ระบบงานควบคุมวัสดุอะไหล่เครือข่ายโทรศัพท์มือถือผ่านเว็บ

Web-based Mobile Phone Network Spare Part Control System

โดย

นายสุรัส ธาราสมบัติ

รหัส 41067112

อาจารย์ที่ปรึกษา

รศ. ดร. วิเชียร เปรมชัยสวัสดิ์

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 1 ปีการศึกษา 2544
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวเรื่อง	ระบบควบคุมวัสดุอะไหล่เครือข่ายโทรศัพท์มือถือผ่านเว็บ
นักศึกษา	นาย สุรัส ธาราสมบัติ
อาจารย์ที่ปรึกษา	รศ.ดร.วิเชียร เปรมชัยสวัสดิ์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2544

บทคัดย่อ

ระบบควบคุมวัสดุอะไหล่เครือข่ายโทรศัพท์มือถือผ่านเว็บ เป็นระบบซึ่งจะใช้ในการควบคุมการเบิกจ่ายวัสดุอะไหล่ เพื่อนำไปใช้ในการซ่อมบำรุงอุปกรณ์ซึ่งถูกติดตั้งอยู่ ณ สถานีฐานต่างๆ ทั่วประเทศ โดยได้มีการนำวิธีการวิเคราะห์และออกแบบระบบแบบเชิงวัตถุ(Object Oriented Analysis and Design Method)มาประยุกต์ใช้ ซึ่งแบบจำลองของระบบงานทั้งหมดจะถูกจัดสร้างอยู่ในรูปแบบของ UML (Unified Modeling Language) ซึ่งสนับสนุนแนวคิดเชิงวัตถุ

นอกจากนี้ระบบควบคุมวัสดุอะไหล่เครือข่ายโทรศัพท์มือถือยังถูกสร้างบนสถาปัตยกรรมที่เป็นการทำงานผ่านเว็บ(Web-based Architecture) ซึ่งจะทำให้สามารถใช้งานผ่านทางอินทราเน็ตของบริษัทซึ่งมีการเชื่อมโยงกันทั่วประเทศได้ และในส่วนของโปรแกรมในฝั่งของเซิร์ฟเวอร์นั้นถูกพัฒนาขึ้นด้วยเทคโนโลยีของภาษาจาวา ซึ่งเป็นการมาผสมผสานกันระหว่าง จาวาเซิร์ฟเล็ต (Java Servlet) จาวาเซิร์ฟเวอร์เพจ (Java Server Page – JSP) และ จาวาบีน (Java Bean)

Title	Web-based Mobile Phone Network Spare Part Control System
Student	Mr. Surus Tarasombat
Advisor	Assoc. Prof. Dr.Wichian Premchaiswadi
Level of study	Master of Science in Information Technology
Major	Information Science
Year	2001

Abstract

Web-based Mobile Phone Network Spare Part Control System is a system that will be used to control spare part requisition for maintain equipment that installed at base station in each region of country . By using Object Oriented Analysis and Design Method , all of system model are constructed in UML (Unified Modeling Language) format that support object oriented concepts.

In addition this system is implemented on Web-based Architecture that be available for using via intranet of company . And in the part of server-side program, it's developed by Java Technology that be integration of Java Servlet , Java Server Page(JSP) and Java Bean.

กิตติกรรมประกาศ

ในการจัดทำโครงการระบบควบคุมวัสดุอะไหล่เครือข่ายโทรศัพท์มือถือผ่านเว็บนี้ ได้รับความสนับสนุนเป็นอย่างดี จากหลายฝ่าย ที่คอยให้คำแนะนำปรึกษา และเสียสละเวลาอันมีค่า จนทำให้การศึกษาโครงการนี้บรรลุผลตามเป้าหมายที่วางไว้ ผู้จัดทำจึงใคร่ขอขอบพระคุณ

1. รศ. ดร. วิเชียร เปรมชัยสวัสดิ์ อาจารย์ที่ปรึกษาโครงการที่ได้ให้ความรู้ คำปรึกษา ในการจัดทำโครงการ
2. พี่ๆ เพื่อนๆ และ น้องๆ ในฝ่าย IT System บริษัทแอดวานซ์ อินโฟร์ เซอร์วิส มหาชน จำกัด ที่คอยให้ความช่วยเหลือ และเป็นกำลังใจในการทำงาน
3. เครื่องคอมพิวเตอร์ ซอฟต์แวร์ ของ ฝ่าย IT System บริษัทแอดวานซ์ อินโฟร์ เซอร์วิส มหาชน จำกัด ที่ใช้เป็นเครื่องมือในการทำงาน

สุรัส ธาราสมบัติ

ผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

บริษัทแอดวานซ์ อินโฟร์ เซอร์วิส มหาชน จำกัด เป็นบริษัทผู้ให้บริการเครือข่ายโทรศัพท์มือถือ ในระบบ NMT และ GSM ในคลื่นความถี่ 900 MHz ในส่วนของสายงาน วิศวกรรม มีการแบ่งหน้าที่ความรับผิดชอบ กระจายไปตามภูมิภาค โดยสามารถแบ่งออก ได้เป็น 6 ภูมิภาค ได้แก่ ภาคกลาง ภาคเหนือ ภาคตะวันออกเฉียงเหนือ ภาคตะวันออก ภาคใต้ และ ภาคตะวันตก โดย ในแต่ละภูมิภาค จะมีการแบ่ง หน้าที่รับผิดชอบในงานบำรุงรักษาสถานีฐาน (Field Maintenance) ออกเป็น ศูนย์ซ่อมบำรุงย่อย (Maintenance Center - MC) อีก เช่น ในภาคเหนือ จะแบ่งออกได้เป็น 3 ศูนย์ซ่อมบำรุงย่อย ได้แก่ MC เชียงใหม่ MC พิษณุโลก และ MC นครสวรรค์ เป็นต้น

หน้าที่ความรับผิดชอบหลัก ของ งานบำรุงรักษาสถานีฐาน จะแบ่งออกได้เป็น 2 ส่วน คือ งานบำรุงรักษาเพื่อป้องกันการเกิดความเสียหาย (Preventive Maintenance) และ งานบำรุงรักษาเพื่อแก้ไขความเสียหายที่เกิดขึ้น (Corrective Maintenance) ซึ่ง งานบำรุงรักษาทั้งสองลักษณะนี้ โดยทั่วไป มักจะเป็นลักษณะการเข้าไปที่สถานีฐานเพื่อทำการซ่อมบำรุง อุปกรณ์ที่เกิดปัญหาขึ้น หรือ เริ่มเสื่อมสภาพ ซึ่งหากอุปกรณ์นั้นๆ ไม่สามารถที่จะซ่อมให้ดีเหมือนเก่าได้ทันที ณ ขณะนั้น ก็จำเป็นที่จะต้องเปลี่ยนเอาอุปกรณ์ดังกล่าวออกจากระบบ แล้วแทนที่ด้วยวัสดุอะไหล่ (Spare part) ที่วิศวกร นำติดตัวไป แล้วจึงนำอุปกรณ์ดังกล่าวกลับมาทำการส่งซ่อมต่อไป

จากลักษณะของการทำงานดังกล่าวจะพบว่า มีการไหล(Flow) ของวัสดุอะไหล่ ภายในระบบค่อนข้างมาก ซึ่งหากไม่มีระบบการควบคุม และจัดการที่ดีแล้ว ก็จะทำให้วัสดุอะไหล่ เหล่านี้ไม่ได้ถูกนำไปใช้งานอย่างเหมาะสม มีประสิทธิภาพ และ คุ้มค่ากับเงินลงทุนพอ จากความจำเป็นเหล่าที่กล่าวมา จึงเป็นที่มาของระบบควบคุมวัสดุอะไหล่เครือข่ายโทรศัพท์มือถือผ่านเว็บ (Web-based Spare Part Control System for Mobile Phone Network) หรือเรียกย่อๆ ว่า ระบบควบคุมวัสดุอะไหล่ (Spare Part Control System – SPCS) นี้

สำหรับแนวคิดที่จะใช้ UML ในการพัฒนาระบบงาน ก็เพราะเหตุผลที่ว่า ในโลกของความเป็นจริง สิ่งต่างๆ ที่เรามองเห็นล้วนแล้วก็คือวัตถุ ทำให้คนเราคุ้นเคยกับการมองสิ่งต่างๆ ที่เห็นเป็น

วัตถุ ฉะนั้นเมื่อมีการนำแนวคิดนี้มาประยุกต์ใช้กับการพัฒนาระบบงานสารสนเทศ ทำให้ประมาณปี ค.ศ. 1960 จึงเกิดมีผู้นำเสนอแนวคิดเชิงวัตถุขึ้น เพื่อใช้ในการวิศวกรรมซอฟต์แวร์ (system engineering) เนื่องด้วยเทคโนโลยีเชิงวัตถุ (Object Technology) มีจุดเด่นที่สำคัญคือ การนำกลับมาใช้ใหม่ (reusability) ของส่วนของโปรแกรม (program components) ทำให้การพัฒนาระบบงานสำเร็จได้อย่างรวดเร็ว และมีคุณภาพที่สูงขึ้น โดยเฉพาะการพัฒนาซอฟต์แวร์ในเชิงวัตถุ มีความง่ายในการบำรุงรักษา (maintenance) เนื่องจากมีคุณสมบัติการสืบทอด (inheritance) (Pressman, S. 1997)

หลายปีมาแล้วที่คำว่า "Object-oriented" (OO) ถูกนำมาใช้ในการพัฒนาระบบงาน ได้แก่ ภาษาในเชิงวัตถุ (Object-oriented programming languages) ตัวอย่างเช่น ภาษา C++, Smalltalk และ Java นอกจากนั้นแล้ว การวิศวกรรมซอฟต์แวร์ (system development process) ก็ได้นำเอาแนวคิดนี้มาใช้ด้วยเช่นกัน โดยจะเห็นว่าเกิดคำศัพท์ต่างๆ อาทิ Object-oriented programming (OOP), Object-oriented design (OOD), Object-oriented domain analysis (OOA), Object-oriented database management systems (OODBMS) และ Object-oriented computer aided software engineering (OOCASE) ซึ่งล้วนแล้วแต่เป็นการนำแนวคิดเชิงวัตถุมาใช้ในการพัฒนาระบบทั้งสิ้น (Pressman, S. 1997)

ในปัจจุบันการพัฒนาระบบงานสารสนเทศมีแนวโน้มจะใช้แนวคิดในเชิงวัตถุมากขึ้น โดยสังเกตได้จากเทคโนโลยีในเชิงวัตถุที่ได้รับความนิยมสูงขึ้นเรื่อยๆ ซึ่งมีสาเหตุมาจากข้อดีของแนวคิดเชิงวัตถุที่ได้กล่าวแล้วข้างต้น ดังนั้นเมื่อเล็งเห็นคุณประโยชน์และความก้าวไกลของแนวคิดนี้ในอนาคต จึงได้ริเริ่มความคิดที่จะพัฒนาระบบ SPCS นี้โดยการใช้ UML (Unified Modeling Language) เป็นเครื่องมือสำหรับการจำลองระบบงาน โดยสนับสนุนการวิเคราะห์ และออกแบบระบบเชิงวัตถุ ซึ่งจะกล่าวรายละเอียดของ UML ในบทที่ 4

1.2 วัตถุประสงค์ของโครงการพัฒนาระบบงาน

1. เพื่อทดลองนำหลักการวิเคราะห์และออกแบบระบบงานด้วย UML มาใช้กับการพัฒนางานจริง
2. เพื่อทดลองผสมผสานการพัฒนาระบบงาน ด้วยแนวคิดเชิงวัตถุ (Object-oriented application) กับฐานข้อมูลเชิงสัมพันธ์ (Relational Database)
3. เพื่อศึกษาวิธีการเทียบเคียง (mapping) ระหว่างการออกแบบฐานข้อมูลโดยใช้แนวคิดเชิง วัตถุด้วย UML กับฐานข้อมูลเชิงสัมพันธ์
4. เพื่อสามารถออกแบบฐานข้อมูล และพัฒนาฐานข้อมูลเชิงสัมพันธ์ โดยใช้ CASE Tool

สามารถพัฒนาโปรแกรมสำหรับระบบงาน โดยใช้ภาษาเชิงวัตถุ ด้วยแนวคิดการเขียนโปรแกรมในเชิงวัตถุ

1.3 ขอบเขตของโครงการพัฒนาระบบงาน

1. วิเคราะห์และออกแบบระบบงาน โดยใช้แนวคิดเชิงวัตถุ (สร้าง UML Diagram ต่างๆ โดยการใช้โปรแกรม Rational Rose 2000 Enterprise Edition)
2. พัฒนาระบบงานด้วยการ mapping จากแบบจำลอง (UML Diagram) ไปสู่โปรแกรมในลักษณะ Web-based application ด้วยภาษา Java โดยเลือกใช้ Java API (Java Application Programming Interface) ที่เหมาะสม
3. ออกแบบ และสร้างฐานข้อมูลเชิงสัมพันธ์ โดยการสร้าง database schema ด้วยการใช้ CASE Tool (Rational Rose 2000 Edition Oracle8 Add-in)

1.4 เนื้อหาในโครงการพัฒนาระบบงาน

เนื้อหาสำหรับโครงการพัฒนาระบบงานในบทแรกนี้จะกล่าวถึงความเป็นมา ความสำคัญ และวัตถุประสงค์ในการทำโครงการพัฒนาระบบงาน บทที่ 2 กล่าวถึงแนวคิดเชิงวัตถุ ส่วนประกอบที่สำคัญของ Object-oriented methodology บทที่ 3 อธิบายถึงภาษาหรือแบบจำลองที่ใช้ได้แก่ UML รวมทั้ง การใช้งาน UML ในขั้นต่างๆ ของการวิเคราะห์และออกแบบระบบด้วย Iterative model บทที่ 4 แนะนำและกล่าวถึงคุณสมบัติ พร้อมตัวอย่างวิธีการใช้งาน Rational Rose 200 Enterprise Edition ในการสร้าง UML diagram และอธิบายคุณสมบัติ และตัวอย่างการทำ forward engineering และการทำ reverse engineering ของโปรแกรมภาษา Java และ UML ด้วย Rational Rose J รวมทั้งการสร้าง database schema ด้วย Rational Rose Oracle8 วิธีการ mapping จาก Class Diagram ไปเป็น relational database schema บทที่ 5 จะกล่าวถึงรายละเอียดในส่วนของการวิเคราะห์ระบบ บทที่ 6 จะกล่าวถึงรายละเอียดในส่วนของการออกแบบระบบ บทที่ 7 จะกล่าวถึงรายละเอียดในส่วนของการสร้างระบบ โดยจะมีการยกตัวอย่างของ source code ในส่วนของ Java Bean Java Servlet และ Java Server Page บทที่ 8 จะกล่าวถึงแนวทางในการบำรุงรักษาระบบหากว่ามีความต้องการในเพิ่มเติมหรือเปลี่ยนแปลงระบบที่ได้ออกแบบไว้ บทที่ 9 บทสรุปของโครงการพัฒนาระบบงาน และภาคผนวก ได้แก่ ตัวอย่างหน้าจอการทำงานของระบบควบคุมวัสดุอะไหล่เครือข่ายโทรศัพท์มือถือผ่านเว็บ

บทที่ 2

Object-oriented methodology

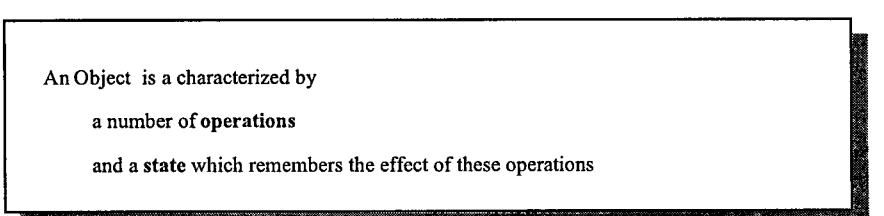
2.1 Object-oriented คืออะไร

"Object-oriented" เป็นแนวคิด หรือ ระเบียบวิธีคิด (paradigm) ของการสร้างหรือพัฒนาระบบงานหนึ่งๆ ดังคำพูดที่ว่า "OO is a technique for system modeling" โดยจะมองระบบเป็นกลุ่มของวัตถุ (Object) ที่มีปฏิกิริยาต่อกัน หรือ "a number of Object that interact" โดยการรวมเอาข้อมูล (Information) และฟังก์ชันการทำงาน (Operation) เข้าไว้ด้วยกันในวัตถุ และกำหนด "วิธีการติดต่อกันระหว่างวัตถุ" (Interface) โดยถ้าวัตถุหนึ่งต้องการจะติดต่อกับอีกวัตถุหนึ่ง จะต้องติดต่อผ่าน Interface ที่กำหนดไว้เท่านั้น (Skylstad, G. and Wuwongse, V. 1998)

จากหลักการนี้จะพบว่าการเปลี่ยนแปลงแก้ไข หน้าที่การทำงาน และข้อมูลใดๆ ในวัตถุหนึ่ง จะมีผลกระทบต่อวัตถุอื่นน้อยมาก ซึ่งจะทำให้การแก้ไขโปรแกรมทำได้สะดวกและรวดเร็วยิ่งขึ้น เป็นการลดต้นทุนในการบำรุงรักษาโปรแกรม นอกจากนี้ยังมีความสามารถในการสืบทอดคุณสมบัติ (Inheritance) จากวัตถุหนึ่งไปอีกวัตถุหนึ่งได้ ทำให้สามารถนำซอฟต์แวร์บางส่วนที่มีอยู่เดิมกลับมาใช้ใหม่ได้ (Reusability) โดยผู้พัฒนาไม่จำเป็นต้อง ทราบว่าซอฟต์แวร์บางส่วนที่นำกลับมาใช้นั้นเขียนขึ้นมาอย่างไร (How does it work?) เพียงแต่ทราบว่าซอฟต์แวร์ส่วนนั้นทำอะไร (What does it work?) และทำการเพิ่มคุณสมบัติอื่นๆ ที่ผู้พัฒนา ต้องการเข้าไป ทำให้การพัฒนาซอฟต์แวร์ใหม่ๆ ทำได้รวดเร็วขึ้น (CDG System. 2000)

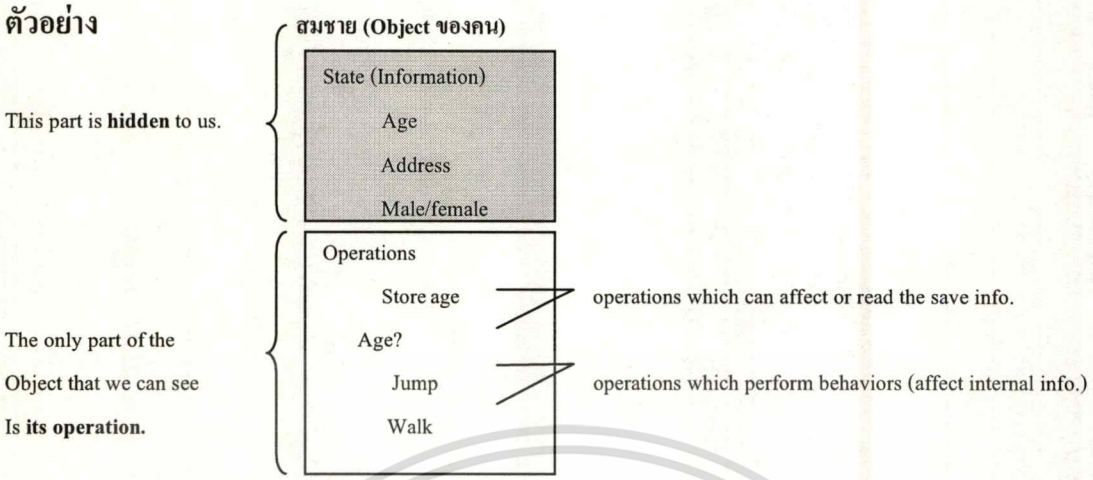
2.2 คำศัพท์ และความหมาย

- **Object** คือ วัตถุ หรือสิ่งที่ประกอบด้วย สถานะ (State) หรือข้อมูล (Information) และฟังก์ชันการทำงาน (Operation) หรือพฤติกรรม (Behavior)



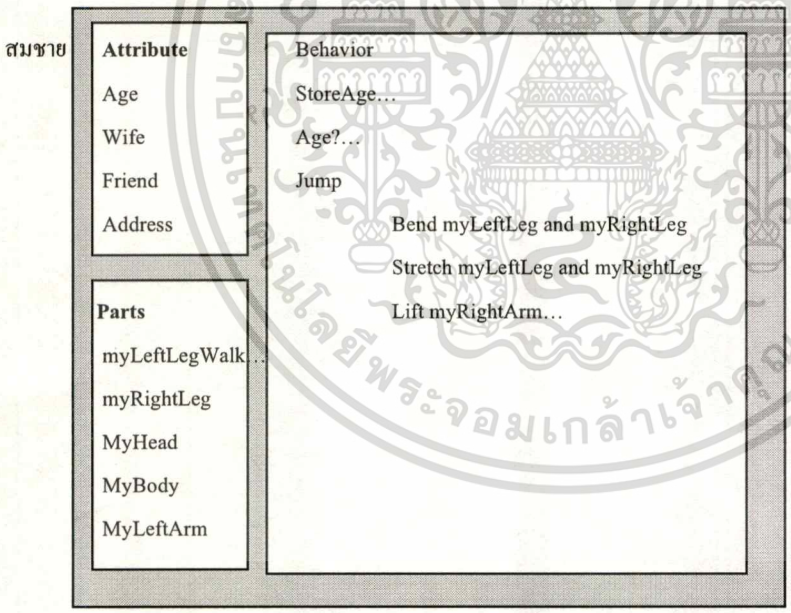
รูปที่ 2-1 ผังแสดงคำนิยามของ Object

ตัวอย่าง



รูปที่ 2-2 ตัวอย่าง Object ของคนที่ชื่อ “สมชาย”

ดังนั้นเมื่อเรามองเข้าไปใน Object จะประกอบด้วย attribute, part และ behavior ดังรูป

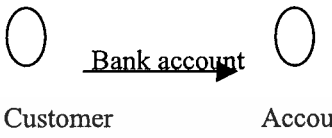


รูปที่ 2-3 ตัวอย่างส่วนประกอบภายใน Object ของคนที่ชื่อสมชาย

- **Message passing**

ความเกี่ยวข้องกันระหว่าง Object แบ่งเป็น 2 ลักษณะคือ

1. Static relation เช่น ลูกค้าเป็นเจ้าของบัญชี



รูปที่ 2-4 ตัวอย่าง Static relation ของ Object

2. Dynamic relation เช่น ผู้ฝากเงินฝากเงินเข้าไปในบัญชี

รูปที่ 2-5 ตัวอย่าง Dynamic relation ของ Object

Dynamic relation โดยความหมายของ OO จะหมายถึง Object หนึ่งจะกระตุ้น (stimuli) ไปยังอีก Object หนึ่ง ส่วนในทาง programming จะเรียกเป็นการส่ง message

ดังตัวอย่าง หากต้องการให้สมชาย jump ก็ต้องส่ง stimulus ไป เมื่อสมชายได้รับก็จะกระทำ action ต่างๆ ได้แก่ ส่ง stimulus ไปยัง arm และ leg

- **Class และ Instance**

Class คือ เซตของ Object

Instance เป็น Object ที่สร้างจาก Class

ตัวอย่าง Class ของคน ประกอบด้วย คนแต่ละคน โดยมี สมชายเป็น Instance ของ Class คน หรือ คนชื่อ "ประดิษฐ์" เป็น Instance ของ Class คนเช่นกัน โดยแต่ละ Instance จะมีความแตกต่างกันระหว่าง Object อื่นๆ (มีเอกลักษณ์ หรือ identity)

2.3 คุณสมบัติที่สำคัญของ Object-oriented

- **Encapsulation และ Information hiding**

ข้อมูล (Information) และ หน้าที่การทำงาน หรือ พฤติกรรม (Operation / Behavior) จะถูกจับมัดรวมเข้าด้วยกัน (Encapsulation) ซึ่งจะสนับสนุนให้เกิด "การซ่อนข้อมูล" (Information hiding) และสิ่งที่เราจะเห็นได้ของ Object คือ Instance ของมัน

ทำให้การเข้าถึงข้อมูลจะกระทำโดยเรียกใช้ Operation ของ Object ขึ้นมาทำงาน ดังคำกล่าวที่ว่า "The only way to reaching the information in an Object is by using its operation." (Skylstad, G. and Wuwongse, V. 1998)

"Abstract Data Type (ADT)" คือ model ที่มี Operation ของมันที่จะกระทบ ต่อ model (Skylstad, G. and Wuwongse, V. 1998)

ข้อดี ของ Encapsulation คือ สามารถลดความซับซ้อนลง เนื่องจากสามารถใช้งานโดยรู้เพียงคุณลักษณะต่างๆ เท่านั้น โดยไม่ต้องรู้ถึงโครงสร้างภายใน

- **Inheritance**

การสืบทอดคุณสมบัติทั่วๆ ไปที่เหมือนกัน ระหว่าง Class

ตัวอย่าง Class ของผู้ชาย inherits Class ของคน

ข้อดี ของ Inheritance คือ

1. สามารถนำนิยามโดยทั่วๆ ไปมาใช้ใหม่ (Reuse common description)
2. การปรับปรุง แก้ไข ทำเพียงที่เดียวเท่านั้น (Modification can be made in only one place)
3. ลดความซ้ำซ้อนสำหรับโปรแกรมย่อยเพื่อให้ง่ายต่อการทำความเข้าใจ (Redundancy can be reduced leading to smaller models that are easier to understand)

"Overriding" คือคุณสมบัติที่สามารถนิยาม behavior หรือ information ใหม่ได้สำหรับ Class ที่สืบทอด มาจาก Super Class

- **Polymorphism**

โดยความหมายคือ "having many forms" หมายถึงการส่ง message เดียวกันให้กับ Object ที่ต่างกันจะแสดงพฤติกรรมที่แตกต่างกัน

ตัวอย่าง การบวกของ String และ int จะมีสัญลักษณ์เดียวคือ + แต่ทำกับข้อมูลต่างชนิดกัน

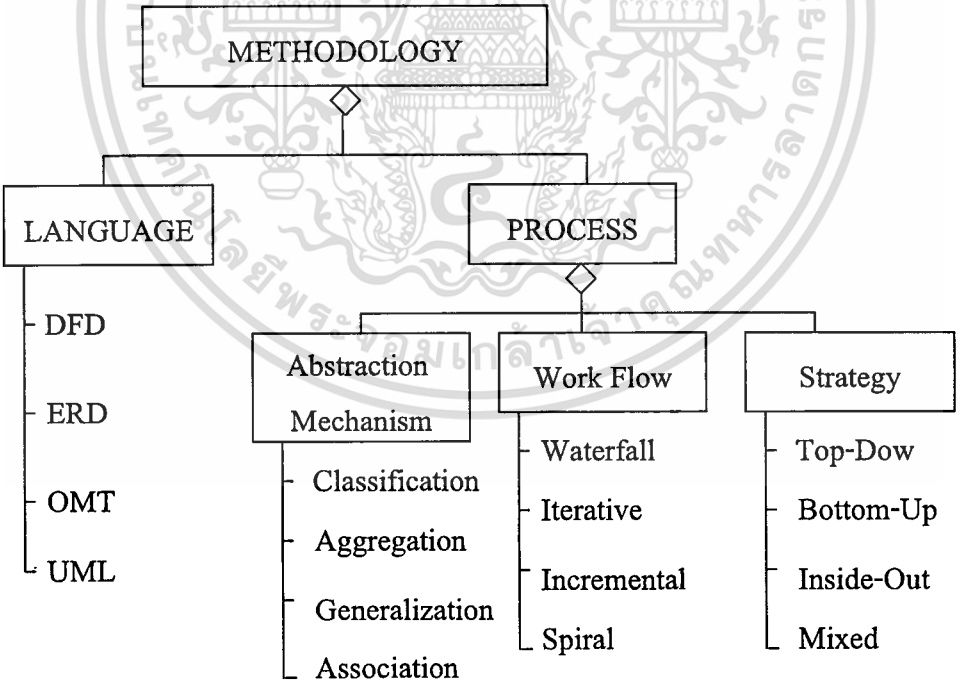
ข้อดี ของ polymorphism คือ สนับสนุนการนำกลับมาใช้ใหม่ (reuse) และมีการเปลี่ยนแปลงแก้ไขได้

2.4 Object-oriented methodology

การพัฒนาระบบด้วย Object-oriented methodology นักวิเคราะห์ระบบจะต้องมีแนวคิดหรือระเบียบวิธีคิด ในเชิงวัตถุที่ได้กล่าวมาแล้ว ตั้งแต่กระบวนการวิเคราะห์ ออกแบบ การเขียนโปรแกรม ไปจนถึงการทดสอบการทำงานของระบบ หรือโปรแกรม (OO analysis, OO design, OO programming และ OO testing) ยกตัวอย่างเช่น การวิเคราะห์ระบบงาน จะต้องวิเคราะห์ให้ได้ว่าควรจะมี Class อะไรบ้าง แต่ละ Class มี พฤติกรรมการสื่อสารระหว่างกันอย่างไร เกิดเหตุการณ์อะไรบ้าง เป็นต้น จึงจะได้ออกมาเป็นระบบที่ต้องการ ต่างจากการวิเคราะห์และออกแบบในระบบแบบดั้งเดิม ที่มองแยกกันระหว่างการออกแบบในส่วนของข้อมูล กับส่วนของกระบวนการทำงาน

2.5 ส่วนประกอบที่สำคัญของ methodology

System modeling methodology โดยทั่วไปประกอบด้วย 2 ส่วน คือ language และ process model โดย language คือสัญลักษณ์ (notation) ที่ใช้ในการอธิบาย แสดงความหมายของระบบงาน ส่วน process หมายถึง การบอกกล่าวว่าควรจะทำอะไรในขั้นตอนใดของการวิเคราะห์และออกแบบระบบงาน โดยสามารถแสดงได้ดังรูป



รูปที่ 2-6 โครงสร้างส่วนประกอบของ System Development Methodology

ดังนั้น Object-oriented methodology จึงประกอบด้วย 2 ส่วน เช่นกัน โดย Object-oriented methodology ที่ศึกษาและใช้ในการจัดทำโครงการคือจะเลือก UML เป็น Language และใช้ Iterative model เป็นส่วนของการไหลเวียนของงาน (work flow) ซึ่ง iterative process model ที่ได้นำมาใช้ ได้ประยุกต์ใช้จาก Rational Unified Process(RUP) ซึ่ง บริษัท Rational ได้คิดค้นขึ้นมา โดยจะกล่าวถึงรายละเอียดของ UML ที่สัมพันธ์กับ iterative process model ในบทถัดไป

บทที่ 3

Unified Modeling Language (UML)

และ Iterative model

3.1 UML คืออะไร

UML (Unified Modeling Language) เป็นภาษา หรือ รูปแบบจำลอง มาตรฐานที่ใช้ในการจำลองแบบในรายละเอียด (blue print) สำหรับการผลิตสร้าง (construct) ซอฟต์แวร์ เช่นเดียวกับแบบจำลองทางธุรกิจ (business modeling) โดย UML จะเป็นการเสนอรูปแบบการปฏิบัติจริงในทางวิศวกรรมที่ดี (best engineering practices) อีกทั้งยังเหมาะกับระบบที่มีขนาดใหญ่และยุ่งยากซับซ้อนอีกด้วย

UML เป็นรูปแบบที่รวมข้อดีของรูปแบบอื่น ๆ เข้าไว้ด้วยกัน คือ

- Data Modeling Concept (Entity Relation Concept)
- Business Modeling (Work Flow)
- Object Modeling
- Component Modeling

นอกจากนี้ UML ยังสามารถใช้ได้ตลอดทั้งโครงการหรือการทำงาน เช่นเดียวกับ SDLC (Software Development Life Cycle)

3.2 ประวัติศาสตร์ของ UML

การกำหนดรูปแบบภาษาการวิเคราะห์และออกแบบเชิงวัตถุ เริ่มปรากฏในระหว่างกลางทศ-วรรษที่ 1970 ถึงปลายทศวรรษที่ 1980 มีการทดลองด้วยแนวความคิดที่แตกต่างกันอย่างแพร่หลายเกี่ยวกับการวิเคราะห์ และออกแบบเชิงวัตถุ และในช่วงระหว่างปี 1989 – 1994 ได้มีการเพิ่มจำนวนของรูปแบบภาษาจากเดิมซึ่งมีน้อยกว่า 10 รูปแบบไปเป็นมากว่า 50 รูปแบบ และผู้ใช้จำนวนมากที่ใช้วิธีการทางเชิงวัตถุ (Object-oriented) ก็ยังไม่พบกับวิธีที่ตนเองพอใจได้

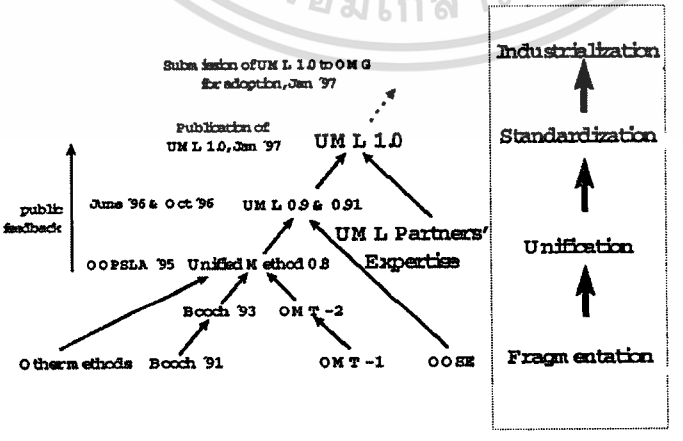
จนกระทั่งกลางทศวรรษที่ 1990 ได้เริ่มมีการพัฒนา UML ขึ้นในเดือนตุลาคม 1994 โดย Grady Booch และ Jim Rumbaugh ของ Rational Software Corporation ได้มีการรวมรูปแบบของ

Booch และ OMT (Object Modeling Technique) เพื่อศึกษาวิธีการในเชิงวัตถุ หลังจากนั้นในเดือนตุลาคม 1995 ก็มีรูปแบบของ Unified Method 0.8 แต่ก็ไม่ประสบความสำเร็จไปในปี 1995 นั้นIvar Jacabson ก็ได้นำรูปแบบนั้นมารวมกับวิธี OOSE (Object-oriented Software Engineering) และสุดท้ายในปี 1996 จึงมี UML 1.0 มาเป็นมาตรฐานโดยการร่วมมือของหลาย ๆ บริษัท เช่น Digital Equipment Corp., i-Logix, Intellicrop, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI และ Unisys

ปัจจุบัน UML ได้รับการรองรับจาก OMG (Object Management Group) ให้เป็นมาตรฐานหนึ่ง และไม่มีใครเป็นเจ้าของ เป็นรูปแบบเปิด กฎเกณฑ์ที่ทำให้ UML ประสบความสำเร็จ คือ

1. UML เป็นสิ่งที่ทำแล้วได้ผลในหลาย ๆ วิธีที่เป็นลักษณะรูปแบบทางภาษาที่เคยเสนอมาในอดีต
2. UML เป็นการรวมรูปแบบที่ใช้ได้ในหลาย ๆ ระบบ (business versus software) , ขั้นตอนการพัฒนาระบบ (development phase)

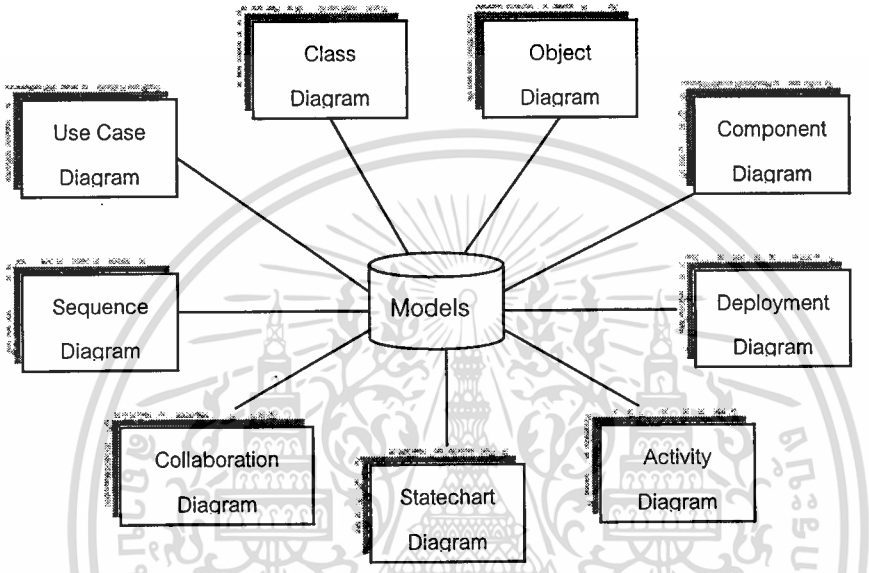
หลาย ๆ องค์กรได้มีการสนับสนุนมาตรฐานของ UML ตั้งแต่พื้นฐานรูปแบบทางภาษา (modeling language) ซึ่งเป็นวิธีการเชิงวัตถุ UML เป็นที่ยอมรับกันอย่างกว้างขวาง โดย UML 1.0 เป็นเวอร์ชันที่สามารถใช้ได้และมีเสถียรภาพ ในเดือนมกราคม 1997 เอกสารเกี่ยวกับ UML Version 1.0 ก็ได้รับการตอบสนองโดย Object Management Group (OMG) Analysis & Design Task Force's RFP-1 และในกลางปี 1997 OMG ได้มีการยอมรับมาตรฐานของ UML จึงทำให้หลาย ๆ องค์กรและผู้ผลิตได้ยอมรับมาตรฐาน UML จึงมีการใช้วิธีการสนับสนุนอื่น ๆ ทางด้าน UML มีเครื่องมือที่ช่วยในการพัฒนา ตลอดจนองค์กรที่ช่วยในการฝึกอบรม



รูปที่ 3-1 แผนภาพแสดงประวัติศาสตร์ของ UML

3.3 ประเภทของ UML Diagram

UML ได้กำหนด diagram ทั้งหมด 9 diagram ดังแสดงในรูปที่ 3-2 เพื่อใช้ในการจำลองแบบระบบงาน โดยการมองในแง่มุมต่างๆ (diagram แต่ละชนิด) เพื่อให้สามารถเข้าใจระบบงานให้มากที่สุด โดยผู้จำลองแบบไม่จำเป็นต้องใช้ทุก diagram โดยสามารถเลือกใช้ diagram ที่เหมาะสม



รูปที่ 3-2 แผนภาพแสดง UML diagram ทั้งหมด 9 diagram

UML ทั้ง 9 diagram สามารถจัดแบ่งออกเป็นกลุ่มต่าง ๆ ได้ดังนี้

3.3.1 Functional Model

Functional Model เป็นแบบจำลองที่ใช้ในการแสดงความต้องการของระบบทั้งหมด ช่วยในการอธิบายรายละเอียดหลักๆ ภายในวัตถุ แสดงให้เห็นการไหลของข้อมูลในแต่ละการทำงาน โดยจะสนใจเพียงแค่ว่ามีงานอะไรบ้างที่ต้องทำ เครื่องมือที่ใช้ในการแสดงความต้องการของระบบคือ Use Case Diagram ใช้ในการแสดงความต้องการของระบบทั้งหมดในลักษณะที่ผู้ใช้งานสามารถเข้าใจได้ง่าย โดยจะถูกนำไปใช้ต่อไปใน Phase ต่างๆ ของการวิเคราะห์และออกแบบระบบ

3.3.2 Object Model

Object Model เป็นแบบจำลองที่ใช้ในการแสดงโครงสร้างของระบบ โดยจะแสดง

ในรูปของ Class ต่างๆ พิจารณาจากความต้องการของระบบ ที่แสดงอยู่ใน Functional model เครื่องมือที่ใช้ในการแสดงโครงสร้างของระบบ คือ Component Diagram แสดงถึงโครงสร้างและความสัมพันธ์ขององค์ประกอบต่างๆ ของซอฟต์แวร์ ส่วน Class Diagram แสดงถึงโครงสร้างและความสัมพันธ์ของ Class Object Diagram แสดงถึงโครงสร้างและความสัมพันธ์ของ Instance และ Deployment Diagram แสดงถึงความสัมพันธ์ทางกายภาพระหว่างส่วนประกอบของฮาร์ดแวร์และซอฟต์แวร์

3.3.3 Dynamic Model

Dynamic Model เป็นแบบจำลองที่ใช้ในการแสดงถึงการทำงานระหว่าง Object ต่างๆ ตามการส่งข้อความ (Message) หรือเมื่อเหตุการณ์ (Event) ต่างๆ ได้เกิดขึ้น Object ในที่นี้หมายถึง Instance ที่สร้างขึ้นจาก Class ที่ได้ออกแบบไว้ใน Object Model มีคุณสมบัติ และพฤติกรรมเช่นเดียวกับ Class ต้นแบบ ในการทำงานของระบบจะประกอบขึ้นจากการส่งข้อความไปมาระหว่าง Object เหล่านั้น เมื่อมีการทำงานไปเรื่อยๆ แล้ว Object อาจจะมีการเปลี่ยนสถานะ(State) ไปตามเหตุการณ์ที่เกิดขึ้นได้ เพื่อให้เป็นตามที่กำหนดไว้ใน Functional Model เครื่องมือที่ใช้คือ Sequence Diagram แสดงให้เห็นถึงการแลกเปลี่ยนข่าวสารระหว่าง Object ต่างๆ, Collaboration Diagram สามารถแสดงถึงความสัมพันธ์ระหว่าง Object ภายใน, Activity Diagram แสดงถึงสถานะการทำงานในแต่ละ action และ State Diagram แสดงวงจรชีวิตของ Object บอกถึงว่าเหตุการณ์ต่างๆ มีผลกระทบอะไรเกิดขึ้นบ้าง

3.4 ความหมายของแต่ละ UML Diagram

จะอธิบายความหมายและสัญลักษณ์ต่างๆ ที่ใช้ใน UML diagram โดยจะอธิบายเฉพาะ diagram ที่เลือกใช้ในการพัฒนาในโครงการพัฒนาระบบงานนี้ 4 diagram ได้แก่ Use Case Diagram, Class Diagram, Component Diagram และ Sequence Diagram ดังนี้

3.4.1 Use Case Diagram

Use Case Diagram เป็นแนวความคิดของไอวาร์ จาคอบสัน (Ivar Jacobson) ซึ่ง OMG ได้รวมไว้ในมาตราฐานของ UML ด้วย และมีวิธีการอื่นๆ นำแนวคิดนี้ไปใช้อย่างกว้างขวาง Use Case Diagram เป็นสิ่งที่ใช้ในการแสดงความต้องการของระบบทั้งหมดในลักษณะที่ผู้ใช้งานสามารถเข้าใจได้ง่าย โดยแสดงความสัมพันธ์ระหว่างผู้ใช้กับระบบต่างๆสัญลักษณ์ได้ดังนี้

- **Actor** คือสิ่งที่อยู่ภายนอกระบบที่จะพัฒนาขึ้นทั้งหมด ที่ต้องการแลกเปลี่ยนหรือส่งข้อมูลให้กับระบบ โดยที่ Actor อาจจะเป็นคน ระบบ หรือโปรแกรมอื่นๆ ก็ได้ โดยมากจะเป็นผู้เริ่มทำงานกับ Use Case เช่นผู้ใช้ระบบ เป็นต้น

รูปที่ 3-3 สัญลักษณ์ของ Actor

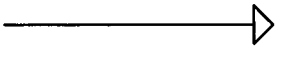
- **Use Case** จะเป็นตัวแทนงานที่เกิดขึ้นในขั้นตอนต่างๆ โดยจะใช้สัญลักษณ์เป็นรูปวงรี หรือวงกลม ถ้า Use Case ใดมีกรอบสี่เหลี่ยมสีเทาล้อมรอบแสดงว่ามี Diagram ย่อยที่ใช้อธิบายรายละเอียดของ Use Case ต่อไป

รูปที่ 3-4 สัญลักษณ์ของ Use Case

- **Communication** เป็นการแสดงความสัมพันธ์หรือการติดต่อสื่อสารกัน (การรับและให้ข้อมูลข่าวสารแก่กันและกัน) ระหว่าง Actor และ Use Case ซึ่งอาจจะเป็นการสื่อสารแบบทางเดียวหรือ 2 ทางก็ได้ แสดงด้วยสัญลักษณ์เส้นที่มีหัวลูกศรสีดำ

รูปที่ 3-5 สัญลักษณ์ของ Communication

- **Relationship** เป็นการแสดงความสัมพันธ์ระหว่าง Use Case กับ Use Case โดยใช้เส้นที่มีหัวลูกศรสีขาว ความสัมพันธ์ระหว่าง Use Case มีได้ 2 แบบ คือ Extends และ Uses



รูปที่ 3-6 สัญลักษณ์ของ relationship

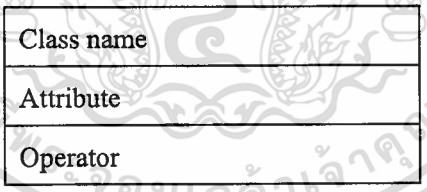
Extends เป็นการเพิ่มการทำงานให้กับ Use Case โดยการเรียกใช้จากอีก Use Case หนึ่ง ลูกศรจะออกจาก Use Case ที่ถูกเรียกใช้งานไปยัง Use Case ที่ต้องการเพิ่มเติมการทำงาน และจะมีข้อความ <<extends>> ระบุอยู่ข้างๆ เส้น

Uses เป็นการแสดงให้เห็นถึงการ Inherit หน้าที่การทำงานจาก Use Case หนึ่ง ไปอีก Use Case หนึ่ง ปลายลูกศรจะอยู่ที่ Use Case หลักที่จะถูกถ่ายทอดความสามารถออกไป โดยจะมีข้อความ <<uses>> ปรากฏอยู่ข้างๆ เส้น

3.4.2 Class Diagram

Class Diagram เป็น model ชนิดหนึ่งที่เป็น Static model จะเป็นการอธิบายถึง Class และความสัมพันธ์ระหว่าง Class ที่มีโครงสร้างของข้อมูลรวมถึงพฤติกรรมของข้อมูลที่แตกต่างกัน ซึ่งClass 1 Class สามารถกำหนดทิศทางของการนำไปสร้างโปรแกรมและการสร้าง Classใน Object oriented ได้

Class ใช้สัญลักษณ์เป็นรูปสี่เหลี่ยม จะอธิบายถึงกลุ่มของ Object ต่างๆ ที่มีคุณลักษณะ (Attribute) หน้าที่การทำงาน (Operation) และความสัมพันธ์ (Relationship) ขั้นพื้นฐาน ซึ่งในแต่ละClass ประกอบไปด้วยรายละเอียดพื้นฐาน 3 ส่วนดังรูป

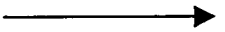


รูปที่ 3-7 สัญลักษณ์ของ class

- Class Name : แสดงชื่อของ Class ที่กำหนดในระบบ
- Attribute : เป็นการกำหนดคุณลักษณะทั้งหมดที่มีภายใน Class บอกถึงรายละเอียดของข้อมูล (Attribute) ภายใน Class เดียวกันจะไม่ซ้ำกัน แต่ชื่อข้อมูลใน Class อาจจะไปซ้ำกับชื่อข้อมูลใน Class อื่นได้
- Operation : เป็นส่วนที่ใช้อธิบายใน Class นั้นมี Method อะไรบ้าง มีการรับค่าชุดของตัวแปร (Argument) อะไรบ้าง หรือมีการส่งค่าออกไปหรือไม่
- ความสัมพันธ์ระหว่าง Class (Relationship)

UML ได้เตรียมความสัมพันธ์ระหว่าง Class ไว้ จะมีความสัมพันธ์อยู่ 4 แบบ ดังนี้

1. Association : จะแสดงความสัมพันธ์ระหว่าง Class จะมีได้ทั้งทางเดียว และ 2 ทาง มีสัญลักษณ์ดังนี้

   ความสัมพันธ์แบบทางเดียว : 

   ความสัมพันธ์แบบสองทาง :

รูปที่ 3-8 สัญลักษณ์ของ Association relationship

2. Aggregation : เป็นรูปแบบพิเศษของ association คือเป็นความสัมพันธ์ระหว่าง Whole และ parts ของมัน คือ Whole จะประกอบไปด้วย parts ต่างๆ ของมัน ดังนั้นการคงอยู่ของ parts จะต้องขึ้นกับ Whole หรือจะเรียกความสัมพันธ์แบบนี้ว่า "Whole-part" ก็ได้ มีสัญลักษณ์ดังนี้

   ความสัมพันธ์แบบ by Value 

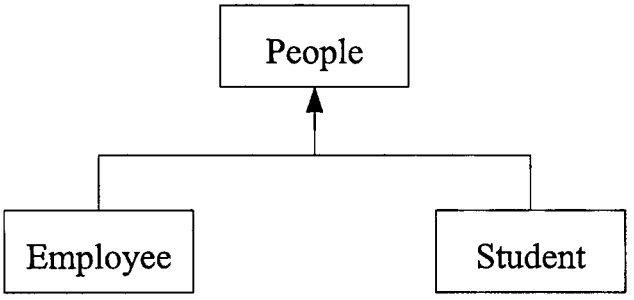
   ความสัมพันธ์แบบ by Reference 

รูปที่ 3-9 สัญลักษณ์ของ Aggregation relationship

3. Depends on : เป็นรูปแบบความสัมพันธ์แบบหนึ่งที่ใช้แสดงความสัมพันธ์กันระหว่างคลาส 2 คลาส ในแง่ที่คลาส หนึ่งเรียกใช้บริการของอีกคลาสหนึ่ง กล่าวคือ Class ของผู้ขอบริการขึ้นอยู่กับบริการของ Class ของผู้ให้บริการ แต่ไม่มีการขึ้นต่อกันภายในโครงสร้างของ Class มีสัญลักษณ์ดังนี้

รูปที่ 3-10 สัญลักษณ์ของ Depend on relationship

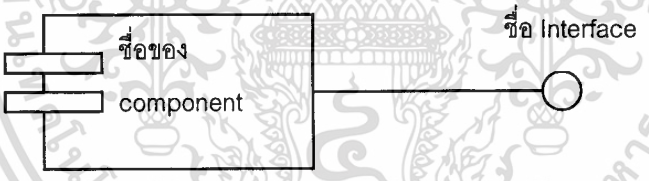
4. Generalization : ความสัมพันธ์รูปแบบนี้ใช้แสดงความสัมพันธ์ระหว่าง Class กับ Class ในแง่ที่ Class หนึ่งถ่ายทอดคุณสมบัติและโครงสร้างจากอีก Class หนึ่ง โดยเรียก Class ที่ถูกถ่ายทอดว่า Super Class และเรียกคลาสที่ทำการถ่ายทอดว่า Sub Class มีสัญลักษณ์ดังตัวอย่างในรูปที่ 3-11



รูปที่ 3-11 ตัวอย่างความสัมพันธ์แบบ Generalization

3.4.3 Component Diagram

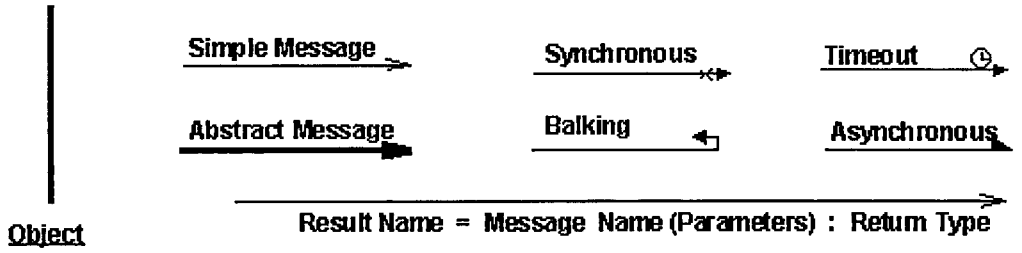
เป็น diagram ที่แสดงถึงโครงสร้างและความสัมพันธ์ระหว่างองค์ประกอบต่างๆ ของซอฟต์แวร์ ซึ่งองค์ประกอบดังกล่าวอาจเป็น source program หรือ library หรือ executable program ก็ได้ โดย Component Diagram จะเป็นแบบจำลองที่ประกอบด้วยองค์ประกอบต่างๆ ของระบบเชื่อมโยงกันโดยใช้ความสัมพันธ์ในลักษณะที่ขึ้นต่อกันในรูปของเส้นประที่มีหัวลูกศรชี้จาก component ลูก ไปยัง component หลัก ดังรูป



รูปที่ 3-12 สัญลักษณ์ที่ใช้ภายใน Component Diagram

3.4.4 Sequence Diagram

เป็น diagram ที่แสดงให้เห็นถึงการทำงานระหว่าง Object ต่างๆ ตามการส่งข้อความ และเมื่อเหตุการณ์ต่างๆได้เกิดขึ้น ซึ่งผู้เขียนโปรแกรมจะใช้ diagram ตัวนี้เป็นตัวช่วยเพื่อที่จะได้เขียนโปรแกรมให้ได้ตามที่ได้ออกแบบไว้ สัญลักษณ์ที่ใช้ใน Sequence Diagram มีดังนี้



รูปที่ 3-9 สัญลักษณ์ต่างๆ ที่ใช้ใน Sequence diagram

- Object : คือ Instance ของ Class ที่ทำหน้าที่รับ-ส่งข้อความ (Message) จาก Object อื่นๆและทำการตอบสนองตามข้อความนั้นๆเพื่อให้เกิดการทำงานในขั้นตอนต่างๆของระบบโดยจะใช้สัญลักษณ์เป็นเส้นตรงแนวตั้งและมีชื่อของ Object และ Class กำหนดอยู่ด้านล่าง
- Message : เป็นข้อความที่ส่งไปมาระหว่าง Object ที่ถูกเรียก ซึ่ง Message จะประกอบไปด้วยส่วนประกอบต่างๆดังต่อไปนี้
  - Result Name : คือตัวแปรที่จะจัดเก็บค่าที่ได้จากการส่งข้อความ (Message)
  - Parameters : คือ Argument ต่างๆที่ส่งให้กับ Object เพื่อทำงานตาม Method ที่กำหนดด้วย Message นั้นๆ
  - Return Type : คือประเภทของข้อมูลที่ส่งกลับมาหลังจากการทำงานของการส่ง Method นั้นๆ ถ้า Method นั้นกำหนดว่ามีการส่งค่ากลับมา
  - Constraint : คือเงื่อนไขการส่ง Message

3.5 Iterative model

จากที่ได้กล่าวไว้ว่าในการทำโครงการนี้ ได้นำ Iterative model ของ Rational Unified Process มาประยุกต์ใช้ ซึ่ง Rational Unified Process มีลักษณะที่สำคัญอยู่ สองประการคือ

ประการที่หนึ่ง Rational Unified Process มีลักษณะเป็น Use-Case Driven ซึ่งหมายความว่า Use-Case เป็น แนวความคิดหลักภายในกระบวนการทำงานหรือ process Use-Case จะถูกใช้ตลอดวงจรการพัฒนา เสมือนเป็น ตัวขับเคลื่อนสำหรับกิจกรรมอื่นๆ เป็น ตัวที่ทำการส่งผ่านข้อมูลไปผ่านไปยัง แบบจำลอง(model)ต่างๆ และ เป็นตัวช่วยในการตรวจสอบความถูกต้องตรงกัน (consistency) ระหว่าง แบบจำลองเหล่านั้น

ประการที่สอง Rational Unified Process มีลักษณะเป็น Architecture-Centric ซึ่งหมายความว่า สถาปัตยกรรมหรือ Architecture ถูกใช้ใน Rational Unified Process เสมือนเป็น สิ่งก่อสร้าง

พื้นฐาน (primary artifact) สำหรับการรวบรวมแนวความคิด การสร้าง การจัดการ และ การก่อให้เกิดระบบภายใต้การพัฒนา

นอกจากลักษณะสองประการนี้แล้ว Rational Unified Process มีการจัดแบ่งกระบวนการทำงานเป็น Lifecyle Phases ออกเป็น 4 phases คือ

- **Inception** เป็นช่วงของการกำหนดขอบเขตของโครงการ
- **Elaboration** เป็นช่วงของการวางแผนโครงการ กำหนดฟังก์ชันการทำงานของระบบ รวมทั้ง สถาปัตยกรรมพื้นฐานที่จะใช้ในการจัดทำระบบ
- **Construction** เป็นช่วงของการสร้างโปรแกรม ซึ่งอาจจะมีได้หลาย iteration
- **Transition** เป็นช่วงของการส่งมอบโปรแกรมหรือระบบงาน ให้กับกลุ่มผู้ใช้งานระบบ

ภายในแต่ละ phase ที่กล่าวมา จะมีจำนวนของ iterations อยู่ จำนวนของ iterations ต่อ phase นั้น จะไม่ตายตัว ในแต่ละ iteration จะได้ผลลัพธ์เป็น โปรแกรมหรือระบบงานที่สามารถทำงานได้ ซึ่งจะเป็นส่วนหนึ่งของโปรแกรมหรือระบบงานใหญ่อีกที ซึ่งเป็นลักษณะเด่นอย่างหนึ่งของ Iterative Model

นอกจากนี้ Rational Unified Process จะมี Workflows หลักๆ ที่ใช้ในการพัฒนาระบบอยู่ ซึ่งแต่ละ workflow ก็จะมีการสร้าง แบบจำลอง ต่างๆ ขึ้นมา ซึ่ง Workflows ดังกล่าวประกอบไปด้วย

- **Business Modeling Workflow**

เป็นขั้นตอนในการทำความเข้าใจถึงโครงสร้างขององค์กร และเป็นการทำความเข้าใจร่วมกันระหว่างผู้ใช้งานกับผู้พัฒนา ซึ่งโดยทั่วไปผลลัพธ์ที่ได้มักจะเป็น business use-case models และเอกสารประกอบอื่นๆ

- **Requirements Workflow**

เป็นขั้นตอนในการทำข้อตกลงร่วมกันว่าระบบจะสามารถทำอะไรได้บ้าง โดยอาจจัดทำเป็น use-case model พร้อม use-case description หรืออาจจะมีการจัดทำ User-Interface Prototype

- **Analysis and Design Workflow**

เป็นขั้นตอนในการทำการแปลงความต้องการระบบ ไปสู่การออกแบบระบบที่ควรจะเป็น ซึ่งจะมีการกำหนดสถาปัตยกรรมของระบบ มีการปรับแต่งแบบที่ได้ทำการออกบบไว้ ให้เหมาะสมกับสภาพแวดล้อมของการสร้างระบบ โดยผลลัพธ์ของขั้นตอนนี้อาจจะประกอบด้วย design model classes design packages software architecture document และ data model

- **Implementation Workflow**

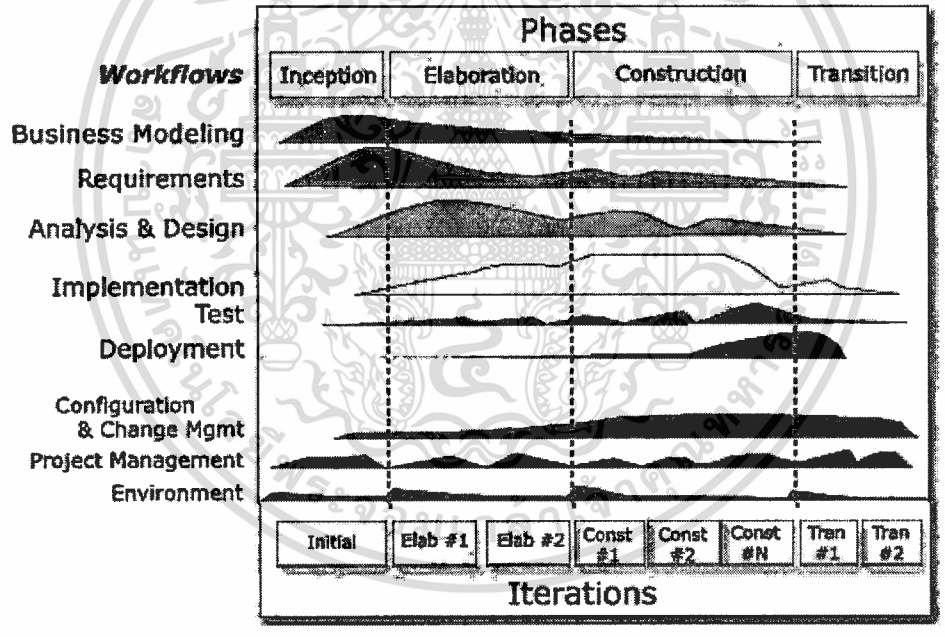
เป็นขั้นตอนในการกำหนดโครงสร้างของ source code มีการสร้าง classes และ

objects ในรูปแบบของ components โดยผลลัพธ์ที่ได้จากขั้นตอนนี้หลักๆ จะเป็น implementation model และ การกำหนด components

- **Test Workflow**

เป็นขั้นตอนของการทบทวนการโต้ตอบกันของ objects การทบทวนการเชื่อมต่อกันที่เหมาะสมของ component ทั้งหมด และ เป็นการทบทวนความต้องการทั้งหมดว่าถูกจัดสร้างอย่างถูกต้องหรือไม่ โดยทั่วไป จะมีการจัดทำ test model มีการกำหนด test cases procedures และ scripts รวมทั้งมีการทำ test plan ไว้ด้วย

รูปที่ 3-10 จะแสดงความสัมพันธ์ระหว่าง Phase Iteration และ Workflow ซึ่งจะเป็นลักษณะที่เรียกว่า Iterative Model โดยในแต่ละ phase จะมีได้หลาย iteration และ ใน หนึ่ง iteration ก็จะมีกิจกรรมที่เกิดขึ้นจากทุกๆ workflow



รูปที่ 3-10 ความสัมพันธ์ระหว่าง Phase Iteration และ Workflow

บทที่ 4

Object-oriented Case Tool : Rational Rose

4.1 Rational Rose 2000 Enterprise Edition

Rational Rose 2000 Enterprise Edition เป็น Visual Modeling tool และ Object-oriented Case Tool ที่สามารถสร้าง UML diagram ทั้งหมด เพื่อใช้ในการจำลองระบบงานได้ ด้วยการวาด diagram ต่างๆ โดยมีตัวจัดการ diagram และกำหนดคุณสมบัติต่างๆ ของ diagram ที่จำเป็น การเชื่อมโยงกันระหว่าง diagram และการจัดการเอกสารจากแบบจำลองที่ได้สร้างขึ้น รวมทั้งมีตัวช่วยอัตโนมัติในการสร้าง Class และ Add-in ต่างๆ เพิ่มเติม เช่น Rational Rose J และ Rational Rose Oracle8 ที่จะกล่าวถึงต่อไป

4.2 Rational Rose J

Rational Rose J เป็นส่วนเพิ่มเติมของ Rational Rose ที่มีคุณสมบัติดังต่อไปนี้ :

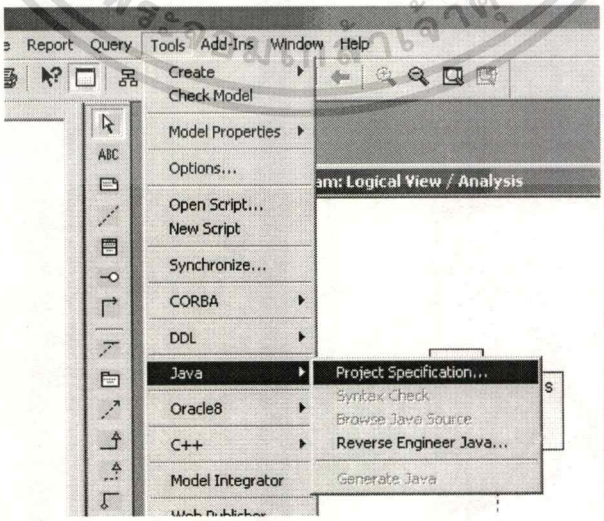
1. สนับสนุนวิวัตนาการจากขั้นวิเคราะห์ไปจนถึงการออกแบบ
2. สนับสนุนการทำงานเป็นทีมโดยการทำ configuration-management และ version control system
3. สามารถสร้าง source code ภาษา Java จาก model
4. สามารถเพิ่มเติม Classpath ภายใน model
5. สนับสนุนไฟล์ชนิด zip, jar และ cab ใน Classpath
6. สามารถลากและวางไฟล์ชนิด java, Class, zip, jar และ cab เข้าไปใน model โดยการทำ reverse-engineering อย่างรวดเร็ว
7. ขยายไฟล์แบบอัตโนมัติสำหรับไฟล์ชนิด zip, jar และ cab เมื่อขณะทำการ reversed-engineering
8. สามารถออกแบบ จำลอง ทำให้เห็นภาพของการสร้างโปรแกรมของภาษา Java ได้แก่ package, Class, interfaces, imports, inheritance, fields, methods และ modifiers
9. สามารถควบคุมคุณสมบัติต่างๆ อย่างอัตโนมัติ ได้แก่ finalizers, และ Static initializers, และสามารถกำหนด ชื่อ field-name ได้เอง
10. สนับสนุนการ application partitioning และ distribution

11. การ mapping อย่างฉลาดจาก model ไปสู่ code
12. สนับสนุน default package ต่างๆ
13. ปรับปรุงการสนับสนุน Class ภายในและ Classes อื่นๆ ที่ไม่อยู่ในระบบ
14. การสร้าง Bean (Get/Set)
15. โปรแกรม editor สามารถแสดงสีได้
16. สนับสนุน framework ขนาดใหญ่ ประกอบไปด้วย component ให้เลือกใช้งาน อาทิ
    - JDK Class Library
    - Java Database Connectivity (JDBC)
    - Java Generic Library (JGL)
    - Application Foundation Classes (AFC)
    - Windows Foundation Classes
    - Enterprise Java Frameworks (JNI, EJB, JTS)

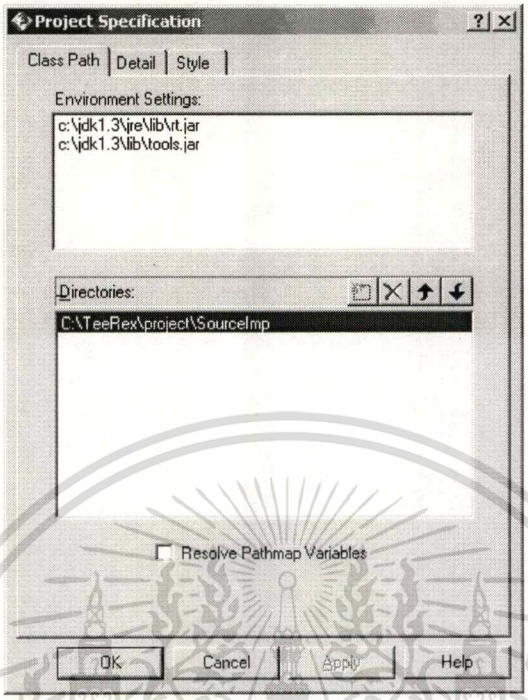
4.2.1 ตัวอย่างการทำ Forward engineering ด้วย Rational Rose J

Forward engineering คือ กระบวนการสร้างโปรแกรมภาษา Java จาก Class หรือ package หรือ component ใน Rational Rose model

ขั้นแรกจะต้องทำการเซต Classpath ให้ชี้ไปยัง directories หรือไฟล์ชนิด .jar, .cab, .zip ที่ต้องการใช้งาน โดยเพิ่มเติมในส่วนของ directories ที่เพิ่มไปจาก Classpath environment variable ที่ถูกเซตไว้ก่อนแล้ว ดังรูปที่ 4-1 และ 4-2

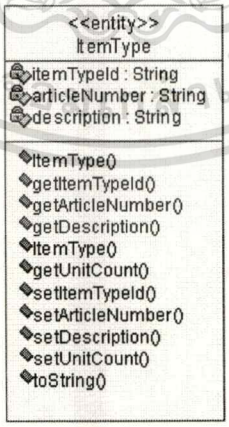


รูปที่ 4-1 วิธีการเข้าสู่หน้าจอการเซต Classpath



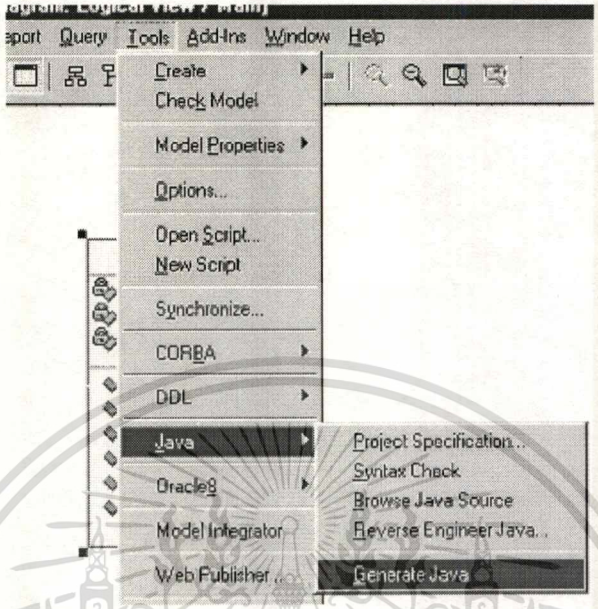
รูปที่ 4-2 หน้าจอขณะทำการเซต Classpath ที่เก็บของไฟล์ .java, .class, .jar, .zip

ขั้นที่ 2 ให้ทำการสร้าง class diagram ที่ต้องการ ดังเช่นตัวอย่างต่อไปนี้ สร้าง Class ItemType

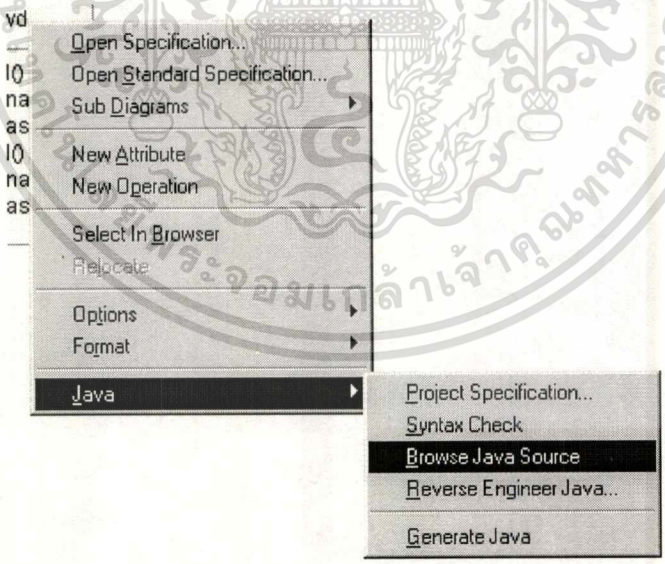


รูปที่ 4-3 Class ItemType ใน Rational Rose Model

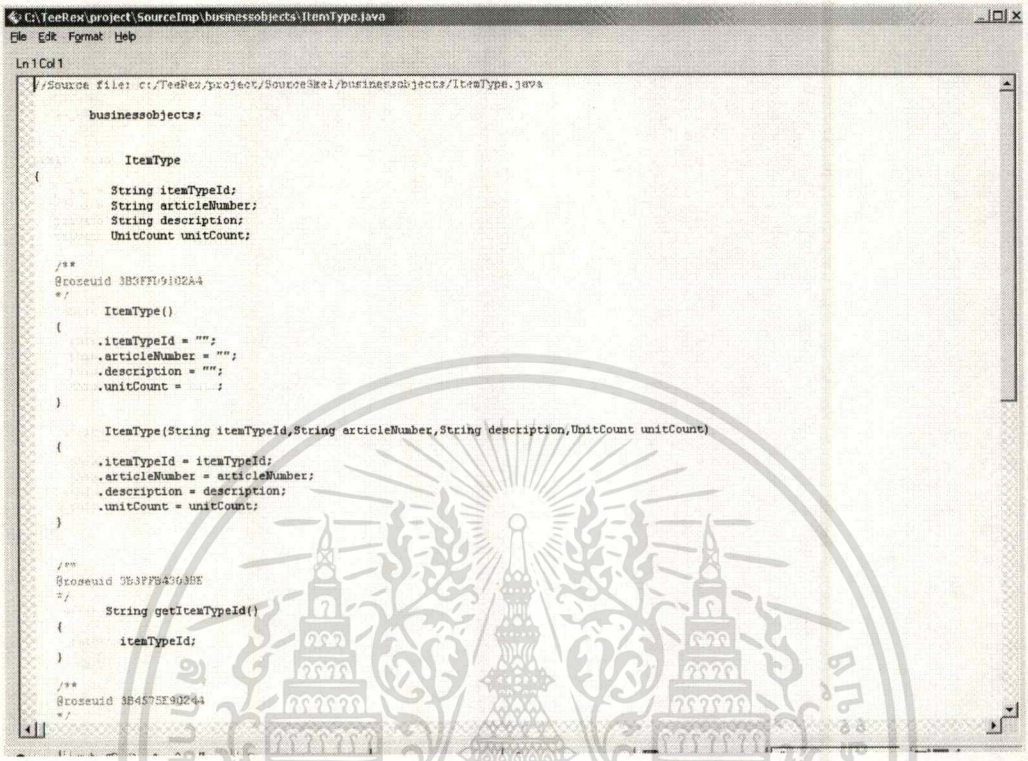
ขั้นที่ 3 เลือก Class ที่ต้องการ แล้วเลือก Tool > Java > Generate Java



รูปที่ 4-4 วิธีการสั่งให้สร้าง source code จาก class ที่ต้องการ



รูปที่ 4-5 วิธีการดูผลลัพธ์ source code



รูปที่ 4-6 ผลลัพธ์ source program

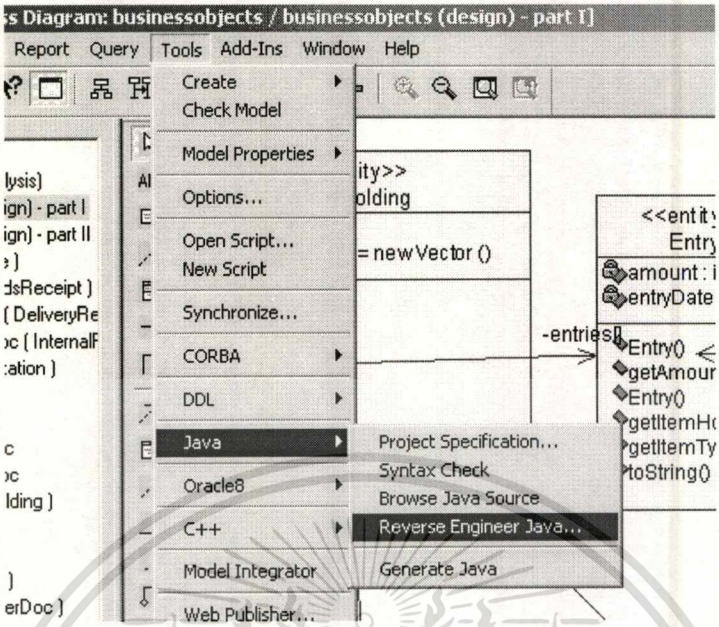
เมื่อทำตามขั้นตอนจะสามารถเข้าดูผลลัพธ์ได้ดังรูปที่ 4-5 และ 4-6 ไฟล์ .java ที่สร้างขึ้นจะบันทึกอยู่ใน directory ที่เซตไว้ใน Classpath

4.2.2 ตัวอย่างการทำ Reverse engineering ด้วย Rational Rose J

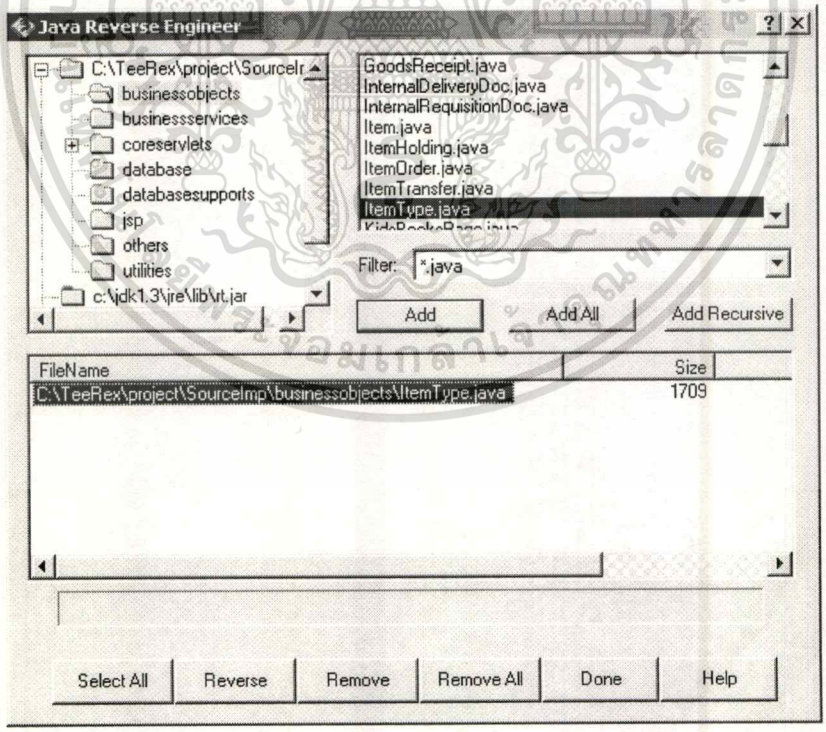
Reverse engineering คือ กระบวนวิเคราะห์ source code ภาษา Java แล้วแปลงกลับไปเป็น class หรือ component model ใน Rational Rose

เช่นเดียวกับกัน ขั้นแรกจะต้องทำการเซต Classpath ให้ชี้ไปยัง directories หรือไฟล์ชนิด .jar, .cab, .zip ที่ต้องการใช้งาน โดยเพิ่มเติมในส่วนของ directories ที่เพิ่มไปจาก Classpath environment variableที่ถูกเซตไว้ก่อนแล้ว ดังรูปที่ 4-1 และ 4-2

ขั้นที่ 2 เลือกไฟล์ .jar, .class ที่ต้องการ โดยเข้าที่เมนู Tool > Java > Reverse Engineering Java ดังรูปที่ 4-7 และ 4-8



รูปที่ 4-7 วิธีการเรียกหน้าจอเพื่อทำการ reverse engineering



รูปที่ 4-8 ตัวอย่างการเลือกไฟล์จาก directory มาทำ reverse engineering

ขั้นสุดท้าย หลังจากได้เลือกไฟล์เรียบร้อยแล้ว ให้กดปุ่ม Reverse โปรแกรมจะทำการ reverse ออกมาเป็น class diagram ที่ต้องการ

4.3 Rational Rose Oracle8

ในทำนองเดียวกัน Rational Rose Oracle8 คือส่วนเพิ่มเติมของ Rational Rose ที่มีคุณสมบัติดังต่อไปนี้ :

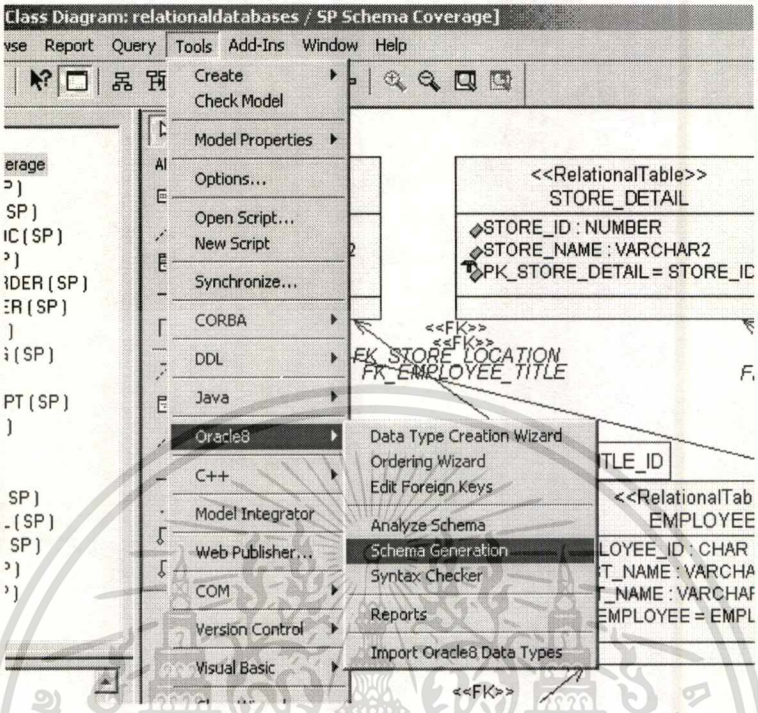
1. สามารถสร้าง Object models จาก Oracle8 relational schema และขยายความสามารถไปสู่มุมมองเชิงวัตถุของ Oracle8
2. สามารถทำให้มองเห็นภาพ หรือจำลอง relational database ที่มีอยู่เดิม และอำนวยความสะดวกในการค้นหาและประกอบเข้าด้วยกันกับ Business Object ซึ่งจะช่วยลดการลงทุนสำหรับการปรับปรุง relational database
3. สามารถทำให้ผู้พัฒนาระบบสร้างแบบจำลองทางธุรกิจในรูปของภาษาที่ใช้ในการพัฒนาโปรแกรมและฐานข้อมูล รวมทั้งตัวช่วยอัตโนมัติที่นำทางในการสร้าง Object model ที่พัฒนาจาก relational ไปสู่ Object-relational โดยสามารถสร้าง relational tables, triggers, Object types, Object views, VARRAYs, nested tables, และคุณสมบัติเชิงวัตถุต่างๆ ใน Oracle8

4.3.1 การทำ database normalization

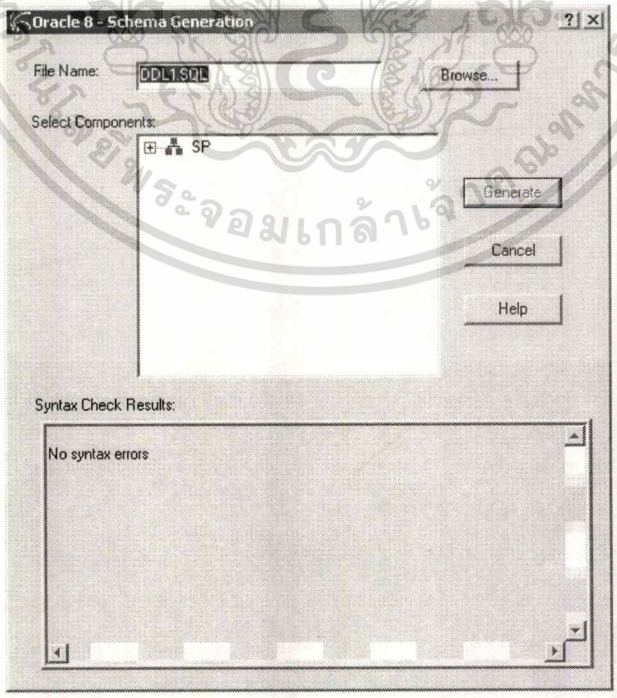
การจำลองแบบของข้อมูลสำหรับ Rational Rose Oracle8 ยังจำเป็นต้องผ่านกระบวนการออกแบบฐานข้อมูล และการทำ normalize ข้อมูลจริงเสียก่อน ในระบบงานนี้ ได้ทำการ Map ข้อมูลจาก Entity Classes ซึ่งจะได้กล่าวรายละเอียดในบทที่ 6 แล้วทำการ normalize ข้อมูลเพิ่มเติม จากนั้นจึงจะนำตารางที่ได้มาจากการ normalize มาสร้างเป็น Class Diagram รวมไว้ใน RelationalDatabases Package แล้วจึงจะนำไปสร้างเป็น database schema ด้วย Rational Rose Oracle8 ซึ่งจะกล่าวถึงในหัวข้อถัดไป

4.3.2 ตัวอย่างการสร้าง database schema ของระบบด้วย Rational Rose Oracle8

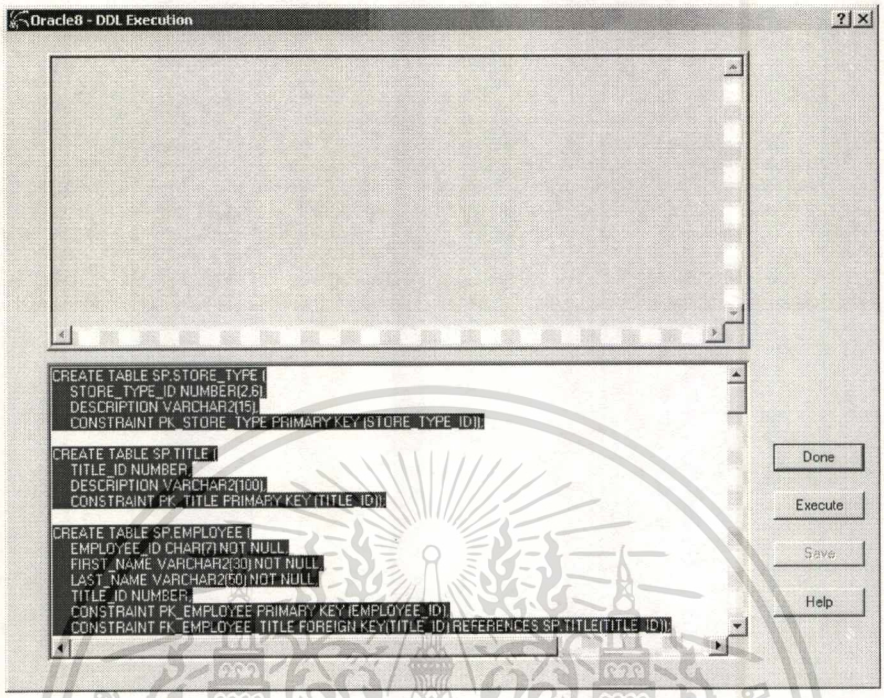
1. จาก Class Diagram ใน RelationalDatabases Package เลือก เมนู Tools > Oracle8 > Schema Generation Rational Rose ดังรูปที่ 4-9
2. ได้ผลลัพธ์ดังรูปที่ 4-10 กด Generate จะได้ Source code (DDL) ดังรูปที่ 4-11



รูปที่ 4-9 Class Diagram ที่จะนำมาสร้าง database schema



รูปที่ 4-10 หน้าจอของโปรแกรม Rational Rose ขณะทำการ Generate schema สำหรับ Oracle8



รูปที่ 4-11 หน้าจอผลลัพธ์หลังจากขั้นตอนที่ 1

<u>หมายเหตุ</u> หากต้องการสั่ง Create Database ด้วย Rational Rose ให้กดปุ่ม Execute โดยระบบจะต้องติดตั้ง DBMS เสร็จสมบูรณ์แล้ว แต่ในกรณีที่ไม่ต้องการจะใช้วิธี copy source code เก็บไว้เพื่อไปนำไปใช้ Create Database ในภายหลัง ผลลัพธ์ที่ได้ คือ Source code (Database Definition Language) ดังนี้

บทที่ 5
วิเคราะห์ระบบ

5.1 ความต้องการของระบบ

จากความเป็นมาที่กล่าวมาทำให้เกิดความต้องการในการพัฒนาระบบงานควบคุมวัสดุอะไหล่ เพื่อรองรับการทำงานในส่วนงานต่างๆ ที่มีความเกี่ยวข้องกับวัสดุอะไหล่นี้ ซึ่งโดยเบื้องต้นแล้ว ระบบงานใหม่นี้จะต้องสามารถทำงานเครือข่ายอินทราเน็ตขององค์กรได้ เนื่องจากกลุ่มผู้ใช้นั้นกระจายอยู่ตามภูมิภาคต่างๆ ของประเทศ และต่อมาในเรื่องของฟังก์ชันการทำงานของระบบนั้น ระบบงานใหม่นี้จะต้องรองรับการทำงานของผู้ใช้ในเรื่องต่างๆ โดยแบ่งตามกลุ่มของการทำงานของระบบดังนี้

5.1.1 การสั่งซื้อ-การรับอุปกรณ์

ผู้ใช้ที่เป็นพนักงานธุรการประจำคลังวัสดุกลาง(Main Store Administrator) จะต้องสามารถบันทึกข้อมูลใบสั่งซื้อที่ได้ออกไปได้ และจะต้องสามารถทำใบรับอุปกรณ์ จากการสั่งซื้อนี้เพื่อเพิ่มยอดของวัสดุอะไหล่เข้าในระบบได้

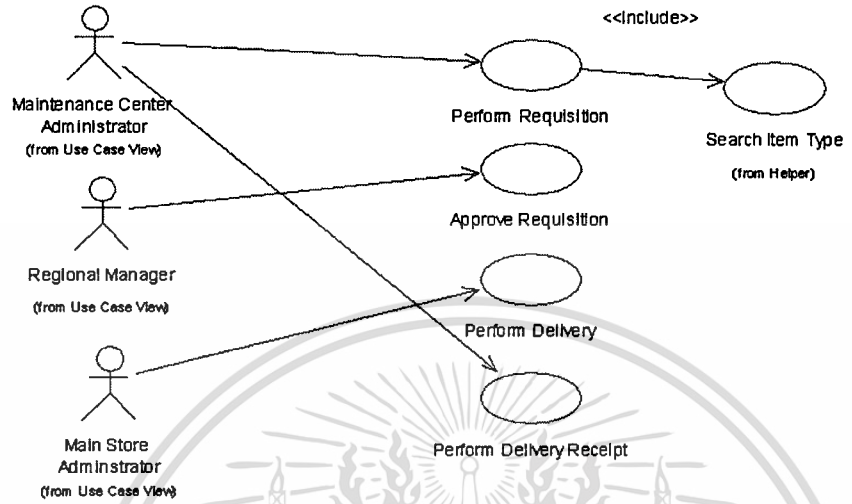
5.1.2 การเบิกอุปกรณ์

ผู้ใช้ที่เป็นพนักงานธุรการประจำศูนย์ซ่อมบำรุงย่อย(MC Administrator) จะต้องสามารถทำใบเบิกอุปกรณ์ เพื่อขอเบิกอุปกรณ์จาก Main Store มาเป็นวัสดุอะไหล่คงคลังไว้ที่ MCได้ โดยใบเบิกทุกใบจะต้องได้รับการอนุมัติจากผู้จัดการส่วนงานวิศวกรรมประจำภูมิภาค (Regional Manager) ด้วย และจะต้องสามารถทำใบรับอุปกรณ์จากการเบิกนี้ได้ เมื่อได้รับอุปกรณ์มาจาก Main Store แล้ว เพื่อทำการเพิ่มยอดวัสดุอะไหล่ภายใน MC

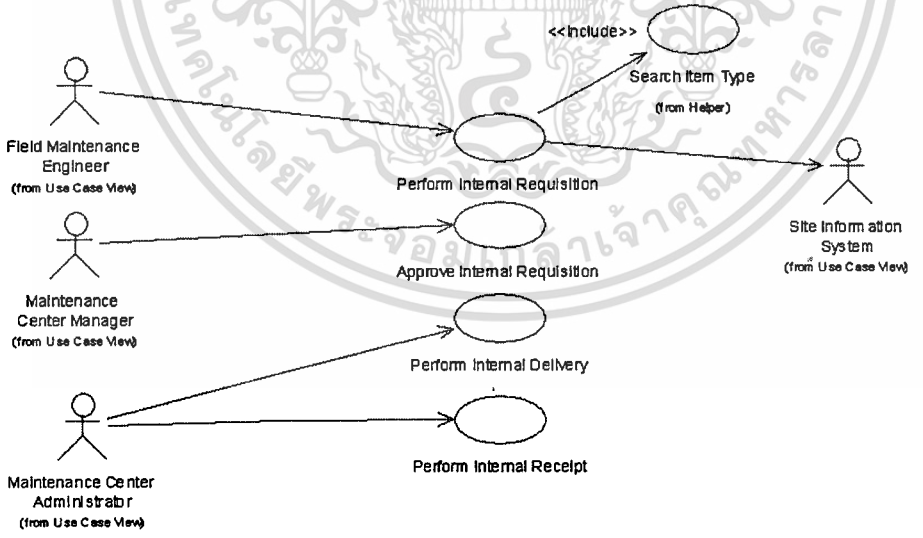
ในฝั่งของ Main Store พนักงานธุรการประจำคลังวัสดุกลางก็จะต้องสามารถทำใบส่งของตามใบเบิกที่ได้รับการอนุมัติแล้ว ได้ เพื่อเป็นการลดยอดของวัสดุอะไหล่ภายใน Main Store

5.1.3 การทำงานภายใน

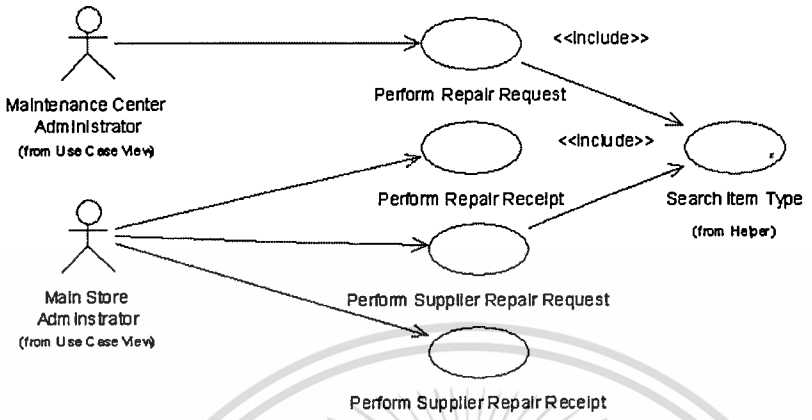
ผู้ใช้ที่เป็นวิศวกรซ่อมบำรุง (Field Maintenance Engineer) ของแต่ละ MC จะต้องสามารถทำใบเบิกอุปกรณ์ภายใน เพื่อเบิกอุปกรณ์ไปทำการซ่อมบำรุงสถานีฐานได้ โดยใบเบิกอุปกรณ์ภายในทุกใบจะต้องได้รับการอนุมัติจากผู้จัดการศูนย์ซ่อมบำรุงย่อย (MC Manager) ด้วย



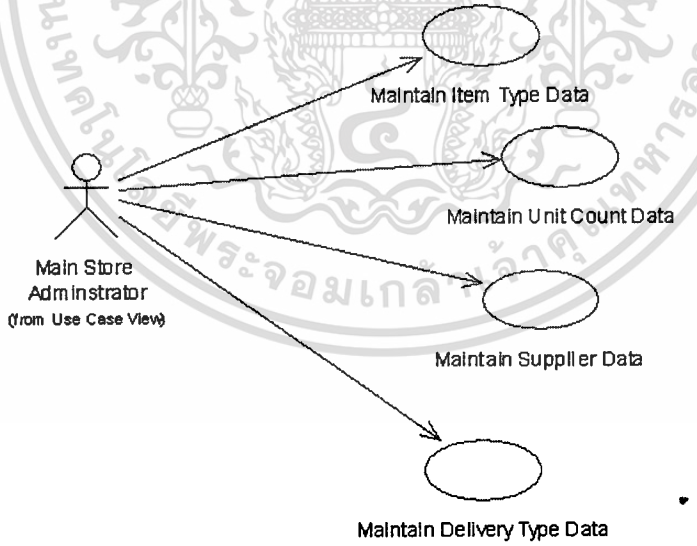
รูปที่ 5-2 Use Case Diagram ของ ระบบการระบบการเบิกจ่ายอุปกรณ์



รูปที่ 5-3 Use Case Diagram ของ ระบบการทำงานภายใน



รูปที่ 5-4 Use Case Diagram ของ ระบบการส่งซ่อม



รูปที่ 5-5 Use Case Diagram ของ ระบบการเพิ่มเติม-แก้ไขข้อมูลหลัก

5.3 Use Case Description

จาก Use Case Diagram เราสามารถที่จะทำการอธิบายรายละเอียด ขั้นตอนการทำงานหรือขั้นตอนการใช้งาน (Scenario) ของแต่ละ Use Case ได้ดังนี้

5.3.1 ระบบการสั่งซื้อ-รับอุปกรณ์ (Purchase Order and Goods Receipt)

5.3.1.1 Use Case - Entry Purchase Order

Actors : พนักงานธุรการประจำคลังวัสดุกลาง (Main Store Administrator)

Description : Main Store Administrator ทำการบันทึกข้อมูลใบสั่งซื้อเพื่อเก็บข้อมูลสำหรับเตรียมที่จะรับสินค้าตามใบสั่งซื้อ

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ Main Store Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือก เมนู ใบสั่งซื้อ
2. System แสดงหน้าจอ ใบสั่งซื้อ
3. Main Store Administrator ทำการกรอกข้อมูลใบสั่งซื้อซึ่งประกอบด้วย :--เลขที่ใบสั่งซื้อ วันที่สั่งซื้อ วันที่จะรับของ และ รหัสผู้จำหน่าย
4. จากนั้น ทำการใส่รายการอุปกรณ์ที่สั่งซื้อ ซึ่งทำได้โดยการกดปุ่ม ค้นหาอุปกรณ์ ซึ่งจะเป็นการทำงานในส่วนของ **Use Case -Search Item Type** ต่อไป
5. System แสดงหน้าจอ ใบสั่งซื้ออีกครั้ง และจะปรากฏรายชื่ออุปกรณ์ที่ได้เลือกไว้
6. Main Store Administrator ใส่จำนวนที่สั่งซื้อตามที่ได้สั่งซื้อจริง และกดปุ่มเพิ่มรายการอุปกรณ์
7. System ทำการเพิ่มรายการอุปกรณ์ เข้าไปในส่วนของรายการอุปกรณ์ใบสั่งซื้อ
8. เมื่อ Main Store Administrator ต้องการใส่รายการอุปกรณ์ต่อไป ให้ทำตามขั้นตอนที่ 4-7
9. เมื่อ Main Store Administrator ต้องการ Save ใบสั่งซื้อนี้ ให้กดปุ่ม Submit Form
10. System จะบันทึกข้อมูลใบสั่งซื้อนี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบสั่งซื้อนี้จะถูกเพิ่มเป็นยอดอุปกรณ์ระหว่างการสั่งซื้อของ Main Store

5.3.1.2 Use Case - Perform Goods Receipt

Actors : พนักงานธุรการประจำคลังวัสดุกลาง (Main Store Administrator)

Description : Main Store Administrator ทำการรับอุปกรณ์จากการสั่งซื้อจาก Supplier

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ พนักงานธุรการประจำคลังวัสดุกลางทำการ Login เข้าสู่ระบบ และทำการเลือก เมนู ใบรับอุปกรณ์จาก Supplier
2. System แสดงหน้าจอรายการใบสั่งซื้อที่ยังได้รับของ หรือได้รับยังไม่ครบตามจำนวนในใบสั่งซื้อ
3. Main Store Administrator ทำการเลือกใบสั่งซื้อที่ต้องการ และ กดปุ่ม รับอุปกรณ์
4. System แสดงหน้าจอใบรับอุปกรณ์ ซึ่งจะแสดงรายละเอียดของใบสั่งซื้อที่ถูกเลือก และในส่วนของรายการอุปกรณ์ในใบรับอุปกรณ์ จะมีช่อง จำนวนที่รับ ให้ Main Store Administrator ทำการกรอก
5. Main Store Administrator ทำการกรอกข้อมูลใบรับอุปกรณ์ ซึ่งประกอบด้วย :- เลขที่ใบแจ้งหนี้ วันที่รับ และทำการใส่จำนวนที่รับจริง ในแต่ละรายการอุปกรณ์ในใบรับอุปกรณ์
6. เมื่อ Main Store Administrator ต้องการ Save ใบรับอุปกรณ์นี้ ให้กดปุ่ม Submit Form
7. System จะบันทึกข้อมูลใบรับอุปกรณ์นี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบรับของนี้จะถูกเพิ่มเป็นยอดอุปกรณ์ดีของ Main Store และ ยอดอุปกรณ์ระหว่างการสั่งซื้อของ Main Store ก็จะถูกลดลง

5.3.2 ระบบการเบิกจ่ายอุปกรณ์ (Requisition)

5.3.2.1 Use Case - Perform Requisition

Actors : พนักงานธุรการประจำศูนย์ซ่อมบำรุงย่อย (MC Administrator)

Description : MC Administrator ทำการเบิกอุปกรณ์จาก Main Store มาเป็นวัสดุอะไหล่คงคลังไว้ที่ Maintenance Center

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ MC Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือกเมนู เบิกอุปกรณ์
2. System แสดงหน้าจอ ใบเบิกอุปกรณ์
3. MC Administrator ทำการกรอกข้อมูลใบเบิกอุปกรณ์ ซึ่งประกอบด้วย :- วันที่เบิก วิธีการขนส่ง สถานที่รับของ(สถานที่ที่ผู้เบิกจะไปรับของจากการขนส่ง) ผู้รับอุปกรณ์ (ชื่อของบุคคลที่จะมารับ หรือ ชื่อของ Supplier ที่จะมารับ)
4. จากนั้น ทำการใส่รายการอุปกรณ์ที่จะทำการเบิก ซึ่งทำได้โดยการกดปุ่ม ค้นหาอุปกรณ์ ซึ่งจะเป็นการทำงานในส่วนของ **Use Case-Search Item Type** ต่อไป
5. System แสดงหน้าจอ ใบเบิกอุปกรณ์อีกครั้ง และจะปรากฏรายชื่ออุปกรณ์ที่ได้เลือกไว้ พร้อมกับจำนวนที่สามารถทำการเบิกได้
6. MC Administrator ใส่จำนวนที่ต้องการเบิก และกดปุ่มเพิ่มรายการอุปกรณ์
7. System จะทำการเพิ่มรายการอุปกรณ์ เข้าไปในส่วนของรายการอุปกรณ์ของใบเบิกอุปกรณ์
8. เมื่อ MC Administrator ต้องการใส่รายการอุปกรณ์ต่อไป ให้ทำตามขั้นตอนที่ 4-7
9. เมื่อ MC Administrator ต้องการ Save ใบเบิกอุปกรณ์นี้ ให้กดปุ่ม Submit Form
10. System จะบันทึกข้อมูลใบเบิกอุปกรณ์นี้เข้าสู่ระบบ

5.3.2.2 Use Case - Approve Requisition

Actors : ผู้จัดการส่วนงานวิศวกรรมประจำภูมิภาค (Regional Manager)

Description : Regional Manager ทำการอนุมัติใบเบิกอุปกรณ์ระหว่าง Main Store กับ Maintenance Center

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ Regional Manager ทำการLogin เข้าสู่ระบบ และทำการเลือกเมนู อนุมัติการเบิก
2. System แสดงหน้าจอรายการใบเบิกอุปกรณ์ ที่ยังไม่ได้ทำใบส่งของทุกใบ รวมถึงใบที่อนุมัติแล้ว หรือยังไม่อนุมัติ เพื่อสามารถแก้ไขได้
3. Regional Manager เลือกใบเบิกอุปกรณ์ที่ต้องการอนุมัติ
4. System แสดงหน้าจอรายละเอียดของใบเบิกอุปกรณ์ที่ถูกเลือก
5. Regional Manager ทำตัดสินใจว่าจะอนุมัติหรือไม่อนุมัติ โดยเลือกจากตัวเลือก อนุมัติ / ไม่อนุมัติ หากเลือก ไม่อนุมัติ จะต้องใส่สาเหตุที่ไม่อนุมัติด้วย จากนั้นกดปุ่ม Save เพื่อทำการบันทึกการอนุมัติ หรือ ไม่อนุมัตินี้
6. System บันทึกสถานะการอนุมัติ หรือ ไม่อนุมัติ ของใบเบิกอุปกรณ์นี้ ซึ่งหาก ใบเบิกอุปกรณ์นี้ได้รับการอนุมัติ System จะนำยอดอุปกรณ์แต่ละรายการในใบเบิกไปทำการเพิ่มเป็นยอดอุปกรณ์ถูกเบิก ของ Main Store และ นำไปลดออกจากยอดอุปกรณ์ดี ของ Main Store

5.3.2.3 Use Case - Perform Delivery

Actors : พนักงานธุรการประจำคลังวัสดุกลาง (Main Store Administrator)

Description : Main Store Administrator ทำการส่งอุปกรณ์ตามใบเบิก

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ Main Store Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือก เมนู ส่งของ
2. System แสดงหน้ารายการใบเบิกอุปกรณ์ที่ผ่านการอนุมัติจาก Regional Manager แล้ว
3. Main Store Administrator ทำการเลือกใบเบิกอุปกรณ์ที่ต้องการจะส่งของให้ และ กดปุ่ม ทำใบส่งของ
4. System แสดงหน้าจอใบส่งของ ซึ่งจะแสดงรายละเอียดของใบเบิกอุปกรณ์ที่เลือก และในส่วนของรายการอุปกรณ์ของใบส่งของ จะมีช่องจำนวนที่ส่ง ให้ Main Store Administrator ทำการกรอก
5. Main Store Administrator ใส่จำนวนที่ส่ง ในแต่ละรายการอุปกรณ์ของใบส่ง หากรายการใดยังไม่สามารถส่งให้ได้ในใบส่งนี้ ก็ให้เว้นว่างไว้
6. เมื่อ Main Store Administrator ต้องการ Save ใบส่งของนี้ ให้กดปุ่ม Submit Form
7. System บันทึกข้อมูลใบส่งของนี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบส่งของ จะถูกนำไปเพิ่มเป็นยอดอุปกรณ์ระหว่างการขนส่ง ของ Maintenance Center ที่ทำการเบิก และ ถูกนำไปลดออกจากยอดอุปกรณ์ถูกเบิกของ Main Store

5.3.2.4 Use Case - Perform Delivery Receipt

Actors : พนักงานธุรการประจำศูนย์ซ่อมบำรุงย่อย (MC Administrator)

Description : MC Administrator ทำการรับอุปกรณ์จากการเบิกมาจาก Main Store เพื่อทำให้ยอดของจำนวนอุปกรณ์ภายใน Maintenance Center เพิ่มขึ้น เพื่อสามารถเบิกไปใช้งานภายในได้

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ MC Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือกเมนู รับของ
2. System แสดงหน้ารายการใบส่งของตามใบเบิกที่ทาง Maintenance Center ได้ทำการเบิกไป
3. MC Administrator ทำการเลือกใบส่งของที่ต้องการจะทำใบรับของ และ กดปุ่ม ทำใบรับของ
4. System แสดงหน้าจอใบรับของ ซึ่งจะแสดงรายละเอียดของใบส่งของที่เลือก และ ในส่วนของรายการอุปกรณ์ของใบรับของ จะมีช่องจำนวนที่รับ ให้ MC Administrator ทำการกรอก
5. MC Administrator ใส่จำนวนที่รับจริง ในแต่ละรายการอุปกรณ์ของใบรับของ หากรายการใดยังไม่ได้รับ ก็ให้เว้นว่างไว้
6. เมื่อ MC Administrator ต้องการ Save ใบรับของนี้ ให้กดปุ่ม Submit Form
7. System บันทึกข้อมูลใบรับของนี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบรับของนี้ จะถูกนำไปเพิ่มเป็นยอดอุปกรณ์ดีของ Maintenance Center ที่ทำการรับ และ ถูกนำไปลดออกจากยอดอุปกรณ์ระหว่างการขนส่งของ Maintenance Center ที่ทำการรับเช่นกัน

5.3.3 ระบบการทำงานภายใน (Internal MC)

5.3.3.1 Use Case - Perform Internal Requisition

Actors : วิศวกรซ่อมบำรุง (Field Maintenance Engineer)

Description : Field Maintenance Engineer ทำการเบิกอุปกรณ์ภายใน Maintenance Center เพื่อนำไปซ่อมบำรุงสถานีฐาน

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ Field Maintenance Engineer ทำการ Login เข้าสู่ระบบ และทำการเลือกเมนู เบิกอุปกรณ์ภายใน
2. System แสดงหน้าจอ ใบเบิกอุปกรณ์ภายใน
3. Field Maintenance Engineer ทำการกรอกข้อมูลวันที่ทำการเบิก
4. จากนั้น ทำการใส่รายการอุปกรณ์ที่จะทำการเบิก ซึ่งทำได้โดยการกดปุ่ม ค้นหาอุปกรณ์ ซึ่งจะเป็นการทำงานในส่วนของ **Use Case-Search Item Type** ต่อไป
5. System แสดงหน้าจอ ใบเบิกอุปกรณ์ภายในอีกครั้ง และจะปรากฏรายชื่ออุปกรณ์ที่ได้เลือกไว้พร้อมกับจำนวนที่สามารถทำการเบิกได้
6. Field Maintenance Engineer ใส่จำนวนที่ต้องการเบิก รหัสสถานีฐานที่จะนำไปซ่อมบำรุง ซึ่งรหัสสถานีฐานนี้จะถูกดึงมาจากระบบ Site Information ซึ่งเป็นระบบภายนอก และกดปุ่มเพิ่มรายการอุปกรณ์
7. System จะทำการเพิ่มรายการอุปกรณ์ เข้าไปในส่วนของรายการอุปกรณ์ของใบเบิกอุปกรณ์ภายใน
8. เมื่อ Field Maintenance Engineer ต้องการใส่รายการอุปกรณ์ต่อไป ให้ทำตามขั้นตอนที่ 4-7
9. เมื่อ Field Maintenance Engineer ต้องการ Save ใบเบิกอุปกรณ์ภายในนี้ ให้กดปุ่ม Submit Form
10. System จะบันทึกข้อมูลใบเบิกอุปกรณ์ภายในนี้เข้าสู่ระบบ

5.3.3.2 Use Case - Approve Internal Requisition

Actors : ผู้จัดการศูนย์ซ่อมบำรุงย่อย (MC Manager)

Description : MC Manager ทำการอนุมัติใบเบิกอุปกรณ์ภายใน

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ MC Manager ทำการLogin เข้าสู่ระบบ และทำการเลือกเมนูอนุมัติการเบิกภายใน
2. System แสดงหน้าจอรายการใบเบิกอุปกรณ์ภายใน ที่ยังไม่ได้ทำใบส่งของทุกใบ รวมถึงใบที่อนุมัติแล้ว หรือยังไม่อนุมัติ เพื่อสามารถแก้ไขได้
3. MC Manager เลือกใบเบิกอุปกรณ์ภายในที่ต้องการอนุมัติ
4. System แสดงหน้าจอรายละเอียดของใบเบิกอุปกรณ์ภายในที่ถูกเลือก
5. MC Manager ทำตัดสินใจว่าจะอนุมัติหรือไม่อนุมัติ โดยเลือกจากตัวเลือก อนุมัติ / ไม่อนุมัติ หากเลือก ไม่อนุมัติ จะต้องใส่สาเหตุที่ไม่อนุมัติด้วย จากนั้นกดปุ่ม Save เพื่อทำการบันทึกการอนุมัติ หรือ ไม่อนุมัตินี้
6. System บันทึกสถานะการอนุมัติ หรือ ไม่อนุมัติ ของใบเบิกอุปกรณ์ภายในนี้ ซึ่งหากใบเบิกอุปกรณ์นี้ได้รับการอนุมัติ System จะนำยอดอุปกรณ์แต่ละรายการในใบเบิกไปทำการเพิ่มเป็นยอดอุปกรณ์ถูกเบิก ของ Maintenance Center ที่ถูกเบิก และ นำไปลดออกจากยอดอุปกรณ์ดี ของ Maintenance Center ที่ถูกเบิก

5.3.3.3 Use Case - Perform Internal Delivery

Actors : พนักงานธุรการประจำศูนย์ซ่อมบำรุงย่อย (MC Administrator)

Description : MC Administrator ทำการส่งอุปกรณ์ตามใบเบิกอุปกรณ์ภายใน

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ MC Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือกเมนู ส่งของภายใน
2. System แสดงหน้ารายการใบเบิกอุปกรณ์ภายในที่ผ่านการอนุมัติจาก MC Manager แล้ว
3. MC Administrator ทำการเลือกใบเบิกอุปกรณ์ที่ต้องการจะส่งของให้ และ กดปุ่ม ทำใบส่งของภายใน
4. System แสดงหน้าจอใบส่งของภายใน ซึ่งจะแสดงรายละเอียดของใบเบิกอุปกรณ์ภายในที่เลือก และ ในส่วนของรายการอุปกรณ์ของใบส่งของภายใน จะมีช่องจำนวนที่ส่งให้ MC Administrator ทำการกรอก
5. MC Administrator ใส่จำนวนที่ส่ง ในแต่ละรายการอุปกรณ์ของใบส่งของภายใน หากรายการใดยังไม่สามารถส่งให้ได้ในใบส่งนี้ ก็ให้เว้นว่างไว้
6. เมื่อ MC Administrator ต้องการ Save ใบส่งของภายในนี้ ให้กดปุ่ม Submit Form
7. System บันทึกข้อมูลใบส่งของภายในนี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบส่งของภายใน จะถูกนำไปเพิ่มเป็นยอดอุปกรณ์ของ สถานีฐาน ที่จะไปซ่อม และ ถูกนำไปลดออกจากยอดอุปกรณ์ถูกเบิกของ Maintenance Center

5.3.3.4 Use Case - Perform Internal Receipt

Actors : พนักงานธุรการประจำศูนย์ซ่อมบำรุงย่อย (MC Administrator)

Description : MC Administrator ทำการรับอุปกรณ์จาก Field Maintenance Engineer เข้า Store ภายใน Maintenance Center ซึ่งอาจเป็นอุปกรณ์ดี หรือ อุปกรณ์เสีย ภายหลังจากการออกไปซ่อมบำรุงสถานีฐาน

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ MC Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือกเมนู รับของภายใน
2. System แสดงหน้ารายการใบส่งของภายในที่ เคยถูกทำไว้ และ ยังรับของเข้าไม่ครบ
3. MC Administrator ทำการเลือกใบส่งของภายในที่ต้องการจะทำใบรับของภายใน และกดปุ่ม ทำใบรับของภายใน
4. System จะแสดงหน้าจอใบรับของภายใน ซึ่งจะแสดงรายละเอียดของใบส่งของภายในที่เลือก และ ในส่วนของรายการอุปกรณ์ของใบรับของ จะมีช่องจำนวนที่รับซึ่งจะแบ่งออกเป็นของดี และ ของเสีย ให้ MC Administrator ทำการกรอก
5. MC Administrator ใส่จำนวนที่รับจริง ตามสถานะคือ ดี หรือ เสีย ในแต่ละรายการอุปกรณ์ของใบรับของภายใน หากรายการใดยังไม่ได้รับ ก็ให้เว้นว่างไว้
6. เมื่อ MC Administrator ต้องการ Save ใบรับของนี้ ให้กดปุ่ม Submit Form
7. System บันทึกข้อมูลใบรับของนี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบรับของนี้ จะถูกนำไปเพิ่มเป็นยอดอุปกรณ์ดีและเสียของ Maintenance Center ที่ทำการรับ และ ถูกนำไปลดออกจากยอดอุปกรณ์ของ สถานีฐาน ที่ได้ไปซ่อมมา

5.3.4 ระบบการส่งซ่อม (Repair)

5.3.4.1 Use Case - Perform Repair Request

Actors : พนักงานธุรการประจำศูนย์ซ่อมบำรุงย่อย (MC Administrator)

Description : MC Administrator ทำการส่งอุปกรณ์เสีย ให้กับ Main Store เพื่อให้ Main Store นำไปส่งซ่อมต่อไป

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ MC Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือกเมนู ส่งซ่อมให้ Main Store
2. System แสดงหน้าจอ ใบส่งซ่อม Main Store
3. MC Administrator ทำการใส่วันที่ส่งซ่อม ในใบส่งซ่อม Main Store
4. จากนั้น ทำการใส่รายการอุปกรณ์ที่จะทำการส่งซ่อม ซึ่งทำได้โดยการกดปุ่ม ค้นหาอุปกรณ์ ซึ่งจะเป็นการทำงานในส่วนของ **Use Case-Search Item Type** ต่อไป
5. System จะแสดงหน้าจอ ใบเบิกอุปกรณ์อีกครั้ง และจะปรากฏรายชื่ออุปกรณ์ที่ได้เลือกไว้พร้อมกับจำนวนที่สามารถทำการส่งซ่อมได้
6. MC Administrator ใส่จำนวนที่ต้องการส่งซ่อม และกดปุ่มเพิ่มรายการอุปกรณ์
7. System จะทำการเพิ่มรายการอุปกรณ์ เข้าไปในส่วนของรายการอุปกรณ์ของใบส่งซ่อม Main Store
8. เมื่อ MC Administrator ต้องการใส่รายการอุปกรณ์ต่อไป ให้ทำตามขั้นตอนที่ 4-7
9. เมื่อ MC Administrator ต้องการ Save ใบส่งซ่อม Main Store นี้ ให้กดปุ่ม Submit Form
10. System บันทึกข้อมูลใบส่งซ่อม Main Store นี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบส่งซ่อม Main Store นี้ จะถูกนำไปเพิ่มเป็นยอดอุปกรณ์ส่งซ่อมของ Maintenance Center ที่ส่งซ่อม และ ถูกนำไปลดออกจากยอดอุปกรณ์เสียของ Maintenance Center ที่ส่งซ่อม

5.3.4.2 Use Case - Perform Repair Receipt

Actors : พนักงานธุรการประจำคลังวัสดุกลาง (Main Store Administrator)

Description : Main Store Administrator ทำการรับอุปกรณ์ เสียที่ Maintenance Center ส่งซ่อมให้ Main Store

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ Main Store Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือก เมนู รับซ่อมจาก MC
2. System แสดงหน้ารายการใบส่งซ่อม Main Store ที่ถูกทำส่งมา
3. Main Store Administrator ทำการเลือกใบส่งซ่อม Main Store ที่ต้องการจะรับซ่อม และ กดปุ่ม ทำใบรับซ่อมจากMC
4. System แสดงหน้าจอใบรับซ่อมจาก MC ซึ่งจะแสดงรายละเอียดของใบส่งซ่อม Main Store ที่เลือก และ ในส่วนของรายการอุปกรณ์ของใบรับซ่อมจาก MC จะมีช่องจำนวนที่รับ ให้ Main Store Administrator ทำการกรอก
5. Main Store Administrator ใส่จำนวนที่รับ ในแต่ละรายการอุปกรณ์ของใบรับซ่อมจาก MC หากรายการใดยังไม่สามารถรับได้ในใบรับซ่อมนี้ ก็ให้เว้นว่างไว้
6. เมื่อ Main Store Administrator ต้องการ Save ใบรับซ่อมจาก MC นี้ ให้กดปุ่ม Submit Form
7. System บันทึกข้อมูลใบรับซ่อมจาก MC นี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบรับซ่อมจาก MC นี้ จะถูกนำไปเพิ่มเป็นยอดอุปกรณ์เสีย ของ Main Store และถูกนำไปลดออกจากยอดอุปกรณ์ส่งซ่อมของ Maintenance Center ที่ส่งซ่อม

5.3.4.3 Use Case - Perform Supplier Repair Request

Actors : พนักงานธุรการประจำคลังวัสดุกลาง (Main Store Administrator)

Description : Main Store Administrator ทำการส่งอุปกรณ์เสีย ให้กับ Supplier เพื่อนำไปซ่อม หรือ เปลี่ยน กลับมา

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ Main Store Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือกเมนู ส่งซ่อมให้ Supplier
2. System แสดงหน้าจอ ใบส่งซ่อม Supplier
3. Main Store Administrator ทำการใส่วันที่ส่งซ่อม และ รหัสผู้จำหน่ายในใบส่งซ่อม Supplier
4. จากนั้น ทำการใส่รายการอุปกรณ์ที่จะทำการส่งซ่อม ซึ่งทำได้โดยการกดปุ่ม ค้นหาอุปกรณ์ ซึ่งจะเป็นการทำงานในส่วนของ **Use Case-Search Item Type** ต่อไป
5. System แสดงหน้าจอ ใบส่งซ่อม Supplier อีกครั้ง และจะปรากฏรายชื่ออุปกรณ์ที่ได้เลือกไว้พร้อมกับจำนวนที่สามารถทำการส่งซ่อมได้
6. Main Store Administrator ใส่จำนวนที่ต้องการส่งซ่อม และกดปุ่มเพิ่มรายการอุปกรณ์
7. System ทำการเพิ่มรายการอุปกรณ์ เข้าไปในส่วนของรายการอุปกรณ์ของใบส่งซ่อม Supplier
8. เมื่อ Main Store Administrator ต้องการใส่รายการอุปกรณ์ต่อไป ให้ทำตามขั้นตอนที่ 4-11
9. เมื่อ Main Store Administrator ต้องการ Save ใบส่งซ่อม Supplier นี้ ให้กดปุ่ม Submit Form
10. System บันทึกข้อมูลใบส่งซ่อม Supplier นี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบส่งซ่อม Supplier นี้ จะถูกนำไปเพิ่มเป็นยอดอุปกรณ์ส่งซ่อมของ Main Store และถูกนำไปลดออกจากยอดอุปกรณ์เสียของ Main Store

5.3.4.4 Use Case - Perform Supplier Repair Receipt

Actors : พนักงานธุรการประจำคลังวัสดุกลาง (Main Store Administrator)

Description : Main Store Administrator ทำการรับอุปกรณ์ที่ถูกซ่อมหรือเปลี่ยนมาแล้วจาก Supplier

Basic Flow :

1. Use Case นี้เริ่มต้นเมื่อ Main Store Administrator ทำการ Login เข้าสู่ระบบ และทำการเลือก เมนู รับซ่อมจาก Supplier
2. System แสดงหน้ารายการใบส่งซ่อม Supplier ที่เคยถูกทำส่งไป
3. Main Store Administrator ทำการเลือกใบส่งซ่อม Supplier ที่ต้องการจะรับ และ กดปุ่มทำใบรับซ่อมจาก Supplier
4. System แสดงหน้าจอใบรับซ่อมจาก Supplier ซึ่งจะแสดงรายละเอียดของใบส่งซ่อม Supplier ที่เลือก และ ในส่วนของรายการอุปกรณ์ของใบรับซ่อมจาก Supplier จะมีช่องจำนวนที่รับ ให้ Main Store Administrator ทำการกรอก
5. Main Store Administrator ใส่จำนวนที่รับ ในแต่ละรายการอุปกรณ์ของใบรับซ่อมจาก Supplier หากรายการใดยังไม่สามารถรับได้ในใบรับซ่อมนี้ ก็ให้เว้นว่างไว้
6. เมื่อ Main Store Administrator ต้องการ Save ใบรับซ่อมจาก Supplier นี้ ให้กดปุ่ม Submit Form
7. System บันทึกข้อมูลใบรับซ่อมจาก MC นี้เข้าสู่ระบบ โดยที่ยอดอุปกรณ์แต่ละรายการในใบรับซ่อมจาก Supplier นี้ จะถูกนำไปเพิ่มเป็นยอดอุปกรณ์ดี ของ Main Store และถูกนำไปลดออกจากยอดอุปกรณ์ส่งซ่อมของ Main Store

5.3.5 ระบบความช่วยเหลือต่างๆ (Helper)

5.3.5.1 Search Item Type

Actors : Use Case Entry Purchase Order Use Case Perform Requisition Use Case Perform Internal Requisition Use Case Perform Repair Request และ Use Case Perform Supplier Repair Request

Description : การทำใบสั่งซื้อ การทำใบเบิกอุปกรณ์ การทำใบเบิกอุปกรณ์ภายใน การทำใบส่งซ่อม Main Store และ การทำใบส่งซ่อม Supplier จะใช้การค้นหาอุปกรณ์เพื่อช่วยในการทำรายการเกี่ยวกับอุปกรณ์

Basic Flow :

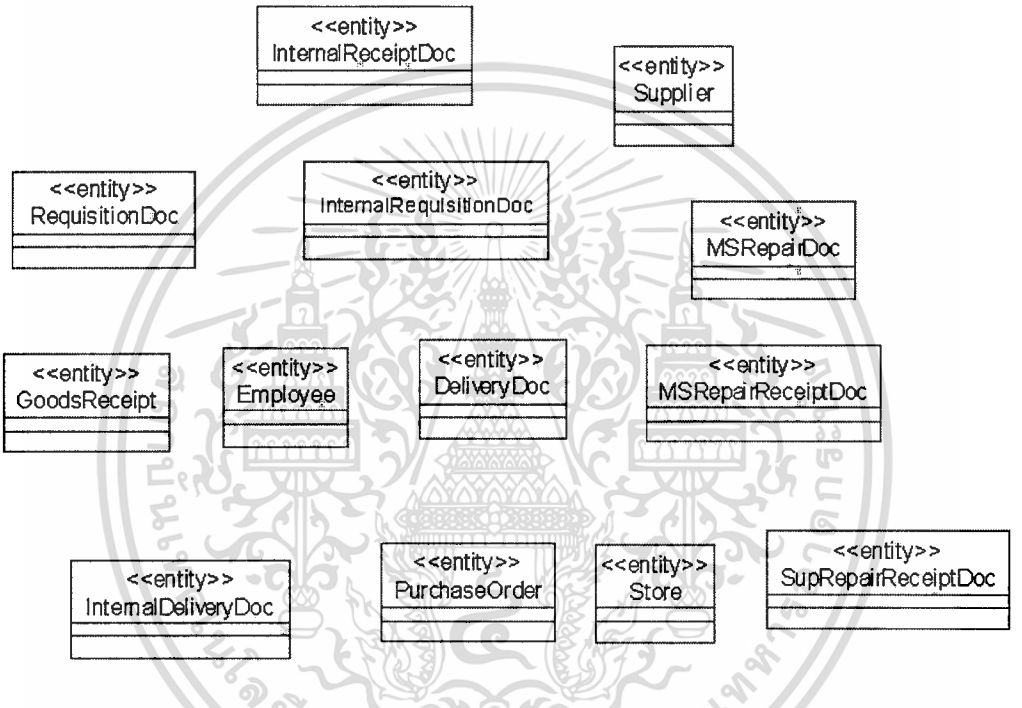
1. Use Case นี้เริ่มต้นเมื่อ ผู้ใช้ ทำการกดปุ่ม ค้นหาอุปกรณ์ ซึ่งสามารถกดปุ่มนี้มาได้จากหน้าจอ การทำใบสั่งซื้อ การทำใบเบิกอุปกรณ์ การทำใบเบิกอุปกรณ์ภายใน และ การทำใบส่งซ่อม
2. System แสดงหน้าจอ ค้นหาอุปกรณ์ ซึ่งจะประกอบด้วย รายละเอียดของอุปกรณ์ที่สามารถใช้ในการค้นหาได้แก่ รหัสอุปกรณ์ และ คำอธิบายอุปกรณ์
3. ผู้ใช้ทำการใส่ข้อมูลที่ต้องการใช้ในการค้นหา จากนั้น กดปุ่ม Submit
4. System จะแสดงรายการอุปกรณ์ที่มีรหัสอุปกรณ์หรือ คำอธิบาย ตรงตามเงื่อนไขที่ค้นหา ซึ่งอุปกรณ์แต่ละรายการจะแสดงข้อมูล รหัสอุปกรณ์ คำอธิบาย และ หน่วยนับ โดยจะทำการค้นหาจากแหล่งที่แตกต่างกันดังนี้
   - ถ้า มาจากหน้าจอทำใบสั่งซื้อ System จะทำการค้นหารายชื่ออุปกรณ์มาจากบัญชีรายชื่ออุปกรณ์
   - ถ้า มาจากหน้าจอทำใบเบิกอุปกรณ์ System จะทำการค้นหารายชื่ออุปกรณ์ จากรายชื่ออุปกรณ์ในMain Store ส่วนของอุปกรณ์ดี ที่ยอดคงเหลือมากกว่าศูนย์
   - ถ้า มาจากหน้าจอทำใบเบิกอุปกรณ์ภายใน System จะทำการค้นหารายชื่ออุปกรณ์ จาก รายชื่ออุปกรณ์ในMaintenance Center ของผู้เบิกในส่วนของอุปกรณ์ดี ซึ่งมียอดคงเหลือมากกว่าศูนย์
   - ถ้า มาจากหน้าจอทำใบส่งซ่อม Main Store System จะทำการค้นหารายชื่ออุปกรณ์ จาก รายชื่ออุปกรณ์ในMaintenance Center ของผู้ส่งซ่อมในส่วนของอุปกรณ์เสีย ซึ่งมียอดคงเหลือมากกว่าศูนย์

- ถ้า มาจากหน้าจอทำใบส่งซ่อม Supplier System จะทำการค้นหารายชื่ออุปกรณ์จาก รายชื่ออุปกรณ์ในMain Store ในส่วนของอุปกรณ์เสีย ซึ่งมียอดคงเหลือมากกว่าศูนย์

5. ผู้ใช้ทำการเลือกรายชื่ออุปกรณ์จากรายการที่แสดง และกดปุ่ม Submit
6. System แสดงหน้าจอเดิมที่ผู้ใช้ทำการกดปุ่มค้นหาอุปกรณ์มาอีกครั้ง และจะปรากฏรายชื่ออุปกรณ์ที่ได้เลือกไว้ โดยจะแสดงรายละเอียดของอุปกรณ์ที่ผู้ใช้ทำการเลือก ในรูปแบบที่แตกต่างกันออกไปดังนี้
   - ถ้า เป็นหน้าจอทำใบสั่งซื้อ System แสดงรหัสอุปกรณ์ คำอธิบายอุปกรณ์ และช่องให้ใส่จำนวนที่สั่งซื้อ
   - ถ้า เป็นหน้าจอทำใบเบิกอุปกรณ์ System แสดงรหัสอุปกรณ์ คำอธิบายอุปกรณ์ จำนวนที่สามารถเบิกได้ของ Main Store และ ช่องให้ใส่จำนวนที่เบิก
   - ถ้า เป็นหน้าจอทำใบเบิกอุปกรณ์ภายใน System แสดงรหัสอุปกรณ์ คำอธิบายอุปกรณ์ จำนวนที่สามารถเบิกได้ของMC และ ช่องให้ใส่จำนวนที่เบิก
   - ถ้า เป็นหน้าจอทำใบส่งซ่อม Main Store System แสดงรหัสอุปกรณ์ คำอธิบายอุปกรณ์ จำนวนที่เสียทั้งหมดของ MC และ ช่องให้ใส่จำนวนที่ส่งซ่อม
   - ถ้า เป็นหน้าจอทำใบส่งซ่อม Supplier System แสดงรหัสอุปกรณ์ คำอธิบายอุปกรณ์ จำนวนที่เสียทั้งหมดของ Main Store และ ช่องให้ใส่จำนวนที่ส่งซ่อม

5.4 Preliminary Analysis Classes

จากขั้นตอนของการทำ Use Case Model และ Use Case Description (Scenario) นั้น ซึ่งเป็น เอกสารความต้องการ (Requirement Document) เราจะนำ เอกสารความต้องการดังกล่าว มาทำการหา preliminary analysis classes หรือ เรียกอีกอย่างว่า Key Abstractions ซึ่งเราสามารถหาได้ดังตัวอย่างในรูปที่ 5-6



รูปที่ 5-6 ตัวอย่าง Key Abstractions ในระบบ SPCS

5.5 Architecture Analysis

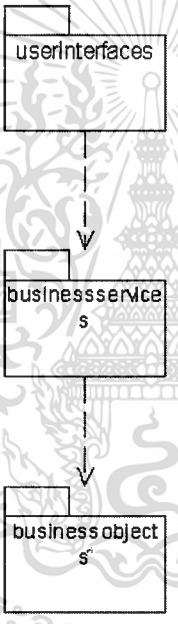
ในขั้นตอนของการวิเคราะห์ระบบ นั้น เราควรที่จะทำการวางสถาปัตยกรรมเบื้องต้น(Initial Architecture) ของระบบ เพื่อเป็นการจัดแบ่งระบบงานออกเป็นส่วนๆไว้ และเพื่อเป็นการวางแนวทางในการวิเคราะห์ระบบในขั้นตอนต่อไป ซึ่งจะต้องทำการวิเคราะห์และออกแบบตามแนวทางของสถาปัตยกรรมที่วางไว้ (Architecture Centric) โดยเราสามารถที่จะเลือกใช้สถาปัตยกรรมที่มีผู้คิดไว้อยู่แล้ว หรือที่เรียกว่า Architecture patterns มาทำการประยุกต์ใช้กับระบบงานของเราได้ โดยในระบบงานนี้ ได้นำ Architecture patterns มาใช้ ดังนี้

- **Layers** โดย Layers pattern นั้น จะเป็นลักษณะที่ ระบบงานจะถูกแบ่งเป็นส่วนๆ ใน

ชั้น(layer)ที่แตกต่างๆกัน โดยที่จะลำดับชั้นดังนี้คือ ชั้นที่มีความยึดติดกับแอปพลิเคชั่น (application-specific layers) จะอยู่ด้านบนสุด ไปจนถึง ชั้นที่มีความยึดติดกับการเทคโนโลยีที่จะในการสร้าง (implementation/technology-specific layers)

- **Model-View-Controller (M-V-C)** โดย MVC pattern นั้นจะเป็นลักษณะที่ระบบงานถูกจัดแบ่งออกเป็น 3 ส่วน คือ Model เป็นส่วนของ กฎเกณฑ์ทางธุรกิจ และ ข้อมูลที่เกี่ยวข้อง View จะเป็นส่วนที่บอกว่าข้อมูลจะถูกแสดงให้ผู้ใช้เห็นอย่างไร และ Controller จะเป็นส่วนที่ทำการประมวลผลข้อมูลที่ผู้ใช้ให้มา

โดยสถาปัตยกรรมของระบบ SPCS จะมีลักษณะดังรูปที่ 5-7

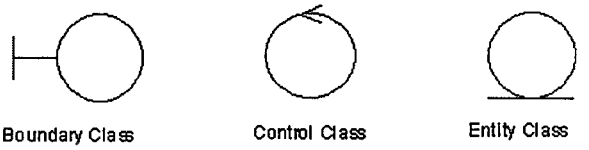


รูปที่ 5-7 Initial Architecture ของระบบ SPCS

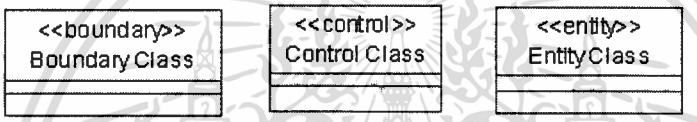
5.6 Analysis Classes

ขั้นตอนต่อมาของการวิเคราะห์ระบบ ก็จะเป็นขั้นตอนในการหา และ กำหนด คลาสในระดับของการวิเคราะห์ (analysis classes) ซึ่งเทคนิคหนึ่งที่จะใช้ในการหา analysis classes ก็คือการใช้มุมมองที่แตกต่างกัน 3 มุมมองของระบบ เป็นส่วนช่วยในการกำหนด หรือ หา class ที่เกี่ยวข้อง ซึ่ง มุมมองทั้งสาม ก็ได้แก่ ขอบเขต(boundary) ระหว่างระบบงานกับผู้ใช้ ข้อมูลที่ระบบงานจะต้องใช้ และ ตรรกะที่ใช้ในการควบคุม(control logic) ของระบบงาน โดยมุมมองทั้งสามนี้ จะถูกกำหนด

ให้เป็นประเภทของ class ของ analysis classes โดยกำหนดให้เป็น stereotype ของ class ซึ่งสามารถแสดงโดยใช้สัญลักษณ์เป็น icon ดังรูปที่ 5-8 หรือ แสดงเป็น label ดังรูปที่ 5-9



รูปที่ 5-8 Boundary class Control class และ Entity class แสดงแบบ icon



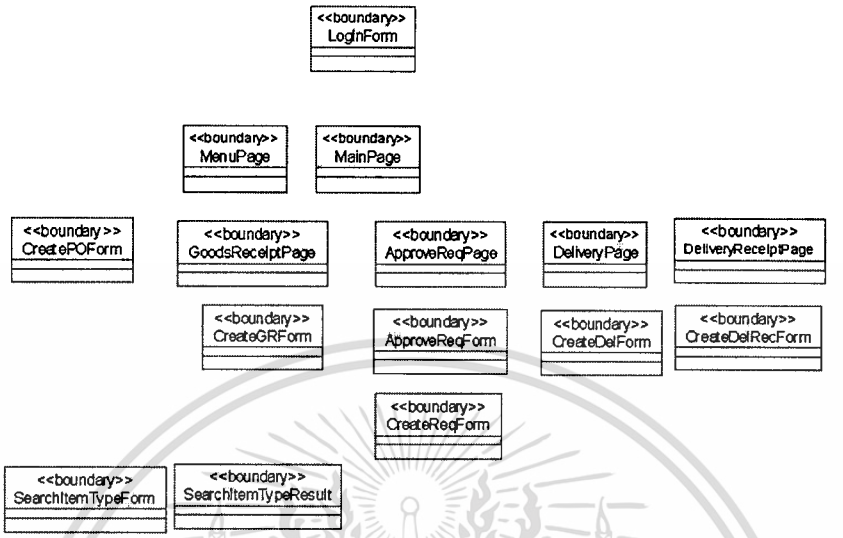
รูปที่ 5-9 Boundary class Control class และ Entity class แสดงแบบ label

5.6.1 Boundary Class

โดยทั่วไปนั้น ในระบบงานหนึ่งๆ อาจจะมี boundary class ได้หลายประเภทดังนี้

- User interface classes คือ class ซึ่งทำหน้าที่ติดต่อสื่อสารกับคนผู้ใช้ระบบงาน
- System interface classes คือ class ซึ่งทำหน้าที่ติดต่อสื่อสารกับระบบงานภายนอก
- Device interface classes คือ class ซึ่งทำหน้าที่ในการเชื่อมโยงกับอุปกรณ์ภายนอก

ในการเริ่มต้นในการกำหนด หรือ หา boundary class นั้น เราสามารถที่จะกำหนดให้ มี หนึ่ง boundary class ต่อหนึ่ง คู่ของ actor และ use-case ซึ่งรูปที่ 5-10 จะแสดงตัวอย่าง boundary classes ที่หาได้ในระบบ SPCS

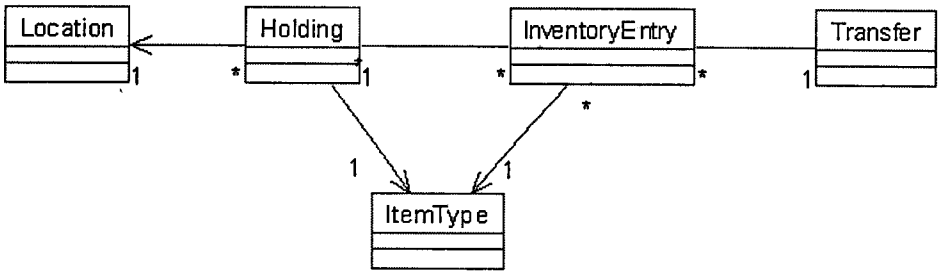


รูปที่ 5-10 ตัวอย่าง boundary classes ในระบบ SPCS

5.6.2Entity Classes

Entity classes จะเป็นการนำเสนอโครงสร้างของข้อมูลในลักษณะ logical ซึ่ง จะเป็นตัวแทนของข้อมูลที่จะถูกจัดเก็บในระบบ และหน้าที่ความรับผิดชอบหลักของ entity classes ก็คือเพื่อจัดเก็บและจัดการข้อมูลในระบบ

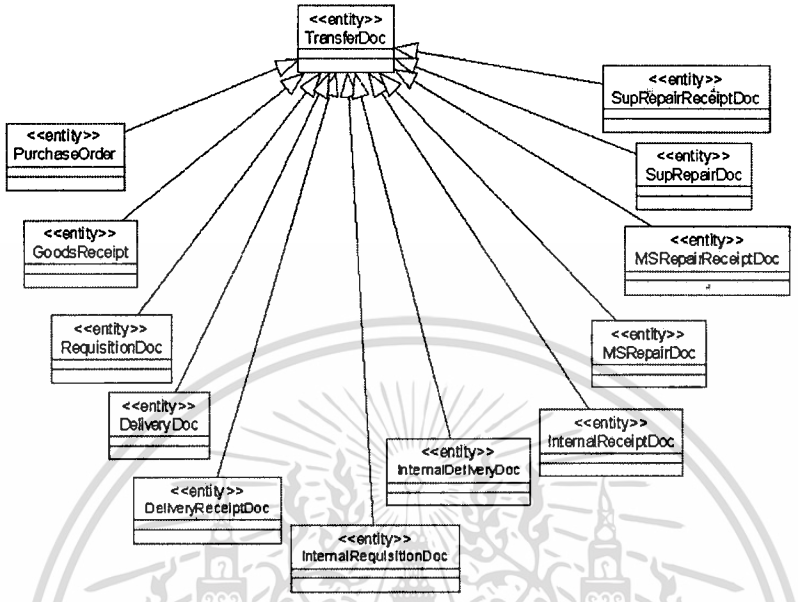
ในการกำหนดหรือ หา entity classes นั้น ส่วนหนึ่งเราสามารถนำมาจาก Key Abstractions ที่เราได้ทำการหาเอาไว้ในขั้นตอนก่อนหน้านี้ และ หาเพิ่มเติมจาก Scenario หรือ Business-Domain Model ต่างๆที่มีคนทำไว้อยู่แล้ว ที่เรียกว่า Analysis Patterns ก็ได้ ซึ่งในระบบงาน SPCS นี้ได้มีการนำ Analysis patterns หรือ Business-Domain Model ของระบบงาน Inventory มาใช้ โดยนำมาประยุกต์กับระบบงานที่ทำอยู่ ซึ่งทำให้สามารถกำหนด และ หา entity classes ได้ง่ายและเร็วขึ้น โดยรูปที่ 5-11 แสดง Business-Domain Model ของระบบงาน Inventory และรูปที่ 5-12 แสดง entity classes ที่หาได้เพิ่มเติมจาก scenario



รูปที่ 5-11 Business-Domain model ของระบบงาน Inventory

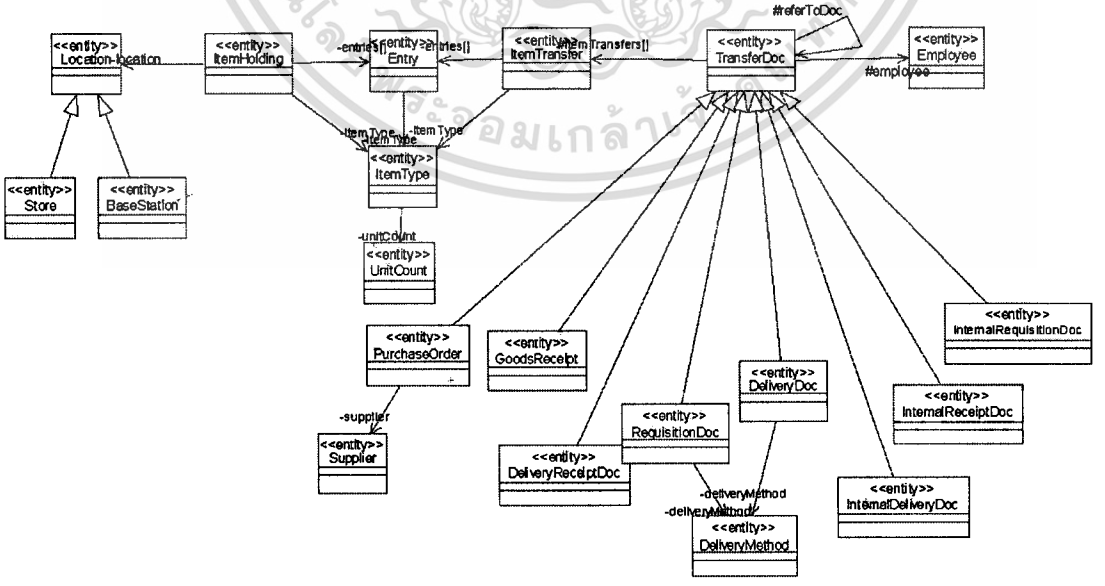
อีกขั้นตอนหนึ่งในการกำหนด entity classes ก็คือ จะต้องทำการหาความสัมพันธ์ระหว่าง entity class เหล่านี้ในรูปแบบของ Generalization ซึ่งเมื่อ generalization ถูกพบ จะต้องมีการสร้าง super-class ขึ้นมาเพื่อจัดเก็บ attributes associations aggregations และ operations ที่ใช้ร่วมกันเอาไว้ แล้วพฤติกรรมที่คล้ายกันหรือใช้ร่วมกันดังกล่าวก็จะถูกลบออกจาก classes ซึ่งกลายเป็น sub-class ของ super-class นั้นด้วย

ในระบบ SPCS นี้มีลักษณะของ generalization เกิดขึ้น เช่น ใบสั่งซื้อ ใบรับของ ใบเบิกอุปกรณ์ ฯลฯ ต่างเป็นclassของ เอกสารที่ใช้ในการบันทึกข้อมูลรายละเอียดของอุปกรณ์ที่จะต้องถูกสร้างขึ้น เมื่อมีการเคลื่อนย้ายอุปกรณ์จากแหล่งที่อยู่หนึ่งไป อีกแหล่งที่อยู่หนึ่งของอุปกรณ์เหมือนๆกัน เพียงแต่เป็นเอกสารต่างประเภท ต่างชนิดกัน เราจึงสามารถกำหนด super-class ให้กับ class เหล่านี้ได้ นั่นก็คือ class ที่ชื่อว่า TransferDoc โดยสามารถเขียนความสัมพันธ์ระหว่าง class เหล่านี้ได้ดังรูปที่ 5-12



รูปที่ 5-12 ตัวอย่างความสัมพันธ์แบบ Generalization ของ entity class ในระบบSPCS

เมื่อเราสามารถกำหนด entity classes เพิ่มเติมจาก Business-Domain Model ได้แล้ว เราก็จะนำ entity classes จากทั้งสองแหล่ง มาผสมผสานกัน ได้ดังรูปที่ 5-13



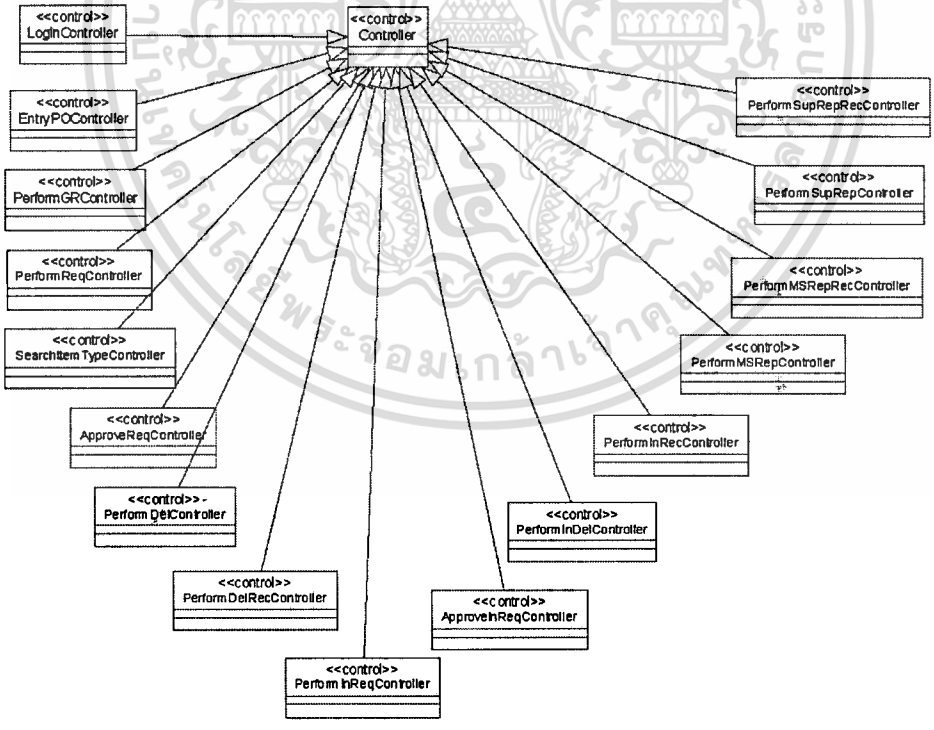
รูปที่ 5-13 ตัวอย่าง entity classes ของระบบ SPCS

5.6.3Control Classes

Control classes จะเป็นส่วนที่จัดการเกี่ยวกับการประสานพฤติกรรมต่างๆที่เกิดขึ้นในระบบ โดยที่ control classes จะทำหน้าที่ในการแยก boundary objects ออกจาก หรือ ไม่ให้ขึ้นกับ entity objects ซึ่งโดยทั่วไป เรามักจะกำหนดให้ มี หนึ่ง control class ต่อ หนึ่ง use case

Control class หนึ่งๆ นั้น จะถูกใช้ในการจำลองพฤติกรรมการควบคุมกระบวนการทำงานซึ่งเฉพาะเจาะจงไปกับ use case ใด use case หนึ่ง ซึ่งก็หมายความว่า control classes จะทำการห่อหุ้มรายละเอียดพฤติกรรมการทำงานของ use case เอาไว้

ในระบบ SPCS นี้ ได้มีการกำหนดให้มี หนึ่ง control class ต่อ หนึ่ง use case ซึ่ง control class เหล่านี้ จะให้มีคำลงท้ายด้วย Controller โดยได้มีการกำหนดให้มี super-class ของ control class เหล่านี้เอาไว้ด้วย ซึ่งใช้ชื่อว่า Controller ดังแสดงในรูปที่ 5-14

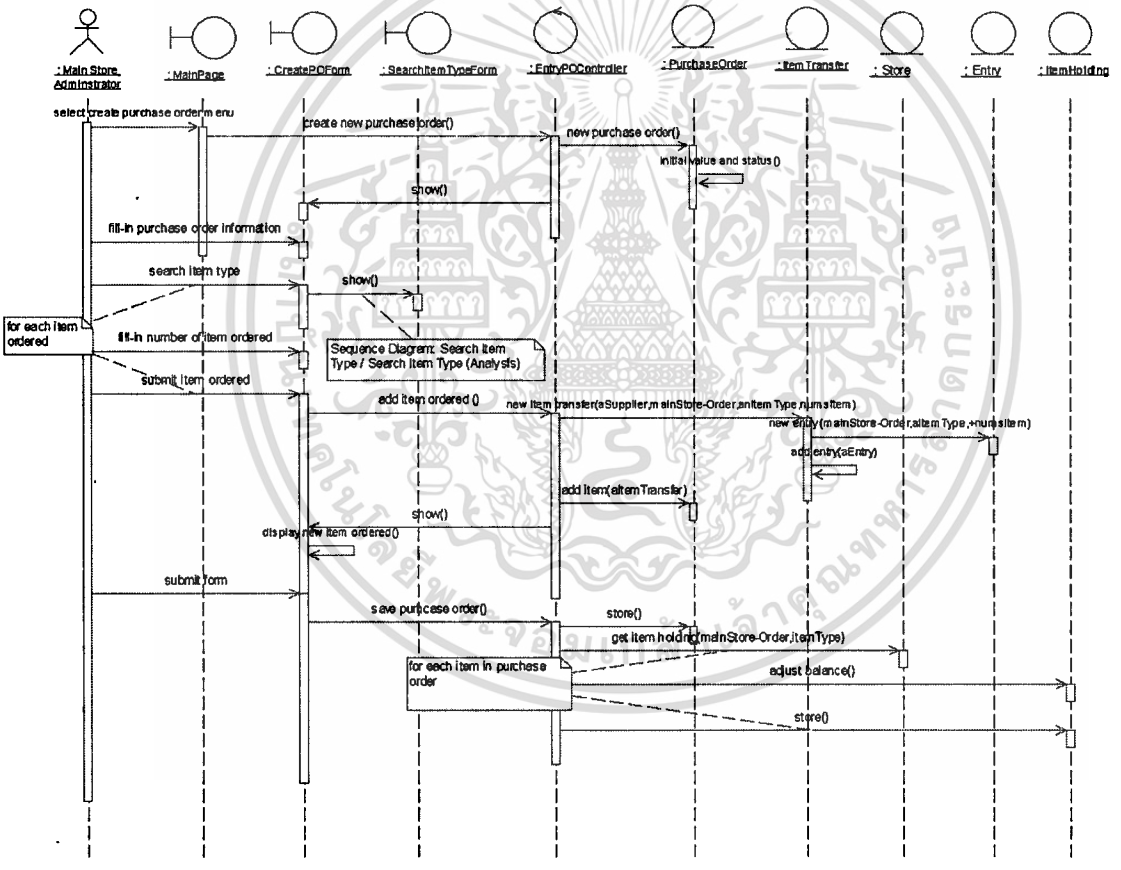


รูปที่ 5-14 ตัวอย่าง control classes ในระบบ SPCS

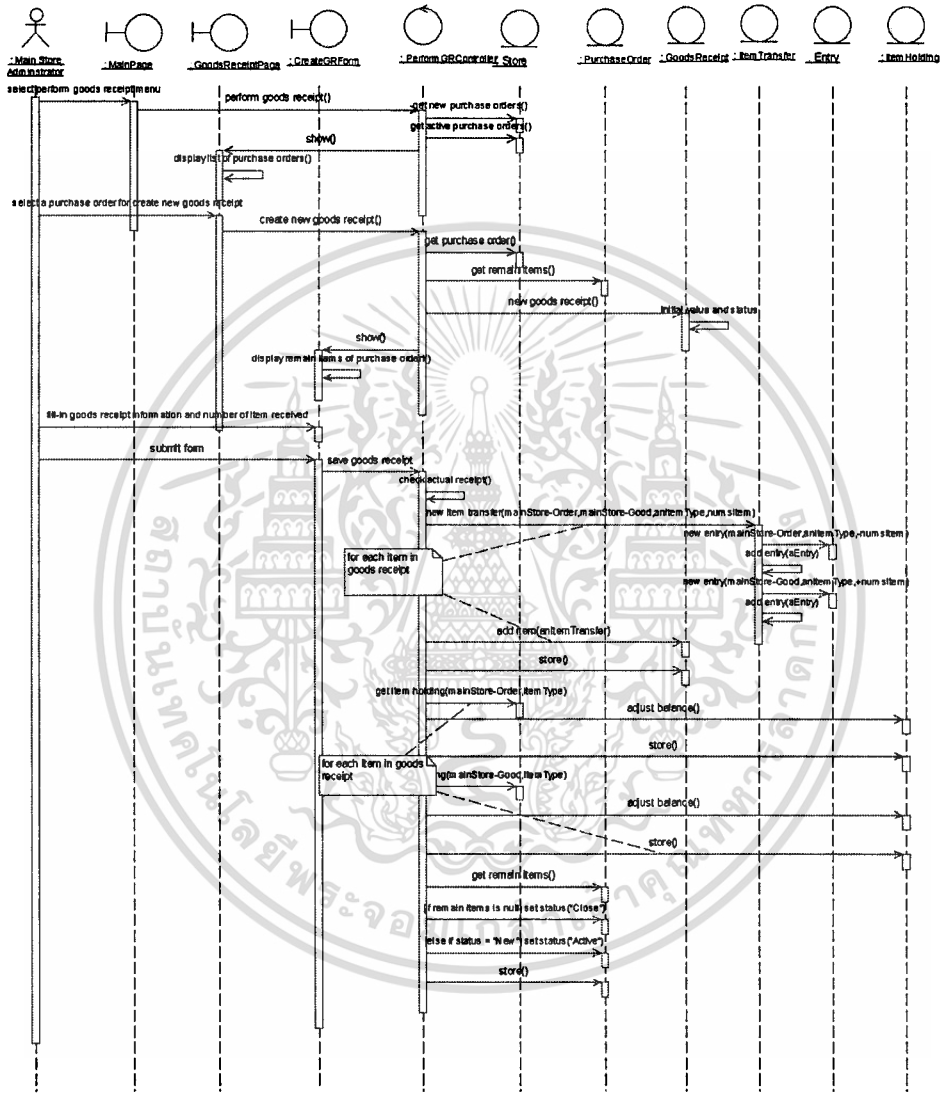
5.7 Analysis Sequence Diagram

เมื่อเราได้ Scenario (Use Case Description) และ Analysis Class แล้ว ในขั้นตอนถัดไป เราจะต้องทำการกำหนดหน้าที่ความรับผิดชอบ หรือ พฤติกรรม ต่างๆ ให้กับ classes ที่หามาได้ โดยจะต้องพิจารณาจาก Scenario ของแต่ละ use case ว่าในแต่ละ scenario ของแต่ละ use case นั้นมีขั้นตอนการทำงานอะไรบ้าง แต่ละขั้นตอน จะกำหนดให้ class ใดมีหน้าที่รับผิดชอบในการทำงาน

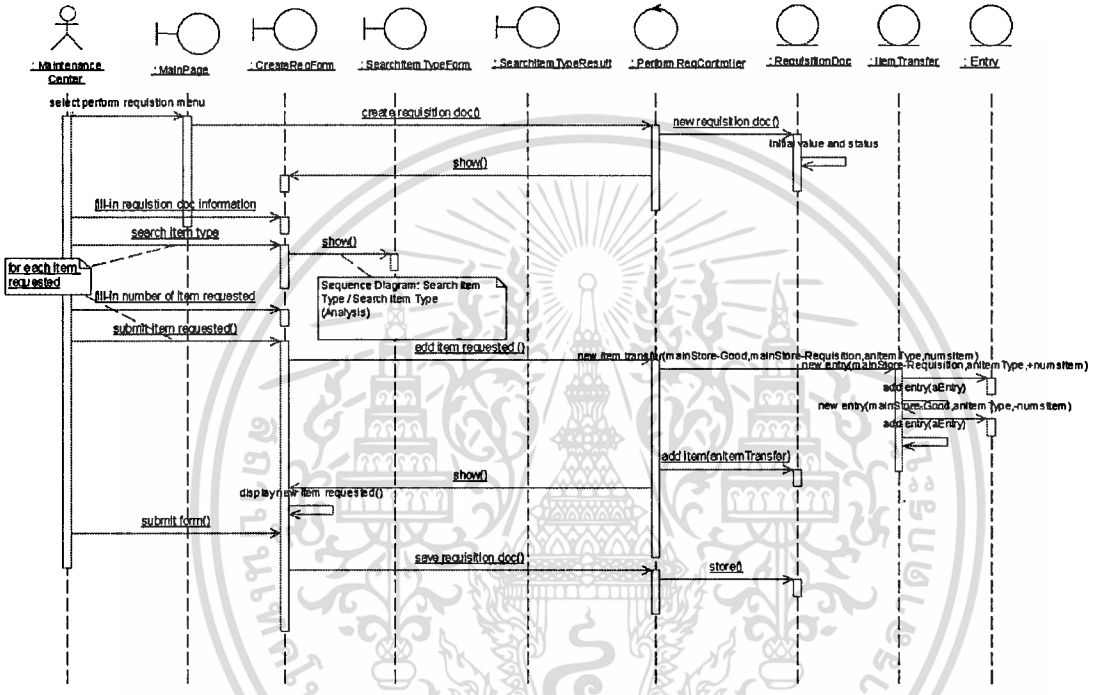
เทคนิควิธีการหนึ่ง ที่จะนำมาใช้ก็คือ การสร้างแบบจำลอง sequence diagram ขึ้นมา ซึ่งโดยทั่วไป เราจะทำการสร้างหนึ่ง sequence diagram ต่อ หนึ่ง use case เป็นอย่างน้อย ซึ่งรูปที่ 5-15 ถึง รูปที่ 5-20 จะแสดงตัวอย่าง sequence diagram ของระบบ SPCS



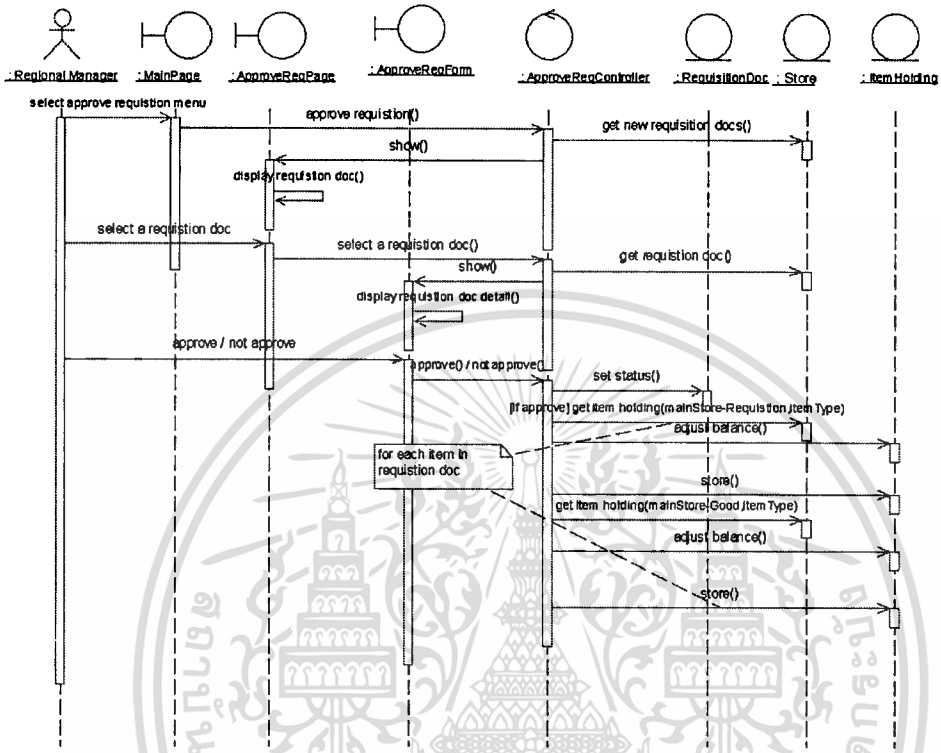
รูปที่ 5-15 Analysis Sequence Diagram ของ Use Case- Entry Purchase Order



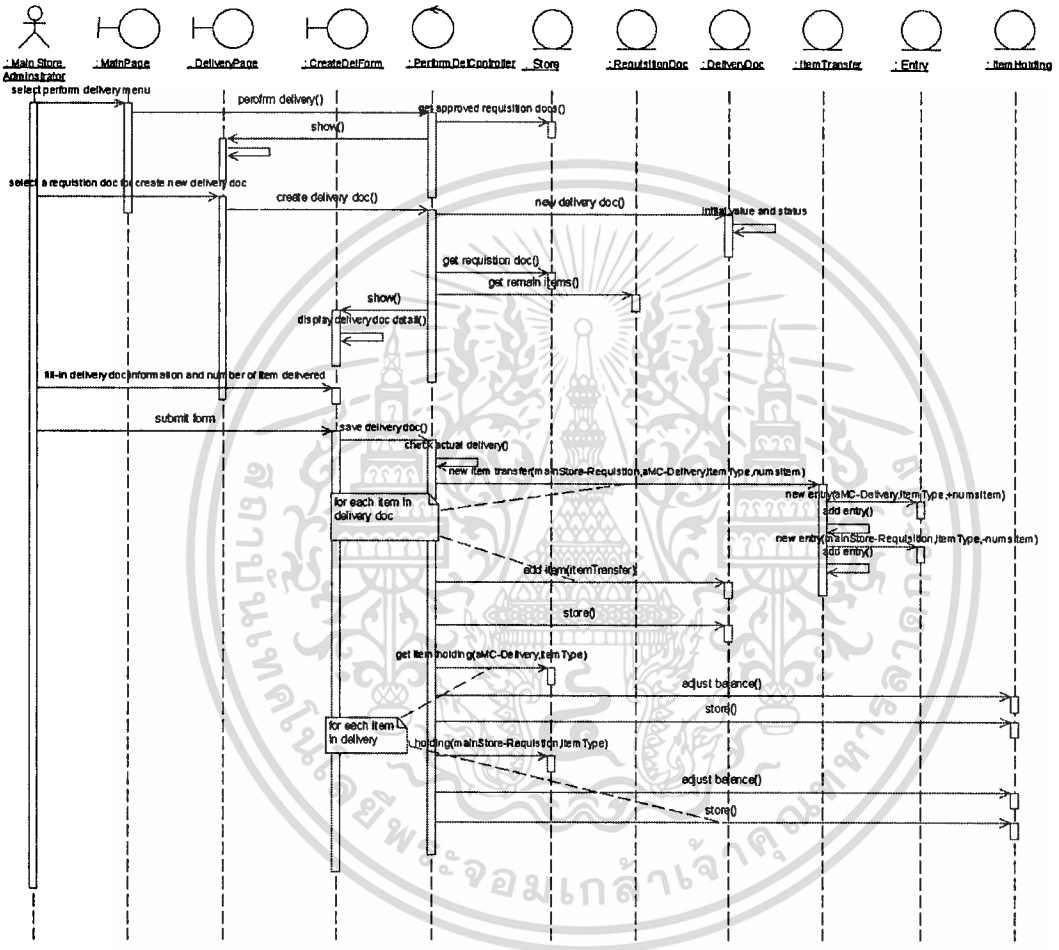
รูปที่ 5-16 Analysis Sequence Diagram ของ Use Case – Perform Goods Receipt



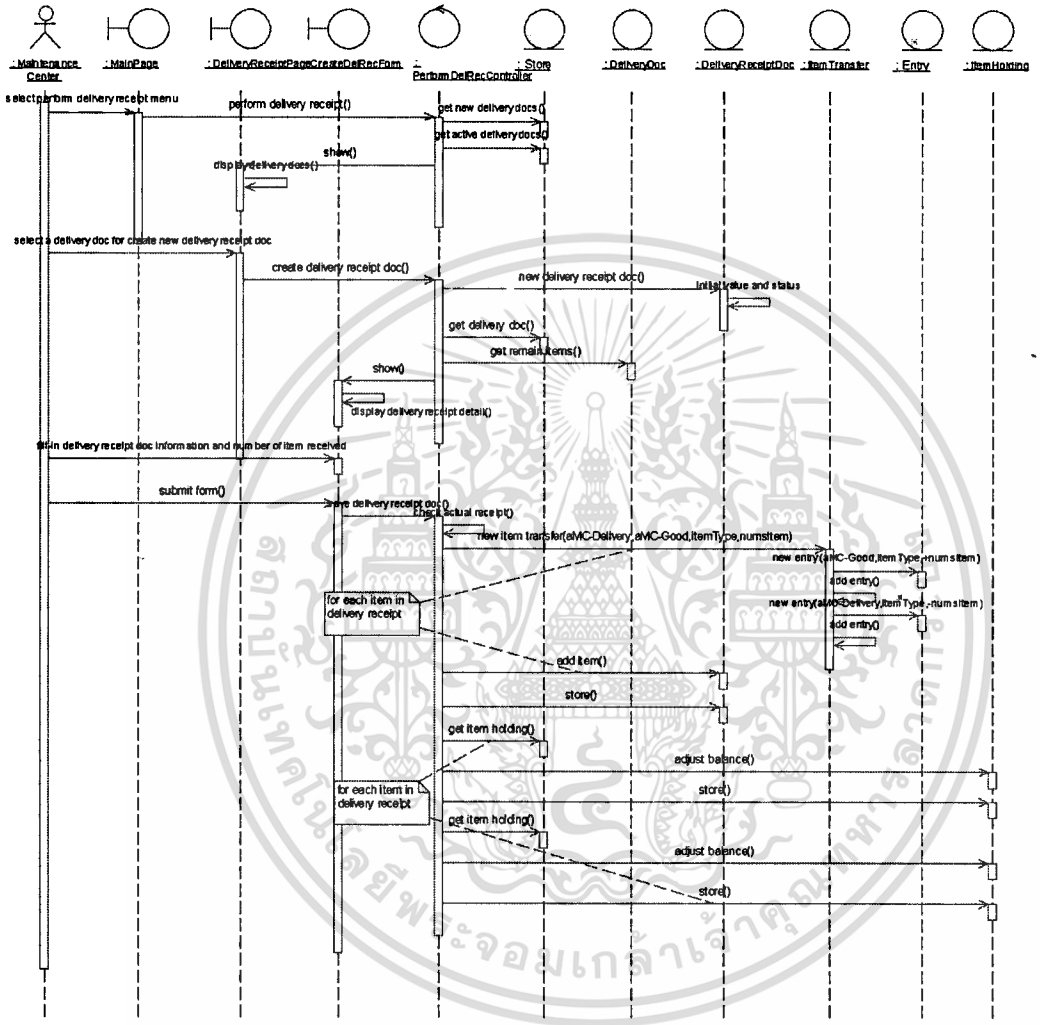
รูปที่ 5-17 Analysis Sequence Diagram ของ Use Case – Perform Requisition



รูปที่ 5-18 Analysis Sequence Diagram ของ Use Case – Approve Requisition



รูปที่ 5-19 Analysis Sequence Diagram ของ Use Case – Perform Delivery



รูปที่ 5-20 Analysis Sequence Diagram ของ Use Case – Perform Delivery Receipt

บทที่ 6
ออกแบบระบบ

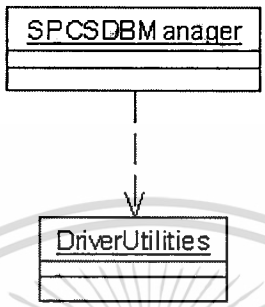
6.1Architectural Design

จากขั้นตอนของ Architecture Analysis นั้น ได้ได้มีการกำหนดสถาปัตยกรรมเบื้องต้นเอาไว้แล้ว ในขั้นตอนของ architectural design นี้ก็จะต้องมีการกำหนดรายละเอียดเพิ่มเติมเกี่ยวกับ กลไกการทำงานต่างๆของระบบ โดยคำนึงถึง เทคโนโลยีที่จะนำมาใช้ในการจัดสร้างระบบ รวมทั้งความเป็นไปได้ในการนำกลับมาใช้ได้อีกครั้ง(reuse) ในระยะยาวด้วย

กลไกการทำงานที่สำคัญอย่างหนึ่งของระบบ ที่จะต้องทำการออกแบบในรายละเอียด ก็คือ กลไกการจัดเก็บข้อมูล หรือ ที่เรียกว่า persistent mechanism ซึ่ง จะแตกต่างกันออกไป ขึ้นอยู่กับเทคโนโลยีของระบบการจัดเก็บข้อมูล หรือ ระบบจัดการฐานข้อมูล ว่า เป็นแบบใด เช่น RDBMS ODBMS หรือ ORDBMS และ เทคโนโลยีของภาษาโปรแกรม โดยในระบบ SPCS นี้ ได้เลือกใช้ RDBMS ของ Oracle และใช้ภาษาโปรแกรม เป็น ภาษา Java

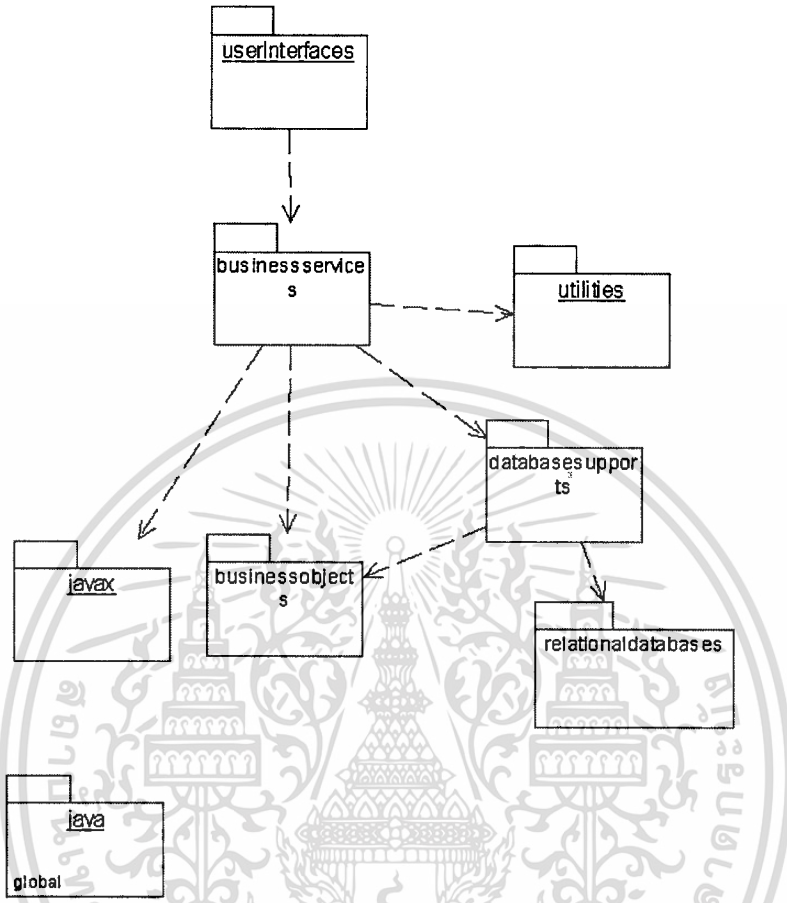
โดยทั่วไปแล้ว ในการออกแบบกลไกการจัดเก็บข้อมูล จากโครงสร้างการทำงานของระบบที่เป็น Object Oriented เข้าสู่ระบบจัดการฐานข้อมูลที่เป็น RDBMS นั้น มักจะมีการกำหนด หรือสร้าง Class ขึ้นมา อย่างน้อย หนึ่ง class เพื่อทำหน้าที่ในการ แปลง (mapping) โครงสร้างข้อมูลที่เป็น Object Oriented ให้ไปเป็น โครงสร้างข้อมูลที่เป็น Relational โดย class ดังกล่าวจะทำหน้าที่ในการแตก object ออกเป็นส่วนย่อยๆ และทำการเขียนลงไปใน table ต่างๆ ที่เกี่ยวข้องใน RDBMS ในทางตรงข้าม การอ่าน object จาก RDBMS นั้น class นี้ก็จะต้องทำหน้าที่ในการสร้างหรือประกอบ object ขึ้นมาจากข้อมูลใน table ต่างๆใน RDBMS ซึ่งภาษาที่ใช้ในติดต่อกับ table ต่างๆใน RDBMS ไม่ว่าจะเป็นการอ่านหรือเขียน ก็จะเป็น ภาษา SQL ซึ่งอาจจะมีความแตกต่างกันบ้าง ตามแต่ละยี่ห้อของ RDBMS โดยในภาษา Java นั้นสามารถที่เชื่อมต่อกับ RDBMS ต่างๆ ผ่านทาง JDBC โดยใช้วิธีการส่งคำสั่ง SQL ไปให้ RDBMS ทำงานต่อ แล้วรับผลลัพธ์ที่ได้มาประมวลผลต่อไป ซึ่งรายละเอียดเกี่ยวกับการติดต่อสื่อสารกับ RDBMS ทั้งหมดจะถูกห่อหุ้ม(encapsulate)ไว้ใน class นี้ ทำให้โปรแกรมอื่นๆ หรือ class อื่นๆที่มาเรียกใช้ class ไม่จำเป็นต้องรู้รายละเอียดต่างๆของการแปลงจาก Object Oriented ไปเป็น RDBMS ภายใน class นี้แต่อย่างใด โดยสามารถเรียกใช้ในลักษณะของ Object Oriented ได้ตามปกติ

ในระบบ SPCS นี้ก็ได้กำหนดให้มี class ที่ทำหน้าที่ดังกล่าวนี้ด้วย นั่นก็คือ class ที่ชื่อว่า SPCSDBManager รวมทั้ง class อื่นๆที่เกี่ยวข้องกับกลไกการจัดเก็บข้อมูล โดยจัดให้อยู่รวมกันใน package ที่ชื่อ databasesupports ดังแสดงรายละเอียดในรูปที่ 6-1



รูปที่ 6-1 classes ใน databasesupports package

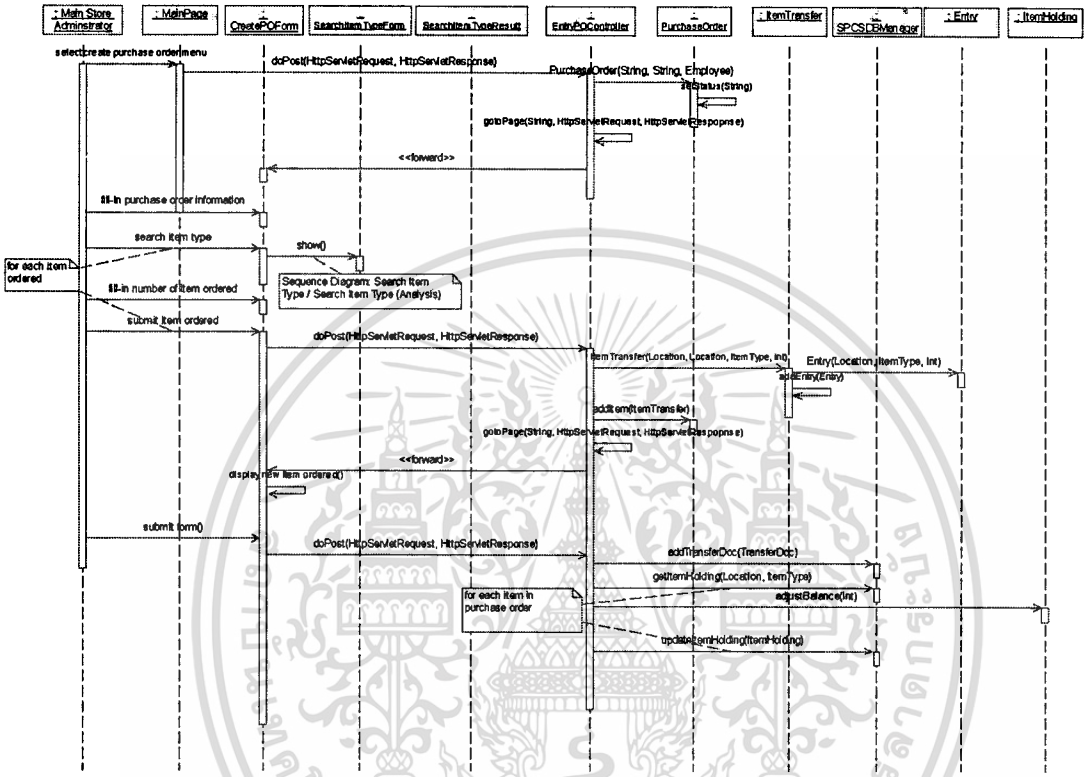
นอกจาก package database supports แล้ว ในระหว่างการทำ Architectural design เราสามารถที่จะหาหรือกำหนดให้มี package อื่นๆ เพิ่ม ได้อีก ขึ้นอยู่กับวิธีการ แนวทาง และเทคโนโลยี ที่เราจะนำมาให้ในการสร้างระบบว่า จะมีรายละเอียดเป็นอย่างไร โดย ทำการออกแบบไว้ในรูปของความสัมพันธ์กันระหว่าง package ต่างๆ เพื่อให้สามารถมองเห็นเป็นโครงสร้างของสถาปัตยกรรมของระบบได้ดังรูปที่ 6-2 จะแสดงโครงสร้างสถาปัตยกรรมของระบบ SPCS ที่ได้จากการทำ Architectural Design



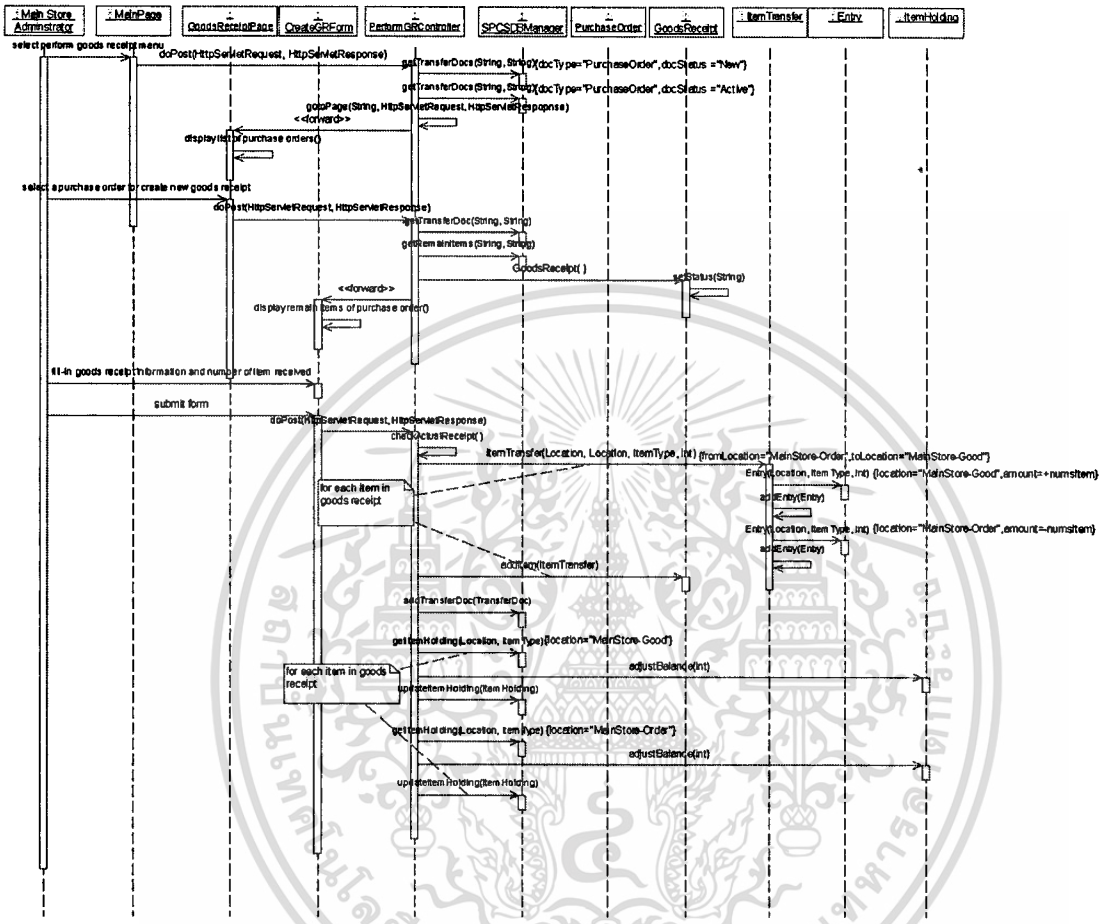
รูปที่ 6-2 Design Architecture ของระบบ SPCS

6.2 Design Sequence Diagram

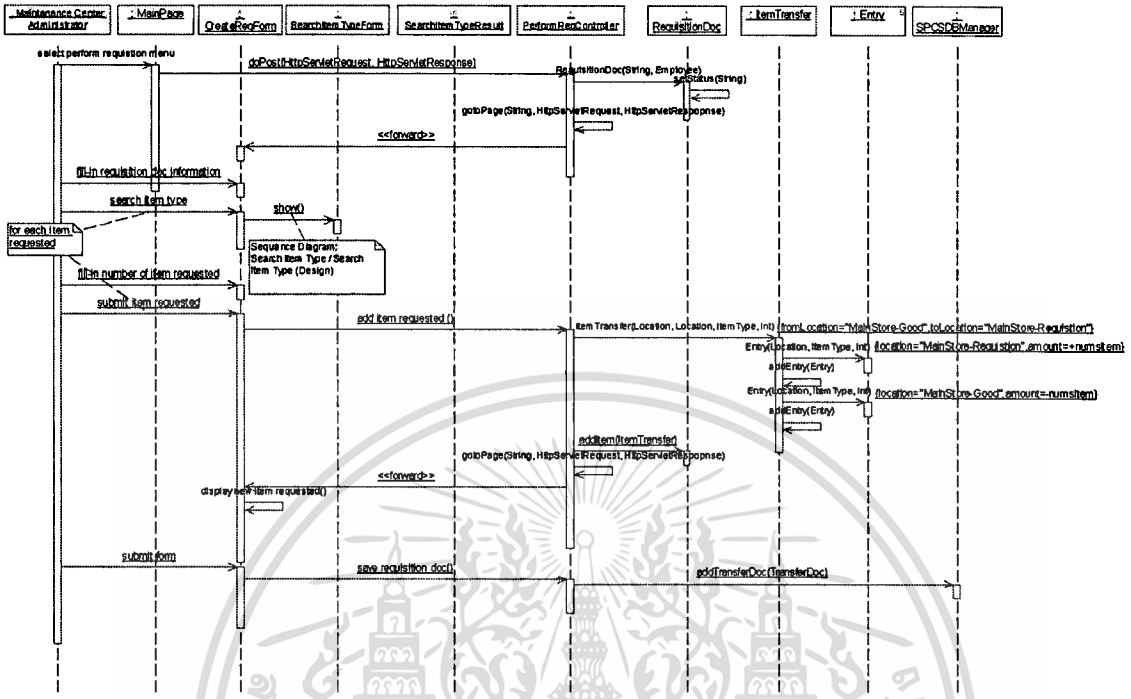
จากขั้นตอนของการวิเคราะห์ระบบ เราสามารถที่จะสร้างแบบจำลอง Sequence Diagram เพิ่มเติม จากระดับ วิเคราะห์(Analysis) ไปเป็น ระดับออกแบบ(Design) ได้ โดยการเพิ่มเติมและปรับปรุงรายละเอียดในระดับของการออกแบบเข้าไป ตัวอย่างเช่น เพิ่มเติมรายละเอียดเกี่ยวกับกลไกการจัดเก็บข้อมูลเข้าไป ทำการกำหนดชื่อที่แน่นอนให้แต่ละ message ที่ object ใช้ในการติดต่อสื่อสารกัน ซึ่งจะกลายไปเป็นชื่อของ operation หรือ method ของ object หรือ class ดังกล่าวนั่นเอง ดังแสดงตัวอย่าง ในรูปที่ 6-3 ถึง 6-6



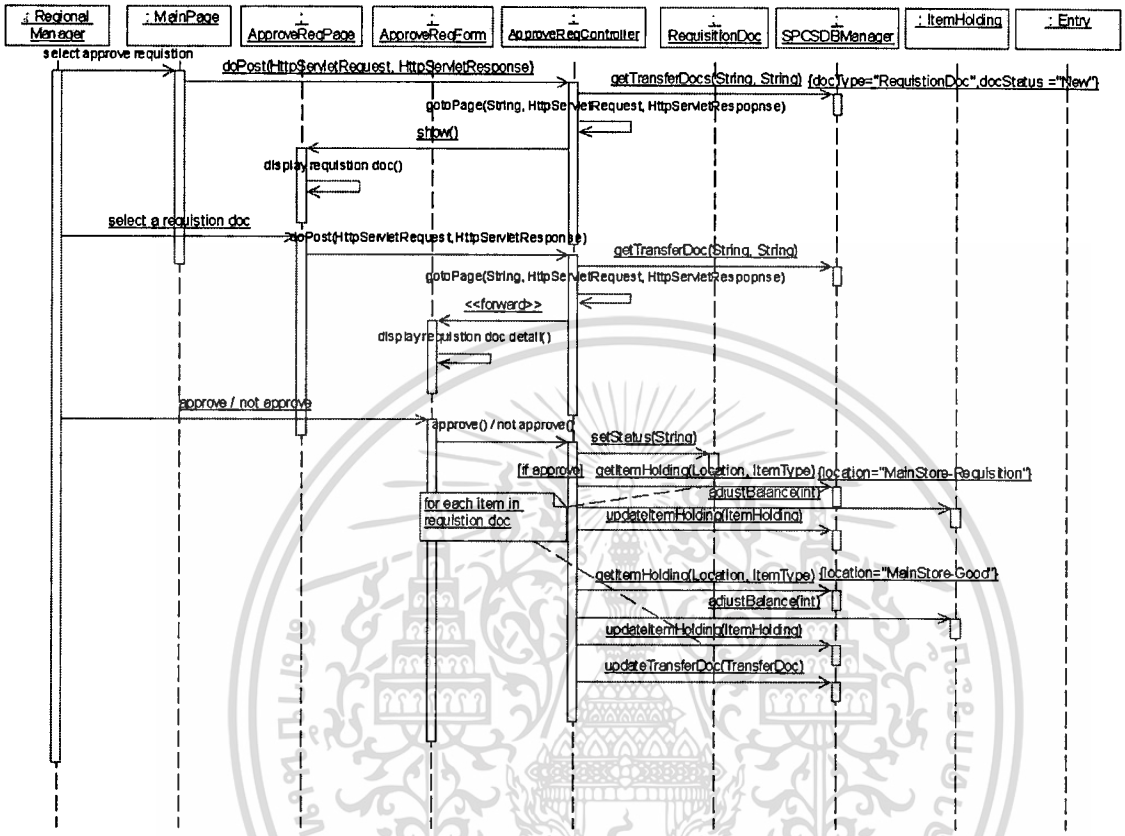
รูปที่ 6-3 Design Sequence Diagram ของ Use Case – Entry Purchase Order



รูปที่ 6-4 Design Sequence Diagram ของ Use Case – Perform Goods Receipt



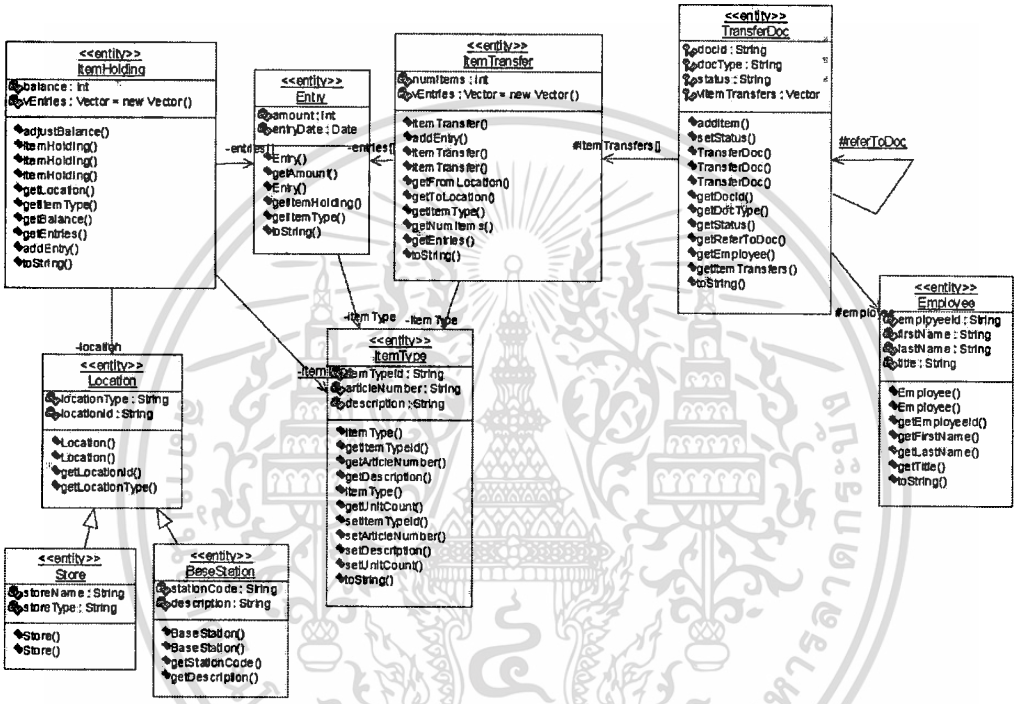
รูปที่ 6-5 Design Sequence Diagram ของ Use Case – Perform Requisition



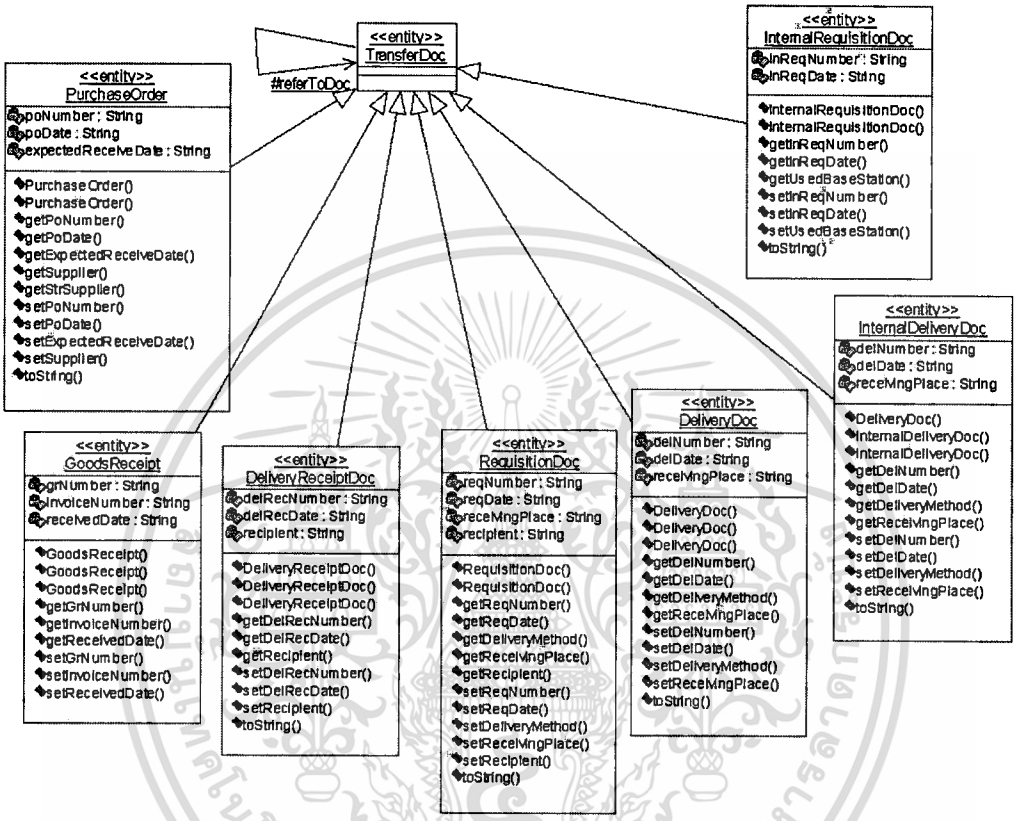
รูปที่ 6-6 Design Sequence Diagram ของ Use Case – Approve Requisition

6.3 Design Classes

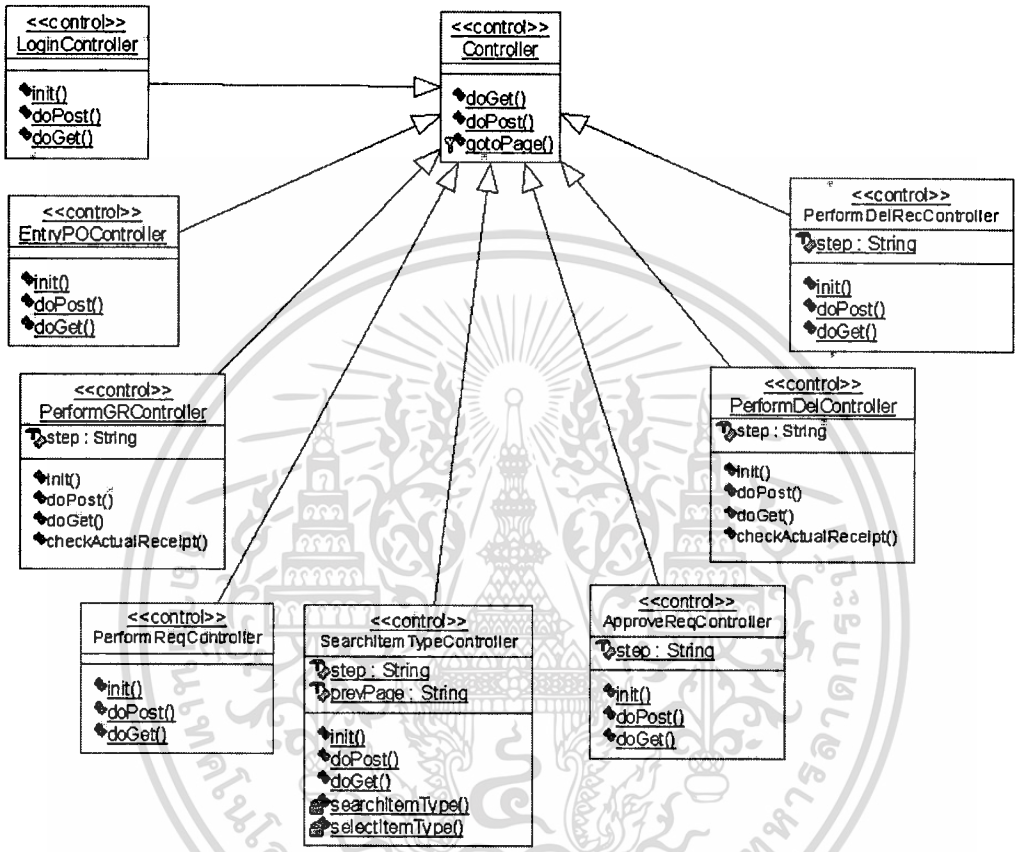
เมื่อทำการสร้างแบบจำลอง Sequence Diagram ในระดับ ออกแบบแล้ว เราสามารถที่จะปรับปรุงหรือเพิ่มเติมรายละเอียดของ class ต่างๆ ในระดับของการออกแบบได้ ดังแสดงตัวอย่างในรูปที่ 6-7 ถึง 6-10



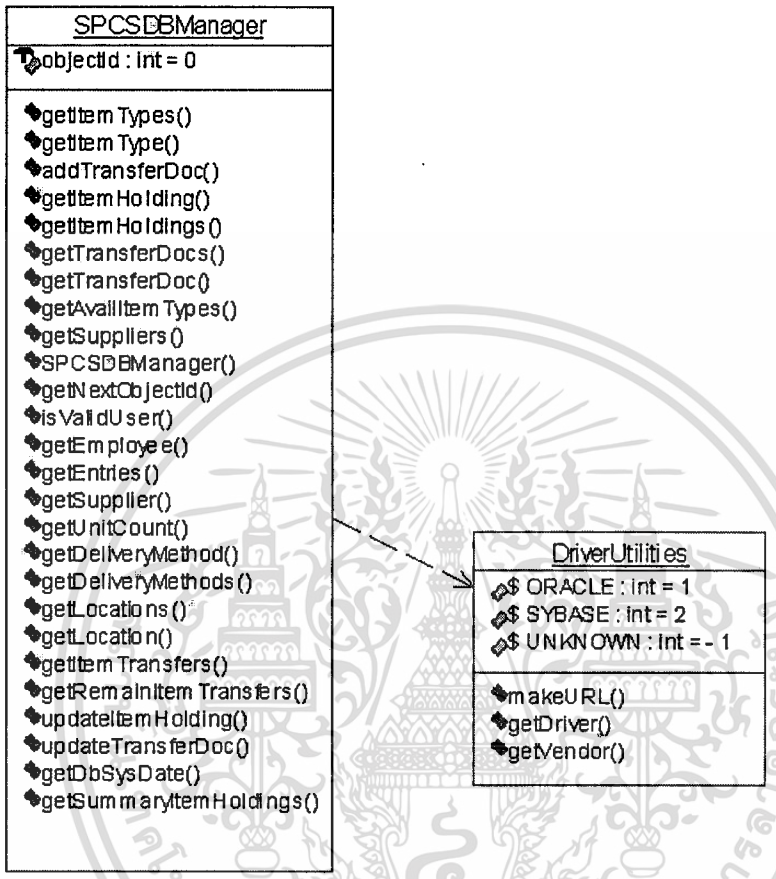
รูปที่ 6-7 Design Class Diagram ของ businessobjects package ส่วนที่ 1



รูปที่ 6-8 Design Class Diagram ของ businessobjects package ส่วนที่ 2



รูปที่ 6-9 Design Class Diagram ของ businessservices package

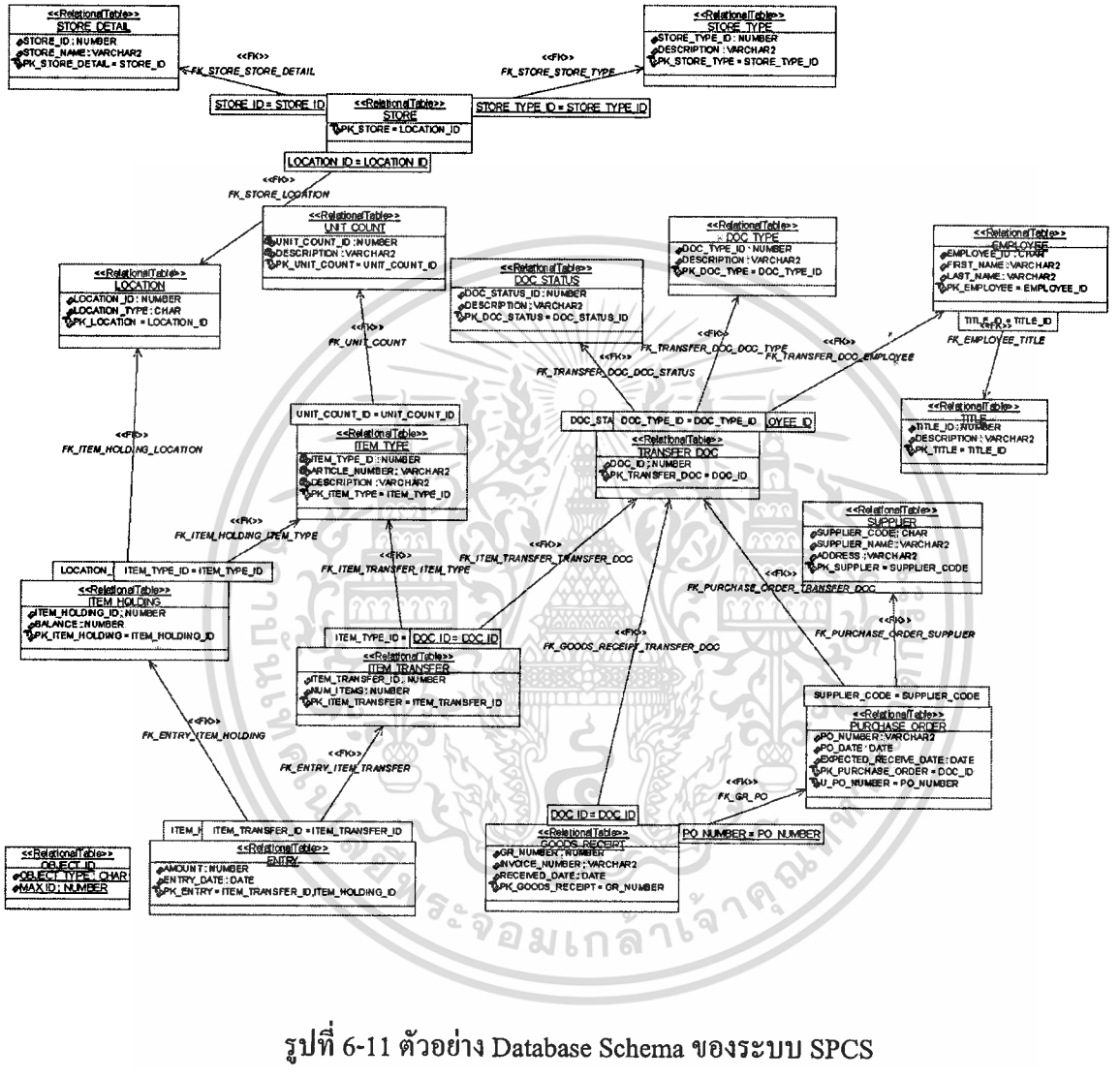


รูปที่ 6-10 Design Class Diagram ของ databasesupports package

6.4 Database Schema Design

จากที่ได้กล่าวไว้แล้วว่า ในระบบ SPCS นี้ ได้มีการใช้ระบบจัดการฐานข้อมูลที่เป็น RDBMS ดังนี้จึงจะต้องมีขั้นตอนในการออกแบบเพิ่มเติมขึ้นมาคือ การออกแบบโครงสร้างฐานข้อมูล หรือ Database Schema ซึ่งโดยทั่วไปแล้ว จะทำการ map จาก class ในกลุ่มของ entity classes โดยมีหลักการเบื้องต้น คือ 1 class ใน entity classes จะเป็น 1 table ใน database schema และ attribute ของ class ก็จะเป็น attribute หรือ column ของ table โดยจะต้องทำการแปลง data type ของ attribute ตาม data type ที่มีใช้อยู่ใน RDBMS ที่นำมาใช้ เช่น จาก String ก็ต้องแปลงไปเป็น VARCHAR2 ใน RDBMS ของ Oracle นอกจากนี้อาจจะมีการทำ normalization เพิ่มเติมจาก table ที่ได้ เพื่อลดความซ้ำซ้อนที่อาจจะเกิดขึ้น

โดยโครงสร้างฐานข้อมูลของระบบ SPCS นั้นสามารถสร้างแบบจำลองให้อยู่ในรูปแบบของ Class Diagram ได้ดังรูปที่ 6-11

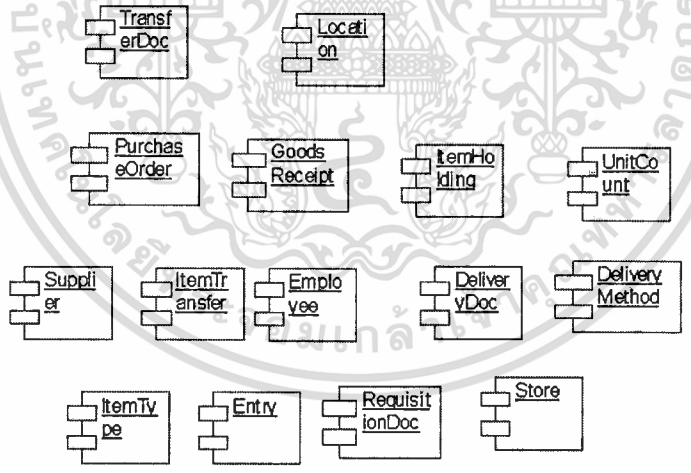


รูปที่ 6-11 ตัวอย่าง Database Schema ของระบบ SPCS

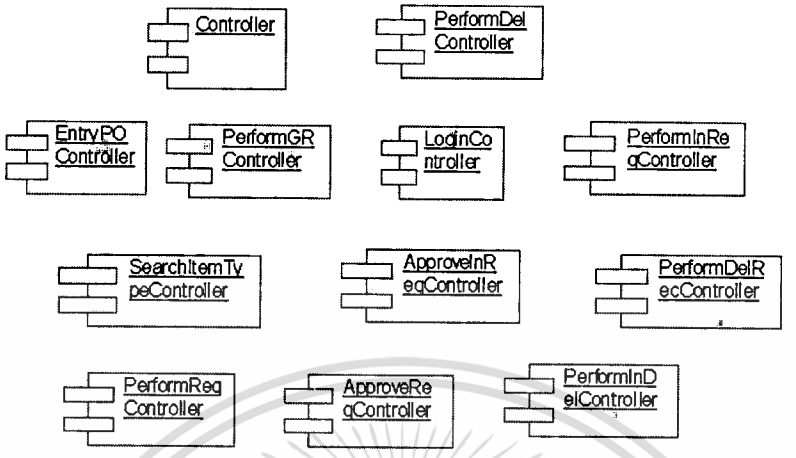
บทที่ 7
สร้างระบบ

7.1 Component Diagram

ภายหลังจากเราได้ออกแบบระบบเรียบร้อยแล้ว ขั้นตอนต่อก็จะเป็นการเริ่มต้นสร้างระบบ หรือ เริ่มต้นเขียนโปรแกรมนั่นเอง ซึ่งก่อนที่จะทำการเขียนโปรแกรมนั้น เราจะต้องทำการกำหนด class ต่างๆที่เราได้ออกแบบเอาไว้ ทั้งในส่วนของ entity classes ใน businessobjects package และ control classes ใน businessservices package ให้กับ component ซึ่งโดยทั่วไปในภาษา java แล้ว มักจะกำหนดให้ หนึ่ง class ต่อ หนึ่ง component ซึ่งก็จะได้เป็น component diagram ดังตัวอย่างในรูปที่ 7-1 และ 7-2



รูปที่ 7-1 ตัวอย่าง component diagram ของ businessobjects package



รูปที่ 7-2 ตัวอย่าง component diagram ของ businessservices package

ซึ่งเมื่อเราได้ component diagram แล้ว เราก็สามารถที่จะนำ component เหล่านี้ไปทำการ generate เป็นโครง หรือ header ของ source code ต่อไป ด้วยวิธีที่ได้กล่าวไว้ในบทที่ 4 ซึ่งส่วนที่เป็น entity classes จะถูกนำไปสร้างต่อเป็น class ภาษา java ในลักษณะที่เรียกว่า Java Bean และส่วนที่เป็น control classes จะถูกนำไปสร้างต่อเป็น class ภาษา java ในลักษณะที่เรียกว่า Java Servlet

ส่วน boundary classes ต่างๆ ใน userinterfaces package นั้น ในระบบ SPCS นี้จะไม่ได้นำมากำหนดให้กับ component และทำการ generate เป็น source code โดยตรง เนื่องจาก class เหล่านี้จะถูก map ไปเป็น web page ต่างๆของโปรแกรมซึ่งจะแบ่งเป็น ไฟล์ html ปกติ กับ ส่วนที่เป็น Java Server Page (JSP) ต่อไป

นอกจากนี้ class ใน package อื่นๆที่ได้กำหนดไว้จากขั้นตอนของการออกแบบ เช่น databasesupports package และ utilities package ก็จะถูกนำไปสร้างเป็น class ภาษา java ต่อไปด้วย

7.2 Source Code

เมื่อเราได้โครงของ source code ของ class ต่างๆแล้ว เราก็จะนำโครงเหล่านี้ไปทำการเขียนโปรแกรมในส่วนของเนื้อการทำงานจริงๆ ต่อไป โดยการเขียนโปรแกรมนั้น จะต้องอ้างอิงจาก Design Sequence Diagram ของ แต่ละ Use-Case ซึ่งในระบบ SPCS นี้ในแต่ละ Use-Case จะประกอบไปด้วยการทำงานร่วมกันระหว่าง Java Bean Java Servlet Java Server Page และ ส่วนที่เป็น Database Support ดังแสดงตัวอย่าง source code ดังนี้

7.2.1 Java Bean

จากที่กล่าวไว้แล้วว่า entity classes ต่างๆ จะถูกนำไปสร้างต่อเป็น java class ในลักษณะที่เรียกว่า Java Bean ซึ่งจะมีลักษณะดังรูปที่ 7-3 ซึ่งจะแสดงตัวอย่าง source code ของ class UnitCount

```
package businessobjects;

public class UnitCount
{
  private String unitCountId;
  private String description;

  public UnitCount()
  {
  }
  public UnitCount(String unitCountId,String description){
   this.unitCountId = unitCountId;
   this.description = description;
  }

  public String getUnitCountId() {

 return this.unitCountId;
  }

  public String getDescription(){
   return this.description;
  }

  public setUnitCountId(String unitCountId)
 {
    this.unitCountId = unitCountId;
 }

   public setDescription(String description)
  {
    this.description = description;
   }

}
```

รูปที่ 7-3 ตัวอย่าง source code ของ class UnitCount

7.2.2 Java Servlet

จากที่กล่าวไว้แล้วว่า control classes ต่างๆ จะถูกนำไปสร้างต่อเป็น java class ในลักษณะที่เรียกว่า Java Servlet ซึ่งจะมีลักษณะดังรูปที่ 7-4 ซึ่งจะแสดงตัวอย่าง source code ของ

class Controller โดยจะสังเกตเห็นว่ามีลักษณะเฉพาะอย่างหนึ่งของ class ที่เป็น Java Servlet ก็คือมีการ import package javax.servlet.* และ package javax.servlet.http.* และมีการ extent หรือ inherit มาจาก class ที่ชื่อว่า HttpServlet

```
package businessservices;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Controller extends HttpServlet {

 public void doGet(HttpServletRequest request,
                   HttpServletResponse response)
   throws ServletException, IOException {
  }


 public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
   throws ServletException, IOException {



 }

 protected void gotoPage(String address,
                   HttpServletRequest request,
                   HttpServletResponse response)
   throws ServletException, IOException {
  RequestDispatcher dispatcher =
   getServletContext().getRequestDispatcher(address);
  dispatcher.forward(request, response);
 }
}
```

รูปที่ 7-4 ตัวอย่าง source code ของ class Controller

และรูปที่ 7-5 จะแสดงตัวอย่าง source code ของ class EntryPOController ซึ่งเป็น control class หลักของ Use-Case Entry Purchase Order

```
package businessservices;

import businessobjects.*;
import databasesupports.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class EntryPOController extends Controller {
  ItemType selectedItemType;
  PurchaseOrder po;
  Employee emp;
  SPCSDBManager dbMgr;
  Supplier[] suppliers;

  public void init() throws ServletException{
      dbMgr = new SPCSDBManager();
      suppliers = dbMgr.getSuppliers();
  }

  public void doPost(HttpServletRequest request,
                HttpServletResponse response)
     throws ServletException, IOException {

    //get object from session
    HttpSession session = request.getSession(true);
    selectedItemType = (ItemType)session.getValue("selectedItemType");
    po = (PurchaseOrder)session.getValue("po");
    emp = (Employee)session.getValue("employee");

    //get po properties from request
    String poNumber = request.getParameter("poNumber");
    String poDate = request.getParameter("poDate");
    String expectedReceiveDate = request.getParameter("expectedReceiveDate");
    String toLocationId = request.getParameter("toLocationId");
    String selectedSupplierCode;
    if (request.getParameter("supplierCode")!=null) {
       selectedSupplierCode = request.getParameter("supplierCode");
    }else{
       selectedSupplierCode = "NA";
    }
    int numItems;
    try {
      numItems = Integer.parseInt(request.getParameter("numItems"));
    } catch(NumberFormatException nfe) {
      numItems = 0;
    }

    if (po == null) {
      po = new PurchaseOrder(dbMgr.getNextObjectId("TransferDoc"),"New",emp);
    }


    if (selectedItemType == null) {
      selectedItemType = new ItemType();
    }

    //set po properties
    po.setPoNumber(poNumber);
    po.setPoDate(poDate);
    po.setExpectedReceiveDate(expectedReceiveDate);
    po.setSupplier(dbMgr.getSupplier(selectedSupplierCode));
```

รูปที่ 7-5 ตัวอย่าง source code ของ class EntryPOController

```
    //po.addItem
    if (request.getParameter("addItem") != null) {
      ItemTransfer it = new ItemTransfer(null,
                                dbMgr.getLocation("Store",toLocationId),
                                selectedItemType,numItems);
      po.addItem(it);
    }//end if


    //put object to session
    session.putValue("suppliers",suppliers);
    session.putValue("selectedSupplierCode",selectedSupplierCode);
    session.putValue("po",po);
    session.putValue("selectedItemType",selectedItemType);
    session.putValue("employee",emp);

    //forward
    if (request.getParameter("addItem") != null) {
      System.out.println("forward to TestPage.jsp with new item");
      gotoPage("/jsp/CreatePOForm.jsp",request, response);
    } else if (request.getParameter("searchItemType") != null) {
      System.out.println("forward to SearchItemTypePage.jsp");
      session.putValue("prevPage","CreatePOForm");
      gotoPage("/jsp/SearchItemTypePage.jsp",request, response);
            } else if (request.getParameter("submitForm") != null) {
      System.out.println("forward to TestMain.html");
               dbMgr.addTransferDoc(po);

      ItemTransfer[] its = po.getItemTransfers();
       for(int i=0;i<its.length;i++){
         ItemHolding itemHolding = dbMgr.getItemHolding(dbMgr.getLocation("Store",toLocationId)
                                        ,its[i].getItemType());
         itemHolding.adjustBalance(+its[i].getNumItems());
         dbMgr.updateItemHolding(itemHolding);
       }


      gotoPage("/jsp/MainPage1.jsp",request, response);
    } else {
      System.out.println("forward to default page");
      gotoPage("/jsp/CreatePOForm.jsp",request, response);
    }


   }// doPost


public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                throws ServletException, IOException {
                        doPost(request,response);
}
}
```

รูปที่ 7-5 (ต่อ)

7.2.3 Java Server Page

จากที่กล่าวไว้แล้วว่า boundary classes ต่างๆ จะถูกนำไปใช้อ้างอิงในการสร้าง web page ต่างๆของระบบ ซึ่งโดยทั่วไป หนึ่ง boundary class ก็จะถูก map ไปเป็น html หรือ Java Server Page หนึ่ง page ดังแสดงในรูปที่ 7-6 ซึ่งแสดงตัวอย่างsource code ของ Java Server Page ที่ชื่อ GoodsReceipt.jsp

```
<html>
<head>
<title>G/R</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-874">
<link rel="stylesheet" href="css/style.css">
</head>

<body bgcolor="#FFFFFF">
<%@ page import="businessobjects.*" %>
<jsp:useBean id="newPos" class="TransferDoc[]" scope="session" />
<jsp:useBean id="activePos" class="TransferDoc[]" scope="session" />

<!--Sub Menu -->
<table width="100%" border="0" bordercolorlight="#000000" cellspacing="0"
bgcolor="#FFFFCC">
  <tr class="style">
    <td width="14%">
      <div align="center"><b><font size="2"><a href="/spcs/jsp/MainPage1.jsp">Ë¹éÒ¨ÍËÅÑ¡
</a></font></b></div>
    </td>
    <td width="14%" >
      <div align="center"><b><font size="2"><a
href="/spcs/servlet/businessservices.EntryPOController">ãºÊÑè§«×éÍ</a></font></b></div>
    </td>
    <td width="21%"  bgcolor="#CCCC00">
      <div align="center"><b><font size="2">ÃÑº¢Í§¨Ò¡ Suppliers </font></b></div>
    </td>
    <td width="32%"><b><font size="2"></font></b></td>
    <td width="19%"><b><font size="2"></font></b></td>
  </tr>
</table>
<p> </p>

<TABLE width="100%" border="0" cellspacing="0" >
<TR class="style"  bgcolor="#FFCCCC">
            <TD><div align="left"><b><font size="2">àÅ¢·ÕèãºÊÑè§«×éÍ</font></b></TD>
            <TD><div align="left"><b><font size="2">ÃËÑÊ¼Ùé¨ÓË¹èÒÂ</font></b></TD>
            <TD><div align="left"><b><font size="2">ª×èÍ¼Ùé¨ÓË¹èÒÂ</font></b></TD>
            <TD><div align="left"><b><font size="2">ÇÑ¹·ÕèÊÑè§«×éÍ</font></b></TD>
</TR>
<TR class="style" >
            <TD><div align="center"><b><font size="2">  </font></b></TD>
</TR>
<%for(int i=0;i< newPos.length;i++) {
      if (i%2==0) {
%>
<TR bgcolor="#FFFFCC">
<%  }else { %>
<TR bgcolor="#FFFF00">
<%  }
      PurchaseOrder po = (PurchaseOrder)newPos[i];
```

รูปที่ 7-6 ตัวอย่าง source code ของไฟล์ GoodsReceipt.jsp

```
%>
        <TD><A HREF="/spcs/servlet/businessservices.PerformGRController?step=2&docId=
<%=po.getDocId()%>"><%=po.getPoNumber()%></A></TD>
        <TD><%=po.getSupplier().getSupplierCode()%></TD>
        <TD><%=po.getSupplier().getSupplierName()%></TD>
        <TD><%=po.getPoDate()%></TD>
</TR>
<% } %>

<TR class="style" >
        <TD><div align="center"><b><font size="2">  </font></b></TD>
</TR>
<%for(int i=0;i< activePos.length;i++) {
    if (i%2==0) {
%>
<TR bgcolor="#FFFFCC">
<%  }else { %>
<TR bgcolor="#FFFF00">
<%  }
    PurchaseOrder po = (PurchaseOrder)activePos[i];

 %>
        <TD><A HREF="/spcs/servlet/businessservices.PerformGRController?step=2&docId=
<%=po.getDocId()%>"><%=po.getPoNumber()%></A></TD>
        <TD><%=po.getSupplier().getSupplierCode()%></TD>
        <TD><%=po.getSupplier().getSupplierName()%></TD>
        <TD><%=po.getPoDate()%></TD>
</TR>
<% } %>
</TABLE>

<p> </p>
</body>
</html>
```

รูปที่ 7-6 (ต่อ)

7.2.4 Database Support

class อีกกลุ่มหนึ่ง ที่มีความสำคัญในระบบ SPCS นี้ก็คือ class ใน databasesupports package โดยเฉพาะ class SPCSDBManager ซึ่งได้กล่าวถึงรายละเอียดของบทบาทและหน้าที่ของclass นี้ไว้ในบทที่ 6 แล้ว ซึ่งรูปที่ 7-7 จะแสดงตัวอย่าง source code ของ class SPCSDBManager โดยจะสังเกตเห็นว่า จะมีการ import package java.sql.* มาใช้งาน รวมทั้งมีการเขียนคำสั่ง SQL ฝังอยู่ใน method ต่างๆของ class ด้วย

```
package databasesupports;

import java.util.*;
import java.sql.*;
import businessobjects.*;
import utilities.*;


public class SPCSDBManager
{

  Connection conn;
  DatabaseUtilities dbUtil;
  ...
  ...
  ...

  ItemType[] itemTypes;
  Supplier[] suppliers;
  Store[] stores;
  ItemTransfer[] itemTransfers;
  Entry[] entries;


  int objectId=0;

  public SPCSDBManager()
  {

  int vendor = DriverUtilities.getVendor("oracle");
  String driver = DriverUtilities.getDriver(vendor);
  String host = "127.0.0.1";
  String dbName = "db1";
  String url = DriverUtilities.makeURL(host,dbName,vendor);
  String username = "sp";
  String password = "sp";

   try {
    Class.forName(driver);
    conn = DriverManager.getConnection(url,username,password);
//    stmt = conn.createStatement();
   }catch(ClassNotFoundException cnfe) {
     System.err.println("Error loading driver: "+ cnfe);
   }catch(SQLException sqle) {
     System.err.println("Error connecting: "+ sqle);
   }
  }

  ...
  ...
  ...

  public ItemType[] getItemTypes(String condition)
  {
   Vector vItemTypes=new Vector();

   try {
    Statement stmt = conn.createStatement();
    String query = "SELECT
ITEM_TYPE_ID,ARTICLE_NUMBER,T1.DESCRIPTION,T2.UNIT_COUNT_ID "+
                "FROM ITEM_TYPE T1,UNIT_COUNT T2 "+
                "WHERE T1.UNIT_COUNT_ID=T2.UNIT_COUNT_ID";
    ResultSet rs = stmt.executeQuery(query);
```

รูปที่ 7-7 ตัวอย่าง source code ของ class SPCSDBManager

```
    while(rs.next()){
      ItemType oItemType= new ItemType(rs.getString(1),rs.getString(2),rs.getString
(3),this.getUnitCount(rs.getString(4)));
      vItemTypes.addElement(oItemType);
    }//while

    itemTypes = new ItemType[vItemTypes.size()];
    for(int i=0;i<vItemTypes.size();i++){
                        itemTypes[i] = (ItemType)vItemTypes.elementAt(i);
                      }
    stmt.close();
   }catch(SQLException sqle) {
      System.err.println("Error connecting: "+ sqle);
   }

   return itemTypes;
  }

  ...
  ...
  ...

  public void addTransferDoc(TransferDoc newTransferDoc)
  {

   TransferDoc doc = newTransferDoc;     String referToDocId;
   if (newTransferDoc.getReferToDoc()!= null) {
      referToDocId = newTransferDoc.getReferToDoc().getDocId();
   }else{
      referToDocId = "null";
   }
  ...
  ...
  ...
 }

    public void updateTransferDoc(TransferDoc updatedTransferDoc)
  {
    try {

String docId = updatedTransferDoc.getDocId();
      String docStatus = updatedTransferDoc.getStatus();


          Statement stmt = conn.createStatement();

  String updTransferDoc = "UPDATE TRANSFER_DOC "+
                          "SET DOC_STATUS_ID = GET_DOC_STATUS_ID('"+docStatus+"') "+
                          "WHERE DOC_ID = "+docId;

          //      System.out.println(updTransferDoc);
                  stmt.executeUpdate(updTransferDoc);
              stmt.close();
    }catch(SQLException sqle) {
                                        System.err.println("Error connecting: "+ sqle);
    }

  }

  ...
  ...
  ...

}
```

รูปที่ 7-7 (ต่อ)

บทที่ 8
บำรุงรักษาระบบ

8.1 การเพิ่มเติมหรือขยายระบบ

การเพิ่มเติมหรือขยายการทำงานภายในระบบ SPCS นั้น อาจแบ่งได้เป็น สอง ลักษณะคือ

ลักษณะแรก เป็น การเพิ่มฟังก์ชันการทำงานใหม่ๆ เข้าไปในระบบ ซึ่งในลักษณะนี้ อาจจะจำเป็นที่จะต้อง เพิ่ม Use-Case ในระบบ จากนั้นก็ทำการวิเคราะห์และออกแบบตามขั้นตอนที่กล่าวมาทั้งหมด โดยฟังก์ชันการทำงานใหม่นี้ อาจสามารถที่จะใช้ object หรือ class ที่มีอยู่เดิมได้ทั้งหมด โดยเฉพาะส่วนที่เป็น entity classes ใน businessobjects package หรืออาจจะมีการสร้าง class ขึ้นมาเพิ่มเติม โดยเฉพาะ control classes ใน businessservices package ซึ่งมักจะขึ้นกับ use case ใด use case หนึ่งโดยเฉพาะ

ลักษณะที่สอง เป็นการเพิ่มเติม business object ใหม่ๆเข้าไปในระบบ โดยอาศัยฟังก์ชันการทำงานที่มีอยู่เดิม ในลักษณะนี้ ก็จะทำให้ต้องทำการสร้าง class ในส่วนของ entity classes แต่ไม่จำเป็นต้องสร้าง control classes ขึ้นมาใหม่ เพียงแต่ทำการปรับปรุงแก้ไข control class เดิมให้สามารถรองรับการทำงานร่วมกัน entity classes ใหม่ที่ถูกสร้างขึ้นมาก็พอ

ตัวอย่างหนึ่ง ของการเพิ่มเติมหรือขยายระบบ เช่น หากต้องการเพิ่มฟังก์ชันการทำงาน ในเรื่องของการส่งมอบทรัพย์สินจาก AIS ให้กับองค์การโทรศัพท์ ก็สามารถทำได้ ไม่ยากนัก เนื่องจาก ลักษณะการทำงานหลักๆ ของฟังก์ชันการทำงานนี้ จะเป็นการโอนย้าย ทรัพย์สินจาก AIS ไปเป็นทรัพย์สินของ องค์การโทรศัพท์ ซึ่งก็คล้ายกับการโยกย้ายของ จากสถานที่หนึ่ง ไป อีกสถานที่หนึ่ง ซึ่งเป็น รูปแบบหลักของระบบ SPCS อยู่แล้ว โดยมีสิ่งที่จะต้องเพิ่มเติม แก้ไข หรือ ได้รับผลกระทบเบื้องต้น ตาม package ต่างๆ ดังนี้

- **buinessservices package**

  จะต้องมีการเพิ่ม control class ขึ้นมาเพื่อทำ business logic ให้กับ Use-Case ใหม่ที่จะเกิดขึ้น

- **businessobjects package**

  จะต้องมีการเพิ่ม entity class เช่น class ของเอกสารแสดงรายละเอียดอุปกรณ์ที่ทำการส่งมอบ โดยอาจจะให้เป็น subclass หนึ่งของ class TransferDoc ก็ได้

- **databasesupports package**

เนื่องในส่วนของ class SPCSDBManager ได้มีการออกแบบในลักษณะที่เน้นความยืดหยุ่น และ การนำกลับมาใช้ได้ใหม่อยู่แล้ว จึงแทบไม่ต้องเพิ่ม method ใดๆ เข้าไป เพียงแต่ ทำการเพิ่มรายละเอียดในการรองรับการทำงานร่วมกับ entity class ใหม่ที่จะเกิดขึ้นเท่านั้น เช่น ใน method addTransferDoc(TransferDoc) ก็จะต้องทำการปรับปรุงเพิ่มเติมให้สามารถรองรับเอกสารการโอนย้ายลักษณะใหม่ๆที่จะเกิดขึ้น โดยไม่ต้องทำการเปลี่ยนแปลง parameter ที่ทำการรับส่ง แต่อย่างใด หาก class ใหม่ที่จะเกิดขึ้น ก็ เป็น subclass หนึ่งของ TransferDoc เช่นกัน

8.2 การเปลี่ยนแปลงระบบ

การเปลี่ยนแปลงระบบ SPCS ในอนาคต โดยเฉพาะอย่างยิ่ง การเปลี่ยนแปลงในส่วนของประเภทหรือยี่ห้อของระบบจัดการฐานข้อมูลนั้น สามารถกระทำได้โดยไม่กระทบกับการทำงานหลักๆของระบบ แต่อย่างใด เนื่องจากเราได้ออกแบบให้ class SPCSDBManager เป็นเพียง class เดียวที่จะทำการติดต่อหรือรู้จักกับระบบจัดการฐานข้อมูล แทน class อื่นๆ ดังนั้น หากจะมีการเปลี่ยนแปลงในส่วนของระบบจัดการฐานข้อมูลแล้ว ก็เพียงแต่ทำการแก้ไขหรือปรับปรุงวิธีการต่างๆในการเข้าถึงฐานข้อมูล ใน method ต่างๆของ class SPCSDBManager ให้สามารถรองรับ DBMS ยี่ห้อใหม่ๆ หรือ ประเภทใหม่ๆ เช่นเป็น ODBMS ได้เท่านั้น

บทที่ 9
สรุปผลการศึกษา

โครงการระบบควบคุมวัสดุอะไหล่เครือข่ายโทรศัพท์มือถือผ่านเว็บนี้ ถือได้ว่าเป็นกรณีศึกษาหนึ่งของการนำวิธีการวิเคราะห์และออกแบบระบบเชิงวัตถุมาใช้ โดยในขั้นต้นนั้นจะได้มีการนำไปติดตั้ง และทดลองใช้งานในลักษณะเป็น pilot project หนึ่งของแผนก IT System ต่อไป โดยระบบ SPCS นี้ จะทำงานอยู่บน ระบบปฏิบัติการ Windows 2000 ระบบจัดการฐานข้อมูล Oracle 8 และใช้ Jakarta Tomcat เป็นโปรแกรม Web Server

และจากการที่ได้นำวิธีการวิเคราะห์และออกแบบระบบเชิงวัตถุมาใช้ทดลองในการพัฒนาระบบที่มีความซับซ้อนค่อนมากอย่างเช่นระบบ SPCS นี้พบว่า การวิเคราะห์และออกแบบเชิงวัตถุนี้ สามารถที่จะทำให้เราสามารถวิเคราะห์และออกแบบระบบได้อย่างมีประสิทธิภาพ รวมทั้งระบบที่ได้ก็มีความยืดหยุ่น และ มีความสามารถในการนำกลับมาใช้ได้ใหม่ ค่อนข้างมาก ดังตัวอย่างที่อธิบายไว้ในบทที่ 8 ในเรื่องของการบำรุงรักษาระบบ

ในส่วนของการศึกษาการพัฒนาด้วย UML นั้น ทำให้ ผู้จัดทำโครงการมีความเชี่ยวชาญในการใช้ UML และ Rational Rose เป็นเครื่องมือในการวิเคราะห์และออกแบบระบบงานในระดับหนึ่ง โดยมีความเห็นว่าน่าจะสามารถนำมาใช้เป็นเครื่องมือช่วยในการพัฒนาระบบงานได้เป็นอย่างดี เพราะสามารถช่วยลดระยะเวลาการทำงานได้ ดังนั้น UML จึงเป็นมาตรฐานที่มีความน่าสนใจและเหมาะสมกับการพัฒนาระบบงานสารสนเทศต่อไปในอนาคต

บรรณานุกรม

เบญจรงค์ นิตยพัฒน์. 2543. **"การพัฒนาห้องสมุดดิจิตอลโดยใช้ UML."** โครงการพัฒนาระบบงาน วิทยาศาสตรมหาบัณฑิต(เทคโนโลยีสารสนเทศ) คณะเทคโนโลยีสารสนเทศ, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

CDG System. 2000. **Object-oriented Knowledge.** [Online]. Available : http://www.cdg.co.th/gcc/html/oo-knowledge.html

Conallen, J. 2000. **Building web applications with UML** : Addison-Wesley.

Eriksson, Hans-Erik and Penker, M. 1998. **UML Toolkit** : John Wiley & Sons.

Fowler, M. 1997. **Analysis patterns : reusable object models** : Addison-Wesley.

Larman, C. 1998. **Applying UML and patterns : an introduction to object-oriented analysis and design** : Prentice Hall.

Pressman, S. 1997. **Software Engineering : A Practitioness's approach. 4th ed.** Singapore : McGraw-Hill.

Quatrani, T. 1998. **Visual Modeling with Rational Rose and UML.** Massachusetts : Addison Wesley Longman.

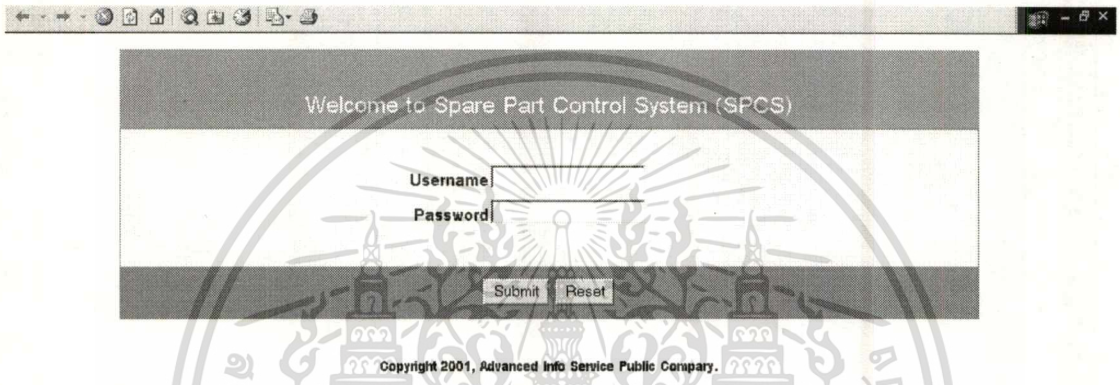
Rational University. 1999. **Object-Oriented Analysis and Design Student Manual**: Rational Software.

Rosenberg, D. 1999. **Use Case Driven Object Modeling with UML: A Practical Approach** : Addison-Wesley.

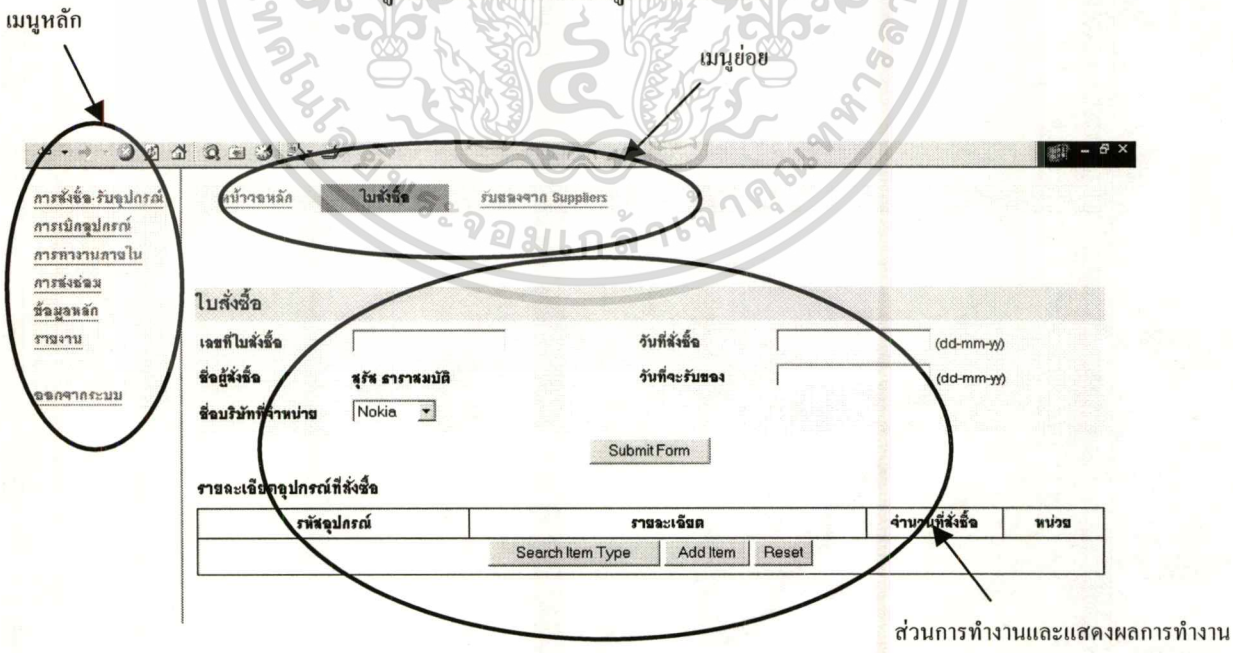
Skylstad, G. and Wuwongse, V. 1998. **"Component and Object-oriented Business System Development with BPR, UML, Java, COM+ and Cobra."** Bangkok : National Electronic and Computer Technology Center. Seminar Document.

ภาคผนวก

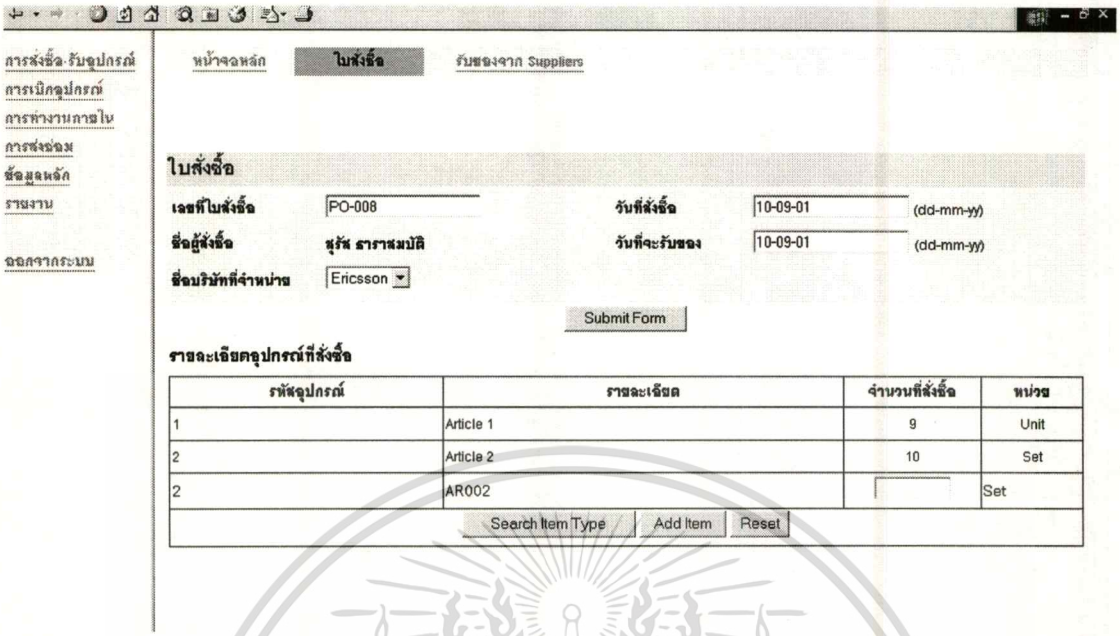
ก. ตัวอย่างหน้าจอของระบบ



รูปที่ ก-1 หน้าจอเข้าสู่ระบบ SPCS

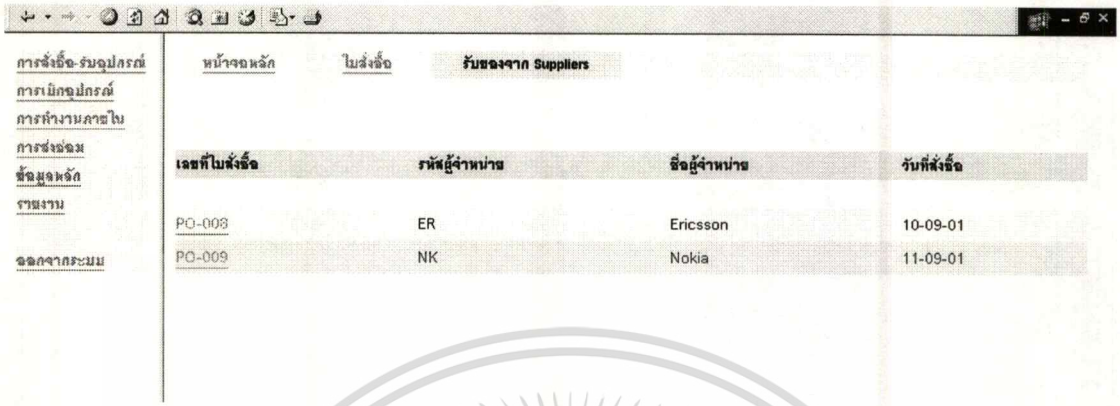


รูปที่ ก-2 รูปแบบหน้าจอมาตรฐานเมื่อเข้าสู่ระบบแล้ว

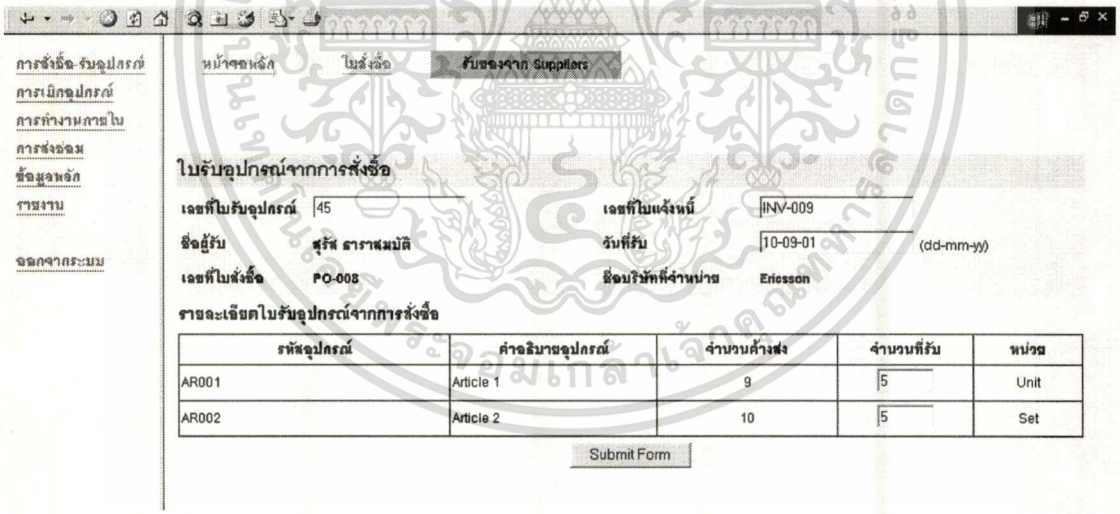


รูปที่ ก-3 หน้าจอขณะกรอกข้อมูลใบสั่งซื้อ

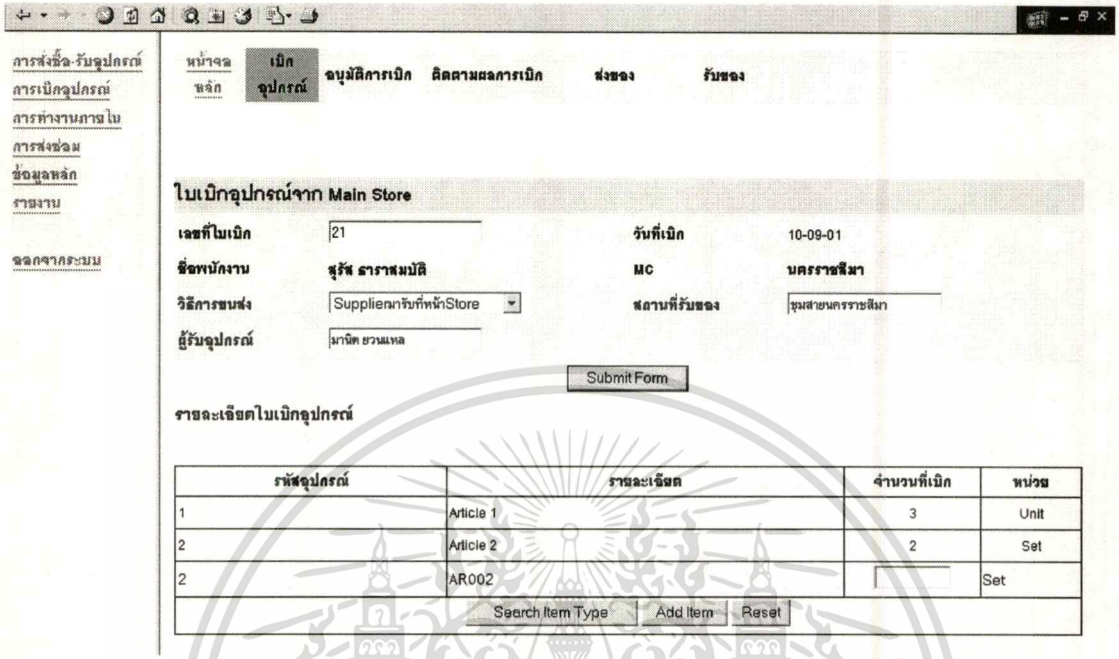
รูปที่ ก-4 หน้าจอการเลือกรายการอุปกรณ์



รูปที่ ก-5 หน้าจอรายการใบสั่งซื้อที่สามารถทำใบรับของได้

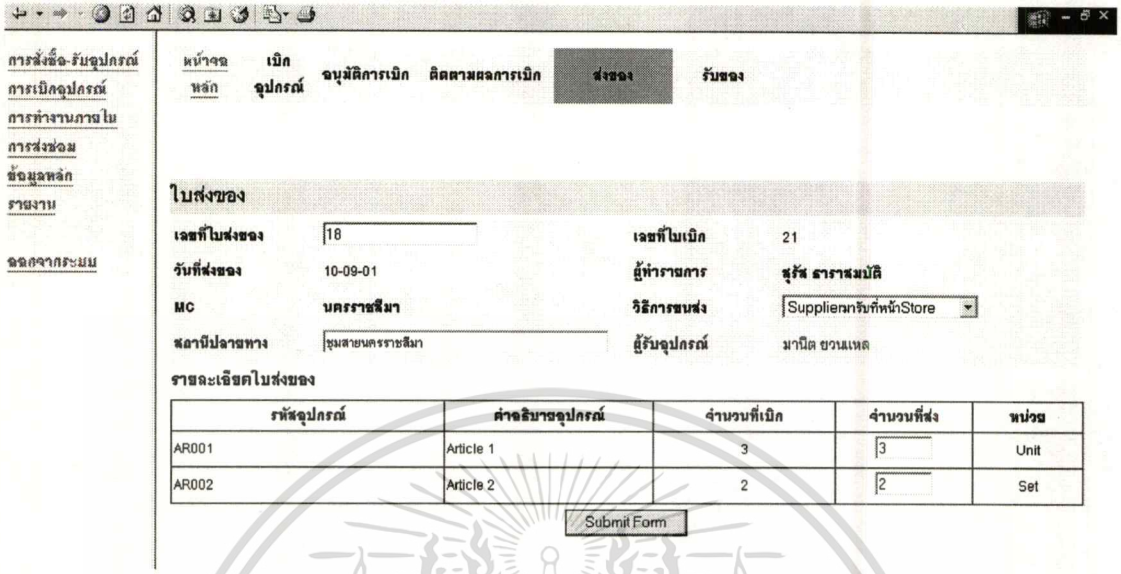


รูปที่ ก-6 หน้าจอการทำใบรับของจากSupplier



รูปที่ ก-7 หน้าจอการทำใบเบิกอุปกรณ์จาก Main Store

รูปที่ ก-8 หน้าจอการอนุมัติใบเบิกอุปกรณ์จาก Main Store



รูปที่ ก-9 หน้าจอการทำใบส่งของ

ประวัติผู้เขียน

ชื่อผู้เขียน	นายสุรัส ธาราสมบัติ
วันเดือนปีเกิด	9 กรกฎาคม 2519
สถานที่เกิด	จังหวัดกรุงเทพมหานคร
วุฒิการศึกษาระดับปริญญาตรี	วท.บ. (วิทยาการคอมพิวเตอร์)
สถานที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ มหาวิทยาลัยหอการค้าไทย
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2539