิร์ฟเวอร์	integer(5)	FK	DBSERVMAPPING
DBMS_ID	รหัสระบบการจัดการฐานข้อมูล	integer(5)		
CFG_DESC	รายละเอียดของคอนฟิกุเรชัน	varchar(255)		

ตารางที่ 5.14 พจนานุกรมข้อมูลของตาราง CAUSEOFFAILURE

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
CAUSE_ID	รหัสสาเหตุข้อบกพร่อง	integer(10)	PK	
CAUSE_NAME	ชื่อสาเหตุข้อบกพร่อง	varchar(30)		

ตารางที่ 5.15 พจนานุกรมข้อมูลของตาราง ENGINESERVER

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
SERVER_CODE	รหัสเครื่องเซิร์ฟเวอร์	integer(5)	PK	
SERVER_NAME	ชื่อเครื่องเซิร์ฟเวอร์	varchar(30)		
SERVER_OS	ระบบปฏิบัติการของเครื่องเซิร์ฟเวอร์	varchar(30)		
SERVER_DESC	รายละเอียดเครื่องเซิร์ฟเวอร์	varchar(255)		

ตารางที่ 5.16 พจนานุกรมข้อมูลของตาราง DBSLOT

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
DBSLOT_CODE	รหัส Database Slot	integer(5)	PK	
DBSLOT_NAME	ชื่อ Database Slot	char(20)		
DBSLOT_STATUS	สถานะของ Database Slot	integer(1)		
DBSLOT_DESC	รายละเอียด Database Slot	varchar(255)		

ตารางที่ 5.17 พจนานุกรมข้อมูลของตาราง DBSLOTMAPPING

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
CFG_CODE	รหัสคอนฟิกุเรชัน	integer(5)	PK/ FK	ENVIRONMENT
DBSLOT_CODE	รหัส Database Slot	integer(5)	PK/ FK	DBSLOT
APPLICATION_PORT	เลขแอพพลิเคชันพอร์ต	integer(5)		
ENGINE_PORT	เลขเอนจิ้นพอร์ต	integer(5)		

ตารางที่ 5.18 พจนานุกรมข้อมูลของตาราง STATUS

แอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์	ตารางอ้างอิง
STATUS_ID	รหัสสถานะ	integer(10)	PK	
STATUS	ชื่อสถานะ	varchar(30)		

บทที่ 6
การออกแบบส่วนต่อประสานกับผู้ใช้

การออกแบบส่วนต่อประสานกับผู้ใช้มีวัตถุประสงค์เพื่อให้ผู้ใช้สามารถทำความเข้าใจและใช้งานได้ง่าย โดยจะช่วยให้ผู้ใช้เห็นถึงส่วนต่างๆ ของระบบทำให้ช่วยอำนวยความสะดวกและสนับสนุนผู้ใช้มากที่สุด เพื่อลดข้อผิดพลาดที่เกิดขึ้นในการใช้งาน สำหรับการออกแบบส่วนต่อประสานกับผู้ใช้นี้เน้นการนำไปใช้ประโยชน์ให้เกิดประสิทธิภาพสูงสุดและง่ายต่อการใช้งาน

6.1 Hierarchical Menu

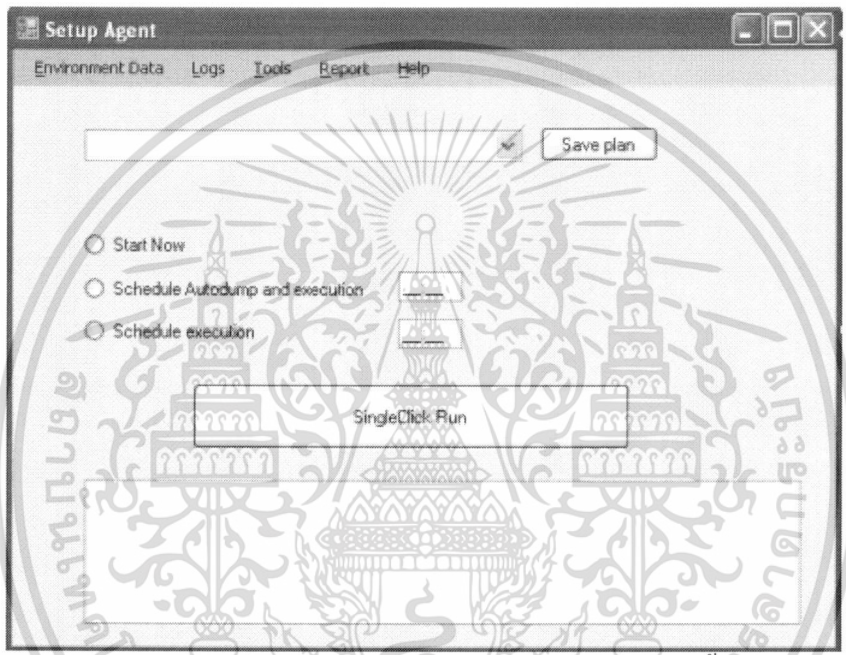
แอพพลิเคชันสำหรับการเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบนั้นมีโครงสร้างของเมนูตามลำดับชั้น ดังนี้

- **Environment Data**
  - Build
  - Test Machine
  - Test Suite
  - Application Server
  - Database Server
  - Engine Server
  - Report Server
  - Configuration Mapping
  - Database Slot
  - DBSlot Mapping
- **Logs**
  - Autodump backup log
  - Autodump restore log
  - Autodump update log
  - Save log file
- **Tools**
  - Setting
  - Planning
  - Database checking
  - Disk space checking

- **Report**
  - Daily Run Report
  - Summary Report

6.2 หน้าจอการทำงานหลัก

หน้าจอการทำงานหลักที่เกี่ยวข้องกับการตั้งเวลาการทดสอบนั้น ประกอบด้วยหน้าจอที่สำคัญ ดังรูป 6.1



รูปที่ 6.1 หน้าจอแอพพลิเคชันสำหรับการเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบ

6.2.1 หน้าจอเกี่ยวกับการตั้งเวลาทดสอบ

เมื่อเข้าสู่ระบบจะพบหน้าจอของการตั้งเวลาการทดสอบ ซึ่งในการตั้งเวลาการทดสอบนั้นมีด้วยกัน 3 รูปแบบ ได้แก่

1) Start Now

เป็นรูปแบบที่จะเริ่มทดสอบทันทีเมื่อกดปุ่ม “SingleClick Run” รูปแบบนี้เหมาะสำหรับกรณีที่มีความพร้อมในการทดสอบทุกอย่าง และเวลาทดสอบเพียงพอ

2) Schedule AutoDump and Execution

เป็นรูปแบบที่สามารถกำหนดเวลาทดสอบล่วงหน้าได้ รูปแบบนี้เหมาะสำหรับกรณีที่ข้อมูลใน Database Slot เป็นข้อมูลเก่าที่ไม่ตรงกับเวอร์ชันของ Build ที่จะทดสอบ และยังไม่พร้อมที่จะทดสอบ ณ ขณะนั้น ทำให้ผู้ทดสอบจำเป็นต้องตั้งเวลาทดสอบล่วงหน้าไว้ เมื่อถึงเวลาที่กำหนด

ระบบจะทำการโหลดข้อมูลเข้า Database Slot และอัพเดตข้อมูลให้ตรงกับเวอร์ชันของ Build จากนั้นจะทำการติดตั้งซอฟต์แวร์ที่เครื่องทดสอบให้อัตโนมัติ และเริ่มดำเนินการทดสอบ

3) Schedule Execution

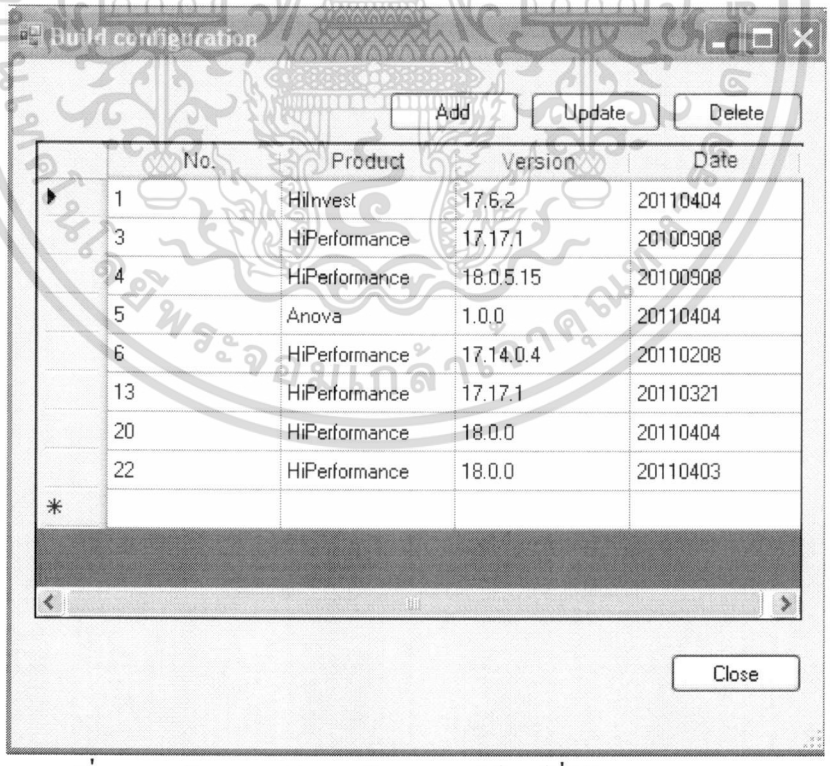
เป็นรูปแบบที่สามารถกำหนดเวลาทดสอบล่วงหน้าได้ รูปแบบนี้เหมาะสำหรับกรณีที่มีข้อมูลใน Database Slot ตรงกับเวอร์ชันของ Build ที่จะทดสอบอยู่แล้ว แต่ยังไม่พร้อมที่จะทดสอบ ณ ขณะนั้น ทำให้ผู้ทดสอบจำเป็นต้องตั้งเวลาทดสอบไว้ล่วงหน้า เมื่อถึงเวลาทดสอบระบบจะทำการติดตั้งซอฟต์แวร์ที่เครื่องทดสอบให้อัตโนมัติและเริ่มดำเนินการทดสอบ

6.2.2 เมนูข้อมูลหลัก (Master Data)

เมนูข้อมูลหลักเป็นเมนูที่เกี่ยวข้องกับข้อมูลที่จำเป็นต้องใช้ในการทดสอบแอพพลิเคชัน ซึ่งมีข้อมูลดังนี้

1. Build

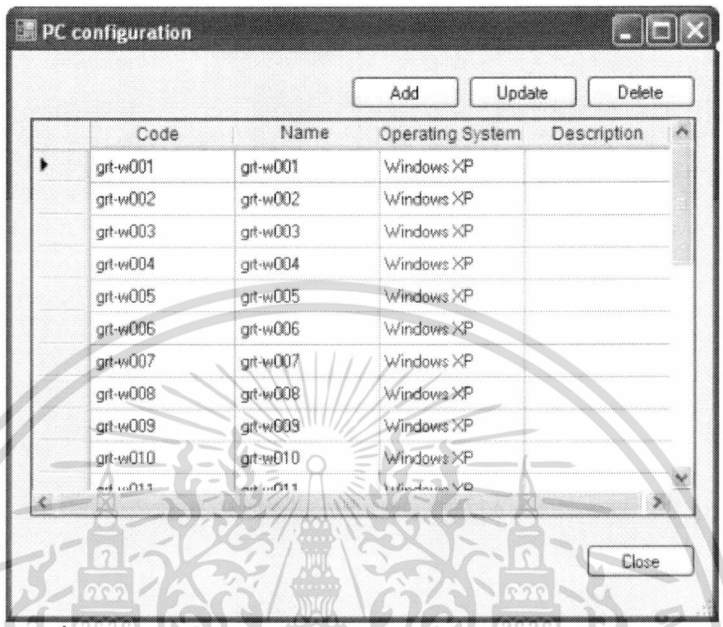
เป็นเมนูที่ใช้เข้าสู่หน้าจอการจัดการข้อมูลเวอร์ชัน วันที่ของซอฟต์แวร์ที่ใช้ทดสอบ ซึ่งสามารถทำการเพิ่ม แก้ไขและลบข้อมูล Build ได้ ดังรูป 6.2



รูปที่ 6.2 หน้าจอการจัดการข้อมูลซอฟต์แวร์ที่ใช้ทดสอบ

2. Test Machine

เป็นเมนูที่ใช้เข้าสู่หน้าจอการจัดการข้อมูลเครื่องที่ใช้ทดสอบ ซึ่งสามารถทำการเพิ่ม แก้ไข และลบข้อมูลเครื่องทดสอบได้ ดังรูป 6.3



รูปที่ 6.3 หน้าจอการจัดการข้อมูลเครื่องที่ใช้ทดสอบ

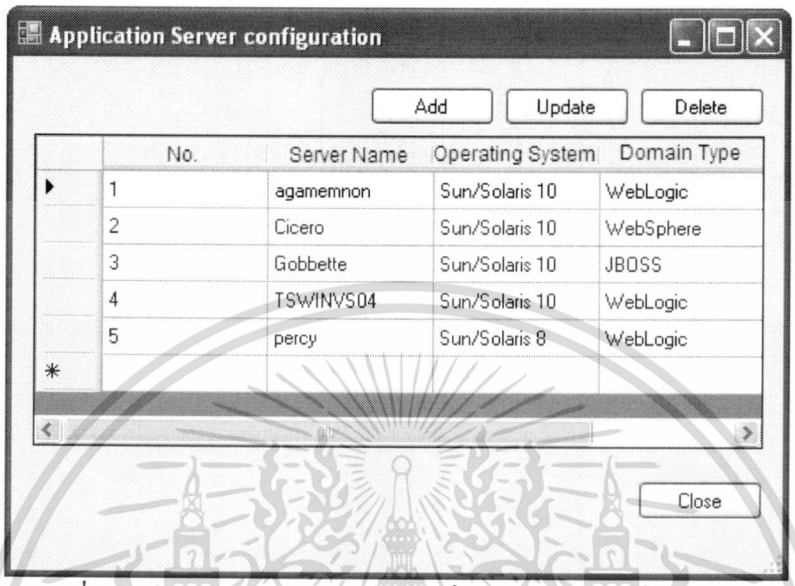
3. Test Suite

เป็นเมนูที่ใช้เข้าสู่หน้าจอการจัดการสคริปต์ที่ใช้ทดสอบ ซึ่งสามารถทำการเพิ่ม แก้ไขและลบข้อมูลสคริปต์ที่ใช้ทดสอบได้ ดังรูป 6.4

รูปที่ 6.4 หน้าจอการจัดการข้อมูลสคริปต์ที่ใช้ทดสอบ

4. Application Server

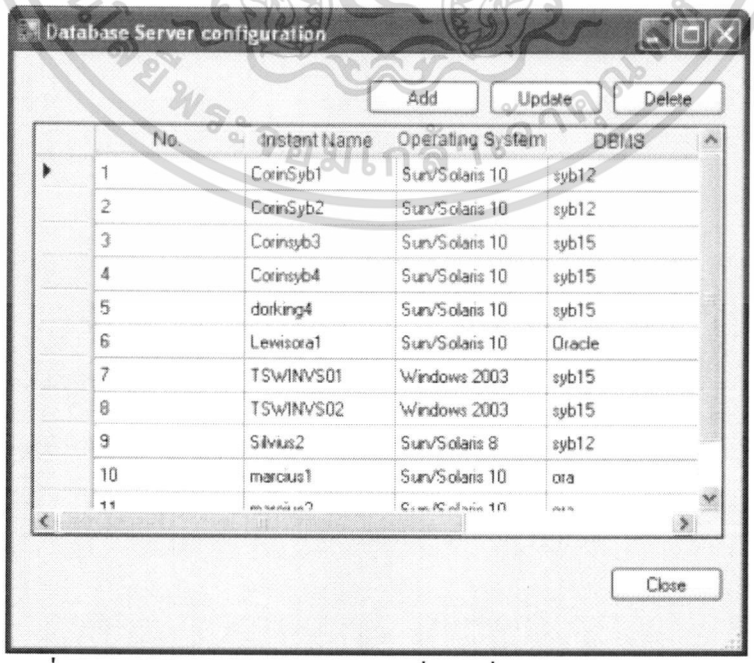
เป็นเมนูที่ใช้เข้าสู่หน้าจอการจัดการเครื่องแอพพลิเคชันเซิร์ฟเวอร์ ซึ่งสามารถทำการเพิ่ม แก้ไขและลบข้อมูลเครื่องแอพพลิเคชันเซิร์ฟเวอร์ได้ ดังรูป 6.5



รูปที่ 6.5 หน้าจอการจัดการข้อมูลเครื่องแอพพลิเคชันเซิร์ฟเวอร์

5. Database Server

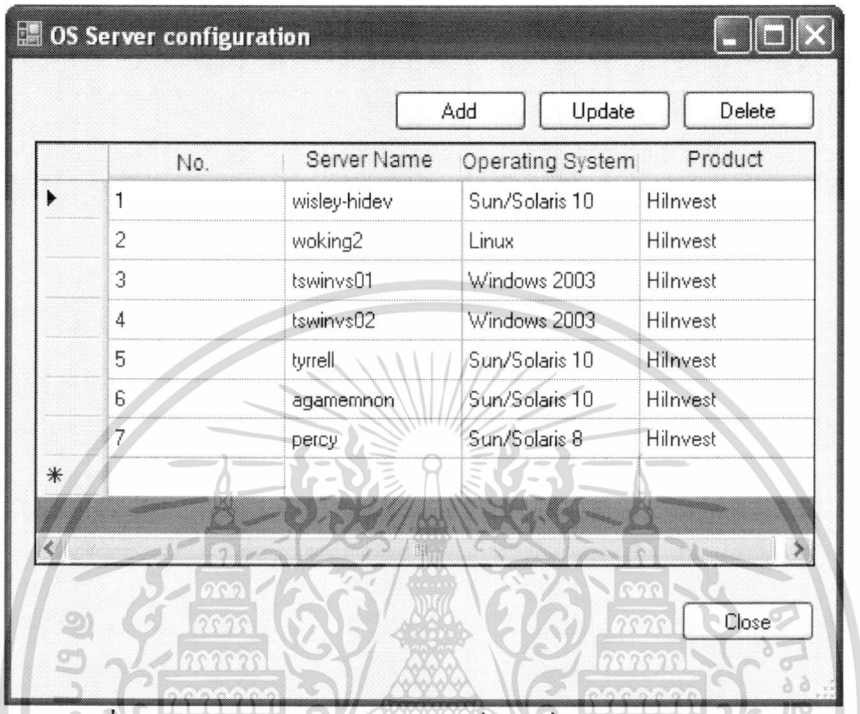
เป็นเมนูที่ใช้เข้าสู่หน้าจอการจัดการเครื่องดาต้าเบสเซิร์ฟเวอร์ ซึ่งสามารถทำการเพิ่ม แก้ไขและลบข้อมูลเครื่องดาต้าเบสเซิร์ฟเวอร์ได้ ดังรูป 6.6



รูปที่ 6.6 หน้าจอการจัดการข้อมูลเครื่องเครื่องดาต้าเบสเซิร์ฟเวอร์

6. Engine Server

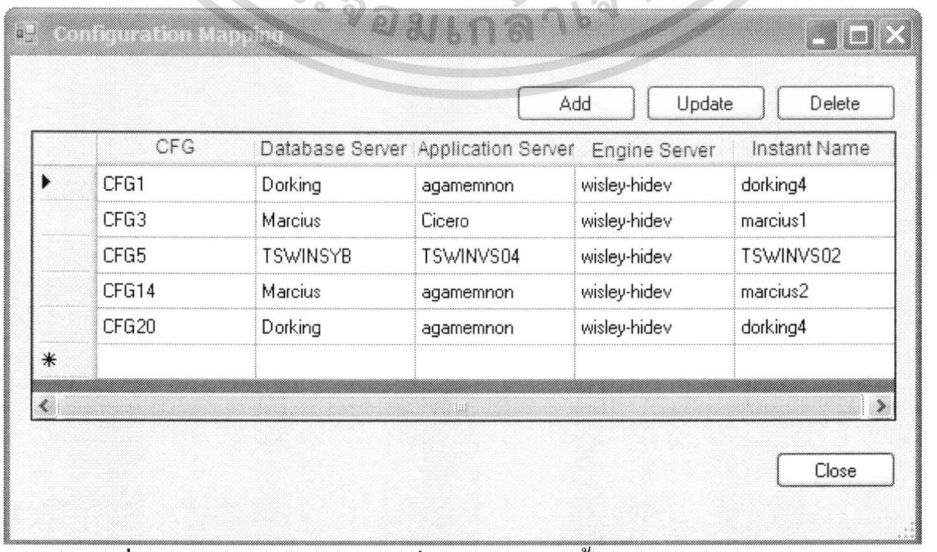
เป็นเมนูที่ใช้เข้าสู่หน้าจอการจัดการเครื่องเอนจินเซิร์ฟเวอร์ ซึ่งสามารถทำการเพิ่ม แก้ไข และลบข้อมูลเครื่องเอนจินเซิร์ฟเวอร์ได้ ดังรูป 6.7



รูปที่ 6.7 หน้าจอการจัดการข้อมูลเครื่องเครื่องเอนจินเซิร์ฟเวอร์

7. Configuration Mapping

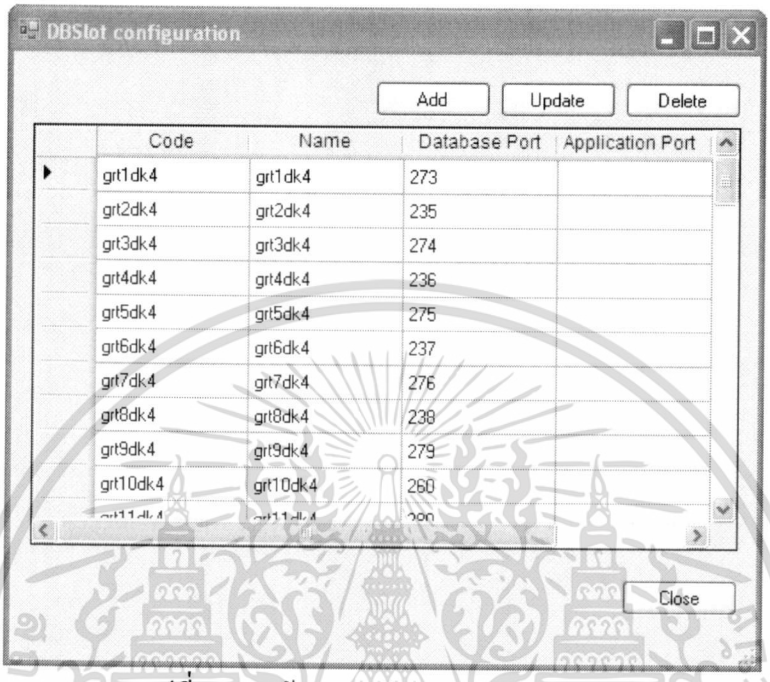
เป็นเมนูที่ใช้เข้าสู่หน้าจอการ Map เครื่องเซิร์ฟเวอร์ทั้ง 3 ประเภทเข้าด้วยกัน ประกอบกันเป็น Environment ใช้ในการทดสอบ ดังรูป 6.8



รูปที่ 6.8 หน้าจอการ Map เครื่องเซิร์ฟเวอร์ทั้ง 3 ประเภทเข้าด้วยกัน

8. Database Slot

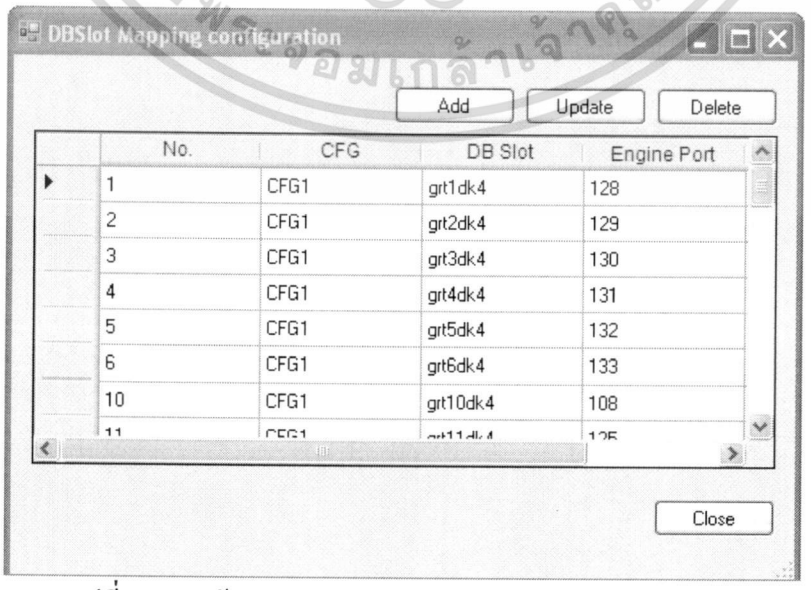
เป็นเมนูที่ใช้เข้าสู่หน้าจอการจัดการ Database Slot เพื่อใช้เป็นที่เก็บข้อมูลในการทดสอบ ซึ่งสามารถทำการเพิ่ม แก้ไข และลบได้ ดังรูป 6.9



รูปที่ 6.9 หน้าจอการจัดการ Database Slot

9. DB Slot Mapping

เป็นเมนูที่ใช้เข้าสู่หน้าจอการ Map Database Slot กับ Environment เข้าด้วยกัน ซึ่งสามารถทำการเพิ่ม แก้ไข และลบได้ ดังรูป 6.10



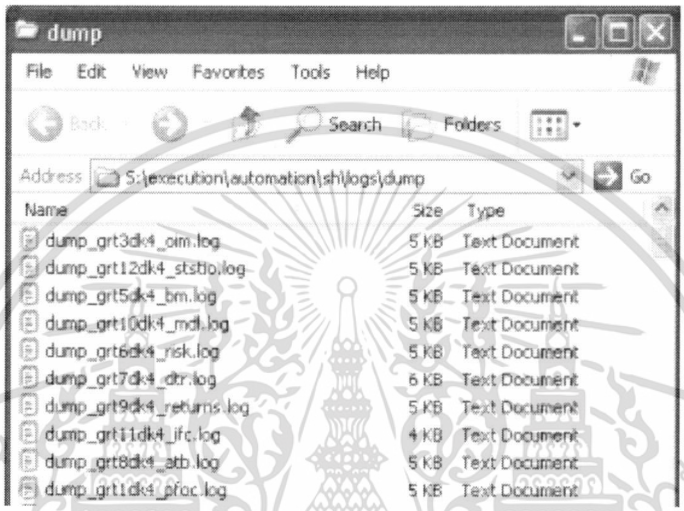
รูปที่ 6.10 หน้าจอการ Map Database Slot กับ Environment

6.2.3 เมนู Log ไฟล์ (Logs)

เมนู Log ไฟล์เป็นเมนูที่เกี่ยวข้องกับการ Log ไฟล์การโหลดข้อมูลเข้า Database Slot และข้อมูลเกี่ยวกับการทำงานของระบบ ซึ่งมีเมนูย่อยดังนี้

1. Autodump backup log

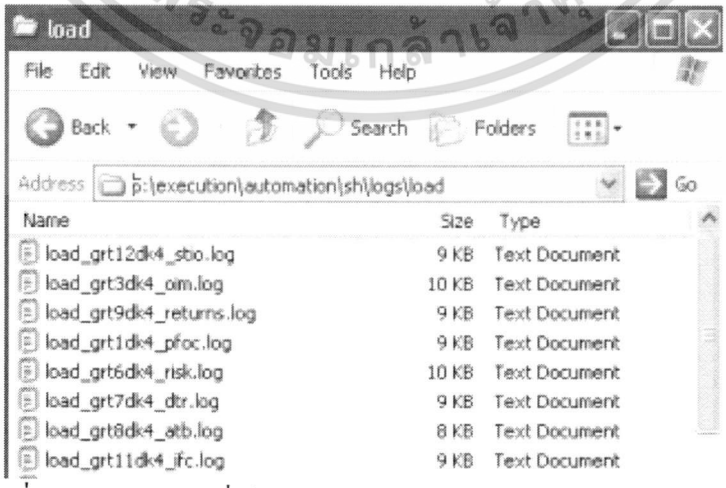
เป็นเมนูสำหรับเชื่อมต่อไปยัง Directory ที่เก็บ log ของการ Compress ข้อมูลที่อยู่ใน Database Slot ดังรูป 6.11



รูปที่ 6.11 Directory ที่เก็บ log ของการ Compress ข้อมูลที่อยู่ใน Database Slot

2. Autodump restore log

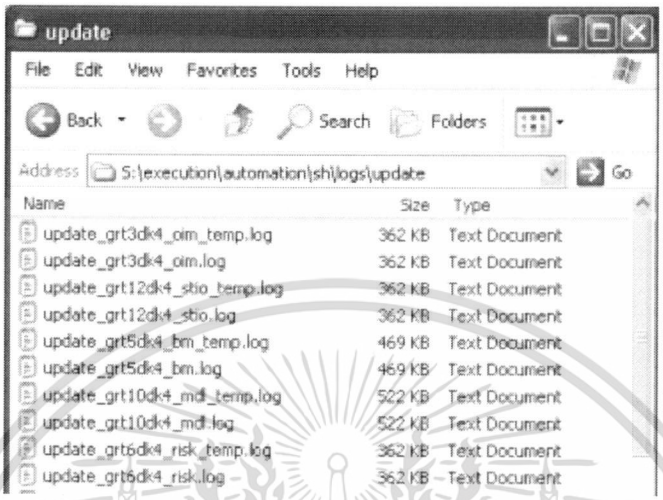
เป็นเมนูสำหรับเชื่อมต่อไปยัง Directory ที่เก็บ log ของการ load ข้อมูลเข้า Database Slot ดังรูป 6.11



รูปที่ 6.12 Directory ที่เก็บ log ของการ load ข้อมูลเข้า Database Slot

3. Autodump update log

เป็นเมนูสำหรับเชื่อมต่อไปยัง Directory ที่เก็บ log ของการอัพเดตข้อมูลใน Database Slot ดังรูป 6.13



รูปที่ 6.13 Directory ที่เก็บ log ของการอัพเดตข้อมูลใน Database Slot

6.2.4 เมนูเครื่องมือ (Tools)

เมนูเครื่องมือเป็นเมนูที่เกี่ยวข้องกับการตั้งค่าต่างๆ การวางแผน การตรวจสอบสภาพแวดล้อมที่เกี่ยวข้องกับการทดสอบ ซึ่งมีเมนูย่อยดังนี้

4. Setting

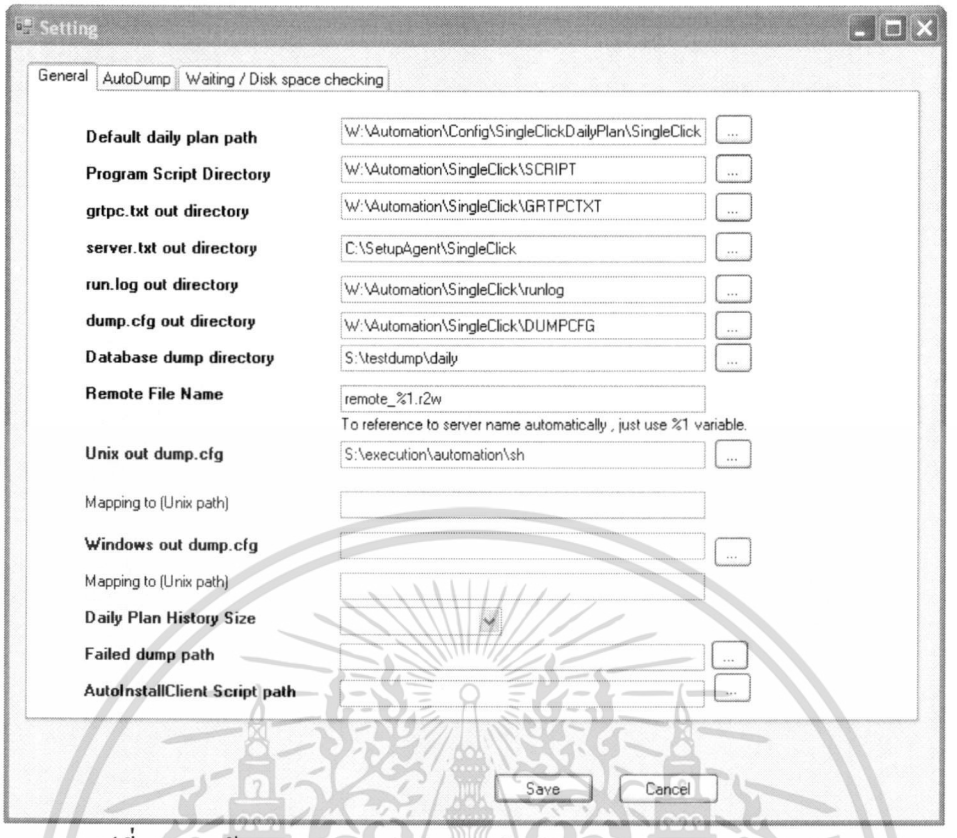
เป็นเมนูที่ใช้เข้าสู่หน้าจอการกำหนดค่าต่างๆในการทดสอบ จะประกอบไปด้วย 3 Tab ดังนี้

1. General Tab

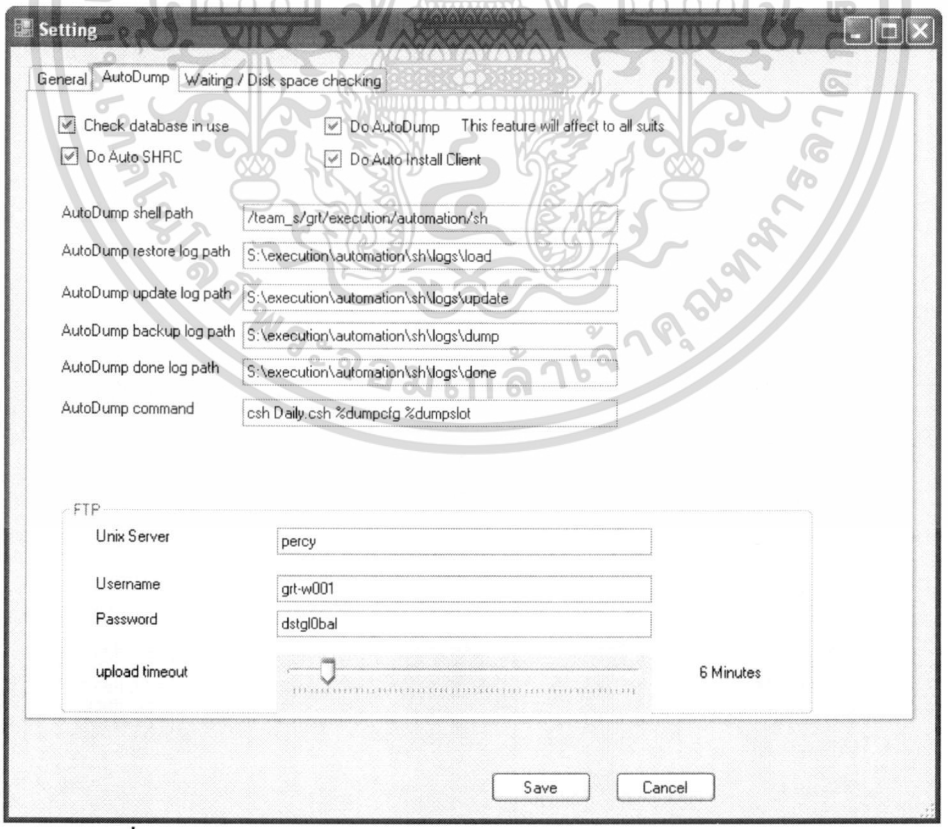
เนื่องจากในการทดสอบจะมีไฟล์ที่จำเป็นเกี่ยวข้องเป็นจำนวนมาก ซึ่งการกำหนดค่าภายในไฟล์ต่างๆจะกำหนดเพียงครั้งแรกครั้งเดียวเท่านั้น ยกเว้นไฟล์ในส่วนของ Daily Plan จำเป็นต้องทำการวางแผนการทดสอบทุกวัน มีการปรับเปลี่ยนทุกครั้ง ถ้าในการบันทึกชื่อไฟล์ใช้เป็นชื่อไฟล์เดิมตลอด จะไม่จำเป็นต้องทำการแก้ไขข้อมูลใน General Tab ดังรูป 6.14

2. AutoDump Tab

หน้าจอสำหรับกำหนดว่าต้องการให้ทำการโหลดข้อมูลเข้าสู่ Database Slot หรือไม่ และสามารถระบุ directory ที่ต้องการให้ระบบเก็บ log การทำงานของ Database Slot ได้ ดังรูป 6.15



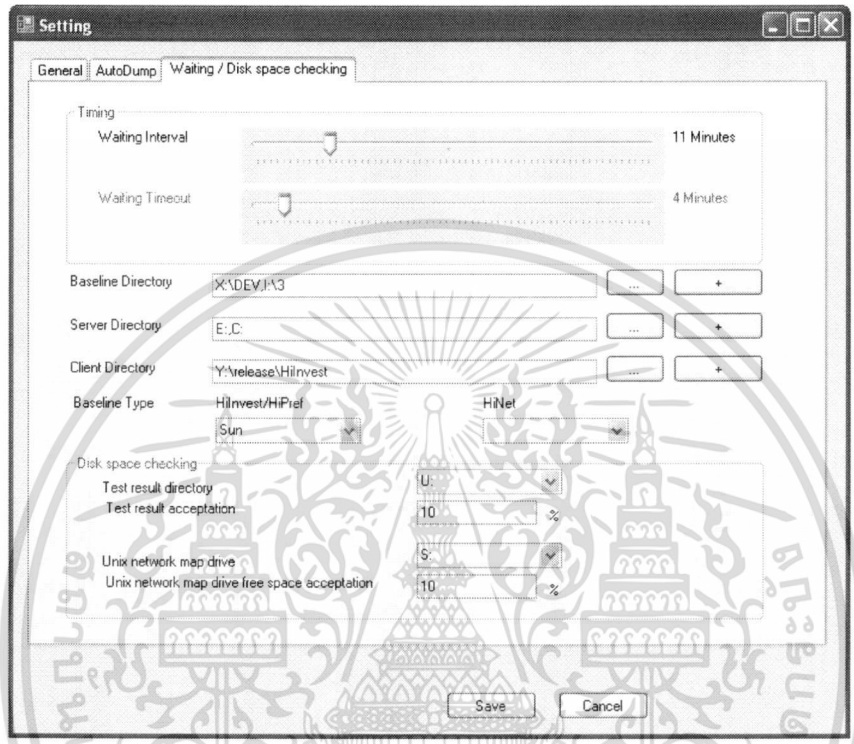
รูปที่ 6.14 หน้าจอการกำหนดค่าต่างๆในการทดสอบของ General Tab



รูปที่ 6.15 หน้าจอการกำหนดค่าต่างๆในการทดสอบของ AutoDump

3. Waiting/Disk space checking

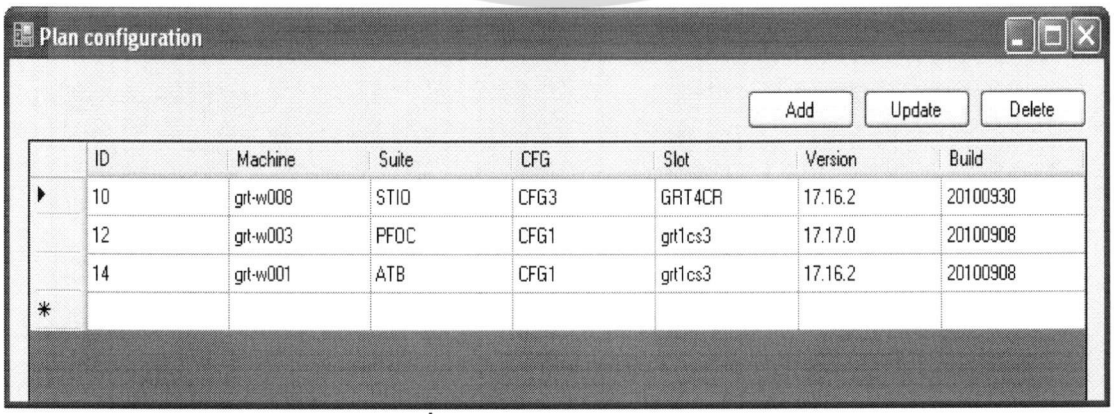
หน้าจอสำหรับกำหนดช่วงเวลาในการรอเวอร์ชันและวันที่ของ Build ที่ต้องการ โดยสามารถกำหนดค่าการได้สูงสุด 24 ชั่วโมง และสามารถกำหนดระยะเวลาของการตรวจสอบได้ เช่น ตรวจสอบทุกๆ 5 นาที เป็นต้น ดังรูป 6.16



รูปที่ 6.16 หน้าจอการกำหนดค่าต่างๆในการทดสอบของ Waiting/Disk space checking

5. Planning

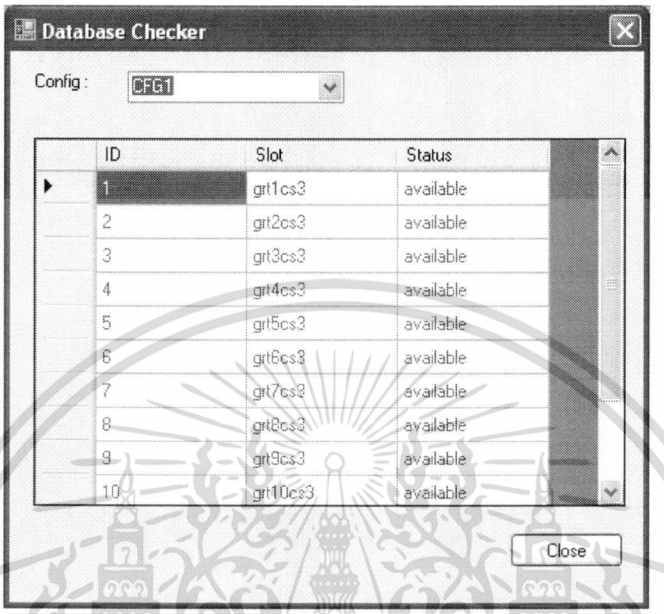
เป็นเมนูที่ใช้เข้าสู่หน้าจอแผนทดสอบ ซึ่งสามารถทำการเพิ่ม แก้ไข และลบข้อมูลแผนทดสอบได้ ดังรูป 6.17



รูปที่ 6.17 หน้าจอแผนทดสอบ

6. Database Slot checking

เป็นเมนูที่ใช้เข้าสู่หน้าจอการตรวจสอบสถานะการใช้งานของ Database Slot ว่าตัวไหนว่างหรือมีการใช้งานอยู่ ดังรูป 6.18



รูปที่ 6.18 หน้าจอการตรวจสอบสถานะการใช้งานของ Database Slot

7. Disk space checking

เป็นเมนูที่ใช้เข้าสู่หน้าจอการตรวจสอบพื้นที่บนเครื่องเซิร์ฟเวอร์ที่ได้ทำการ Map ว่ามีพื้นที่เพียงพอที่จะให้ใช้ทดสอบได้หรือไม่ ดังรูป 6.19

รูปที่ 6.19 หน้าจอการตรวจสอบพื้นที่บนเครื่องเซิร์ฟเวอร์

6.3 การทดสอบระบบ

ตัวอย่างการใช้งานระบบเพื่อทดสอบแอพพลิเคชัน เมื่อทางทีมได้รับการร้องขอให้ทำการทดสอบแอพพลิเคชัน โดยมีรายละเอียดของการร้องขอดังนี้

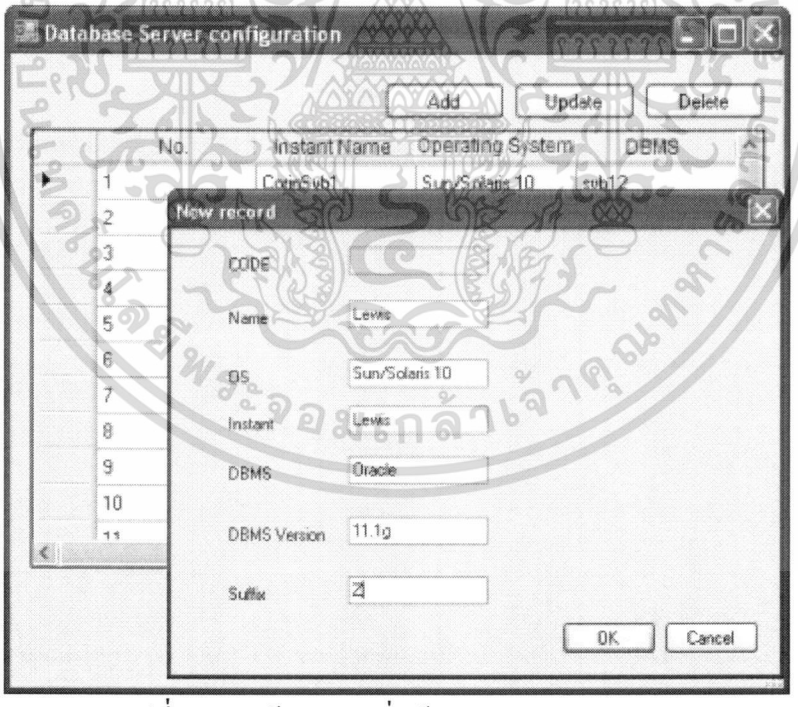
- ทดสอบ Regression Test ด้วย Build เวอร์ชัน 17.17.0 ของวันที่ 20110215
- ให้ทดสอบบนระบบปฏิบัติการยูนิกซ์ (Unix)
- ใช้ Database Server ที่มี DBMS เป็น Orcale 11.1g
- ใช้ Application Server ที่เป็น WebLogic
- ทดสอบ Test Suite ที่ชื่อ Benchmark (BM)

จากนั้นผู้ทดสอบจะต้องทำการเตรียม Environment ต่างๆให้พร้อมและตั้งเวลาทดสอบ ซึ่งมีขั้นตอนการทำงาน ดังนี้

1. ส่วนของการเตรียม Environment

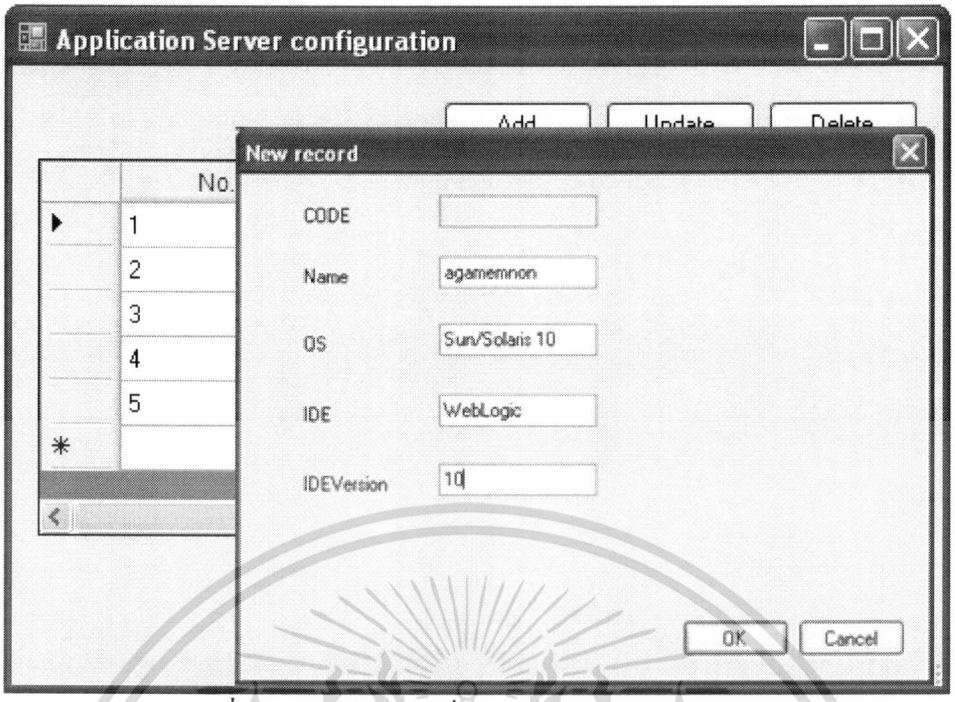
1. Automated Test Developer จะต้องตรวจสอบและเพิ่มข้อมูลในระบบ ดังนี้

1.1 ตรวจสอบและเพิ่มข้อมูล Database Server ที่มี DBMS เป็น Orcale 11.1g โดยไปที่เมนู Database Server แล้วทำการเพิ่มข้อมูล ดังรูป 6.20



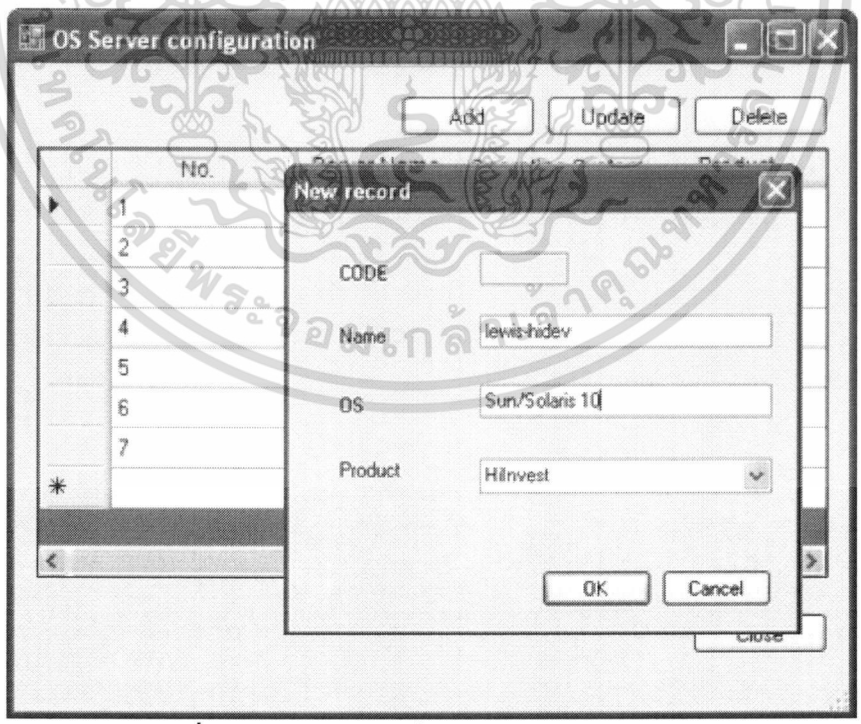
รูปที่ 6.20 หน้าจอการเพิ่มข้อมูล Database Server

1.2 ตรวจสอบและเพิ่มข้อมูล Application Server ที่มี WebLogic โดยไปที่เมนู Application Server แล้วทำการเพิ่มข้อมูล ดังรูป 6.21



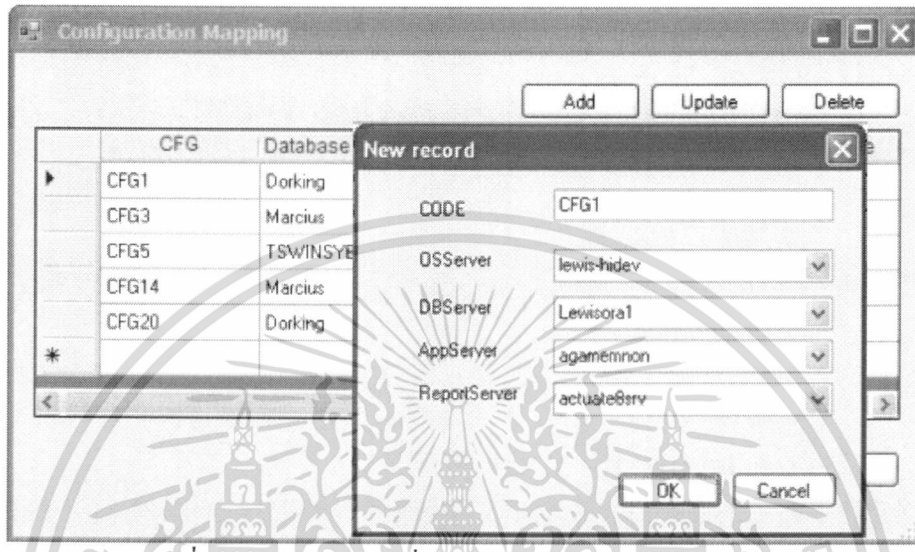
รูปที่ 6.21 หน้าจอการเพิ่มข้อมูล Application Server

1.3 ตรวจสอบและเพิ่มข้อมูล Engine Server ที่มีระบบปฏิบัติการยูนิกซ์ (Unix) โดยไปที่เมนู Engine Server แล้วทำการเพิ่มข้อมูล ดังรูป 6.22



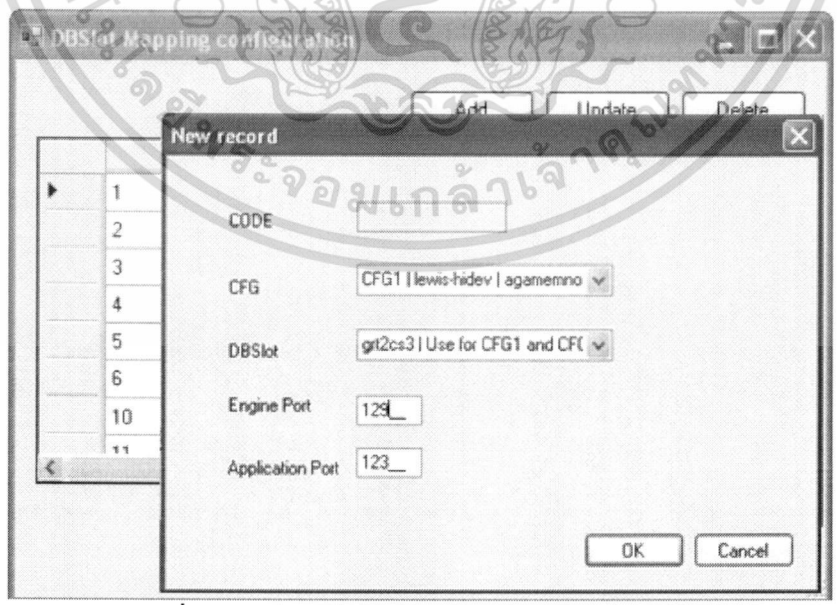
รูปที่ 6.22 หน้าจอการเพิ่มข้อมูล Engine Server

1.4 เมื่อได้ข้อมูลเซิร์ฟเวอร์ทั้ง 3 เซิร์ฟเวอร์แล้ว Automated Test Developer จะนำข้อมูลเหล่านั้นไปทำการ Mapping สร้างเป็น Environment ของการทดสอบขึ้นมา โดยไปที่เมนู Configuration Mapping ดังรูป 6.23 จากรูปเป็นการนำเซิร์ฟเวอร์ทั้ง 3 เซิร์ฟเวอร์มาสร้างเป็น Environment ของการทดสอบที่ชื่อว่า CFG1



รูปที่ 6.23 หน้าจอการเพิ่มข้อมูล Configuration Mapping

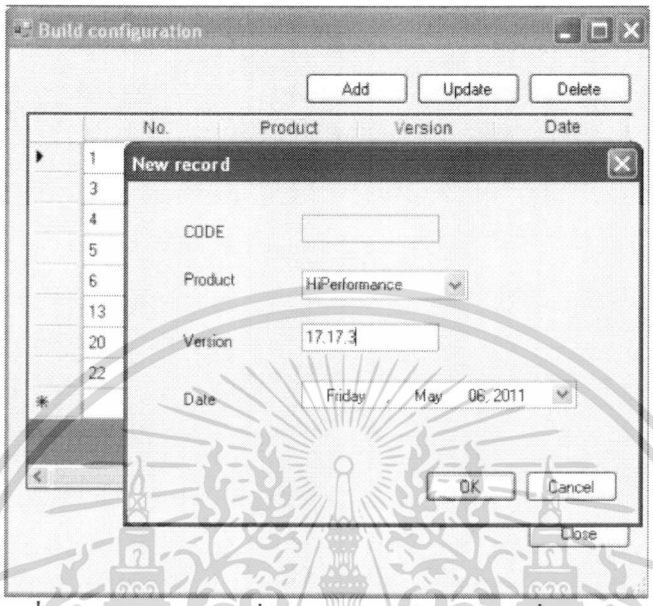
1.5 ทำการ Mapping Environment กับ Database Slot โดยไปที่เมนู DB Slot Mapping ดังรูป 6.24



รูปที่ 6.24 หน้าจอการเพิ่มข้อมูล DB Slot Mapping

2. Automated Test Executor จะต้องตรวจสอบและเพิ่มข้อมูลในระบบ ดังนี้

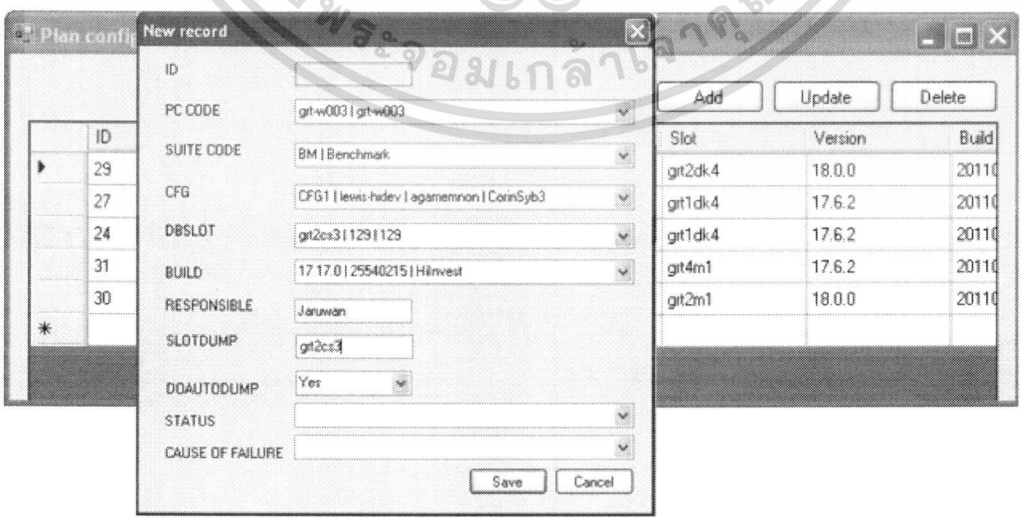
2.1 ตรวจสอบและเพิ่มข้อมูล Build โดยไปที่เมนู Build แล้วทำการเพิ่มข้อมูล ด้วยการระบุเลขเวอร์ชัน และวันที่ของซอฟต์แวร์ ดังรูป 6.25



รูปที่ 6.25 หน้าจอการเพิ่มข้อมูลเวอร์ชันและวันที่ของ Build

2. ส่วนของการวางแผนการทดสอบ

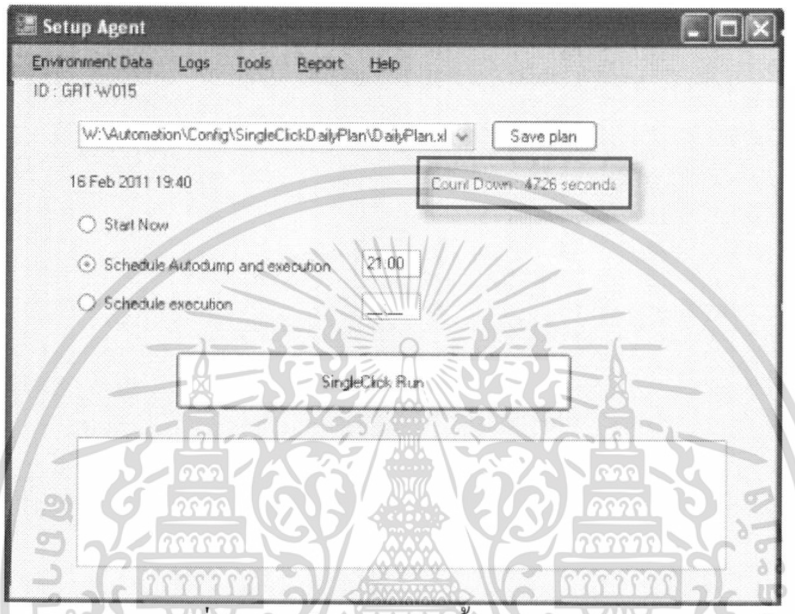
เมื่อตรวจสอบ Environment ต่างๆ ที่เกี่ยวข้องและพร้อมที่จะทดสอบแล้ว Automated Test Executor จะไปทำการวางแผนการทดสอบว่าต้องการทดสอบ Test Suite ตัวไหน โดยไปที่หน้าจอ Planning และวางแผนทดสอบตามที่ได้รับการร้องขอมา แล้วทำการบันทึก



รูปที่ 6.26 หน้าจอการเพิ่มข้อมูลแผนการทดสอบ

3. ส่วนของการตั้งเวลาทดสอบ

เมื่อทำการวางแผนการทดสอบเรียบร้อยแล้ว Automated Test Executor จะไปตั้งเวลาทดสอบ โดยเลือกรูปแบบการตั้งเวลาแบบ Schedule Autodump and Execution ผู้ทดสอบจะต้องทำการระบุเวลาที่จะเริ่มทำการทดสอบ และกดปุ่ม SingleClick Run เพื่อให้ระบบทำงาน ระบบก็จะทำการนับเวลาถอยหลังไปจนถึงเวลาที่ได้ระบุไว้ ดังรูปที่ 6.27

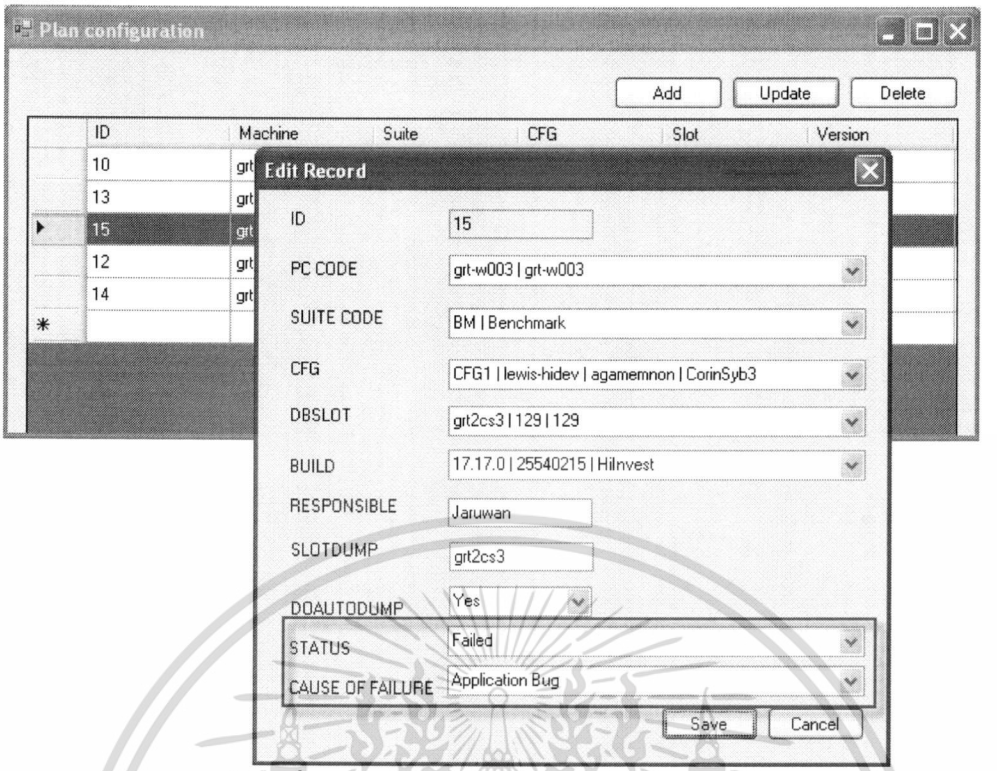


รูปที่ 6.27 หน้าจอการการตั้งเวลาทดสอบ

เมื่อถึงเวลาที่ได้ระบุไว้ ระบบจะเริ่มทำการทดสอบ โดยแยกการทำงานออกเป็น 2 ส่วน คือ ส่วนที่เกี่ยวกับเซิร์ฟเวอร์ จะทำการเปิดโปรแกรม Reflection ขึ้นมาเพื่อทำเชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์และทำการโหลดข้อมูลเข้าสู่ Database Slot และอีกส่วนหนึ่งเป็นส่วนที่เกี่ยวกับเครื่องทดสอบ โดยระบบจะทำการติดตั้งซอฟต์แวร์ที่ใช้ทดสอบตามวันที่และเวอร์ชันของซอฟต์แวร์ที่ได้วางแผนทดสอบไว้

4. ส่วนของการเก็บผลและบันทึกผลการทดสอบ

เมื่อการทดสอบสิ้นสุดลง Automated Test Executor จะได้รับอีเมลแจ้ง จากนั้น Automated Test Executor จะเข้ามาเก็บผลและบันทึกผลลงในระบบ โดยเข้าไปที่หน้าจอ Planning แล้วเลือกข้อมูลที่ต้องการบันทึกผล กด update เพื่ออัพเดตผลการทดสอบ ดังรูปที่ 6.28



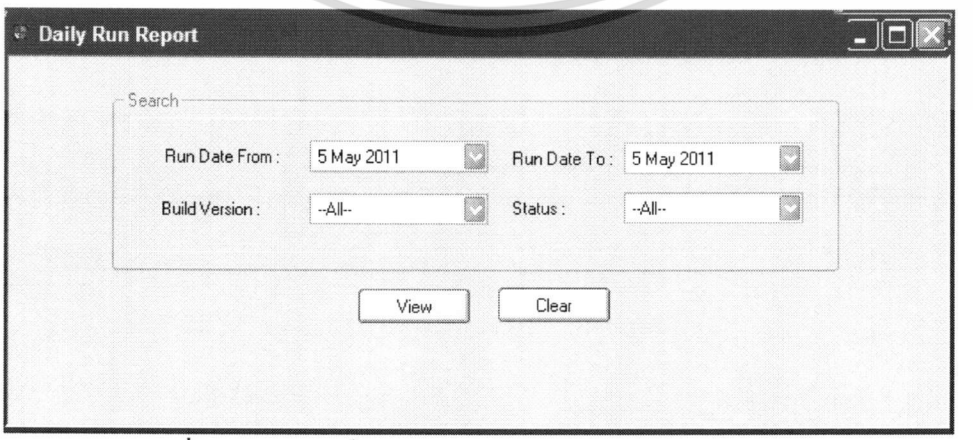
รูปที่ 6.28 แสดงการบันทึกผลการทดสอบ

5. ส่วนของการออกรายงานผลการทดสอบ

รายงานผลการทดสอบจะประกอบไปด้วยรายงาน 2 รูปแบบ คือ

5.1 Daily Run Report

เป็นรายงานผลการทดสอบที่ผู้ทดสอบได้ทำการทดสอบในแต่ละวัน รวมถึงแสดงสถานะของสคริปท์ทดสอบที่ได้ทำการทดสอบ โดยเข้าไปที่เมนู Report -> Daily Run Report ระบบจะแสดงหน้าจอ Daily Run Report ซึ่งมีเงื่อนไขของการค้นหารายงาน ดังรูปที่ 6.29 เมื่อกด View ระบบจะแสดงหน้าจอรายงานขึ้นมาแสดง ดังรูปที่ 6.30



รูปที่ 6.29 หน้าจอเงื่อนไขการออกรายงาน Daily Run Report

Daily Run Report

Report Date : 07/05/2011
Report By : Jaruwan A.

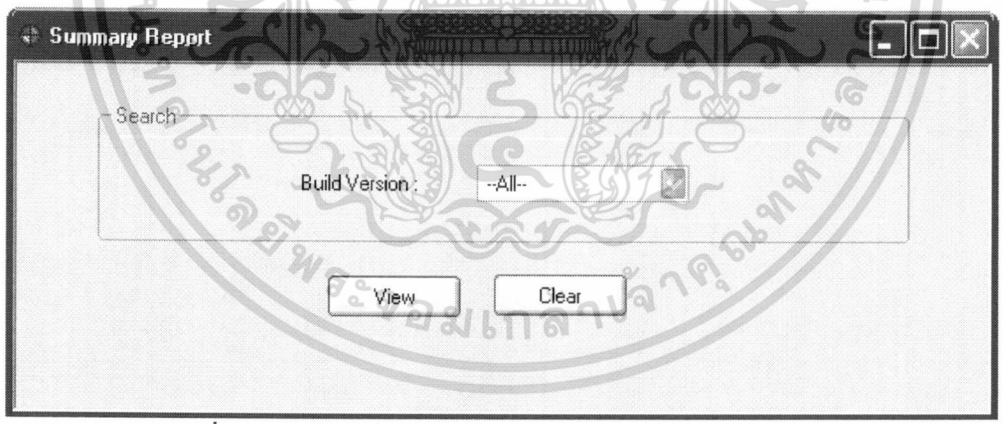
Build Version	Build Date	Configuration	Suite Name	Suite Code	Result	Cause Of Failure
17.17.0	20110504	CFG11	Portfolio Characteristic	PFOC	Failed	Configuration issue
			Interface data warehouse 2	IDW2	Dump	
			Interface data warehouse	IDW	Running	
			Attribution	ATB	Passed	
17.16.1	20110505	CFG1	Portfolio Characteristic	PFOC	Failed	Spreadsheet mismatched
18.0.0	20110507	CFG1	Portfolio Characteristic	PFOC	Failed	Application Bug
18.0.0	20110507	CFG11	Model Build	MDL	Failed	Application Bug
18.0.0	20110507	CFG12	Portfolio Characteristic	PFOC	Passed	
17.17.0	20110507	CFG12	Model Build	MDL	Failed	Configuration issue
18.0.0	20110507	CFG3	Stability Issuer On	STIO	Passed	

Page 1 of 1

รูปที่ 6.30 รายงาน Daily Run Report

5.2 Summary Report

รายงานสรุปผลการทดสอบในภาพรวมของการทดสอบ เพื่อส่งให้กับผู้บริหาร โดยเข้าไปที่เมนู Report -> Summary Report ระบบจะแสดงหน้าจอ Summary Report ซึ่งมีเงื่อนไขของการค้นหารายงาน ดังรูปที่ 6.31 เมื่อกด View ระบบจะแสดงหน้าจอรายงานขึ้นมาแสดง ดังรูปที่ 6.32



รูปที่ 6.31 หน้าจอเงื่อนไขการออกรายงาน Summary Report

Summary Report

Build Verion : --All--

Print Date : 07/05/2011

Version	CFG	Suite Name	Status	Cause Of Failure
17.17.0	CFG11	Attribution	Passed	
		Benchmark	Passed	
		Interface data warehouse	Running	
		Interface data warehouse 2	Dump	
		Model Build	Failed	Application Bug
		Portfolio Characteristic	Failed	Configuration issue
		Returns	Failed	Application Bug
		Stability Issuer On	Passed	
18.0.0	CFG12	Attribution	Passed	
		Benchmark	Passed	
		Interface data warehouse	Running	
		Interface data warehouse 2	Dump	
		Model Build	Failed	Application Bug
		Portfolio Characteristic	Failed	Configuration issue
		Returns	Failed	Application Bug
		Stability Issuer On	Passed	

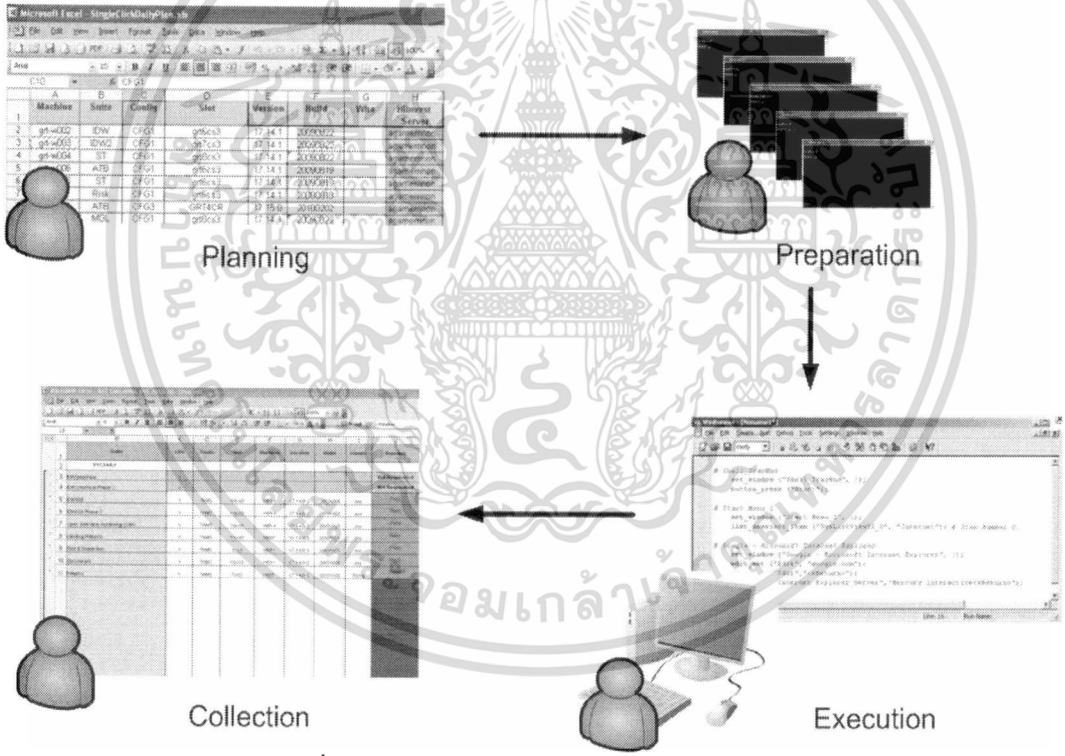
Page 1 of 1

รูปที่ 6.32 รายงาน Summary Report

บทที่ 7
บทสรุปและข้อเสนอแนะ

7.1 บทสรุป

แอพพลิเคชันสำหรับเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบ ได้พัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับผู้ทดสอบ แอพพลิเคชันดังกล่าวจะครอบคลุมการทำงานในเรื่องของการวางแผนการทดสอบ การจัดเตรียม Test Environment ต่างๆ และการตั้งเวลาการทดสอบ โดยจะปรับเปลี่ยนรูปแบบการทำงานของผู้ทดสอบจากเดิมที่ต้องจัดเตรียม Test Environment แบบ Manual ให้เป็นแบบอัตโนมัติมากขึ้น หรือการจัดเก็บข้อมูลการทดสอบที่จัดเก็บแบบ File System ให้เก็บอยู่ในรูปของฐานข้อมูล รูปแบบการทำงานของเดิม แสดงได้ดังรูปที่ 7.1



รูปที่ 7.1 แผนภาพแสดงการทำงานในปัจจุบัน

จากรูปสามารถอธิบายการทำงานได้ดังนี้

1. เริ่มจากผู้ทดสอบจะต้องทำการวางแผนการทดสอบในแต่ละวันที่ต้องการทดสอบในไฟล์สเปรทชีท (Spreadsheet)
2. เมื่อวางแผนการทดสอบเรียบร้อยแล้วผู้ทดสอบจะไปทำการเตรียม Test Environment ต่างๆ แบบ Manual ได้แก่

2.1 ตรวจสอบเวอร์ชันและวันที่ของซอฟต์แวร์ที่จะทดสอบ

2.2 เตรียม Database Slot

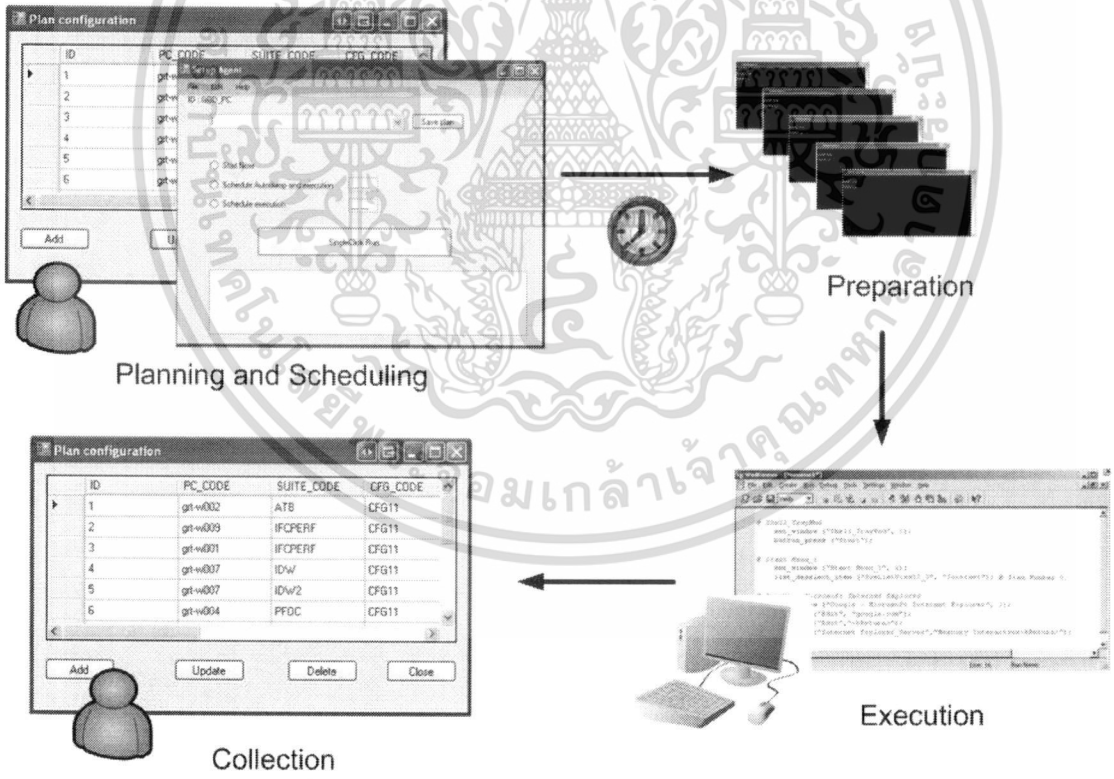
2.3 โหลดข้อมูลเข้าสู่ Database Slot และอัพเดตข้อมูลให้ตรงกับเวอร์ชันและวันที่ของซอฟต์แวร์ที่จะทดสอบ

2.4 ตรวจสอบเครื่องที่จะทดสอบ

2.5 ติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ

3. เมื่อเตรียม Test Environment เรียบร้อยแล้ว ผู้ทดสอบจะต้องเข้าไปที่เครื่องทดสอบแต่ละเครื่อง เพื่อสั่งการให้ Automated Test Scripts ทำงาน
4. เมื่อการทดสอบเสร็จสิ้น ผู้ทดสอบจะบันทึกผลการทดสอบในไฟล์สเปรทชีท (Spreadsheet)

สำหรับแอพพลิเคชันสำหรับเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบที่พัฒนาขึ้นนั้นจะปรับเปลี่ยนรูปแบบการทำงานดังรูปที่ 6.2



รูปที่ 7.2 แผนภาพแสดงการทำงานของแอพพลิเคชัน

จากรูปสามารถอธิบายการทำงานได้ดังนี้

1. เริ่มจากผู้ทดสอบจะต้องทำการวางแผนการทดสอบในแต่ละวันที่ต้องการทดสอบโดยใช้แอพพลิเคชัน ข้อมูลการวางแผนจะถูกจัดเก็บลงฐานข้อมูล

2. เมื่อวางแผนการทดสอบเรียบร้อยแล้วผู้ทดสอบจะตั้งเวลาการทดสอบ
3. เมื่อถึงเวลาที่ได้กำหนดไว้ ระบบจะเริ่มทำงานโดยจะทำการเตรียม Test Environment ต่างๆ อย่างอัตโนมัติ ดังนี้
    3.1 ตรวจสอบเวอร์ชันและวันที่ของซอฟต์แวร์ที่จะทดสอบ โดยจะรอจนกว่ามีเวอร์ชันและวันที่ของซอฟต์แวร์ที่ต้องการ
    3.2 โหลดข้อมูลเข้าสู่ Database Slot และอัพเดตข้อมูลให้ตรงกับเวอร์ชันและวันที่ของซอฟต์แวร์ที่จะทดสอบ
    3.3 ติดตั้งซอฟต์แวร์ที่เครื่องทดสอบ
4. เมื่อระบบทำการเตรียม Test Environment ต่างๆ เรียบร้อยแล้ว ระบบจะไป Trigger สั่งการให้ Automated Test Scripts ทำงาน
5. เมื่อการทดสอบเสร็จสิ้น ผู้ทดสอบจะบันทึกผลการทดสอบลงในแอพพลิเคชันเพื่อจัดเก็บข้อมูลลงฐานข้อมูล

โครงการพัฒนาระบบสารสนเทศนี้ได้เริ่มศึกษาตั้งแต่ระบบงานปัจจุบัน ความต้องการของผู้ใช้ ศึกษาความเป็นไปได้ด้านต่างๆ ตลอดจนค้นคว้าหาข้อมูลนำมาวิเคราะห์และออกแบบตามหลักการวิเคราะห์เชิงวัตถุ ออกแบบส่วนต่อประสานผู้ใช้งาน ทฤษฎีที่เกี่ยวข้อง ศึกษาและพัฒนาตามขั้นตอนและหลักการพัฒนาระบบ การออกแบบหน้าจอในรูปแบบวินโดว์แอพพลิเคชันซึ่งได้ใช้ Visual Studio .Net 2005 ประโยชน์ที่ได้รับจากการพัฒนาแอพพลิเคชันสำหรับเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบ มีดังนี้

1. ช่วยลดการทำงานของผู้ทดสอบที่ต้องทำการจัดเตรียม Test Environment แบบ Manual ให้มีความเป็น Automated มากขึ้น เช่นจากเดิมผู้ทดสอบจะต้องเข้าไปที่เครื่องทดสอบเพื่อสั่งการให้ Automated Test Scripts ทำงาน

2. เมื่อผู้ทดสอบใช้เวลาในการจัดเตรียม Test Environment น้อยลง ทำให้ผู้ทดสอบมีเวลาที่จะวิเคราะห์ถึงสาเหตุของข้อบกพร่องต่างๆ ที่พบได้มากขึ้น หรือมีเวลาศึกษา เรียนรู้เงื่อนไขทางธุรกิจขององค์การมากขึ้น

3. ช่วยลดข้อผิดพลาดที่อาจเกิดจากการเก็บข้อมูลในสเปรทชีท (Spreadsheet) โดยให้มีการจัดเก็บในรูปของฐานข้อมูล

4. ระบบสามารถตั้งเวลาการทดสอบได้ จึงช่วยเพิ่มประสิทธิภาพและประสิทธิผลของจำนวนการทดสอบให้ได้ปริมาณการทดสอบที่มากขึ้น

7.2 ข้อเสนอแนะ

แม้ว่าแอพพลิเคชันสำหรับเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบที่ได้พัฒนาขึ้นมาจะสามารถช่วยให้การทำงานของผู้ทดสอบซอฟต์แวร์สามารถทำงานได้สะดวกมากขึ้น ช่วยลดการทำงานแบบ Manual ให้น้อยลง ลดความผิดพลาดที่เกิดจากผู้ทดสอบ ลดเวลาในการเตรียมสภาพแวดล้อมให้น้อยลงได้ก็ตาม แต่ในการทดสอบจริงนั้นจะต้องมีการทดสอบบนสภาพแวดล้อมที่หลากหลายระบบปฏิบัติการ เช่น ระบบปฏิบัติการยูนิกซ์ (Unix) ระบบปฏิบัติการวินโดว์ (Windows) เป็นต้น ซึ่งแอพพลิเคชันที่พัฒนาขึ้นมาใช้งานในเบื้องต้นนั้นจะรองรับการทดสอบบนระบบปฏิบัติการยูนิกซ์เท่านั้น เพราะชุดคำสั่งแต่ละระบบปฏิบัติการนั้นไม่เหมือนกัน ทำให้การทดสอบบนระบบปฏิบัติการวินโดว์ ยังคงต้องใช้การเตรียมสภาพแวดล้อมด้วยวิธีแบบ Manual อยู่

อย่างไรก็ตามในอนาคตอาจบูรณการแอพพลิเคชันสำหรับเตรียมสภาพแวดล้อมและตั้งเวลาการทดสอบให้มีประสิทธิภาพมากขึ้น ด้วยการพัฒนาให้รองรับการทดสอบบนสภาพแวดล้อมที่หลากหลายมากขึ้น ให้ครอบคลุมกับการทดสอบจริง ทั้งนี้เพื่อให้รูปแบบการทดสอบเป็นไปในทางเดียวกัน อีกทั้งยังช่วยเพิ่มศักยภาพในการทดสอบแบบอัตโนมัติอีกด้วย

บรรณานุกรม

อริสไทด์ คารัคโช่. 2550. **Reflection ของ WRQ พัฒนาการเข้าถึงเครื่องคอมพิวเตอร์แม่ข่าย (Host).** [Online] เข้าถึงได้จาก: http://www.thaibusinessnews.com/readnews.aspx.

Patton, R. 2000. **Software Testing.** Indianapolis: Sams Publishing.

Poston, M. R. 1994a. "Moving from Manual to Automated Test Execution". **Communications of the ACM.** 37(9): 124-129.

Poston, M. R. 1994b. "Automating Testing from Object Models". **Communications of the ACM.** 37(9): 48-58.

Sodhani, P. 2009. "Pre-Deployment Load Testing". **Software Testing & Performance Magazine.** 6(5): 29-32

Pressman, R.S. 2005. **วิศวกรรมซอฟต์แวร์.** แปลและเรียบเรียงโดย พรฤดี เนติโสภากุล. 2549. กรุงเทพฯ: ท้อป.

ประวัติผู้เขียน

ชื่อ-นามสกุล	นางสาวจารุวรรณ อริยะพัฒนพาณิชย์
วัน เดือน ปีเกิด	25 พฤษภาคม 2525
สถานที่เกิด	กรุงเทพมหานคร
ที่อยู่	462 ซ.มังกร ถ.มัง อ.ป้อมปราบ กทม. 10100
ประวัติการศึกษา	2547 วิทยาศาสตรบัณฑิต สาขาสถิติประยุกต์
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ประสบการณ์การทำงาน	
พ.ศ.2553-ปัจจุบัน	ตำแหน่งเจ้าหน้าที่อาวุโสฝ่ายตรวจสอบคุณภาพ
	บริษัท โพรเกรสซอฟต์แวร์ จำกัด
พ.ศ.2549-2553	ตำแหน่งวิศวกรควบคุมคุณภาพ
	บริษัท ดีเอสที เวิลด์วายด์เซอร์วิส จำกัด