การพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์ด้วยเทคโนโลยีคอมพลัส

Component-Based Software Development with COM+

นายอนุสรณ์ กฤษณวงษ์หงษ์

นางสาวอัญชลี ฐิตรัตนพล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

การพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์ด้วยเทคโนโลยีคอมพลัส

Component-Based Software Development with COM+

โดย

นายอนุสรณ์ กฤษณวงษ์หงษ์

นางสาวอัญชลี จิตรัตนพล

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา

ดร. ชุติเมษฎ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์ด้วยเทคโนโลยีคอมพลัส

Component-Based Software Development with COM+

ผู้จัดทำ

1. นายอนุสรณ์ กฤษณวงษ์หงษ์ รหัสประจำตัว 41014512
2. นางสาวอัญชลี ฐิตรัตนพล รหัสประจำตัว 41014534

_____ อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

_____ อาจารย์ที่ปรึกษา

(ดร. ชุติเมษฎ์ ศรีนิลทา)

การพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์

นายอนุสรณ์ กฤษณวงษ์หงษ์
นางสาวอัญชลี ฐิตรัตนพล
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ดร. ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ในปัจจุบันซอฟต์แวร์ที่พัฒนาขึ้นมามักมีปัญหาทางด้านการแก้ไขข้อบกพร่อง การบำรุงรักษา รวมไปถึงความน่าเชื่อถือในการใช้งาน ถึงแม้ว่าจะมีการปรับเปลี่ยนมาเป็นการพัฒนามาเป็นการพัฒนาเชิงวัตถุแล้วก็ตาม แต่ยังไม่สามารถตอบสนองความต้องการพัฒนาซอฟต์แวร์ที่ต้องการความรวดเร็วในการพัฒนา ซึ่งทำให้เกิดแนวความคิดใหม่ของการนำกลับมาใช้ใหม่ ซึ่งมีประสิทธิภาพสูงกว่า และเชื่อถือได้มากกว่ารูปแบบเดิม

เทคโนโลยีคอมพลัส (Component Object Model Plus: COM+) เป็นหนึ่งในเทคโนโลยีที่มีความสามารถในการจัดการกับความต้องการต่างๆ ดังที่กล่าวมา โดย COM+ พัฒนามาจาก COM ซึ่งเป็นเทคโนโลยีในการพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์ซึ่งเป็นอิสระต่อภาษาที่ใช้ในการพัฒนา นอกจากนี้ COM+ ยังจัดเตรียมบริการต่างๆ ที่ช่วยอำนวยความสะดวกในด้านต่างๆ เช่น ความสามารถในการจัดการทรานส์แอ็กชัน การจัดการด้านความปลอดภัย เป็นต้น โดยไม่ต้องเขียนโปรแกรมเพิ่มเติมเลย

วิทยานิพนธ์ฉบับนี้ศึกษาถึงวิธีในการพัฒนาระบบงานขนาดใหญ่ โดยใช้วิธีการพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์ (COM) การเรียกใช้บริการของคอมพลัส การกระจายโหลดการทำงาน รวมไปถึงการเรียกใช้บริการที่พัฒนาจากเทคโนโลยีอีกตัวหนึ่ง คือ Enterprise JavaBeans (EJB)

ตัวอย่างการพัฒนาระบบงานขนาดใหญ่ที่วิทยานิพนธ์ฉบับนี้นำเสนอคือ ระบบการจองแพ็กเกจท่องเที่ยวและระบบการจองโรงแรม ซึ่งใช้หลักการพัฒนาเชิงคอมโพเน็นต์ ตามโครงสร้างของวินโดวส์ดีเอ็นเอ (Windows DNA) โดยภาษาที่ใช้ในการพัฒนา คือ ภาษาวิชวลเบสิก เอเอสพี และเอเอสพีดอตเน็ตเว็บเซอร์วิส (ASP.NET Web Services) ส่วนระบบการจองสายการบินและระบบการจองรถโดยสารจะเรียกใช้บริการที่พัฒนาจากเทคโนโลยี EJB ผ่านโพรโตคอลมาตรฐานคือ โซป (SOAP: Simple Object Access Protocol)

Component-Based Software Development with COM+

Anuson Kissanawonghong
Anchalee Thitarattanaphol
Dr. Worawat Limpoka Advisor
Dr. Shutimet Srinilta Advisor

ABSTRACT

Recently, software development has a problem such as updating application, maintenance and reliability. Object-oriented development cannot solve these problems completely. It also cannot meet requirement of rapid software development. So the concept of "Reusability" that is more efficiency and reliability than legacy system was used.

Component Object Model Plus (COM+) Technology is one of base technology that can manage with non-functional requirement. COM+ was developed from COM, which is Component-Based software development technology, that is language heterogeneity. Additionally, COM+ has other service that help developer can develop easier such as Microsoft Transaction Server (MTS), COM+ Security.

This thesis is study in developing large-scale system by Component Based Development Methodology, use COM+ Service, Load Balancing and use service that developed from Enterprise JavaBeans (EJB) Technology.

Example of large scale system development in this thesis are hotel agency and travelling agency with reserve package tour, hotel room, bus seat and airline seat by using component based development rely on Windows DNA Architecture. Languages that were used to develop are Visual Basic, ASP and ASP.Net Web Services. In part of Airline seat reservation and Bus seat reservation are use service that develop from EJB Technology by using Simple Object Access Protocol (SOAP)

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์วรวัฒน์ ลิ้มโภคา และอาจารย์ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา นอกจากนั้นยังได้ให้ความรู้และแนวคิดในการทำงานรวมถึงการใช้ชีวิต อันมีค่ายิ่งให้แก่ผู้เขียน ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอขอบคุณพี่ ๆ เพื่อนๆ ห้อง OLALA และเพื่อนๆ ห้องเน็ตเวิร์ก ที่ช่วยเป็นกำลังใจและให้ประสบการณ์ในการใช้ชีวิตในมหาวิทยาลัยที่มีค่ายิ่งใช้กับผู้เขียน

สุดท้ายนี้ ต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา ในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

นายอนุสรณ์ กฤษณวงษ์หงษ์
นางสาวอัญชลี ฐิตรัตนพล

สารบัญ

หน้าที่

สารบัญตาราง

ตารางที่ หน้าที่

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

การพัฒนาโปรแกรมคอมพิวเตอร์นั้นได้มีการพัฒนาจากการใช้การพัฒนาโดยรูปแบบโครงสร้าง (Structural Development) ไปเป็นการพัฒนาเชิงวัตถุ (Object Oriented Development) ซึ่งก็มีข้อดีที่สามารถจัดการกับระบบที่มีความซับซ้อนสูงได้อย่างมีประสิทธิภาพมากขึ้น แต่ในขณะเดียวกัน การพัฒนาเชิงวัตถุก็มีจุดบกพร่องอยู่เป็นจำนวนมาก เช่น จุดบกพร่องของหลักการสืบทอดคุณสมบัติ (Inheritance) การนำกลับมาใช้ใหม่ ซึ่งยังไม่มีประสิทธิภาพเพียงพอ เป็นต้น ซึ่งทำให้เกิดแนวความคิดใหม่ของการนำกลับมาใช้ใหม่ ซึ่งมีประสิทธิภาพสูงกว่า และเชื่อถือได้มากกว่ารูปแบบเดิม

การพัฒนาโดยสถาปัตยกรรม COM เป็นพื้นฐาน (COM based Architecture) เป็นอีกแนวทางหนึ่ง ซึ่งได้ถูกนำมาใช้ โดยมีจุดเด่น ดังนี้

- มีความเป็นอิสระในการพัฒนา ไม่ขึ้นกับเครื่องพัฒนาใด ๆ เช่น COM ออบเจ็กต์ที่ถูกสร้างจาก Visual C++ ก็สามารถเรียกใช้โดยโปรแกรมที่พัฒนาโดย Visual Basic ได้
- มีความสมบูรณ์ในตัวเอง ไม่ต้องคอมไพล์ใหม่เมื่อมีการเรียกใช้จากไคลเอนต์
- มีความสะดวกในการเรียกใช้ การเรียกใช้ไม่ว่าตัว COM ออบเจ็กต์นั้นจะอยู่ที่ไหนก็ไม่มีความแตกต่างกันมากนัก
- เป็นหลักการพื้นฐานของ OLE และ ActiveX
- เนื่องจากมีการใช้อินเทอร์เฟซเป็นตัวติดต่อกับไคลเอนต์ ซึ่งอินเทอร์เฟซนั้นจะไม่มีการเปลี่ยนแปลง
- การเปลี่ยนแปลงภายในตัวเซิร์ฟเวอร์จะไม่มีผลต่อตัวไคลเอนต์โดยตรง

เมื่อระบบงานนั้นต้องการที่จะเพิ่มความสามารถในการเชื่อมต่อกับอินเตอร์เน็ต ทางบริษัทไมโครซอฟท์ก็ได้เสนอสถาปัตยกรรมที่เป็นสภาพแวดล้อมที่เหมาะสมกับแพลตฟอร์มของไมโครซอฟท์ ซึ่งเรียกว่า Windows DNA องค์ประกอบหนึ่งที่สำคัญของ Windows DNA คือ สภาพแวดล้อม COM+ รันไทม์ หรือ MTS เวอร์ชัน 3.0 (Microsoft Transaction Server 3.0)นั่นเอง ซึ่งมีบริการใหม่ ๆ เพิ่มเติมจาก MTS เวอร์ชันก่อนหน้า รวมทั้งยังใช้งานง่ายยิ่งกว่าเวอร์ชันก่อนอีกด้วย

ปริญญานิพนธ์ฉบับนี้ได้นำเสนอทั้งทฤษฎี และตัวอย่างการสร้างทั้ง COM คอมโพเน็นต์ และการออกแบบระบบงานขนาดใหญ่ โดยใช้หลักการคอมโพเน็นต์ ซึ่งมีความละเอียดพอสมควร

1.2 วัตถุประสงค์ของโครงงาน

ภายในโครงงานนี้ได้ทำการศึกษาถึงทฤษฎี การสร้าง และการนำ COM ออบเจ็กต์ไปใช้งานโดยมีจุดประสงค์ดังนี้

1. เข้าใจถึงทฤษฎีพื้นฐานของ COM ลักษณะภายในตัว COM ออบเจ็กต์ และองค์ประกอบที่เกี่ยวข้อง

2. รู้วิธีการสร้าง COM ออบเจ็กต์ ทั้งแบบ in-process, out-of-process และแบบ remote server
3. เข้าใจถึงทฤษฎีพื้นฐานของสภาพแวดล้อม COM+ รันไทม์ และบริการต่างๆ ที่มาพร้อมกันกับ DNA
4. สามารถออกแบบระบบงานเชิงคอมโพเน็นต์ และสร้างระบบงานที่ออกแบบในลักษณะคอมโพเน็นต์ โดยใช้เทคโนโลยี COM+ ร่วมกับ Windows DNA
5. ศึกษาเทคโนโลยีคลัสเตอร์ (Cluster) ของไมโครซอฟท์ ที่ช่วยสร้างแอพพลิเคชั่นที่มีความสามารถในการกระจายโหลด (Scalability) และความน่าเชื่อถือในการใช้งาน (Reliability)
6. ศึกษาวิธีการเรียกใช้คอมโพเน็นต์ข้ามแพลตฟอร์มระหว่างแพลตฟอร์มของไมโครซอฟท์ (COM+) กับ ซัน (EJB)

1.3 ขอบเขตของโครงงาน

1. ออกแบบและสร้างแอพพลิเคชั่นโดยใช้สถาปัตยกรรมแบบ 3 เทียร์ส ร่วมกับการใช้หลักการพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์ด้วยเทคโนโลยี COM+
2. สร้างแอพพลิเคชั่นให้สามารถเรียกใช้คอมโพเน็นต์ข้ามแพลตฟอร์มได้
3. ใช้ทฤษฎีของ Network Load Balancing และ Component Load Balancing ในการเพิ่ม Scalability และ Reliability ให้กับแอพพลิเคชั่น

1.4 วิธีการดำเนินงานของโครงงาน

1. ศึกษาทฤษฎีพื้นฐานของ COM และ COM+
2. ศึกษาถึงโครงสร้างทางสถาปัตยกรรมของ Windows DNA
3. ศึกษาทฤษฎีของ XML และ SOAP ในการเรียกใช้งานคอมโพเน็นต์ระหว่างแพลตฟอร์มของไมโครซอฟท์กับแพลตฟอร์มของซันไมโครซิสเต็มส์
4. ออกแบบระบบแอพพลิเคชั่นในลักษณะงานเชิงคอมโพเน็นต์
5. ศึกษาการทำงานของ Network Load Balancing (NLB) ที่มีอยู่ใน Windows 2000 Advanced Server
6. ศึกษาการปรับแต่งค่า NLB เพื่อให้เครื่องเซิร์ฟเวอร์หลายเครื่องสามารถทำงานร่วมกันได้
7. ศึกษาการทำงานของ Component Load Balancing (CLB)
8. ศึกษาวิธีการใช้งานโปรแกรม Microsoft Application Center 2000 เพื่อให้สามารถทำ Component Load Balancing ได้

บทที่ 2
ทฤษฎีคอมโพเน็นต์

2.1 การพัฒนาเชิงคอมโพเน็นต์

คอมโพเน็นต์ คือ ส่วนย่อยของระบบที่ไม่ขึ้นอยู่กับส่วนอื่นๆ และถูกซ่อนรายละเอียดไว้ภายใน ซึ่งหน้าที่ของคอมโพเน็นต์จะถูกกำหนดไว้อย่างเหมาะสมเท่าที่ไคลเอ็นต์ต้องการใช้งานเท่านั้น ส่วนการจำกัดหน้าที่บางอย่างของคอมโพเน็นต์นั้นจะกระทำโดยสถาปัตยกรรม (Architecture) หรือ คอนเทนเนอร์ (container) ที่คอมโพเน็นต์นั้นดีพลอย (deploy) อยู่

คอมโพเน็นต์อาจพัฒนา ทดสอบ และดีพลอยในสภาพแวดล้อมที่แยกจากส่วนอื่นๆ ของระบบ ถึงแม้ว่าในความเป็นจริงนั้น คอมโพเน็นต์จะถูกใช้งานร่วมกับคอมโพเน็นต์อื่นๆ ก็ตาม แต่การทำแบบนี้มีความสำคัญต่อการทดสอบและคุณภาพของคอมโพเน็นต์

เราอาจพิจารณาว่าส่วนใดๆ ของระบบซอฟต์แวร์เป็นคอมโพเน็นต์ได้ เช่น ออบเจ็กต์ เมธอด เพราะระบบถูกสร้างขึ้นจากการประกอบของส่วนต่างๆ นี้ แต่การพิจารณาแบบนี้ไม่เหมาะกับการพัฒนาซอฟต์แวร์เชิงคอมโพเน็นต์ เพราะหมายความว่าผู้พัฒนาจะต้องเกี่ยวข้องกับคอมโพเน็นต์เสมอ

เพื่อความหมายที่สมบูรณ์ของคอมโพเน็นต์จะต้องพูดถึงอินเทอร์เฟซ เพราะไคลเอ็นต์จะใช้อินเทอร์เฟซในการทำงานกับคอมโพเน็นต์

การพัฒนาเชิงคอมโพเน็นต์นั้น มีความหมายกับคนแต่ละกลุ่มแตกต่างกันไปตามความต้องการ ซึ่งมีลักษณะเด่นบางอย่างที่นิยามความเป็นระบบคอมโพเน็นต์ดังต่อไปนี้

1. คอมโพเน็นต์มีลักษณะของออบเจ็กต์ คือ การซ่อนรายละเอียด (Encapsulated), โพลีมอร์ฟิค (polymorphic), การกำหนดหน้า, การกำหนดอินเทอร์เฟซ

2. คอมโพเน็นต์ออกแบบภายใต้เฟรมเวิร์ก (Framework) ซึ่งได้สร้างข้อจำกัดบางอย่างไว้ เช่น ต้องไม่มีหลายเธรด, ไม่มีการติดต่อกับภายนอกโดยไม่ผ่านบริการของเฟรมเวิร์ก

3. คอมโพเน็นต์สามารถอยู่ได้โดยไม่พึ่งพาคอมโพเน็นต์อื่นๆ ยกเว้นคอมโพเน็นต์ของเฟรมเวิร์กที่คอมโพเน็นต์ดังกล่าวใช้อยู่

4. ทุกคอมโพเน็นต์มีอินเทอร์เฟซสามัญที่แน่นอน (Fix and common) และอินเทอร์เฟซนี้เปลี่ยนแปลงไม่ได้

5. คอมโพเน็นต์สามารถอธิบายตนเองได้ โดยอินเทอร์เฟซของคอมโพเน็นต์จะต้องมีข้อมูลมากพอที่สามารถทำให้ไคลเอ็นต์สามารถเข้าใจวิธีใช้คอมโพเน็นต์นั้นได้

2.1.1 เฟรมเวิร์ก คือ สภาพแวดล้อมที่มีลักษณะดังนี้

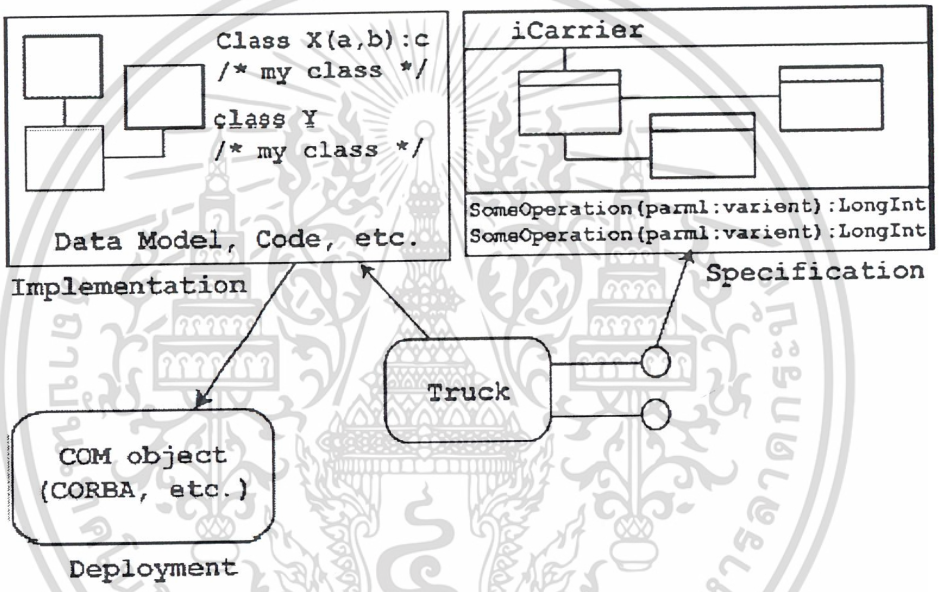
1. สร้างอินสแทนซ์ของคอมโพเน็นต์ตอนรันไทม์
2. ทำให้คอมโพเน็นต์ค้นพบคอมโพเน็นต์อื่นๆ ได้
3. ทำให้คอมโพเน็นต์ต่างๆ สามารถติดต่อกันได้

4. จัดหาบริการสามัญ เช่น เพอร์ซิสเทนซ์ (persistence), ทรานส์แอ็กชัน (transaction), ความไม่ขึ้นกับสถานที่, การรักษาความปลอดภัย, มอนิเตอริง (monitoring)

2.1.2 ส่วนประกอบของคอมโพเน็นต์

คอมโพเน็นต์จะประกอบด้วยส่วนสำคัญ 3 ส่วนดังนี้

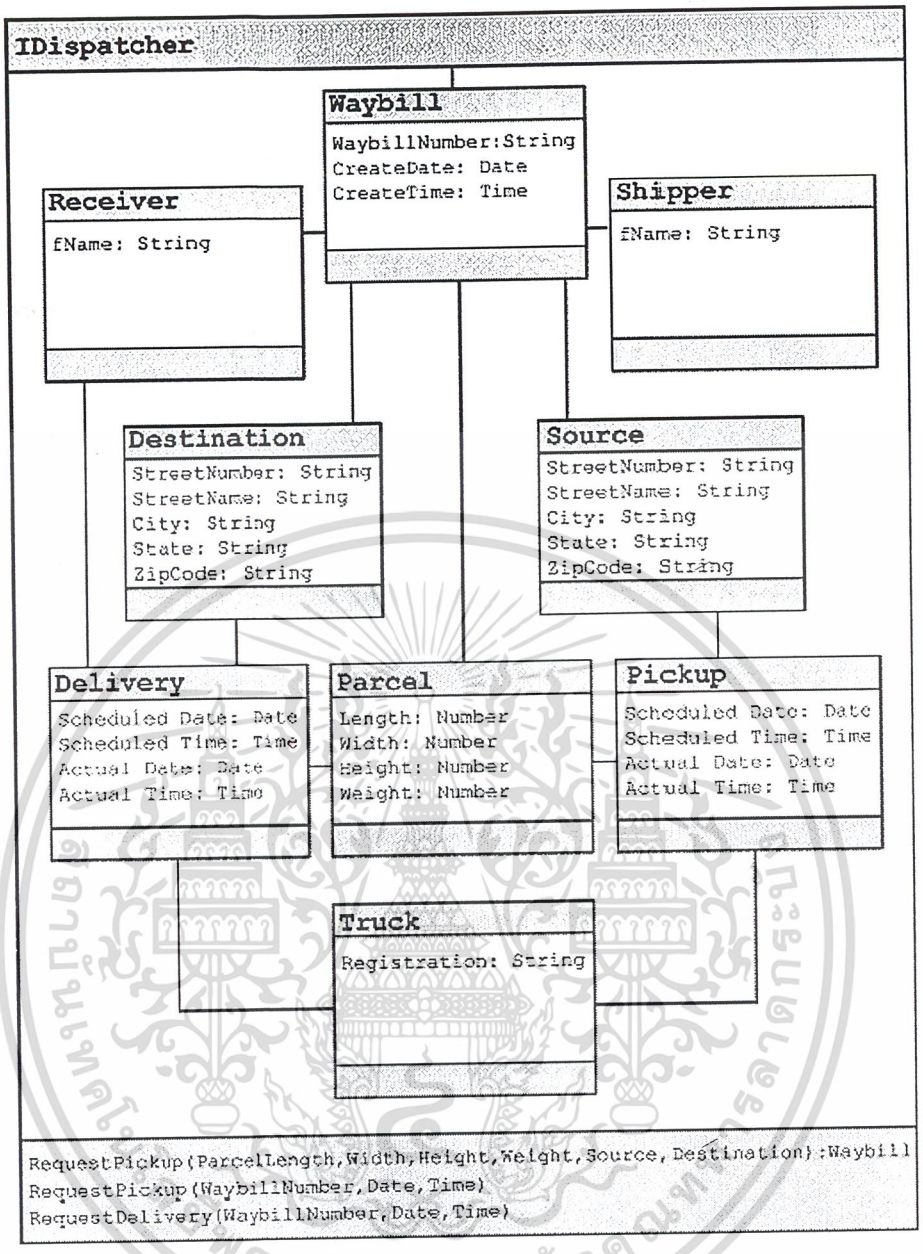
- อินเทอร์เฟซ (interface) - คอมโพเน็นต์จะถูกเรียกใช้งานผ่านทางอินเทอร์เฟซ
- อิมพลีเมนต์เตชัน (implementation) - เป็นโค้ดที่กำหนดการทำงานของคอมโพเน็นต์
- ดีพลอยเมนต์ (deployment) – เป็นเอ็กซิคิวต์ไฟล์จะใช้ในการทำให้คอมโพเน็นต์ทำงานได้ ทำหน้าที่จัดหารันไทม์เอ็นไวรอนเมนต์ในการควบคุมการทำงานของคอมโพเน็นต์และจัดหาเซอร์วิสที่จำเป็น



รูปที่ 2-1 คอมโพเน็นต์และส่วนประกอบต่าง ๆ ของมัน

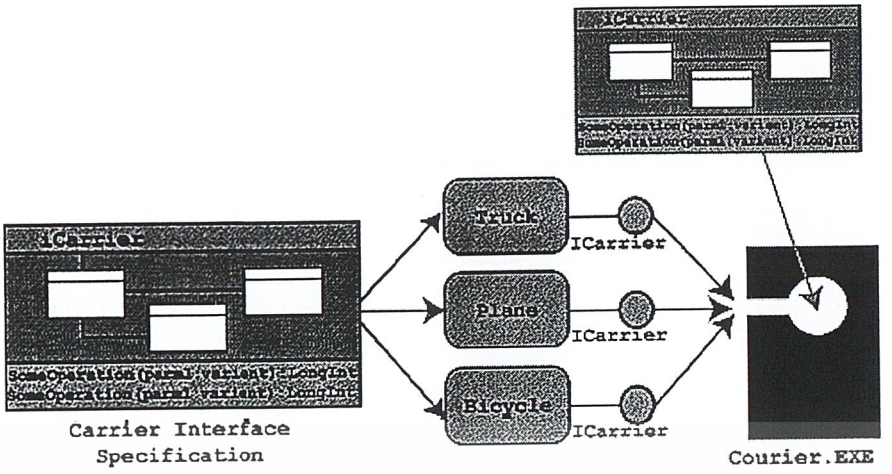
2.1.3 คุณสมบัติของคอมโพเน็นต์

- **Encapsulated** - เป็นกระบวนการในการซ่อนโค้ดหรืออิมพลีเมนต์เตชันของคอมโพเน็นต์ ผู้ใช้จะรู้เพียง อินเทอร์เฟซและจะใช้งานผ่านมัน โดยไม่จำเป็นต้องรู้ถึงอิมพลีเมนต์เตชัน ทำให้การเปลี่ยนแปลงอิมพลีเมนต์เตชันไม่มีผลกระทบต่อผู้ใช้
- **Descriptive** - เนื่องจากคอมโพเน็นต์จะต้องติดต่อผ่านอินเทอร์เฟซเท่านั้น มันจะต้องมีอินฟอร์เมชันของมันเองที่ผู้ใช้สามารถจะเข้าใจได้ โดยอินฟอร์เมชันนั้นจะต้องอธิบายถึงอินเทอร์เฟซ , อิมพลีเมนต์เตชัน และดีพลอยเมนต์



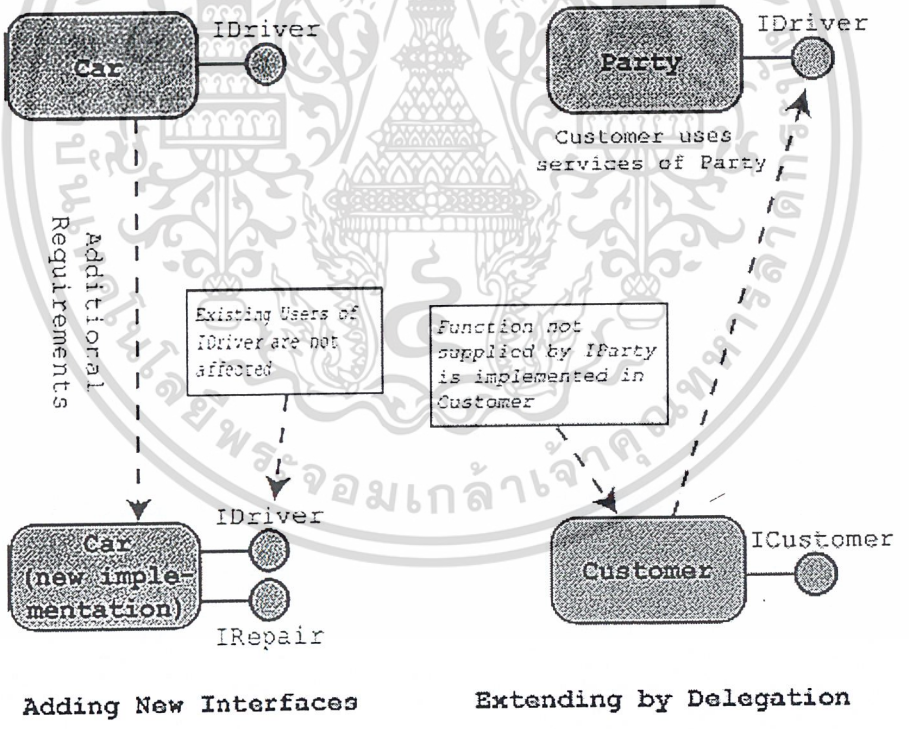
รูปที่ 2-2 ตัวอย่างของอินเทอร์เฟซ

- **Replaceable** - คอมโพเน็นต์มีความสามารถในการเปลี่ยนคอมโพเน็นต์หนึ่งกับคอมโพเน็นต์ใด ๆ ที่มีอินเทอร์เฟซเหมือนกัน



รูปที่ 2-3 คอมโพเน็นต์และการแทนที่

- **Extensible** - สามารถที่จะเพิ่มความสามารถได้ มี 2 วิธีคือ การเพิ่มอินเทอร์เฟซและมอบหมายหน้าที่ (Delegating Responsibility) การสร้างคอมโพเน็นต์ใหม่สามารถมอบหมายหน้าที่ให้กับบริการที่มีอยู่ในคอมโพเน็นต์ที่มีอยู่แล้ว



รูปที่ 2-4 การเพิ่มเติมคอมโพเน็นต์

2.2 งานที่เหมาะสมกับการพัฒนาเชิง คอมโพเน็นต์

การพัฒนาเชิงคอมโพเน็นต์ เป็นวิธีการพัฒนาคอมโพเน็นต์โดยใช้คอมโพเน็นต์เฟรมเวิร์ก ซึ่งการคงหน้าที่ของคอมโพเน็นต์ไว้อย่างถูกต้อง (Well defined responsibility) กระทำได้โดยให้คอมโพเน็นต์เชิงธุรกิจ หลีกเลี่ยงโค้ดที่ไม่สามารถทำงานกับอินเทอร์เฟซที่แน่นอนได้

เฟรมเวิร์กคอมโพเน็นต์จะซ่อนบริการต่างๆ ของเอ็นไวรอนเม็นต์ ซึ่งบริการต่างๆ เหล่านี้เป็นมาตรฐานแล้ว เช่น message oriented middle-ware, Transaction Monitors, การรักษาความปลอดภัย และพิสูจน์สิทธิ์

2.3 ประโยชน์ของการพัฒนาเชิงคอมโพเน็นต์

การพัฒนาเชิงคอมโพเน็นต์มีประโยชน์อย่างมากด้านเกี่ยวกับเทคนิค และประโยชน์เหล่านี้ยังนำไปสู่ประโยชน์ด้านธุรกิจด้วย

ประโยชน์ด้านเทคนิค

1. จัดการความซับซ้อนได้ดีขึ้น ทำให้คุณภาพของการแก้ปัญหาดีขึ้น
2. งานซับซ้อนที่ไม่เกี่ยวกับฟังก์ชันธุรกิจ ถูกรวมไว้ในเฟรมเวิร์ก
3. การออกแบบ, อิมพลีเมนต์, ทดสอบ ส่วนต่างๆอย่างอิสระ ทำให้สามารถพัฒนาส่วนต่างๆ ได้พร้อมกัน
4. การไม่ขึ้นต่อกันของคอมโพเน็นต์ลดผลกระทบของการเปลี่ยนแปลง requirement ไม่ให้กระจายไปทั้งระบบ

ประโยชน์ด้านธุรกิจ

1. ผลิตภัณฑ์มีคุณภาพมากขึ้น
2. เร่งเวลาออกสู่ตลาด
3. ใช้ทรัพยากรบุคคลได้คุ้มค่าขึ้น
4. ลดค่าใช้จ่าย
5. เพิ่มปริมาณการนำกลับมาใช้ ในโครงการต่อไป

2.4 แนวโน้มของระบบในอนาคต

- ความต้องการระบบที่ซับซ้อนยิ่งขึ้น
- ความต้องการให้ระบบติดต่อกันจากธุรกิจสู่ธุรกิจ (B2B)
- ความต้องการให้ระบบ โดยด้านเครือข่าย ดีขึ้น

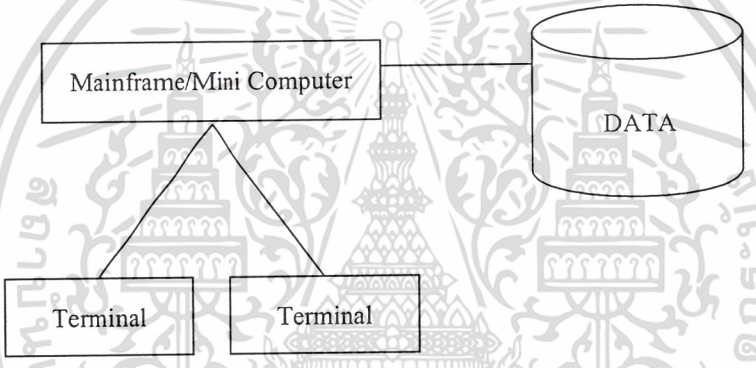
คอมโพเน็นต์สามารถเติมเต็มความต้องการเหล่านี้ได้ เพราะระบบสามารถสร้างจากระบบย่อยที่อิสระต่อกัน ทำให้ระบบรวมมีความแข็งแกร่งขึ้น เพราะชิ้นส่วนย่อยหนึ่งไม่ยึดติดกับความยุ่งยากภายในชิ้นส่วนย่อยอื่น

บทที่ 3
วิวัฒนาการของสถาปัตยกรรมแบบกระจาย

3.1 สถาปัตยกรรมแบบหนึ่งเทียร์ (One Tier Architecture/Centralize Host)

สถาปัตยกรรมแบบหนึ่งเทียร์ คือระบบผู้ใช้หลายคนที่เรียบง่ายที่สุด โดยจะมีการเก็บข้อมูลและประมวลผลอยู่ที่คอมพิวเตอร์ส่วนกลางทั้งหมด (เช่น เมนเฟรมคอมพิวเตอร์หรือมินิคอมพิวเตอร์) ผู้ใช้จะติดต่อกับคอมพิวเตอร์ส่วนกลางโดยผ่านทางเทอร์มินอล ซึ่งเป็นเพียงแค่อุปกรณ์อินพุตและเอาต์พุตเพียงอย่างเดียว ไม่มีการประมวลผล ในระบบเช่นนี้มีข้อเสียคือ

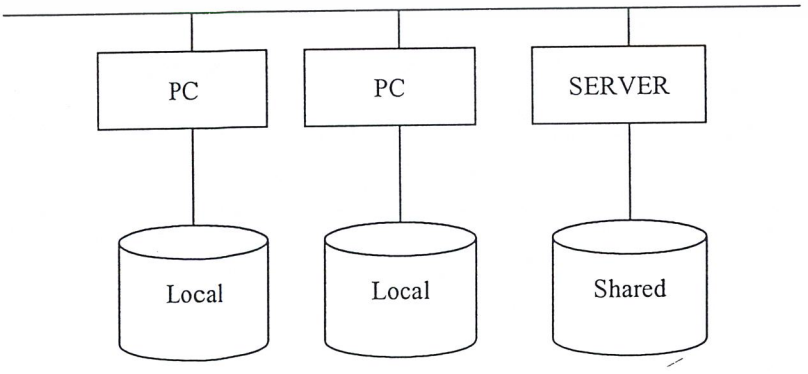
- การที่ทุกๆ กระบวนการทำงานทำในคอมพิวเตอร์ส่วนกลางทั้งหมด ทำให้ต้องแบ่งกำลังการประมวลผลไปให้ผู้ใช้แต่ละคน อาจมีผลกระทบต่อเวลาการตอบสนอง ทำให้ตอบสนองได้ช้าลง
- ถ้าเกิดความขัดข้องที่คอมพิวเตอร์ส่วนกลาง ผู้ใช้ทุกคนจะไม่สามารถใช้งานได้
- ค่าใช้จ่ายในการบำรุงรักษาทางด้านฮาร์ดแวร์ และซอฟต์แวร์สูงมาก



รูปที่ 3-1 สถาปัตยกรรมแบบหนึ่งเทียร์

3.2 เครือข่ายภายในและการแบ่งปันทรัพยากรภายในเครือข่าย (PC LANs/ทรัพยากร Sharing LANs)

คอมพิวเตอร์ส่วนบุคคล (Personal Computer: PC) ได้ถือกำเนิดขึ้นมาเพื่อทดแทนความไม่ยืดหยุ่นของระบบเมนเฟรม พีซีนั้นมีค่าใช้จ่ายทางด้านซอฟต์แวร์และฮาร์ดแวร์ถูกกว่าระบบเมนเฟรมมาก พีซีในยุคแรกเริ่มเป็น พีซีที่มีการใช้งานอยู่เครื่องเดียว ในก้าวต่อมาจึงมีการเชื่อมต่อพีซีเข้าด้วยกันผ่านเครือข่ายภายใน (Local Area Network: LAN)



รูปที่ 3-2 เครือข่ายภายในของเครื่องคอมพิวเตอร์ส่วนบุคคล

ในส่วนที่เป็นเซิร์ฟเวอร์นั้น อาจเป็นเซิร์ฟเวอร์ที่ให้บริการแฟ้มข้อมูล หรือเซิร์ฟเวอร์ที่ให้บริการเครื่องพิมพ์ (File/Print Server) ก็ได้ แต่แอพพลิเคชันจะยังคงทำงานอยู่บนพีซีของตัวเอง การที่ข้อมูลอยู่บนดิสก์ที่ใช้ร่วมกันที่เซิร์ฟเวอร์ และแอพพลิเคชันทำงานอยู่ที่พีซี ทำให้เกิดการถ่ายโอนข้อมูลจำนวนมากและเป็นไปอย่างต่อเนื่อง ยกตัวอย่างเช่น การคิวรี่ข้อมูลจากข้อมูลทั้งหมด 10,000 เรคคอร์ด เซิร์ฟเวอร์จะต้องส่งข้อมูลทั้ง 10,000 เรคคอร์ดนั้นมาประมวลผลคำสั่ง SQL ที่เครื่องไคลเอนต์

3.3 สถาปัตยกรรมแบบสองเทียร์ส (ไคลเอนต์/เซิร์ฟเวอร์)

3.3.1 ดาต้าเบสเซิร์ฟเวอร์ (Database Server)

ในระบบดาต้าเบสเซิร์ฟเวอร์นั้นดาต้าและดาต้าเบสเอ็นจิ้น (DBMS) จะรันอยู่ที่เซิร์ฟเวอร์ ส่วนหน้าจอใช้งานจะรันอยู่บนพีซี ซึ่งเรียกว่าไคลเอ็นต์ ในการสืบค้นข้อมูลนั้น จะส่งคำสั่ง SQL statement ไปสืบค้นที่เซิร์ฟเวอร์และเซิร์ฟเวอร์จะส่งเฉพาะผลลัพธ์กลับไปที่ไคลเอ็นต์ ซึ่งจะช่วยลดความหนาแน่นของข้อมูลในระบบเครือข่ายลงได้ ไคลเอ็นต์กับเซิร์ฟเวอร์จะเป็นอิสระต่อกัน แต่จะมีแอพพลิเคชั่นทำงานร่วมกันอยู่ แอพพลิเคชั่นฝั่งเซิร์ฟเวอร์คือ ดาต้าเบสเอ็นจิ้น ส่วนแอพพลิเคชั่นฝั่งไคลเอ็นต์ ก็คือแอพพลิเคชั่นต่างๆ ที่ต้องการติดต่อกับดาต้าเบส แอพพลิเคชั่นทั้งสองส่วนจะติดต่อสื่อสารกันผ่านเน็ตเวิร์กโพรโตคอล ซึ่งเป็นความรับผิดชอบของ DBMS โดย DBMS จะจัดเตรียมโครงสร้างพื้นฐานทั้งหมดให้

ข้อเสียของไคลเอ็นต์/เซิร์ฟเวอร์ ก็คือการที่แอพพลิเคชั่นนั้นรันอยู่บนไคลเอ็นต์ การแก้ไขแอพพลิเคชั่นจึงต้องทำทุกๆไคลเอ็นต์ ทำให้มีค่าใช้จ่ายในการบำรุงรักษาสูงมาก

3.3.2 แอพพลิเคชั่นเซิร์ฟเวอร์

แอพพลิเคชั่นเซิร์ฟเวอร์จะมีความสามารถมากกว่าการแชร์ไฟล์หรือเครื่องพิมพ์ในเครือข่ายภายในหรือการจัดการดาต้าเบสในดาต้าเบสเซิร์ฟเวอร์ แอพพลิเคชั่นเซิร์ฟเวอร์นั้นสามารถใส่แอพพลิเคชั่นที่มีฟังก์ชั่นการทำงานต่างๆ ไว้ และเรียกใช้เมื่อต้องการได้ โดยแอพพลิเคชั่นที่จะเรียกใช้ฟังก์ชั่นที่อยู่บนเซิร์ฟเวอร์จะต้องใช้เน็ตเวิร์กโพรโตคอล เช่น TCP/IP, SPX/IPX หรือโพรโตคอลในระดับสูง เช่น DCOM, COBRA, RMI

3.4 สถาปัตยกรรมแบบสามเทียร์ส

ในระบบสถาปัตยกรรมแบบสองเทียร์สนั้น ทุกๆ ไคลเอ็นต์ติดต่อโดยตรงกับดาต้าเบส ซึ่งการทำเช่นนี้เหมาะสมกับงานขนาดเล็ก แต่ไม่เหมาะสมกับองค์กรใหญ่ๆ ที่มีผู้ใช้จำนวนมาก โดยเฉพาะอย่างยิ่งในระบบอินเตอร์เน็ตจะมีผู้ใช้ที่ใช้งานพร้อมกันเป็นจำนวนมากในช่วงเวลาที่การใช้งานมากๆ (peak time) จะทำให้เกิดปัญหาการขาดแคลนทรัพยากรได้ ด้วยเหตุนี้ เทียร์ที่สามคือแอพพลิเคชั่นเทียร์หรือบิสิเนสเทียร์ จึงเกิดขึ้นมาคั่นระหว่างพรีเซนเตชั่นเทียร์และดาต้าเบสเทียร์เพื่อเพิ่มความยืดหยุ่น และมีความสามารถในการกระจายโหลดได้มากขึ้น ยกตัวอย่างเช่นบิสิเนสเลเยอร์สามารถจัดการพูล (Pool) การเชื่อมต่อของดาต้าเบสได้ โดยจะไม่ยกเลิกการเชื่อมต่อกับดาต้าเบสเมื่อไม่ใช้งาน แต่จะเก็บลงพูลแทน ทำให้รองรับไคลเอ็นต์ได้มากขึ้น บิสิเนสเลเยอร์นั้นจะแยกไคลเอ็นต์ออกจากดาต้าเบส ไม่ให้ขึ้นต่อกันโดยตรง

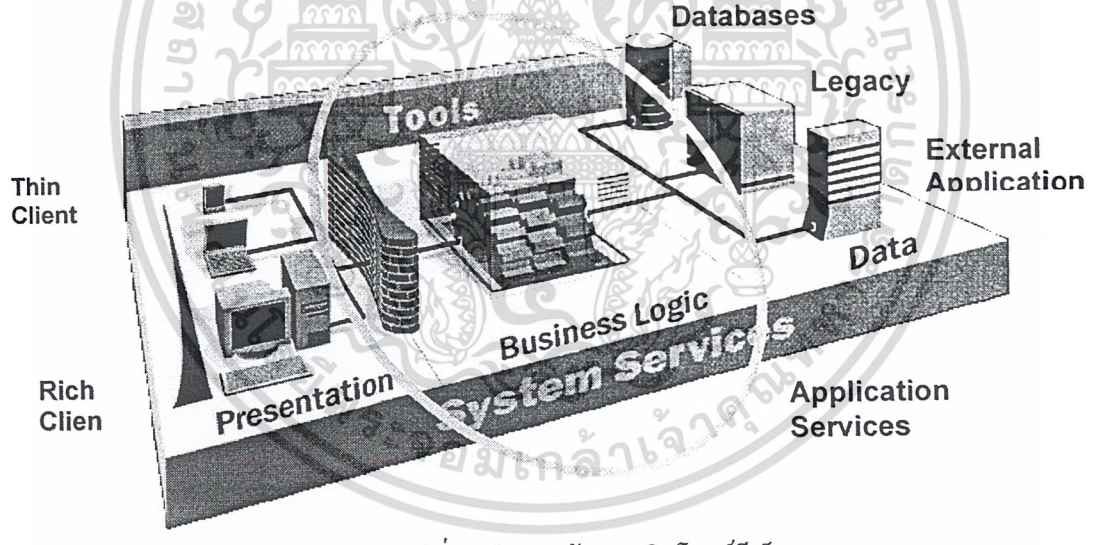
ดังนั้นดาต้าเบสสามารถเปลี่ยนแปลงได้ โดยไม่มีผลกระทบต่อโค้ดในพรีเซนเตชั่นเลเยอร์เลย บิสิเนสเลเยอร์ก็สามารถสร้าง Business Rule ที่เป็นสามัญซึ่งจะใช้ได้ในหลายๆ ไคลเอ็นต์แอพพลิเคชั่นทำให้เกิดการนำโค้ดต้นฉบับมาใช้ใหม่ ถ้าฟังก์ทางธุรกิจเปลี่ยนแปลงไป ก็ปรับปรุงเฉพาะบิสิเนสเลเยอร์เท่านั้น

3.4.1 วินโดวส์ดีเอ็นเอ (Windows DNA)

ไมโครซอฟต์สนับสนุนการกระจายแอพพลิเคชั่นด้วยสถาปัตยกรรมวินโดวส์ดีเอ็นเอ (Windows Distributed Internet Application Architecture: Windows DNA)

วินโดวส์ดีเอ็นเอนำเสนอแอพพลิเคชั่นแบบสามเทียร์ส โดยอยู่บนพื้นฐานของ Component Object Model (COM) จุดประสงค์ของวินโดวส์ดีเอ็นเอ คือ การแยกโลจิกทางธุรกิจออกจากระบบไคลเอ็นต์/เซิร์ฟเวอร์ไปสู่วินโดวส์เทียร์ซึ่งรันอยู่บนวินโดวส์ 2000

วินโดวส์ดีเอ็นเอไม่ใช่ผลิตภัณฑ์ของบริษัทใดๆ แต่เป็นแนวทางการพัฒนาแอพพลิเคชั่นแบบสามเทียร์สโดยทางไมโครซอฟต์ได้จัดเตรียมแพลตฟอร์ม คอมโพเน็นต์เทคโนโลยี โครงสร้างภายในแอพพลิเคชั่น บริการและเครื่องมือต่างๆ ให้ หัวใจของวินโดวส์ดีเอ็นเอนั้น คือการรวมโมเดลการเขียนโปรแกรมที่อยู่บนพื้นฐานของ COM เข้าด้วยกัน



รูปที่ 3-3 โครงสร้างของวินโดวส์ดีเอ็นเอ

3.4.1.1 พรีเซนเตชั่นเทียร์

Rich Client เป็น Win 32 แอพพลิเคชั่นซึ่งสามารถเข้าถึงทุก ๆ ทรัพยากรของคอมพิวเตอร์ที่เป็นไคลเอนต์ได้ Rich Client ติดต่อกับ Middle Tier โดยผ่านทาง DCOM

Thin Client จะติดต่อกับ Web Server ผ่านทาง HTTP Protocol เทคโนโลยีของไมโครซอฟต์สำหรับ Thin Client ก็คือ Web Browser (Internet Explorer)

3.4.1.2 บิสิเนสโลจิกเทียร์ (Business Logic Tier)

บิสิเนสลอจิกเลเยอร์หรือ Middle Tier นั้น เป็นส่วนที่สำคัญที่สุดของสถาปัตยกรรมแบบสามเทียร์ บริการใหม่ ๆ ของไมโครซอฟต์จะอยู่ที่เลเยอร์นี้ เช่น

- Web Service โดยผ่านทาง Internet Information Server (IIS)
- Transaction และ Component Service ผ่านทาง Microsoft Transaction Server (MTS)
- Queuing และ Asynchronous Service ผ่านทาง Microsoft Message Queue (MSMQ)
- Server Side Scripting ผ่านทาง Active Server Page (ASP) ซึ่งอยู่บน IIS

3.4.1.3 ดาต้าแอ็กเซสเลเยอร์ (Data Access Layer)

ในดาต้าแอ็กเซสเลเยอร์นี้ ไมโครซอฟต์มีผลิตภัณฑ์ที่เป็นตัวจัดการฐานข้อมูลคือ Microsoft SQL Server 2000 แต่ก็ยังกำหนด Data Access Interface ซึ่งสนับสนุนผลิตภัณฑ์อื่น ๆ ด้วย เช่น Open Database Connectivity (ODBC), OLE DB

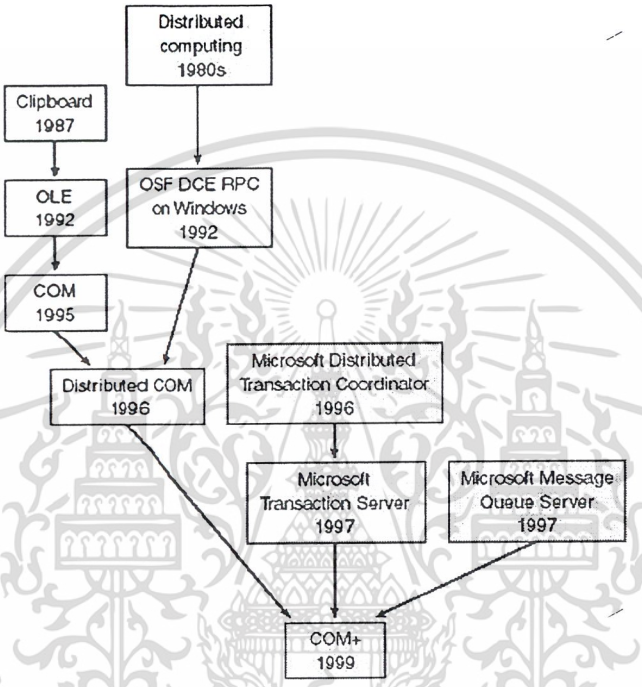
คุณสมบัติที่สำคัญที่สุดของ Data Access Technology ของไมโครซอฟต์ ก็คือ ความสามารถในการเข้าถึงทุก ๆ แหล่งข้อมูลได้อย่างสะดวกโดยผ่าน Interface เดียวกัน ซึ่งเรียกว่า Uniform Data Access (UDA)

บทที่ 4

Component Object Model (COM)

4.1 วิวัฒนาการของ COM+

COM+ ไม่ใช่เทคโนโลยีที่สร้างขึ้นมาเพียงชั่วข้ามคืน แต่เป็นการรวมเอาเทคโนโลยีจากหลายเส้นทางมารวมกัน ซึ่งสามารถสรุปได้ดังรูป



รูปที่ 4-1 วิวัฒนาการของ COM+

4.1.1 จาก OLE สู่ COM+

ระบบปฏิบัติการเอ็มเอสดอสของไมโครซอฟท์ถูกพัฒนาขึ้นบนคอมพิวเตอร์ที่มีประสิทธิภาพต่ำ (เทียบกับมาตรฐานในปัจจุบัน) จากนั้นอินเทลได้พัฒนาประสิทธิภาพของซีพียูอย่างมาก ทำให้ไมโครซอฟท์มองเห็นความเป็นไปได้ในการทำงานพร้อมๆ กันของหลายแอพพลิเคชัน

ด้วยการกำเนิดของมัลติทาสกิ้งของวินโดวส์ ทำให้คลิปบอร์ด (Clipboard) และ Dynamic Data Exchange (DDE) เป็นจุดเริ่มต้นของการสื่อสารระหว่างโพรเซส โดย DDE มีพื้นฐานการทำงานแบบส่งข้อความระหว่างแอพพลิเคชัน แต่โครงสร้างของ DDE ก็ยังซับซ้อนยากที่จะใช้งานได้ง่าย

คลิปบอร์ดมีการใช้งานที่ง่ายได้แก่ การตัด คัดลอก และแปะ ทำให้สามารถสร้างเอกสารที่ประกอบไปด้วยเนื้อหาหลายรูปแบบ เช่น รูปภาพ เสียง ตัวอย่างการใช้งานคลิปบอร์ด เช่น การสร้างรายงานที่เป็นเวิร์ดก็สามารถแทรกสเปรดชีตจากเอ็กเซลได้

ด้วยความง่ายในการอิมพลีเมนต์ของนักพัฒนาและง่ายต่อการใช้งาน ทำให้คลิปบอร์ดเป็นที่นิยมจนถึงปัจจุบัน แต่ปัญหาก็คือการแก้ไขคลิปบอร์ดจะทำไม่ได้ เช่น ถ้าต้องการแก้ไขคลิปบอร์ดก็ต้องทำขั้นตอนการคัดลอกและแปะซ้ำอีก

แนวคิดของ object linking and embedding (OLE) เกิดขึ้นเพื่อแก้ปัญหานี้ ซึ่ง OLE เปิดตัวครั้งแรกกับวินโดวส์ 3.1 โดยได้เพิ่มความสามารถในการนำเอกสารมาประกอบกัน เช่น การเชื่อมเอ็กซ์เซลสเปรดชีตในเอกสารเวิร์ด เมื่อแก้ไขสเปรดชีต ข้อมูลใหม่ก็จะเชื่อมโยงไปยังเอกสารเวิร์ดโดยอัตโนมัติ โดยอันที่จริงแล้วก็ใช้ DDE ทำงานอยู่ภายใน อย่างไรก็ตาม OLE ก็ได้จุดประกายแนวคิดของคอมโพเน็นต์ที่สามารถเสียบเข้ากับแอพพลิเคชันอื่นได้

OLE2 เปิดตัวในปี 1993 โดยเพิ่มความสามารถโดย in-place activation (visual editing) ทำให้สามารถแก้ไขเอกสารที่ซ้อนอยู่ภายในโดยที่ไม่ต้องละจากหน้าต่างเดิม แต่นักพัฒนาก็ยังมองไม่เห็นความสำคัญว่า OLE2 เป็นรากฐานของคอมโพเน็นต์ หลังจากนั้นกว่าปี OLE ก็ได้หลบฉากและปล่อยให้ COM เข้ามาแสดงบทบาทแทน

ซอฟต์แวร์ขนาดใหญ่ประสบปัญหาในการดูแลโค้ดขนาดใหญ่ที่อาจจะมากถึง 1 ล้านบรรทัด ซึ่งเป็นเรื่องยากที่จะแก้ไขและทดสอบทั้งระบบ ในบางครั้งการแก้ไขเล็กๆ น้อยๆ อาจทำลายระบบในจุดอื่นๆ ได้

COM เปรียบเสมือนกาวเชื่อมคอมโพเน็นต์ต่างๆเข้าด้วยกัน ทำให้ซอฟต์แวร์ที่ไม่มีความเกี่ยวข้องกันสามารถทำงานร่วมกันได้ ซึ่งคอมโพเน็นต์ต่างๆ สามารถนำกลับมาใช้ใหม่ในจุดที่แตกต่างกันได้ เช่น คอมโพเน็นต์ทางธุรกิจที่ใช้ในเดสก์ทอปแอพพลิเคชันสามารถนำไปใช้กับ Microsoft Internet Information Server (IIS) ได้

การพัฒนาCOMของไมโครซอฟต์มีความตั้งใจที่จะแก้ปัญหาของอุตสาหกรรมซอฟต์แวร์ซึ่งมีความซับซ้อนมากขึ้นทำให้เกิดปัญหาตามมา ดังนี้

- ซอฟต์แวร์สมัยใหม่มีขนาดใหญ่และซับซ้อน ทำให้ใช้เวลานานในการพัฒนา ยากต่อการดูแล และมีความเสี่ยงในการที่จะขยายความสามารถเพิ่มเติมในภายหลังให้กับซอฟต์แวร์นั้น
- แอพพลิเคชันยังคงพัฒนาในแบบเดี่ยว ซึ่งจะมาเป็นแพ็กเกจไม่มีทางเลือกในการเพิ่ม ลบ และอัพเกรดได้
- แอพพลิเคชันไม่ยอมให้มีการรวมตัวกับแอพพลิเคชันอื่นๆ ไม่ว่าจะเป็นข้อมูลหรือฟังก์ชัน

ด้วยความซับซ้อนของปัญหาทั้งหมด ทำให้ไม่มีเทคโนโลยีเดี่ยวใดๆ สามารถแก้ปัญหาได้ทั้งหมด แต่ซอฟต์แวร์ที่ปฏิบัติตามแนวทางของ COM สามารถแก้ปัญหาข้างต้นได้ดีขึ้น

COM+ เป็นวิวัฒนาการจาก COM และไมโครซอฟต์ทรานส์แอ็กชันเซิร์ฟเวอร์ (MTS) โดยนำทั้ง 2 เทคโนโลยีมารวมตัวกัน

4.1.2 RPC และ COM+

ในอดีตการติดต่อกันระหว่างคอมพิวเตอร์ผ่าน LAN เป็นความสำเร็จอันยิ่งใหญ่ สำหรับพีซีแล้ว การที่จะได้รับการยอมรับในการใช้งานแอพพลิเคชันที่สำคัญจะต้องรองรับการทำงานแบบกระจายที่ทำงานได้เทียบเท่ากับการทำงานแบบรวมศูนย์ ซึ่งจะต้องเกิดจากความร่วมมือของกลุ่มอุตสาหกรรมในการสร้างมาตรฐานร่วมกัน

ช่วงปลายทศวรรษ 1980 มีการสร้างมาตรฐานขึ้นโดยกลุ่ม Open Software Foundation (OSF) ได้สร้าง Distributed Computing Environment (DCE) เพื่อที่จะสร้างระบบการทำงานแบบกระจายที่มีความสามารถทัดเทียมกับการทำงานแบบรวมศูนย์

ผลลัพธ์ที่ได้จากDCEคือมาตรฐานในการติดต่อสื่อสารระหว่างคอมพิวเตอร์รู้จักกันในชื่อ Remote Procedure Calls (RPCs) ซึ่ง COM+ ใช้ RPCs สำหรับการติดต่อกันระหว่างคอมพิวเตอร์

4.2 COM คอมโพเน็นต์

COM คือสถาปัตยกรรมซอฟต์แวร์คอมโพเน็นต์ ที่ทำให้สามารถสร้างแอพพลิเคชั่นหรือระบบได้ จากคอมโพเน็นต์ต่างๆ ทั้งที่สร้างขึ้นมาหรือซื้อมาจากผู้ขายซอฟต์แวร์ต่าง ๆ

COM สามารถแตกออกไปได้หลายอย่าง เช่น คอมพาวด์ด็อกคิวเมนต์ (Compound Document), คัสทอมคอนโทรล (Custom Control), ดาต้าทรานส์เฟอร์ (Data Transfer) ซึ่งใช้กลไกพื้นฐานร่วมกันที่สามารถทำให้ไบนารีซอฟต์แวร์คอมโพเน็นต์จากหลายๆ ผู้ขายซอฟต์แวร์ สามารถทำงานร่วมกันได้ กลไกดังกล่าวได้แก่

- นิยามมาตรฐานไบนารี ที่ทำให้ คอมโพเน็นต์ ต่าง ๆ ทำงานร่วมกันได้
- เป็นอิสระต่อภาษาโปรแกรม (Programming Language-independent)
- มีหลายแพลตฟอร์ม(Microsoft Windows NT, Apple, Macintosh, Unix)
- ความสามารถในการขยายความสามารถ

นอกจากนี้ COM ยังมีความสามารถเพิ่มเติม คือ

- การติดต่อระหว่างคอมโพเน็นต์ แม้จะข้ามโพรเซสหรือเครือข่าย
- การจัดการแบ่งปันหน่วยความจำระหว่าง คอมโพเน็นต์
- รายงานสถานะและข้อผิดพลาด
- ไดนามิกโหลดดิง (Dynamic Loading) ของคอมโพเน็นต์

4.3 ปัญหาของคอมโพเน็นต์ซอฟต์แวร์

ปัญหาที่สำคัญที่สุด คือ ทำอย่างไรถึงจะสร้างระบบที่ใช้คอมโพเน็นต์ที่มาจากต่างที่มาและต่างเวลาได้

- *การทำงานร่วมกันได้ระดับพื้นฐาน*
  ทำอย่างไรให้ผู้พัฒนาคอมโพเน็นต์มั่นใจได้ว่า คอมโพเน็นต์ที่สร้างขึ้นมาสามารถทำงานร่วมกับคอมโพเน็นต์ ของผู้พัฒนาอื่นๆได้

- *การจัดการเวอร์ชัน*
  ทำอย่างไรจะอัพเกรดคอมโพเน็นต์ โดยไม่ต้องอัพเกรดทั้งระบบ
- *ความอิสระจากภาษา*
  ทำอย่างไรให้คอมโพเน็นต์ ที่สร้างจากต่างภาษาสามารถทำงานร่วมกันได้
- *Transparent Cross-Process Interoperability*
  ทำอย่างไรให้นักพัฒนาเขียนคอมโพเน็นต์ in-process หรือ Cross-process โดยใช้รูปแบบเดียว

4.4 พื้นฐานของ COM

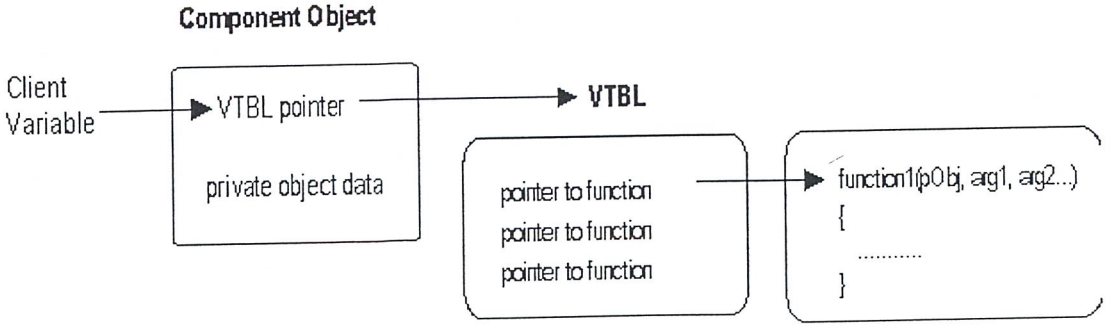
COM ได้กำหนดหลักเกณฑ์ต่างๆ ได้แก่

- มาตรฐานไบนารี (Binary Standard) เพื่อให้สามารถเรียกฟังก์ชันระหว่างคอมโพเน็นต์ได้
- วิธีการในการรวมฟังก์ชันไปไว้ในอินเทอร์เฟซ
- ความสามารถพื้นฐานของ อินเทอร์เฟซ
  1. วิธีที่สามารถค้นหาอินเทอร์เฟซของคอมโพเน็นต์แบบไดนามิก
  2. Reference Counting ที่ทำให้คอมโพเน็นต์สามารถจัดการช่วงชีวิตของตนเองได้
- กลไกในการแบ่งแยกคอมโพเน็นต์และอินเทอร์เฟซที่แตกต่างกัน
- ตัวสร้างคอมโพเน็นต์ เพื่อการสร้างการติดต่อของคอมโพเน็นต์ในกรณี In-process และ Cross-network

4.5 มาตรฐานไบนารี

COM นิยามมาตรฐานเลย์เอาต์ ของ Virtual Function Tables (vTables) ในหน่วยความจำ และมาตรฐานในการเรียกฟังก์ชันผ่าน vTables ดังนั้นภาษาใดก็ตามที่สามารถเรียกฟังก์ชันด้วยพอยน์เตอร์ (C, C++, Small Talk, Ada, Basic) ก็สามารถเขียนคอมโพเน็นต์และสามารถใช้งานคอมโพเน็นต์อื่น ๆ ที่เป็นมาตรฐานเดียวกันได้

Double Indirection (ไคลเอ็นต์ถือพอยน์เตอร์ ชี้ไปที่พอยน์เตอร์ ที่ชี้ไปที่ vTable) ทำให้สามารถแบ่ง vTable ใช้ได้ ระหว่างอินสแทนซ์ของคลาสเดียวกัน ในระบบที่มีอินสแทนซ์หลายร้อยตัวนั้น การแบ่งกันใช้ vTable จะช่วยลดความต้องการหน่วยความจำลงมาก



รูปที่ 4-2 โครงสร้างของ vTable

4.6 อินเทอร์เฟซ

COM คอมโพเน็นต์ติดต่อกันผ่านอินเทอร์เฟซ ซึ่งเป็นกลุ่มของฟังก์ชัน

อินเทอร์เฟซ เป็นเหมือนข้อตกลงของคอมโพเน็นต์ว่า จะทำงานตามฟังก์ชันที่กำหนดไว้ได้ ชื่อของอินเทอร์เฟซโดยปกติจะขึ้นต้นด้วยอักษร I ซึ่งผู้ใช้สามารถสร้างอินเทอร์เฟซของตนเองได้ พอยน์เตอร์ที่ชี้ไปยังคอมโพเน็นต์ออบเจ็กต์ที่จริงแล้วเป็นพอยน์เตอร์ซึ่งชี้ไปที่อินเทอร์เฟซหนึ่งของคอมโพเน็นต์ นั่นหมายความว่าเราใช้พอยน์เตอร์เรียกเมธอด

ตัวอย่าง อินเทอร์เฟซ

```
interface ILookup : public IUnknown
{
  public:
    virtual HRESULT __stdcall LookupByName( LPTSTR lpName,TCHAR **lplpNumber) = 0;
    virtual HRESULT __stdcall LookupByNumber(LPTSTR lpNumber, TCHAR **lplpName) = 0;
  };
```

คุณสมบัติของ อินเทอร์เฟซ

1. *อินเทอร์เฟซ ไม่ใช่คลาส*

   อินเทอร์เฟซไม่สามารถสร้างเป็นคอมโพเน็นต์ออบเจ็กต์ได้ เพราะไม่มีส่วนอิมพลีเมนต์ ซึ่งตรงกันข้ามกับคลาสจะใช้งานอินเทอร์เฟซได้ เมื่อมีการสร้างคอมโพเน็นต์ออบเจ็กต์ที่อิมพลีเมนต์อินเทอร์เฟซดังกล่าวเสียก่อน โดยคอมโพเน็นต์ออบเจ็กต์คลาสคนละตัวนั้น สามารถอิมพลีเมนต์อินเทอร์เฟซเดียวกันด้วยวิธีที่ต่างกันได้ จะเห็นได้ว่าพื้นฐานของโพลีมอร์ฟิซึมถูกประยุกต์ใช้กับคอมโพเน็นต์ออบเจ็กต์อย่างเต็มที่

2. *อินเทอร์เฟซ ไม่ใช่คอมโพเน็นต์ออบเจ็กต์*

   อินเทอร์เฟซ เป็นเพียงกลุ่มของฟังก์ชันที่เกี่ยวข้องกัน และเป็นมาตรฐานไบนารีที่ไคลเอ็นต์และคอมโพเน็นต์ออบเจ็กต์ใช้ติดต่อกันด้วย

   คอมโพเน็นต์ออบเจ็กต์ จะ อิมพลีเมนต์ ด้วยภาษาใดก็ได้ตราบเท่าที่สามารถมี พอยน์เตอร์ ชี้ไปที่ ฟังก์ชันสมาชิกของอินเทอร์เฟซ

3. *ไคลเอ็นต์ จะกระทำกับพอยน์เตอร์ชี้ไปอินเทอร์เฟซ เท่านั้น*

   เมื่อไคลเอ็นต์ใช้งานคอมโพเน็นต์ออบเจ็กต์ นั้น ไคลเอ็นต์ไม่มีอะไรมากไปกว่า พอยน์เตอร์ที่สามารถเข้าถึงฟังก์ชันของอินเทอร์เฟซ (เรียกว่า อินเทอร์เฟซพอยน์เตอร์) ซึ่งพอยน์เตอร์ดังกล่าวจะซ่อนรายละเอียดการอิมพลีเมนต์ภายใน ทำให้ไม่สามารถเห็นดาต้าของคอมโพเน็นต์ออบเจ็กต์ ซึ่งจุดนี้จะต่างกับ ออบเจ็กต์พอยน์เตอร์ของ C++ ที่ ไคลเอ็นต์ อาจเข้าถึงได้โดยตรง

   การ Encapsulation ดังกล่าว ทำให้เกิดมาตรฐานไบนารีที่มีประสิทธิภาพ สามารถทำ Local/Remote Transparency ได้

4. *คอมโพเน็นต์ออบเจ็กต์ สามารถอิมพลีเมนต์ได้หลาย อินเทอร์เฟซ*

เหมือน คลาสทั่วไปที่สามารถมีหลายเซตของบริการ

5. *อินเทอร์เฟซ เป็น Strongly Typed*

ทุกอินเทอร์เฟซ มีสิ่งระบุตนเอง นั่นคือ Globally Unique ID (GUID) ซึ่งกำจัดโอกาสที่จะมีอินเทอร์เฟซซ้ำซ้อนออกไป การใช้งานอินเทอร์เฟซนั้นต้องอ้างถึง GUID นี้ ทำให้หมดปัญหาเรื่องชื่อซ้ำกันได้

6. อินเทอร์เฟซ เปลี่ยนแปลงไม่ได้

อินเทอร์เฟซ ของ COM ไม่มีวันเปลี่ยนแปลง เมื่อมีการเพิ่มเติม ฟังก์ชันจะต้องสร้างอินเทอร์เฟซใหม่

หมายเหตุ อินเทอร์เฟซ ที่คล้ายกันสามารถร่วมกันใช้งานอิมพลีเมนต์ภายในได้ เพื่อเหตุผลในการสนับสนุนถึงไคลเอ็นต์เดิมและไคลเอ็นต์ ใหม่

ด้วยความเป็นหนึ่งเดียวของอินเทอร์เฟซของ COM ทำให้มีประโยชน์ คือ

1. *ความสามารถในการเปลี่ยนแปลงฟังก์ชัน*

QueryInterface ทำให้ออบเจ็กต์สามารถเพิ่มอินเทอร์เฟซใหม่เพื่อไคลเอ็นต์ใหม่ ในขณะที่ยังคงไว้ซึ่งความเข้ากันได้กับโค้ดของไคลเอ็นต์ที่มีอยู่ โดยไม่ต้องมีการคอมไพล์ใหม่ นับเป็นกุญแจหลักที่ใช้แก้ปัญหาเวอร์ชัน ซึ่งเป็นความต้องการพื้นฐานของคอมโพเน็นต์ซอฟต์แวร์ อินเทอร์เฟซของ COM เปลี่ยนแปลงไม่ได้ แต่เราสามารถเพิ่มฟังก์ชันใหม่ๆ ผ่านอินเทอร์เฟซใหม่ในขณะที่คอมโพเน็นต์นั้นยังคงสนับสนุนอินเทอร์เฟซเก่าด้วย ซึ่งเป็นการยืนยันความถูกต้องของการ Backward Compatibility เมื่อคอมโพเน็นต์มีการเปลี่ยนแปลง

2. *ใช้งานออบเจ็กต์ ได้ง่าย และเร็ว*

การเรียกฟังก์ชันของ COM นั้น เป็นการเรียกผ่านอินเทอร์เฟซสองตัว ซึ่งนับเป็น Overhead ที่น้อยมาก

3. *อินเทอร์เฟซ รียูส*

มีบางกลุ่มของการทำงาน ที่ใช้งานได้กับหลาย ๆ คอมโพเน็นต์ เช่น การอ่านเขียน byte stream ซึ่งคอมโพเน็นต์สามารถ รียูสอินเทอร์เฟซได้

4. *Local/Remote Transparency*

COM สามารถเข้ากั้นกลางระหว่างการเรียกอินเทอร์เฟซไปยังออบเจ็กต์ ทำให้ไคลเอ็นต์ใช้วิธีการเรียกโพรซีเยอร์ระยะไกล (Remote Procedure Call) ไปยังออบเจ็กต์ที่อยู่คนละโพรเซส หรือคนละเครื่องได้ เหมือนกับวิธีการเรียกโพรซีเยอร์ที่อยู่ในโพรเซส เดียวกัน

5. *เป็นอิสระจาก Programming Language*

ภาษาโปรแกรมที่สามารถสร้าง Structure ของพอยน์เตอร์ และเรียกฟังก์ชันผ่านพอยน์เตอร์ได้ จะสามารถสร้างและใช้งาน คอมโพเน็นต์ออบเจ็กต์ ได้

COM ต่างจาก Object–Oriented Programming ตรงที่ COM นั้นเป็น Binary Object Standard ไม่ใช่ Source Code Standard

4.7 Globally Unique Identifiers (GUID)

COM ใช้ตัวระบุที่ไม่ซ้ำกันเลยทั่วโลก (เป็น Integer 128 บิต ซึ่งการันตีว่าจะไม่ซ้ำกัน) เป็นตัวแบ่งแยกอินเทอร์เฟซและ คอมโพเน็นต์ออบเจ็กต์ คลาส

ตัวระบุดังกล่าว คือ UUIDs (Universally Unique Identifiers) ซึ่งนิยามโดย Open Software Foundation's Distributed Computing Environment

ชื่อที่มนุษย์อ่านได้นั้นเหมาะสำหรับใช้ในงานที่มีขนาดไม่ใหญ่นัก แต่ถ้าใช้ GUID จะทำให้มั่นใจได้ว่า COM คอมโพเน็นต์ จะไม่เชื่อมต่อกับคอมโพเน็นต์,อินเทอร์เฟซหรือ เมธอด ที่ผิด

ตัวอย่าง GUID

DEFINE_GUID(CLSID_PHONEBOOK, 0xc4910d70, 0xba7d, 0x11cd, 0x94, 0xe8, 0x08, 0x00, 0x17, 0x01, 0xa8, 0xa3);

DEFINE_GUID(IID_ILOOKUP, 0xc4910d71, 0xba7d, 0x11cd, 0x94, 0xe8, 0x08, 0x00, 0x17, 0x01, 0xa8, 0xa3);

GUIDs จะถูกฝังไว้ในตัวคอมโพเน็นต์ไบนารี และถูกใช้โดย COM System แบบ Dynamic ตอน bind time เพื่อยืนยันความถูกต้องในการเชื่อมต่อกับคอมโพเน็นต์

4.8 C++, คอมโพเน็นต์ และ อินเทอร์เฟซ

หนึ่งในประโยชน์ของการพัฒนาด้วยภาษาแบบ object-oriented อย่าง C++ และ Java คือ การ encapsulate ได้อย่างมีประสิทธิภาพ ออบเจ็กต์จะซ่อนรายละเอียดของอิมพลีเมนต์ และจะมีพับลิคอินเทอร์เฟซที่ให้ไคลเอ็นต์มาใช้งานออบเจ็กต์ ซึ่ง COM จัดเตรียมความสามารถส่วนนี้ โดยนิยามวิธีมาตรฐานในการอิมพลีเมนต์ และแสดงอินเทอร์เฟซของ COM-based ออบเจ็กต์

ด้านล่างคือโค้ด ของ C++ คลาสแบบ ธรรมดา

```
class Math
{
//ส่วน อินเทอร์เฟซ
public:
        long Add ( long Op1, long Op2 ) ;
        long Subtract (long Op1, long Op2 ) ;
//ส่วน อิมพลีเมนต์
private:
        string m_strVersion;
        string get_Version();
};
```

```
long Math::Add ( long Op1, long Op2 )
{
        return Op1+Op2 ;
}
long Math::Subtract (long Op1, long Op2 )
{
        return Op2-Op2 ;
}
```

คลาสข้างต้นมีฟังก์ชันพื้นฐานทางคณิตศาสตร์ ต่อไปจะเป็นการเปลี่ยนคลาสเป็น COM ออบเจ็กต์ ที่สามารถใช้ได้กับภาษาใดก็ได้ที่สนับสนุนอินเทอร์เฟซของ COM

ขั้นตอนแรก คือนิยามอินเทอร์เฟซ โดยใช้แอ็บสแทรกต์คลาส

```
class IMath
{
//ส่วน อินเทอร์เฟซ
public:
        virtual long Add ( long Op1, long Op2 ) = 0 ;
        virtual long Subtract (long Op1, long Op2 ) = 0 ;
};
```

จากนั้นก็สืบทอดแอ็บสแทรกต์คลาสและทำ อิมพลีเมนต์ เหมือน คลาสธรรมดา

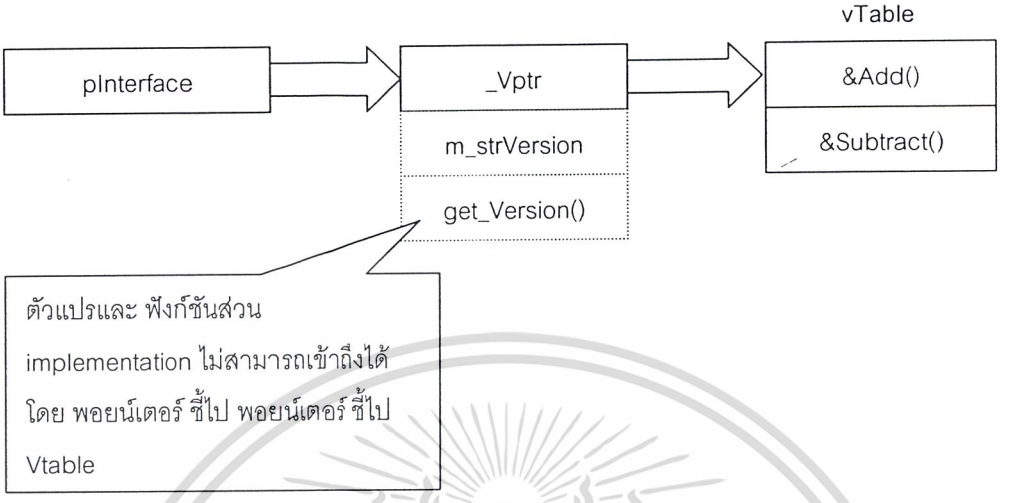
```
class Math : public IMath
{
public:
        long Add ( long Op1, long Op2 ) ;
        long Subtract (long Op1, long Op2 ) ;
};
```

นี่คือขั้นตอนแรกของการทำให้ คลาสสามารถใช้ได้กับภาษาที่ไม่ใช่ C++ ซึ่ง คลาสใหม่นี้เป็นแอ็บสแทรกต์คลาส (มีอย่างน้อย 1 pure virtual function และมีสมาชิกเป็น พับลิค เมธอด เท่านั้น)

คลาสใหม่นี้มีชื่อว่า IMath ซึ่งอักษร I บ่งบอกให้รู้ว่าเป็นการประกาศอินเทอร์เฟซสิ่งสำคัญที่สุดของ IMath คลาสคือ การบังคับให้สร้าง vTable ของ C++ สำหรับ คลาสที่สืบทอดต่อไป

การใช้ virtual function ใน base คลาสเป็นหัวใจของการออกแบบ COM ส่วนนิยามของ แอ็บสแทรกต์คลาสทำให้เกิด vTable ที่มีเฉพาะพับลิคเมธอด คลาส IMath ไม่มีดาต้าเมมเบอร์ (data member) และส่วนอิมพลีเมนต์ โดยมีจุดประสงค์ในการบังคับให้คลาสที่สืบทอด (Math) ต้องอิมพลีเมนต์เมธอดของคอมโพเน็นต์อินเทอร์เฟซ

การเข้าถึงคอมโพเน็นต์ต้องกระทำผ่าน vTable พอยน์เตอร์เท่านั้น เพราะฉะนั้น การเข้าถึงส่วนอิมพลีเมนต์ โดยตรงจึงเป็นไปไม่ได้ COM อินเทอร์เฟซ จึงเป็นแค่ พอยน์เตอร์ ที่ชี้ไปที่ พอยน์เตอร์ที่ ชี้ไปที่ vTable แบบ C++ เท่านั้น



รูปที่ 4-3 รายละเอียดของ vTable

4.9 IUnknown

เมื่อต้องการสร้าง COM คอมโพเน็นต์นั้น งานแรกที่จะต้องทำคือ การอิมพลีเมนต์อินเทอร์เฟซพื้นฐานคือ IUnknown ซึ่งทุกอินเทอร์เฟซจะต้องมีส่วนอิมพลีเมนต์ของ IUnknown ด้วย

อินเทอร์เฟซ IUnknown มีจุดประสงค์ 2 ประการ

1. วิธีการมาตรฐานของการที่ ไคลเอ็นต์ ขออินเทอร์เฟซที่ต้องการจากคอมโพเน็นต์ ซึ่งเมธอด QueryInterface ทำหน้าที่ส่วนนี้
2. วิธีการจัดการช่วงชีวิตของคอมโพเน็นต์จากภายนอก ซึ่งมีเมธอด AddRef และ Release ทำหน้าที่นี้

ต่อไปเป็น Definition ของ IUnknown

```
Class IUnknown
{
        virtual HRESULT QueryInterface (REFIID riid, void** ppv) = 0 ;
        virtual ULONG   AddRef()=0;
        virtual ULONG   Release()=0;
}
```

IUnknown เป็นส่วนประกอบของ COM อินเทอร์เฟซ ซึ่งเป็นแอ็บสแทรกต์คลาส โดย คลาสใดก็ตามที่สืบทอดจากอินเทอร์เฟซนี้ ก็ต้อง อิมพลีเมนต์ ทั้ง 3 เมธอด ด้านบน เป็นผลให้เกิด vTable ของ 3 เมธอด ดังกล่าว

4.9.1 HRESULT

เมธอดของ COM อินเทอร์เฟซ และฟังก์ชัน API ส่วนใหญ่จะคืนค่าเป็น HRESULT (ยกเว้น AddRef และ Release) HRESULT ใน Win32 นิยามเป็น DWORD (32 บิต) ที่บรรจุข้อมูลของผลลัพธ์ในการเรียกฟังก์ชัน

บิตสูงสุด ใช้บอกว่าฟังก์ชันนั้นทำงานสำเร็จหรือล้มเหลว ส่วน 15 บิตต่อมาบอกกลุ่มของ return โค้ด ที่เกี่ยวข้องกัน และ 16 บิตสุดท้ายบอกข้อมูลจำเพาะที่เกิดขึ้น

4.9.2 Iunknown::QueryInterface

เมธอด QueryInterface รับค่าเป็นตัวอ้างอิง (Reference) ของ Interface Identifier (IID) ซึ่งเป็นตัวระบุที่มีขนาด 128 บิต (GUID) และคืนค่าเป็นตัวชี้ไปที่อินเทอร์เฟซ (เช่น IUnknown, IMath) ของ COM ออบเจ็กต์ ตัวชี้ดังกล่าวคืนค่าทางพารามิเตอร์ตัวที่สอง ซึ่งเป็นตัวชี้ไปยังตัวชี้แบบ void (pointer to a void pointer)

COM สามารถทำโพลีมอร์ฟีซึม และการค้นหาอินเทอร์เฟซแบบ dynamic ได้ด้วย เมธอด QueryInterface นี้เอง

COM ออบเจ็กต์ ต้องอิมพลีเมนต์อย่างน้อย 1 อินเทอร์เฟซขึ้นไป ซึ่งออบเจ็กต์ที่อิมพลีเมนต์หลายอินเทอร์เฟซนั้นจะให้ไคลเอ็นต์ เรียกหาอินเทอร์เฟซต่าง ๆ โดยการเรียก QueryInterface โดยส่ง IID เข้ามา และออบเจ็กต์จะคืนค่าเป็นตัวอ้างอิงของอินเทอร์เฟซนั้น ถ้าไคลเอ็นต์เรียกหาอินเทอร์เฟซที่ออบเจ็กต์ไม่สนับสนุนแล้ว การเรียกเมธอด QueryInterface จะล้มเหลว

ตัวอย่าง คลาส IMath และการประกาศ CLSID และ IID

```
DEFINE_GUID( CLSID_Math,
    0xa888f560, 0x58e4, 0x11d0, 0xa6, 0x8a, 0x0, 0x0, 0x83, 0x7e, 0x31, 0x0);
DEFINE_GUID( IID_IMath,
    0xa888f561, 0x58e4, 0x11d0, 0xa6, 0x8a, 0x0, 0x0, 0x83, 0x7e, 0x31, 0x0);
class IMath : public IUnknown
{
public :
    virtual HRESULT Add ( long Op1, long Op2 , long* pResult) = 0 ;
    virtual HRESULT Subtract (long Op1, long Op2, long* pResult ) = 0 ;
};
```

จะเห็นได้ว่าการคืนค่าจากเมธอด มีความยุ่งยากขึ้นเพราะต้องเป็นลักษณะของ พอยน์เตอร์ ทั้งนี้เนื่องจากกฎของ COM ที่ต้องคืนค่าเป็น HRESULT ทุกๆเมธอด

4.9.3 IUnknown::AddRef และ IUnknown::Release

การจัดการช่วงชีวิตของคอมโพเน็นต์ทำโดยเมธอด AddRef และ Release โดยปกติ COM คอมโพเน็นต์ จะมีหลายอินเทอร์เฟซซึ่งแต่ละอินเทอร์เฟซอาจติดต่อกับหลายไคลเอ็นต์ จึงต้องมีวิธีการทำ reference counting โดยเมื่อไคลเอ็นต์รีเควสต์อินเทอร์เฟซก็จะเพิ่มค่าเคาน์เตอร์ และเมื่อไคลเอ็นต์ จบการทำงานกับอินเทอร์เฟซก็จะลดค่าเคาน์เตอร์ สุดท้ายเมื่อเคาน์เตอร์ลดลงถึง 0 คอมโพเน็นต์ก็จะทำลายตัวเอง

ดังนั้น ตัวอย่าง Math คลาสจึงต้องมีเคาน์เตอร์ภายใน โดยตัวอย่างตั้งชื่อว่า m_ref เมื่อคืนค่าเป็นอินเทอร์เฟซจะต้องเพิ่มค่าเคาน์เตอร์ และไคลเอ็นต์ต้องเรียก IUnknown::Release เมื่อจบการใช้งานอินเทอร์เฟซ

```
extern long g_lObjs;
class Math : public IMath
{
public:
        HRESULT QueryInterface (REFIID riid, void** ppv) ;
        ULONG   AddRef() ;
        ULONG   Release() ;

        HRESULT Add ( long Op1, long Op2, long* pResult ) ;
        HRESULT Subtract (long Op1, long Op2, long* pResult ) ;

        Math();
        ~Math();
private:
        DWORD m_ref;
};
Math::Math()
{
        m_ref=0;
        InterLockedIncrement(&g_lObjs);
}
Math::~Math()
{
        InterLockedDecrement(&g_lObjs);
}
```

```
HRESULT Math::Add ( long Op1, long Op2, long* pResult )
{
        *pResult=Op1+Op2 ;
        return S_OK;
}
HRESULT Math::Subtract (long Op1, long Op2, long* pResult )
{
        *pResult=Op2-Op2 ;
        return S_OK;
}
HRESULT Math::QueryInterface ( REFID riid, voi** ppv)
{
        *ppv = 0;
        if ( riid == IID_Iunknown || riid == IID_IMath)
                *ppv=this;
        if(*ppv)
        {
                AddRef();
                return S_OK;
        }
        return E_NOINTERFACE ;
}
ULONG Math::AddRef()
{
        InterlockedIncrement ( &m_ref);
        return m_ref;
}
ULONG Math::Release()
{
        if ( InterlockedDecrement ( &m_ref ) == 0 )
        {
                delete this;
                return 0;
        }
```

```
    return m_ref;
}
```

4.10 ประเภทของ COM server

มี 3 แบบคือ

- In-process server เป็นไลบรารี (DLL) ที่ทำงานภายในโพรเซสเดียวกันกับไคลเอ็นต์ เช่น แอกทีฟเอ็กซ์คอนโทรล (ActiveX control) ซึ่งการติดต่อกับ In-process server จะกระทำโดยตรงไปยังคอมโพเน็นต์
- Out-of-process server หรือ Local server เป็นแอพพลิเคชัน (EXE) ที่ทำงานคนละโพรเซสกับไคลเอ็นต์ แต่อยู่ในเครื่องเดียวกัน

  Local server ทำงานช้ากว่า in-process เพราะระบบปฏิบัติการต้องสวิตช์โพรเซสและก็อปปี้ข้อมูลที่ต้องถ่ายโอนไปจากไคลเอ็นต์ไปยังserver
- Remote server ทำงานคนละเครื่องกับไคลเอ็นต์ เพราะฉะนั้นจึงเป็นคนละโพรเซสกัน ซึ่งการทำงานจะใช้ Distributed COM (DCOM)

  ข้อดีของ DCOM คือไม่ต้องการการโปรแกรมเพิ่มเติมในการทำงานระยะไกล DCOM ทำงานโดยใช้พรอกซี่ที่กั้นกลางระหว่างอินเทอร์เฟซกับออบเจ็กต์ ซึ่งพรอกซี่จะทำงานในส่วนเรียกฟังก์ชันในระยะไกลให้เอง จุดสำคัญคือ ไคลเอ็นต์เรียกใช้งานคอมโพเน็นต์เหมือนกับเป็น in-process server

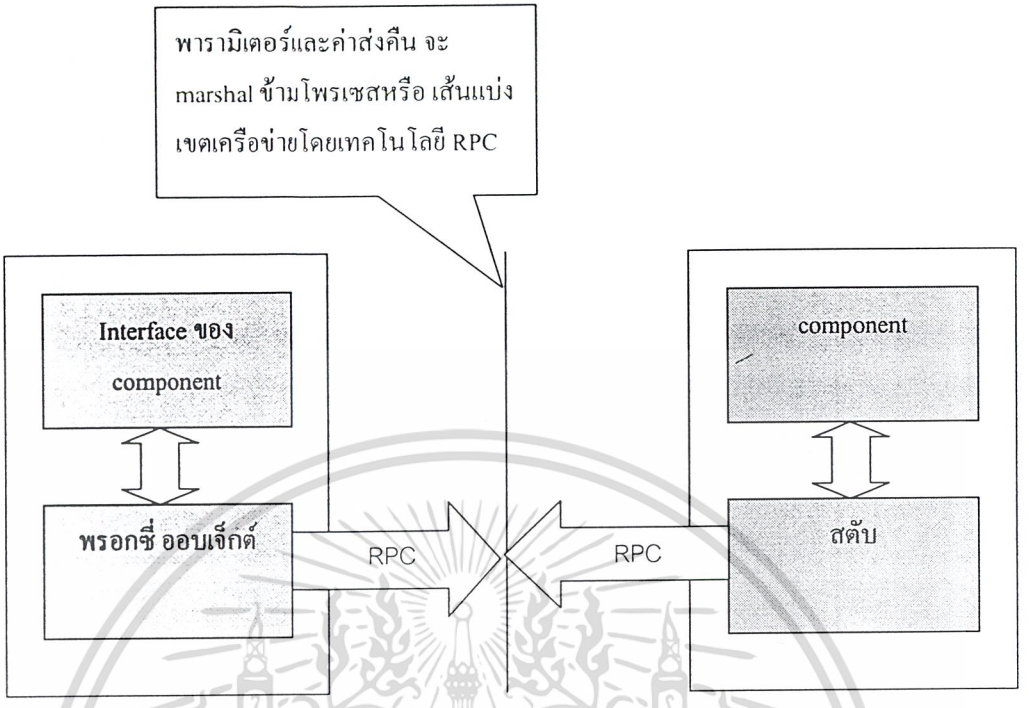
  เนื่องจาก DCOM ทำงานผ่านเครือข่าย ต้องมีการถ่ายโอนข้อมูลข้ามเครือข่าย จึงทำให้ server แบบนี้มีความเร็วน้อยที่สุด

จะเห็นได้ว่าประโยชน์ของ COM คือ location transparency โดยมุมมองของไคลเอ็นต์ จะไม่ทราบเลยว่าคอมโพเน็นต์อยู่ที่ใด เมื่อไคลเอ็นต์สร้างอินสแทนซ์ของ COM เป็นหน้าที่ของ COM ที่จะหาที่ตั้งของ COM และสร้างออบเจ็กต์ให้ประโยชน์ส่วนนี้จึงทำให้ COM คอมโพเน็นต์ เหมาะกับงาน multi-tier

4.11 มาแชลลิง (Marshaling)

มาแชลลิง คือกระบวนการการถ่ายโอนอาร์กิวเมนต์ของฟังก์ชันข้ามโพรเซสหรือขอบเขตเครือข่าย ซึ่งต้องมีการก็อปปี้ค่าตัวแปรต่าง ๆ ส่งไปอีกโพรเซสหนึ่ง

สำหรับ in-process server ไม่จำเป็นต้องทำมาแชลลิงจึงทำให้คอมโพเน็นต์ชนิดนี้มีประสิทธิภาพที่สุด



รูปที่ 4-4 รีโมตคอมโพเน็นต์

4.12 Class Factories

COM กำหนดอินเทอร์เฟซมาตรฐานชื่อ IClassFactory ซึ่งบังคับให้ทุกคอมโพเน็นต์ต้องบรรจุไว้

```
class  IClassFactory  : public IUnknown
{
public:
   virtual HRESULT CreateInstance ( LPUNKNOWN pUnk, REFIID riid, void** ppv)=0 ;
   virtual HRESULT LockServer (BOOL fLock) = 0 ;
};
```

คลาส Factory เป็น COM คอมโพเน็นต์ ที่มีจุดประสงค์ใช้ในการสร้าง COM คอมโพเน็นต์ตัวอื่น ๆ ในคอมโพเน็นต์ไบนารีใด ๆ (DLL หรือ EXE) จะต้องมีส่วนอิมพลีเมนต์ของคลาส factory สำหรับการสร้างคอมโพเน็นต์ได้จากภายนอก

IClassFactory มี 2 เมธอด

CreateInstance : สร้าง อินสแทนซ์ของ คอมโพเน็นต์

LockServer : ไคลเอ็นต์สามารถ lock binary file ให้คงอยู่ในหน่วยความจำ ทำให้สามารถสร้างคอมโพเน็นต์ได้เร็วขึ้น

```
extern long g_Locks;
class MathClassFactory : public IClassFactory
{
protected:
        long m_ref;
public:
        MathClassFactory();
        ~MathClassFactory();
        // IUnknown
        HRESULT QueryInterface (REFIID riid, void** ppv);
        ULONG   AddRef() ;
        ULONG   Release() ;
        // IClassFactory
        HRESULT CreateInstance (LPUNKNOWN pUnk, REFIID riid, void** ppv) ;
        HRESULT LockServer (BOOL fLock);
};
MathClassFactory::MathClassFactory()
{
        m_ref=0;
}
MathClassFactory::~MathClassFactory()
{
}
MathClassFactory::QueryInterface (REFIID riid, void** ppv)
{
        *ppv = 0;
        if ( riid == IID_Iunknown || riid == IID_IClassFactory)
                *ppv=this;
        if(*ppv)
        {
                AddRef();
                return  S_OK;
        }
        return  E_NOINTERFACE ;
```

```
}
MathClassFactory:: AddRef()
{
        return InterlockedIncrement ( &m_ref ) ;
}
MathClassFactory:: Release()
{
        if ( InterlockedDecrement ( &m_ref ) == 0 )
        {
                delete this;
                return 0;
        }
        return m_ref;
}

HRESULT MathClassFactory::CreateInstance ( LPUNKNOWN pUnk, REFIID riid, void** ppv)
{
        Math* pMath;
        HRESULT hr;

        *ppv = 0 ;

        pMath = new Math ();

        if ( pMath == 0 )
                return ( E_NOINTERFACE );
        hr = pMath->QueryInterface( riid , ppv)

        if ( FAILED(hr) )
                delete pMath;
        return hr;
}
HRESULT MathClassFactory:: (BOOL fLock)
{
```

```
        if ( fLock)
                InterlockedIncrement ( &g_Locks )
        else
                InterlockedDecrement ( &g_Locks);
        return  S_OK;
}
```

แต่ละคอมโพเน็นต์ต้องการคลาส factory ดังนั้นคอมโพเน็นต์ไบนารี จึงต้องมีจุดที่ใช้เข้าถึงคลาส factory โดยสำหรับ DLL นั้นต้อง export 2 ฟังก์ชัน นั่นคือ DllGetClassObject และ DllCanIUnloadNow ส่วน EXE จะ register คลาส factory โดยใช้ COM API ที่ชื่อ CoRegisterClassObject

ในตัวอย่างจะใช้ in-process server จึงใช้ ฟังก์ชัน DllGetClassObject

```
long g_lObjs;
long g_Locks;
STDAPI DllGetClassObject (REFCLSID rclsid, REFIID riid, void** ppv)
{
        HRESULT hr;
        MathClassFactory *pcf;
        pcf=0;
        //ตรวจสอบว่า CLSID ที่ต้องการใช่คอมโพเน็นต์นี้หรือไม่
        if ( rclsid != CLSID_MATH)
                return (E_FAIL);
        pcf = new MathClassFactory();
        if (pcf == 0 )
                return (E_OUTOFMEMORY);
        hr = pcf->QueryInterface(riid,ppv);

        if ( FAILED(hr))
        {
                delete pcf;
                pcf=0;
        }
        return hr;
}
```

```
STDAPI DllCanUnloadNow(void)
{
        if ( g_Objs || g_Locks )
                return S_FALSE
        else
                return S_OK;
}
```

การ export ฟังก์ชันใน DLL ทำได้โดยสร้างแฟ้ม server.def

```
LIBRARY "SERVER"
DESCRIPTION "SERVER Windows Dynamic Link Library"

EXPORTS
        DllGetClassObject PRIVATE
        DllCanUnloadNow PRIVATE
```

4.13 The Registry

Windows registry จะเป็นสิ่งที่ใช้บอกว่า COM คอมโพเน็นต์ ตั้งอยู่ที่ใด

Key ที่สำคัญคือ CLSID ซึ่งเป็น key ย่อยของ HKEY_CLASSES_ROOT ซึ่งจะอธิบายถึงคอมโพเน็นต์ทุกตัวที่ติดตั้งบนระบบ

ต่อไปเป็นตัวอย่างของไฟล์ที่ต้องถูกลงทะเบียนในรีจิสตรีของระบบ

```
REGEDIT
HKEY_CLASS_ROOT\Math.Component.1 = Math Component Description
HKEY_CLASS_ROOT\Math.Component.1\CurVer = Math.Component.1
HKEY_CLASS_ROOT\\Math.Component.1\CLSID = {A888F560-58E4-11d0-A68A-0000837e3100}
HKEY_CLASS_ROOT\CLSID\{A888F560-58E4-11d0-A68A-0000837e3100} = Math Component Description
Component
HKEY_CLASS_ROOT\CLSID\{A888F560-58E4-11d0-A68A-0000837e3100}\ProgID= Math.Component.1
HKEY_CLASS_ROOT\CLSID\{A888F560-58E4-11d0-A68A-0000837e3100}\VersionIndepentProgID=
Math.Component
HKEY_CLASS_ROOT\CLSID\{A888F560-58E4-11d0-A68A-0000837e3100}\InprocServer32 = c:\server\server.dll
HKEY_CLASS_ROOT\CLSID\{A888F560-58E4-11d0-A68A-0000837e3100}\NotInsertable
```

3 บรรทัดแรกเป็นตัวระบุของการโปรแกรมของ Component CLSID ของคอมโพเน็นต์เป็น GUID แต่ไม่ง่ายต่อการจดจำ COM จึงมี ProgID ซึ่งง่ายต่อการจำ บรรทัดที่ 3 เป็นการจับคู่ของ CLSID กับ ProgID

โค้ดชุดล่างเป็นข้อมูลที่สำคัญของคอมโพเน็นต์ที่จะบอกที่ตั้งของคอมโพเน็นต์ ซึ่งก็คือส่วน InProcServer32

4.14 COM Activation

1. เริ่มต้น COM โดยเรียกฟังก์ชัน CoInitialize
2. ไคลเอ็นต์ใช้ CoGetClassObject ใช้เรียกหาคลาส factory อินเทอร์เฟซ (ระบุ CLSID) จากนั้น service control manager (SCM) จะนำ CLSID ไปหาที่ตั้งของคอมโพเน็นต์ไบนารี ใน registry และจะเรียกฟังก์ชัน DllGenClassObject ของคอมโพเน็นต์นั้น
3. คอมโพเน็นต์ได้รับคลาส factory อินเทอร์เฟซพอยน์เตอร์ แล้วเรียกฟังก์ชัน CreateInstance (ระบุ IID ที่ต้องการ) เพื่อสร้างอินสแทนซ์ของคอมโพเน็นต์จริง ๆ
4. ไคลเอ็นต์ release คลาส factory อินเทอร์เฟซ
5. ไคลเอ็นต์สามารถใช้เมธอดของอินเทอร์เฟซ ได้

หมายเหตุ ฟังก์ชัน CoCreateInstance เป็นฟังก์ชันที่อำนวยความสะดวกที่ซ่อนการทำงานตั้งแต่ 2-4

ต่อไปเป็นตัวอย่างโค้ดของไคลเอ็นต์

```
int main()
{
        Coinitialize();
        IclassFactory* pcf;
        HRESULT hr = CoGetClassObject ( CLSID_MATH, CLSCTX_INPROC, NULL,
IID_IclassFactory, (void**)&pcf);
        IUnknown* punk;
        hr = pcf->CreateInstance (NULL, IID_IUnknown, (void**)&punk);
        pcf->release();
        IMath* pmath = NULL;
        hr = punk->QueryInterface ( IID_IMath, (void**)&pmath);
        long result;
        pmath->Add( 5 , 10 , &result);
        pmath->Release();
        CoUninitialize();
        return 0;
}
```

4.15 COM และความอิสระต่อภาษา

COM มีวิธีที่ทำให้ภาษาโปรแกรมมิ่งทุกภาษาเข้าใจอินเทอร์เฟซและ coclass นั่นคือ Interface Definition Language (IDL) ซึ่ง IDL ยังช่วยสนับสนุนการทำ RPC อีกด้วย

IDL ใช้ syntax แบบ C แต่ได้เพิ่มคุณสมบัติบางอย่างให้เหมาะกับการอธิบายคอมโพเน็นต์และยังได้เพิ่ม attribute ที่บอกทิศทางของข้อมูลที่จะเข้าหรือออกจากคอมโพเน็นต์ด้วย

ตัวอย่าง IDL

```
import "oaidl.idl"
import "ocisl.idl"

[
        object,
        uuid(a888f560-58e4-11d0-a68a-0000837e3100),
        dual,
        helpstring("IMath_interface"),
        pointer_default(unique)
]
interface IMath : Idistpatch
{
        HRESULT Add ( [in] long, [in] long, [out, retval] long* pResult);
        HRESULT Subtract ( [in] long, [in] long, [out, retval] long* pResult);
};
[
        uuid(a888f506-58e4-11d0-a68a-0000837e3100),
        version(1.0),
        helpstring("Math 1.0 Type Library")
]
library  MathLib
{
        importlib("stdole32.tlb");
        importlib("stdole2.tlb");

        [
        uuid(a888f561-58e4-11d0-a68a-0000837e3100),
        helpstring("Math class")
```

```
    ]
    coclass MathComponent
    {
    [default] interface IMath;
    };
};
```

4.16 Microsoft IDL Compiler (MIDL)

เมื่ออินเทอร์เฟซถูกนิยามในรูปแบบ IDL แล้ว สามารถใช้ MIDL แปลงอินเทอร์เฟซเป็น โค้ดภาษา C/C++ ได้ แต่โค้ด C/C++ นั้นไม่สามารถเข้าใจได้โดยภาษาโปรแกรมมิ่งอื่น ๆ

เพื่อหลีกเลี่ยงปัญหาของ MIDL ในการที่ต้องปรับปรุงให้สามารถสร้างโค้ดภาษาอื่น ๆ เพิ่มขึ้นนั้น Microsoft แก้ปัญหาโดนนิยามรูปแบบการอินเทอร์เฟซที่ทุก ๆ ภาษาเข้าใจได้ นั่นคือ Type Library

เนื่องจาก COM เป็นมาตรฐานไบนารี ดังนั้น Type Library จึงเป็น IDL ในรูปแบบไบนารี ที่สามารถฝังตัวลงไปกับ COM คอมโพเน็นต์ ทำให้ คอมไพเลอร์ อื่น ๆ เช่น VB, Delphi เข้าใจอินเทอร์เฟซของคอมโพเน็นต์โดยการใช้ Type Library

4.17 การนำ COM คอมโพเน็นต์กลับมาใช้ใหม่

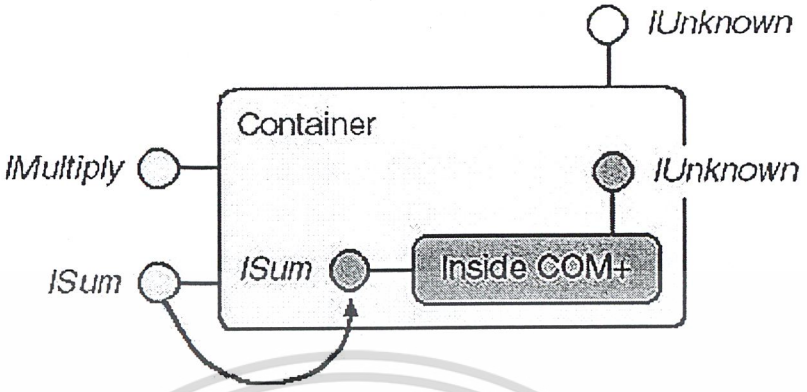
COM+ ถูกวิจารณ์ในเรื่องการไม่สนับสนุนการสืบทอดคุณสมบัติแบบอิมพลีเมนเตชัน แต่การสืบทอดคุณสมบัติไม่ได้เป็นวิธีเดียวในการนำโค้ดกลับมาใช้ใหม่ การสืบทอดคุณสมบัติแบบอิมพลีเมนเตชันที่พื้นฐานบนภาษาเขียนโปรแกรมมักจะมีปัญหาคือ เบสคลาสที่ไม่คุณภาพพอทำให้โปรแกรมมีปัญหาเพราะการเปลี่ยนแปลงของเบสคลาส

การนำโค้ดกลับมาใช้ใหม่ของ COM+ ทำได้โดยการสร้างอินเทอร์เฟซซึ่งถูกอิมพลีเมนต์โดยออบเจ็กต์ จากนั้นแอพพลิเคชันที่เขียนขึ้นโดยภาษาใดๆ ก็สามารถนำโค้ดกลับมาใช้ใหม่ตราบเท่าที่อยู่ในสภาพแวดล้อมของ COM+

คอนเทนเมนต์ (containment) และ แอกกรีเกชัน (aggregration) เป็นเทคนิคในการรวมออบเจ็กต์หลายตัวเข้าด้วยกัน ซึ่งมีประโยชน์เมื่อต้องการเพิ่มบริการเข้าไปในออบเจ็กต์โดยไม่ต้องแก้ไขซอร์สโค้ดเลย

4.17.1 คอนเทนเมนต์

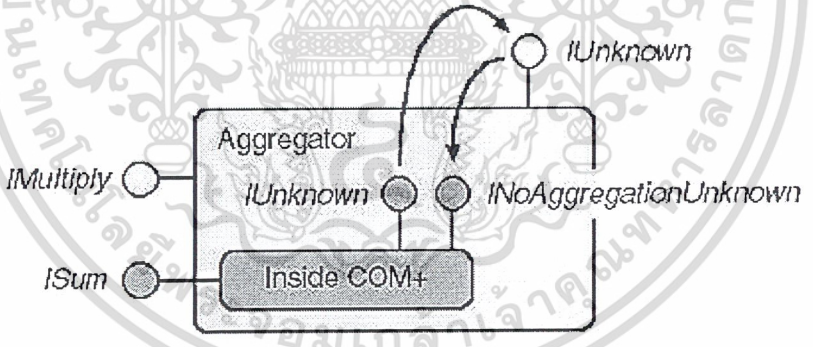
แนวคิดพื้นฐานของคอนเทนเมนต์ คือออบเจ็กต์หนึ่งอิมพลีเมนต์อินเทอร์เฟซอื่นอย่างสมบูรณ์



รูปที่ 4-5 แสดงการดีลีเกตการเรียกเมธอดโดยวิธีคอนเทนเมนต์

เทคนิคของคอนเทนเมนต์นั้นง่ายๆ ตามแนวคิดที่ว่า COM+ ออบเจ็กต์สามารถเป็นไคลเอ็นต์ของออบเจ็กต์อื่นได้ จากรูปจะเห็นว่าออบเจ็กต์สนับสนุนอินเทอร์เฟซใหม่ โดยที่จริงแล้วแค่ส่งผ่านรีเควสต์ของไคลเอ็นต์ไปยังออบเจ็กต์อื่นอีกทอดหนึ่งเท่านั้น

4.17.2 แอกกรีเกชัน



รูปที่ 4-6 COM+ ออบเจ็กต์ที่ถูกแอกกรีเกรต

แอกกรีเกชันเป็นรูปแบบพิเศษของคอนเทนเมนต์ แทนที่จะให้ออบเจ็กต์ใช้สตับเมธอดในการดีลีเกตไปยังออบเจ็กต์จริง กลับอนุญาติให้เปิดเผยอินเทอร์เฟซของออบเจ็กต์ใน (inner object) ออกมาโดยตรง

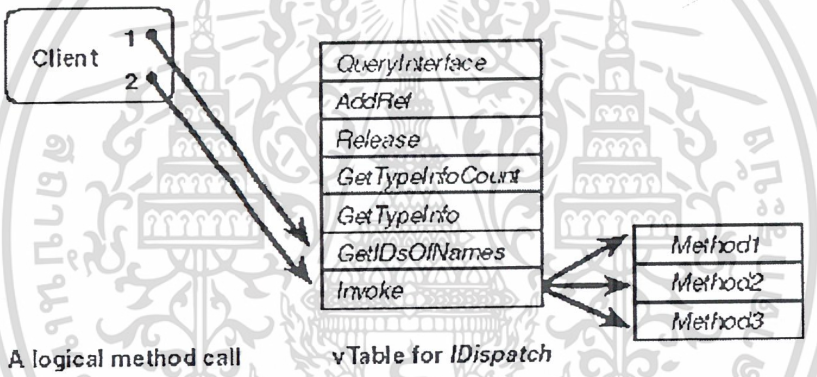
การอิมพลีเมนต์แบบแอกกรีเกชันนั้นมีความยุ่งยากเพราะต้องมีการประสานงานระหว่างออบเจ็กต์ในและนอก ซึ่งออบเจ็กต์ทั้งสองต้องเป็นไปตามกฎของ COM+ อย่างเคร่งครัด

4.18 อินเทอร์เฟซ IDispatch

ออโตเมชัน (Automation) เป็นกลไกที่ใช้ในการไบดิง (binding) ขณะรันไทม์ (run-time) หรือที่เราเรียกว่า เลทไบดิง (late binding) โดยใช้หลักการของอินเทอร์เฟซ IDispatch

อินเทอร์เฟซ IDispatch จะอนุญาตให้ไคลเอนต์ทำการเรียกใช้เมธอด (Method) ในออโตเมชันเซิร์ฟเวอร์ (Automation Server) แบบไดนามิก (dynamic) การตอบโต้กันระหว่างไคลเอนต์กับเซิร์ฟเวอร์จะใช้เลทไบดิง ซึ่งขณะที่ไคลเอนต์ทำการคอมไพล์นั้น ไม่ต้องรู้ว่าคอมโพเน็นต์ (component) ที่จะเรียกใช้นั้นอยู่ในอินเทอร์เฟซชื่ออะไร ซึ่งทำให้ไคลเอนต์นั้นเป็นอิสระกับคอมโพเน็นต์ เนื่องจากจะทำการเรียกใช้จริงๆ ตอนรันไทม์เท่านั้น ถ้าหากมีการเปลี่ยนแปลงอะไรที่คอมโพเน็นต์แล้ว ตัวไคลเอนต์ก็ไม่ต้องคอมไพล์ใหม่ (ตัวอินเทอร์เฟซต้องไม่มีการเปลี่ยนแปลง แต่สามารถเพิ่มเมธอดใหม่เข้าไปได้)

อินเทอร์เฟซ IDispatch จะมีส่วนที่ขยายเพิ่มจากอินเทอร์เฟซที่สร้างขึ้นมา 4 เมธอด ดังแสดงในรูปที่ 4-6 ในส่วนของการทำเลทไบดิงของออโตเมชันไคลเอนต์นั้นจะใช้เมธอด GetIDsOfNames และเมธอด Invoke



รูปที่ 4-7 อินเทอร์เฟซ

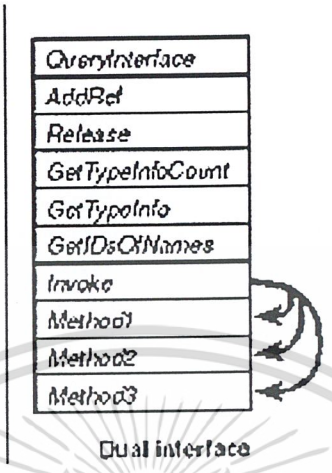
การทำงานของออโตเมชันนั้นเริ่มจากไคลเอนต์ทำการอ้างอิงถึงอินเทอร์เฟซ IDispatch จากนั้นไคลเอนต์จะถามออบเจกต์ว่า สนับสนุนเมธอดที่ไคลเอนต์ต้องการหรือไม่ โดยการเรียกใช้ GetIDsOfNames ซึ่งไคลเอนต์ต้องส่งชื่อของเมธอดเข้าไปเป็นอาร์กิวเมนต์ของ GetIDsOfNames ถ้าออบเจกต์นั้นสนับสนุนเมธอดที่ร้องขอเข้ามา GetIDsOfNames ก็จะคืนค่า DISPID

หลังจากไคลเอนต์ได้รับ DISPID แล้ว ก็จะเรียกใช้เมธอด Invoke โดยส่ง DISPID เข้าไปเป็นอาร์กิวเมนต์ แล้วเมธอด Invoke นี้ก็จะไปติดต่อกับเมธอดที่ถูกอ้างอิงถึงโดย DISPID (ซึ่งก็คือ เมธอดที่ไคลเอนต์ร้องขอมานั่นเอง) เพียงเท่านี้ ไคลเอนต์ ก็จะทำการเรียกใช้เมธอดที่ต้องการได้

ถึงแม้ว่าอินเทอร์เฟซ IDispatch จะทำให้ไคลเอนต์ที่เขียนด้วยภาษาพวกสคริปต์ สามารถเรียกใช้คอมโพเน็นต์ได้ แต่ก็มีข้อเสียคือ การเรียกใช้คอมโพเน็นต์ผ่านอินเทอร์เฟซ IDispatch นั้น ต้องมีการเรียกใช้เมธอดถึง 2 ครั้ง คือ ครั้งแรกเป็นการเรียกใช้ GetIDsOfNames ส่วนครั้งที่สองเป็นการเรียกใช้ Invoke ทำให้ใช้เวลาในการเอ็กซิคิวต์ (execute) นานกว่าอินเทอร์เฟซแบบ Custom มาก

4.19 ดูอัลอินเทอร์เฟซ (Dual Interface)

ดูอัลอินเทอร์เฟซ เป็นอินเทอร์เฟซที่ผสมกันระหว่างอินเทอร์เฟซ Custom กับอินเทอร์เฟซ IDispatch ดังแสดงในรูปที่ 4-7



รูปที่ 4-8 Dual Interface

คอมโพเน็นต์ที่รองรับการเรียกใช้โดยดูอัลอินเทอร์เฟซนี้ ทำให้ทั้งไคลเอนต์ที่เป็นเลทไบดิง และเออลีไบดิง (early binding) สามารถเรียกใช้คอมโพเน็นต์ได้โดยผ่านอินเทอร์เฟซตัวเดียวกัน โดยไคลเอนต์ที่เป็นเลทไบดิงก็จะใช้ในส่วนอินเทอร์เฟซ IDispatch ส่วนไคลเอนต์ที่เป็นเออลีไบดิง ก็จะใช้อินเทอร์เฟซ Custom

4.20 เทคนิคในการไบดิง (Binding Techniques)

เทคนิคในการไบดิง มี 3 แบบ ได้แก่

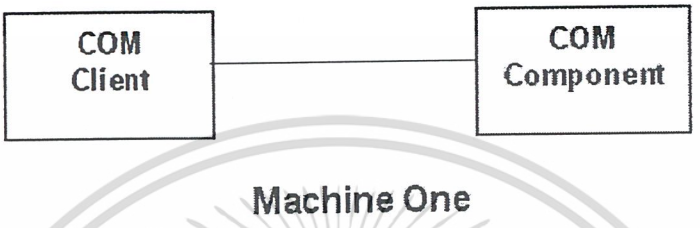
- *เลทไบดิง (Late binding)* ใช้เมื่อไคลเอนต์ไม่มีไทป์ไลบรารี (type library) และเรียกใช้ออบเจ็กต์ผ่านอินเทอร์เฟซ IDispatch ได้เพียงอย่างเดียว แต่เลทไบดิงนี้มีข้อเสียคือไคลเอนต์ต้องรู้มาก่อนว่าเมธอดที่ต้องการเรียกใช้นั้นอยู่ในออบเจ็กต์ตัวไหน มิฉะนั้นแล้วจะไม่สามารถเรียกใช้เมธอดนั้นได้เพราะไม่มีการเช็คไทป์ (type) ขณะ คอมไพล์ เหมือนกับการไบดิงวิธีอื่น นอกจากนี้แล้วประสิทธิภาพของเลทไบดิงนี้ก็ต่ำสุด เพราะมีการเรียกใช้เมธอดของอินเทอร์เฟซ IDispatch ถึง 2 ครั้ง (GetIDsOfNames, Invoke) จึงจะทำการเรียกใช้ออบเจ็กต์ได้
- *เออลีไบดิง (Early binding)* ใช้เมื่อไคลเอนต์เรียกใช้ออบเจ็กต์ผ่านอินเทอร์เฟซ IDispatch ได้อย่างเดียว แต่มีไทป์ไลบรารีซึ่งจะทำให้ไคลเอนต์รู้ว่า DISPID ขณะ คอมไพล์ จึงไม่ต้องเรียกใช้เมธอด GetIDsOfNames แต่ไคลเอนต์ก็ยังต้องเรียกใช้เมธอด Invoke อยู่ดี การไบดิงแบบเออลีไบดิงนี้จะเร็วกว่าเลทไบดิง แต่ก็ช้ากว่า vTable binding
- *vTable ไบดิง (vTable binding)* เป็นการไบดิงที่ดีที่สุด เร็วที่สุดเมื่อเทียบการไบดิงอีก 2 วิธี โดยจะใช้เมื่อไคลเอนต์มีไทป์ไลบรารี และคอมโพเน็นต์รองรับการเรียกใช้เมธอดทั้ง

อินเทอร์เฟซแบบ IUnkonwn อย่างเดียว หรือดูอัลอินเทอร์เฟซ (แต่ต้องไม่รองรับอินเทอร์เฟซ แบบ IDispatch อย่างเดียว)

บทที่ 5
เทคโนโลยี DCOM

5.1 พื้นฐานของ DCOM

DCOM ขยายความสามารถ transparent ของ COM โดย COM object ที่มีอยู่แล้วสามารถใช้งานแบบ remote ได้โดยไม่ต้องแก้ไข code ใดๆ



รูป 5-1 การเรียกใช้ COM Component ภายในเครื่องเดียวกัน

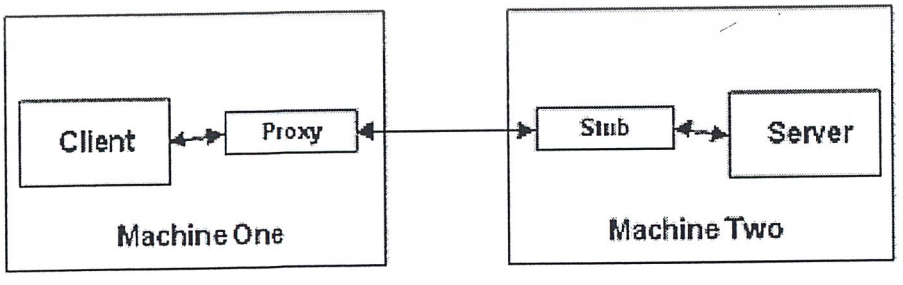
5.2 สถาปัตยกรรมของ DCOM

จากรูปด้านบน ไคลเอ็นต์จะติดต่อเซิร์ฟเวอร์ในเครื่องเดียวกัน สำหรับ DCOM แล้ว ไคลเอ็นต์สามารถอยู่บนอีกเครื่องหนึ่งได้ โดยการโปรแกรมยังคงเดิม เพียงแต่ว่า object reference นั้นถูกส่งผ่านข้ามเครื่อง การซ่อนรายละเอียดของการสื่อสารระยะไกลเรียกว่า location transparency

5.3 พร็อกซี่และสตับ

Proxy และ Stub จะถูกใช้เมื่อมีการสื่อสารระหว่างไคลเอ็นต์และ object ที่อยู่คนละเครื่อง

Proxy และ Stub ทำตัวเป็นตัวแทรกกลางในการเรียกฟังก์ชั่น (intercepting) ระหว่างไคลเอ็นต์และคอมโพเน็นต์ ซึ่งมันจะช่วยในการส่งข้อมูลข้ามเครือข่าย รวมถึงซ่อนรายละเอียดในการเรียกข้ามเครื่องอย่างอัตโนมัติ โดย proxy นั้นจะถูกสร้างขึ้นที่ฝั่งไคลเอ็นต์ ส่วน stub จะถูกสร้างที่ฝั่งเซิร์ฟเวอร์ ดังรูป



รูป 5-2 การเรียกใช้ COM Component ข้ามเครื่อง

ทั้ง proxy และ stub ต่างก็เป็น COM object โดย proxy จะจำลองตัวเองเป็น COM server โดยที่ไคลเอ็นต์ (ตัวเดิม)ไม่จำเป็นต้องรู้ว่าติดต่อกับ COM Server ตัวจริงหรือไม่ ส่วน stub จะจำลองตัวเป็นไคลเอ็นต์โดยที่ COM server ไม่รู้ตัวเช่นกัน

5.4 มาแชลลิ่ง (Marshalling)

เมื่อไคลเอ็นต์เรียกฟังก์ชั่นบน proxy object ตัว proxy จะนำ function identifier และ function parameters มาทำเป็น package ใน buffer และส่งไปยัง stub

จากนั้น stub จะถอดข้อมูลจาก package และเรียกฟังก์ชันบนคอมโพเน็นต์จริงๆ เมื่อทำงานเสร็จแล้ว stub จะนำ parameter ที่ต้องส่งคืน proxy (ByRef parameter) และค่าที่ต้อง return ของฟังก์ชั่นมาใส่ package ส่งคืน proxy เมื่อ proxy ได้รับ package ที่ส่งคืนมาจาก stub ตัว proxy จะถอดข้อมูลจาก package และคืนค่าแก่ไคลเอ็นต์ ซึ่งกระบวนการ package และ ถอด package นี้เรียกว่า marshalling

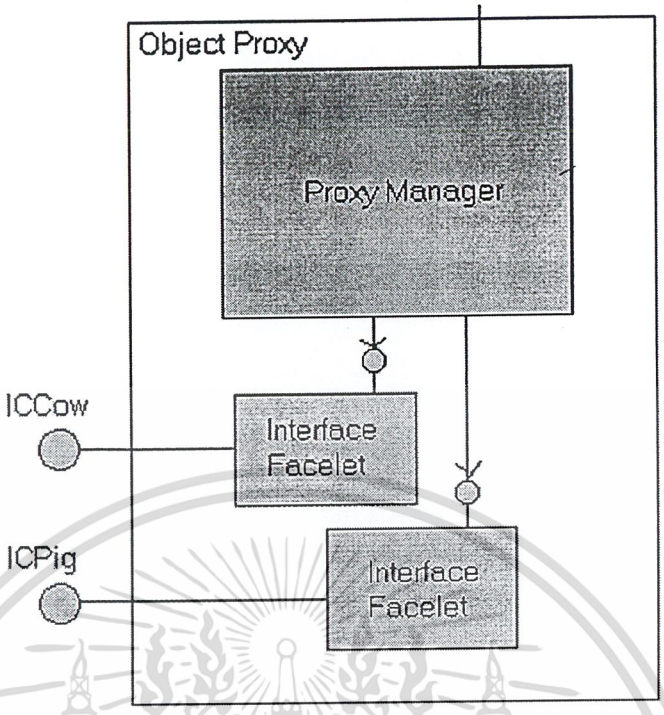
Marshalling เป็นคำศัพท์มาตรฐานเมื่อพูดถึงการที่ COM Component ทำการติดต่อกัน (ผ่าน proxy และ stub) ซึ่งแยกได้เป็น 2 ประเภท คือ Standard marshalling และ Custom marshalling โดย Standard marshalling จะเป็นรูปแบบในการสื่อสารแบบมาตรฐานซึ่งการทำงานจะเป็นไปอย่างอัตโนมัติโดย COM (VB COM จะใช้ standard marshalling เสมอ) ส่วน Custom marshalling เป็นรูปแบบในการสื่อสารที่ผู้ใช้กำหนดขึ้นเอง ซึ่งทำให้มีความยืดหยุ่นในการสื่อสารมากขึ้น (แต่ใน VB ไม่สามารถกำหนด Custom marshalling ขึ้นมาใช้ได้ เพราะต้องใช้ interface IMarshal ช่วยในการสร้าง)

5.5 วิธีการสื่อสารกันระหว่างพร็อกซี่และสตับผ่าน RPC Channels

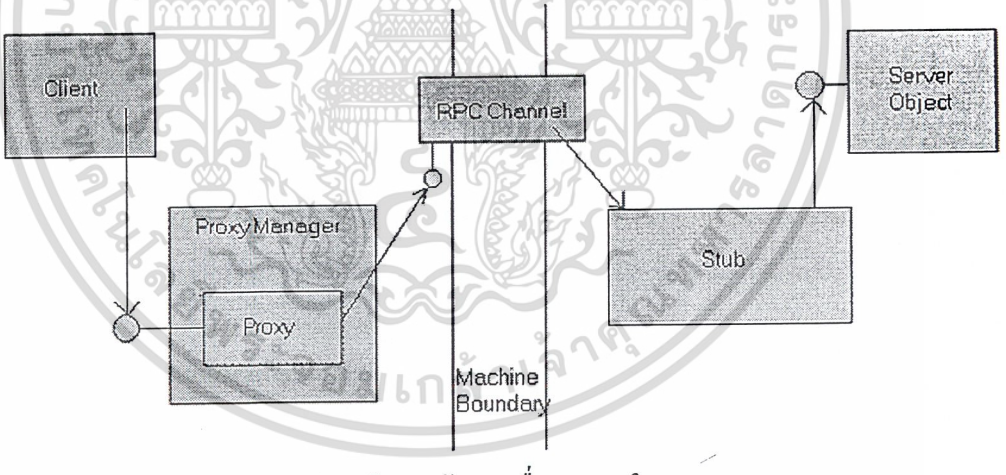
เมื่อไคลเอ็นต์ที่เป็น COM object เรียกใช้ฟังก์ชั่นของ COM object ที่อยู่บนเครื่องคอมพิวเตอร์อีกเครื่องหนึ่ง ตัว proxy object จะส่งข้อความไปยัง stub ผ่านช่องทาง RPC (Remote Procedure Call channel)

5.6 พร็อกซี่เมเนเจอร์ (Proxy Manager)

Proxy manager จะทำการสร้าง interface proxy object of facelet สำหรับแต่ละอินเตอร์เฟสที่ถูกเรียกใช้งานผ่าน QueryInterface และจะใช้เทคนิคที่เรียกว่า aggregatation ในการรวบรวมแต่ละ Interface Facelet ให้กลายเป็น Object Proxy เพียงตัวเดียว ดังรูป



รูป 5-3 การสร้าง Interface Facelet โดย Proxy Manager



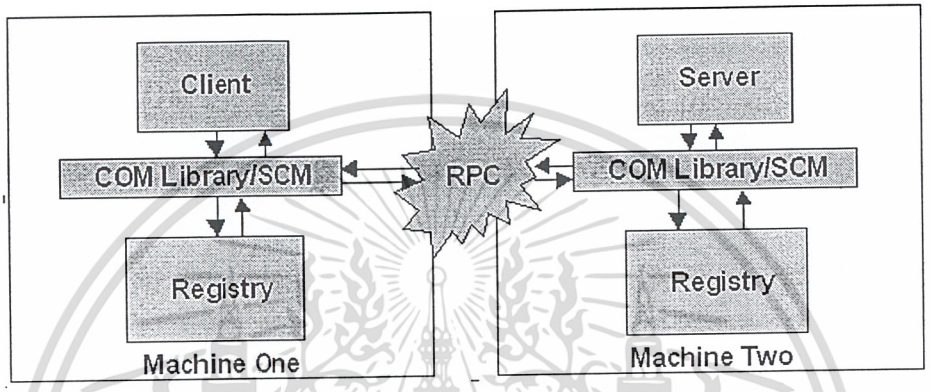
รูป 5-4 โครงสร้างการสื่อสารภายในของ DCOM

5.7 Microsoft's RPC Implementation (MS-RPC)

DCOM ใช้ RPC เป็นพื้นฐานหนึ่งในการสร้างบล็อกสำหรับการสื่อสารคอมโพเน็นต์แบบทางไกล แต่เนื่องจาก RPC นั้น มีพื้นฐานอยู่บนฟังก์ชั่นไม่ใช่ object ทางไมโครซอฟท์จึงสร้าง ORPC ขึ้นมาเพื่อให้สามารถใช้งานกับ object ได้

5.8 การสร้างรีโมตออบเจ็กต์ (Remote Object Creation)

เมื่อไคลเอ็นต์ทำการร้องขอให้ COM Library สร้างอินแสตนต์ของ COM server โดย COM library จะใช้ SCM (Service Control Manager) เพื่อพิจารณาว่าคอมโพเน็นต์นั้นอยู่ที่ใด โดยใช้ CLSID ที่ไคลเอ็นต์ส่งมาให้ ถ้าพบว่าคอมโพเน็นต์ไม่อยู่บนเครื่องเดียวกัน SCM บนเครื่องไคลเอ็นต์ จะทำการสื่อสารกับ SCM ที่อยู่บนเครื่องปลายทางเพื่อทำการร้องขอบริการในการสร้างคอมโพเน็นต์ที่แท้จริง เมื่อ server object ถูกสร้างขึ้นมาและพร้อมทำงาน object reference จะถูกส่งกลับมาผ่านทาง COM library จากนั้นไคลเอ็นต์นั้นก็จะสามารถเรียกใช้บริการของ component object ได้ผ่านทาง object reference นั้น



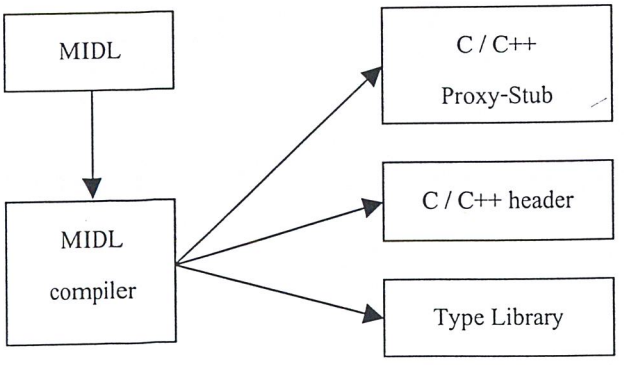
รูป 5-5 การทำงานภายในของ DCOM

5.9 การสร้าง DCOM เซิร์ฟเวอร์และไคลเอ็นต์

ก่อนอื่น จะต้องมีการสร้าง Type Library ขึ้นมาก่อน เพื่อให้ไคลเอ็นต์ใช้ในการอ้างอิง โดยใช้เครื่องมือตัวหนึ่งของไมโครซอฟท์ คือ MIDL Compiler ซึ่งจะได้ไฟล์ต่างๆ มา เช่น Proxy, Stub, Tlb เป็นต้น จากนั้นสำเนา type library file มาให้ไคลเอ็นต์เพื่อให้ไคลเอ็นต์รับรู้ว่า COM Server มีบริการใดให้ใช้บ้าง จากนั้นทำการ Implement Code ทั้งฝั่งเซิร์ฟเวอร์และไคลเอ็นต์

5.10 Microsoft Interface Definition Language (MIDL)

MIDL เป็นไฟล์ที่อธิบายลักษณะของ COM object โดยการเริ่มสร้าง COM จะต้องสร้าง MIDL และ compile ด้วย MIDL compiler



รูป 5-6 การเรียกใช้ MIDL Compilet

Proxy-Stub จะนำไป compile เป็น DLL สำหรับการทำ Marshalling

C / C++ header นำไป implement ทำ COM server สำหรับ visual C++

Type Library นำไป implement ทำ COM server สำหรับ visual Basic, visual J++ และ ภาษาอื่นๆ

5.11 ตัวอย่างการใช้งาน DCOM

5.11.1 ตัวอย่าง IDL Source

```
// KengExeServer2.idl : IDL source for KengExeServer2.dll
//
// This file will be processed by the MIDL tool to
// produce the type library (KengExeServer2.tlb) and marshalling code.

import "oaidl.idl";
import "ocidl.idl";
        [
                object,
                uuid(F8E231C0-B2A0-4189-848C-50D0C6932E2F),
                dual,
                helpstring("IWork Interface"),
                pointer_default(unique)
        ]
        interface IWork : IDispatch
        {
                [id(1), helpstring("method Echo")] HRESULT Echo([in] BSTR* in, [out , retval] BSTR* out);
                [id(2), helpstring("method Deposit")] HRESULT Deposit([in] int amount);
                [id(3), helpstring("method Withdraw")] HRESULT Withdraw([in] int amount);
                [id(4), helpstring("method GetBalance")] HRESULT GetBalance([out, retval] int * amount);
                [id(5), helpstring("method GetComputerName")] HRESULT GetComputerName([out , retval] BSTR* out);
                [id(6), helpstring("method Commit")] HRESULT SaveState();
                [id(7), helpstring("method Rollback")] HRESULT RestoreState();
        };
```

```
[
        uuid(CCB41427-0838-4AAC-A9FC-0ED05811FAC2),
        version(1.0),
        helpstring("KengExeServer2 1.0 Type Library")
]
library KENGEXESERVER2Lib
{
        importlib("stdole32.tlb");
        importlib("stdole2.tlb");
        [
                uuid(960026A3-DD31-424A-913F-8FD0E464921B),
                helpstring("Work Class")
        ]
        coclass Work
        {
                [default] interface IWork;
        };
};
```

5.11.2 ตัวอย่างการ Implement COM server ด้วยภาษา C++

ตัวอย่าง Header file ภาษา C++

```
MIDL_INTERFACE("F8E231C0-B2A0-4189-848C-50D0C6932E2F")
   IWork : public IDispatch
   {
   public:
      virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE Echo(
         /* [in] */ BSTR __RPC_FAR *in,
         /* [retval][out] */ BSTR __RPC_FAR *out) = 0;

      virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE Deposit(
         /* [in] */ int amount) = 0;

      virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE Withdraw(
         /* [in] */ int amount) = 0;
```

```
    virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE GetBalance(
        /* [retval][out] */ int __RPC_FAR *amount) = 0;

    virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE GetComputerName(
        /* [retval][out] */ BSTR __RPC_FAR *out) = 0;

    virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE SaveState( void) = 0;

    virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE RestoreState( void) = 0;
};
```

นำมา implements ด้วย Coclass

```
class ATL_NO_VTABLE CWork :
    public CComObjectRootEx<CComSingleThreadModel>,
    public CComCoClass<CWork, &CLSID_Work>,
    public ISupportErrorInfo,
    public IDispatchImpl<IWork, &IID_IWork, &LIBID_KENGEXESERVER2Lib>
{
public:
    CWork()
    {
        iBalance=0;
        iBakBalance=0;
    }

DECLARE_REGISTRY_RESOURCEID(IDR_WORK)
DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CWork)
    COM_INTERFACE_ENTRY(IWork)
    COM_INTERFACE_ENTRY(IDispatch)
    COM_INTERFACE_ENTRY(ISupportErrorInfo)
END_COM_MAP()
```

```
// ISupportsErrorInfo
        STDMETHOD(InterfaceSupportsErrorInfo)(REFIID riid);


// IWork
public:
        STDMETHOD(SaveState)();
        STDMETHOD(RestoreState)();
        STDMETHOD(GetComputerName)(/*[out , retval]*/ BSTR* out);
        STDMETHOD(GetBalance)(/*[out, retval]*/ int * amount);
        STDMETHOD(Withdraw)(/*[in]*/ int amount);
        STDMETHOD(Deposit)(/*[in]*/ int amount);
        STDMETHOD(Echo)(/*[in]*/ BSTR* in, /*[out , retval]*/ BSTR* out);
private:
        int iBalance ;
        int iBakBalance;
};
// ATL wizard code
STDMETHODIMP CWork::Echo(BSTR* in, BSTR* out)
{
        // TODO: Add your implementation code here
        *out = *in;
        return S_OK;
}
STDMETHODIMP CWork::Deposit(int amount)
{
        // TODO: Add your implementation code here
        iBalance += amount;
        return S_OK;
}
STDMETHODIMP CWork::Withdraw(int amount)
{
        // TODO: Add your implementation code here
        if(amount > iBalance)
```

```
        {
                return Error("not enough money");
        }
        iBalance -= amount;
        return S_OK;
}
STDMETHODIMP CWork::GetBalance(int *amount)
{
        // TODO: Add your implementation code here
        *amount = iBalance;
        return S_OK;
}
STDMETHODIMP CWork::GetComputerName(BSTR *out)
{
        // TODO: Add your implementation code here
        CComBSTR p;
        char tmp[10];
        unsigned long x = 10;
        ::GetComputerName(tmp,&x);
        p.Append(tmp);
        *out = p.Detach();
        return S_OK;
}
STDMETHODIMP CWork::SaveState()
{
        // TODO: Add your implementation code here
        iBakBalance=iBalance;
        return S_OK;
}
STDMETHODIMP CWork::RestoreState()
{
        // TODO: Add your implementation code here
        iBalance=iBakBalance;
        return S_OK;
```

}

จากนั้นทำการ Build จะได้ executable file ออกมา

5.11.3 การสร้าง DCOM ไคลเอ็นต์

สำหรับ visual C++ มี 2 ทางเลือก อ

- implement client โดยอิงกับ interface header ที่สร้างจาก MIDL compiler
- import type library เพื่อสร้าง interface header

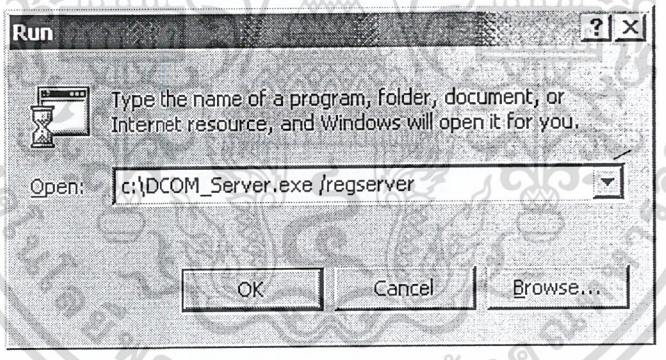
สำหรับ visual basic จะ import type library เพื่อ implement client

สำหรับ C# (.NET Client) จะ import type library เพื่อสร้างเป็น .NET class wrapper เพื่อทำให้สามารถทำงานในสภาพแวดล้อมของ .NET ได้

เมื่อสร้างเสร็จแล้ว ทำการ Build ให้ได้ executable file ออกมา

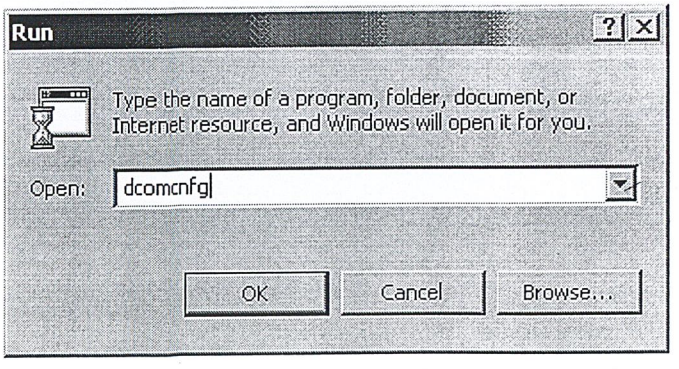
5.12 วิธีติดตั้ง DCOM ทางฝั่งไคลเอ็นต์

1. นำไฟล์ของ COM Server (ในที่นี้ คือ DCOM_Server.exe) มาเก็บไว้ที่เครื่องของไคลเอ็นต์เพื่อทำการ Registration เข้าสู่ Registry โดยพิมพ์ข้อความที่ RUN (สมมุติว่า นำไฟล์มาเก็บที่ c:\ ) ดังรูป



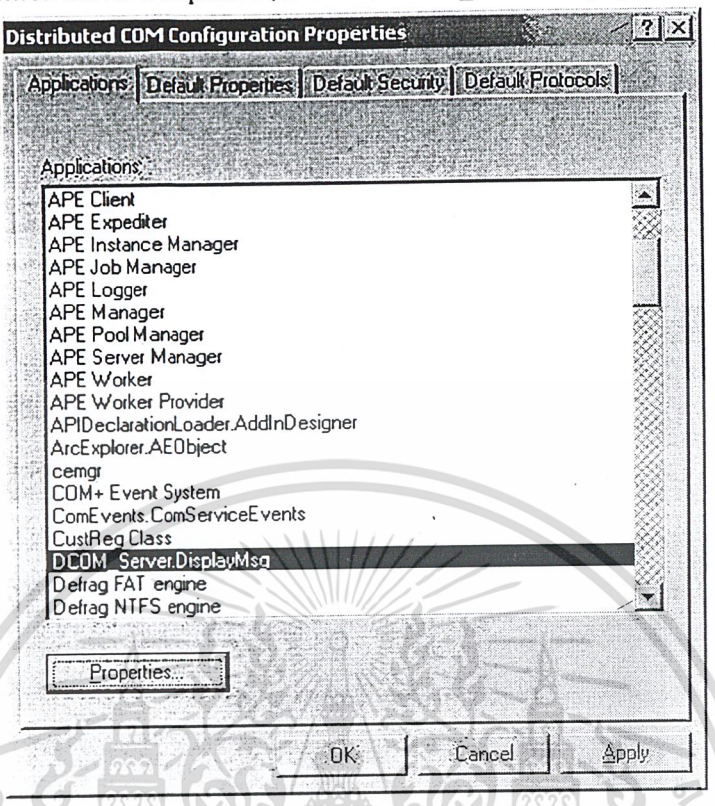
รูป 5-7 การเรียกใช้ลงทะเบียน COM Component

2. หลังจากนั้น ให้รันโปรแกรม dcomcnfg.exe ขึ้นมาดังรูป



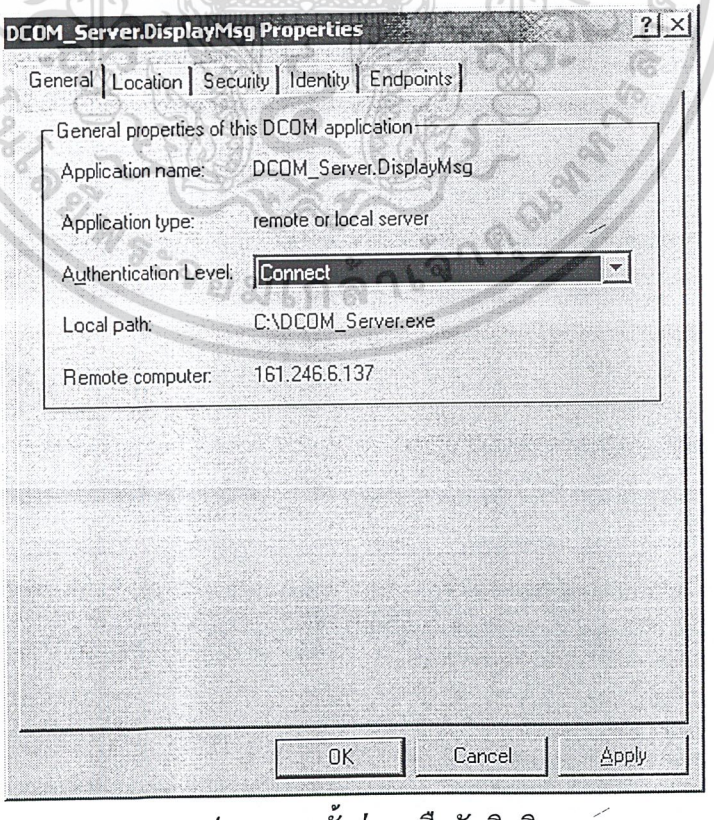
รูป 5-8 การเรียกใช้ dcomcnfg.exe

3. จากนั้นให้เลือก Server Component (ในที่นี้คือ DCOM_Server.DisplayMsg) ดังรูป



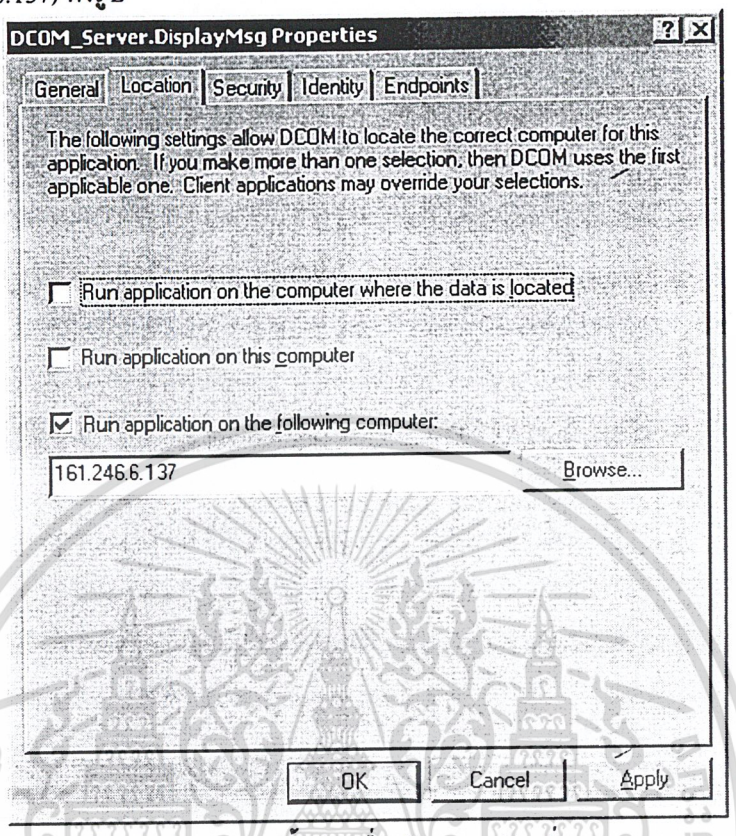
รูป 5-9 หน้าจอของ dcomcnfg

4. คลิ๊กเลือกที่ Properties แล้วตั้งค่าดังรูป เพื่อให้เกิดการตรวจสอบสิทธิการใช้งานเมื่อมีการเชื่อมต่อ



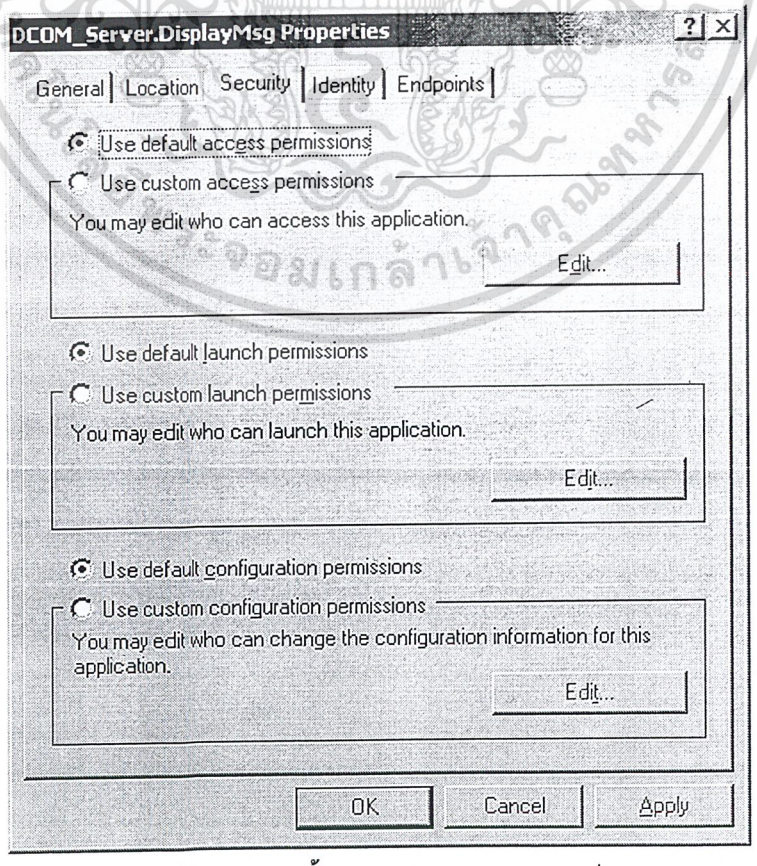
รูป 5-10 การตั้งค่าการยืนยันสิทธิ

5. เลือกไปที่แท็ป Location เพื่อกำหนดให้ทำการสร้าง object ที่เครื่องเซิร์ฟเวอร์ (ในที่นี้ คือ 161.246.6.137) ดังรูป



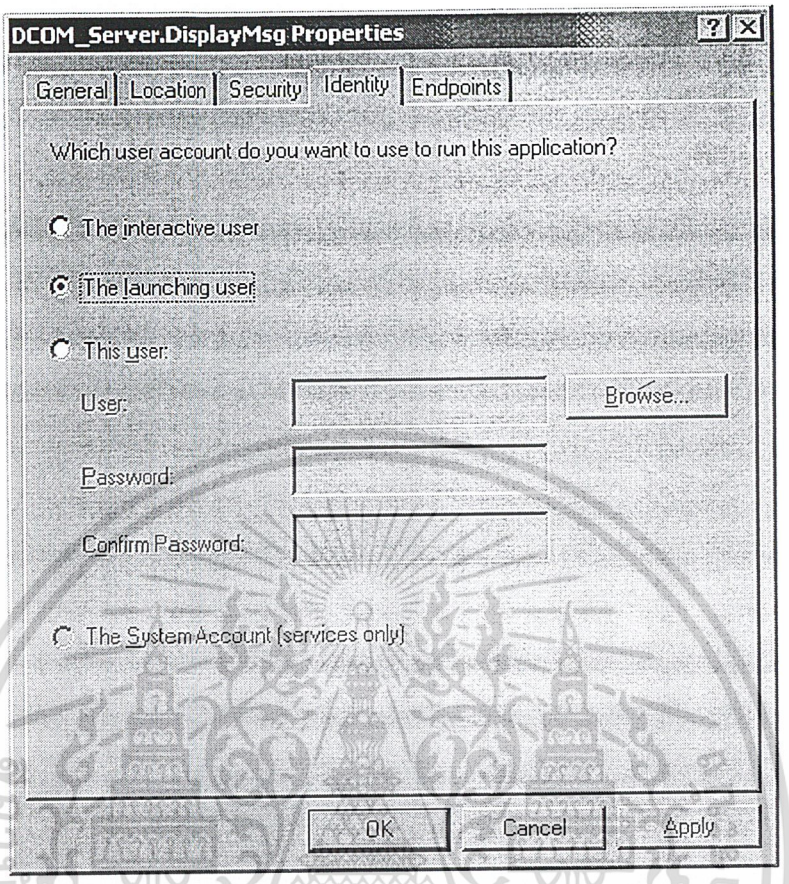
รูป 5-11 การตั้งค่าเครื่องรันแอพพลิเคชั่น

6. เลือกไปที่แท็ป Security เพื่อกำหนดมาตรการรักษาความปลอดภัย ดังรูป



รูป 5-12 การตั้งค่าการเข้าถึงคอมโพเน็นต์

7. เลือกไปที่แท็ป Identity เพื่อกำหนดบัญชีผู้ใช้ที่ทำการรันโปรแกรม ดังรูป

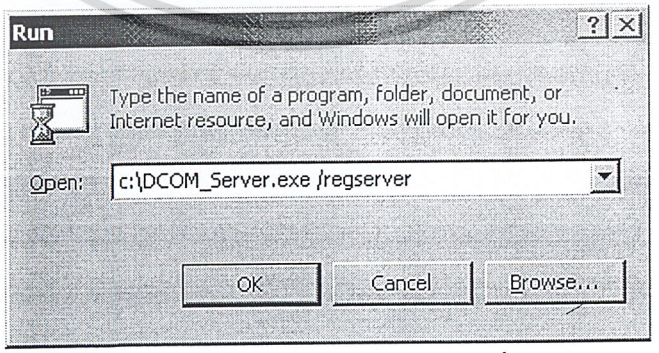


รูป 5-13 การกำหนดบัญชีผู้ใช้ที่สามารถใช้งานคอมโพเน็นต์ได้

8. คลิ๊ก OK 2 ครั้ง เพื่อออกจากโปรแกรม

5.13 วิธีติดตั้ง DCOM ทางฝั่งเซิร์ฟเวอร์

พิมพ์ข้อความ ดังรูป (สมมุติว่า ไฟล์อยู่ที่ c:\ )



รูป 5-14 การลงทะเบียน COM Component ฝั่งเซิร์ฟเวอร์

Tactics

เนื่องจาก การทดสอบนี้ใช้ Microsoft Windows 2000 จึงเกิดปัญหาด้าน Security ขึ้น จึงต้องตั้งค่า UserName และ Password ของทั้งเครื่องไคลเอ็นต์และเซิร์ฟเวอร์ให้เหมือนกัน จึงจะสามารถทำงานได้

บทที่ 6
เทคโนโลยี COM+

6.1 COM+

COM+ รวมหน้าที่สองอย่างไว้ด้วยกันคือ

- สถาปัตยกรรมพื้นฐานการโปรแกรมมิ่ง สำหรับการสร้างซอฟต์แวร์คอมโพเน็นต์ซึ่งสถาปัตยกรรมนี้ถูกนิยามโดยข้อกำหนดของ COM
- บริการของคอมโพเน็นต์ซึ่งเป็นส่วนของรันไทม์เอ็นไวรอนเมนต์ ทำให้สามารถสร้างซอฟต์แวร์ที่มีความสามารถมากขึ้น

คอมโพเน็นต์ที่ทำงานในเอ็นไวรอนเมนต์ข้างต้นเรียกว่า คอนฟิกคอมโพเน็นต์ (Configured Component)

ส่วนคอมโพเน็นต์อื่นๆ ที่ไม่ได้ใช้ความสามารถดังกล่าวเรียกว่า อันคอนฟิกคอมโพเน็นต์ (Unconfigured Component) โดยทำงานภายใต้กฎมาตรฐานของ COM

สถาปัตยกรรมพื้นฐานการโปรแกรมมิ่งของ COM+ จัดเตรียมรูปแบบการสร้าง คอมโพเน็นต์ซอฟต์แวร์ไว้ให้ แต่โดยมุมมองของนักพัฒนานั้นยังไม่เพียงพอต่อการพัฒนางานในองค์กร เพราะนักพัฒนาต้องสร้าง คอมโพเน็นต์ทางธุรกิจที่เป็นส่วนหนึ่งของระบบที่มีขนาดใหญ่ ซึ่งมักใช้วิธี การทำงานแบบไคลเอ็นต์/เซิร์ฟเวอร์ หรือ Three-tier ทำให้ต้องใช้ความพยายามมากแม้เป็นแค่การสร้างคอมโพเน็นต์อย่างง่าย เพราะจุดประสงค์ของคอมโพเน็นต์ทางธุรกิจนั้นต้องการต้องการความแข็งแกร่งและปลอดภัย ถ้าไคลเอ็นต์หลายตัวเชื่อมต่อกับคอมโพเน็นต์พร้อมๆกันนั้น นักพัฒนาต้องมั่นใจว่าเฉพาะไคลเอ็นต์ที่มีสิทธิเท่านั้นที่สามารถใช้งานคอมโพเน็นต์ได้ ปัญหาอีกประการคือ scalability ถ้าคอมโพเน็นต์ถูกใช้งานพร้อมกันโดยผู้ใช้ที่มีปริมาณมาก

ระบบขนาดใหญ่ต้องรับมือกับความล้มเหลวในระหว่างการทำงานที่ซับซ้อนและเกี่ยวข้องกับเซิร์ฟเวอร์ฐานข้อมูล โดยไคลเอ็นต์อาจต้องการส่งข้อมูลจากเครื่องตนเองไปเก็บที่เซิร์ฟเวอร์ฐานข้อมูล ถ้ามีการล้มเหลวระหว่างการทำงานดังกล่าว จะไม่สามารถมั่นใจในความถูกต้องของฐานข้อมูลได้ถ้าไม่ได้ใช้ทรานส์แอ็กชันโพรโตคอล (Transaction protocol) จะเห็นได้ว่าปัญหาหลักของคอมโพเน็นต์ซอฟต์แวร์ คือ รูปแบบพื้นฐาน ที่นักพัฒนาต้องรับมือกับการอิมพลีเมนต์การทำงานมากมายเหล่านี้ ซึ่งมีความเกี่ยวข้องน้อยมากกับจุดประสงค์ของตัวแอพพลิเคชันเอง

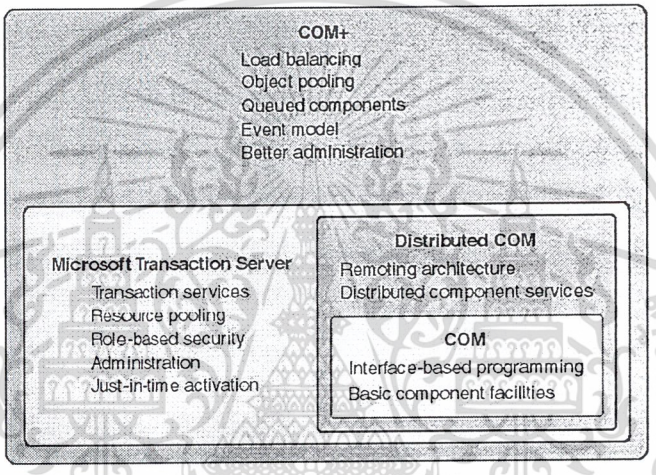
ไมโครซอฟท์ตระหนักว่าการสร้างระบบ Server-side ที่แข็งแกร่งต้องใช้ความพยายามมาก เช่น งานที่เกี่ยวกับเธรด, การทำงานพร้อมกัน, การรักษาความปลอดภัย, ความแข็งแกร่ง เป็นสิ่งที่สำคัญต่อระบบแบบกระจาย (distributed system) และการพัฒนาซอฟต์แวร์ที่มีลักษณะแบบนี้ต้องใช้ความพยายามอย่างมากมาย

ไมโครซอฟท์จึงได้สร้าง Microsoft Transaction Server (MTS) ซึ่งเป็น รันไทม์ เอ็นไวรอนเมนต์ตัวแรกของวินโดวส์ที่จัดหาบริการเหล่านี้แก่คอมโพเน็นต์ ต่อมารันไทม์เอ็นไวรอนเมนต์ และบริการของคอมโพเน็นต์ของ MTS ได้พัฒนาเป็น COM+

COM+ จัดหารันไทม์เอ็นไวรอนเมนต์ที่รับมือกับปัญหาของระบบ server-side ทำให้นักพัฒนาสามารถสร้าง in-process คอมโพเน็นต์ที่ทำงานตามหน้าที่ที่เกี่ยวกับธุรกิจจริงๆ ได้ง่าย อย่างไรก็ตาม in-process คอมโพเน็นต์ใดๆ ก็สามารถทำงานใน COM+ รันไทม์เอ็นไวรอนเมนต์ได้ แต่คอมโพเน็นต์ที่ไม่ได้ถูกออกแบบมาให้ทำงานในเอ็นไวรอนเมนต์ เช่น executable คอมโพเน็นต์จะไม่สามารถใช้ประโยชน์จาก COM+

6.2 บริการของคอมโพเน็นต์

บริการของ COM+ คอมโพเน็นต์จัดหาอิมพลีเมนต์มาตรฐานของบริการที่นักพัฒนา คอมโพเน็นต์ใช้บ่อยครั้ง ทำให้สามารถใส่ใจในปัญหาทางธุรกิจมากขึ้น

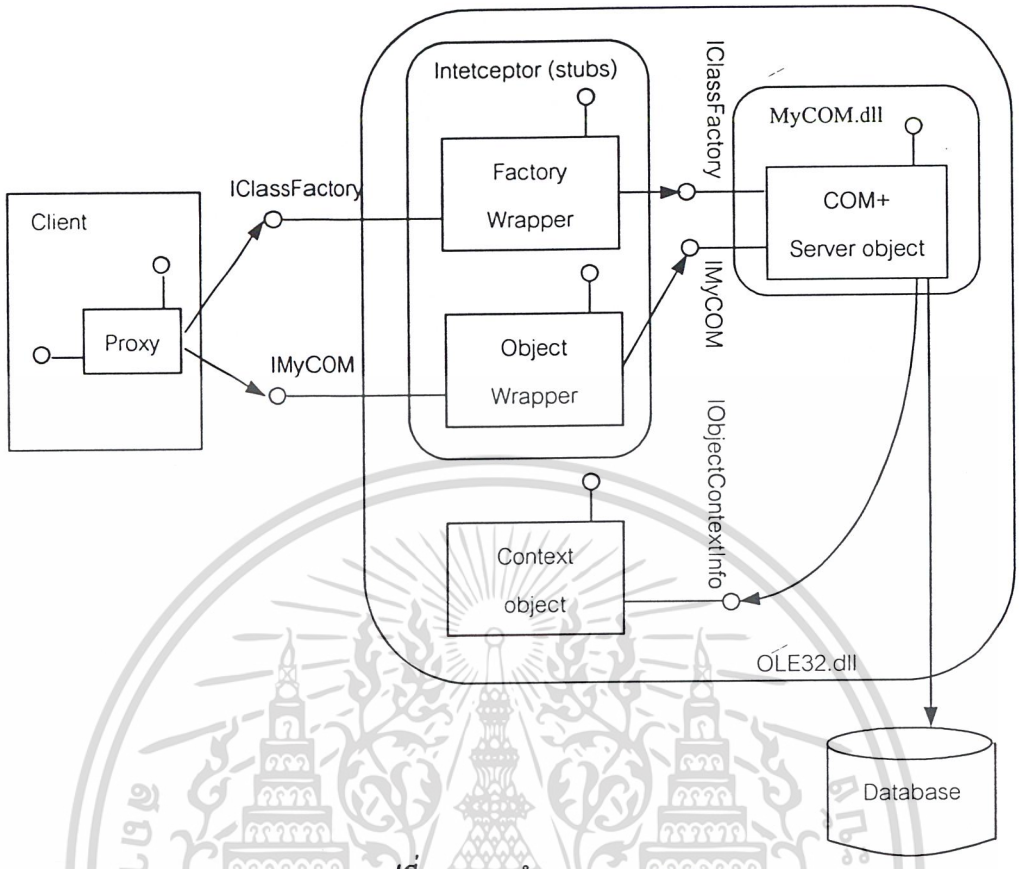


รูปที่ 6-1 วิวัฒนาการจาก COM สู่ COM+

6.3 รูปจำลองของ COM+

รูปจำลองของ COM+ แสดงดังรูปที่ 6-2 ซึ่งประกอบด้วย

1. COM+ surrogate โพรเซส (DLLHOST.EXE)
2. COM+ คลาส โหลดดิง และ รีโมต เฟรมเวิร์ก (OLE32.DLL)
3. คอนเท็กซ์ออบเจ็กต์ และ wrapper proxies สำหรับแต่ละคอมโพเน็นต์
4. COM+ เซิร์ฟเวอร์คอมโพเน็นต์
5. COM+ ไคลเอ็นต์
6. ระบบสนับสนุนต่างๆ ได้แก่ service control manager (SCM), Microsoft Transaction Coordinator (MS-DTC), Microsoft Message Queue (MSMQ), COM-Transaction Integrator (COM-TI)



รูปที่ 6-2 การทำงานของ COM+

คอมโพเน็นต์ที่ทำงานภายใต้การควบคุมของ COM+ surrogate เรียกว่า COM+ คอมโพเน็นต์ซึ่งคอมโพเน็นต์เหล่านี้ถูกดีพลอย โดยทำงานใน COM+ surrogate (DLLHOST.EXE) ซึ่งถูกจัดการโดย COM+ class loading และรีโมตเฟรมเวิร์ก (OLE32.DLL)

เช่นเดียวกับ COM คอมโพเน็นต์ที่ออบเจ็กต์ต้องอิมพลีเมนต์ IClassFactory ที่ทำหน้าที่เป็นแหล่งผลิตอินสแทนซ์ใหม่ของคอมโพเน็นต์

COM+ แทรก interceptor (ประกอบด้วย Factory Wrapper และ Object Wrapper) ระหว่าง COM+ คอมโพเน็นต์จริงซึ่งถูกจัดการโดยเฟรมเวิร์ก กับไคลเอ็นต์ของคอมโพเน็นต์นั้น

เมื่อไคลเอ็นต์เรียกใช้งาน COM+ คอมโพเน็นต์แล้ว wrapper (Factory และ Object) ก็จะเข้ามาแทรก และใช้วิธีการจัดการอินสแทนซ์ที่เรียกว่า Just In Time Activation (JITA) กับการเรียกใช้งานนั้น จากนั้น wrapper จะเรียกใช้งาน COM+ คอมโพเน็นต์จริงๆอีกทอดหนึ่ง นอกจากนี้แล้วข้อมูลคุณสมบัติของการดีพลอยคอมโพเน็นต์ได้แก่ รูปแบบทรานส์แอ็กชันและการรักษาความปลอดภัย จะดำเนินการใน wrapper ออบเจ็กต์ นี้ด้วย

สำหรับทุก COM+ คอมโพเน็นต์จะมีคอนเท็กซ์ออบเจ็กต์ที่อิมพลีเมนต์อินเทอร์เฟซ IObjectContextInfo ซึ่งคอนเท็กซ์ออบเจ็กต์มีหน้าที่เก็บรักษาข้อมูลของคอมโพเน็นต์ เช่น ข้อมูล transaction, การรักษาความปลอดภัย และข้อมูลของการดีพลอย โดย COM+ คอมโพเน็นต์เรียกใช้งานคอนเท็กซ์ออบเจ็กต์ โดยผ่านอินเทอร์เฟซ IobjectContextInfo

6.4 สิ่งที่ต้องคำนึงถึงในการสร้างมิดเดิ้ลเทียร์แอพพลิเคชั่น

การสร้าง Middle-tier application จะมีสิ่งที่ต้องคำนึงถึงได้แก่

1. Scalability

ปัจจัยที่ทำให้ scalability เป็นหัวข้อสำคัญที่ควรจะพิจารณาเป็นเพราะความนิยมในการใช้งานอินเตอร์เน็ต ซึ่งหนึ่งในจุดประสงค์หลัก COM+ ก็จัดการกับ scalability นั่นเอง

รูปแบบการใช้งานของอินเตอร์เน็ตจะต่างกับ enterprise application ขนาดใหญ่ตรงที่ เราไม่สามารถทราบจำนวนผู้ใช้งานได้ล่วงหน้า เมื่อเว็บไซต์ได้รับความนิยมมากก็มีโอกาสที่เว็บไซต์จะไม่สามารถรับมือกับโหลดที่มีขนาดมากในช่วงเวลาหนึ่งได้

นอกจากระยะเวลาที่ไซต์ไม่สามารถทำงานจะทำความเสียหายแล้ว ระยะเวลาในการตอบสนองต่อผู้ใช้งานก็เป็นปัญหาที่สำคัญอีกด้วย

ทางแก้หนึ่งคือการเพิ่มจำนวนของเมนเฟรม แต่ก็เกิดคำถามว่าควรจะใช้จำนวนเท่าไรเพราะสิ่งเหล่านี้มีต้นทุนสูง และปัญหาก็ไม่ได้แก้ด้วยฮาร์ดแวร์เท่านั้น ทางแก้ที่มีประสิทธิภาพประกอบด้วย software ที่เหมาะสมที่ทำงานบนฮาร์ดแวร์ที่ถูกที่สุด ซึ่งคำตอบที่ชัดเจนมากก็คือ PC นั่นเอง

COM+ ช่วยการ scalability ด้วย just-in-time activation, object pooling, load balancing

2. Reliability

แอพพลิเคชันที่ให้บริการแก่หลายผู้ใช้งานไม่ว่าจะเป็นเว็บหรือเอ็นเตอไพรส์ขนาดใหญ่มักจะเป็นงานที่มีความสำคัญและต้องการความเชื่อถือได้ในการทำงาน ซึ่งความล้มเหลวของระบบการทำงานแบบกระจาย เช่นฮาร์ดแวร์เสีย เป็นสิ่งที่หลีกเลี่ยงไม่ได้ ซึ่งทางแก้ไขคือทำอย่างไรเมื่อมีการเสียของจุดหนึ่งในระบบจะไม่ทำให้ระบบทั้งหมดล้มเหลวและทำให้ข้อมูลที่สำคัญหายไป

อีกมุมมองของความน่าเชื่อถือคือ การลดข้อผิดพลาดของซอฟต์แวร์เพราะค่าเสียหายของ distributed enterprise นั้นสูงกว่า desktop application มาก

COM+ สนับสนุนการพัฒนาระบบที่น่าเชื่อถือโดยเตรียมบริการที่จำเป็นต่างๆ, transaction, component load balancing

3. Complexity

ความซับซ้อนของระบบทำให้เสียค่าใช้จ่ายและเวลามาก ซึ่งการใช้งานเทคโนโลยีซอฟท์แวร์ใหม่ก็ทำให้ต้องเรียนรู้ API มากมาย

COM+ ทำให้การเขียนโปรแกรมง่ายขึ้น เช่น COM+ สนับสนุน automatic transaction ช่วยการทำงานแบบ distributed transaction โดยไม่ต้องเพิ่มโค้ดพิเศษเลย, COM+ มี security ที่ง่าย ซึ่งสนับสนุนการทำ synchronization ที่ง่ายขึ้น

6.5 Declarative, Attribute-Based Programming

หัวใจของการเขียนโปรแกรมของ COM+ คือรูปแบบของการประกาศความต้องการโดยใช้แอตทริบิวต์ซึ่ง COM+ runtimeจะจัดการกับความต้องการนี้เอง

ข้อดีคือความสะดวกของโปรแกรมเมอร์ที่จะประกาศแอตทริบิวต์ซึ่งง่ายกว่าการเขียนโปรแกรมที่ซับซ้อนซึ่งง่ายต่อการเกิดความผิดพลาด ในทางกลับกันกับโค้ดที่ทำโดยระบบปฏิบัติการซึ่งมีความน่าเชื่อถือมากกว่า

บริการพื้นฐานของ COM+ นั้นยากที่จะอิมพลีเมนต์ให้ถูกต้องด้วยตนเอง และเรายังสามารถปรับแต่งค่าแอตทริบิวต์โดยไม่ต้องแก้ไขโค้ดเลย

1. อพาร์ตเมนต์ ในรูปแบบของเขียนโปรแกรมแบบใช้การประกาศ

ตัวอย่างของการใช้แอตทริบิวต์ คือ การใช้งานฟังชันของ COM apartment ที่ช่วยให้เขียนโค้ดที่ทำงานพร้อมๆ กันโดยที่ไม่ต้องเขียนโค้ดจัดการ synchronization เองเลย

SINGLE-THREAD APARTMENT

สามารถใช้งาน synchronization แบบอัตโนมัติแก่คลาสโดยประกาศให้คลาสมี apartment เธรดโมเดลโดยกำหนดค่า "Apartment" แก่ เธรดโมเดล ในรีจิสตรีของ CLSID หมายความว่าเมื่อสร้างออบเจ็กต์ของคลาส มันจะอยู่ใน STA ของตนเอง ถ้าผู้ใช้ทำงานในเธรดเดียวกับออบเจ็กต์ก็จะสามารถใช้งานอินเทอร์เฟซได้ตรงๆ โดยถ้าเธรดอื่นพยายามที่จะเข้ามาใช้งานในขณะที่เจ้าของ อพาร์ตเมนต์ กำลังใช้งานอยู่ เธรดอื่นๆจะต้องทำงานผ่านพร็อกซี่ตามกฎการทำงานของ COM

พร็อกซี่ดังกล่าวใช้แมสเสจคิวของวินโดวส์ในการทำ synchronization แบบอัตโนมัติ ซึ่งเธรดที่สองจะต้องรอจนกว่าเธรดแรกจะทำงานเสร็จ จุดสำคัญคือพร็อกซี่ทำให้ COM สามารถ "แทรก" การใช้งานเมธอดเพื่อการทำsynchronization

โครงสร้างของเธรด synchronization ดังกล่าวกระทำโดยไม่ต้องเขียนโค้ดในแอพพลิเคชั่นชันเลย

2 การประกาศความต้องการทรานส์แอ็กชัน

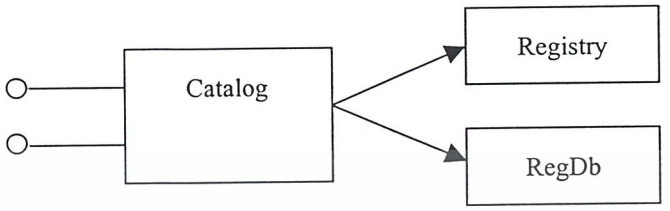
COM+ ขยายความคิดของการโปรแกรมแบบแอตทริบิวต์ไปอีกมาก ซึ่งทรานส์แอ็กชันก็เป็นตัวอย่างในการใช้งานแอตทริบิวต์ของCOM+

ถ้าเกิดความต้องการในการสร้างคอมโพเน็นต์ที่ทำงานกับบัญชีในธนาคารที่เก็บข้อมูลใน SQL Server ซึ่งการจัดการทรานส์แอ็กชัน ทำโดยรีซอร์สแมเนเจอร์ เราสามารถประกาศ "Transaction support" แก่คอมโพเน็นต์ของเราแบบ "Required" เมื่อคอมโพเน็นต์ทำงาน COM+ runtime จะรับทราบว่าคอมโพเน็นต์ของเราต้องการทรานส์แอ็กชันและจะแทรกโค้ดที่เรียกว่า "interceptor" ที่จะทำรายการทรานส์แอ็กชันให้

ผลลัพธ์ที่เกิดขึ้นคือการรับประกันว่าการทำงานจะเสร็จสมบูรณ์หรือไม่ก็ไม่เกิดขึ้นเลย โดยกระบวนการทั้งหมดไม่ต้องเขียนโค้ดเลย

6.6 COM+ Catalog

COM มาตรฐานจะเก็บข้อมูลลงในรีจิสเตอร์ แต่ COM+ ต้องการข้อมูลการปรับแต่งที่มากกว่า ข้อมูลเหล่านี้จะถูกแยกเก็บในฐานข้อมูลที่เรียกว่า RegDb ซึ่งฐานข้อมูลนี้ไม่สามารถเข้าถึงได้โดยตรง การเข้าถึงข้อมูลเหล่านี้ต้องกระทำผ่าน ระบบออบเจ็กต์ที่เรียกว่า Catalog Manager หรือเรียกสั้นๆ ว่า Catalog ที่จัดบริการการเข้าใช้ RegDb และรีจิสตรีดังรูปด้านล่าง

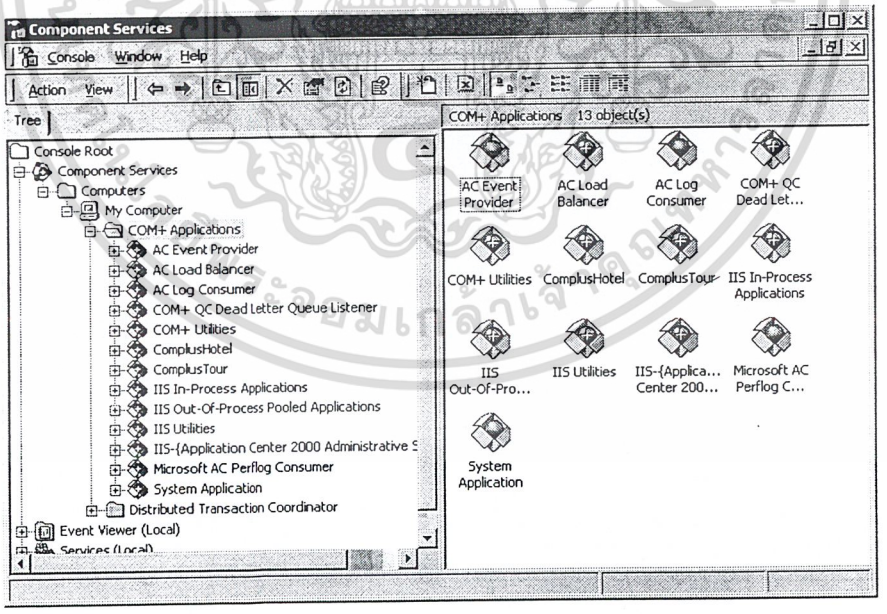


รูปที่ 6-3 COM+ catalog

Catalog Administration object สามารถเข้าถึงได้โดยโปรแกรมและสคริปต์ ซึ่งเปิดโอกาสให้ควบคุมการทำงานได้อย่างอัตโนมัติ

1. Component service Snap-in

วินโดวส์ 2000 มีเครื่องมือ จัดการแบบ Snap-in ซึ่งทำงานอยู่บน Catalog บางครั้งเรียกว่า COM+ Explorer โดยสามารถใช้งานเครื่องมือดังกล่าวผ่านทางเมนู Start | Programs | Administrative Tools | Component services ดังรูป



รูปที่ 6- 4 COM+ Explorer

เราสามารถปรับค่าแอตทริบิวต์ของคอมโพเน็นต์ด้วย COM+ Explorer โดยเลือกคอมโพเน็นต์ที่ต้องการและคลิกขวาเลือกแถบคุณสมบัติ

6.7 แนวทางของ COM+

บริการต่างๆของ COM+ เกิดขึ้นได้เพราะโครงสร้างพื้นฐานของ COM+ ที่สนับสนุนให้บริการต่างๆ

6.7.1 แนวทางของ COM

COM จัดลำดับชั้นในรูปแบบ server/ class/ interface/ method

Class ของ COM คือ Concrete implementation ของ abstract data type แบบ interface

Interface มีอย่างน้อย 1 เมธอดซึ่งแสดงถึงความสามารถของอินเทอร์เฟซ และมี signature ที่อธิบายพารามิเตอร์และการคืนค่า ใช้เป็นข้อตกลงในการใช้งานอินเทอร์เฟซ

Object เป็นอินแสตนซ์ของคลาสและอาจจะมีข้อมูลภายในตัวเอง ซึ่งข้อมูลนี้เป็นไพรเวตสามารถเข้าถึงโดยเมธอดของมันเองเท่านั้น

อพาร์ตเมนต์ แบ่งออบเจ็กต์เป็นกลุ่ม โดยเป็นกลุ่มที่ใช้งาน concurrency แบบเดียวกัน ซึ่งอพาร์ตเมนต์ที่ออบเจ็กต์อาศัยอยู่เป็นเป็นส่วนหนึ่งที่ใช้เป็น ID ของออบเจ็กต์

ในขณะที่ class, interface, method เป็นตัวแทนของโครงสร้างทางโลจิคอล ในส่วนบนสุดของลำดับชั้น COM คือ server ซึ่งเป็นส่วน physical จริงๆ ลักษณะเป็นไบนารีโมดูลที่บรรจุอย่างน้อย 1 คลาส และอาจอยู่ในรูปแบบ EXE หรือ DLL

6.7.2 แนวทางของ COM+

COM+ ยังคงรักษาแนวคิดของ class/ interface/ method และ อพาร์ตเมนต์ แต่แนวคิดของserver ไม่สำคัญอีกต่อไป เพราะคลาสใน COM+ ต้องเป็นแบบ DLL ถ้าต้องการใช้งานเป็น EXE server แล้ว COM+ จะใช้งานคอมโพเน็นต์ใน Surrogate มาตรฐานเรียกว่า DLLHOST.EXE

6.7.2.1 แอพพลิเคชัน

COM+ เริ่มแนวคิดที่เรียกว่า application ซึ่งเป็นกลุ่มของคลาสที่ใช้งานคุณสมบัติต่างๆ ร่วมกัน เช่น ความปลอดภัย, การระบุตัวตน ดังนั้นลำดับชั้นของ COM+ คือ application/class/interface

6.7.2.2 คอมโพเน็นต์

ในไมโครซอฟท์ แพลตฟอร์มเอสดีเค (Platform SDK) ได้อธิบายความหมายของคอมโพเน็นต์ว่าเป็นหน่วยของไบนารี่โค้ดที่สามารถสร้างไบนารี่ออบเจ็กต์ รวมถึงมีความสามารถในการลงทะเบียนตนเองด้วย

จุดสำคัญ คือคอมโพเน็นต์มีชื่อเอกลักษณ์ด้วย CLSID, ProgID และลงถูกทะเบียนในรีจิสตรี

คุณสมบัติในการเป็น COM+ คอมโพเน็นต์คือ

1. คอมโพเน็นต์เป็น COM class ที่อิมพลีเม็นต์ IUnknown

2. คอมโพเน็นต์เป็น DLL และมีโครงสร้างสำหรับการสร้าง object, CLSID, class factory และ DLL ต้องมีฟังชัน DllGetClassObject, DllCanUnloadNow
3. คอมโพเน็นต์มีความสามารถในการลงทะเบียนตนเองด้วยการสนับสนุน DllRegisterServer และ DllUnegisterServer
4. คอมโพเน็นต์ต้องมีไทป์ไลบรารีที่บรรจุใน DLL
5. คอมโพเน็นต์ต้องสามารถมาแชลอินเตอร์เฟซได้ โดยดูอัลอินเตอร์เฟซ (Dual interface) จะมาแชลโดย Automation และต้องจัดหาพร็อกซี่/สตับสำหรับ custom interface

สำหรับทุกๆ คลาสที่สร้างด้วย ActiveX DLL โดย Visual Basic และ ATL จะมีคุณสมบัติดังกล่าวครบถ้วน

6.7.2.3 คอมโพเน็นต์ที่ถูกปรับแต่งและ ไม่ถูกปรับแต่ง

คอมโพเน็นต์ที่มีคุณสมบัติครบถ้วนตามด้านบนสามารถใช้งาน COM+ service ได้แต่จะไม่เป็นไปในทันทีจนกว่าจะมีการติดตั้งบน COM+ application

คอมโพเน็นต์สามารถเพิ่มโดยวิธีการใช้เครื่องมือและการโปรแกรม เมื่อคอมโพเน็นต์ถูกติดตั้งจะเรียกว่าคอมโพเน็นต์ที่ถูกปรับแต่ง (configured component) ส่วนคอมโพเน็นต์ที่ไม่ถูกติดตั้งจะเรียกว่าคอมโพเน็นต์ที่ไม่ถูกปรับแต่ง (unconfigured component)

6.7.3 ชนิดของ COM+ Application

พื้นฐานของ COM+ application คือ server application ซึ่งอยู่ในโพรเซสของตนเอง คือ DLLHOST.EXE เมื่อเซิร์เวอร์ทำงานผิดพลาดจะไม่ทำให้ไคลเอ็นต์โพรเซสเสียหาย เพราะอยู่กันคนละโพรเซส และไคลเอ็นต์สามารถใช้งานผ่านทาง DCOM โดย server application มีความสามารถในการแทรกบริการอย่างเต็มรูปแบบ

library application ทำงานในโพรเซสของไคลเอ็นต์ ซึ่งไม่มีการป้องกันการทำงานผิดพลาด และจำกัดความสามารถในการแทรกบริการเท่านั้น จุดประสงค์หลักของ library application คือการนำโค้ดกลับมาใช้สำหรับ COM+ application มากกว่าที่จะบริการไคลเอ็นต์ตรงๆ

proxy application ประกอบด้วยข้อมูลที่จะติดตั้งบนไคลเอ็นต์ เพื่อให้มีความสามารถในการเรียกใช้งานจากระยะไกล

preinstalled COM+ application เป็น application ของ COM+ และ IIS ซึ่งไม่สามารถแก้ไขเปลี่ยนแปลงได้

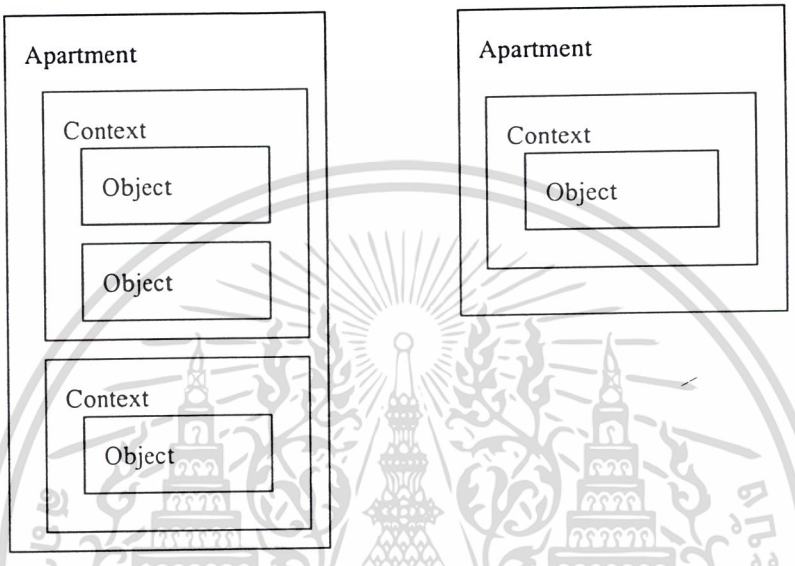
6.8 สถาปัตยกรรมของ COM+

COM+ ตั้งอยู่บน 3 แนวคิดคือ context, activation และ interception

6.8.1 Context

คอนเท็กซ์เป็นกลุ่มของออบเจ็กต์ในอพาร์เมนต์ที่มีความต้องการตอนรันไทม์เหมือนกัน ทุกๆ COM object ที่ถูกแอกทิเวตจะต้องสัมพันธ์กับ 1 คอนเท็กซ์เท่านั้น และคอนเท็กซ์จะอาศัยใน 1 อพาร์ตเมนต์เท่านั้น

1 อพาร์ตเมนต์ สามารถมีได้หลายคอนเท็กซ์ และคอนเท็กซ์สามารถมีได้หลายออบเจ็กต์ ดังนั้นออบเจ็กต์ที่อยู่คนละอพาร์ตเมนต์ก็ต้องอยู่คนละคอนเท็กซ์ด้วย ดังรูป



รูปที่ 6-5 แสดงความสัมพันธ์ระหว่างอพาร์ตเม็นต์ คอนเท็กซ์ และออบเจ็กต์

ทุกอพาร์ตเมนต์จะที่ default context เมื่อคอมโพเน็นต์ที่ไม่ถูกปรับแต่งถูกแอกทิเวต จะถูกนำเข้าไปอยู่ใน default context โดยทั่วไปแล้ว คอนเท็กซ์นี้จะไม่ได้ใช้งาน

สำหรับคอมโพเน็นต์ที่ถูกปรับแต่ง คุณสมบัติของคอนเท็กซ์ถูกกำหนดโดยแอตทริบิวต์ของคอมโพเน็นต์ที่เก็บไว้ใน catalog ซึ่งคอนเท็กซ์นี้เป็นกุญแจสำคัญของการใช้งาน COM+ service

1. Context Object

คอนเท็กซ์เป็นสิ่งที่ใช้อ้างอิงถึงกลุ่มของ COM object ซึ่งการเชื่อมต่อกับคอนเท็กซ์นั้นจริงๆ แล้วก็เป็น COM object เช่นกัน เรียกว่า "Context Object" ซึ่งจะเก็บข้อมูลของคอนเท็กซ์นั้นๆ

โค้ดในเมธอดสามารถเข้าถึงคอนเท็กซ์ออบเจ็กต์โดยใช้ API CoGetObjectContext (หรือ GetObjectContext ใน visual basic) ซึ่งคืนค่าเป็นพอยเตอร์ไปที่ interface IObjectContext จากนั้นสามารถใช้เมธอดของอินเทอร์เฟซดังกล่าวเพื่อแก้ไขคุณสมบัติของคอนเท็กซ์ออบเจ็กต์

2. Call Context

บางส่วนของคอนเท็กซ์ใช้ควบคุมเรื่องความปลอดภัย ซึ่งจะใช้ API CoGetCallContext ซึ่งจะคืนค่าเป็น ISecurityCallContext ซึ่งมีข้อมูลเกี่ยวกับความปลอดภัยในการใช้งานเมธอด

6.8.2 Activation

จุดสำคัญของการใช้งานคอนเท็กซ์นั่นคือ คอนเท็กซ์ใช้อธิบายสภาพแวดล้อมของออบเจ็กต์ไม่ใช่โครงสร้างข้อมูลภายในออบเจ็กต์

ออบเจ็กต์โดยตัวเองแล้วไม่มีคอนเท็กซ์ ซึ่งกระบวนการในการเชื่อมออบเจ็กต์กับคอนเท็กซ์เรียกว่า "Activation"

สภาพแวดล้อมของออบเจ็กต์คือคอนเท็กซ์และคอนเท็กซ์นี้มีความสำคัญมากในการพิจารณาพฤติกรรมของออบเจ็กต์ขณะรันไทม์

การนำออบเจ็กต์ให้มาอยู่ในสถานะพร้อมต้องใช้ 2 ขั้นตอนใน COM+

1. การสร้าง เมื่อออบเจ็กต์ถูกสร้างด้วย class factory เป็น COM ธรรมดา
2. activation เมื่อออบเจ็กต์ถูกเชื่อมเข้ากับคอนเท็กซ์แล้วก็พร้อมใช้งานได้

การทำลายออบเจ็กต์ก็มี 2 ขั้นตอนเช่นกัน

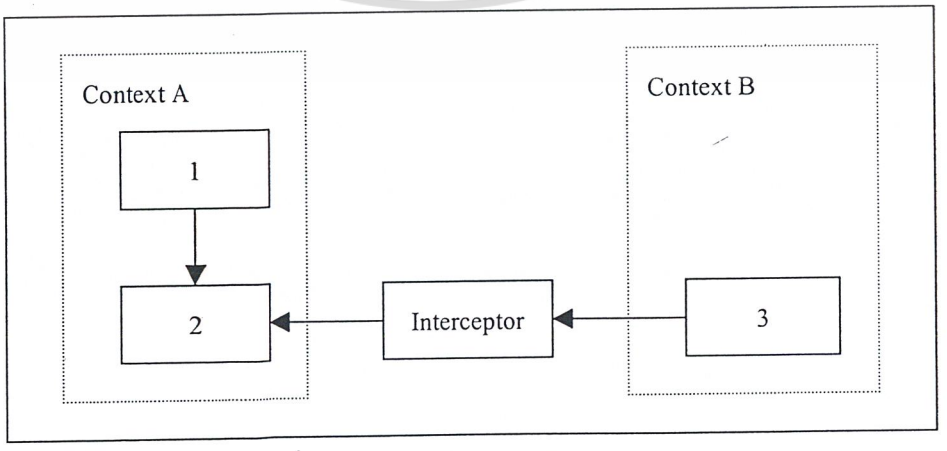
1. deactivate เป็นการกำจัดคอนเท็กซ์ออกจากออบเจ็กต์
2. ทำลายออบเจ็กต์

โดยปกติแล้ว activation จะเกิดขึ้นหลังจากการสร้าง และ deactivate จะเกิดก่อนการทำลาย แต่ COM+ มีความสามารถที่เรียกว่า object pooling ซึ่งสามารถเก็บออบเจ็กต์ที่ถูกสร้างไว้แล้วได้ เมื่อไคลเอ็นต์ต้องการ activate object ก็จะดึงออบเจ็กต์ออกมาจาก pool และเมื่อใช้งานออบเจ็กต์เสร็จก็จะ deactivate และใส่กลับเข้าไปใน pool

6.8.3 Interception

เมื่อออบเจ็กต์ถูก activate เข้ากับคอนเท็กซ์ การเรียกใช้งานออบเจ็กต์หนึ่งไปยังอีกออบเจ็กต์ที่อยู่ในคอนเท็กซ์เดียวกันจะไม่ต้องการการแทรกแซงจาก COM+ แต่ออบเจ็กต์ที่อยู่คนละคอนเท็กซ์จะมีสภาพแวดล้อมที่ไม่เหมือนกัน ดังนั้นเมื่อมีการเรียกข้ามขอบเขตของคอนเท็กซ์จะต้องมีการแทรกแซง (Interception) ของ COM+

รูปต่อไปนี้แสดงอพาร์ตเมนต์ที่มี 2 คอนเท็กซ์ ได้แก่ A และ B ซึ่งการเรียกจาก 3 ไปยัง 2 เป็นการข้ามขอบเขตของคอนเท็กซ์ ดังนั้นจึงต้องแทรกแซงโดยCOM+ โดยใช้ Interceptor



รูปที่ 6-6 เรียกข้าม Context ต้องการ interceptor

1. Interceptor

Interceptor คือพร็อกซีขนาดเล็กที่จัดการโดย COM+ รันไทม์เพื่อแก้ปัญหาการไม่เข้ากันของ context Interceptor จะมีอินเทอร์เฟซที่เหมือนกับออบเจ็กต์เป้าหมาย ซึ่งในมุมมองของไคลเอ็นต์ จะไม่ทราบว่าทำงานกับ Interceptor แทนที่จะเป็นออบเจ็กต์จริงๆ

Interceptor สามารถทำงานก่อนการเรียกออบเจ็กต์ และสามารถทำงานหลังการทำงานของออบเจ็กต์ก่อนที่จะคืนค่ากลับไปยังไคลเอ็นต์

Interceptor ถือว่าเป็นขนาดเล็กเพราะไม่จำเป็นต้องใช้ thread switch ซึ่งต่างจากพร็อกซีทั่วไปที่ต้องการ thread switch หรือแม้กระทั่ง process switch

2. interface pointer marshalling

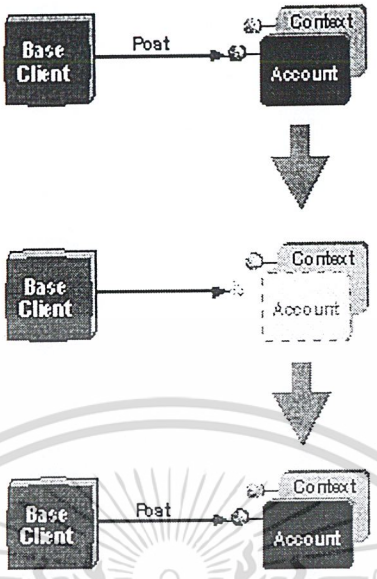
ถ้าเราผ่าน interface pointer เป็นพารามิเตอร์ของการเรียกเมธอดที่ข้ามคอนเท็กซ์ แล้ว COM+ จะแปลงพอยน์เตอร์เป็นพร็อกซีอย่างอัตโนมัติ และเมื่อมีการ call back มาจากข้างนอกก็จะต้องผ่าน interceptor

ถ้าต้องการแบ่งกันใช้งาน interface pointer ในลักษณะของตัวแปร global เราจะต้องรับผิดชอบในการ marshalling interface ด้วยตนเอง

6.8.4 Just-in-Time (JIT) Activation

ลักษณะที่สำคัญของ COM+ คือความสามารถในการขยาย middle-tier คอมโพเน็นต์เพื่อการรองรับการใช้งานไคลเอ็นต์พร้อมกันหลายพันคน โดยไคลเอ็นต์ที่พยายามจะสร้าง COM+ ออบเจ็กต์ จะได้รับ reference ของ wrapper ออบเจ็กต์ ซึ่งไม่ใช้ reference ของออบเจ็กต์ของคอมโพเน็นต์จริงๆ

COM+ ออบเจ็กต์จะเริ่มต้นทำงานที่สถานะ deactivated เมื่อไคลเอ็นต์เรียกเมธอดของออบเจ็กต์ที่มีสถานะ deactivated แล้ว COM+ จะ activate ออบเจ็กต์ให้อย่างอัตโนมัติ วิธีการนี้เรียกว่า just-in-time activation ทำให้ไคลเอ็นต์ได้ รับ reference ไปยัง ออบเจ็กต์ ได้โดยที่อาจจะยังไม่ใช้งานในทันทีทันใด



รูปที่ 6-7 Just-in-time activation

COM+ สามารถ deactivate ออบเจ็กต์ ได้แม้ว่าไคลเอ็นต์ยังมี reference ไปยังคอมโพเน็นต์นั้นอยู่ ซึ่งการ deactivate สามารถทำได้โดย release ทุก reference ของออบเจ็กต์ ทำให้ COM+ คอมโพเน็นต์ถูกทำลายไปเมื่อ reference count ลดลงถึงศูนย์

เมื่อไคลเอ็นต์ เรียกใช้งานออบเจ็กต์ที่อยู่ในสถานะ deactrivated แล้ว COM+ จะ reactivate ให้อย่างอัตโนมัติ ดังนั้นขณะที่ไคลเอ็นต์รู้สึกว่าทำงานกับออบเจ็กต์เดียวในความเป็นจริงอาจจะทำงานกับหลาย อินสแทนซ์ของ คลาส เดียวกันก็ได้

just-in-time activation ช่วยให้ COM+ คอมโพเน็นต์สามารถแบ่งปันทรัพยากรของเซิร์ฟเวอร์ใช้ได้อย่างเหมาะสม ยกตัวอย่าง ไคลเอ็นต์ ที่ใช้งานเพียง 10% ของช่วงชีวิตของออบเจ็กต์ เวลาที่เหลืออีก 90% นั้นด้วย just-in-time activation จะทำให้เซิร์ฟเวอร์สามารถใช้ทรัพยากรรองรับงานได้มากขึ้น

6.9 COM+ Threading Model

อพาร์ตเมนต์ จะรวมกลุ่มออบเจ็กต์ที่มีความต้องการ Concurrency เหมือนกัน ในวินโดวส์เอ็นที 4.0 มีอพาร์ตเมนต์ อยู่ 2 แบบคือ STA และ MTA

STA(Single-Threaded Apartments)

STAจะจัดลำดับการเข้าใช้งานออบเจ็กต์โดยใช้วินโดวส์แมสเสจคิวซึ่งเป็นบริการของวินโดวส์ที่สร้างและซ่อนไว้โดย STA ทำให้ไม่มีการเข้าใช้งานเมธอดของออบเจ็กต์ที่อยู่ใน STA พร้อมกัน ทำให้ไม่ต้องมีการปกป้องข้อมูลของอินแสตนซ์ในกรณีที่มีการเข้าถึงพร้อมกัน ซึ่งช่วยให้การเขียนโปรแกรมง่ายขึ้น

MTA(Multithreaded Apartment)

MTAไม่มีกลไกการจัดลำดับการเข้าถึง ดังนั้นหลาย ๆ เธรดในMTAสามารถเข้าถึงออบเจ็กต์เดียวกันได้พร้อมๆ กัน เธรดที่ทำงานใน MTA บางครั้งเรียกว่า free thread ดังนั้นเมธอดของคลาสที่เรียกโดย free thread จะต้องเป็น thread safe ซึ่งต้องการทำซิงโครไนเซชันเพื่อปกป้องข้อมูลในส่วนคริติคอลเซ็กชัน แต่ประสิทธิภาพที่ได้จะมากขึ้น

ในแต่ละโพรเซสสามารถมีได้หลาย STA แต่มีได้เพียง MTA เดียวเท่านั้น

DLL server ไม่สามารถเรียก CoInitializeExโดยตัวเองจึงต้องประกาศเธรดดิงโมเดลในรีจิสตรี ซึ่งค่าที่เป็นไปได้ของ โมเดลเธรด มีทั้งหมด 5 ค่าได้แก่

1) Single มีชนิดของ อพาร์ตเมนท์ เป็น STA ออบเจ็กต์ประเภทนี้จะไม่เป็น Thread-Safe ไม่ใช่แค่ตัวออบเจ็กต์เท่านั้นแต่ยังรวมถึง Class Factory สำหรับคลาส และฟังก์ชัน DllGetClassObject และ DllCanUnloadNow ก็เช่นกัน Single-Thread สามารถทำการเรียกใช้ทุก ๆ เมธอดที่อยู่ในเซิร์ฟเวอร์

2) อพาร์ตเมนต์ มีชนิดของอพาร์ตเมนท์เป็น STA ออบเจ็กต์ประเภทนี้จะสามารถป้องกันปัญหาของระบบ Multi-Thread ได้เฉพาะในระบบดับคลาสเท่านั้น เนื่องจาก Class Factory และ ฟังก์ชัน DllGetClassObject และ DllCanUnloadNow ในคลาสของมันเป็น Thread-Safe แต่ในระดับ อินแสตนซ์จะถูกจำกัด เนื่องจากการเข้าถึงแต่ละอินแสตนซ์จะถูกกำหนดให้เป็น Single-Thread เท่านั้น

3) Free มีชนิดของ อพาร์ตเมนท์ เป็น MTA ออบเจ็กต์ประเภทนี้สามารถป้องกันปัญหาของระบบ มัลติเธรด ทั้งสถานะต่ออินแสตนซ์ และสถานะต่อคลาสโดยใช้ Thread Synchronization Primitives เช่น Critical Section หรือ Mutex ออบเจ็กต์เหล่านี้สามารถทำงานในสถานะที่เป็น มัลติเธรดได้อย่างปลอดภัย

4) Both สามารถมีชนิดของอพาร์ตเมนท์ ได้ทั้ง STA และ MTA ออบเจ็กต์ประเภทนี้จะสร้างตัวเองอยู่ในอพาร์ตเมนท์เดียวกับไคลเอนต์ของมันซึ่งทำให้มีประสิทธิภาพสูงสุด และทำงานในเธรดเดียวกับผู้ที่เรียกมัน และเนื่องจากมันสามารถทำงานได้ในระบบที่เป็นมัลติเธรดมันจึงป้องกันปัญหาที่จะเกิดในระดับคลาส และ ระดับอินแสตนซ์

5) Neutral มีชนิดของอพาร์ตเมนท์เป็น TNA (Thread Neutral Apartment) ออบเจ็กต์ประเภทนี้จะทำงานในเธรดของผู้เรียก แต่อยู่ในอพาร์ตเมนท์ของมันเองแทนที่จะเป็นอพาร์ตเมนท์ของผู้เรียก เนื่องจาก เธรดที่อยู่ในอพาร์ตเมนท์ประเภทอื่นในโพรเซสเดียวกันสามารถเข้าถึง TNA ได้ทุกเวลา และออกจากอพาร์ตเมนท์เมื่อมันทำงานเมื่อการทำงานของเมธอดนั้นเสร็จ เนื่องจากอพาร์ตเมนท์ประเภทนี้สามารถทำการ Thread-Synchronization primitives ได้ภายในตัวของมันเอง หรือสามารถ Thread-Synchronize ได้ด้วยบริการของ COM+

6.10 ทรานส์แอ็กชันใน COM+

COM+ สนับสนุนการทำทรานส์แอ็กชันแบบกระจายสำหรับ COM คอมโพเน็นต์ โดยเราไม่ต้องเขียนโค้ดที่ซับซ้อน กระบวนการต่างๆ เป็นดังนี้

1. ปรับแต่งออบเจ็กต์ที่ทำงานภายใต้ COM+

2. ตั้งค่าแอตทริบิวต์ของออบเจ็กต์ว่าจะให้เป็นแบบ Requires a Transaction, Requires a New Transaction หรือ Supports Transactions
3. ใช้เมธอด SetComplete ของออบเจ็กต์คอนเท็กซ์ (Object Context) เมื่อพร้อมที่จะคอมมิททรานส์แอ็กชัน
4. ใช้เมธอด SetAbort ของออบเจ็กต์คอนเท็กซ์ (Object Context) เมื่อพร้อมที่จะโรลแบ็กทรานส์แอ็กชัน

1. สรุปความหมายของค่าต่างๆ ของทรานส์แอ็กชันแอตทริบิวต์ทั้งหมดของ COM+ ออบเจ็กต์ดังนี้

Requires a Transaction

ออบเจ็กต์นี้จะต้องทำงานภายใต้ทรานส์แอ็กชัน COM+ จะเริ่มต้นทรานส์แอ็กชันใหม่สำหรับออบเจ็กต์นี้ก็ต่อเมื่อ ไม่มีออบเจ็กต์อื่น ๆ ที่อยู่ในคอนเท็กซ์เดียวกัน เริ่มต้นทรานส์แอ็กชันเลย

Requires a New Transaction

ออบเจ็กต์นี้จะต้องทำงานภายใต้ทรานส์แอ็กชัน COM+ จะเริ่มต้นทรานส์แอ็กชันใหม่ให้เมื่อออบเจ็กต์นี้ถูกสร้างขึ้น

Supports Transaction

ออบเจ็กต์นี้จะทำงานภายใต้ทรานส์แอ็กชันก็ต่อเมื่อ ผู้สร้าง (Creator) ได้สร้างออบเจ็กต์ที่เริ่มต้นทรานส์แอ็กชันไว้ก่อนแล้ว ไม่เช่นนั้นออบเจ็กต์จะทำงานโดยปราศจากทรานส์แอ็กชัน

Does Not Support Transactions

ออปเจ็กต์นี้ไม่ต้องทำงานแบบทรานส์แอ็กชัน และ COM+ จะไม่รับออบเจ็กต์นี้ภายใต้ทรานส์แอ็กชันใหม่ (เป็นค่าดีฟอลต์)

Disabled

ออบเจ็กต์ไม่สนใจที่จะเลือกค่าใดๆ ในทรานส์แอ็กชันแอตทริบิวต์ ตัวเลือกนี้ทำให้คอมโพเน็นต์มีลักษณะเป็นคอมโพเน็นต์ที่ยังไม่ได้ตั้งค่าอะไรเลย

2. การเริ่มต้นทรานส์แอ็กชัน

หลังจากที่ตั้งค่าทรานส์แอ็กชันให้กับคอมโพเน็นต์ COM+ รันไทม์จะอ่านค่าทรานส์แอ็กชันแอตทริบิวต์เมื่อเริ่มสร้างออบเจ็กต์ หลังจากนั้นจะติดต่อกับทรานส์แอ็กชันโคออร์ดิเนเตอร์ที่เรียกว่า Distribute Transaction Coordinator (DTC) โดยจะบอกให้ DTC เริ่มต้นทรานส์แอ็กชันสำหรับออบเจ็กต์นี้ DTC จะเริ่มต้นทรานส์แอ็กชันให้กับออบเจ็กต์แล้วส่งค่า GUID ของทรานส์แอ็กชันกลับมา (DTC จะกำหนด GUID ให้กับทุกทรานส์แอ็กชัน) GUID นี้จะเก็บอยู่ในคอนเท็กซ์ของออบเจ็กต์นั้น เราสามารถใช้เมธอด IsInTransaction และ GetTransactionId ของอินเทอร์เฟซ IObjectContextInfo เพื่อดูว่าออบเจ็กต์นี้ทำงานอยู่ภายใต้ทรานส์แอ็กชันหรือไม่ และมี GUID ทรานส์แอ็กชันอะไร

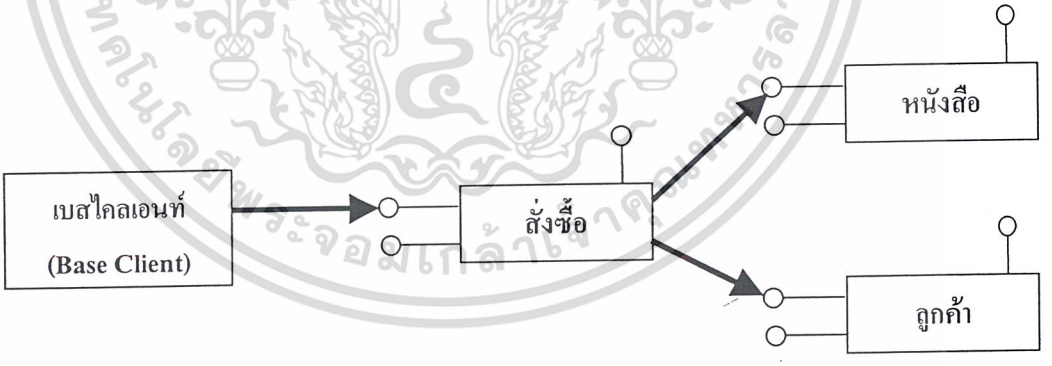
ออบเจ็กต์ที่เป็นตัวเริ่มต้นทรานส์แอ็กชันจะเรียกว่า ออบเจ็กต์ราก (Root Object) ของทรานส์แอ็กชัน ออบเจ็กต์รากมีความสำคัญมากเนื่องจากว่า COM+ รันไทม์จะไม่คอมมิททรานส์แอ็กชัน จนกว่าออบเจ็กต์รากจะถูกดีแอกติเวท (Deactived) ออบเจ็กต์อื่น ๆ ที่อยู่ภายใต้ทรานส์แอ็กชันเดียวกันจะเป็นเพียงตัวลงความเห็นว่าควรจะคอมมิททรานส์แอ็กชันหรือไม่ แต่ออบเจ็กต์รากเป็นตัวกำหนดว่าเมื่อไหร่ที่จะคอมมิททรานส์แอ็กชัน

เมื่อ COM+ ออบเจ็กต์ที่อยู่ภายใต้ทรานส์แอ็กชันทำการติดต่อกับรีซอร์สเมเนเจอร์ รีซอร์สเมเนเจอร์นั้นก็จะอยู่ในทรานส์แอ็กชันด้วย รีซอร์สเมเนเจอร์จะติดต่อกับ DTC เพื่อบอกว่าขณะนี้มันเป็นส่วนหนึ่งของทรานส์แอ็กชัน ถ้าออบเจ็กต์ติดต่อกับรีซอร์สเมเนเจอร์หลายตัว รีซอร์สเมเนเจอร์เหล่านั้นทั้งหมดก็จะอยู่ใน ทรานส์แอ็กชันเช่นกัน

เมธอด SetComplete เป็นการโหวตเพื่อคอมมิททรานส์แอ็กชัน ออบเจ็กต์รากเป็นตัวบ่งบอกว่าขณะนี้พร้อมคอมมิทแล้ว DTC จะทำงานโดยใช้โปรโตคอลทู-เฟสคอมมิท เพื่อคอมมิทหรือโรลแบ็กทรานส์แอ็กชันตามขั้นตอนที่ได้กล่าวมาแล้วขั้นต้น

3. ทรานส์แอ็กชันที่เกี่ยวข้องกับออบเจ็กต์มากกว่า 1 ตัว

ก่อนหน้านี้เราได้กล่าวถึง COM+ ออบเจ็กต์ตัวเดียวที่ทำงานในลักษณะที่เป็นทรานส์แอ็กชันและมีการติดต่อกับรีซอร์ทเมเนเจอร์ตั้งแต่สองตัวขึ้นไป แต่ในหลาย ๆ กรณีทรานส์แอ็กชันมักจะเกี่ยวข้องหรือใช้ออบเจ็กต์หลาย ๆ ตัว ตัวอย่างเช่นการสั่งหนังสือออนไลน์ สมมติว่าให้ออบเจ็กต์การสั่งซื้อเป็นออบเจ็กต์รากของทรานส์แอ็กชัน ออบเจ็กต์การสั่งซื้อมีการสร้างและเรียกใช้ออบเจ็กต์ลูกค้ากับออบเจ็กต์หนังสือดังรูปที่ 6-8



รูปที่ 6-8 ออบเจ็กต์หลาย ๆ ตัวทำงานอยู่ในทรานส์แอ็กชันเดียวกัน

เมื่อ COM+ ออบเจ็กต์ถูกสร้างโดย COM+ ออบเจ็กต์ด้วยกัน ออบเจ็กที่ถูกสร้างขึ้นใหม่จะมีลักษณะการทำงานแบ่งได้เป็น 3 ทาง คือ (1) อยู่ในทรานส์แอ็กชันเดียวกับออบเจ็กที่สร้างมัน (2) สร้างทรานส์แอ็กชันของตนเองขึ้นมาใหม่ (3) ไม่ทำงานเป็นทรานส์แอ็กชันเลย กรณีทั้งสามจะเกิดขึ้นอยู่กับการตั้งค่าทรานส์แอ็กชันแอตทริบิวต์ให้กับออบเจ็กต์ต่าง ๆ

ถ้าออบเจ็กต์สั่งซื้อและออบเจ็กต์ลูกค้าต่างมีทรานส์แอ็กชันเป็นของตัวเอง ออบเจ็กต์สั่งซื้ออาจคอมมิทงานของตน โดยที่ออบเจ็กต์ลูกค้าอาจทำงานแล้วเกิดปัญหาทำให้ต้องโรลแบ็กงานของตน จะเห็นได้ว่าออบเจ็กต์ทั้งสองอาจทำงานขัดแย้งกันได้ซึ่งเป็นเหตุการณ์ที่เราไม่ต้องการ หากออบเจ็กต์ทั้งคู่ทำงานอยู่ในทรานส์แอ็กชันเดียวกัน การคอมมิทหรือโรลแบ็กทรานส์แอ็กชันจะขึ้นอยู่กับออบเจ็กต์ทั้งหมดที่อยู่ในทรานส์แอ็กชันนั้น ออบเจ็กต์แต่ละตัวจะเสนอการคอมมิทโดยเรียกใช้เมธอด SetComplete ของอินเทอร์เฟส IObjectContext หรืออาจเสนอให้โรลแบ็กโดยใช้เมธอด SetAbort ของอินเทอร์เฟส IObjectContext เช่นเดียวกัน ถ้าออบเจ็กต์ทุกตัวเสนอให้คอมมิท ทรานส์แอ็กชันจึงจะคอมมิท หากมีตัวใดตัวหนึ่งเสนอให้โรลแบ็ก ทรานส์แอ็กชันจะต้องโรลแบ็ก สิ่งสำคัญประการหนึ่งคือ การเรียกใช้เมธอด SetComplete เป็นเพียงการเสนอว่าออบเจ็กต์นั้นยอมรับและพร้อมคอมมิท ทรานส์แอ็กชัน ออบเจ็กต์จะยังไม่ทราบว่าทรานส์แอ็กชันคอมมิทแล้วจริง ๆ หรือไม่ ดังนั้น COM+ ออบเจ็กต์ที่ทำงานในลักษณะที่เป็นทรานส์แอ็กชันไม่ควรสรุปว่าสถานะของตนหลังจากใช้เมธอด SetComplete ไปแล้วจะเป็นสถานะสุดท้ายที่ถูกต้องเสมอไปเมื่อทรานส์แอ็กชันคอมมิท ตัวอย่างเช่น ถ้าออบเจ็กต์ลูกค้าทำงานโดยหักเงินของบัญชีลูกค้าในฐานข้อมูล ออบเจ็กต์ลูกค้ายังไม่ควรเปลี่ยนแปลงสถานะยอดเงินจนกว่าทรานส์แอ็กชันจะคอมมิท เพราะหากเปลี่ยนแปลงแล้วออบเจ็กต์อื่น ๆ ที่ทำงานในทรานส์แอ็กชันเดียวเสนอให้โรลแบ็กทรานส์แอ็กชัน จะทำให้ออบเจ็กต์ลูกค้าอยู่ในสถานะที่ไม่ถูกต้องโดยเงินในบัญชีลูกค้าถูกหักออกไปโดยที่การสั่งหนังสือทำงานไม่สำเร็จ ในการป้องกันปัญหาดังกล่าว COM+ จะดีแอกติเวทออบเจ็กต์ทั้งหมดที่เกี่ยวข้องกับทรานส์แอ็กชันเมื่อ ทรานส์แอ็กชันคอมมิท การดีแอกติเวทออบเจ็กต์เป็นการเพิ่ม scalability ของระบบ แต่เหตุผลของในเรื่องของ ทรานส์แอ็กชันคือเพื่อความความถูกต้องของงานที่ปฏิบัติ

6.11 COM+ และ Scalability

6.11.1 Microsoft Clustering Technology

ทางแก้ปัญหาโดยทั่วไปเมื่อมีการเพิ่มโหลดการทำงานคือการเพิ่มฮาร์ดแวร์โดยการซื้อระบบที่ใหญ่ขึ้นมาแทนที่ระบบเดิม แต่วิธีนี้ใช้ค่าใช้จ่ายมากและยังไปติดอยู่ที่ขนาดที่ใหญ่ที่สุดของระบบเดี่ยวด้วย

ทางแก้อีกทาง คือมัลติโพรเซสเซอร์ ซึ่งมีอยู่ 2 แบบคือ

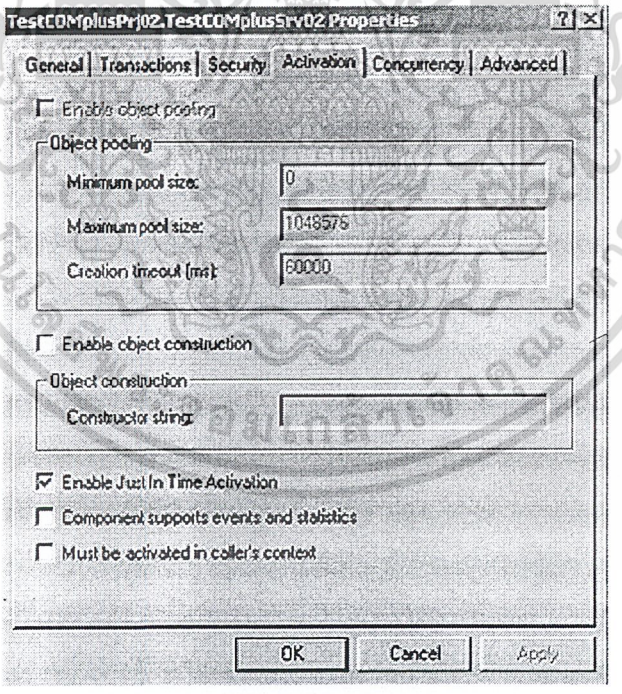
1. Tightly couple คือ Symmetric, shared-memory multiprocessor (SMP) ซึ่งระบบจะใช้งานระบบปฏิบัติการเพียงสำเนาเดียว และมีการจัดการการทำงานของเธรดต่างๆอย่างอัตโนมัติ แต่ SMP มีข้อจำกัดเนื่องจากการแบ่งกันใช้ช่องทางเดินของหน่วยความจำ (shared-memory bus)
2. Loosely couple คือ clusters ซึ่งมีความสเกลมากกว่าและยังทนทานต่อการเสียหายอีกด้วย เพราะclustersประกอบด้วยหลายโหนดที่อิสระต่อกัน จึงไม่มีการแบ่งกันใช้ช่องทางเดินของหน่วยความจำ และเมื่อโหลดใดเสียหายก็ไม่ทำให้ทั้งระบบเสียหายไปด้วย สำหรับ cluster แบบนี้เป็นแบบที่ใช้ในโครงงานนี้ สามารถอ่านรายละเอียดที่บท 8

6.11.2 Object Pooling

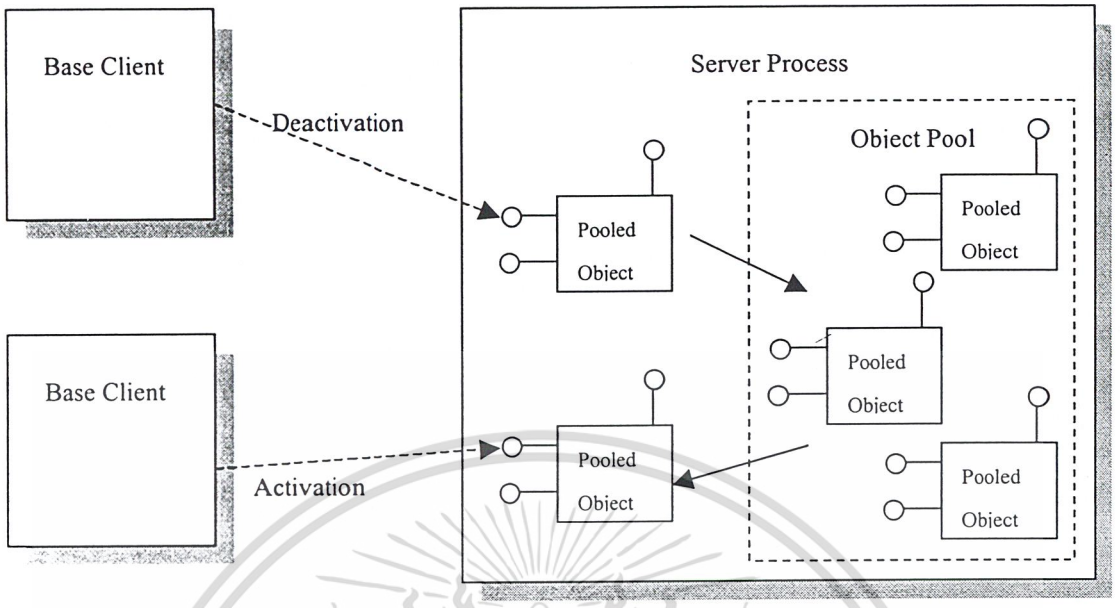
สำหรับแอพพลิเคชั่นที่ใช้ JIT Activation ในการ Activate และ Deactivate ออบเจ็กต์เป็นจำนวนมาก ๆ นั้นค่อนข้างจะเป็นวิธีที่สิ้นเปลืองที่จะต้องสร้างออบเจ็กต์ใหม่ ทุกๆ ครั้งที่มีการเริ่มต้น ทรานแซคชั่น Object Pool เป็นวิธีการที่ถูกนำมาใช้ในการแก้ปัญหาดังกล่าวโดยเมื่อไคลเอนต์จะทำการ Activate ออบเจ็กต์นั้น COM+ Runtime จะส่งอินแสตนซ์จาก Object Pool มาให้ถ้าหากมีออบเจ็กต์ชนิดนั้นอยู่ใน Object Pool แต่หากไม่มีก็จะทำการสร้างอินแสตนซ์ใหม่ขึ้นมา ไคลเอนต์สามารถใช้ออบเจ็กต์ได้นานเท่าที่มันต้องการและเมื่อไคลเอนต์ปล่อยออบเจ็กต์นั้น ออบเจ็กต์ก็จะไม่ถูกทำลายแต่จะถูกนำกลับไปไว้ที่ Object Pool เพื่อรอการเรียกใช้ครั้งต่อไป ซึ่งจะมีลักษณะดังรูปที่ 6-9

สำหรับการที่จะทำให้ออบเจ็กต์มีความสามารถที่เป็นออบเจ็กต์ที่สามารถใช้ใน Object Pool ได้นั้นก็สามารถจะเซตแอททริบิวต์ได้เหมือนกับบริการอื่น ๆ ของ COM+ Runtime โดยมีขั้นตอนดังนี้

1. คลิกขวาที่คอมโพเน็นต์ในโฟลเดอร์ Components ของแอพพลิเคชั่น
2. เลือก Properties... จากเมนู Context จะปรากฏไดอะลอก Component Properties
3. เลือกหน้า Activation
4. เซตเชคบ็อกซ์ Enable object pooling
5. กด OK



รูปที่ 6-9 ไดอะล็อก Component Properties



รูปที่ 6-10 Object Pooling

จากวิธีที่ได้กล่าวมานั้นเป็นวิธีที่จะเซตให้ออบเจ็กต์สามารถนำมาใช้ร่วมกัน (Pool) ได้แต่ก่อนที่เซตค่านั้นได้ตัวออบเจ็กต์จะต้องมีลักษณะดังนี้ด้วยคือ

1. ต้องอยู่ในอพารต์เมนต์ชนิด Thread Neutral หรือ Free-Thread
2. ต้องไม่มีสถานะ
3. ต้องไม่มี Thread-affinity
4. ต้องสามารถรวมกันได้ (aggregatable)

นอกจากจะเซตให้ออบเจ็กต์นั้นสามารถใช้งานร่วมกันได้แล้วยังสามารถกำหนดจำนวนอินแสตนซ์ของออบเจ็กต์ที่ต้องมีอย่างน้อยที่สุด และมากที่สุดเท่าที่จะมีได้ โดยหากออบเจ็กต์ที่มีอยู่ใน Object Pool ยังน้อยกว่าจำนวนที่ต้องมีอย่างน้อยที่สุด เมื่อมีการเรียกใช้จากไคลเอนต์ตัวใหม่ COM+ Runtime ก็จะสร้างอินแสตนซ์ใหม่ให้ทันที แต่หากมีอยู่เกินกว่าจำนวนที่น้อยที่สุดแล้วแต่ถูกใช้อยู่ทุกตัว และยังมีจำนวนที่น้อยกว่าจำนวนที่มากที่สุดเท่าที่จะมีได้แล้ว ก็จะสร้างอินแสตนซ์ใหม่ให้กับไคลเอนต์นั้น แต่ถ้าเกินขอบเขตที่มากที่สุดแล้ว ไคลเอนต์จะต้องรอจนกว่าจะมีออบเจ็กต์ว่าง แล้วจึงนำเอาออบเจ็กต์นั้นไปใช้งานได้

สำหรับ COM+ ออบเจ็กต์ที่ต้องการแสดงสถานะว่าสามารถที่จะถูกนำกลับไปที่ Pool ได้ หรือให้ทำการ Activate และ Deactivate เองต้องทำการอิมพลีเมนต์อินเตอร์เฟส IObjectControl โดยมีเมธอด Activate และ Deactivate ที่ใช้ในการ Activate และ Deactivate ออบเจ็กต์ตามลำดับ และ เมธอด CanBePooled ที่ใช้ในการบอก COM+ Runtime ว่าสามารถนำกลับไปใช้ได้หรือไม่ในขณะนั้น

6.12 โมเดลความปลอดภัยของ COM+ (COM+ Security Model)

COM+ มีกลไกที่ช่วยรักษาความปลอดภัย เรียกว่า COM+ Security ซึ่งช่วยให้ผู้พัฒนาไม่ยุ่งยากในการจัดการกับกลไกการรักษาความปลอดภัยของวินโดวส์หรือ RPC ซึ่งคอมโพเน็นต์ต่างๆ จะสามารถสืบทอดข้อดีจากบริการรักษาความปลอดภัยตัวใหม่ได้อย่างอัตโนมัติ โมเดลรักษาความปลอดภัยนี้ถูกสร้างขึ้นมาจาก Security Support Provider Interface

ไมโครซอฟต์วินโดวส์ 2000 สนับสนุนโมเดลรักษาความปลอดภัยแบบกระจาย โดยมุ่งประเด็นไปใน2 เรื่องหลัก คือ

- การพิสูจน์ตน (Authentication) โดยปกติการพิสูจน์ตนจะถูกเรียกใช้เมื่อผู้ใช้ล็อกอินเข้าสู่ระบบเน็ตเวิร์กหรือเรียกใช้บริการจากเครื่องอื่นและระบบต้องการรู้ว่าผู้ใช้งานนั้นมีสิทธิใช้งานจริง
- การควบคุมการเข้าถึง (Access control) ใช้จำกัดสิทธิผู้ใช้แต่ละคนว่าอนุญาตให้ใช้บริการนั้นๆ หรือไม่

6.12.1 เป้าหมายหลักของโมเดลรักษาความปลอดภัยใน COM+

มี 4 ประการ คือ

1. Activation Control
2. Access Control
3. Authentication Control
4. Identify Control

Activation Control ใช้ควบคุมว่าใครสามารถ launch คอมโพเน็นต์ได้ ส่วน *Access Control* เป็นการจำกัดการเข้าถึงออบเจ็กต์ของคอมโพเน็นต์ (คือ ในบางกรณีผู้ใช้อาจได้รับอนุญาตให้เข้าถึงฟังก์ชันบางส่วนของคอมโพเน็นต์เท่านั้น) ตัวอย่างเช่น ผู้ใช้ทั่วไปทำการเรียกใช้ผ่านทางเว็ปจะได้รับอนุญาตให้เรียกใช้แค่บางฟังก์ชั่นของคอมโพเน็นต์เท่านั้น แต่ถ้าเป็นผู้ดูแลระบบจะสามารถเรียกใช้ได้ทุกฟังก์ชั่นเป็นต้น

Authentication Control นั้นถูกใช้เพื่อให้แน่ใจได้ว่า ข้อมูลที่ส่งในระบบเน็ตเวิร์กนั้นจะสามารถเห็นได้โดยผู้ใช้ที่มีสิทธิ์เท่านั้น นอกจากนี้เรายังสามารถกำหนดระดับของ Authentication Control เพื่อเข้ารหัสข้อมูลก่อนที่จะส่งผ่านเครือข่าย

ส่วน *Identify Control* ใช้กำหนดหลักฐานด้านความปลอดภัยของคอมโพเน็นต์ที่ทำงาน ซึ่งอาจได้มาจากการปรับแต่งที่บัญชีผู้ใช้หรือจากไคลเอ็นต์แอพพลิเคชั่น

การกำหนดความปลอดภัยของ COM+ คอมโพเน็นต์ สามารถทำได้ 2 วิธีคือ

1. Declarative Security
2. Programmatic Security

Declarative Security เป็นการกำหนดความปลอดภัยใน COM+ catalog ภายนอกคอมโพเน็นต์ซึ่งกระทำโดยผู้ดูแลระบบ ส่วน Programmatic Security เป็นการเขียนโปรแกรมเพื่อควบคุมความปลอดภัยทำโดยผู้พัฒนาคอมโพเน็นต์

โมเดลความปลอดภัยทั้ง 4 แบบด้านบนสามารถใช้ Declarative Security กำหนดได้ทั้งหมดผ่านทาง COM+ catalog โดยใช้ Component Services administrative tools หรือ Distributed COM Configuration utility (dcomcnfg.exe) (ในปัจจุบันใช้ในการจัดการกับ unconfigured components) แต่ Programmatic Security ไม่สามารถใช้กับโมเดลแบบ Activation Security และ Identity Security ได้ เพราะการกำหนดค่าเหล่านี้ต้องถูกกำหนดก่อนที่คอมโพเน็นต์จะถูก launch

6.12.2 COM+ Security Packages

COM+ จำเป็นต้องมีโมเดลรักษาความปลอดภัยที่สามารถใช้ได้ทุกๆ แพล็ตฟอร์มที่มีการใช้ COM+ Services จึงไม่ควรใช้ระบบรักษาความปลอดภัยของวินโดวส์ ดังนั้น COM+ ได้กำหนดโมเดลรักษาความปลอดภัยระดับสูงขึ้นมาโดยใช้กลไกรักษาความปลอดภัยของพื้นฐานของวินโดวส์และ RPC โมเดลรักษาความปลอดภัยที่ยืดหยุ่นนี้ถูกสร้างบน Security Support Provider Interface (SSPI) ซึ่งเป็น API พื้นฐานที่ออกแบบมาเพื่อสนับสนุนความปลอดภัยในสภาวะแวดล้อมแบบกระจายการทำงาน

การออกแบบของ SSPI คล้ายกับสถาปัตยกรรมของ ODBC โดยการอิมพลีเม็นต์ SSPI API ใน DLL (เรียกว่า Security Service Provider (SSP)) ทำให้แอพพลิเคชั่นที่เขียนให้ทำงานกับ security provider หนึ่ง สามารถปรับเปลี่ยนไปใช้กับ security provider อื่นได้โดยการปรับแต่งเพียงเล็กน้อย ทุก security providersต้องอิมพลีเม็นต์ SSPI ซึ่งภายในอาจใช้กลไกรักษาความปลอดภัยใดก็ได้

วินโดวส์ 2000 สนับสนุนโมเดลรักษาความปลอดภัยทั้งแบบ NTLMSSP ซึ่งเป็นการกลไกการยืนยันสิทธิโดยไม่ต้องส่งรหัสผ่านเข้าสู่ระบบเน็ตเวิร์กเลย และเคอบิรอส (Kerberos)

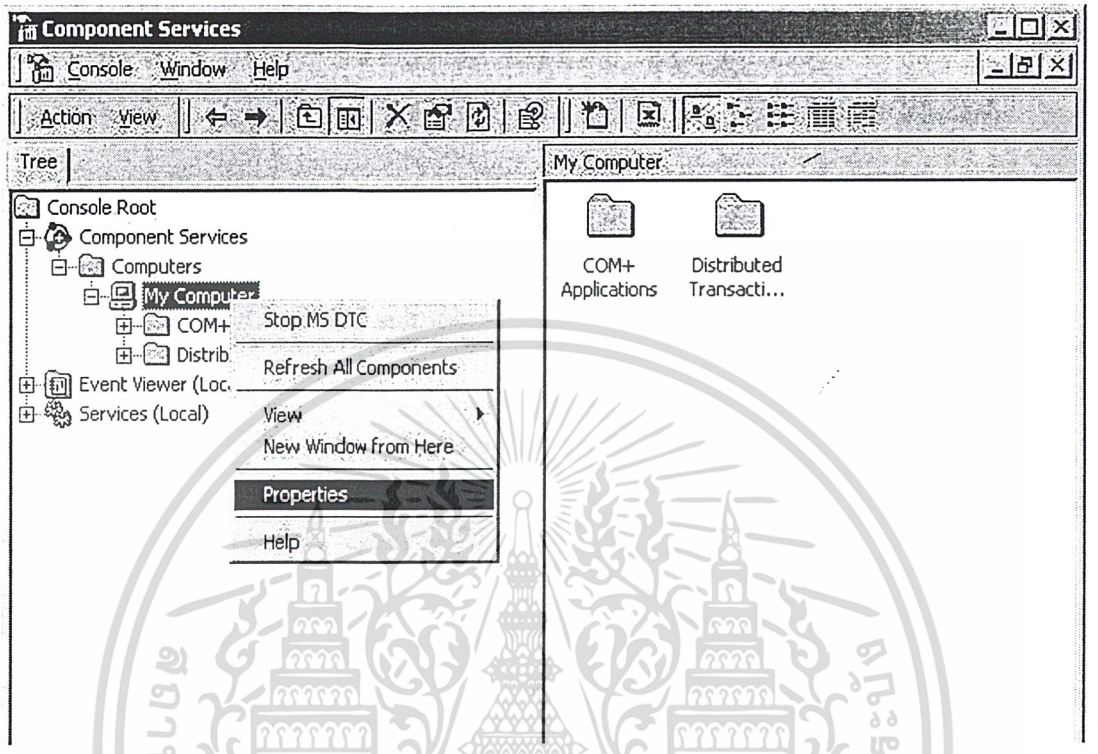
6.12.3 Declarative Security

COM+ catalog ใช้ในการปรับแต่งข้อมูลเกี่ยวกับโมเดลรักษาความปลอดภัยของ COM+ ซึ่งเรียกใช้ได้ผ่านทาง Component Services administration tool

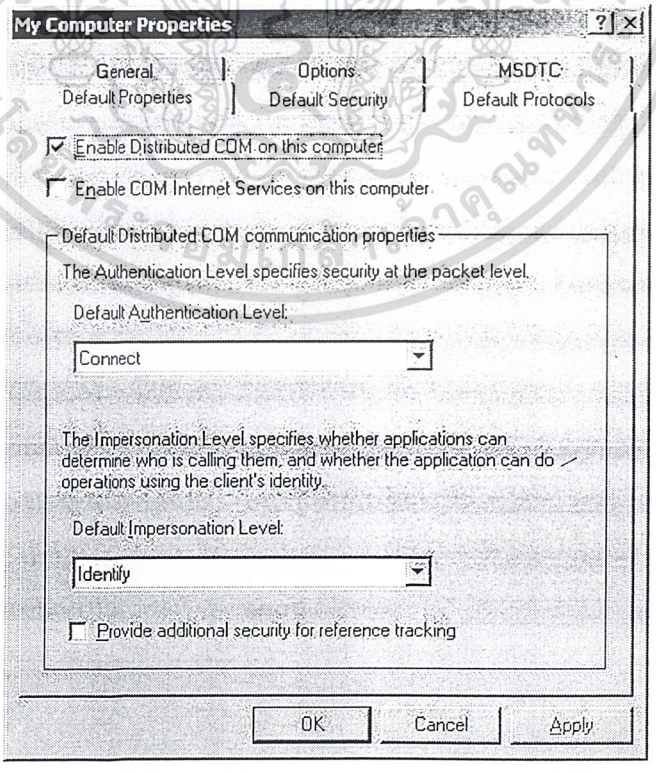
ข้อมูลของ Declarative security แบ่งออกเป็น default security และ component security โดย default security ใช้กำหนดค่าปกติในการ launch และการเข้าถึงทุกๆ คอมโพเน็นต์ที่กำลังใช้งานบนเครื่องนั้นๆ ส่วน component security ใช้กำหนดความปลอดภัยให้แก่คอมโพเน็นต์แต่ละตัว ซึ่งการกำหนดค่าที่นี้จะทับค่าที่ได้กำหนดไว้ใน default security (คือ override) และถ้ามีการกำหนดค่าที่ Programmatic security ค่าที่กำหนดไว้ที่ Programmatic security ก็จะทับทั้งค่าที่กำหนดไว้ใน default security และ component security

เปิด Component Services administration tool คลิ๊กขวาที่ My Computer จากนั้น เลือก Default Properties tab ดังรูปที่ 6-11 จะได้ผลดังรูปที่ 6-12 จากรูปตัวเลือก “Enable Distributed COM on this computer” หมายถึงว่าให้สามารถใช้ DCOM บนเครื่องนี้ได้ ซึ่งถ้าตัวเลือกนี้ไม่ถูกเลือกแล้ว จะไม่สามารถ

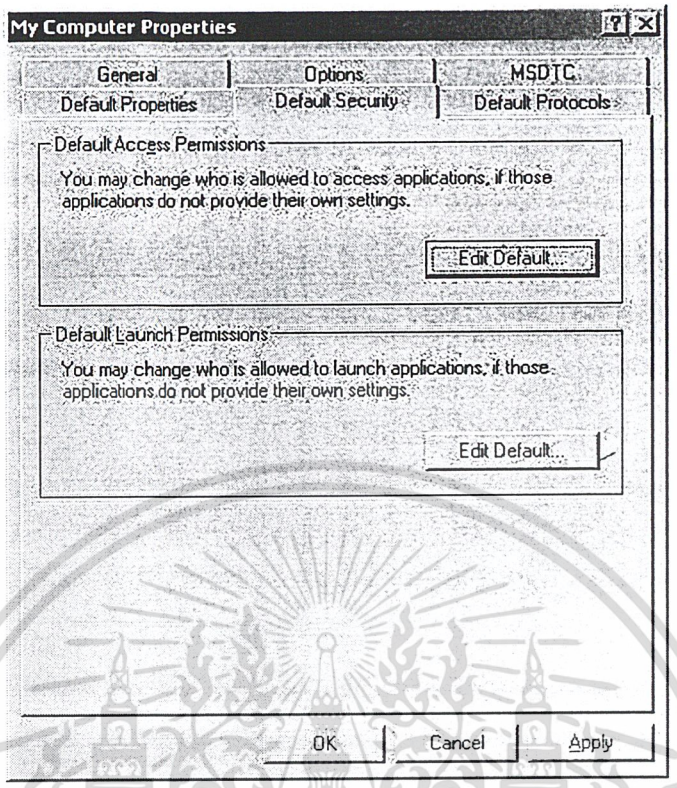
ทำการเรียกข้ามเครื่องได้เลย ส่วนตัวเลือก "Enable COM Internet Services on this computer" นั้น ใช้เลือกให้ COM+ Internet Service นั้นสามารถใช้งานได้หรือไม่ (ค่าปกติ คือ ไม่เลือก) เราสามารถกำหนดค่าระดับรักษาความปลอดภัยในระบบได้ในแท็ปต่างๆ เหล่านี้



รูปที่ 6-11 การเปิดเมนูเพื่อกำหนดค่ารักษาความปลอดภัย



รูปที่ 6-12 การกำหนดค่าปกติของการยืนยันสิทธิ



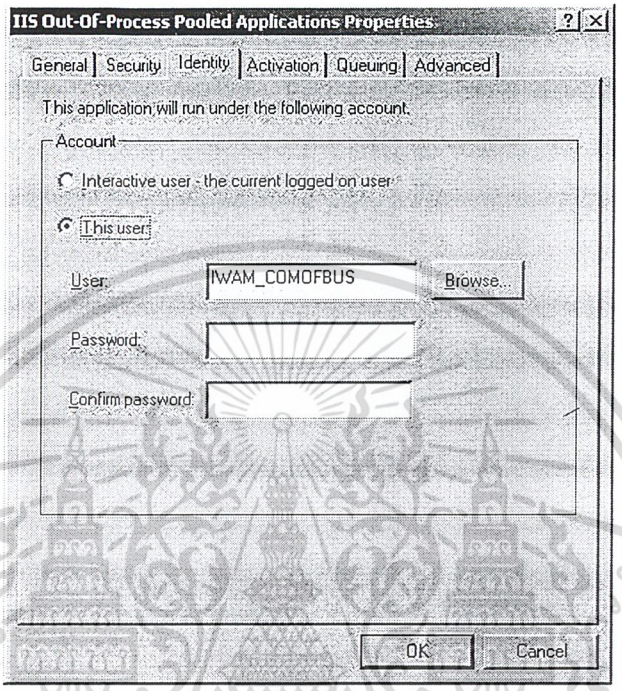
รูปที่ 6-13 การกำหนดค่าปกติของการเข้าถึง

ก่อนหน้าที่จะมี COM+ เราสามารถกำหนดความปลอดภัยแบบ Declarative Security โดยใช้แอพพลิเคชั่น dcomcnfg ปัญหาหลักของ dcomcnfg คือค่าความปลอดภัยจะครอบคลุมทั้งหมดของ COM เซิร์ฟเวอร์, COM คลาสทั้งหมดที่อยู่ใน COM เซิร์ฟเวอร์เดียวกันมีค่าความปลอดภัยเหมือนกันหมด ซึ่งไม่เหมาะสมหาก COM เซิร์ฟเวอร์ประกอบด้วยคลาสหรืออินเตอร์เฟสที่ต้องการค่าความปลอดภัยแตกต่างกัน ตัวอย่างเช่น เรามีคอมโพเน็นต์ที่ประกอบด้วย 2 อินเตอร์เพส อินเตอร์เฟสแรกประกอบด้วยเมธอดต่าง ๆ ที่จัดเตรียมไว้ให้ผู้ใช้ทั่วไปใช้งาน ส่วนอินเตอร์เฟสที่สองเป็นของผู้ดูแลระบบเพื่อใช้ในการจัดการควบคุมคอมโพเน็นต์โดยไม่อนุญาตให้ผู้ใช้ทั่วไปใช้อินเตอร์เฟสนี้ ในกรณีตัวอย่างที่กล่าวมา dcomcnfg ไม่สามารถทำได้ วิธีเดียวที่ทำได้คือใช้ API ของ COM Security อย่างไรก็ดีในการใช้ COM+ รันไทม์ ก็ยังสามารถใช้ประโยชน์จากวิธีนี้ได้เมื่อต้องการให้ Role ของผู้ใช้เป็นเงื่อนไขสำหรับการทำงานของเมธอดในคอมโพเน็นต์ เช่นให้บริการที่แตกต่างกันจากเมธอดเดียวกัน เมื่อผู้ใช้อยู่ใน Role ที่แตกต่างกันเป็นต้น

สำหรับ COM+ เราสามารถควบคุมการใช้เซิร์ฟเวอร์ได้ตั้งแต่ระดับคลาส อินเตอร์เฟส และเมธอดซึ่ง เรียกว่า Fine-Grained Security โดยที่ไม่ต้องเขียนโปรแกรมเพิ่มเติม การใช้ Fine-Grained Security เราต้องทำความเข้าใจเกี่ยวกับโรล (Role) ซึ่งเป็นสิ่งสำคัญในการกำหนดความปลอดภัยบน COM+

6.12.4 การปรับแต่งคอมโพเน็นต์แต่ละตัว

การปรับแต่งคอมโพเน็นต์แต่ละตัว ทำได้โดยคลิ๊กขวาที่คอมโพเน็นต์ที่ต้องการใน Component Services administration tool จะได้ดังรูปที่ 6-14



รูปที่ 6-14 แสดงการกำหนดค่า Identity Security

6.12.5 Role

COM+ ใช้โมเดลความปลอดภัยแบบใหม่โดยแสดงอยู่ในรูปแบบของโรล โรลคือกลุ่มของผู้ใช้ COM+ แอพพลิเคชั่นที่มีค่าความปลดภัยเหมือนกัน การกำหนดโรลมีลักษณะเหมือนชีวิตจริงที่มีการกำหนดบทบาทให้บุคคลว่ามีบทบาทอะไรในสังคม สามารถทำหรือไม่อนุญาตให้ทำอะไรได้บ้าง ตัวอย่าง เช่น COM+ แอพพลิเคชั่นที่ออกแบบสำหรับธนาคารแบ่งโรลออกเป็น Teller role, Manager role และ Loan Office role ผู้ใช้ที่อยู่ในโรลเดียวกันมีค่าความปลอดภัยเหมือนกัน แต่ละโรลอนุญาตและไม่อนุญาตให้ทำอะไรเหมือนกัน เมื่อกำหนดโรลให้กับ COM+ แอพพลิเคชั่น เราสามารถเพิ่มผู้ใช้หรือกลุ่มของผู้ใช้เข้าไปในโรลได้ 2 วิธี คือใช้คอมโพเน็นต์เซอร์วิสเอ็กซพลอเรอร์ (Declarative Security) หรือเขียนโปรแกรมโดยเรียกข้อมูลจากคอนเท็กซ์เพื่อดูว่าผู้ใช้มีโรลอะไร หากเป็นโรลที่ไม่อนุญาตก็สามารถหยุดการใช้งานหรือแจ้งความผิดพลาดที่เกิดขึ้น (Programmatic Security)

6.12.5.1 การสร้าง Role

เราสามารถเพิ่มโรลให้ COM+ Application โดยมีขั้นตอนดังนี้

1. คลิกขวาที่โฟลเดอร์ Roles ของแอพพลิเคชั่นที่ต้องการเพิ่มโรล
2. เลือก New > Role จะปรากฏไดอะลอก Role
3. ตั้งชื่อโรลที่ต้องการแล้วคลิก OK

เมื่อสร้างโรลแล้วต่อมาคือเพิ่มผู้ใช้เข้าไปในโรลและกำหนดส่วนต่าง ๆ ของแอพพลิเคชั่น (คอมโพเน็นต์, อินเตอร์เฟส, เมธอด) ว่าต้องการให้โรลใดเรียกใช้ได้

6.12.5.2 การเพิ่มผู้ใช้เข้าไปในโรล

มีขั้นตอนดังนี้

1. คลิกขวาที่โฟลเดอร์ Users ที่อยู่ในไดเรกทอรีของโรลที่ต้องการเพิ่มผู้ใช้
2. เลือก New > Users จะปรากฏไดอะลอกให้เลือกผู้ใช้ (ไดอะลอกที่ปรากฏขึ้นอยู่ว่ามีใช้ Active Directory หรือไม่)
3. เลือกผู้ใช้ที่ต้องการแล้วคลิก OK

6.12.6 การกำหนดความปลอดภัยให้แต่ละส่วนของแอพพลิเคชั่น

ก่อนอื่นเราต้องเลือกออบชั่น Access Control จึงจะใช้งานได้ โดยค่าดีฟอลต์แล้ว Access Control จะไม่ได้เลือกไว้ การเลือกออบชั่น Access Control มีขั้นตอนดังนี้

1. เปิดคอมโพเน็นต์เซอร์วิสเอ็กซพลอเรอร์
2. คลิกขวาที่แอพพลิเคชั่นที่ต้องการ เลือก Properties
3. คลิกที่แท็บ Security ของไดอะลอกคอมโพเน็นต์ที่ปรากฏขึ้นมา
4. เลือกเช็คบ็อกซ์ Enforce access checks for this application แล้วคลิก Apply หรือ OK

การเปลี่ยนแปลงค่าความปลอดภัยของ COM+ จะยังไม่เกิดผลทันทีจนกว่าเราจะรีสตาร์ทแอพพลิเคชันใหม่

การกำหนดความปลอดให้แต่ละส่วนของแอพพลิเคชั่นมีขั้นตอนดังนี้

1. คลิกขวาที่คอมโพเน็นต์ อินเตอร์เฟส หรือเมธอดที่ต้องการ เลือก Properties
2. คลิกที่แท็บ Security ของไดอะลอกที่ปรากฏขึ้นมา
3. เลือกโรลที่ต้องการที่อยู่ Role explicit set for selected item(s):
4. คลิก Appy หรือ OK

6.12.7 Programmatic Security

เราใช้ declarative security สำหรับกำหนดความปลอดภัยเป็นส่วนใหญ่ และใช้ programmatic security สำหรับงานประเภทพิเศษที่ต้องการควบคุมสูง

เนื่องจาก Declarative security ไม่สามารถปรับแต่ง role ภายในเมธอดเดียวได้ แต่เราสามารถทำได้ในส่วน programmatic นี้ โดยการใช้ context object ที่ COM+ มีให้ เพื่อเรียกเมธอด IObjectContext::IsSecurityEnabled และ IObjectContext::IsCallerInRole

บทที่ 7
XML

ในปี ค.ศ. 1991 Tim Berners-Lee ได้พัฒนา ภาษา HTML ขึ้นเพื่อใช้ในการสร้างเว็บเพ็จ หลังจากนั้น HTML ก็แพร่หลายและใช้กันอย่างกว้างขวาง จนทำให้มีเว็บเพ็จจำนวนมากมายมหาศาล

HTML เป็นมาตรฐานที่ได้รับการปรับปรุงและเสริมต่อมาเป็นลำดับ แต่ HTML ก็ยังเป็นมาตรฐานที่เน้นการนำข้อมูลข่าวสารมาแสดงผลบนจอภาพ ดังนั้นโครงสร้างหลักจึงได้แก่ หัวเรื่อง ชื่อเรื่อง ภาพ ชุดตัวพิมพ์ การแสดงผลด้วยรูปภาพ เสียง และมัลติมีเดีย เพียงเท่านี้ก็ทำให้นักพัฒนาเว็บสามารถออกแบบเว็บเพ็จที่ได้อย่างน่ามหัศจรรย์ และมีประสิทธิภาพยิ่ง

7.1 ข้อด้อยของ HTML

ด้วยข้อจำกัดและความต้องการของผู้ใช้ยังมีอีกมาก ทำให้ HTML มีข้อด้อยบางประการที่จะต้องปรับปรุง ดังต่อไปนี้

- แท็กของ HTML มีจำกัด และไม่อนุญาติให้เราสร้างแท็กขึ้นเองได้ ทำให้การทำงานมีขีดจำกัด แม้ว่าจะมี HTML เวอร์ชั่นใหม่ๆ ออกมา แต่ก็ยังไม่สามารถตอบสนองกับความต้องการของผู้ใช้ได้ทั้งหมด เพราะรูปแบบแท็กของ HTML มีลักษณะตายตัวว่าแต่ละแท็กเอาไว้ทำอะไร โดยที่แท็กต่างๆ จะถูกกำหนดโดยองค์กร W3C (World Wide Web Consortium) ซึ่งเป็นองค์กรที่มีหน้าที่วางมาตรฐานการทำเอกสารต่างๆบนเว็บ และนอกจากนี้ผู้ผลิต Web Browser ค่ายใหญ่ๆ เช่น Internet Explorer ของบริษัทไมโครซอฟต์ หรือ Netscape ของบริษัทเน็ตสเคปคอมมิวนิเคชั่นก็พยายามดึงดูดผู้ใช้บราวเซอร์ให้มาใช้ผลิตภัณฑ์ของตนมากขึ้น โดยการเพิ่มเติมแท็กจากแท็กมาตรฐานที่มีอยู่ ทำให้แต่ละบราวเซอร์แสดงผลลัพธ์จากไฟล์ HTML เดียวกันแตกต่างกัน
- เอกสารที่เขียนด้วย HTML มุ่งเน้นเฉพาะการแสดงผลเป็นหลัก ส่วนการสื่อความหมายของเอกสารนั้นนับว่าน้อยมาก ทำให้การค้นหาข้อมูลที่ต้องการจากเครือข่ายอินเตอร์เน็ตเป็นไปอย่างยากลำบาก คือ ได้ผลลัพธ์ที่ไม่เกี่ยวข้องออกมาเป็นจำนวนมาก
- HTML สามารถให้มุมมองเพียงด้านเดียวแก่ผู้เข้ามาเยี่ยมชม คือเราไม่สามารถทำให้ผู้ใช้หลายๆ คนมองเห็นเพจเดียวกันแตกต่างกันได้ ดังนั้นจึงมีต้องใช้ ASP และ DHTML (Dynamic HTML) เพื่อให้มองได้หลายมุม ซึ่งในงานหลายอย่างต้องการคุณสมบัติดังกล่าว โดยเฉพาะอย่างยิ่งการทำธุรกิจบนอินเตอร์เน็ตที่จำเป็นจะต้องมีการจัดแบ่งกลุ่มลูกค้า เช่น จัดกลุ่มลูกค้าให้แต่คนได้มองเฉพาะในสินค้าส่วนที่เขาสนใจ
- HTML ไม่สนับสนุนหลัก “การนำกลับมาใช้ใหม่ (reuse)” การแก้ไขเอกสารแต่ละครั้งค่อนข้างลำบาก จึงไม่สะดวกกับงานที่มีการเปลี่ยนแปลงข้อมูลบ่อย

- การเปลี่ยนกลับไปมาระหว่าง HTML กับงานอื่นทำได้ยาก เช่น เมื่อเราสร้าง HTML แล้ว จะนำกลับมาใช้งานใหม่ได้ยาก เช่น มีเอกสารเวิร์ดและแปลงเป็น HTML แต่หากจะแปลงกลับทำได้ยากที่จะทำให้ได้เหมือนเดิม

ในปี พ.ศ. 2539 Jon Bosak ได้เสนอแนวคิดต่อ W3C (World Wide Web Consortium) และนำเสนอภาษาใหม่ โดยนำแนวทางบางอย่างมาจาก SGML Standard Generalize Markup Language ซึ่งได้รับการพัฒนามานานแล้ว แต่มีความซับซ้อน ทำให้ใช้งานยาก ข้อเสนอใหม่นี้เรียกว่า XML (Extensible Markup Language) โดยเน้นให้มีส่วนช่วยแสดงส่วนที่เป็นเนื้อหาหรือสาระของข้อมูล

7.2 เป้าหมายหลักของ XML

เป้าหมายที่สำคัญของ XML เน้นที่จะนำไปใช้งานในอินเตอร์เน็ต จึงมีเป้าหมายหลักดังนี้

- เพื่อเป็นมาตรฐานในการทำเอกสารบนเครือข่ายอินเตอร์เน็ตที่ผู้ใช้ทุกคนต่างมีความเข้าใจและมีข้อตกลงที่ตรงกัน นั่นคือเอกสาร XML จะต้องเป็นรูปแบบที่สามารถใช้ได้ทั่วไปบนอินเตอร์เน็ต
- เพื่อรองรับการแลกเปลี่ยนข้อมูลระหว่างแอพพลิเคชั่นโดยไม่ต้องขึ้นกับแพลตฟอร์มหรือผู้ผลิตรายใดรายหนึ่ง
- มาตรฐานของ XML จะต้องเข้ากันได้กับมาตรฐานของ SGML ซึ่ง XML นั้นนับเป็นส่วนหนึ่งของ SGML ทำให้องค์กรกำลังใช้ SGML อยู่แล้วก็สามารถนำ XML มาใช้ร่วมด้วยกันได้
- เอกสาร XML จะต้องมีรูปแบบที่ผู้พัฒนาสามารถเขียนโปรแกรมเพื่อนำเอาเอกสารดังกล่าวไปประมวลผลได้โดยง่าย
- การออกแบบเอกสาร XML ควรทำได้ด้วยความรวดเร็ว
- เป็นการแยกส่วนของข้อมูลออกจากการแสดงผลอย่างชัดเจน คือ XML จะไม่บอกเราว่าการแสดงผลที่ได้จะเป็นเช่นไร แต่จะบอกว่าตัวเนื้อข้อมูลจริงๆ นั้นคืออะไร ทำให้การเปลี่ยนแปลงข้อมูลจะไม่ส่งผลกระทบใดๆ ต่อการแสดงผลเลย ในการกลับกันการแก้ไขส่วนการแสดงผลก็จะไม่มีผลกระทบกับเนื้อข้อมูลจริงๆ เช่นกัน นอกจากนี้การนำข้อมูลไปแสดงผลนั้น เราก็จะสามารถเลือกได้ว่าจะใช้อะไรเป็นตัวดึงข้อมูลมาแสดงผล เช่น ใช้ CSS, XSL, VBScript, JavaScript เป็นต้น
- เอกสารที่เขียนด้วย XML จะเป็นเอกสารที่สื่อความหมายได้ดีเพราะภาษา XML จะมีลักษณะของการให้ความหมายแก่ข้อมูลโดยผู้สร้างเอกสารจะเป็นผู้กำหนดนิยามและความหมายของแท็กต่างๆ เองตามข้อตกลงของ W3C ซึ่งจะทำให้การอ่านเอกสารเป็นไปได้ง่ายและเป็นมาตรฐาน เอกสารจะมีความชัดเจน ไม่กำกวม
- XML ออกแบบอย่างพิถีพิถันเน้นความจำเป็น กระทัดรัด เข้าใจง่าย และได้ประโยชน์กว้างขวาง
- XML สนับสนุนการประยุกต์เข้ากับงานต่าง ๆ และสนับสนุนโปรแกรมประยุกต์ต่าง ๆ

- XML เน้นเรื่องการประมวลผลเอกสาร จึงเหมาะกับงานทางด้านการวิเคราะห์เอกสาร การผลิตเอกสาร การแลกเปลี่ยน และการแสดงผล
- การเขียน XML ทำได้ตั้งแต่การใช้ Text editor ทั่ว ๆ ไป และไม่ต้องการเครื่องมือที่ซับซ้อน อย่างไรก็ดี ย่อมต้องมีผู้เขียน XML editor ให้ใช้งานได้ง่ายขึ้น
- XML เป็นมาตรฐานที่กำหนดแล้วใช้งานได้ทันที โดยที่ Browser และอุปกรณ์ต่าง ๆ พร้อมใช้งานร่วมกัน

7.3 การประยุกต์ใช้ XML

ในปัจจุบันมีการจัดการเอกสารจำนวนมากการบนเครือข่าย เช่น การสร้างห้องสมุดดิจิตอล การแลกเปลี่ยนข้อมูลข่าวสารระหว่างกัน การประยุกต์ใช้ XML จึงทำได้อย่างกว้างขวาง ตัวอย่างเช่น

- XML สามารถใช้ได้ในหลากหลายภาษา และผสมกันได้หลากหลายภาษา เนื่องจาก XML สนับสนุน UNICODE
- การพัฒนา XML Processor ทำให้สามารถดึงเอกสาร XML มาใช้งานได้ง่าย และใช้ร่วมกับโปรแกรมประยุกต์อื่นได้ง่าย เช่น โปรแกรม DB2, Oracle, SAP เป็นต้น
- XML ช่วยทำให้เกิดการรับส่งข้อมูลแบบ EDI โดยทำให้แนวทางการเชื่อมโยงและสร้างความเป็นเอกสารหรือมาตรฐานระหว่างองค์กร
- XML มีสภาพช่วยในการขนส่งข้อมูลไปยังปลายทางเพื่อให้แปลความหมายและใช้งานได้อย่างเต็มประสิทธิภาพ
- มีการสร้างการประยุกต์ และนำเสนอผลลัพธ์ไปใช้งานจาก XML ได้มาก
- การประยุกต์การดำเนินกิจกรรมบนเครือข่ายมีมาก เช่น e-Business, EDI, e-Commerce การจัดการ Supply chain, Demand chain management การดำเนินการแบบ intranet และ web base application

7.4 XML จะแทนที่ HTML หรือไม่

แม้ว่าแต่เดิม XML จะถูกสร้างขึ้นมาเพื่อแก้ไขข้อบกพร่องของภาษา HTML แต่ XML ก็ไม่ใช่สิ่งที่จะมาแทน HTML อย่างที่หลายๆ คนเข้าใจผิด เพราะ XML เป็นเพียงส่วนเนื้อหาที่แท้จริงของเอกสารเท่านั้น ส่วนในเรื่องการแสดงผลนั้น ลำพัง XML อย่างเดียวไม่สามารถบอกได้ว่าจะแสดงผลเป็นเช่นไร (ซึ่งก็คือ XML ไม่สามารถแสดงผลได้ด้วยตัวของมันเอง) ดังนั้น HTML ก็จะยังคงอยู่ต่อไปอย่างแน่นอน อย่างน้อยก็อีกระยะหนึ่ง ในขณะเดียวกัน HTML ก็จะมีพัฒนาการและการปรับเปลี่ยนโดยดึงเอาข้อดีของทั้ง XML และ HTML มารวมกันกลายเป็น XHTML ขึ้นมาใช้งานแทน

7.5 โครงสร้าง XML

XML ถูกวางให้มีบทบาทสำคัญในฐานะที่เป็นตัวกลางของการสื่อสารระหว่างโปรแกรม เพราะเราสามารถสื่อสารระหว่างระบบงานที่แตกต่างกันโดยใช้ความสามารถของ XML SOAP และ HTTP ซึ่ง

จะไม่มีปัญหากับ firewall เพราะ HTTP ใช้ port 80 ในการติดต่อสื่อสาร นี่คือตัวอย่างของไฟล์ XML

Prolog

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="show_book.xsl"?>
<!DOCTYPE catalog SYSTEM "catalog.dtd">
<!--catalog last updated 2000-11-01-->
```

Document Elements

```
<catalog xmlns="http://www.example.com/catalog/">
      <book id="bk101">
            <author>&#71;Abercrombie,Kim</author>
            <title>XML Developer's &#x47;uide</title>
            <genre>Computer</genre>
            <price>44.95</price>
            <publish_date>2000-10-01</publish_date>
            <description><![CDATA[An in-depth look at
creating applications with XML, using <, >, ]]> and
&amp;amp;.</description>
      </book>
      <book id="bk109">
            <author>Kress, Peter</title>
            <title>Paradox Lost</title>
            <genre>Science Fiction</genre>
            <price>6.95</price>
            <publish_date>2000-11-02</publish_date>
            <description>After an inadvertant trip through
a Heisenberg Uncertainty Device, James Salway discovers the
problems of being quantum.</description>
      </book>
</catalog>
```

รูปที่ 7-1 ตัวอย่างไฟล์ XML

จากภาพข้างบนจะเห็นว่า XML จะประกอบด้วย 2 ส่วนคือ

1. ส่วนที่เป็น **Prolog** โดยบรรทัดแรกจะเหมือนการประกาศทั่วไปว่า XML ตัวนี้เป็น version 1 ซึ่งเป็น version ปัจจุบัน และถ้าเราต้องการให้สามารถแสดงภาษาไทยได้ ก็จะต้องเปลี่ยน encoding เป็น **encoding="windows-874"** ถัดบรรทัดลงมาจะเป็นการ link file XSL เข้ามาในไฟล์ XML เพื่อจัดรูปแบบการแสดงผล (เหมือนกับที่เรา link ไฟล์ CSS เข้ามาช่วยจัดรูป HTML ของเรา) บรรทัดต่อมา เป็นการกำหนดรูปแบบของไฟล์ XML ว่าถูกต้องตามที่กำหนดใน DTD หรือไม่ ที่ต้องมีการทำเช่นนี้ ก็เพราะว่าเราสามารถเพิ่มเติมส่วนต่างๆ เข้าใน XML ของเราได้ตามชอบใจ ซึ่งถ้าเราใช้ของเราคนเดียวก็จะไม่มีปัญหาอะไร แต่ถ้าเราต้องมีการแลกเปลี่ยนข้อมูลกัน ซึ่งก็จะใช้รูปแบบเดียวกัน โดย DTD จะเป็นตัวกำหนดรูปแบบความถูกต้องของไฟล์ XML (Well Formed)

2. ส่วนที่เป็น **Document Elements** คือส่วนที่เป็นข้อมูล โดยจะแบ่งข้อมูลเป็นส่วนๆ เป็น element โดย element บนสุดจะเรียก Root tag หรือโหนดแม่ แล้วจึงค่อยแตกเป็นโหนดลูก จากตัวอย่างข้างบนนี้ เราจะมี catalog เป็น Root node และมีโหนดลูกเป็น book แต่ละเล่ม และในโหนดลูกก็สามารถ

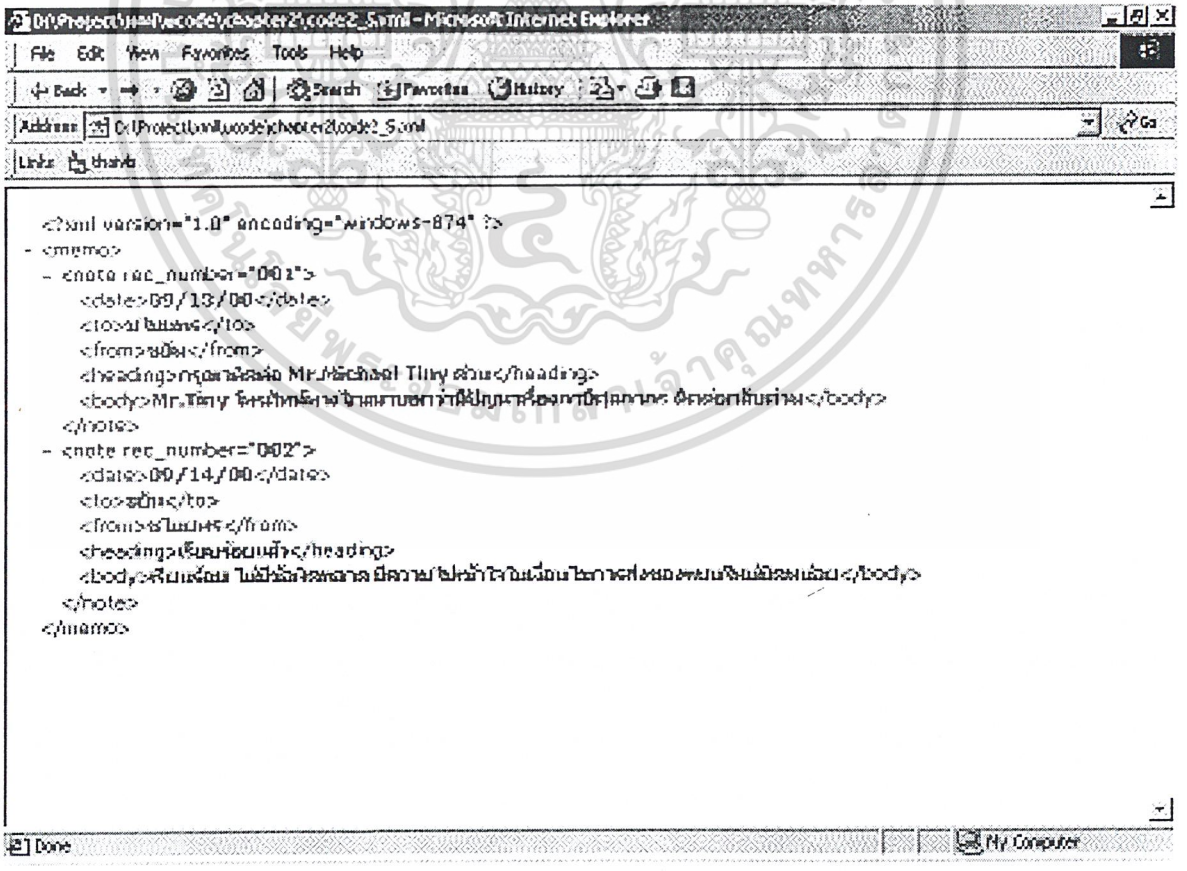
จะมี element ย่อย ๆ ได้อีกตามความต้องการ และในแต่ละ element ก็อาจจะมี attribute ได้เหมือนกันโดยจะอยู่ในรูป " " จากตัวอย่างข้างบน เราใช้ attribute id เป็นตัวแยก book แต่ละเล่ม

7.6 ข้อควรระวังในการเขียน XML

ในการสร้างเอกสาร XML มีข้อควรระวัง ดังต่อไปนี้

- เมื่อเปิดแท็กแล้วจะต้องปิดแท็กด้วยเสมอ เช่น ย่อหน้าที่ 1
- ในแท็กของ XML นั้น อักษรตัวใหญ่และเล็กนั้นมีความแตกต่างกัน (Case Sensitive) เช่น <Ticket>จะไม่เหมือนกับ <ticket>
- การจัดการเรื่องตำแหน่งแท็กของ XML ต้องเป็นไปตามลำดับ คือจำเป็นต้องเปิดและปิดเป็นวงๆ ไป เช่น ข้อความนี้จะเป็นตัวหนาและเอียง
- จำเป็นต้องมี Root Tag โดยมีแท็กลูกอื่นๆ อยู่ในวงของแท็กแม่นี้ และถ้าแท็กลูกมีแท็กหลานแท็กหลานก็จะต้องอยู่เป็นวงๆ ภายในกรอบของแท็กลูก จะไม่มีการฉีกวงได้เหมือนใน HTML
- attribute ของ XML ต้องอยู่ ภายในกรอบของเครื่องหมายฟันหนู (") เสมอ

เมื่อเราพิมพ์ Browse ดูไฟล์นี้ในโปรแกรม Internet Explorer จะได้ผลลัพธ์ดังรูป



รูปที่ 7-2 ผลลัพธ์จาก Browse ดูในโปรแกรม Internet Explorer

7.7 วิเคราะห์โครงสร้างของเอกสารแบบ DOM และ SAX

ในการพัฒนาแอพพลิเคชั่นโดยมีการนำ XML ไปใช้งานนั้น สิ่งหนึ่งที่ผู้พัฒนาจำเป็นต้องทำคือ การพาร์ส (parse) เอกสาร XML เพราะเนื่องมาจากว่าโปรแกรมจะมองเห็นเอกสารเป็นเพียงอักขระธรรมดาๆ เท่านั้น เราจึงต้องมีการเขียนโปรแกรมเพื่อให้แอพพลิเคชั่นสามารถเข้าใจได้ว่าเอกสาร XML ของเรามีการจัดโครงสร้างเป็นอย่างไร โดยสิ่งที่เราใช้ในการพาร์สเอกสารเราจะเรียกว่าพาร์สเซอร์ (Parser)

จากตัวอย่างที่ผ่านมาแล้ว เราสามารถใช้โปรแกรม Internet Explorer มาใช้กับเอกสาร XML ของเราได้ เนื่องจากโปรแกรมนี้ได้มีการฝังโปรแกรมพาร์เซอร์เอาไว้ในตัวอยู่แล้ว เรียกว่า Microsoft XML Parser เราจึงไม่ต้องทำการกำหนดการอ้างอิงพาร์สเซอร์เหมือนเวลาที่เราเขียนโปรแกรมทั่วๆ ไป

ในปัจจุบันมีมาตรฐานหรือวิธีการหลักๆ อยู่ 2 อย่างในการพาร์สเอกสาร XML ได้แก่ DOM (Document Object Model) และ SAX(Simple API for XML) ซึ่งทั้ง 2 วิธีนี้ต่างก็มีจุดเด่นและจุดด้อยอยู่ในตัวเอง การใช้งานจึงขึ้นกับความต้องการในการใช้งานของแอพพลิเคชั่นที่ใช้เอกสาร XML นั้นว่ามีความเหมาะสมกับวิธีการใดมากที่สุด

7.7.1 การพาร์สเอกสาร XML โดยใช้ DOM (Document Object Model)

XML DOM เป็น API (Application Programming Interface) สำหรับเอกสาร XML ซึ่งจะกำหนดว่าเอกสาร XML จะมีการเข้าถึง การโยกย้ายและการจัดการต่างๆ ได้อย่างไรบ้าง ด้วยวิธีการแบบ DOM ผู้พัฒนาจะสามารถเข้าไปในโครงสร้างระดับต่างๆของเอกสาร XML ได้ รวมทั้งยังสามารถเพิ่ม ลบ และแก้ไขอีลิเมนต์ต่างๆ ของเอกสารได้อีกด้วย

DOM เป็นมาตรฐานของ W3C โดยมีเป้าหมายคือ ต้องสามารถสร้างอินเตอร์เฟซที่เป็นมารตรฐานและจะต้องใช้ได้กับแอพพลิเคชั่นได้หลากหลาย

การทำงานของ DOM จะเริ่มต้นเมื่อโปรแกรมหรือแอพพลิเคชั่นเรียกพาร์สเซอร์เพื่อทำการโหลดเอกสาร XML เข้าสู่หน่วยความจำของคอมพิวเตอร์ เมื่อเอกสารถูกโหลดเสร็จเรียบร้อยแล้ว DOM ก็จะทำให้โปรแกรมสามารถดึงข้อมูลมาใช้งานหรือดำเนินต่างๆ กับข้อมูลได้

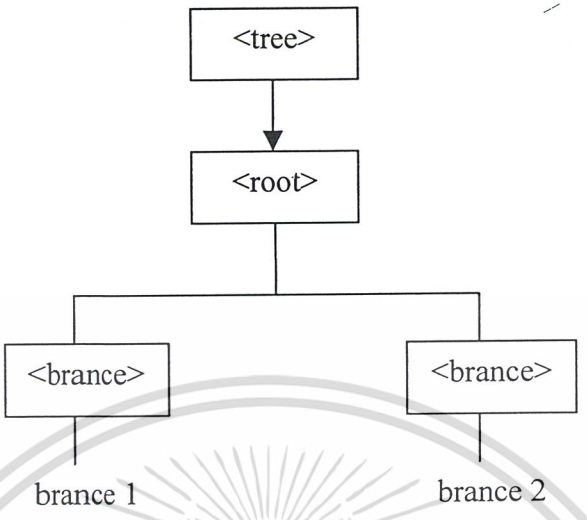
DOM จะมีมุมมองไปยังเอกสาร XML เป็นโครงสร้างแบบต้นไม้ (Tree) โดยระดับบนสุดของทรีเราจะเรียกว่า documentElement ซึ่งอีลิเมนต์ดังกล่าวจะแตกสาขาออกไปเป็นอีลิเมนต์ย่อยๆ ตั้งแต่ 1 อีลิเมนต์ขึ้นไป ซึ่งเราจะเรียกว่า โหนดลูก (childNodes)

จากเอกสาร XML ที่มีรายละเอียดดังด้านล่างนี้

```
<tree>
        <root>
                <brance> brance 1 </brance>
                <brance> brance 2 </brance>
        </root>
</tree>
```

รูปที่ 7-3 ตัวอย่างไฟล์ XML

DOM จะมองเอกสารข้างต้นเป็นทรี ดังนี้คือ



รูปที่ 7-4 DOM Tree

จุดเด่นของ DOM

- โปรแกรมจะพาร์สเอกสารเพียงครั้งเดียว เมื่อ DOM ได้สร้างทรีขึ้นมาแล้ว ทรีนั้นก็จะอยู่ในหน่วยความจำตลอดจนกว่าเราจะสั่งยกเลิก และในขณะที่ทรีอยู่ในหน่วยความจำ เราก็สามารถเข้าถึงข้อมูลที่อยู่ภายในทรีได้อย่างรวดเร็ว
- ข้อมูลที่สร้างเป็นทรีจะทำให้ข้อมูลเกิดความสัมพันธ์ ทำให้การเรียกใช้ทำได้ง่าย

จุดด้อยของ DOM

- ในกรณีที่เอกสารมีขนาดใหญ่และซับซ้อน การสร้างทรีจะทำได้ช้า และกินหน่วยความจำมาก
- ผู้พัฒนาโปรแกรมจำเป็นจะต้องเข้าใจถึงโครงสร้างโดยรวมของเอกสารเป็นอย่างดี จึงจะสามารถเขียนโปรแกรมโดยใช้ DOM ได้ กล่าวคือในส่วนของการเขียนโค้ดด้วยวิธีการของ DOM นั้น จะค่อนข้างยาวและยุ่งยากอยู่พอสมควร

7.7.2 การพาร์สเอกสาร XML โดยใช้ SAX (Simple API for XML)

SAX เป็น API ที่มีลักษณะการทำงานก็ต่อเมื่อมีเหตุการณ์ (เช่นการร้องขอข้อมูล) ใดๆ เกิดขึ้น หรือเรียกโมเดลการทำงานแบบนี้ว่า event-based model กล่าวคือ SAX จะไม่มีการสร้างภาพรวมของเอกสารไว้ก่อนเลยว่า เอกสารนั้นมีโครงสร้างเป็นอย่างไร เมื่อแอพพลิเคชั่นมีการร้องขอมาจึงจะรายงานหรือทำงานตามเหตุการณ์ของการทำพาร์สซิ่ง (parsing) ไปยังแอพพลิเคชั่น

หลักการทำงานของ SAX จะมองเพียงจุดเริ่มต้นและจุดสิ้นสุดของเอกสารและของอีลีเมนต์เท่านั้น และจะสนใจเฉพาะสิ่งที่ต้องการเท่านั้น ทำให้ SAX เข้าถึงข้อมูลได้ง่ายกว่า DOM รวมถึงการเขียนโปรแกรมก็ทำได้ง่ายกว่า เพราะมองข้อมูลในลักษณะเส้นตรงและสามารถพาร์สเอกสารที่มีขนาดใหญ่

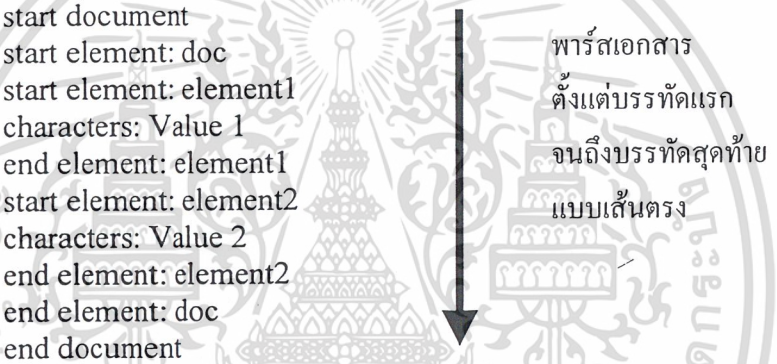
มากกว่าจำนวนหน่วยความจำที่เรียกได้ ดังนั้นโปรแกรมระบบฐานข้อมูลใหญ่ๆ หลายบริษัทจึงนิยมใช้วิธีนี้

จากเอกสาร XML ที่มีรายละเอียดดังด้านล่างนี้

```
<tree>
        <root>
                <brance> brance 1 </brance>
                <brance> brance 2 </brance>
        </root>
</tree>
```

รูปที่ 7-5 ตัวอย่างไฟล์ XML

SAX จะมองเอกสารข้างต้นไปทีละบรรทัด ในลักษณะเหตุการณ์เชิงเส้น ดังนี้



รูปที่ 7-6 SAX Event

ตัวอย่างการพาร์สแบบ SAX ข้างต้นเป็นการพาร์สทั้งเอกสาร ในทางปฏิบัติแล้วเรามักไม่ได้มีการพาร์สเอกสารทั้งหมด เมื่อเราได้ข้อมูลที่เราต้องการแล้วราก็จะหยุด วิธีการพาร์สแบบ SAX จึงทำงานได้อย่างรวดเร็วและเขียนโปรแกรมได้ง่ายมาก

จุดเด่นของ SAX

- เข้าถึงข้อมูลได้ง่ายและรวดเร็ว
- เขียนโปรแกรมได้ง่าย เนื่องจากการเข้าถึงข้อมูลเป็นลักษณะเส้นตรง
- ใช้หน่วยความจำน้อย สามารถเข้าถึงข้อมูลหรือเอกสารที่มีขนาดใหญ่กว่าหน่วยความจำที่มีของเครื่องได้

จุดด้อยของ SAX

- ไม่สามารถบอกความสัมพันธ์ของข้อมูลได้
- เป็นวิธีการที่ไม่ค่อยมีประสิทธิภาพหากเป็นการเข้าถึงข้อมูลเดิมบ่อยๆ

บทที่ 8
SOAP

8.1 แนะนำ SOAP

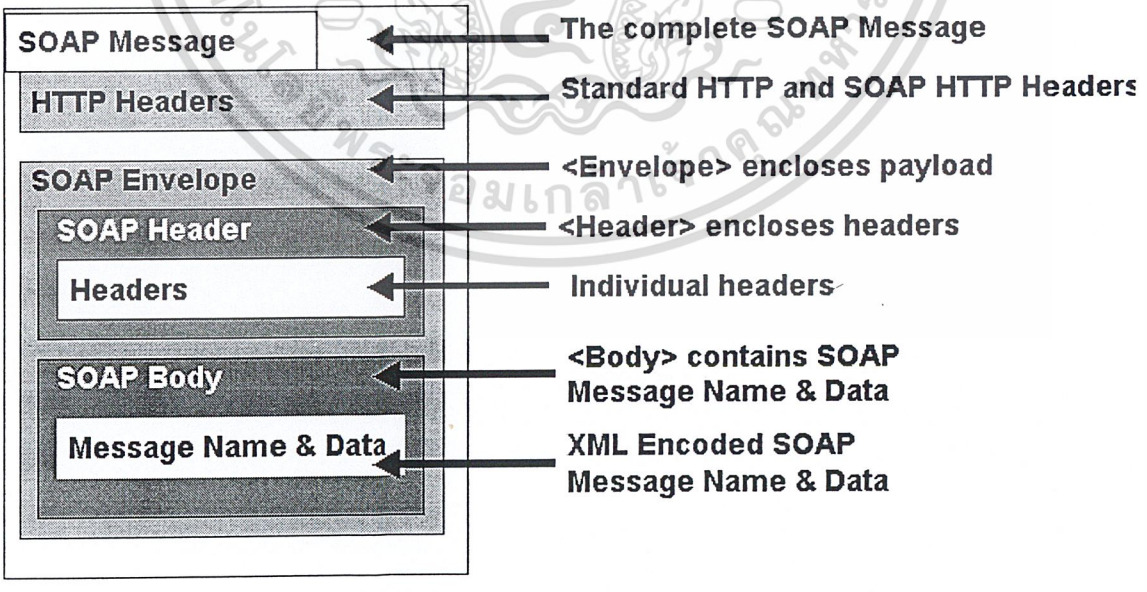
SOAP (Simple Object Access Protocol) เป็นโพรโตคอลที่ใช้ XML เป็นพื้นฐาน เพื่อให้ซอฟต์แวร์คอมโพเน็นต์ และแอพพลิเคชันสามารถติดต่อกันผ่าน HTTP ซึ่งเป็นมาตรฐานอินเตอร์เน็ตโพรโตคอลได้

เนื่องจากการสื่อสารระหว่างแอพพลิเคชัน เป็นสิ่งจำเป็นต่อการพัฒนาแอพพลิเคชันบนอินเตอร์เน็ตเป็นอย่างมาก แต่แอพพลิเคชันแบบกระจายกันทำงาน (distributed application) ในปัจจุบันใช้ Remote Procedure Call (RPC) สื่อสารกันระหว่างออบเจ็กต์ เช่น DCOM และ CORBA ซึ่งไม่ได้ใช้พอร์ตเดียวกับ HTTP ที่เป็นพอร์ตมาตรฐานสำหรับให้บริการเว็บ ดังนั้น RPC จึงนำมาปรับใช้กับอินเตอร์เน็ตได้ยาก และมีปัญหาทางด้านความปลอดภัย ไฟร์วอลล์และพร็อกซีเซิร์ฟเวอร์จะไม่ยอมให้ส่งข้อมูลชนิดนี้ได้ตามปกติ วิธีที่ดีกว่าคือใช้ HTTP เพราะเป็นที่ยอมรับโดยอินเตอร์เน็ตบราวเซอร์ และเซิร์ฟเวอร์ทุกชนิด ซึ่ง SOAP ถูกสร้างมาเพื่อใช้ในกรณีนี้

ข้อดีของ SOAP ก็คือ SOAP ไม่ขึ้นกับคอมโพเน็นต์เทคโนโลยี และภาษาการเขียนโปรแกรมใดๆ สามารถเขียนได้ง่ายและขยายเพิ่มเติมได้

8.2 ส่วนประกอบของ SOAP

SOAP เมสเซจใช้ไวยากรณ์ของ XML ในการสร้าง ประกอบด้วย 3 อีลีเม็นต์มาตรฐาน คือ SOAP Envelope, SOAP Header และ SOAP Body



รูปที่ 8-1 โครงสร้างของ SOAP

SOAP เมสเซจ จะต้องเป็นไปตามกฎนี้

- **Envelope** เป็นอีลีเม็นต์ที่อยู่บนสุด ต้องมีอีลีเม็นต์นี้เสมอ
- **Header** อาจจะมีหรือไม่มีก็ได้ แต่ถ้ามีต้องเป็น child element แรกของ Envelope
- **Body** ต้องเป็น child element แรกของ envelope หรือ header

ตัวอย่างของ SOAP เมสเซจ ตัวอย่างนี้คือ GetLastTradePrice SOAP Request ที่ส่งไปยัง StockQuote service ประกอบด้วยสตริงพารามิเตอร์ symbol และคืนค่า float กลับมากับ SOAP response SOAP envelope element อยู่ที่จุดบนสุดของเอกสาร XML XML namespace ใช้เพื่อป้องกันการสับสนของ SOAP identifier

SOAP request จะเป็นดังนี้

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP response จะเป็นดังนี้

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
```

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

นอกจากนั้นใน SOAP เมสเซจ จะมีการใช้ XML namespace ทุกๆ อีลีเม็นต์ในเอกสารจะขึ้นต้นด้วย namespace namespace จะถูกกำหนดโดยใช้ xmlns attribute ดังนี้

```
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
```

แต่หากไม่ต้องการให้ขึ้นต้นด้วย namespace ก็สามารถทำได้ ดังนี้

```
<Envelope
        xmlns="http://schemas.xmlsoap.org/soap/envelope/">
```

8.2.1 Envelope

envelope element เป็นอีลีเม็นต์บนสุดของเอกสาร XML ที่ใช้แสดงเมสเซจ อาจประกอบด้วยแอตทริบิวต์ คือ envelope namespace และ encodingStyle

1. **Envelope Namespace** – SOAP เมสเซจจะแสดงเวอร์ชันโดยใช้ namespace ของ envelope element namespace จะถูกนำไปใช้ขึ้นต้น envelope, header และ body element
2. **EncodingStyle Attribute** – ใช้แสดงวิธี serialization ของ SOAP เมสเซจ แอตทริบิวต์นี้สามารถมีได้ในทุกอีลีเม็นต์และจะมีผลกับ content ของอีลีเม็นต์นั้นและ child element ทั้งหมดของมันที่ไม่ได้ประกาศแอตทริบิวต์

8.2.2 Body

เป็นส่วนของข้อมูลที่ใช้ในการแลกเปลี่ยน โดยทั่ว ๆ ไปข้อมูลจะเป็นการ marshall RPC call และ error report child element ของ body element จะถูกเรียกว่า Body entry

body entry จะต้องเป็นไปตามกฎนี้

- body entry จะต้องแสดงด้วยชื่อเต็มที่ประกอบด้วย namespace URI และ ชื่อของมัน (local name)
- SOAP encodingStyle attribute อาจจะมีได้ เพื่อแสดงถึงวิธี encode ของ body entry นั้น

8.2.3 Header

เป็นส่วนเพิ่มเติมของเมสเซจสามารถประกอบด้วยอินฟอร์เมชันที่ระบุไปยังแอพพลิเคชัน ส่วนเพิ่มเติมนี้อาจนำไปใช้อิมพลีเม็นต์เป็น authentication, transaction management เป็นต้น ทุก ๆ child element ของ header element จะถูกเรียกว่า Header entry

header entry จะต้องเป็นไปตามกฎดังนี้

- header entry จะต้องแสดงด้วยชื่อเต็มที่ประกอบด้วย namespace URI และ ชื่อของมัน (local name)
- อาจมี SOAP encodingStyle attribute ที่ใช้สำหรับ header entry
- SOAP mustUnderStand attribute และ SOAP actor attribute จะมีหรือไม่ก็ได้ เพื่อใช้บอกว่าจะจัดการกับ entry นั้นอย่างไรและโดยใคร

```
<soap:Header>
  <m:local xmlns:m="http://www.w3schools.com/local/"
           soap:actor="http://www.w3schools.com/appml" />
    <m:language>en</m:language>
    <m:currency>USD</m:currency>
  </m:local>
</soap:Header>
```

1) **SOAP Actor Attribute** – SOAP เมสเซจสามารถถูกส่งไปยังปลายทางผ่านตัวกลางของ SOAP (SOAP intermediary) ตัวกลางของ SOAP คือแอพพลิเคชันที่มีความสามารถทั้งรับและส่งต่อ SOAP เมสเซจ ในการส่งต่อตัวกลางจะต้องไม่ส่งต่อ header element ไปให้กับแอพพลิเคชัน โดยตัวกลางนี้จะถูกกำหนดเป็น URI ถ้าไม่มี attribute นี้หมายถึงผู้รับ SOAP เมสเซจเป็นปลายทางสุดท้าย
2) **SOAP MustUnderStand Attribute** – ใช้แสดงว่า header entry นี้จำเป็นหรือเป็นเพียงออปชันสำหรับผู้รับที่จะนำไปประมวลผล ค่าของมันคือ “1” หรือ “0” ถ้าไม่มีการระบุจะมีค่าเท่ากับ “0”

โดย "0" หมายถึงเป็นออปชัน และ "1" หมายถึงจำเป็น โดยจะต้องประมวลผลให้ถูกต้องตาม semantic หรือต้อง fail เช่น เมสเซจเป็นส่วนหนึ่งของ transaction ถ้าปลายทางไม่รองรับ transaction จะต้องไม่ประมวลผลและส่งผลผิดพลาดกลับไปในกรณีที่ mustUnderStand attribute เป็น "1" แต่ถ้าเป็น "0" จะยังคงประมวลผลต่อไป

8.3 SOAP Fault Element

ข้อความแสดงความผิดพลาดจากแอพพลิเคชันของ SOAP จะเก็บอยู่ใน fault element ซึ่งถ้ามีจะต้องปรากฏใน body element เพียงครั้งเดียวใน SOAP เมสเซจ ตัวอย่างอาจเป็นดังนี้

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:MustUnderstand</faultcode>
      <faultstring>Mandatory Header error.</faultstring>
      <faultactor>http://www.wrox.com/heroes/endpoint.asp</faultactor>
      <detail>
        <w:source xmlns:w="http://www.wrox.com/">
          <module>endpoint.asp</module>
          <line>203</line>
        </w:source>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

SOAP fault element มี element ย่อย ๆ ดังตารางนี้

Sub Element	Description
<faultcode>	โค้ดที่ระบุถึงการ error
<faultstring>	ข้อความการ error
<faultactor>	ใครเป็นสาเหตุของการ error
<detail>	ระบุอินฟอร์เมชันของการ error

ตารางที่ 8-1 SOAP Fault Element

ค่าของ faultcode สามารถมีค่าได้ดังตารางนี้

Error	Description
VersionMismatch	namespace ภายใน SOAP Envelope element ไม่ถูกต้อง
MustUnderstand	child element ของ Header element กับ mustUnderstand attribute ที่มีค่า "1" ผู้รับไม่รองรับ
Client	เมสเซจมีรูปแบบไม่ถูกต้อง หรือมีอินฟอร์เมชันที่ไม่ถูกต้อง
Server	เกิดปัญหากับเซิร์ฟเวอร์ ไม่สามารถประมวลผลได้

ตารางที่ 8-2 ค่า faultcode

8.4 SOAP Encoding

SOAP encoding มีวิธีการ map จาก type ของโปรแกรมมิ่งไปเป็น XML 2 วิธี คือจากภายนอกโดยใช้ WSDL (Web Services Description Language) ที่บอกถึง type ของข้อมูลที่รับหรือส่ง หรือใช้ xsi:type attribute ในกรณีที่ภาษาที่ใช้ไม่รองรับ WSDL โดยทั้ง 2 วิธีจะใช้ XML schema ในการระบุ type ตัวอย่างของการใช้ xsi จะเป็นดังนี้

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <soap:Body>
    <m:MixedMessage xmlns:m="http://www.wrox.com/mix/">
```

```
      <param1 xsi:type="xsd:string">OU812</param1>
      <param2 xsi:type="xsd:integer">2001</param2>
      <param3 xsi:type="xsd:double">3.14159</param3>
    </m:MixedMessage>
  </soap:Body>
</soap:Envelope>
```

SOAP รองรับ type ของข้อมูลได้ทั้ง simple type และ compound type เช่น struct และ array

8.4.1 ตัวอย่าง Struct

struct book ที่เขียนด้วยภาษา c++ จะเป็นดังนี้

```
struct Book
{
    string author;
    string name;
    int page;
};
```

Soap เมสเซจที่ไม่ใช้ WSDL แต่ใช้ xsi จะเป็นดังนี้

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:searchBook xmlns:ns1="http://soapinterop.org/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<bookResult xmlns:ns2="http://soapinterop.org/xsd" xsi:type="ns2:Book">
  <author xsi:type="xsd:string">Ed Roman</author>
  <name xsi:type="xsd:string">Mastering EJB</intro>
  <page xsi:type="xsd:int">500</page>
```

```
</bookResult>
</ns1:searchBook>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

แต่ถ้าใช้ WSDL ในส่วนของการประกาศ schema จะต้องประกอบด้วย schema ดังนี้

```
<types>
  <schema targetNamespace='http://soapinterop.org/xsd'
    xmlns='http://www.w3.org/2001/XMLSchema'
    xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
    xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
    elementFormDefault='qualified'>
      <complexType name="Book">
        <sequence>
          <element name='author' type='string'/>
          <element name='name' type='string'/>
          <element name='page' type='int'/>
        </sequence>
      </complexType>
  </schema>
</types>
```

8.4.2 ตัวอย่าง Array

ตัวอย่างนี้เป็น array ของ struct book ในตัวอย่างก่อน Soap เมสเซจที่ไม่ใช้ WSDL แต่ใช้ xsi จะเป็นดังนี้

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
```

```
<ns1:BookArrayResult xmlns:ns1="http://soapinterop.org/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <BookArray xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
    xsi:type="ns2:Array" xmlns:ns3="http://soapinterop.org/xsd"
    ns2:arrayType="ns3:Book[3]">
    <item xsi:type="ns3:Book">
      <author xsi:type="xsd:string">Ed Roman</author>
      <name xsi:type="xsd:string">Mastering EJB</name>
      <page xsi:type="xsd:int">500</page>
    </item>
    <item xsi:type="ns3:Book">
      <author xsi:type="xsd:string">Roger Session</author>
      <name xsi:type="xsd:string">The Battle of the  middleware</name>
      <page xsi:type="xsd:int">350</page>
    </item>
    <item xsi:type="ns3:Book">
      <author xsi:type="xsd:string">Chris Dix</author>
      <name xsi:type="xsd:string">Professional XML Web Service</name>
      <page xsi:type="xsd:int">550</page>
    </item>
  </BookArray>
</ns1:BookArrayResult>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

แต่ถ้าใช้ WSDL ในส่วนของการประกาศ schema จะต้องประกอบด้วย schema ดังนี้

```
<types>
  <schema targetNamespace='http://soapinterop.org/xsd'
    xmlns='http://www.w3.org/2001/XMLSchema'
    xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
```

```
xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
elementFormDefault='qualified'>
  <complexType name="Book">
    <sequence>
      <element name='author' type='string'/>
      <element name='name' type='string'/>
      <element name='page' type='int'/>
    </sequence>
  </complexType>
  <complexType name ='ArrayOfBook'>
  <complexContent>
    <restriction base='SOAP-ENC:Array'>
      <attribute ref='SOAP-ENC:arrayType' wsdl:arrayType='typens:Book[]'/>
    </restriction>
  </complexContent>
 </complexType>
</schema>
</types>
```

8.5 SOAP ใน HTTP

ในการส่ง SOAP ผ่านทาง HTTP จะต้องใช้ content-type เป็น text/xml แต่ใน SOAP request จะต้องมี header SOAPAction ภายใน HTTP header SOAPAction จะเป็นตัวบอกให้เซิร์ฟเวอร์รู้ว่า HTTP Post นั้นเป็น SOAP เมสเซจ และค่าของ header คือ URI ที่แสดงถึงจุดหมายของ SOAP เมสเซจ ส่วน SOAP response จะต้องมี status code ตามมาตรฐานของ HTTP โดย 200-299 แสดงว่าสำเร็จ แต่ถ้า response เมสเซจ เป็นการ fault แล้ว status code จะต้องเป็น 500 ซึ่งแสดงถึง internal server error ตัวอย่างของ response ที่มี status code เป็น 500 อาจเป็นดังนี้

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml
Content-Length: ###
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

```
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:VersionMismatch</faultcode>
        <faultstring>The SOAP namespace is incorrect.</faultstring>
      <faultactor>http://www.wrox.com/endpoint.asp</faultactor>
      <detail>
        <w:errorinfo xmlns:w="http://www.wrox.com/">
          <desc>The SOAP namespace was blank.</desc>
        </w:errorinfo>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

8.6 SOAP สำหรับ RPC

จุดประสงค์ของการออกแบบ SOAP คือทำ RPC โดยใช้ XML การเรียก RPC นั้นจะ map เข้ากับ HTTP request และ RPC response จะ map กับ HTTP response

ในการทำ mechod call จำเป็นที่จะต้องมีอินฟอร์เมชันดังนี้

1. URI ของออบเจ็กต์ปลายทาง (target object)
2. ชื่อของเมธอด
3. คำอธิบายของเมธอด (method signature) ส่วนนี้เป็นออปชัน
4. พารามิเตอร์สำหรับเมธอด
5. ข้อมูลส่วนหัว (header data) เป็นออปชัน

พิจารณาเมธอดดังนี้

```
double GetStockQuote ( [in] stirng sSymbol);
```

ถ้า namespace ของเมธอดคือ "http://www.worxstox.com/" แล้วการเรียกเมธอดโดย request stock quote ใช้ symbol OU812 จะเป็นดังนี้

```
<q:GetStockQuote xmlns:q="http://www.wroxstox.com/">
  <q:sSymbol xsi:type="xsd:string">0U812</q:sSymbol>
</q:GetStockQuote>
```

ชื่อเมธอดและชื่อของอีลีเม็นต์จะต้อง match กันเช่นเดียวกับพารามิเตอร์

สำหรับ response จะต้องชื่ออีลีเม็นต์จะต้องเป็นชื่อเมธอดตามด้วย response ดังนี้

```
<q:GetStockQuoteResponse xmlns:q="http://www.wroxstox.com/"
   <q:ret xsi:type="xsd:double">100.0</q:ret>
</q:GetStockQuoteResponse>
```

ชื่ออีลีเม็นต์ของค่าที่คืนมาสามารถเปลี่ยนได้ โดยจะกำหนดได้จากโปรแกรมที่เขียนหรือจาก WSDL

8.7 SOAP Toolkit

SOAP toolkit คือเครื่องมือที่จะทำหน้าที่ในการประมวลผล SOAP request และส่ง response กลับไปให้ไคลเอ็นต์ โดย SOAP toolkit แต่ละตัวใช้ได้บนแพล็ตฟอร์มที่ต่างกัน รองรับเทคโนโลยี web service ต่างกัน บางตัวอาจจะรองรับ WSDL หรือ UDDI ในขณะที่บางตัวรองรับอย่างใดอย่างหนึ่งหรือไม่รองรับทั้งสองอย่าง และถึงแม้ว่า SOAP จะเป็นอิสระต่อแพล็ตฟอร์ม แต่ใน SOAP toolkit บางตัวยังไม่สามารถทำงานข้ามผลิตภัณฑ์ (interoperability) ได้ เนื่องจากรองรับเทคโนโลยีของ web service ไม่เหมือนกัน เช่น xsi ในบางผลิตภัณฑ์จะบังคับให้ SOAP เมสเซจจะต้องระบุ type ด้วย xsi หากไม่มีก็จะ falut ซึ่งในปัจจุบันแต่ละผลิตภัณฑ์ก็พยายามพัฒนาให้สามารถทำงานด้วยกันได้

ชื่อ	แพล็ตฟอร์ม	แหล่งที่มา
4s4c	COM	http://www.4s4c.com
A SOAP for RPC NT Service	COM	http://www.whitemesa.com
Apache Axis	Java	http://xml.apache.org/axis
Apache SOAP	Java	http://xml.apache.org/soap

CapeConnect	Java	http://www.capeclear.com
Glue	Java	http://www.themindelectric.com
IBM Web Services Toolkit	Java	http://www.alphaworks.ibm.com
IdooXoap for Java and C++	Java , C++	http://www.zvon.org
Microsoft SOAP Toolkit	COM	http://www.microsoft.com
PocketSOAP	COM	http://www.pocketsoap.com
SOAP::Lite	Perl	http://www.soaplite.com
Visual Studio .NET	COM	http://www.microsoft.com

ตารางที่ 8-3 SOAP Toolkit

8.8 โครงสร้างภายในของ SOAP Toolkit ของ Microsoft

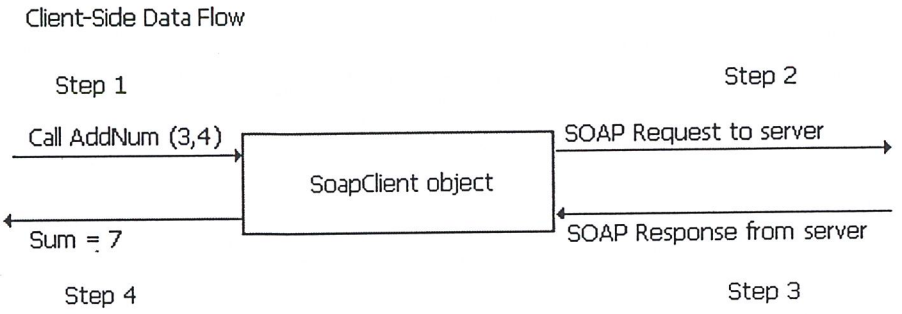
The Web Services Description Language (WSDL) เป็นรูปแบบหนึ่งของ XML เพื่อใช้อธิบายว่าเซิร์ฟเวอร์นั้นมีเซอร์วิส (Services) และ operation ภายในเซอร์วิสอะไรให้ใช้บ้าง ซึ่ง operation ภายในเซอร์วิสนั้นจะอธิบายรูปแบบที่ไคลเอนต์ (client) ต้องใช้ในการเรียก operation

The Web Services Meta Language (WSML) file เป็นส่วนที่เพิ่มขึ้นมาจากไฟล์ WSDL เพื่อใช้ในการเชื่อมโยงแต่ละ operation ในเซอร์วิสกับ COM object โดย WSML file จะพิจารณาว่า COM object ตัวใดที่จะต้องถูกโหลด (load) ขึ้นมาเมื่อมีการเรียกใช้ operation ในเซอร์วิส

ทั้งไฟล์ WSDL และ WSML จะต้องอยู่ที่เซิร์ฟเวอร์ โดยไคลเอนต์ที่ต้องการที่จะส่ง SOAP request ไปที่เซิร์ฟเวอร์ จะต้องได้รับ WSDL จากเซิร์ฟเวอร์ก่อน เพื่อหารู้รูปแบบของ SOAP request ในการติดต่อ

8.8.1 การสร้าง SoapClient Object

ทางฝั่งไคลเอนต์ เมื่อแอพพลิเคชั่นส่ง message ไปที่ Soap Client object เพื่อเรียกใช้ operation หนึ่งๆ นั้น(ตามรูปนี้คือการบวกเลข 2 ตัว) SoapClient object จะทำการสร้าง SOAP request ส่งไปที่เซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ทำการประมวลผลเสร็จเรียบร้อยแล้ว ก็จะส่งผลลัพธ์ในรูปแบบของ SOAP response กลับมาให้ไคลเอนต์ หลังจากนั้น SoapClient object จึงจะทำการแปลและส่ง message ที่เก็บผลลัพธ์ของ operation นั้นกลับไปให้แอพพลิเคชั่น ตามไดอะแกรมด้านล่าง



รูปที่ 8-2 แสดงการ Client –Side Data Flow

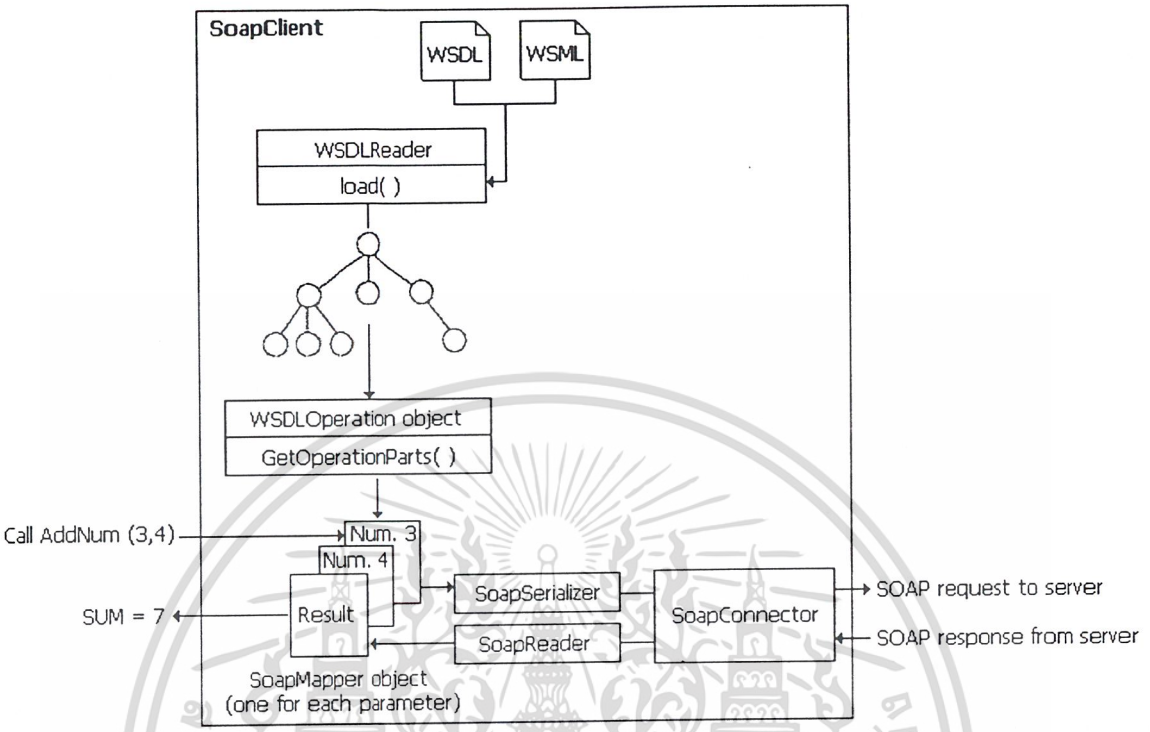
แอพพลิเคชั่นต้อง instantiate Soap object ก่อนที่จะทำการเรียกใช้ operation ใดๆ (SoapClient object นี้ถูกสร้างเก็บไว้ในไฟล์ MSSOAP1.dll) โดยใช้เมธอด SoapClient.mssoapinit (ชื่อไฟล์ WSDL, ชื่อเซอร์วิส, ข้อมูลพอร์ต) โดยข้อมูลพอร์ตจะใช้ในการแบ่งแยกช่องทางที่ใช้ในการส่ง SOAP message

ในระหว่าง initialize SoapClient object จะตรวจสอบเมธอดทั้งหมดที่อยู่ในไฟล์ WSDL ด้วยการทำงานแบบไดนามิคนี้ทำให้เราสามารถที่จะเรียกใช้เมธอดใดก็ได้ที่ได้อธิบายไว้ในไฟล์ WSDL ดังแสดงตาม Microsoftฎ Visual Basicฎ Scripting Edition (VBScript) code ด้านล่างนี้

```
SET soapclient = CreateObject("MSSOAP.SoapClient")
Call soapclient.mssoapinit("http://localhost/MyWSDL.wsdl", "TestService", "TestPort")
' Now the client can call an operation listed in the portType element
' specified when calling mssoapinit().
wscript.echo soapclient.AddNumbers(2, 3)
```

8.8.2 การประมวลผลภายใน SoapClient Object

การทำงานภายใน SoapClient object แสดงได้ดังรูปด้านล่าง



รูปที่ 8-3 แสดงโครงสร้างภายในของ SOAP

WSDL Reader object จะทำการโหลดไฟล์ WSDL และไฟล์ WSML เข้าไปไว้ใน DOM แล้วทำการวิเคราะห์ โดย WSDLReader object จะสร้าง WSDLOperation object เพื่อทำการเรียกใช้งาน operation (ตามรูปนี้คือ AddNum(3, 4)) หลังจากนั้น WSDLOperation จะเรียกเมธอด GetOperationParts เพื่อดึงรูปแบบ message ของอินพุทและเอาท์พุทของ operation ออกมา แล้ว SoapClient จะสร้าง Soap Mapper objects ให้กับแต่ละส่วนที่ได้จากเมธอด GetOperationParts และโหลดค่าที่ได้มาจากการเรียกใช้ operation เข้าไว้ใน objects เหล่านี้ พอถึงขั้นตอนนี้ SoapSerializer objects จะสร้าง SOAP request message จาก SoapMapper objects ที่เหมาะสมและส่งไปที่เซิร์ฟเวอร์

เมื่อเซิร์ฟเวอร์ประมวลผลเสร็จแล้วก็จะส่ง SOAP response กลับไปที่ไคลเอนต์ SoapClient object จะทำการโหลดผลลัพธ์ของ operation ไปเก็บใน SoapMapper object ที่เหมาะสมแล้วส่งผลลัพธ์กลับไปให้แอพพลิเคชั่นของผู้ใช้

8.8.3 การสร้าง SoapServer Object

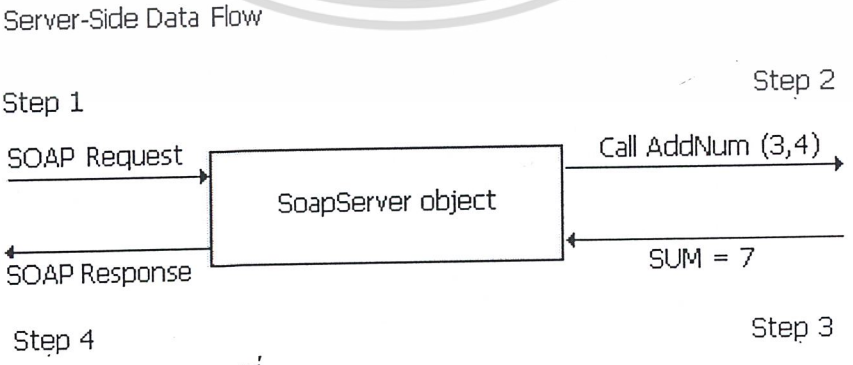
ส่วนทางฝั่งเซิร์ฟเวอร์ เราสามารถกำหนดรูปแบบการรับ SOAP request (SOAP listener) ได้ 2 รูปแบบ คือ Internet Server API (ISAPI) server และActive Server Pages (ASP) server

ในไฟล์ WSDL จะมี URL ที่ระบุว่าจะใช้ไฟล์ใดบนเซิร์ฟเวอร์ในการรับ SOAP request เพื่อใช้ในการพิจารณาว่าจะรับ SOAP request อย่างไรบนเซิร์ฟเวอร์ ตามโค้ดด้านล่างนี้ ตัวพิมพ์หนาจะเป็นส่วนที่ใช้ในการเรียกไปที่ ISAPI listener

```
<definitions>
...
 <service name='DocSample1' >
  <port name='DocSample1PortType' binding='tns:DocSample1Binding' >
   <soap:address
     location='http://localhost/DocSample1Test/DocSample1.wsdl' />
  </port>
 </service>
...
</definitions>
```

การทำงานทางฝั่งเซิร์ฟเวอร์

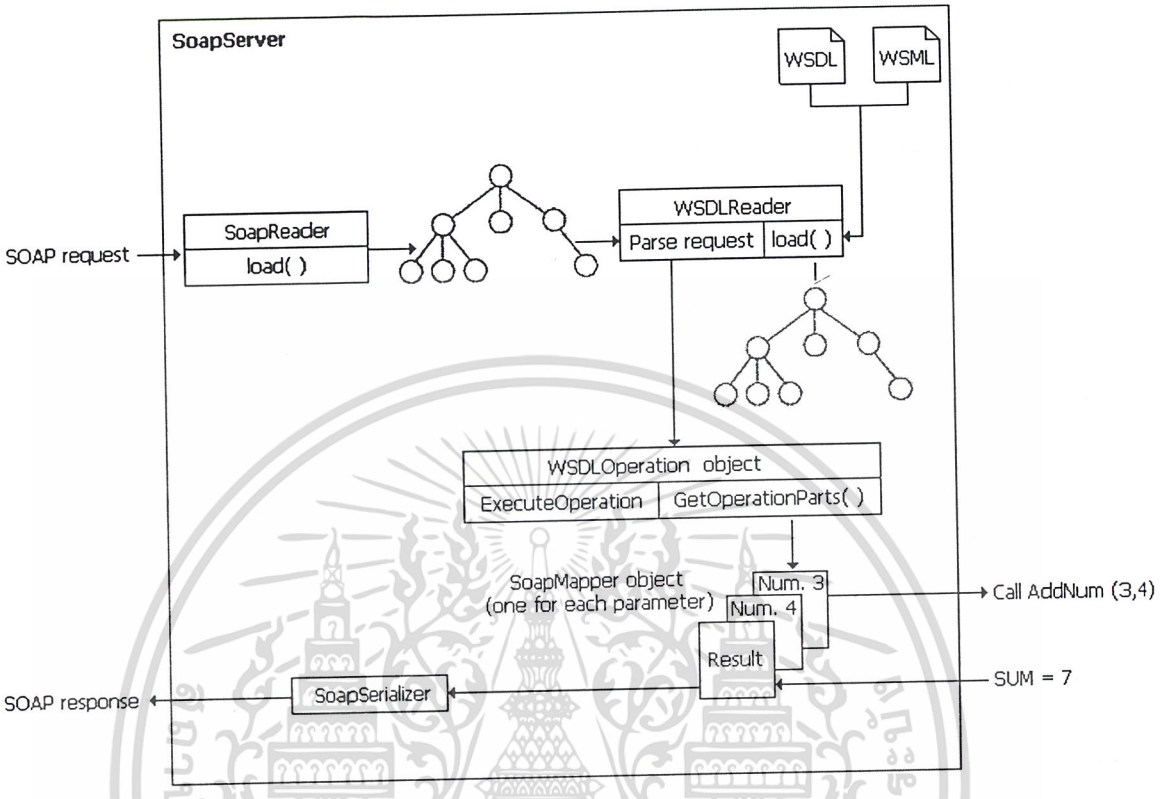
ไม่ว่า SOAP request จะเรียกไปยัง ISAPI หรือ ASP listener ก็ตาม ข้อมูลที่รับส่งก็ยังเหมือนกัน เมื่อ SoapServer object ได้รับ SOAP request จากไคลเอนต์ ก็จะทำการแปลและเรียกไปยังเมธอด COM ที่ตรงกับ operation ที่ต้องการ (ตามไดอะแกรมนี้คือเมธอด AddNum) หลังจากนั้นเมธอดที่ถูกเรียกใช้งนก็จะส่งผลลัพธ์กลับมาให้ SoapServer object ซึ่งมันก็จะแปลงผลลัพธ์ให้อยู่ในรูปของ SOAP response แล้วจึงส่งกลับไปที่ไคลเอนต์



รูปที่ 8-4 แสดง Server-Side Data Flow

8.8.4 การประมวลผลภายใน SoapServer Object

การทำงานภายใน SoapServer object แสดงได้ดังรูปด้านล่าง



รูปที่ 8-5 แสดงโครงสร้างภายใน SOAP Server

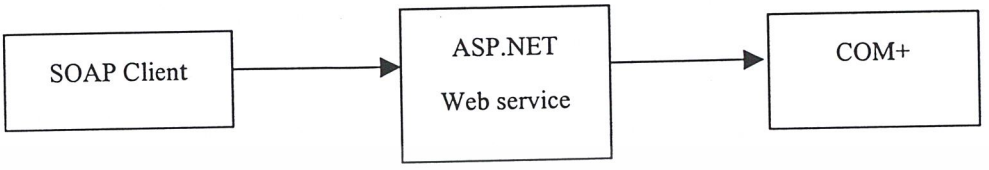
เมื่อได้รับ SOAP request จากไคลเอนต์แล้ว SoapReader object ก็จะโหลด request message ไปไว้ใน DOM structure หนึ่ง ในขณะที่ WSDLReader จะโหลดไฟล์ WSDL และไฟล์ WSMLไปไว้ในอีก DOM structure หนึ่ง ตัว**WSDLReader** จะแปล request และสร้าง WSDLOperation object ขึ้นเพื่อใช้ใน requested operation หลังจากนั้น WSDLOperation object จะเรียกใช้เมธอด GetOperationParts เพื่อดึงรูปแบบ message ของอินพุทและเอาท์พุทของ operation ออกมา แล้ว SoapServer จะสร้าง Soap Mapper objects ให้กับแต่ละส่วนที่ได้จากเมธอด GetOperationParts และโหลดค่าที่ได้มาจากการเรียกใช้ operation เข้าไว้ใน objects เหล่านี้ พอถึงขั้นตอนนี้ SoapServer object จะทำการเรียกไปที่ COM method ที่ตรงกับที่ร้องขอเข้ามา

เมื่อ COM method ประมวลผลเสร็จแล้วก็จะส่งผลลัพธ์กลับไปที่ SoapServer object ซึ่ง SoapSErver object ก็จะทำการโหลดผลลัพธ์ของ operation ไปเก็บใน SoapMapper object ที่เหมาะสม แล้วจึงใช้ SoapSerializer object ส่ง SOAP response message กลับไปให้ไคลเอนต์แอพพลิเคชั่นของผู้ใช้

8.9 การสร้าง SOAP Client และ SOAP Server ด้วย .NET Framework

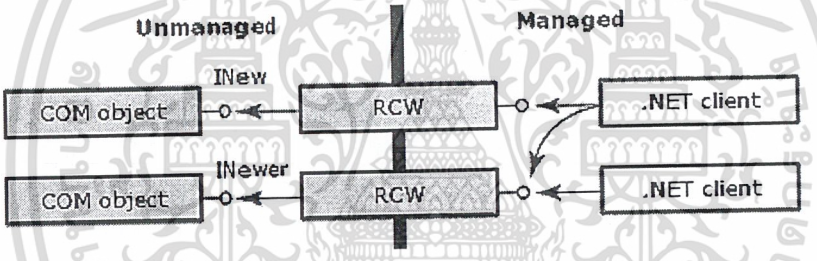
8.9.1 การใช้ ASP.NET ทำ SOAP Web Service

ใช้ความสามารถในการทำ COM+ Interop โดยสร้างคลาสของ ASP.NET มาเรียกใช้งาน business logic ที่สร้างด้วย VB6



รูปที่ 8-6 การใช้ ASP.NET ทำ SOAP web service

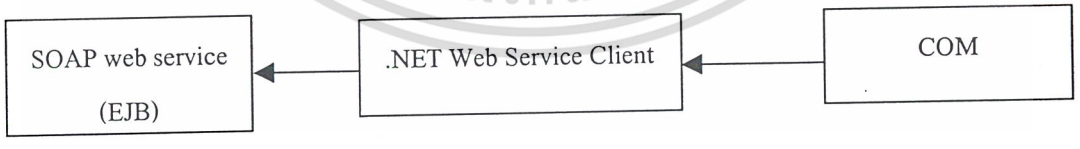
อาศัยความสามารถในการทำ marshall / unmarshall object ให้อยู่ในรูปแบบ SOAP ของ ASP.NET โดย ASP.NET ต้องทำ Interop โดยอาศัย Runtime Callable Wrapper ช่วยจำลอง COM object ให้ .NET มองเห็นเป็น .NET object ธรรมดา



รูปที่ 8-7 Runtime Callable Wrapper

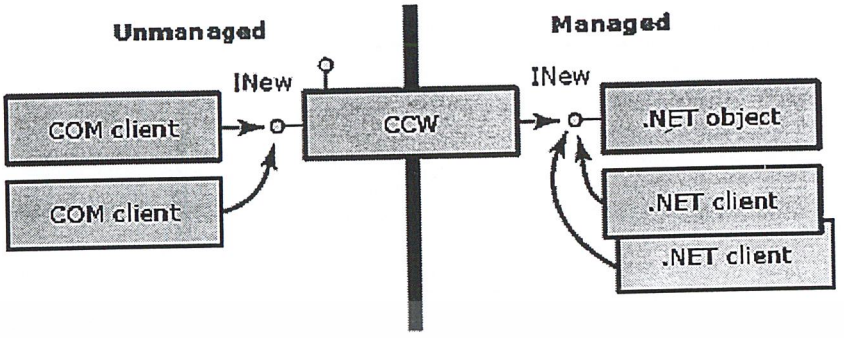
8.9.2 การใช้ .NET ทำ SOAP Client

ในทางกลับกันที่ COM ต้องการใช้บริการจาก SOAP ก็กระทำผ่าน .NET web service client



รูปที่ 8-8 การใช้ .NET ทำ SOAP client

แต่การที่จะทำให้ COM ติดต่อกับ .NET class ได้นั้นต้องผ่านกระบวนการ COM Interop โดยการสร้าง COM callable wrapper ซึ่งจะทำให้ COM สามารถเห็น .NET class เป็น COM object ธรรมดา



รูปที่ 8-9 COM callable wrapper ที่ทำให้ COM client สามารถเห็น .NET object

บทที่ 9
คลัสเตอร์

9.1 คลัสเตอร์ (Cluster) คืออะไร

คลัสเตอร์เป็นการเชื่อมเครื่องคอมพิวเตอร์หลายๆ เครื่องติดต่อกันผ่านระบบแลน ให้เป็นระบบเดี่ยว เพื่อให้มีความสามารถเพิ่มขึ้นในหลายๆ ด้าน ได้แก่ ความสามารถในการทำงานได้ตลอดเวลา (availability), เพิ่มขีดความสามารถในการขยายระบบ (scalability), เพิ่มความสามารถให้จัดการกับคลัสเตอร์โดยผ่านเครื่องๆเดียว (manageability)

ระบบปฏิบัติการ Microsoft® Windows® 2000 Advanced Server and Datacenter Server ได้รวมเทคโนโลยีคลัสเตอร์ไว้ 2 แบบ คือ

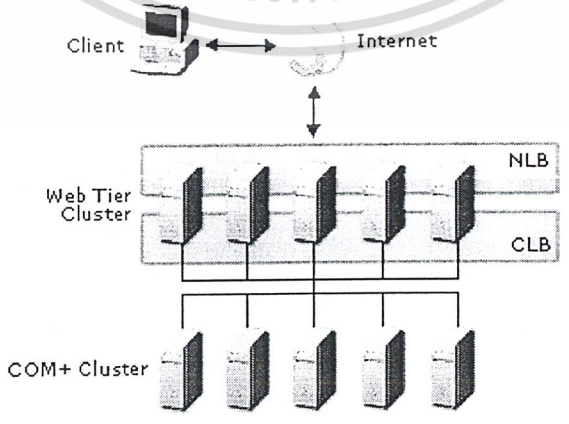
- คลัสเตอร์เซอร์วิส สำหรับการทำ เฟลโอเวอร์ (failover) *อธิบายในหัวข้อที่ 9.3*
- เน็ตเวิร์กโหลดบาลานซิง (Network Load Balancing) เพิ่มสเกลาบิลิตี (scalability) และอไวลาบิลิตี (availability) ให้กับเอ็นเตอร์ไพรส์แอพพลิเคชัน เช่น เว็บ, เทอร์มินัลเซอร์วิส, พรอกซี

การเพิ่มขีดความสามารถในการทำงาน (Scalability) เพิ่มประสิทธิภาพของเซิร์ฟเวอร์เบสโปรแกรม (Server-based program) เช่น เว็บเซิร์ฟเวอร์โดยกระจายไคลเอ็นต์รีเควสต์ไปยังหลายเซิร์ฟเวอร์ในคลัสเตอร์เดียวกัน โดยสามารถเพิ่มเซิร์ฟเวอร์ได้จนถึง 32 เครื่อง

การทำงานได้ตลอดเวลา (High availability) NLB เพิ่มอไวลาบิลิตีโดยตรวจจับความล้มเหลวของเซิร์ฟเวอร์โดยอัตโนมัติ และจัดแบ่งไคลเอ็นต์รีเควสต์ไปยังเซิร์ฟเวอร์ใหม่ (repartitioning) ภายใน 10 วินาที ทำให้ผู้ใช้สามารถใช้งานได้อย่างต่อเนื่อง

นอกจากนี้ยังมีคอมโพเน็นต์โหลดบาลานซิง (Component Load Balancing - CLB) ที่มากับแอพพลิเคชันเซ็นเตอร์ 2000 (Microsoft Application Center 2000)

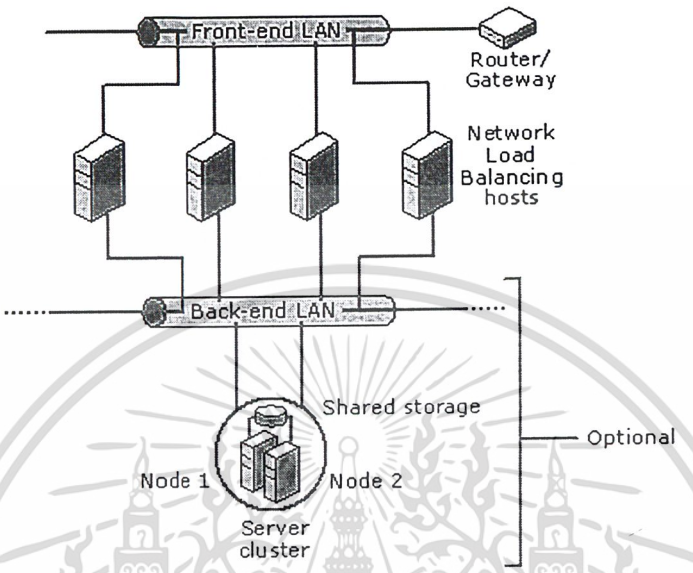
CLB มีความสามารถในการโหลดบาลานซ์ (load-balance) COM+ คอมโพเน็นต์ที่อยู่บนเซิร์ฟเวอร์หลายๆ เครื่อง เรียกกลุ่มของเซิร์ฟเวอร์นี้ว่า COM+ คลัสเตอร์ดังรูป



รูปที่ 9-1 COM+ cluster

9.2 เน็ตเวิร์กโหลดบาลานซิง (Network Load Balancing: NLB)

NLB กระจาย IP ทราฟฟิกไปยัง TCP/IP เซอร์วิสหลายตัวโดยที่แต่ละตัวทำงานบนคนละเซิร์ฟเวอร์บนคลัสเตอร์เดียวกัน NLB แบ่งไคลเอ็นต์รีเควสต์เป็นส่วนๆ ไปยังโฮสต์ต่างๆ โดยไคลเอ็นต์ไม่ต้องรับรู้ และทำให้ไคลเอ็นต์สามารถใช้งานคลัสเตอร์ โดยใช้ "virtual" IP addresses เพียงที่เดียวก็เพียงพอ



รูปที่ 9-2 การใช้งานเน็ตเวิร์กโหลดบาลานซิง

9.2.1 ข้อดีของ NLB

NLB นั้นมีข้อดีเหนือการโหลดบาลานซ์ด้วยซอฟต์แวร์อื่นๆ อย่างเช่น round robin DNS (RRDNS) ซึ่งสามารถกระจายโหลดไปยังหลายเซิร์ฟเวอร์ได้แต่ไม่มีกลไกในการตรวจสอบความล้มเหลวของเซิร์ฟเวอร์ ถ้าเซิร์ฟเวอร์ใดล้มเหลวแล้ว RRDNS ยังคงส่งงานต่อไปจนกระทั่งผู้ดูแลระบบเน็ตเวิร์กตรวจพบและนำเซิร์ฟเวอร์นั้นออกจาก DNS address list ทำให้ไคลเอ็นต์ได้รับการบริการที่ไม่ต่อเนื่อง

NLB ยังได้เปรียบการโหลดบาลานซ์แบบอื่นอีก ทั้งที่ใช้ฮาร์ดแวร์และซอฟต์แวร์ซึ่งมีจุดอ่อนตรงที่เป็น single point of failure หรือเป็นคอขวดของประสิทธิภาพการทำงานเพราะใช้ตัวกลางในการกระจายงาน (centralized dispatcher)

เนื่องจาก NLB ไม่เจาะจงความต้องการฮาร์ดแวร์ ดังนั้นอุปกรณ์คอมพิวเตอร์มาตรฐานทั่วไปก็สามารถใช้งานได้ ทำให้ลดค่าใช้จ่ายลงมากเมื่อเทียบกับการใช้ฮาร์ดแวร์ช่วยจัดการกระจายโหลด (Hardware Load-Balancer)

9.2.2 การติดตั้งและการจัดการ NLB

NLB เป็นคอมโพเน็นต์หนึ่งใน Microsoft Windows 2000 Advanced Server และ Datacenter Server ซึ่งถูกติดตั้งอย่างอัตโนมัติ โดยเราสามารถเปิดใช้งานได้ผ่านพรอพเพอร์ตีของระบบแลน

NLB เป็นบริการแบบเลือกได้ของการเชื่อมต่อ LAN เรียกว่า คลัสเตอร์อะแดปเตอร์ซึ่งไม่ต้องมีการเปลี่ยนแปลงฮาร์ดแวร์ใดๆในการติดตั้งและใช้งาน NLB เพราะ NLB นั้นทำงานได้กับทุกๆ อีเทอร์เน็ตอะแดปเตอร์ และ Fiber Distributed Data Interface (FDDI) อะแดปเตอร์

1. *IP Address*

เมื่อใช้งาน NLB จะต้องปรับแต่งพารามิเตอร์โดยใช้พรอพเพอร์ตีไดอะล็อกบ็อกซ์ (Properties dialog box) โดยคลัสเตอร์จะต้องถูกกำหนด primary IP address ซึ่งเป็น virtual IP address สำหรับทุกๆ โฮสต์ที่เป็นสมาชิกของคลัสเตอร์

รีโมตคอนโทรลโปรแกรม (remote control program) ซึ่งเป็นส่วนหนึ่งของ NLB จะใช้ primary IP address ในการระบุ คลัสเตอร์ที่ต้องการจะควบคุม

แต่ละคลัสเตอร์ยังสามารถมี dedicated IP address ซึ่งใช้ในการสื่อสารโดยตรงไปยังโฮสต์นั้น โดย NLB จะไม่โหลดบาลานซ์ dedicated IP address แต่จะโหลดบาลานซ์ทราฟฟิกขาเข้าทุกๆ IP ที่ไม่ใช่ dedicated IP address

2. *โฮสต์ ไพรออริตี (Host Priority)*

แต่ละคลัสเตอร์โฮสต์จะต้องถูกกำหนดค่าโฮสต์ไพรออริตี ที่ต่างกันมีค่าตั้งแต่ 1 ถึง 32 (หมายความว่าใน 1 คลัสเตอร์จะมีโฮสต์ ได้ 32 เครื่อง) โดยเลขที่ต่ำกว่าจะมีไพรออริตี ที่สูงกว่า ซึ่งโฮสต์ที่มีไพรออริตีสูงสุด (เลขต่ำที่สุด) จะเป็นค่าตั้งต้น โฮสต์ดังกล่าวจะรับผิดชอบไคลเอ็นต์ทราฟฟิกที่เป็น virtual IP address ที่ไม่ระบุความต้องการที่จะโหลดบาลานซ์ ทำให้สามารถมั่นใจว่าเซิร์ฟเวอร์แอพพลิเคชันที่ไม่ได้ปรับเพื่อการโหลดบาลานซ์ จะรับผิดชอบไคลเอ็นต์ทราฟฟิกที่เครื่องเครื่องเดียวเท่านั้น

3. *พอร์ต รูลส์ (Port Rules)*

NLB ใช้พอร์ต รูลส์สำหรับการปรับแต่งโหลดบาลานซ์ให้ใช้งานกับพอร์ตย่านต่างๆ

3.1 Multiple-host หมายความว่าไคลเอ็นต์รีเควสต์จะถูกกระจายไปยังทุกๆคลัสเตอร์โฮสต์ ตามสัดส่วนของเปอร์เซ็นต์ของโหลด

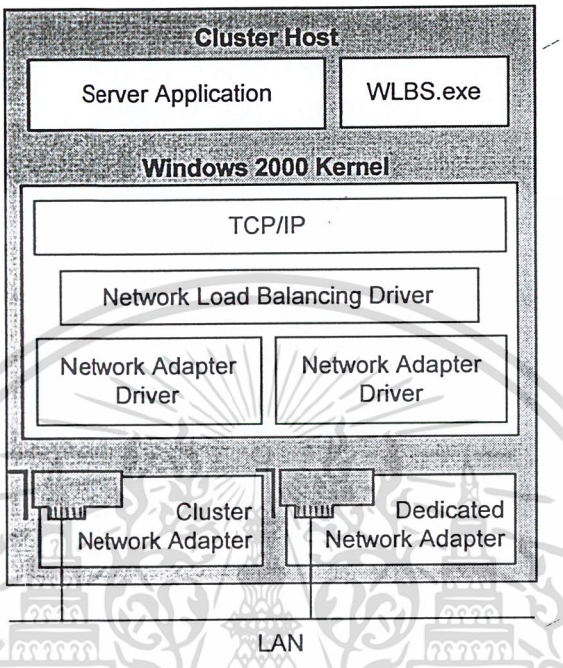
3.2 Single-host หมายความว่า การโหลดบาลานซ์จะส่งไคลเอ็นต์รีเควสต์ตรงไปยังโฮสต์ที่มี handling priority สูงสุด ซึ่ง handling priority จะใช้แทนโฮสต์ไพรออริตี สำหรับพอร์ตย่านต่างๆ และทำให้โฮสต์ต่างสามารถรับผิดชอบ เซิร์ฟเวอร์ แอพพลิเคชัน ที่ไม่เหมือนกันได้

9.2.3 สถาปัตยกรรมของ NLB

เพื่อเพิ่มทรูพุท (throughput) และอไวลาบิลิตีนั้น NLB ใช้สถาปัตยกรรมการกระจายอย่างสมบูรณ์ (a fully distributed software architecture)

ไดรเวอร์ NLB ที่เหมือนกันจะทำงานพร้อมกันบนแต่ละโฮสต์ในคลัสเตอร์ ไดรเวอร์จะทำให้คลัสเตอร์ที่อยู่บนซับเน็ต (subnet) เดียวกันสามารถรับเน็ตเวิร์กทราฟฟิกขาเข้าโดยใช้ primary IP address ของคลัสเตอร์

ในแต่ละคลัสเตอร์โฮสต์นั้น ไดรเวอร์จะทำหน้าที่เป็นตัวกรอง (filter) ระหว่างไดรเวอร์ของเน็ตเวิร์กอะแดปเตอร์กับ TCP/IP stack ทำให้โฮสต์สามารถเลือกรับเน็ตเวิร์กทราฟฟิกขาเข้าเพียงบางส่วนได้ ทำให้เกิดโหลดบาลานซ์ระหว่างคลัสเตอร์โฮสต์



รูปที่ 9-3 แสดง โครงสร้างเน็ตเวิร์กโหลดบาลานซิง

โครงสร้างของ NLB เพิ่มทรูพุท โดยใช้การบรอดคาสต์ซับเน็ต (broadcast subnet) ไปทุกๆ คลัสเตอร์โฮสต์ การใช้ตัวกรองจะเร็วกว่าการใช้ dispatcher-based เพราะไม่ต้องการขั้นตอนการเร้าต์ (route) แพ็กเก็ต และไม่ขึ้นกับฮาร์ดแวร์อิมพลีเมนต์เร้าต์ติ้ง (implement routing) ของอุปกรณ์ต่างๆ ด้วย โดยทรูพุทจะเพิ่มขึ้นตามความเร็วของเน็ตเวิร์กและเซิร์ฟเวอร์

ข้อดีของ NLB ที่เลือกใช้การกระจายอย่างสมบูรณ์ คือ การเพิ่มอไวลาบิลิตีโดยในคลัสเตอร์ที่มีจำนวนโฮสต์เท่ากับ N จะมีรูปแบบการเฟลโอเวอร์ (fail-over) อยู่ N-1 แบบ ซึ่งต่างกับ dispatcher-base ที่มี single point of failure ที่ต้องแก้ไขโดยใช้ dispatcher สำรอง ทำให้มีรูปแบบการล้มเหลวเพียงแบบเดียวเท่านั้น

9.2.4 อัลกอริทึมโหลดบาลานซิง

NLB ใช้อัลกอริทึมการกรองแบบกระจาย (distributed filtering) ที่จะส่งไคลเอ็นต์รีเควสต์ไปยังคลัสเตอร์โฮสต์ เพราะอัลกอริทึมทำให้คลัสเตอร์โฮสต์สามารถตัดสินใจได้อย่างอิสระและมีความรวดเร็ว

อัลกอริทึมนี้ถูกปรับแต่งให้ทำงานได้ดีกับไคลเอ็นต์ปริมาณมากซึ่งแต่ละไคลเอ็นต์มีรีเควสต์ปริมาณน้อยๆ ซึ่งโดยทั่วไปก็เป็นลักษณะไคลเอ็นต์ของเว็บเซิร์ฟเวอร์

เมื่อปริมาณไคลเอ็นต์มีน้อยและการเชื่อมต่อของไคลเอ็นต์ยาวนาน จะทำให้ NLB อัลกอริทึม มีประสิทธิภาพน้อย อย่างไรก็ตามด้วยความง่ายและความเร็วของอัลกอริทึมนี้ ทำให้ทำงานได้อย่างมีประสิทธิภาพสูง มีทรูพุทสูงและเวลาตอบสนอง (response time) ต่ำ เหมาะกับงานหลากหลายชนิดที่เป็นแบบไคลเอ็นต์/เซิร์ฟเวอร์ แอพพลิเคชัน

NLB จะโหลดบาลานซ์รีเควสต์โดยใช้ค่าเปอร์เซ็นต์โหลด ที่ตั้งไว้ในพรอพเพอร์ตีไดอะล็อกบ็อกซ์ หลังจากนั้นไคลเอ็นต์จะถูกกระจายไปยังคลัสเตอร์โฮสต์โดยวิธีทางสถิติ

โหลดบาลานซ์จะเปลี่ยนแปลงอย่างไดนามิก (dynamic) เมื่อมีโฮสต์ที่เข้าหรือออกจากคลัสเตอร์ แต่จะไม่ตอบสนองต่อโหลดบนแต่ละโฮสต์ (CPU load, memory usage)

เมื่อมีแพ็กเก็ตเข้ามา ทุกๆ โฮสต์จะแมพ (map) พร้อมๆ กันเพื่อจะได้รู้ว่าโฮสต์ใดควรจะรับแพ็กเก็ตนั้นๆ

การแมพใช้ฟังก์ชันสุ่ม (randomize function) คำนวณหาค่าโฮสต์ไพรออริตี โดยใช้ข้อมูล IP ของไคลเอ็นต์ พอร์ต และข้อมูลของสเตจ เพื่อ optimize โหลดบาลานซ์

โฮสต์ที่รับผิดชอบก็จะส่งแพ็กเก็ตต่อไปยัง TCP/IP ส่วนโฮสต์อื่นก็ทิ้งแพ็กเก็ตนั้นไป การแมพนี้จะไม่เปลี่ยนแปลงจนกระทั่งมีการเปลี่ยนสมาชิกของคลัสเตอร์โฮสต์เพื่อที่จะแน่ใจว่า IP address และพอร์ตหนึ่งจะแมพไปยังโฮสต์เดิมเสมอ

โดยทั่วไปคุณภาพของโหลดบาลานซ์จะขึ้นอยู่กับจำนวนของไคลเอ็นต์รีเควสต์ เทียบได้กับการทอยเหรียญที่สถิติของการออกด้านหัวจะเข้าใกล้ความน่าจะเป็นที่ ½ เมื่อเพิ่มจำนวนการทอยให้มากขึ้น

อัลกอริทึมการโหลดบาลานซ์ มีสมมติฐานว่า ไคลเอ็นต์ IP address และหมายเลขพอร์ตเป็นอิสระในเชิงสถิติ ซึ่งสมมติฐานนี้จะผิดพลาดได้ ถ้าไฟร์วอลล์ (firewall) ของเซิร์ฟเวอร์แทนที่ไคลเอ็นต์ address ด้วย IP เดี่ยว ในขณะเดียวกันก็เปิดใช้งานไคลเอ็นต์ affinity ไปด้วยจะทำให้ไคลเอ็นต์รีเควสต์ทั้งหมดไปทำงานที่โฮสต์เดียว อย่างไรก็ตาม ถ้าไม่เปิดใช้งานไคลเอ็นต์ affinity ก็จะทำให้การกระจายงานเป็นไปอย่างปกติ

9.2.5 Convergence – Redistributed the Load on NLB Cluster

หลายสถานการณ์ที่ต้องมีการไคลเอ็นต์ต้องแมพใหม่ เพราะมีการเปลี่ยนแปลงสมาชิกของคลัสเตอร์ คือ สมาชิกของคลัสเตอร์ออกจากคลัสเตอร์ หรือเข้าร่วมคลัสเตอร์จะทำให้เกิด event triggers convergence นำไปสู่การคำนวณรายการสมาชิกของคลัสเตอร์ใหม่ และคำนวณการแมพจากไคลเอ็นต์รีเควสต์ไปยังสมาชิกของคลัสเตอร์อีกครั้ง

9.2.6 NLB Heart Beat

NLB ใช้กลไกของ heart beat เพื่อตรวจสอบสถานะของโฮสต์ต่างๆซึ่ง message ที่ส่งหากันเป็น Ethernet-level broadcast ซึ่งส่งไปทุกๆ สมาชิกของโหลดบาลานซ์คลัสเตอร์

ช่วงเวลาปกติอยู่ที่ 1 วินาที ซึ่ง NLB จะคิดว่าสมาชิกทำงานเป็นปรกติถ้าการแลกเปลี่ยน heartbeat message เป็นไปอย่างปรกติ

ถ้าไม่ได้รับ message เป็นช่วงเวลาหนึ่ง NLB จะเริ่มการ convergence โดยปรกติช่วงเวลาดังกล่าวจะตั้งไว้ที่ 5 วินาที

ตามค่าปรกติของ NLB จะใช้เวลา 5 วินาทีสำหรับการตรวจพบโฮสต์ที่ล้มเหลว และอีก 5 วินาทีสำหรับการกระจายคลัสเตอร์ครั้งใหม่

ระหว่างการ convergence นั้น NLB จะลดคาบเวลาของ heartbeat ลงครึ่งหนึ่งเพื่อเร่งกระบวนการ convergence ให้เสร็จเร็วขึ้น

9.2.7 Load Balancing Distribution

มี 3 อัลกอริทึม แบ่งตามไคลเอ็นต์ affinity ซึ่งเป็นส่วนหนึ่งของพอร์ตรูลส์

1. No Affinity

การกระจายโหลดจะใช้การกรองแบบกระจายที่จะส่งไคลเอ็นต์รีเควสต์ไปยังคลัสเตอร์โฮสต์ โดย NLB จะไม่สร้างความสัมพันธ์ระหว่างไคลเอ็นต์กับคลัสเตอร์โฮสต์ทำให้ไคลเอ็นต์สามารถโหลดบาลานซ์ไปได้ทุกๆ โฮสต์

เมื่อมีแพ็กเก็ตมาถึง ทุกๆ โฮสต์จะตัดสินใจพร้อมๆ กันว่าโฮสต์ใดควรจะรับผิดชอบรีเควสต์ซึ่งใช้ฟังก์ชันสุ่ม คำนวณว่าสมาชิกคลัสเตอร์ตัวใดควรประมวลผลแพ็กเก็ตนั้น โดยใช้ IP address ของไคลเอ็นต์เป็นตัวตัดสินใจ

affinity แบบนี้ มีประสิทธิภาพสูงที่สุด แต่ไม่เหมาะกับการทำงานที่มีการเก็บข้อมูลเซสชัน (session) ไว้ที่เซิร์ฟเวอร์เพราะไคลเอ็นต์จะไม่สามารถทำงานได้เมื่อถูกโหลดบาลานซ์ไปยังเซิร์ฟเวอร์ที่ไม่มีเซสชันเดิม

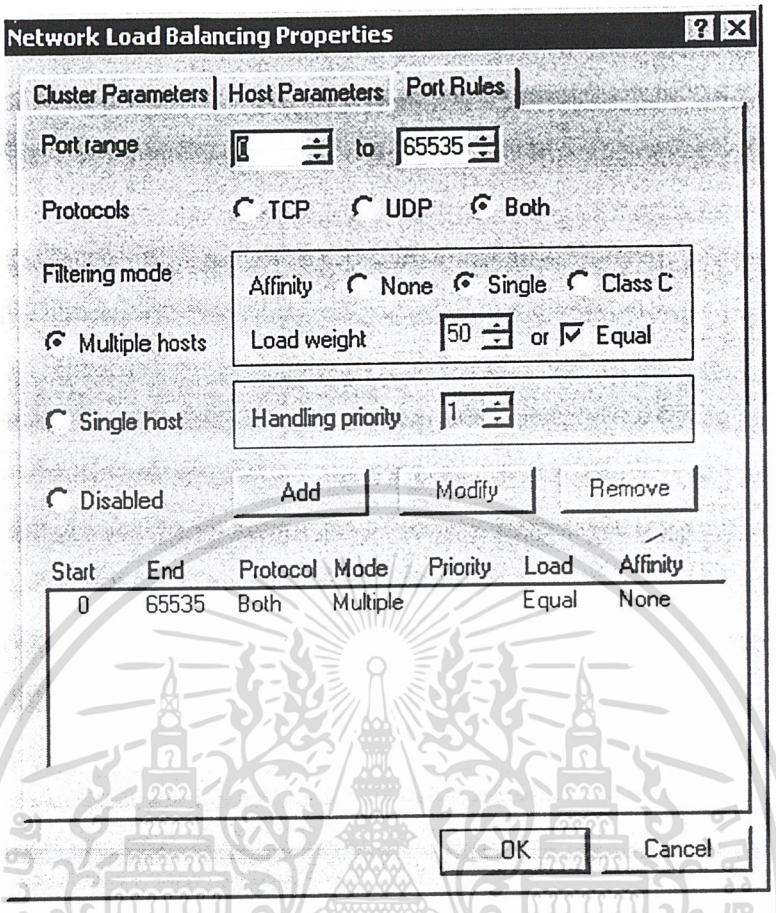
2. Single Affinity

NLB จะสร้างความสัมพันธ์ระหว่างไคลเอ็นต์กับโฮสต์โดยใช้ IP แบบเต็มของไคลเอ็นต์ (ไม่ใช้ข้อมูลหมายเลขพอร์ต) ทำให้รีเควสต์ที่มี IP เดียวกันจะทำงานบนโฮสต์เดียวกันเสมอ

Affinity แบบนี้ทำงานได้ดีที่สุดสำหรับไคลเอ็นต์ที่ใช้เซสชันในอินทราเน็ต (intranet) ซึ่งไคลเอ็นต์ประเภทนี้ไม่เหมาะกับ No Affinity เพราะเซสชันอาจผิดพลาดและไม่เหมาะกับกับ Class C Affinity เพราะอินทราเน็ตไคลเอ็นต์มักมี IP อยู่ในย่านแคบๆ ทำให้ทุกๆ IP อยู่ใน class C เดียวกัน ส่งผลให้คลัสเตอร์โฮสต์เดียวต้องรับผิดชอบไคลเอ็นต์ทั้งหมด ในขณะที่โฮสต์อื่นๆ นั้นไม่ได้รับงานเลย

3. Class C Affinity

หมายเลขพอร์ตของไคลเอ็นต์ จะไม่มีส่วนในการคำนวณการกระจายโหลดซึ่งอัลกอริทึมการแมพ จะใช้ส่วนหนึ่งของ Class C IP address (24 bit บน) ของไคลเอ็นต์ทำให้รีเควสต์ที่อยู่ใน Class C เดียวกันจะรับผิดชอบโดยโฮสต์เดียวกันเสมอ ซึ่ง affinity แบบนี้ ให้ประสิทธิภาพสูงสุดกับการทำงานบนอินเตอร์เน็ต



รูปที่ 9-4 การตั้งค่า affinity

9.3 Component Load Balancing (CLB)

CLB คือซอฟต์แวร์ที่ทำไดนามิกโหลดบาลานซิง สำหรับ COM+ แอพพลิเคชันคอมโพเน็นต์

<u>หมายเหตุ</u> มีเพียง COM+ activation เท่านั้นที่สามารถโหลดบาลานซ์โดย CLB ซึ่งคิว (queued) คอมโพเน็นต์ไม่สามารถโหลดบาลานซ์ได้

การแยกคอมโพเน็นต์เทียร์ออกจาก COM+ คอมโพเน็นต์รีเควสต์ทำให้เกิดลักษณะ ดังนี้

- สามารถ scale out คอมโพเน็นต์เทียร์อย่างอิสระต่อเว็บเทียร์
- สามารถกระจายเวิร์กโหลด (work load) ระหว่างเทียร์

  <u>หมายเหตุ</u> โดยปกติ การสเกลเว็บเทียร์ จะได้ประสิทธิภาพที่ดีกว่าการกระจายงานไปยังคอมโพเน็นต์เซิร์ฟเวอร์เทียร์
- สามารถจัดการบิสิเนสลอจิกเทียร์อย่างอิสระต่อฟรอนต์เอนด์ (front-end) เว็บเทียร์หรือแบ็กเอนด์ดาต้าเบสเทียร์
- เพิ่มเลเยอร์ของการรักษาความปลอดภัยแอพพลิเคชัน

9.3.1 สถาปัตยกรรมของ CLB

- Routing List

หลังจากการสร้างเว็บคลัสเตอร์หรือ COM+ Routing Server (สมาชิก ฟรอนต์เอนด์ members) จะต้องระบุสมาชิกของ COM+ แอพพลิเคชันคลัสเตอร์ (สมาชิกแบ็กเอนด์) ที่ต้องการให้รับผิดชอบคอมโพเน็นต์รีเควสต์

รายชื่อของคอมโพเน็นต์เซิร์ฟเวอร์ถูกเก็บใน metabase และ registry ซึ่ง CLB จะใช้ในการตัดสินใจว่าสมาชิกใน COM+ แอพพลิเคชันคลัสเตอร์ตัวใดต้องถูก poll

- Response Time Table

CLB ซอฟต์แวร์ที่ทำงานบนแต่ละเครื่องของเว็บคลัสเตอร์หรือ COM+ Routing Server จะ poll เครื่องที่อยู่ใน routing list ทุก 200 มิลลิวินาที จากนั้น จะสร้าง in-memory table ที่มีรายการของสมาชิก COM+ คลัสเตอร์เรียงลำดับตาม response time (มี response time น้อย ก็จะอยู่อันดับต้นๆ)

เว็บคลัสเตอร์หรือ COM+ routing เซิร์ฟเวอร์จะส่งผ่าน activation รีเควสต์ไปยังสมาชิกของ COM+ คลัสเตอร์ตามลำดับใน response time table แบบ round robin ซึ่งจะทำเป็นวงรอบไปจนกว่าจะมีการ poll ครั้งใหม่ และมีการจัดลำดับใน response time table อีกครั้ง

- The CLB Activator

โปรแกรมนี้จะประมวลผล คำสั่ง CoCreateInstance และ CoGetClassObject ที่เข้ามาสำหรับคอมโพเน็นต์ที่กำกับให้เป็น “support dynamic load balancing”

หลังจากตรวจสอบ response time table โปรแกรมจะเปลี่ยนค่า RemoteServerName ที่เข้ามาให้เป็นชื่อของคอมโพเน็นต์เซิร์ฟเวอร์ที่ควรจะรับผิดชอบรีเควสต์นั้นๆ

COM+ บน routing เซิร์ฟเวอร์จะส่งต่อรีเควสต์ไปยัง COM+ เซิร์ฟเวอร์ ที่ถูกเลือกซึ่งจะสร้างออบเจ็กต์ และส่งกลับไปให้ไคลเอ็นต์ การเรียกเมธอดที่ต่อจากนี้จะกระทำโดยตรงจากไคลเอ็นต์ไปยังคอมโพเน็นต์เซิร์ฟเวอร์ไปจนกระทั่งหมดช่วงชีวิตของออบเจ็กต์

9.3.2 Component Load Balancing Scenarios

มี 3 CLB โมเดลหลักที่แอพพลิเคชันเซ็นเตอร์สนับสนุน

- Two-tier กับโหลดบาลานซิงแบบเต็ม (with full load balancing)

  ฟรอนต์เอนด์ เว็บคลัสเตอร์ส่งผ่านรีเควสต์ไปยังแบ็กเอนด์ CLB คลัสเตอร์
- Three-tier กับโหลดบาลานซิงแบบเต็ม (with full load balancing)

  ฟรอนต์เอนด์ เว็บคลัสเตอร์ส่งผ่านรีเควสต์ไปยังโหลดบาลานซ์มิดเดิลเทียร์ที่เร้าต์รีเควสต์ไปยัง CLB คลัสเตอร์
- Three-tier กับเฟลโอเวอร์

  ฟรอนต์เอนด์เว็บคลัสเตอร์ส่งผ่านรีเควสต์ไปยังสมาชิก 2 ตัวในมิดเดิลเทียร์ (สมาชิก 1 ตัวเป็นแบ็กอัพ) ที่เร้าต์รีเควสต์ไปยังแบ็กเอนด์คลัสเตอร์

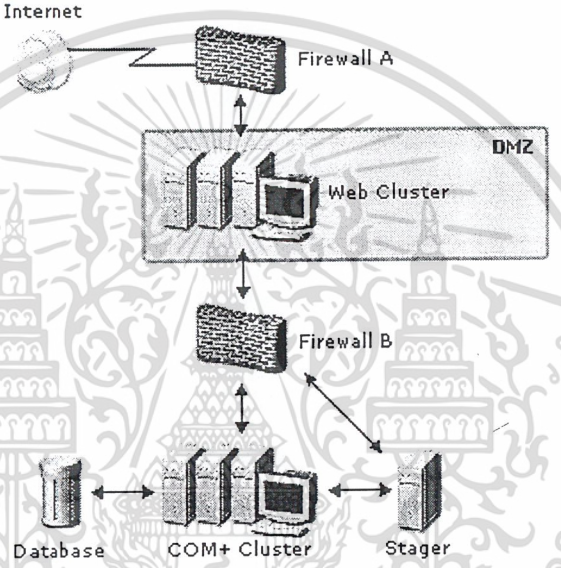
เมื่อใดที่ควรใช้ multi-tier โหลดบาลานซิง

ถึงแม้ว่าแอพพลิเคชัน เซ็นเตอร์จะสนับสนุนรูปแบบ multi-tier แต่ก็ไม่ได้หมายความว่าควรใช้รูปแบบนี้เสมอ

ต้องวิเคราะห์ทั้งความต้องการทั้งด้านเทคนิคและธุรกิจ ก่อนที่จะตัดสินใจว่าจะแยกหรือไม่แยกคอมโพเน็นต์เทียร์

เหตุผลหลักที่ควรแยกเว็บออกจาก COM+ แอพพลิเคชันเซิร์ฟเวอร์เทียร์ คือ

- การรักษาความปลอดภัย สามารถเพิ่มเลเยอร์ ของ ไฟร์วอลล์ ระหว่างเทียร์ COM+ คลัสเตอร์ถูกปกป้องด้วยไฟร์วอลล์ที่อนุญาติเฉพาะให้สร้างคอมโพเน็นต์ที่มีคำสั่งมาจากเว็บเทียร์เท่านั้น



รูปที่ 9-5 COM+ คลัสเตอร์หลังไฟร์วอลล์

- Administrative partitioning
  แยกกลุ่มผู้พัฒนาและดูแล HTML/ASP ออกจาก COM+ แอพพลิเคชัน ช่วยป้องกันปัญหาที่จะเกิดระหว่างกลุ่ม
- แบ่ง COM+ แอพพลิเคชัน ใช้ระหว่างหลายๆ เว็บคลัสเตอร์
- บางสถานการณ์ ถ้าแยก COM+ เซิร์ฟเวอร์ ออกมาจะทำให้เวลาตอบสนองดีขึ้น เช่นเว็บเซิร์ฟเวอร์ ที่บริการ HTML page เป็นหลัก

9.3.2.1 เหตุผลหลักที่ไม่ควรแยกเว็ปและ COM+ แอพพลิเคชัน ออกจากกัน

- ประสิทธิภาพ

Remote access มีต้นทุนสูงกว่าแบบ local ดังตารางด้านล่าง

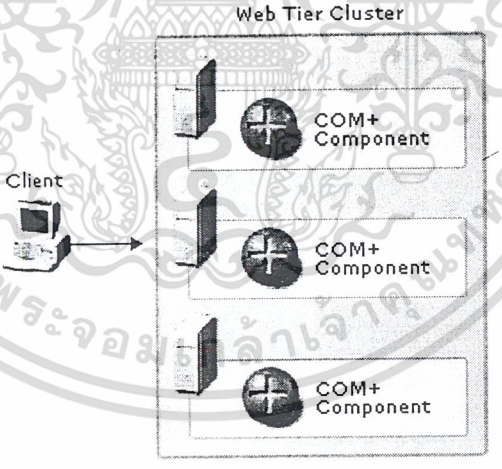
Scenario	*Calls per Second*	*Relative Speed*
COM+ Server Application, run over a 10BaseT network	*625*	*1.0x*
COM+ Server Application, Out Proc, same box	*1923*	*3.08x*
COM+ Library Application, In Proc, same box	*3333*	*5.33x*

ตารางที่ 9-1 ตารางวัดเวลาที่ใช้ในการเรียกคอมโพเน็นต์

ตารางแสดงข้อมูลจำนวนการ เรียกต่อวินาที ใช้ single thread คอมโพเน็นต์ เขียนด้วย Visual Basic 6 ที่คืนค่าเป็น สตริง "Hello, world"

ประสิทธิภาพจะลดลงถ้าฟรอนต์เอนด์เดี่ยว คลัสเตอร์ถูกแยกออกเป็น 2 คลัสเตอร์ โดยไม่ได้มีการเพิ่มจำนวนฮาร์ดแวร์เห็นได้ชัดว่าระบบที่ต้องการประสิทธิภาพสูง ไม่เหมาะกับ CLB ซึ่งผู้ออกแบบเว็บไซต์ต้องคำนึงถึงข้อจำกัดนี้เสมอ

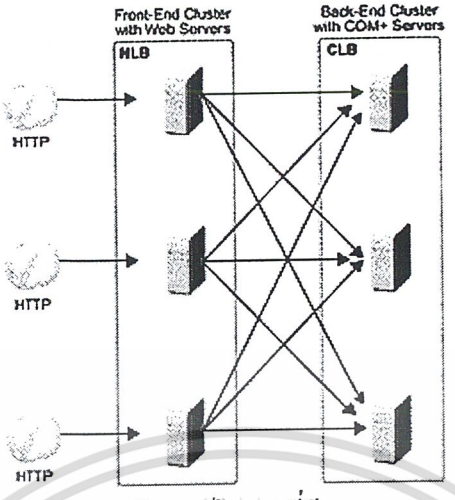
CLB ไม่สามารถแก้ปัญหาได้ดีถ้าระบบต้องการทรูพุทที่สูง ในกรณีนี้จะเป็นการดีกว่าถ้าจะติดตั้ง COM+ คอมโพเน็นต์ ลงบนเว็บเทียร์คลัสเตอร์เลย ทำให้ CLB หายไปแต่ยังสามารถโหลดบาลานซ์ได้โดยใช้ NLB



รูปที่ 9-6 COM+ คอมโพเน็นต์บนเว็บเทียร์

- ความซับซ้อนในการดูแล ซึ่งการจัดการ 2 คลัสเตอร์ ยุ่งยากกว่าคลัสเตอร์เดียว
- เป็นการยากที่จะใช้ฮาร์ดแวร์ได้อย่างเต็มประสิทธิภาพ เพราะมีโอกาสที่เทียร์ใด เทียร์หนึ่งจะมีความรับผิดชอบที่ต่ำกว่าเป็นสาเหตุให้เกิดคอขวด ซึ่งทำให้ต้องเพิ่มการเฝ้าดูการใช้ฮาร์ดแวร์
- การบำรุงรักษาเกิดผลกระทบ เมื่อมีการเพิ่มสมาชิก COM+ แอพพลิเคชันคลัสเตอร์ ทำให้ฟรอนต์เอ็นด์ ต้องอัพเดตสมาชิกคอมโพเน็นต์

9.3.2.2 Two-tier with full load balancing



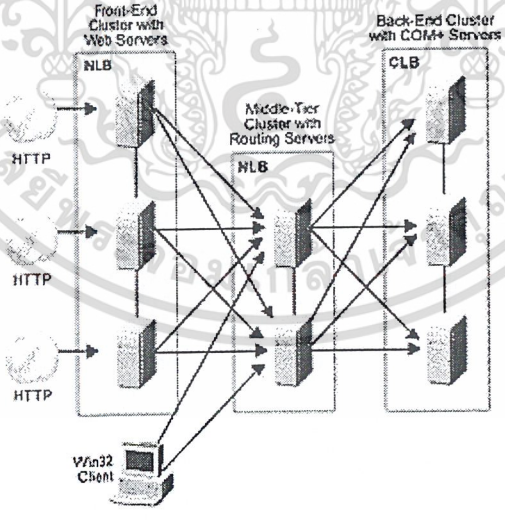
รูปที่ 9-7 Two-tier คลัสเตอร์โมเดล ที่มี NLB และ CLB คลัสเตอร์

แอพพลิเคชันเซ็นเตอร์เว็บคลัสเตอร์ใช้ NLB บนฟรอนต์เอนด์และ CLB ทำคอมโพเน็นต์ routing list ไปยังคอมโพเน็นต์เซิร์ฟเวอร์ที่แบ็กเอนด์คลัสเตอร์

Out-of-process COM+ object จะทำงานที่สมาชิกของ CLB คลัสเตอร์

คอมโพเน็นต์ routing list และ response time table อยู่บนสมาชิกของฟรอนต์เอนด์คลัสเตอร์

9.3.2.3 Three-tier with full load balancing

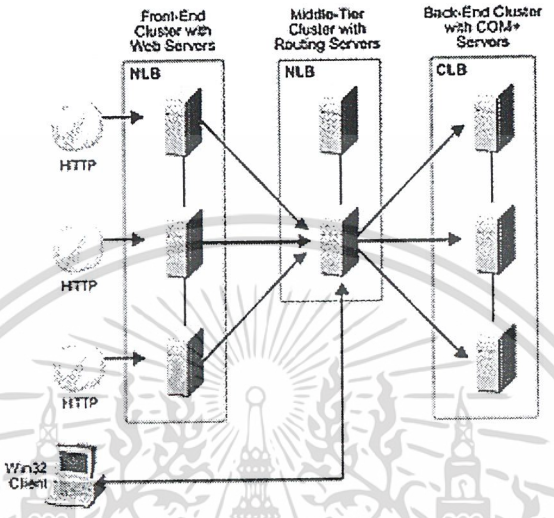


รูปที่ 9-8 Three-tier คลัสเตอร์ ที่ทำ load balancing ทั้ง 3 เทียร์

โมเดลนี้เป็นโมเดลเสมือนกับโมเดลแบบเฟลโอเวอร์ในหัวข้อถัดไป

ความแตกต่างที่เห็นได้คือ โมเดลนี้มีการโหลดบาลานซิงแบบเต็มในมิดเดิลเทียร์โดย HTTP ทราฟฟิกรับผิดชอบโดยฟรอนต์เอนด์ และคลัสเตอร์ตรงชั้นกลางรับผิดชอบรีเควสต์ของฟรอนต์เอนด์และ win 32 แอพพลิเคชัน

9.3.2.4 Three-tier with fail over



รูปที่ 9-9 Three-tier คลัสเตอร์ ที่ middle tier ใช้เฟลโอเวอร์

ฟรอนต์เอนด์คลัสเตอร์ของเว็บเซิร์ฟเวอร์ ส่งคอมโพเน็นต์รีเควสต์ไปยังมิดเดิลเทียร์ที่ทำโหลดบาลานซ์ที่ติดตั้งเป็น COM+ routing คลัสเตอร์

<u>หมายเหตุ</u> โมเดล นี้เลียนแบบเฟลโอเวอร์แบบปกติโดยสถานะเฟลโอเวอร์ ต้องมีความสามารถอย่างน้อยเท่ากับสถานะปกติ ถ้าไม่มีความต้องการในเรื่องนี้ก็ควรใช้โมเดลทรีเทียร์แบบที่แล้วดีกว่า เพราะสามารถใช้ประโยชน์จากเร้าต์ติ้งเทียร์ได้เต็มที่

มิดเดิลเทียร์มีสมาชิก 2 ตัว แต่ตัวคอนโทรลเลอร์ (controller) เท่านั้นที่ทำโหลดบาลานซ์ โดยสมาชิกตัวที่ 2 ทำหน้าที่เป็นตัวสำรองที่สามารถเข้าทำงานแทนเมื่อคอนโทรลเลอร์เสีย

โมเดลนี้รองรับการใช้งานมิดเดิลเทียร์โดย win32 ไคลเอ็นต์ด้วย

9.4 คลัสเตอร์เซอร์วิส (Cluster Service)

คลัสเตอร์เซอร์วิสใน Windows 2000 เป็นชื่อใหม่ของ Microsoft Cluster Server (MSCS) ใน Windows NT Server 4.0, Enterprise Edition ซึ่งคลัสเตอร์เซอร์วิสเป็นหนึ่งในสองเทคโนโลยีของ Microsoft Windows Clustering technologies ในผลิตภัณฑ์ตระกูล Microsoft Windows 2000 Server (อีกเซอร์วิสหนึ่งคือ Network Load Balancing) โดยคลัสเตอร์เซอร์วิสจะจัดเตรียมคุณสมบัติ Failover สำหรับ back-end application (เช่น database, file server, enterprise resource planning(ERP) and messaging system) และเซอร์วิสอื่นที่ต้องการความคงตัวและความถูกต้องของข้อมูลสูง (high availability and data integrity)

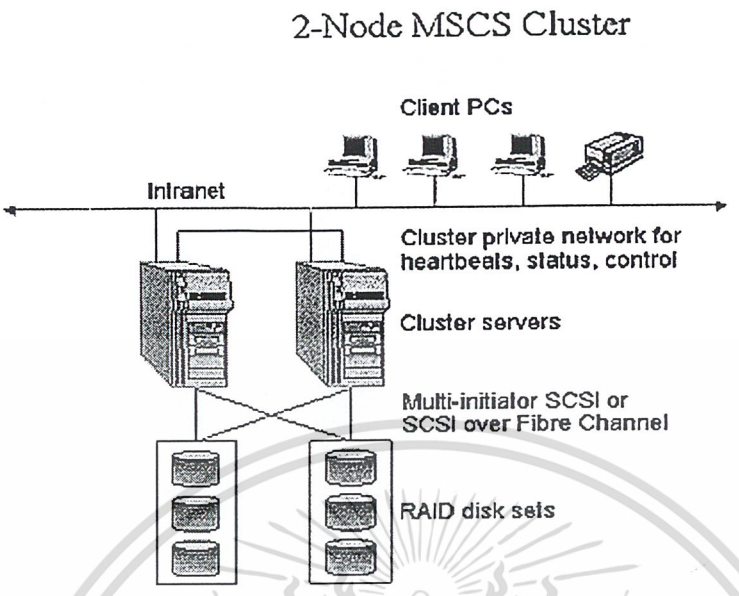
คลัสเตอร์เซอร์วิสช่วยให้เราสามารถเชื่อมเซิร์ฟเวอร์หลายๆ ตัวเข้าเป็น Server Cluster ซึ่งทำให้ความสามารถทั้งทางด้านความคงตัว (available) และการจัดการข้อมูลและโปรแกรมที่รันในคลัสเตอร์เพิ่มมากขึ้น นอกจากนี้คลัสเตอร์เซอร์วิสยังเพิ่มข้อได้เปรียบในเทคโนโลยีด้านคลัสเตอร์ริ่งอีก 3 ข้อ คือ

- **Improved availability** โดยการทำให้เซอร์วิสและแอพพลิเคชันใน server cluster มีความต่อเนื่องในการทำงาน ในระหว่างที่ฮาร์ดแวร์หรือซอฟต์แวร์คอมโพเนนต์ล้มเหลวในการทำงาน หรือในระหว่างที่มีการบำรุงรักษาเครื่องเซิร์ฟเวอร์
- **Increased scalability** ช่วยทำให้สามารถเพิ่มจำนวนโพรเซสเซอร์ และจำนวนเมมโมรี่ขึ้นได้ โดยใน Windows 2000 Advanced Server เพิ่มโพรเซสเซอร์ได้สูงสุด 8 โพรเซสเซอร์ และเมมโมรี่ได้สูงสุด 8 กิกะไบต์ ส่วนใน Windows 2000 Data Center เพิ่มโพรเซสเซอร์ได้สูงสุด 32 โพรเซสเซอร์ และเมมโมรี่ได้สูงสุด 64 กิกะไบต์)
- **Improved manageability** ช่วยให้ผู้ดูแลระบบสามารถใช้เครื่องคอมพิวเตอร์เพียงเครื่องเดียวในการจัดการกับอุปกรณ์ (device) และทรัพยากร (resource) ภายในคลัสเตอร์ทั้งหมด

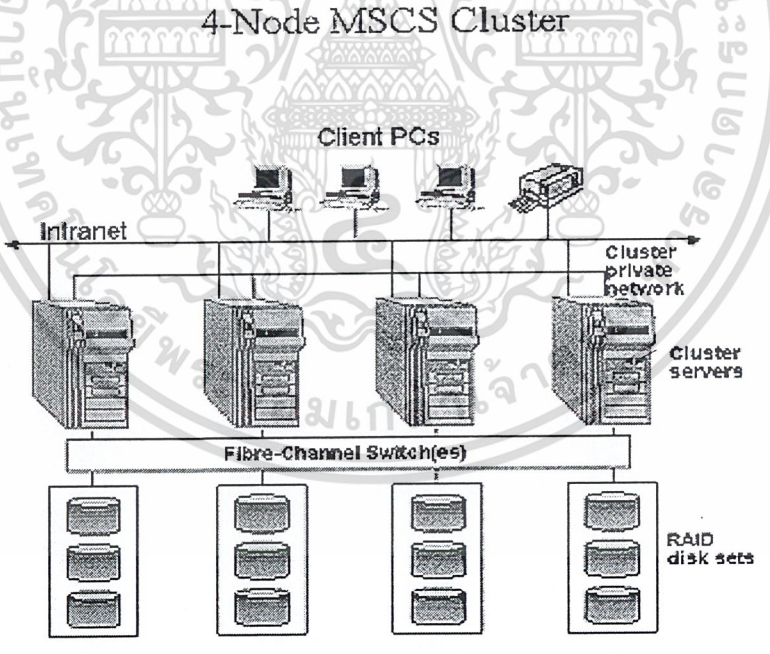
คลัสเตอร์ เป็นกลุ่มของเครื่องคอมพิวเตอร์ที่ทำงานร่วมกันเสมือนเป็นเครื่องหนึ่ง (โดยเครื่องคอมพิวเตอร์แต่ละเครื่องจะถูกเรียกเป็นโหนด (node)) ทำให้มีความสามารถในการกระจายโหลด และถ้ามีทรัพยากรใดในคลัสเตอร์ไม่สามารถทำงานได้ (อาจเป็นได้ทั้งฮาร์ดแวร์และซอฟต์แวร์) คลัสเตอร์ก็ยังสามารถดำเนินงานต่อไปได้โดยใช้ทรัพยากรอื่นที่อยู่ในคลัสเตอร์นั้นแทน

คลัสเตอร์เซอร์วิสตั้งอยู่บนพื้นฐานของ share-nothing model ของสถาปัตยกรรมคลัสเตอร์ ทำให้แต่ละเซิร์ฟเวอร์มีอุปกรณ์เป็นของตัวเอง และสามารถจัดการอุปกรณ์เหล่านั้นได้ ส่วนอุปกรณ์ที่ใช้ร่วมกันนั้น (Device common) เช่น quorum resource, common disk array และ connection media จะมีเจ้าของและถูกควบคุมเพียงเซิร์ฟเวอร์เดียวในช่วงเวลาหนึ่งๆ

External storage devices ที่ใช้ร่วมกันในคลัสเตอร์จำเป็นต้องใช้ SCSI devices และรองรับมาตรฐานการเชื่อมต่อแบบ PCI-based SCSI หรือใช้ SCSI over fiber channel SCSI bus ที่มีหลาย initiators แทนก็ได้

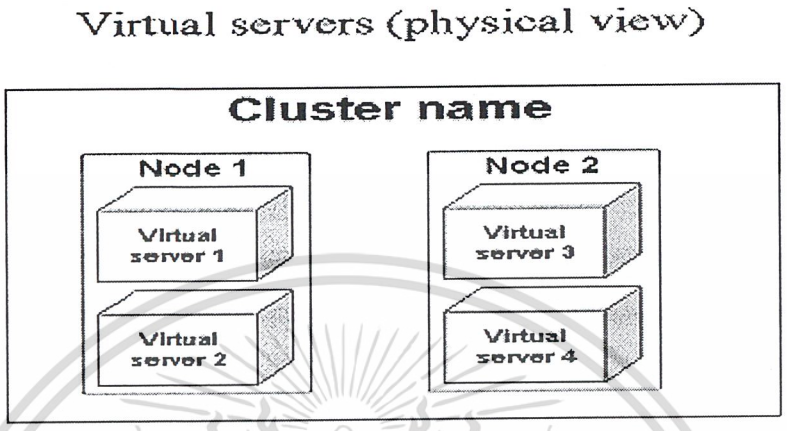


รูปที่ 9-10 Two-node server cluster running Windows 2000 Advanced Server or Windows NT Server 4.0, Enterprise Edition



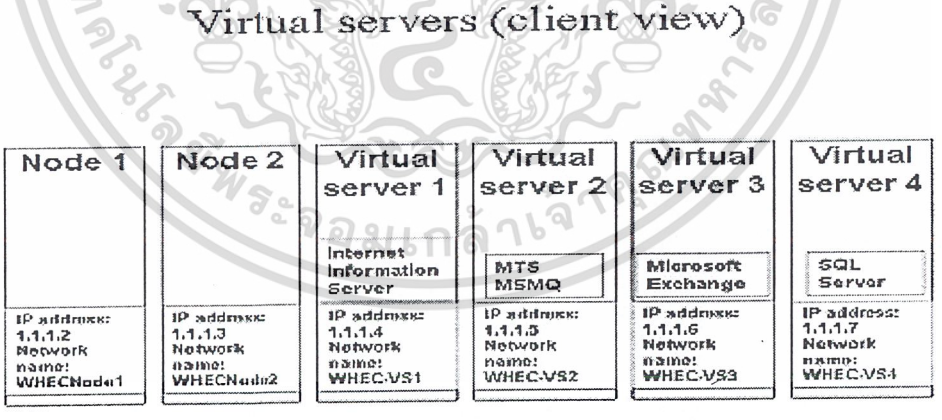
รูปที่ 9-11 Four-node server cluster running Windows 2000 Data Center Server

แอพพลิเคชันและเซอร์วิสที่รันบนเซิร์ฟเวอร์คลัสเตอร์ สามารถมองเสมือนเป็นเซิร์ฟเวอร์หนึ่งๆ (Virtual Servers) โดยที่ผู้ใช้หรือไคลเอนต์จะไม่รู้เลยว่าโฮสต์ที่กำลังติดต่อด้วยนั้นเป็นเวอร์ชวลเซิร์ฟเวอร์ (แอพพลิเคชันหรือเซอร์วิสที่ไม่ได้ถูกใช้โดยไคลเอนต์ สามารถรันบนคลัสเตอร์โหนดได้ โดยไม่เกี่ยวข้องกับ virtual server เลย)



รูปที่ 9-12 Physical view of virtual servers under Cluster service

คลัสเตอร์เซอร์วิสจะจัดการกับเวอร์ชวลเซิร์ฟเวอร์เสมือนเป็นกลุ่มของทรัพยากร (resource group)โดยแต่ละกลุ่มจะมี IP Address และ network name ที่ชี้ไปยัง IP Address นั้นเป็นของตัวเอง เมื่อไคลเอนต์ต้องการติดต่อกับ virtual server ใดก็สามารถติดต่อได้โดยรู้แค่ IP Address ของ virtual server นั้นไม่จำเป็นต้องรู้ว่า virtual server นั้นอยู่ที่โหนดใด



รูปที่ 9-13 Client view of Cluster service virtual servers

ในชั่วขณะที่แอพพลิเคชันหรือเซิร์ฟเวอร์ล้มเหลวในการทำงาน ไคลเอนต์จะตรวจพบความล้มเหลวของแอพพลิเคชันใน session ของมัน และจะพยายามเชื่อมต่อใหม่ในลักษณะเดิมก่อนที่จะล้มเหลว ซึ่งการเชื่อมต่อครั้งหลังนี้จะประสบความสำเร็จ เนื่องจากคลัสเตอร์เซอร์วิสจะเปลี่ยนหลายเลข IP

Address ที่ชี้ไปยัง virtual server ที่ล้มเหลวไปยังโหนดในคลัสเตอร์ที่ยังสามารถทำงานได้ ซึ่งไคลเอนต์จะไม่รู้เลยว่าในขณะนั้นแอพพลิเคชันได้ย้ายไปรันยังโหนดอื่นแล้ว

9.4.1 การตรวจสอบความล้มเหลวในการทำงาน

เมื่อโหนดหรือแอพพลิเคชันในคลัสเตอร์ล้มเหลวในการทำงาน คลัสเตอร์เซอร์วิสจะทำการรันแอพพลิเคชันที่ล้มเหลวขึ้นมาอีกครั้งหรืออาจจะทำการกระจายงานจากระบบที่ล้มเหลวในการทำงานไปยังโหนดอื่นในคลัสเตอร์ที่ยังสามารถทำงานได้ นอกจากนี้คลัสเตอร์เซอร์วิสยังสามารถตรวจสอบและป้องกันความล้มเหลวในการทำงานรวมไปถึง bi-directional failover, application failover, parallel recovery, and automatic failback อีกด้วย

คลัสเตอร์เซอร์วิสสามารถตรวจสอบความล้มเหลวในการทำงานของแต่ละรีซอร์สหรือโหนดอย่างอิสระและยังสามารถเคลื่อนย้ายหรือเริ่มต้นจัดการกับแอพพลิเคชัน ข้อมูล และไฟล์รีซอร์สที่อยู่บนเซิร์ฟเวอร์ที่ยังใช้งานได้อย่างอิสระ

คลัสเตอร์เซอร์วิสมีกลไก 2 แบบในการตรวจสอบความล้มเหลวของทรัพยากร คือ

- Heartbeat ใช้ในการตรวจสอบว่า โหนดใดล้มเหลวในการทำงาน
- Resource Monitor and resource DLLs ใช้ในการตรวจสอบว่า รีซอร์สใดล้มเหลวในการทำงาน

9.4.1.1 การตรวจสอบว่า โหนดใดล้มเหลวในการทำงาน

ในช่วงคาบเวลาหนึ่งๆ แต่ละโหนดจะมีการแลกเปลี่ยน datagram messages (เรียกว่า heartbeat) กับโหนดอื่นๆ ที่อยู่ในคลัสเตอร์ซึ่งกันและกัน ผ่านทาง private cluster network การแลกเปลี่ยน heartbeat นี้ทำให้สามารถตรวจสอบได้ว่าโหนดหรือแอพพลิเคชันอื่นๆ ยังสามารถใช้งานได้อยู่หรือไม่ ถ้ามีเซิร์ฟเวอร์ใดไม่ตอบสนองในการแลกเปลี่ยน heartbeat เซิร์ฟเวอร์อื่นจะเริ่มทำขบวนการ failover แล้วจัดการกับรีซอร์สและแอพพลิเคชันของเซิร์ฟเวอร์ที่ล้มเหลวในการทำงาน

9.4.1.2 การตรวจสอบว่า resource ใดไม่สามารถใช้งานได้

ทั้ง Failover Manager และ Resource Monitors จะมีหน้าที่ในการตรวจสอบและกู้รีซอร์สที่ล้มเหลวในการทำงานขึ้นมา โดย Resource Monitors จะคอยติดตามสถานะของรีซอร์ส โดยใช้ resource DLLs ทำการสำรวจ (polling) ซึ่งจะมีการทำงาน 2 ขั้นตอน คือ *LooksAlive* query และ*IsAlive* query เมื่อ Resource Monitor ตรวจพบว่ารีซอร์สใดไม่สามารถใช้งานได้ก็จะแจ้งต่อ Failover Manager

Failover Manager จะมีหน้าที่ในการบำรุงรักษาสถานะของรีซอร์ส และยังมีหน้าที่ในการกู้ รีซอร์สที่ได้รับแจ้งจาก Resource Monitors อีกด้วย โดยการทำ automatic failback

บทที่ 10
การออกแบบระบบโรงแรม

การออกแบบระบบโรงแรมประกอบด้วยบริการต่าง ๆ ต่อไปนี้

1. การลงทะเบียน และ บริการสมาชิก
2. การสืบค้นข้อมูลโรงแรม / ห้องพัก
3. การจองห้องพัก
4. การเรียกดูรายการจองห้องพัก
5. การยืนยันการจองห้องพัก
6. การยกเลิกการจองห้องพัก
7. การบริหารราคาห้องพัก

แต่ละส่วนมีรายละเอียดดังนี้

10.1 การลงทะเบียน

เมื่อผู้ใช้ต้องการใช้บริการจองห้องพักผ่านระบบอินเทอร์เน็ต จะต้องลงทะเบียนก่อน โดยระบุข้อมูลต่อไปนี้

- ชื่อ นามสกุล และเพศของผู้ใช้บริการ
- ที่อยู่และเบอร์โทรศัพท์
- อีเมล์แอดเดรส
- รหัสผ่าน

โดยที่ชื่อผู้ใช้บริการนี้เป็นชื่อของบุคคลที่ทำการจอง ไม่จำเป็นต้องเป็นชื่อของคนที่จะเข้าพักก็ได้ และจะต้องระบุอีเมล์แอดเดรสเพื่อใช้ในการล็อกอิน และระบุรหัสผ่านเพื่อใช้ในการยืนยันตัวบุคคลในการเข้าใช้บริการของระบบ

เมื่อผู้ใช้บริการลงทะเบียนเป็นสมาชิกแล้ว ก็สามารถเรียกดูและแก้ไขข้อมูลที่ลงทะเบียนไว้ได้ เมื่อเสร็จสิ้นการใช้บริการ ก็สามารถล็อกเอาต์ เพื่อออกจากระบบได้

10.2 การสืบค้นข้อมูลโรงแรม / ห้องพัก

ส่วนของการสืบค้นข้อมูลนี้ ผู้ใช้บริการอาจจะยังไม่ต้องลงทะเบียน ก็สามารถใช้บริการได้ ผู้ใช้สามารถสืบค้นข้อมูลโรงแรม หรือ ข้อมูลห้องพักที่ว่างอยู่ในช่วงวันที่ต้องการได้ โดยระบุข้อมูล ดังนี้

- รายละเอียดของโรงแรม หรือ ห้องพักที่ต้องการ
- วันที่จะเข้าพัก และออกจากที่พัก
- จำนวนห้องพักที่ต้องการ

ผลลัพธ์จะแสดงรายละเอียดของโรงแรม และข้อมูลห้องพักที่ว่างที่ตรงกับเงื่อนไขที่ผู้ใช้ป้อนเข้าไป

ข้อมูลที่สามารถใช้เป็นเงื่อนไขได้ มีดังนี้

- ชื่อโรงแรม
- ระดับของโรงแรม (1 - 5 ดาว)
- สถานที่ตั้ง เช่น ทวีป , ประเทศ , จังหวัด , อำเภอ
- ราคาห้องพักต่ำสุด หรือ สูงสุดที่ต้องการ
- ระดับของห้องพัก เช่น ห้องพักธรรมดา , ห้องพิเศษ , ห้องสูท , ห้องเดอลุกซ์
- วันเข้าพัก และ วัน ออกจากที่พัก

การระบุวันเข้าพักและวันออก นั้น จะต้องระบุเป็นคืนที่จะพักที่โรงแรม เช่น ต้องการเข้าพักโรงแรมในคืนที่ 12 มิถุนายน 2545 เพียงคืนเดียว ก็ต้องระบุ วันเข้าพักเป็น 12/6/2545 และวันออกเป็น 12/6/2545 เป็นต้น

10.3 การจองห้องพัก

เมื่อผู้ใช้สืบค้นข้อมูลของโรงแรม และห้องพักที่ต้องการได้แล้ว ก็สามารถเลือกรายการที่ต้องการเพื่อทำการจองได้ โดยระบบโรงแรมนี้จะมีตะกร้าไว้บริการ ตะกร้าจะบรรจุรายการห้องพักที่ต้องการ แต่จะยังไม่ทำการจองให้ ผู้ใช้บริการสามารถเลือกรายการห้องพักที่มาจากการสืบค้นมาใส่ในตระกร้าและลบออกจากตะกร้าได้ตามต้องการ และสามารถเรียกดูรากการในตะกร้าได้ตลอดเวลา แม้ว่าจะปิดบราวเซอร์ไปก็ตาม ข้อมูลในตะกร้าก็จะยังคงอยู่ โดยจะมีอายุ 1 วัน นับจากการเรียกดูตะกร้าครั้งสุดท้าย เมื่อผู้ใช้บริการต้องการจองรายการห้องพักในตะกร้า ก็สามารถทำการจองได้ทันที แต่จะไม่ยืนยันว่าทุกรายการห้องพักที่อยู่ในตะกร้านั้นจะจองสำเร็จเสมอทุกครั้งไป เพราะในขณะที่รายการห้องพักที่ต้องการอยู่ในตะกร้า ก็อาจมีผู้ใช้บริการคนอื่นทำการจองไปแล้วก็ได้ ถ้ามีเหตุการณ์เช่นนี้เกิดขึ้น ระบบจะไม่ทำการจองห้องพักที่อยู่ในตะกร้าให้เลยแม้แต่รายการเดียว

รายการห้องพักที่อยู่ในตะกร้านี้จะถูกลบโดยอัตโนมัติ เมื่อผู้ใช้บริการล็อกเอาต์ออกจากระบบ

ในขั้นตอนการจองห้องพักที่อยู่ในตะกร้านี้ ผู้ใช้จะต้องเป็นสมาชิกของระบบเท่านั้น โดยจะต้องระบุอีเมล์แอดเดรส และรหัสผ่าน หากผู้ใช้ยังไม่ได้เป็นสมาชิก ก็จะต้องทำการลงทะเบียนและล็อกอินเข้าสู่ระบบก่อน

การจองห้องพักนี้เป็นการจองล่วงหน้า ผู้ใช้จะต้องทำการยืนยันการจอง และชำระเงินก่อนวันเข้าพัก 7 วัน (ระยะเวลานี้อาจปรับเปลี่ยนได้ตามนโยบายของโรงแรม) เมื่อทำการจองเสร็จสมบูรณ์แล้ว ระบบจะแจ้งหมายเลขห้องพัก และกำหนดวันที่จะต้องมายืนยันและชำระเงิน ให้ผู้ใช้บริการทราบ

ในกรณีที่ผู้จองไม่มายืนยันการจอง และชำระเงินภายในระยะเวลาที่กำหนด ระบบจะยกเลิกการจองให้โดยอัตโนมัติ และจะส่งอีเมล์ไปแจ้งให้ผู้จองทราบด้วย

เมื่อทำการจองรายการห้องพักที่อยู่ในตะกร้าแล้ว รายการห้องพักที่อยู่ในตะกร้าจะถูกลบโดยอัตโนมัติ

10.4 การเรียกดูรายการจองห้องพัก

ผู้ใช้บริการสามารถล็อกอินเข้าสู่ระบบ แล้วเรียกดูรายการห้องพักที่จองไว้ได้ ระบบจะแสดงรายการจองห้องพักทั้งหมดที่จองไว้ แต่จะไม่แสดงรายการห้องพักที่ผ่านวันเข้าพักมาแล้ว จะแสดงเฉพาะรายการที่ยังไม่ถึงวันเข้าพักเท่านั้น และจะแจ้งให้ทราบด้วยว่า รายการใดบ้างที่ยืนยันและชำระเงินแล้ว ส่วนรายการที่ยังไม่ได้ยืนยันและชำระเงิน ก็จะแจ้งให้ทราบถึงกำหนดสุดท้ายที่สามารถทำการยืนยันและชำระเงินได้

10.5 การยืนยันการจองห้องพัก

ผู้ใช้บริการสามารถล็อกอินเข้าสู่ระบบ แล้วเรียกดูรายการห้องพักที่จองไว้ได้ เมื่อต้องการยืนยันการจอง ระบบจะแสดงจำนวนเงินที่ต้องชำระ ผู้จองจะต้องระบุข้อมูลบัตรเครดิตเพื่อชำระเงิน การยืนยันการจองจะเสร็จสิ้นก็ต่อเมื่อ การชำระเงินเสร็จสมบูรณ์แล้ว

10.6 การยกเลิกการจองห้องพัก

ผู้ใช้สามารถล็อกอินเข้าสู่ระบบ และเลือกรายการจองที่ต้องการยกเลิกได้ แต่ต้องเป็นรายการที่ยังไม่ยืนยันและชำระเงิน

10.7 การบริหารราคาห้องพัก

ในงานส่วนนี้เป็นหน้าที่ของผู้จัดการโรงแรม โดยสามารถเปลี่ยนแปลงราคาและส่วนลดของห้องพักตามระดับห้องพักในแต่ละโรงแรมได้

ส่วนลดของห้องพักจะแบ่งตามลักษณะวันที่เข้าพัก เช่น วันธรรมดา (วันจันทร์ - วันพฤหัสบดี) จะลดราคาระดับหนึ่ง , วันศุกร์ - วันอาทิตย์ ก็จะลดราคาอีกระดับหนึ่ง ซึ่งน้อยกว่าวันธรรมดา ส่วนวันหยุดเทศกาล อาจจะไม่ลดราคาเลย

10.8 ออกแบบระบบงาน

การออกแบบระบบโรงแรมตามความต้องการข้างต้น แสดงได้ดังนี้

- แสดงการไหลของข้อมูลด้วย ดาต้าโฟลว์ไดอะแกรม (Data Flow Diagram)
- แสดงการออกแบบฐานข้อมูลเชิงสัมพันธ์ได้ด้วย อีอาไดอะแกรม (ER Diagram)
- แสดงการออกแบบเชิงออกเจ็กโอเรียนเต็ดด้วยคลาสไดอะแกรม (Class Diagram)

10.8.1 ดาต้าโฟลว์ไดอะแกรม

ดาต้าโฟลว์ไดอะแกรมของระบบโรงแรมชั้นคอนเท็กเลเวล (Context Level) จะประกอบด้วยโพรเซสชื่อ Hotel System ซึ่งเป็นระบบหลัก และ มี เอ็กซ์เทอร์นอล เอ็นทิตี้ (External Entity) อีก3 ตัวคือ

- *Manager* คือผู้จัดการโรงแรม มีหน้าเพิ่ม-ลดโรงแรม หรือห้องพัก, แก้ไขข้อมูลโรงแรม หรือห้องพัก ,การแก้ไขข้อมูลส่วนลด
- *Customer* คือลูกค้าของโรงแรม ที่จะเข้ามาสมัครสมาชิก, ค้นหาห้องพักที่ว่าง, จองห้องพัก, เรียกดูรายการจอง ยืนยัน และ ยกเลิกรายการจอง
- *Bank* คือระบบธนาคารที่จะโอนเงินจากบัญชีของลูกค้าเข้าบัญชีของโรงแรมเมื่อมีการชำระเงิน

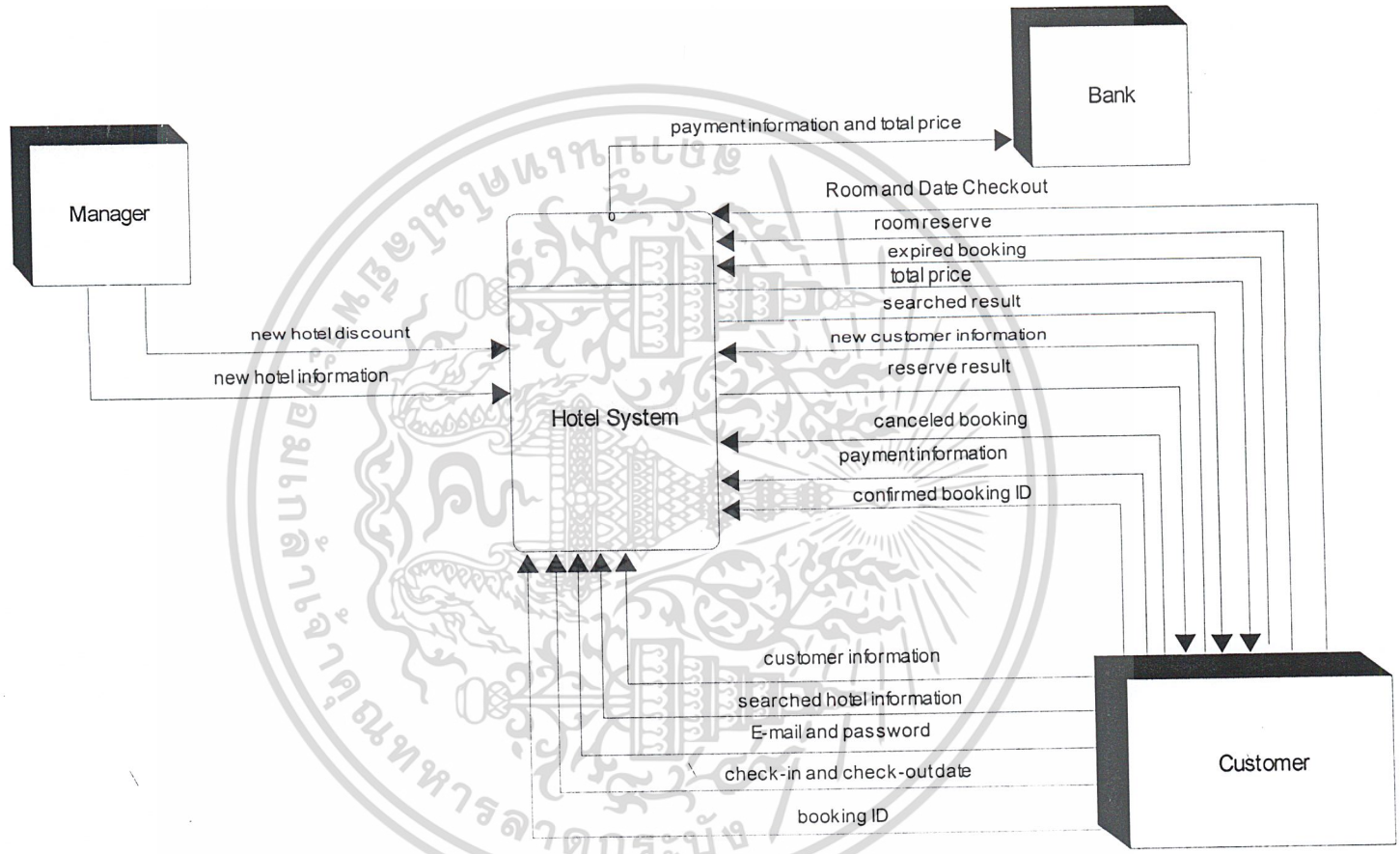
ในชั้นถัดมา ของ ดาต้าโฟลว์ไดอะแกรม ประกอบด้วย 3 โพรเซสคือ

- *Customer Service Sub System* เป็นโพรเซสที่ทำหน้าที่ให้บริการสมาชิกของระบบ เช่นการลงทะเบียน , การแก้ไขข้อมูลสมาชิก และ การล็อกอินเข้าสู่ระบบ
- *Booking Service Sub System* เป็นโพรเซสที่ทำหน้าที่เกี่ยวกับการค้นหาข้อมูลห้องพัก , การจองห้องพัก , การเรียกดูรายการจองห้องพัก , การยืนยัน และ การยกเลิกรายการจองห้องพัก
- *Hotel Information Maintenance* เป็นโพรเซสที่ทำหน้าที่เกี่ยวกันการจัดการข้อมูลโรงแรมเช่น การเพิ่ม-ลดโรงแรม หรือ ห้องพัก, การแก้ไขข้อมูลโรงแรม การแก้ไขส่วนลด

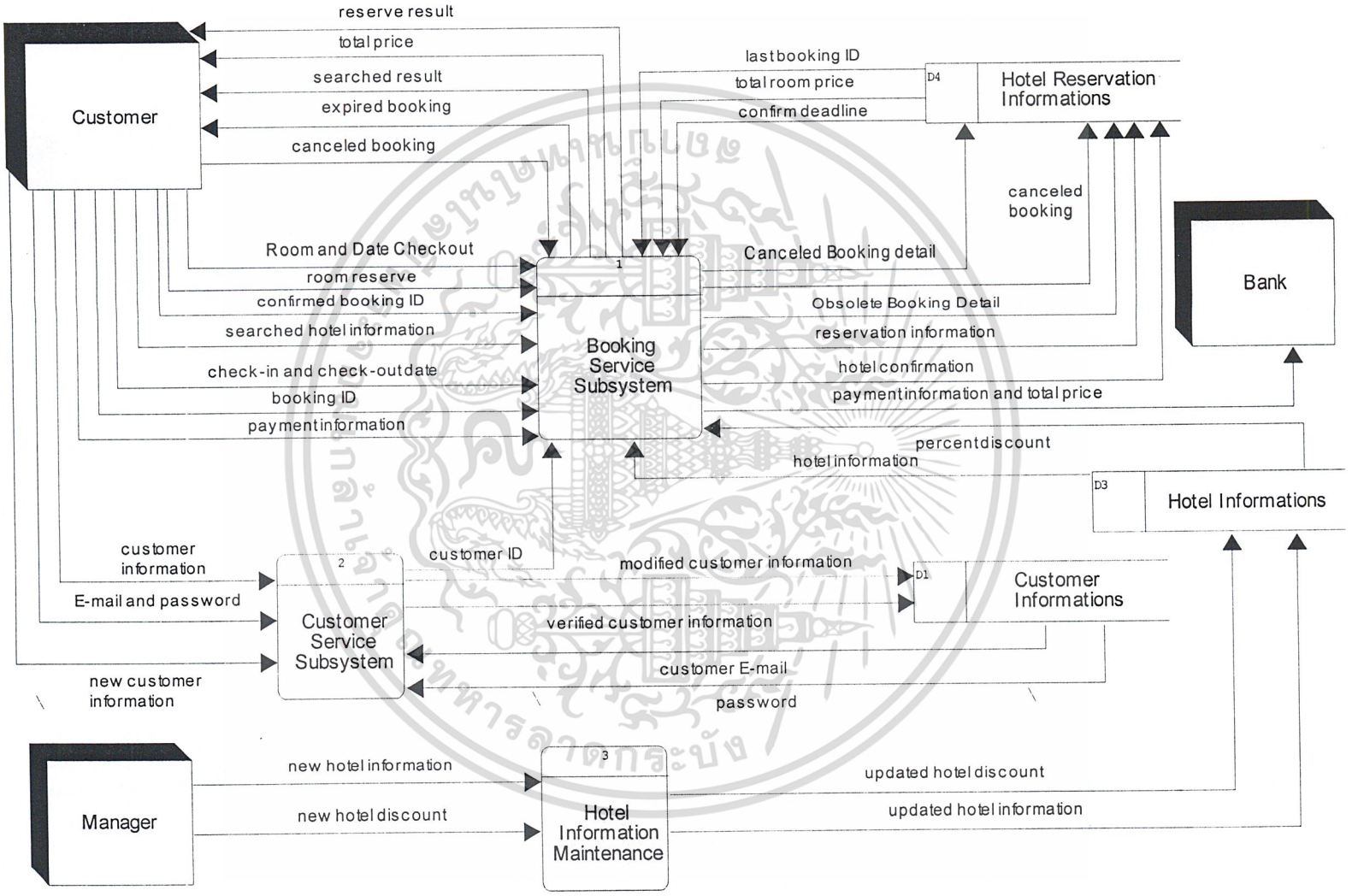
นอกจากนี้ยังประกอบด้วยดาต้าสโตร์(Data Store) อีก 3 ตัว คือ

- *Customer Information* เก็บข้อมูลทั้งหมดของสมาชิกที่เข้ามาลงทะเบียน
- *Hotel Information* เก็บข้อมูลทั้งหมดโรงแรม เช่น ข้อมูลรายละเอียดของโรมแรม , ข้อมูลห้องพัก , ข้อมูลชนิดของห้องพัก , ข้อมูลส่วนลดห้องพัก เป็นต้น
- *Hotel Reservation Information* เก็บข้อมูลการจองโรงแรมทั้งหมดของสมาชิก เช่น วันที่เข้าพัก ราคาจอง เลขที่ห้องพักที่ได้ เป็นต้น

ดาต้าโฟลว์ไดอะแกรมของระบบโรงแรม แสดงได้ดังรูป

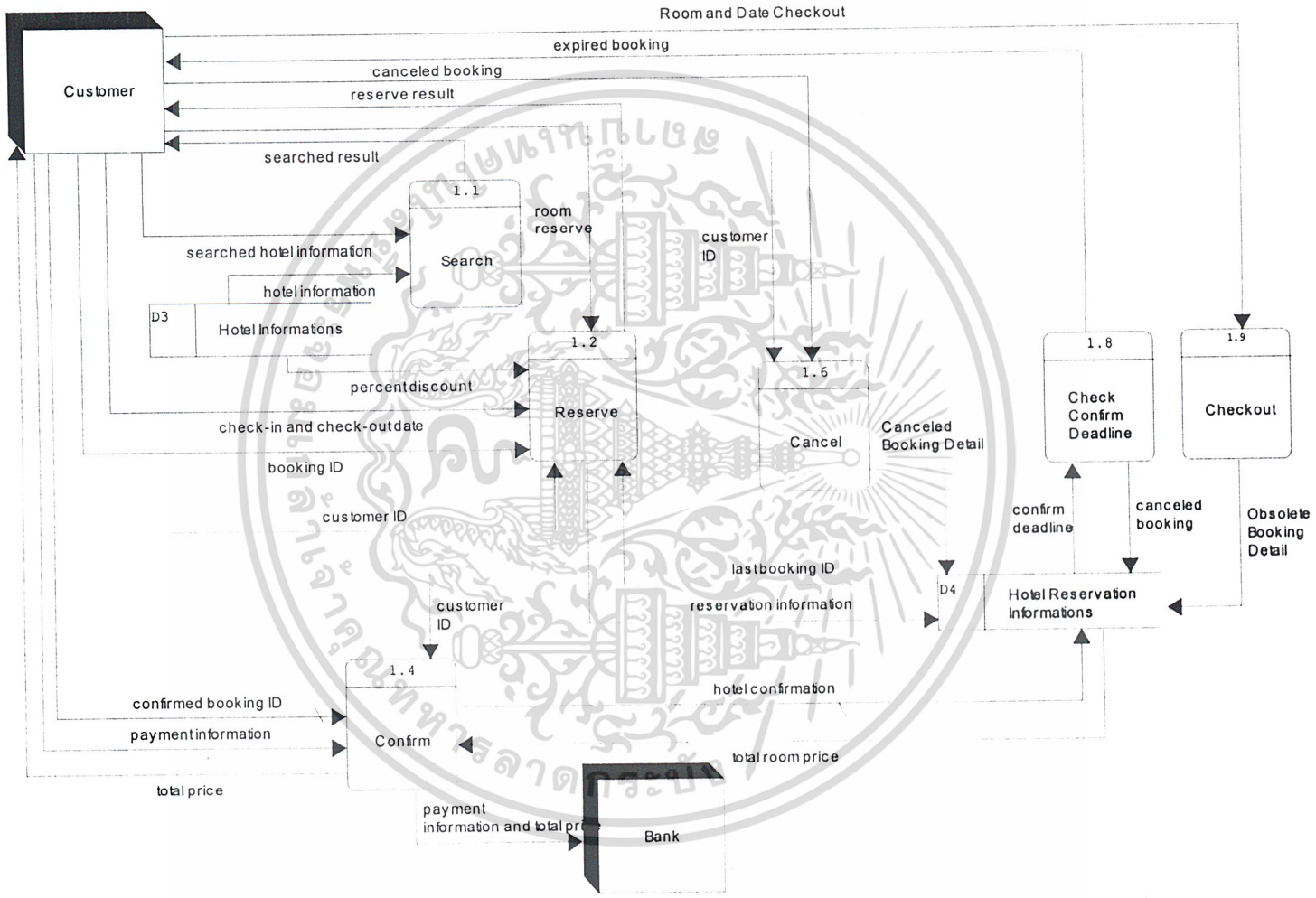


รูปที่ 10-1 แสดง Data Flow Diagram – Context Level ของระบบโรงแรม



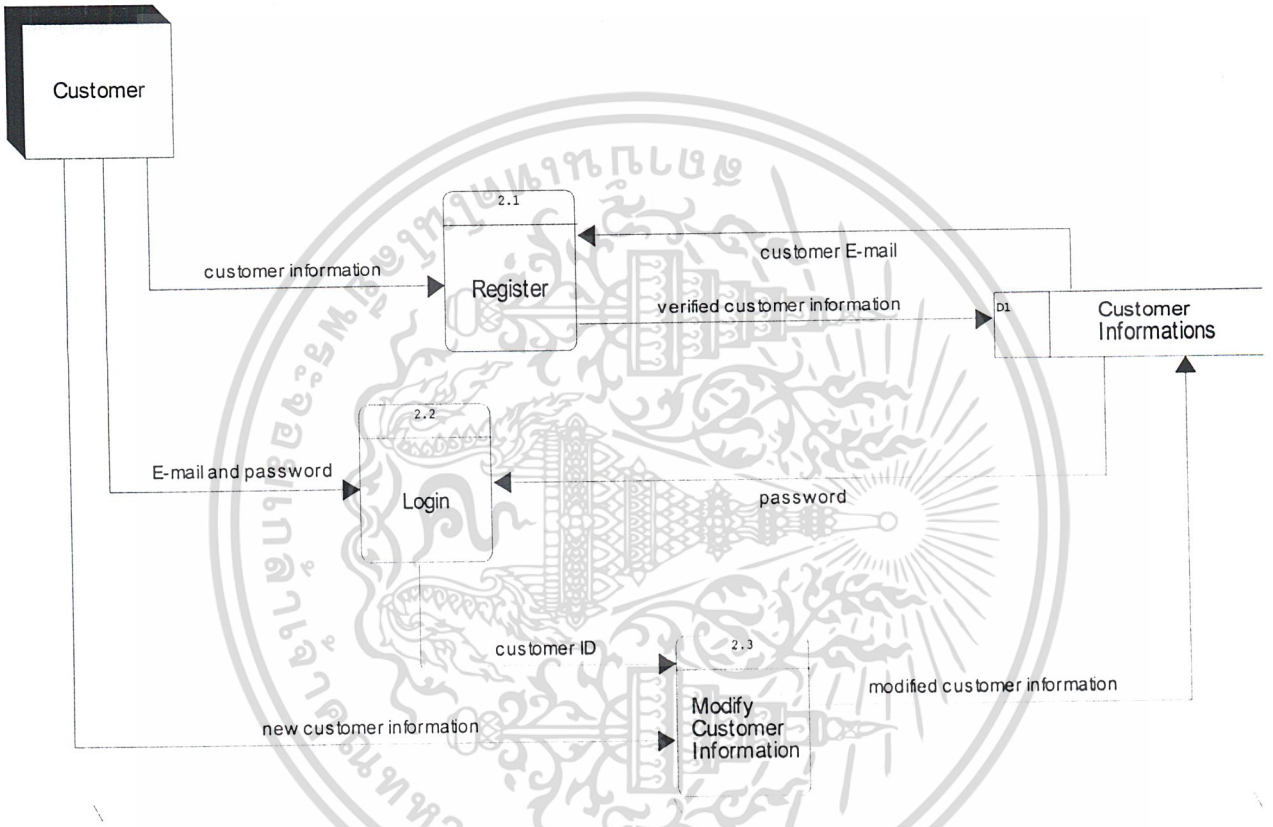
รูปที่ 10-2 แสดง Data Flow Diagram – Level1 ของระบบโรงแรม

124



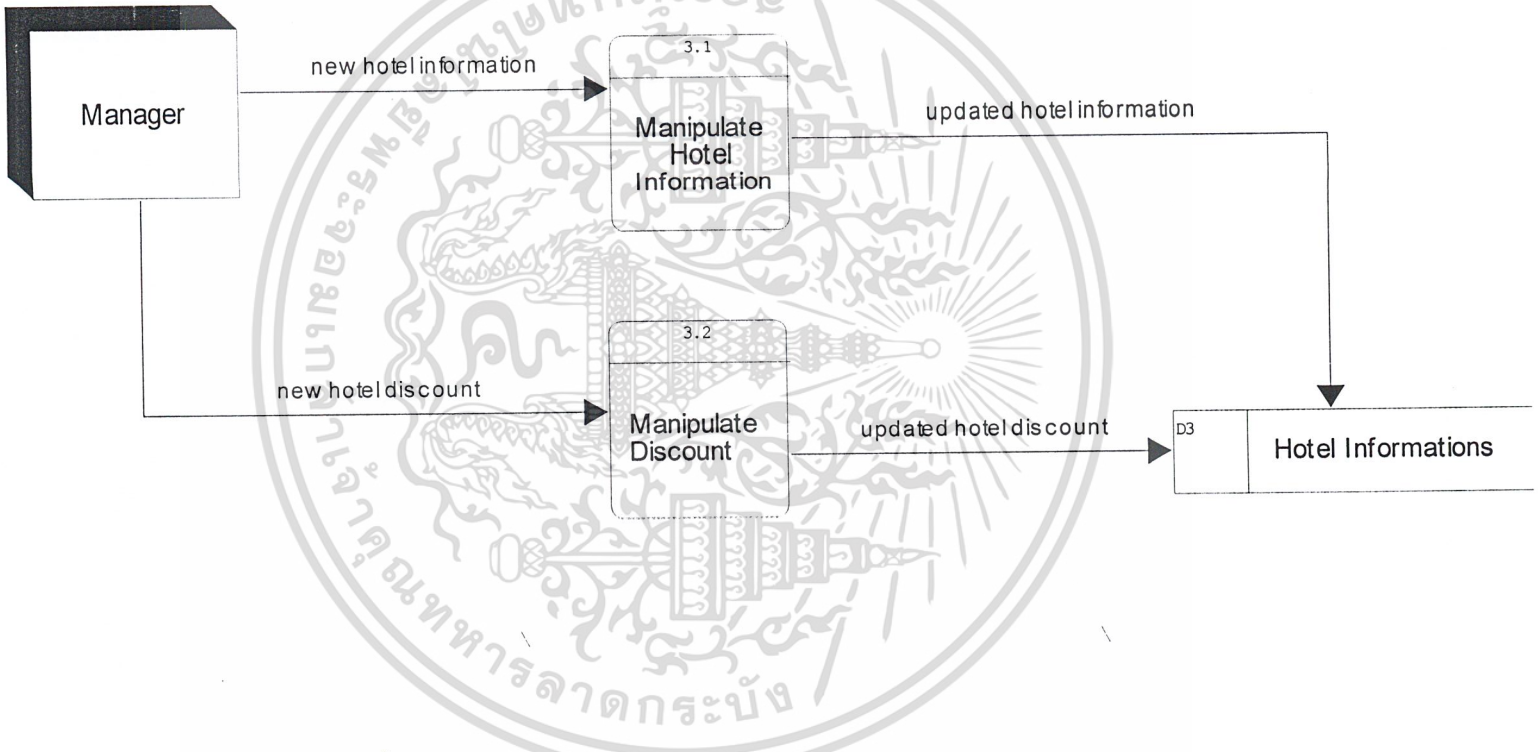
รูปที่ 10-3 แสดง Data Flow Diagram – Level2-1 ของระบบโรงแรม

125



รูปที่ 10-4 แสดง Data Flow Diagram – Level2-2 ของระบบโรงแรม

126



รูปที่ 10-5 แสดง Data Flow Diagram – Level 2-3 ของระบบโรงแรม

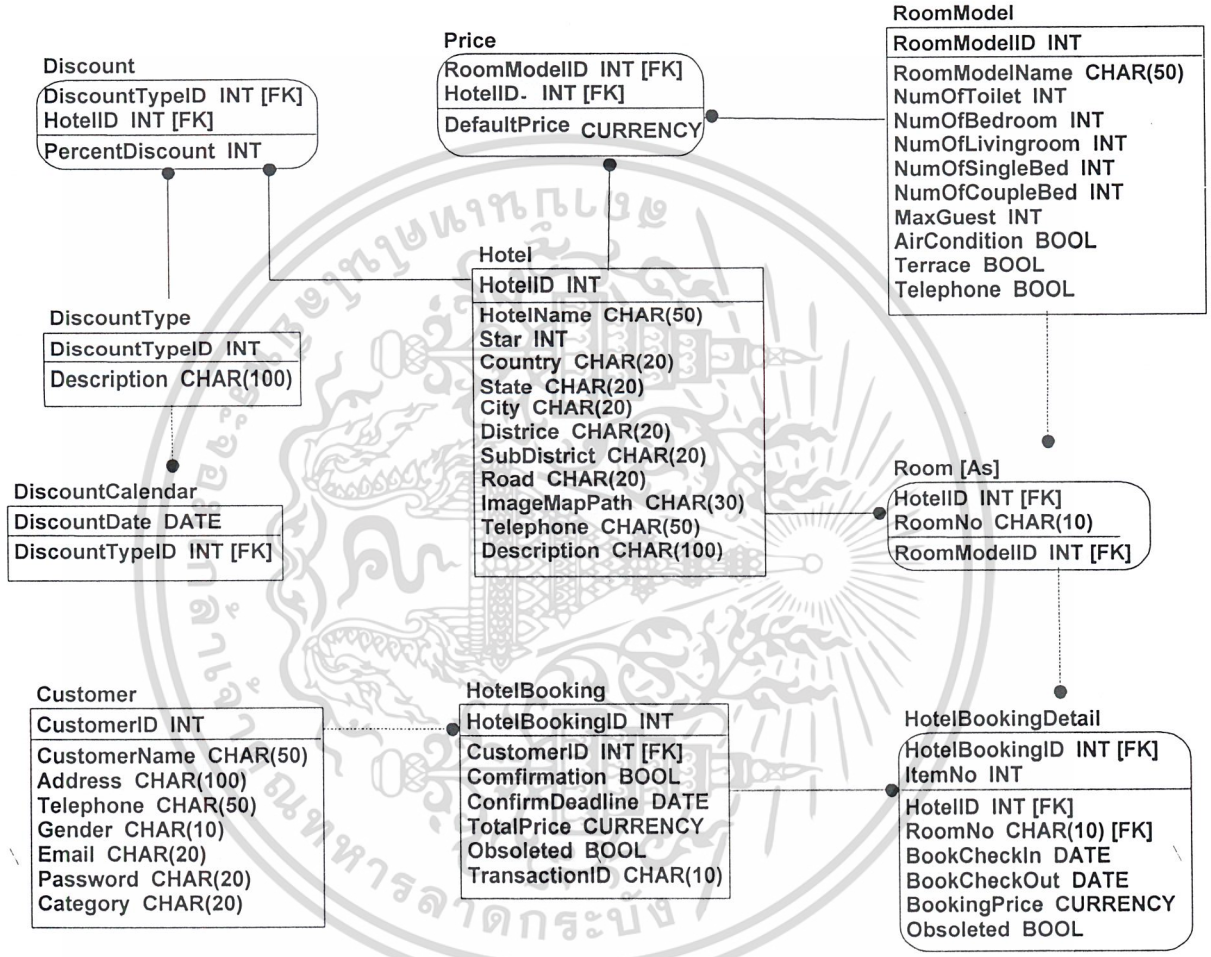
127

10.8.2 อีอาไดอะแกรม

อีอาไดอะแกรมของระบบโรงแรม ประกอบด้วย 10 เอ็นทิตี้ ไทพ์(Entity Type)ซึ่ง แบ่งได้ 4 ส่วนดังนี้

- ส่วนข้อมูลสมาชิก ประกอบด้วย 1 เอ็นทิตี้ ไทพ์ คือ Customer ซึ่งเก็บข้อมูลสมาชิกของระบบ
- ส่วนข้อมูลโรงแรมและห้องพัก ประกอบด้วย 3 เอ็นทิตี้ ไทพ์ คือ Hotel, Room, RoomModel ซึ่งเก็บข้อมูลรายละเอียดของโรงแรม เช่น ชื่อโรงแรม, ที่ตั้ง , เบอร์โทรศัพท์ ข้อมูลชนิดห้องพัก เช่น จำนวนห้องน้ำ ,จำนวนเตียงคู่,จำนวนเตียงเดี่ยว, ข้อมูลห้องพักของแต่ละโรงแรมว่าห้องพักใดเป็นชนิดอะไร
- ส่วนข้อมูลราคาและส่วนลด ประกอบด้วย 4 เอ็นทิตี้ ไทพ์ คือ Price, Discount, DiscountType, DiscountCalendar ซึ่งเก็บข้อมูลรายละเอียดของส่วนลด เอ็นทิตี้ ไทพ์ DiscountType จะเก็บข้อมูลประเภทของส่วนลดต่างๆ เช่น วันธรรมดา(วันอาทิตย์ถึงวันพฤหัสบดี) จะเป็นประเภทที่ 1 , วันสุดสัปดาห์ (วันศุกร์และวันเสาร์) จะเป็นประเภทที่ 2 และวันหยุดเทศกาล เช่นวันปีใหม่ วันสงกรานต์ จะเป็นประเภทที่ 3 วันหยุดเทศกาลที่ตรงกับวันสุดสัปดาห์ก็จะเป็นประเภทที่ 3 ด้วย เป็นต้น เอ็นทิตี้ ไทพ์ Discount จะเก็บข้อมูลส่วนลดของโรงแรมต่างๆ ว่าโรงแรมนั้นจะให้ส่วนลดกับประเภทของวันต่างๆเท่าใด เช่น โรงแรม ก. จะให้ส่วนลดกับวันธรรมดา 30% , วันสุดสัปดาห์ 20% และวันหยุดเทศกาล 5% ส่วนโรงแรม ข. จะให้ส่วนลดกับวันธรรมดา 25% วันสุดสัปดาห์ 15% และวันหยุดเทศกาลไม่ลดราคาเลย เป็นต้น เอ็นทิตี้ ไทพ์ Price จะเก็บข้อมูลราคาของชนิดห้องพักของแต่ละโรงแรม เอ็นทิตี้ ไทพ์ DiscountCalendar จะเก็บข้อมูลว่าวันต่างๆใน 1 ปี เป็นวันประเภทใด
- ส่วนข้อมูลการจองห้องพัก ประกอบด้วย 2 เอ็นทิตี้ ไทพ์ คือ HotelBooking, HotelBookingDetail ซึ่งเก็บข้อมูลการจองห้องพัก เช่น โรงแรมและหมายเลขห้องพักที่จอง , วันที่เข้าพักและออก , ราคาจอง , กำหนดวันสุดท้ายที่ให้ยืนยันได้ เป็นต้น

อีอาไดอะแกรม ของระบบโรงแรมแสดงได้ดังรูป



รูปที่ 10-6 แสดง ER Diagram ของระบบโรงแรม

10.8.3 คลาสไดอะแกรม

คลาสไดอะแกรมของระบบโรงแรมประกอบด้วย 3 คลาสคือ CustomerService, BookingService, HotelService และ BankingService (BankingService จะไม่กล่าวถึงเพราะอยู่นอกระบบงาน)

สรุป คลาส, อินเตอร์เฟส และ เมดทอด ของ คลาสไดอะแกรม ได้ดังนี้

Class	Interface	Method
CustomerService	ICustomer	Login()
		Register()
		GetCustomer()
		EditCustomer()
BookingService	IBookmanager	Reserve()
		CancelBooking()
		CancelBookingDetail()
		CancelBookingOverComfirmDeadline()
		SetObsoleteBooking()
	IBookViewer	GetHotelBooking()
		GetHotelBookingDetail()
		GetHotelBookingDetailInBasketForm()
	IBasket	CreateBasket()
		AddBasketItem()
		RemoveBasketItem()
		RemoveAllBasketItem()
		ViewBasket()
HotelService	ISearch	Search()
	IHotelManager	AddHotel()
		AddRoom()
		AddRoomModel()
		SetPrice()
		SetDiscount()
		AddDiscountType()
		AddDiscountCalendar()
		DeleteHotel()
		DeleteRoom()

Class	Interface	Method
HotelService	IhotelManager	DeleteRoomModel()
		DeletePrice()
		DeleteDiscount()
		DeleteDiscountType()
		DeleteDiscountCalendar()
		GetRoomModelFromHotelID()
		GetPrice()
		GetDiscountCalendar()
		GetAllDiscountType()
		GetDiscount()
		CountRoom()

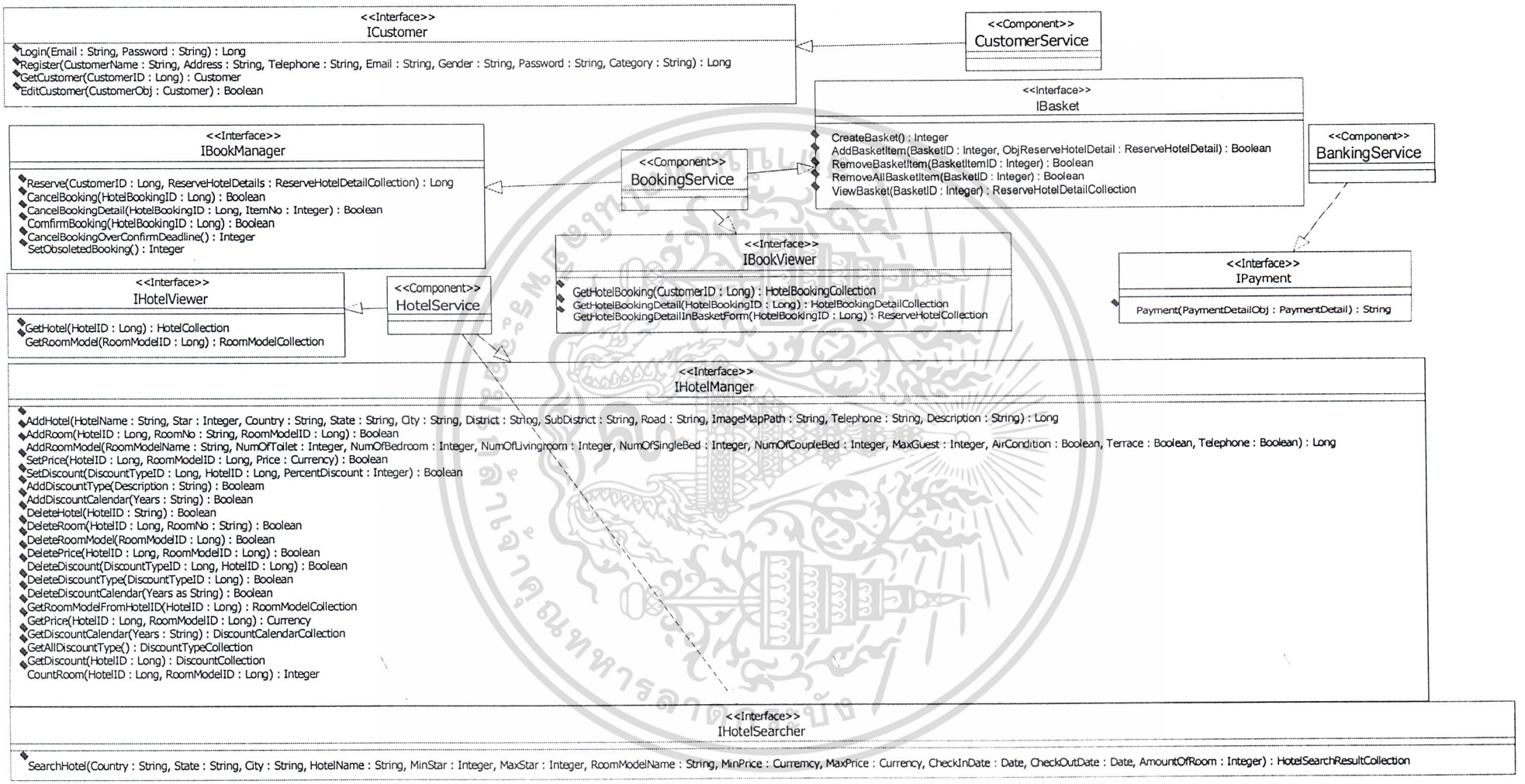
ตารางที่ 10-1 แสดงคลาส อินเตอร์เฟส และเมธอดของระบบโรงแรม

รายละเอียดของเมธอดที่สำคัญ

Register() จะเป็นการลงทะเบียนเป็นสมาชิกของระบบ โดยอีเมล์ที่ระบุจะต้องไม่ซ้ำกับที่มีอยู่แล้ว

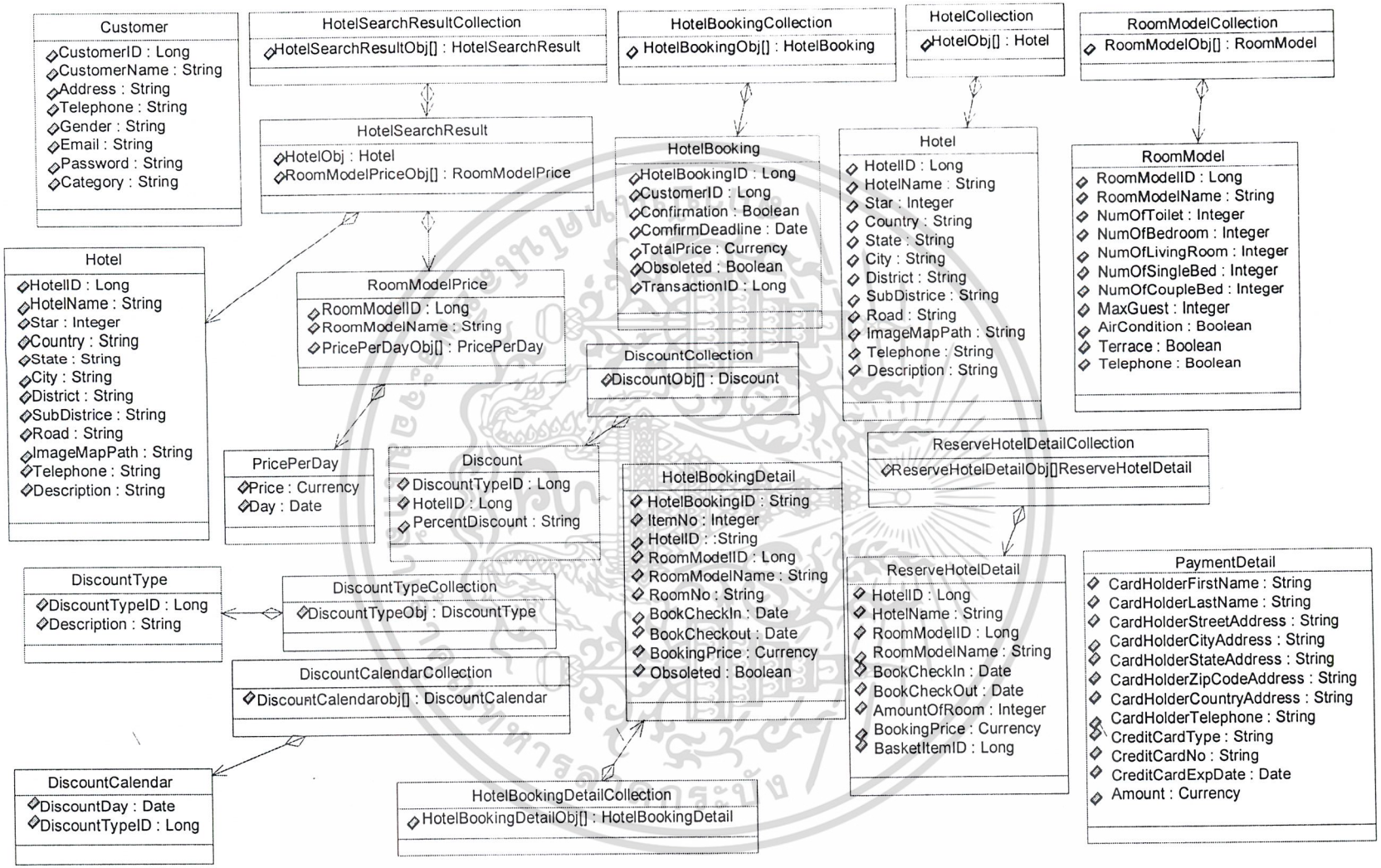
Reserve() จะเป็นการจองรายการที่อยู่ในตะกร้า การจองจะต้องเป็นอะตอมมิก คือ ถ้าจองได้ต้องจองให้ได้ทุกรายการ ถ้าจองไม่ได้ต้องไม่จองให้ซักรายการเดียว

คลาสไดอะแกรมของระบบโรงแรมแสดงได้ดังรูป



รูปที่ 10-7 แสดง Class Diagram ของระบบโรงแรม

132



รูปที่ 10-8 แสดง Class Diagram - Data Object ของระบบโรงแรม

บทที่ 11
การออกแบบระบบทัวร์

การออกแบบระบบทัวร์ แบ่งออกเป็น ส่วนต่างๆ ดังต่อไปนี้

1. การลงทะเบียน และ บริการสมาชิก
2. การจองชุดเดินทาง
3. การจองโรงแรม
4. การจองเครื่องบิน และ รถโดยสาร
5. การเรียกดูรายการจอง
6. การยกเลิก และ ยืนยันการจอง
7. การบริหารชุดเดินทาง

แต่ละส่วนมีรายละเอียดดังนี้

11.1 การลงทะเบียน

สำหรับผู้ใช้ที่ต้องการใช้บริการของระบบทัวร์จะต้องทำการลงทะเบียนกับระบบก่อน โดยผู้ที่ลงทะเบียนนี้จะเป็นคนเดียวกัน หรือคนละคนกับผู้เดินทางก็ได้ (ผู้ใช้ 1 คน อาจจะจองชุดการเดินทางให้กับผู้เดินทางหลาย ๆ คนก็ได้)

ข้อมูลที่ใช้ในการลงทะเบียน ประกอบด้วย

- ชื่อ นามสกุล (ต้องระบุ)
- ที่อยู่
- เบอร์โทรศัพท์
- อีเมล์แอดเดรส (ต้องระบุ)
- รหัสผ่าน (ต้องระบุ)

สำหรับข้อมูลเหล่านี้ ทางระบบเก็บไว้เพื่อใช้ในการติดต่อกับผู้จองในกรณีที่เกิดปัญหา เช่น ชุดเดินทางที่จองไว้มีผู้จองน้อยมากจนต้องยกเลิก ทางระบบก็ต้องบอกผู้จองว่า ชุดเดินทางนี้ได้ถูกยกเลิกไปแล้ว

ผู้ใช้ที่ทำการลงทะเบียนไว้แล้ว เมื่อเข้ามาใช้บริการครั้งต่อไป จะใช้ อีเมล์แอดเดรส และรหัสผ่านที่ตนเองตั้งไว้เป็นการยืนยันตัวบุคคล

11.2 การจองชุดเดินทาง

ประกอบด้วยขั้นตอนต่าง ๆ ดังต่อไปนี้

- ระบุรายละเอียดต่าง ๆ ในการค้นหาชุดเดินทาง
- จองชุดการเดินทาง
- เปลี่ยนแปลงรายละเอียดการจองชุดเดินทาง

การระบุรายละเอียดในการค้นหาชุดเดินทาง

ผู้ใช้สามารถทำการค้นหาชุดเดินทางได้จากข้อมูลต่อไปนี้

- สถานที่ที่ต้องการไป อาจจะเป็น ชื่อสถานที่ , ชื่อจังหวัด หรือชื่อประเทศก็ได้
- จำนวนวันที่เดินทาง
- จำนวนผู้เดินทาง
- วันที่ออกเดินทาง

ผลที่ได้จากการค้นหา คือ ชุดเดินทางต่าง ๆ ที่ผู้ใช้สามารถจองได้ โดยรายละเอียดของชุดเดินทางแต่ละชุดนั้น จะต้องบอกผู้ใช้ด้วยว่า ผู้ใช้จะต้องทำการจองภายในเวลาเท่าใด

การจองชุดการเดินทาง

หลังจากค้นหาชุดเดินทาง และพบชุดเดินทางที่ต้องการแล้ว ผู้ใช้ก็จะทำการจองชุดเดินทางได้ โดย จะต้องใส่ข้อมูลของผู้เดินทางทุกคนที่ผู้ใช้ทำการจองให้ (ที่ขั้นตอนนี้ ถ้าผู้ใช้คนใดยังไม่ได้ลงทะเบียนกับระบบ ก็ต้องลงทะเบียนก่อน แต่ถ้าลงทะเบียนแล้วแต่ยังไม่ได้ล็อกอิน ก็ต้องล็อกอินโดยใช้อีเมล์แอดเดรส และรหัสผ่านที่ตั้งไว้ตอนลงทะเบียน)

ในระบบทัวร์นี้ จะมีตะกร้าไว้ให้บริการเช่นเดียวกับระบบโรงแรม แต่ตะกร้าของระบบทัวร์ จะต่างจากระบบโรงแรมคือ รายการจองในตะกร้าจะบรรจุได้ทั้งรายการจองชุดเดินทาง, โรงแรม ,รถโดยสาร และ เที่ยวบิน โดยจะสามารถเพิ่ม หรือลดรายการจองของทั้ง 4 ระบบได้ตามต้องการ

เมื่อผู้ใช้จองชุดการเดินทาง และใส่ข้อมูลของผู้เดินทางทุกคนแล้ว ทางระบบก็จะแจ้งผลการจองให้ผู้ใช้ทราบ พร้อมทั้งแจ้งให้ทราบด้วยว่า ผู้ใช้จะต้องทำการยืนยันการจองภายในวันที่เท่าไร ซึ่งระบบจะตั้งไว้ที่ก่อนวันออกเดินทาง 7 วัน

การเปลี่ยนแปลงรายละเอียดการจองชุดเดินทาง

ถ้าผู้ใช้ต้องการเปลี่ยนแปลงรายละเอียดการจองชุดเดินทางก็ทำได้ โดยสามารถยกเลิกรายการที่ไม่ต้องการ หรือ ลดจำนวนผู้เดินทางได้

11.3 การจองโรงแรม

ประกอบด้วยบริการต่าง ๆ คือ

- การสืบค้นข้อมูลโรงแรม / ห้องพัก
- การจองห้องพัก
- การยืนยัน หรือ ยกเลิกรายการจองโรงแรม

การสืบค้นข้อมูลโรงแรม / ห้องพัก

ผู้ใช้จะต้องระบุข้อมูล ดังนี้

- รายละเอียดของโรงแรม หรือ ห้องพัก

- วันที่จะเข้าพัก และ ออกจากที่พัก
- จำนวนห้องที่เข้าพัก

ผลลัพธ์จะแสดงรายละเอียดของโรงแรม และห้องพักที่ว่าง ที่ตรงตามเงื่อนไขที่ระบุไว้

รายละเอียดของโรงแรมและห้องพักที่ใช้เป็นเงื่อนไข มีดังนี้

- ชื่อโรงแรม
- ระดับของโรงแรม (1 - 5 ดาว)
- สถานที่ตั้ง เช่น ทวีป , ประเทศ , จังหวัด , อำเภอ
- ราคาห้องพักต่ำสุด หรือ สูงสุดที่ต้องการ
- ระดับของห้องพัก เช่น ห้องพักธรรมดา , ห้องพิเศษ , ห้องสูท , ห้องเดอลุกซ์
- วันเข้าพัก และ วัน ออกจากที่พัก

การจองห้องพัก

การจองห้องพัก จะนำข้อมูลจากการค้นหามาใช้ในการจอง โดยให้ผู้ใช้เลือกรายการที่ต้องการจากผลลัพธ์ของการค้นหา ระบบจะใช้ชื่อบริษัททัวร์ในการจองห้องพักให้กับสมาชิก และจะเก็บรหัสการจอง ซึ่งได้รับมาจากระบบโรงแรมไว้ เพื่อใช้ในการอ้างอิงกับสมาชิกต่อไป

11.4 การจองเครื่องบิน และ รถโดยสาร

ประกอบด้วยบริการต่างๆ คือ

- การสืบค้นข้อมูลต่างของระบบรถโดยสาร – สายการบิน
- การจองที่นั่ง

การค้นหาข้อมูลของรถโดยสาร

ผู้ที่เข้ามาใช้บริการสามารถหาเส้นทางการเดินทางได้โดยกรอกข้อมูลที่จำเป็น เกี่ยวกับรายละเอียดต่างๆ ที่ต้องการค้น ข้อมูลที่ใช้ประกอบเป็นเงื่อนไข มีดังนี้

- สถานที่ต้นทาง
- สถานที่ปลายทาง
- ตัวเลือกการเดินทาง (ไปอย่างเดียว หรือไปกลับ)
- วัน เวลา ที่จะออกเดินทาง
- บริษัทที่ต้องการ
- ต้องการที่นั่งริมหน้าต่างหรือไม่
- ต้องการต่อรถหรือไม่
- ต้องการห้องน้ำในรถหรือไม่
- จำนวนที่นั่งที่ต้องการ

ผลลัพธ์ที่คืนกลับมามีรายละเอียดดังนี้

- ชื่อสายการเดินรถ
- สถานีต้นทาง / ปลายทาง
- วัน และ เวลารถออก / เวลารถถึงที่หมาย
- ราคารวม

การค้นหาข้อมูลของสายการบิน

ผู้ที่เข้ามาใช้บริการสามารถหาเส้นทางการเดินทางได้โดยกรอกข้อมูลที่จำเป็น เกี่ยวกับรายละเอียดต่างๆ ที่ต้องการค้นหา จากนั้นระบบจะทำการค้นการเส้นทางให้ และคืนผลลัพธ์กลับมาเป็นรายละเอียดต่างๆ ของเที่ยวการเดินทางที่ตรงกับที่กำหนดเงื่อนไขการค้นหา ข้อมูลที่ใช้ประกอบเป็นเงื่อนไขมีดังนี้

- สถานที่ต้นทาง
- สถานที่ปลายทาง
- ตัวเลือกการเดินทาง (ไปอย่างเดียว หรือไปกลับ)
- วัน เวลา ที่จะออกเดินทาง
- ชนิดของที่นั่ง
- สายการบิน
- ต้องการที่นั่งริมหน้าต่างหรือไม่
- ต้องการที่นั่งสูบบุหรี่หรือไม่
- ต้องการต่อเที่ยวบินหรือไม่
- จำนวนที่นั่งที่ต้องการ

ผลลัพธ์ที่คืนกลับมามีรายละเอียดดังนี้

- ชื่อเที่ยวบิน
- สถานีต้นทาง – ปลายทาง
- วัน และ เวลาขึ้นเครื่อง/เวลาลงเครื่อง
- ราคารวม

การจองที่นั่ง

เช่นเดียวกับการจองโรงแรม ระบบทัวร์จะใช้ชื่อบริษัททัวร์ในการจองที่นั่งและเที่ยวบินให้กับสมาชิก และจะเก็บรหัสการจอง ซึ่งได้รับมาจากระบบจองรถโดยสารและระบบจองเที่ยวบินไว้ เพื่อใช้ในการอ้างอิงกับสมาชิกต่อไป

11.5 การเรียกดูรายการจอง

ผู้ใช้บริการสามารถเรียกดูรายการจองทั้งหมดได้ ไม่ว่าจะเป็นชุดเดินทาง, โรงแรม ,รถโดยสาร หรือ เที่ยวบิน โดยแต่ละรายการจอง จะมีข้อมูลการจองของทั้ง 4 ระบบอยู่ แต่ถ้าไม่ได้จองระบบใดก็จะไม่แสดงรายการของระบบนั้น กล่าวคือ จะแสดงรายการจองเฉพาะระบบที่จองไว้เท่านั้น

ในส่วนของการแสดงรายการจองชุดเดินทาง จะสามารถ เรียกดู ชื่อผู้เดินทางที่ได้ระบุไว้ตอนจองได้ และยังสามารถลบรายชื่อผู้เดินทางได้ด้วย (คือลดจำนวนคนเดินทางนั่นเอง) แต่จะไม่สามารถเพิ่มรายชื่อผู้เดินทางได้ เพราะชุดเดินทางนั้นๆ อาจเต็มแล้วก็ได้

11.6 การยกเลิก หรือ ยืนยันการจอง

มีรายละเอียดดังนี้

การยกเลิกการจองชุดเดินทาง , โรงแรม . รถเดินทาง และเครื่องบิน

การยกเลิกนี้ทำได้ทั้งการยกเลิกทุกรายการ หรือ ยกเลิกบางรายการก็ได้ แต่จะทำได้ก็ต่อเมื่อยังไม่ได้ทำการยืนยันการจอง และยังอยู่ภายในช่วงเวลาที่อนุญาตให้มีการเปลี่ยนแปลงรายการจองได้ (คือ ก่อนวันสิ้นสุดการยืนยันการจอง ซึ่งก็คือ 7 วันก่อนวันออกเดินทาง)

การยืนยันการจองชุดเดินทาง , โรงแรม . รถเดินทาง และเครื่องบิน

การยืนยันการจองชุดเดินทางนี้ เป็นการโอนเงินให้กับระบบ โดยการยืนยันนี้เป็นการยืนยันทุกรายการที่ผู้ใช้ทำการจองไว้ ผู้ใช้จะทำการยืนยันเป็นบางรายการไม่ได้ (ถ้าต้องการยืนยันเป็นบางรายการก็ให้ไปยกเลิกรายการที่ไม่ต้องการก่อน และจึงกลับมายืนยันการจอง)

ถ้าผู้ใช้คนใดที่ไม่ได้ทำการยืนยันการจองภายในระยะเวลาที่กำหนด ระบบจะทำการยกเลิกการจองในครั้งนั้นโดยอัตโนมัติ แล้วส่งอีเมล์ไปบอกผู้ใช้ว่า ได้ทำการยกเลิกการจองไปแล้ว

11.7 การบริหารชุดเดินทาง

เริ่มจากการสร้างชุดเดินทาง ซึ่งประกอบด้วย

- จำนวนวันของการเดินทาง
- สถานที่ และเวลาที่ไปในแต่ละวัน

ส่วนวันออกเดินทางนั้นจะกำหนดตอนเปิดชุดเดินทาง

การเปิดชุดเดินทางนั้น จะเปิดได้ก็ต่อเมื่อมีการจองที่พัก , รถ , เครื่องบิน ให้กับชุดเดินทางนั้นแล้ว พร้อมทั้งกำหนดวันออกเดินทาง , มัคคุเทศก์ , ราคาต่อคนซึ่งมี 2 อัตรา คือ อัตราเด็ก และ อัตราผู้ใหญ่ และกำหนดจำนวนผู้เดินทางที่ชุดเดินทางนั้น ๆ จะรองรับได้ ส่วนเหตุผลที่ต้องสร้างชุดเดินทางก่อน แล้วจึงเปิดชุดเดินทางนั้น ก็เพราะว่า ใน 1 ชุดเดินทางนั้น จะมีการออกเดินทางหลายครั้งได้ เช่น ชุดเดินทางไปทัวร์ฮ่องกง ซึ่งมีออกเดินทางกันทุกอาทิตย์ เป็นต้น

หลังจากเปิดชุดเดินทางแล้ว ระบบจึงจะเปิดให้ผู้ใช้ทำการจองได้ ซึ่งผู้ใช้จะจองและยืนยันการจองได้ภายในระยะเวลาที่กำหนดไว้ (ทางระบบได้กำหนดไว้ว่า 7 วันก่อนวันออกเดินทาง)

11.8 การออกแบบระบบงาน

การออกแบบระบบทัวร์ตามความต้องการข้างต้น แสดงได้ดังนี้

- แสดงการไหลของข้อมูลด้วย ดาต้าโฟลว์ไดอะแกรม (Data Flow Diagram)
- แสดงการออกแบบฐานข้อมูลเชิงสัมพันธ์ได้ด้วย อีอาไดอะแกรม (ER Diagram)
- แสดงการออกแบบเชิงออกเจ็กโอเรียนเต็ดด้วยคลาสไดอะแกรม (Class Diagram)
- แสดงการเรียกใช้คอมโพเนนต์ด้วยคอมโพเนนต์ไดอะแกรม (Class Diagram)

11.8.1 ดาต้าโฟลว์ไดอะแกรม

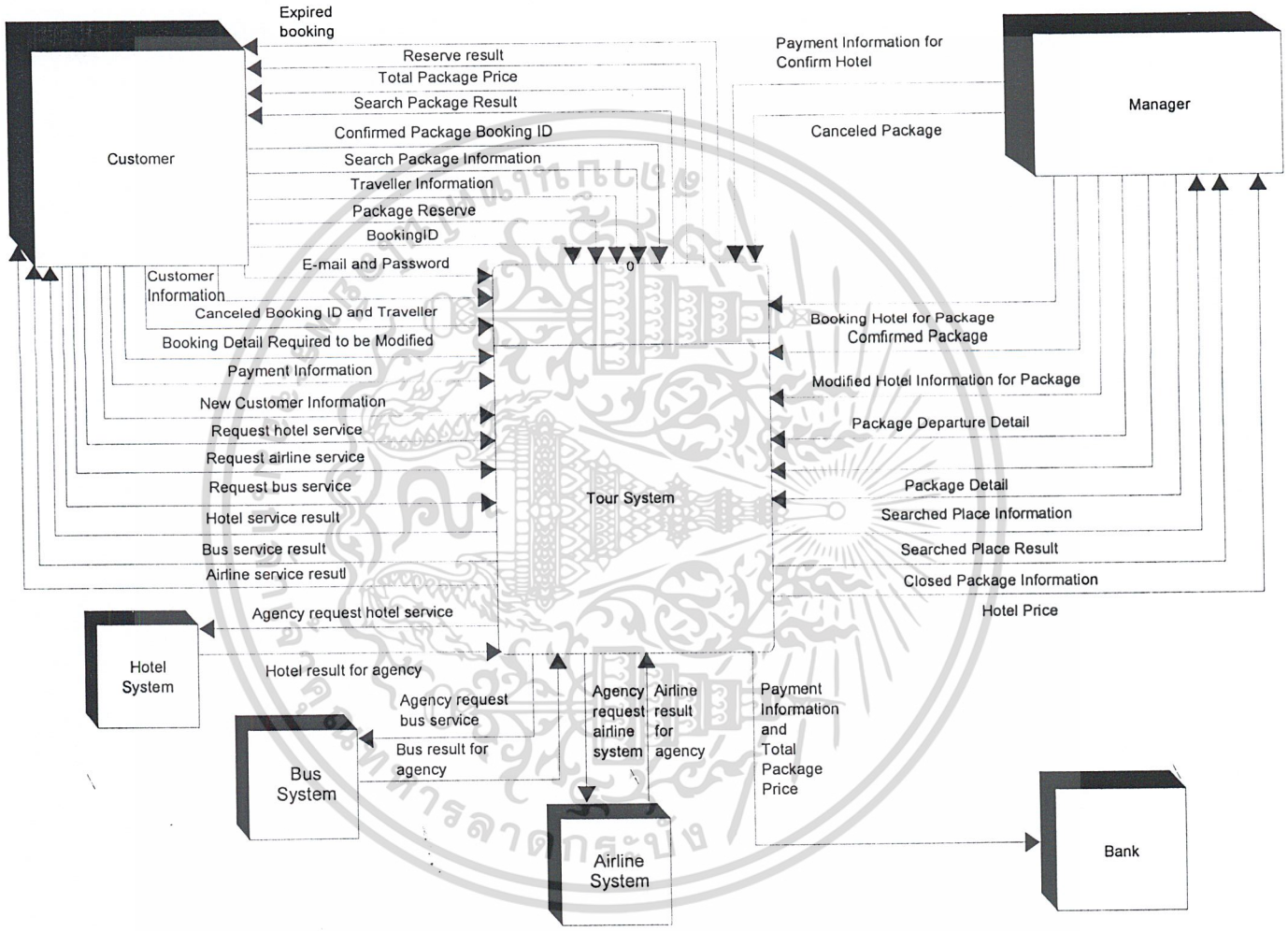
ดาต้าโฟลว์ไดอะแกรมของระบบทัวร์ในชั้นคอนเทคเลเวล(Context Level) ประกอบด้วยโพรเซสหลักคือ Tour System และ เอ็กซ์เทอร์นอล เอ็นทิตี้(External Entity) อีก 6 ตัวคือ

- *Customer* คือลูกค้าของระบบทัวร์
- *Manager* คือผู้จัดการทัวร์ มีหน้าที่สร้างและดูแลชุดการเดินทาง
- *Bank* คือ คือระบบธนาคารที่จะโอนเงินจากบัญชีของลูกค้าเข้าบัญชีของทัวร์เมื่อมีการชำระเงิน
- *Hotel System* คือ ระบบโรงแรมที่ทัวร์ใช้บริการอยู่
- *Bus System* คือ ระบบรถโดยสารที่ทัวร์ใช้บริการอยู่
- *Airline System* คือ ระบบสายการบินที่ทัวร์ใช้บริการอยู่

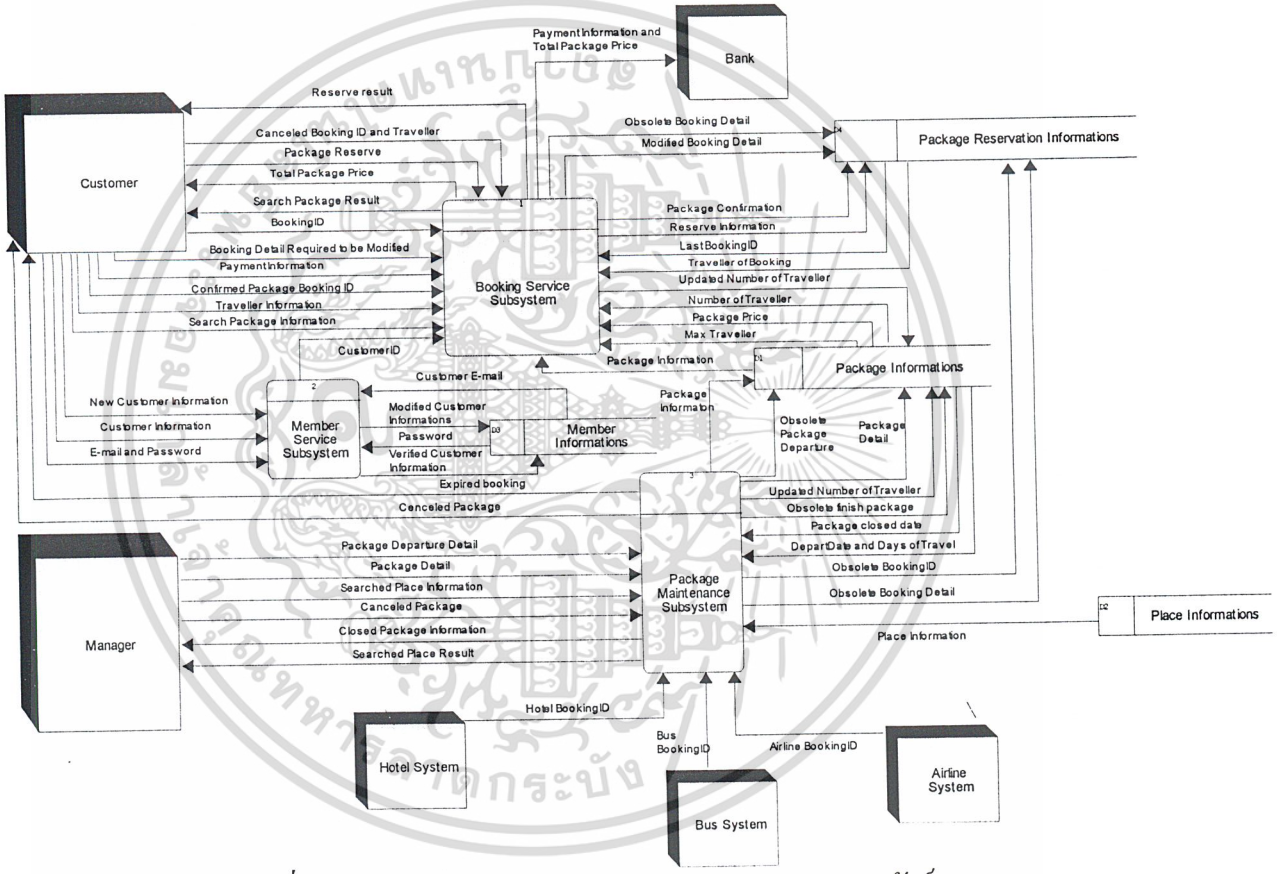
ในชั้นถัดมาประกอบด้วยโพรเซสหลัก 5 โพรเซส คือ

- *Member Service Sub System* เป็น โพรเซส ที่ทำหน้าที่ให้บริการสมาชิกของระบบ เช่น การลงทะเบียน , การแก้ไขข้อมูลสมาชิก และ การล็อกอินเข้าสู่ระบบ
- *Booking Service Subsystem* เป็นโพรเซสที่ให้บริการค้นหาและจองชุดการเดินทาง รวมถึงการยกเลิกและยืนยันชุดการเดินทางด้วย
- *Package Maintenance Subsystem* เป็นโพรเซสที่ทำหน้าที่เกี่ยวกับการสร้างชุดการเดินทาง การเปิดชุดการเดินทางเพื่อให้สมาชิกจองได้
- *Manipulate External System for Package* เป็นโพรเซสที่ทำหน้าที่จัดการจองโรงแรม รถโดยสารและเที่ยวบินสำหรับชุดการเดินทางหนึ่งๆ
- *External Service Subsystem* เป็นโพรเซสที่ทำหน้าที่เกี่ยวกับการจองรถโดยสาร เที่ยวบินและโรงแรมให้กับสมาชิก

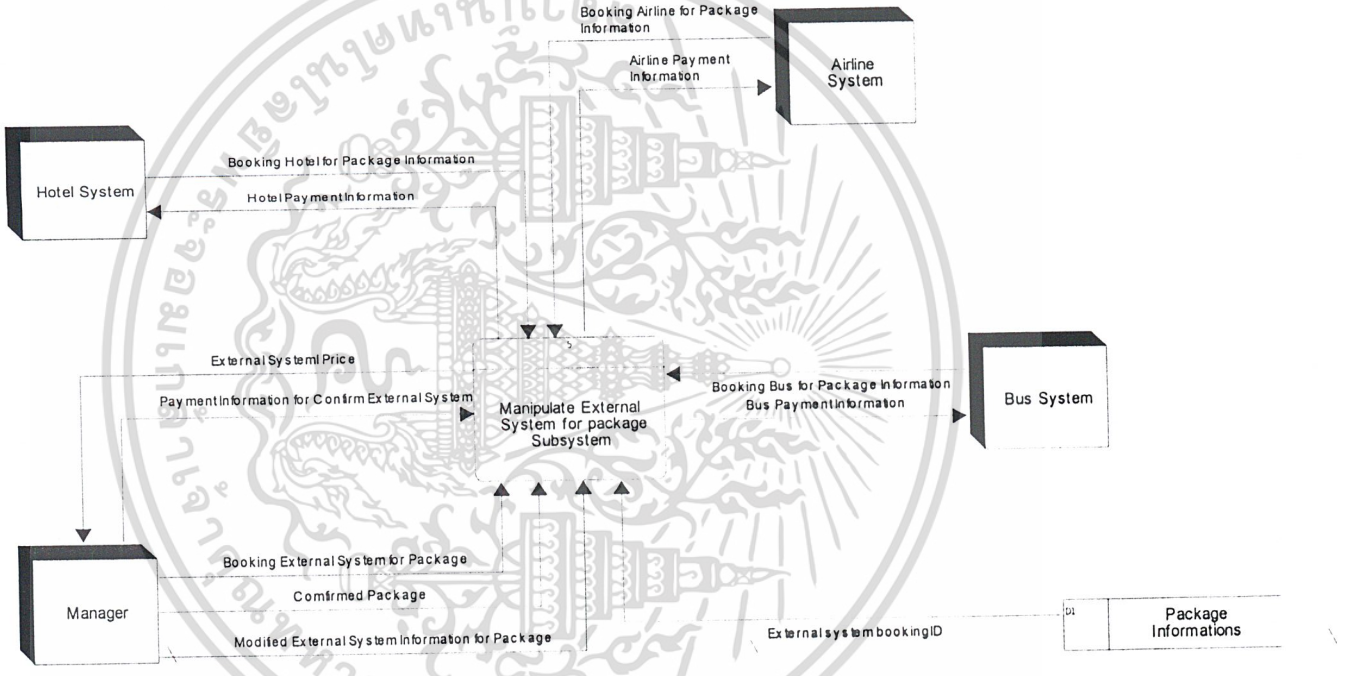
ดาต้าโฟลว์ไดอะแกรมของระบบทัวร์แสดงได้ดังรูป



รูปที่ 11-1 แสดง Data Flow Diagram – Context Level ของระบบทัวร์



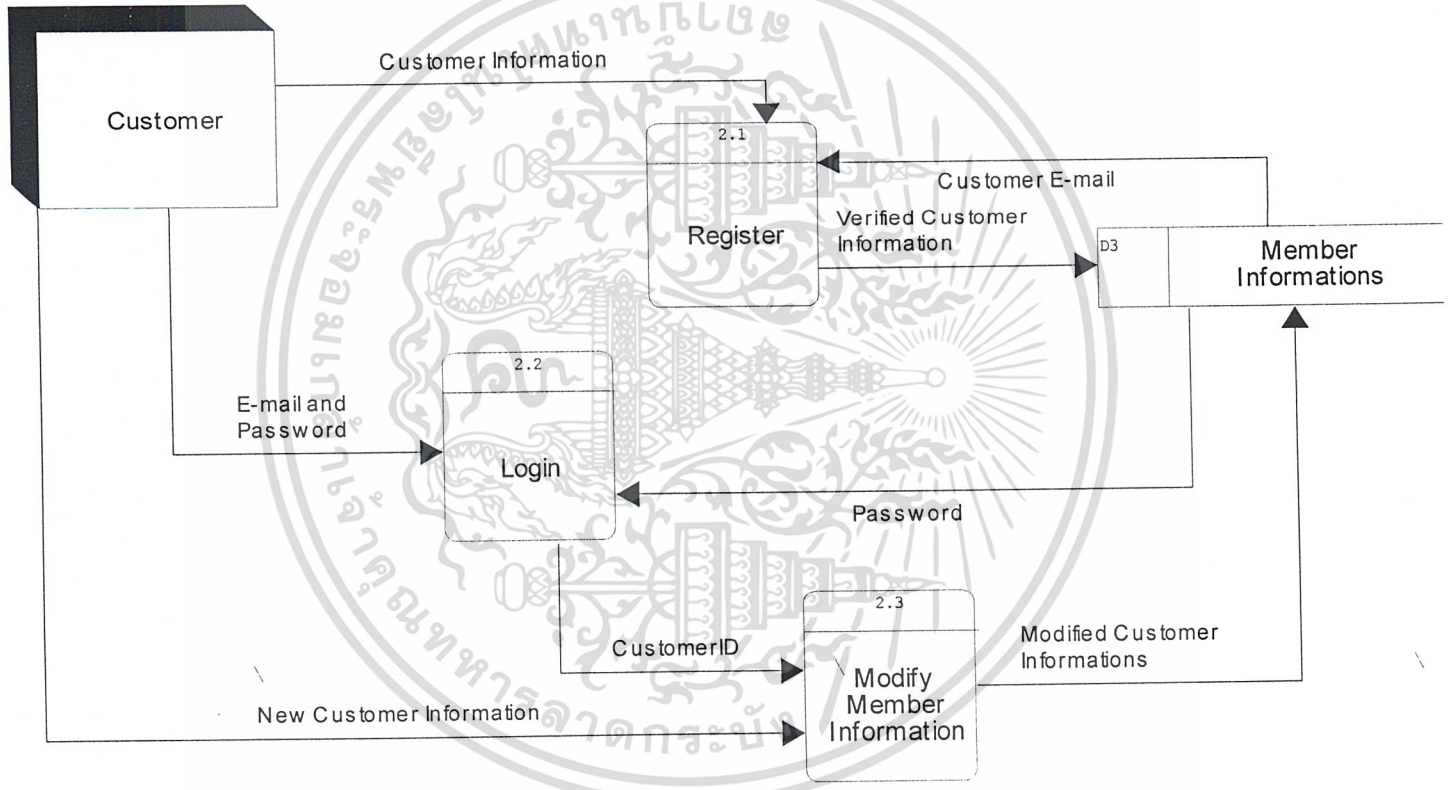
รูปที่ 11-2 แสดง Data Flow Diagram – Level1-1 ของระบบทัวร์



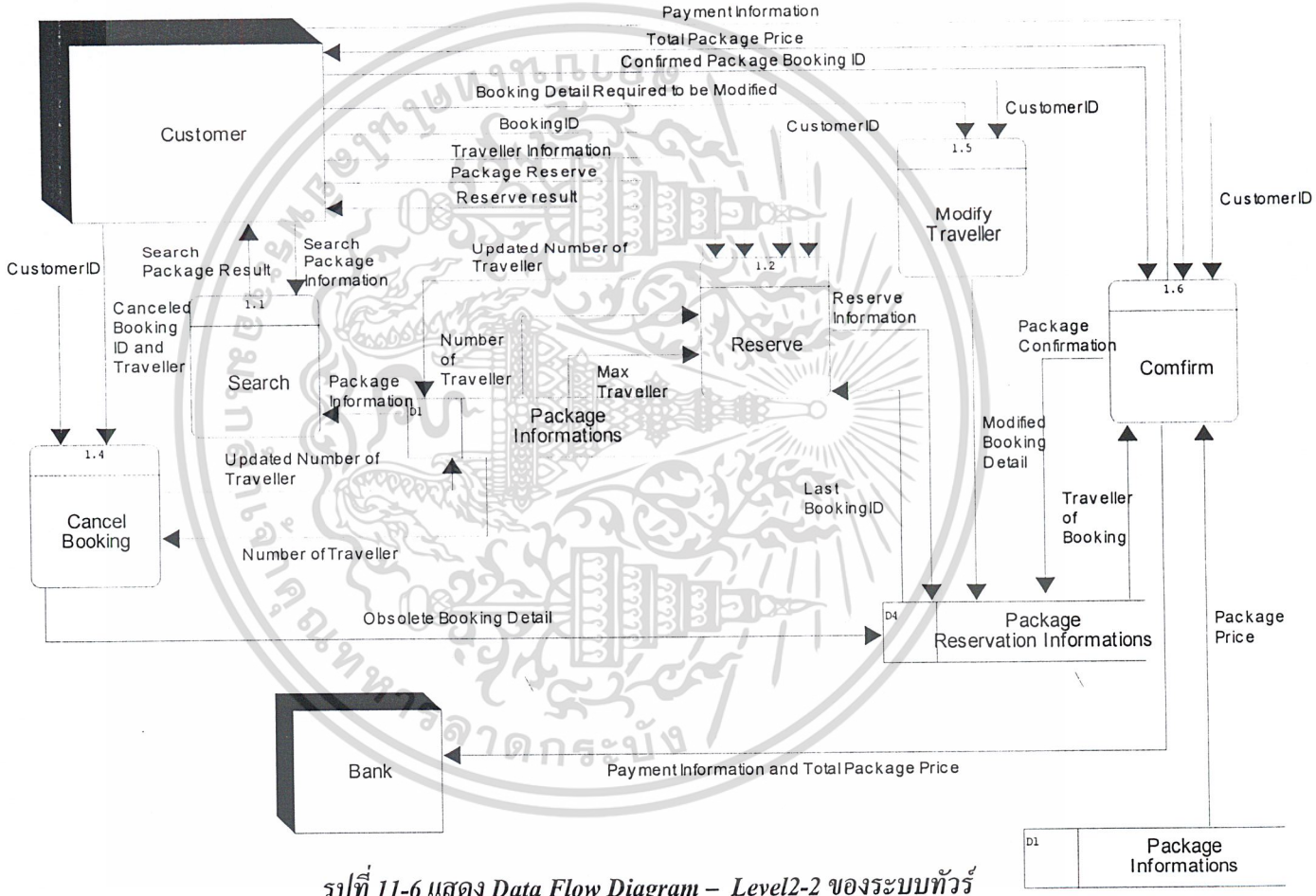
รูปที่ 11-3 แสดง Data Flow Diagram – Level1-2 ของระบบทัวร์

142

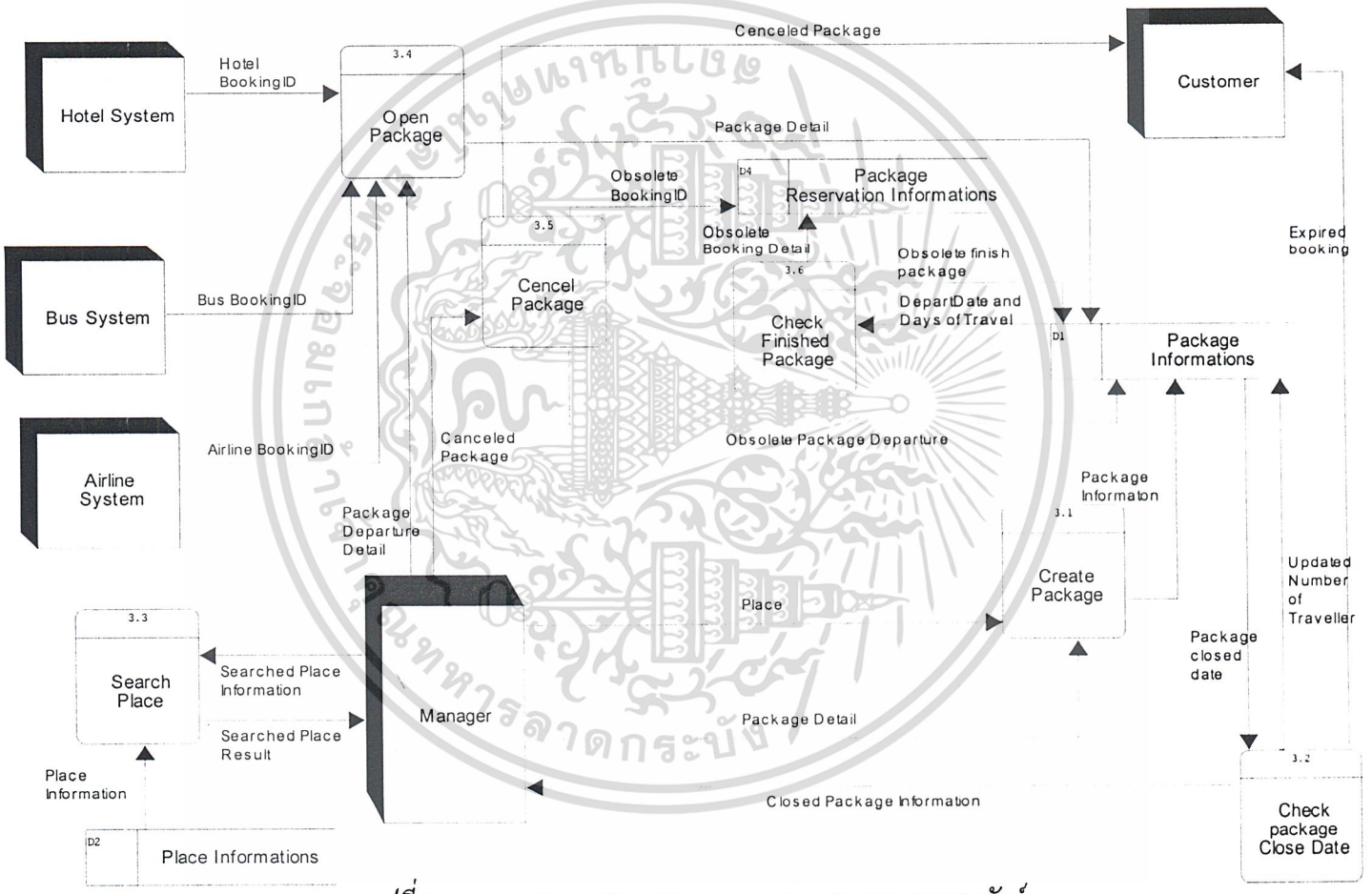
รูปที่ 11-4 แสดง Data Flow Diagram – Level1-3 ของระบบทัวร์



รูปที่ 11-5 แสดง Data Flow Diagram – Level2-1 ของระบบทัวร์

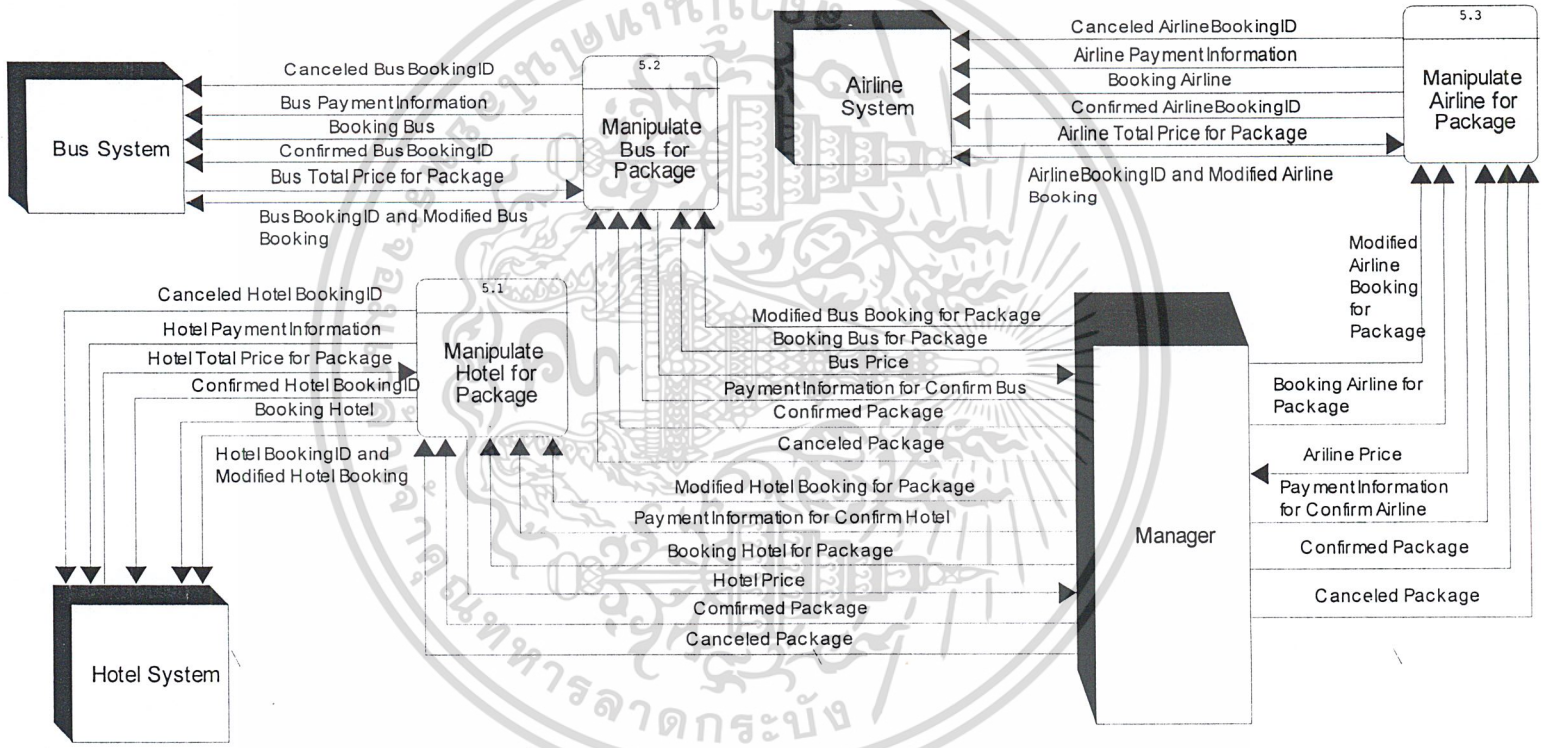


รูปที่ 11-6 แสดง Data Flow Diagram – Level2-2 ของระบบทัวร์



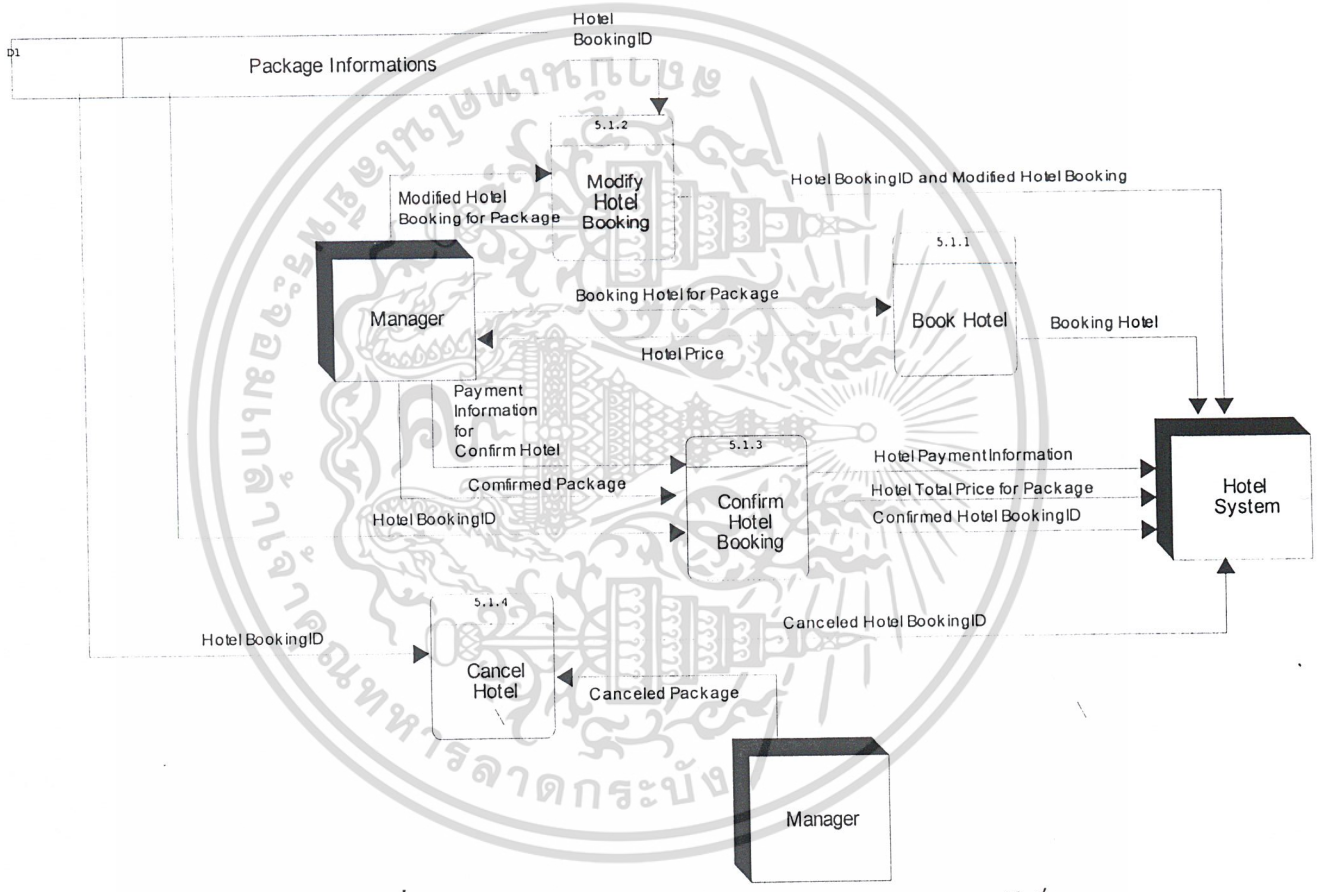
รูปที่ 11-7 แสดง Data Flow Diagram – Level2-3 ของระบบทัวร์

146



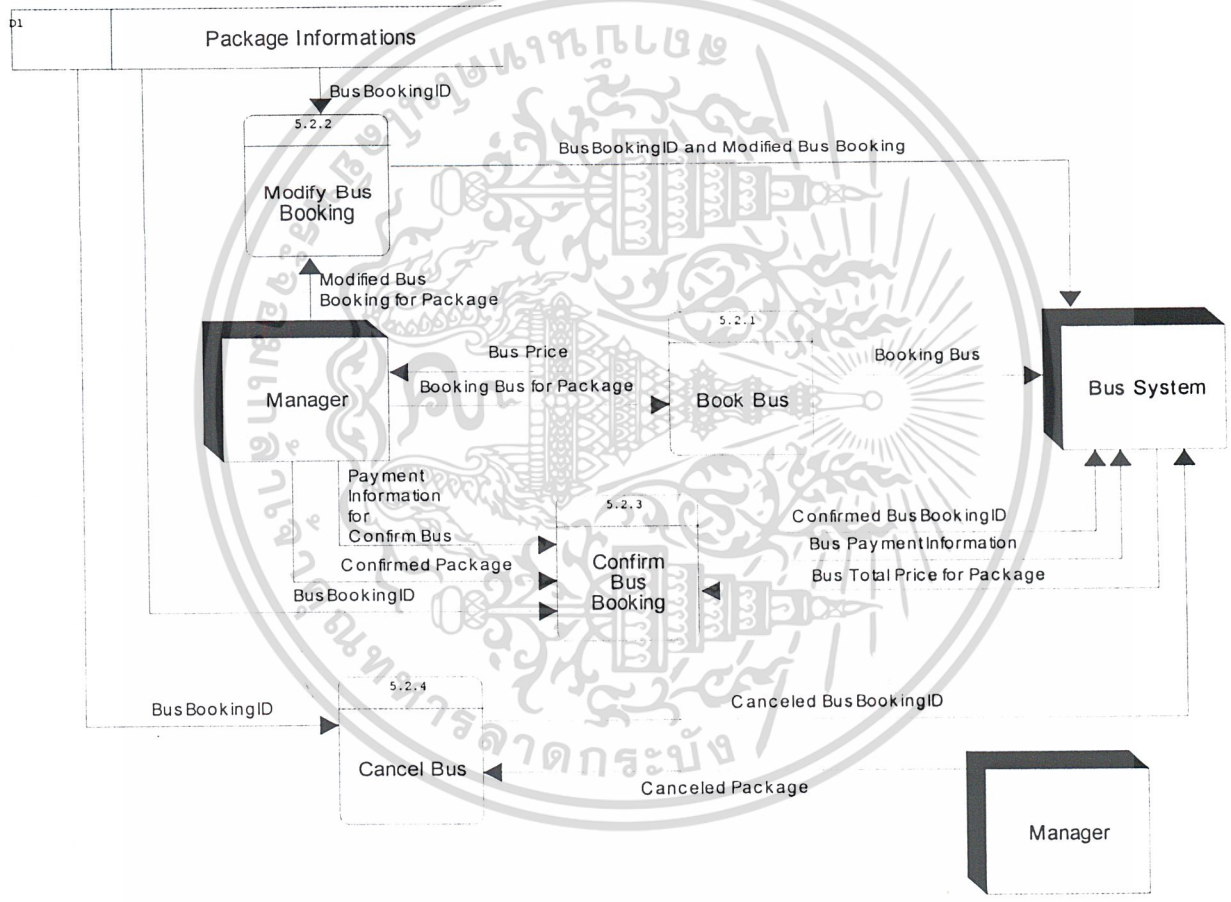
รูปที่ 11-8 แสดง Data Flow Diagram – Level2-4 ของระบบทัวร์

147



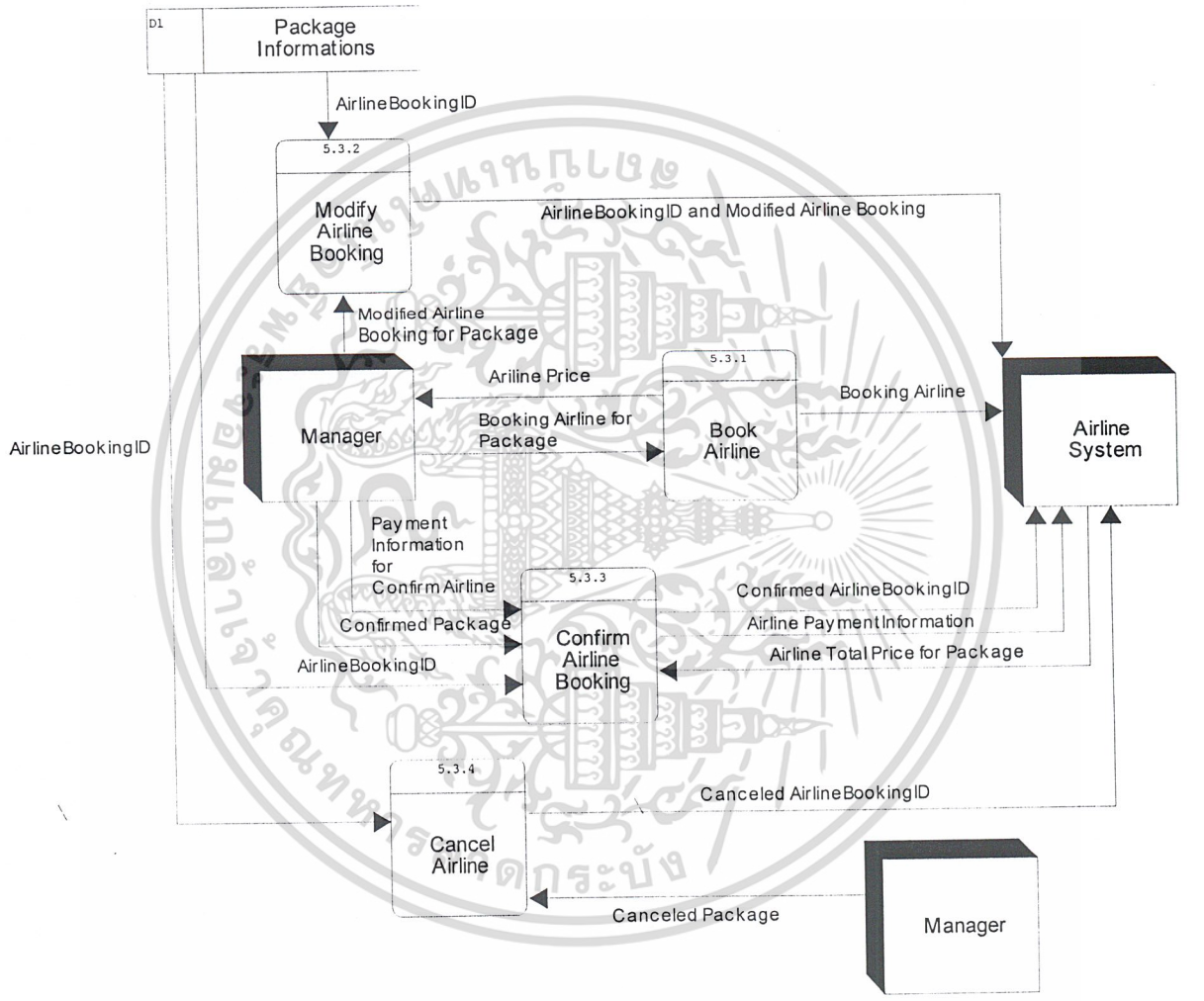
รูปที่ 11-9 แสดง Data Flow Diagram – Level3-1 ของระบบทัวร์

148



รูปที่ 11-10 แสดง Data Flow Diagram – Level3-2 ของระบบทัวร์

149



รูปที่ 11-11 แสดง Data Flow Diagram – Level3-3 ของระบบทัวร์

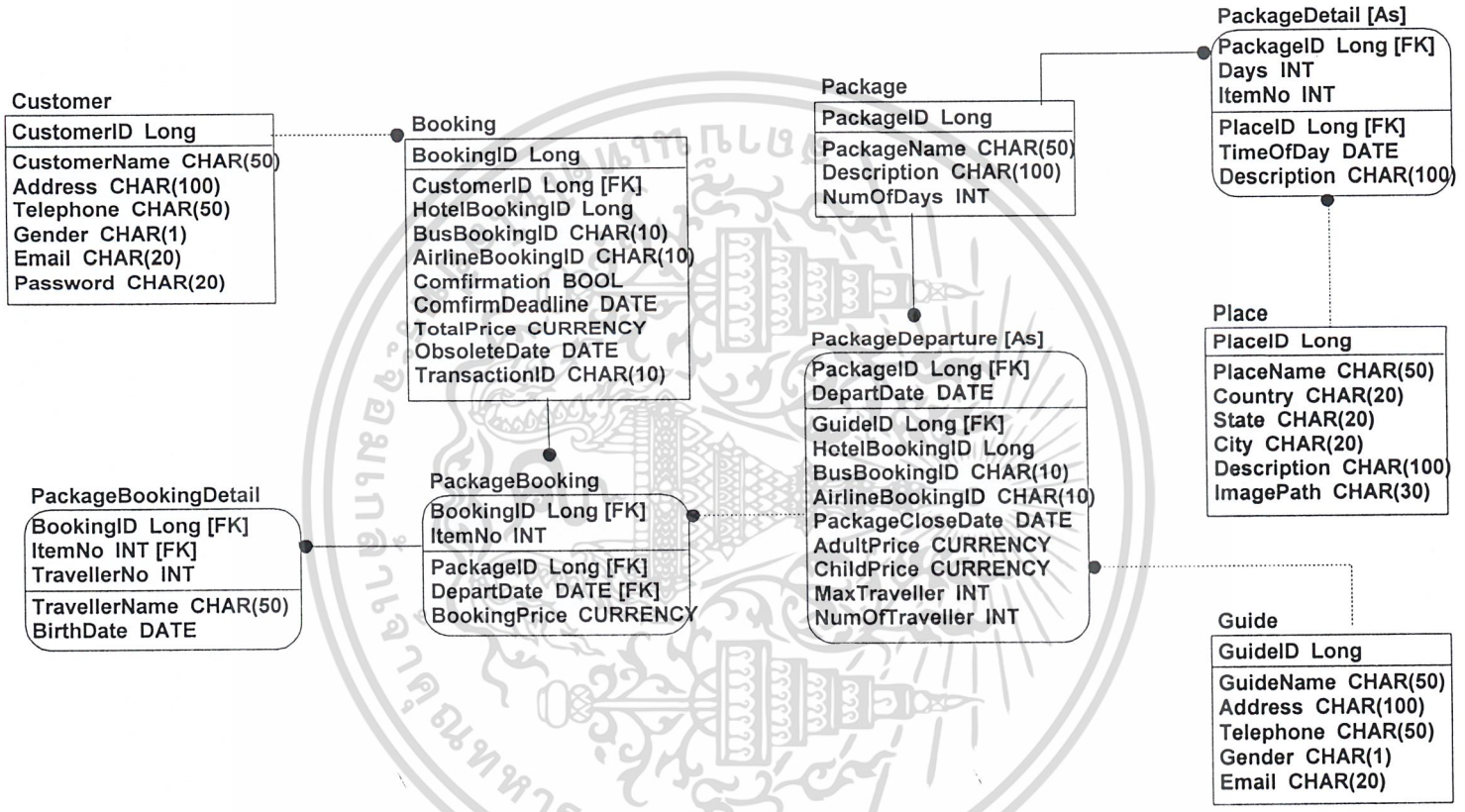
150

11.8.2 อีอาไดอะแกรม

อีอาไดอะแกรมของระบบทัวร์ประกอบด้วย 9 เอ็นทิตี้ ไทพ์(Entity Type) คือ ซึ่งแบ่งได้ 4 ส่วน คือ

- ส่วนข้อมูลสมาชิก ประกอบด้วย 1 เอ็นทิตี้ ไทพ์ คือ Customer ซึ่งเก็บข้อมูลสมาชิกของระบบ
- ส่วนข้อมูลชุดเดินทาง ประกอบด้วย 4 เอ็นทิตี้ ไทพ์ คือ Package , PackageDetail , PackageDeparture และ Place ซึ่งเก็บข้อมูลชุดการเดินทางที่มีอยู่ , ชุดการเดินทางที่เปิดให้จอง , รายละเอียดสถานที่ของชุดการเดินทาง เป็นต้น
- ส่วนข้อมูลการจอง ประกอบด้วย 3 เอ็นทิตี้ ไทพ์ คือ Booking , PackageBooking และ PackageBookingDetial ซึ่งเก็บข้อมูลการจองของลูกค้าไม่ว่าจะเป็นการจองชุดการเดินทาง จองโรงแรม จองรถโดยสารและจองเที่ยวบิน โดยข้อมูลการจองโรงแรม รถโดยสาร และเครื่องบิน จะอยู่ที่ระบบนั้นๆ ตามลำดับ ส่วนระบบทัวร์จะเก็บแค่หมายเลขอ้างอิงไว้เท่านั้น
- ส่วนข้อมูลไกด์ ประกอบด้วย 1 เอ็นทิตี้ ไทพ์ คือ Guide ซึ่งเก็บข้อมูลไกด์ของบริษัท เช่น ชื่อ ที่อยู่ เพศ เบอร์โทรศัพท์ และอีเมล์ เป็นต้น

อีอาไดอะแกรมของระบบแสดงได้ดังรูป



รูปที่ 11-12 แสดง ER Diagram ของระบบทัวร์

11.8.3 คลาสไดอะแกรม

คลาสไดอะแกรมของระบบทัวร์ประกอบด้วย 3 คลาสคือ CustomerService, BookingService, TourService และ BankingService (BankingService จะไม่กล่าวถึงเพราะอยู่นอกระบบงาน)

สรุป คลาส, อินเตอร์เฟส และ เมดถอด ของ คลาสไดอะแกรม ได้ดังนี้

Class	Interface	Method
CustomerService	ICustomer	Login()
		Register()
		GetCustomer()
		EditCustomer()
BookingService	IBookmanager	Reserve()
		CancelAllBooking()
		CancelBooking()
		CancelPackageBookingDetail()
		CancelHotelBooking()
		CancelHotelBookingDetail()
		CancelBusBooking()
		CancelBusBookingDetail()
		CancelAirlineBooking()
		CancelAirlineBookingDetail()
		ConfirmBooking()
	IBookViewer	GetBooking()
		GetHotelBooking()
		GetHotelBookingDetail()
		GetPackgeBooking()
		GetackgeBookingDetail()
		ViewBusBookingInfo()
		ViewAirlineBookingInfo()
	IBasket	CreateBasket()
		AddPackageBasketItem()
		AddHotelBasketItem()
		AddBusBasketitem()
		AddAirlineBasketItem()

Class	Interface	Method
BookingService	IBasket	RemovePackageBasketItem()
		RemoveHotelBasketItem()
		RemoveBusBasketItem()
		RemoveAirlineBasketItem()
		ViewBasket()
HotelService	ISearch	SearchPackage()
		SearchHotel()
		SearchBus()
		SearchAirline()
	IHotelManager	CreatePackage()
		CreatePackageDetail()
		OpenPackage()
		AddPlace()
		AddGuide()
		DeletePackage()
		DeletePacageDetail()
		DeletePackageDeparture()
		DeletePlace()
		DeleteGuide()
		DeleteRoomModel()
		DeletePrice()

ตารางที่ 11-1 แสดงคลาส อินเตอร์เฟส และเมดถอดของระบบทัวร์

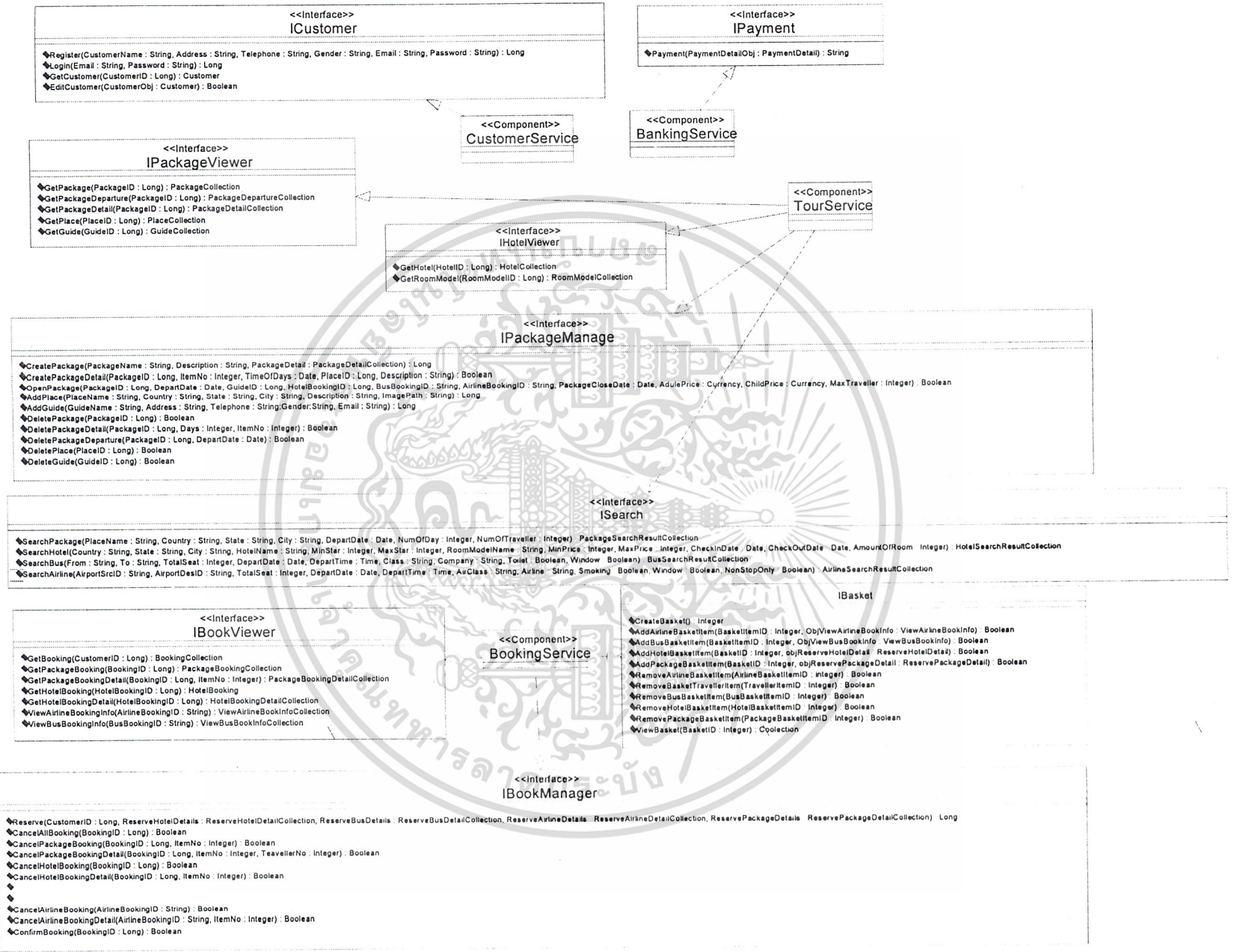
รายละเอียดของเมดถอดที่สำคัญ

Register() จะเป็นการลงทะเบียนเป็นสมาชิกของระบบ โดยอีเมล์ที่ระบุจะต้องไม่ซ้ำกับที่มีอยู่

Reserve() จะเป็นการจองรายการที่อยู่ในตะกร้า การจองจะต้องเป็นอะตอมมิก คือ ถ้าจองได้ต้องจองให้ได้ทุกรายการ ถ้าจองไม่ได้ต้องไม่จองให้ซักรายการเดียว รายการในตะกร้าที่เป็นส่วนของการจองโรงแรม รถโดยสารและเครื่องบินนั้น ระบบทัวร์จะต้องเรียกใช้บริการของระบบภายนอก กล่าวคือ จะต้องไปทำการจองให้แทนสมาชิกและจะเก็บในที่อ้างอิงเอาไว้แทน

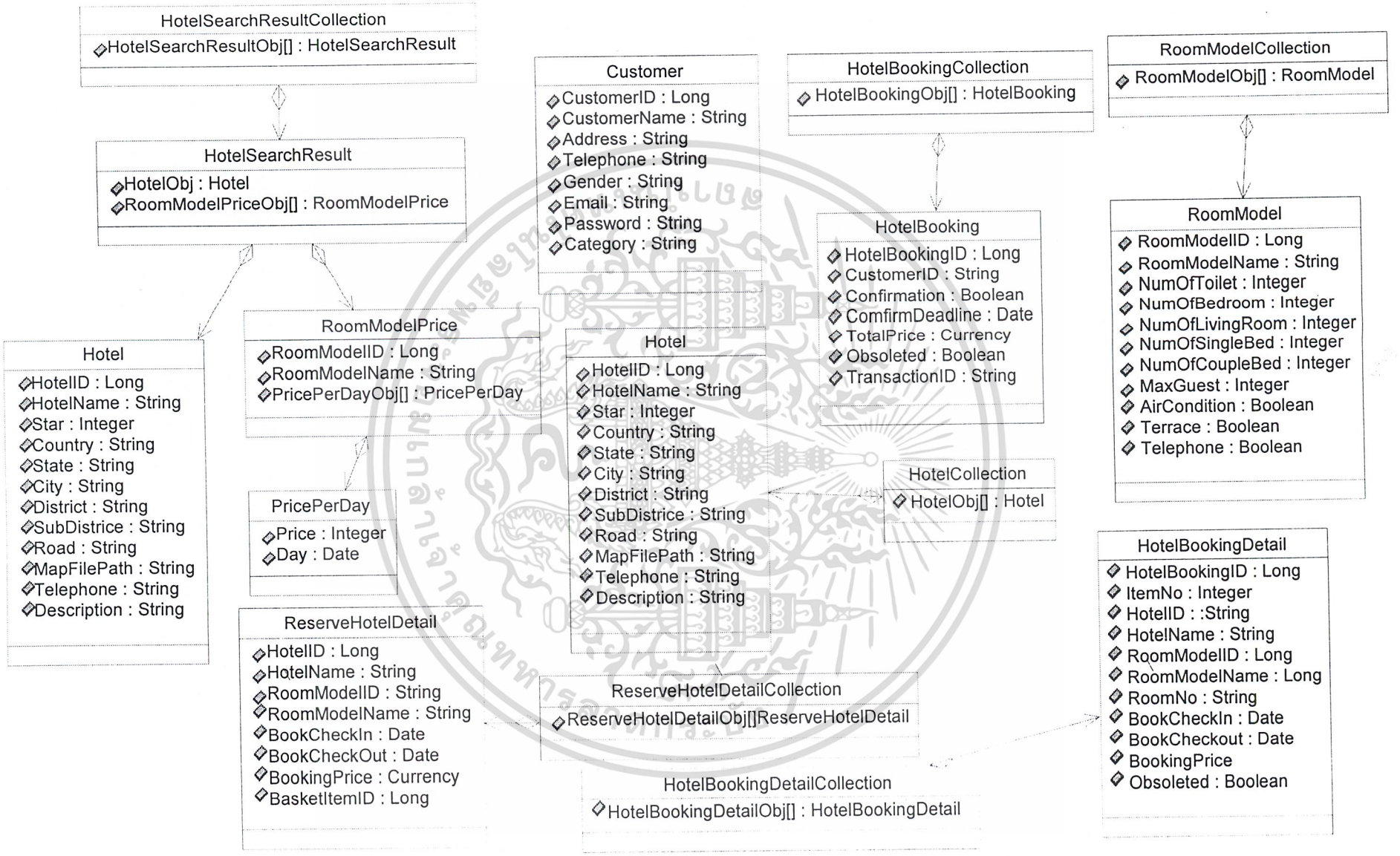
เมดถอดที่เกี่ยวกับระบบโรงแรม ระบบรถโดยสารและระบบสายการบิน เช่น ViewAirlineBookingInfo(), ViewBusBookingInfo(), GetHotelBooking(), GetHotelBookingDetail() เมดถอดเหล่านี้จะต้องไปเรียกใช้บริการของระบบดังกล่าว ตามลำดับ

Class Diagram ของระบบทัวร์แสดงได้ดังรูป



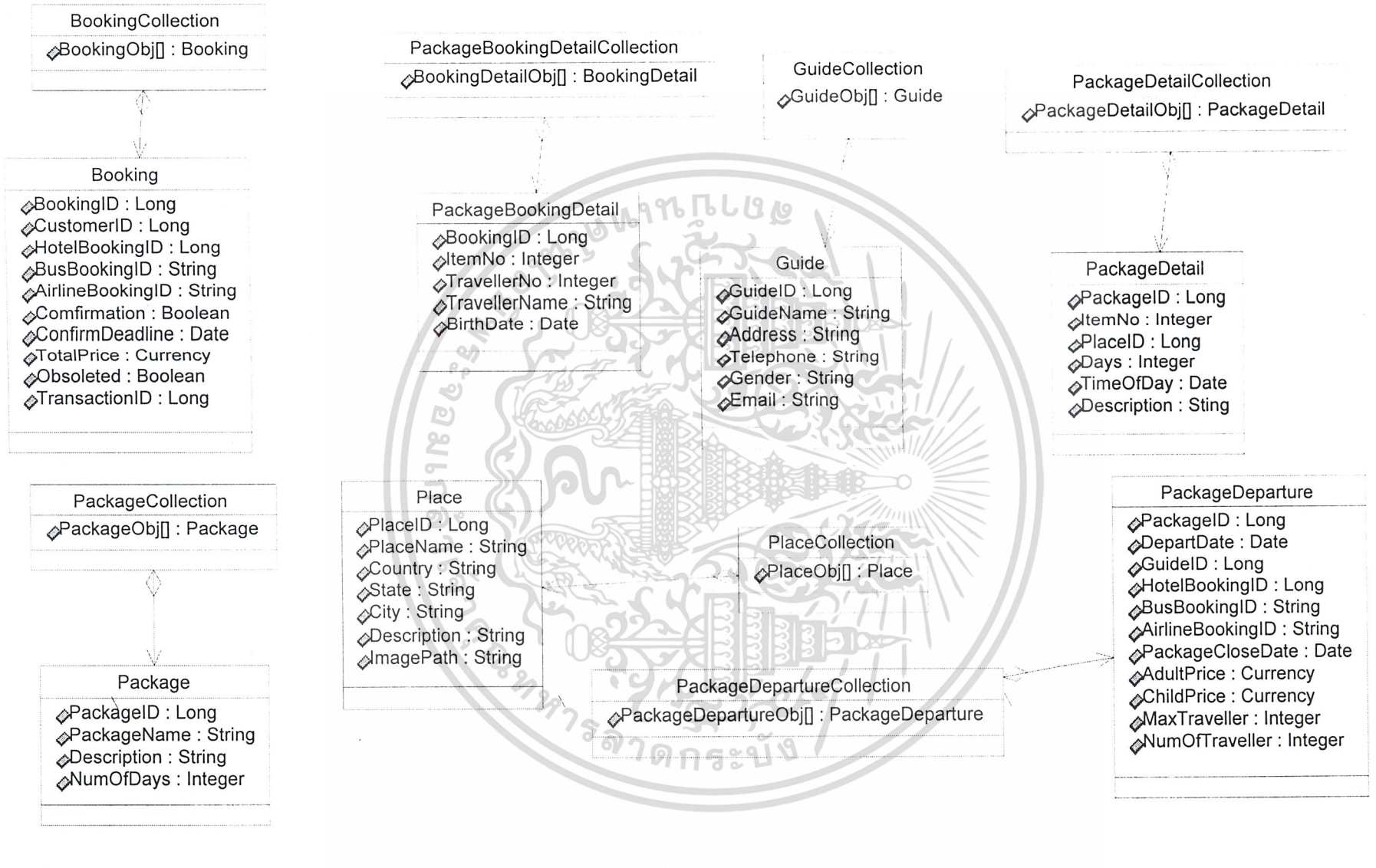
รูปที่11-13 แสดง Class Diagram ของระบบทัวร์

155

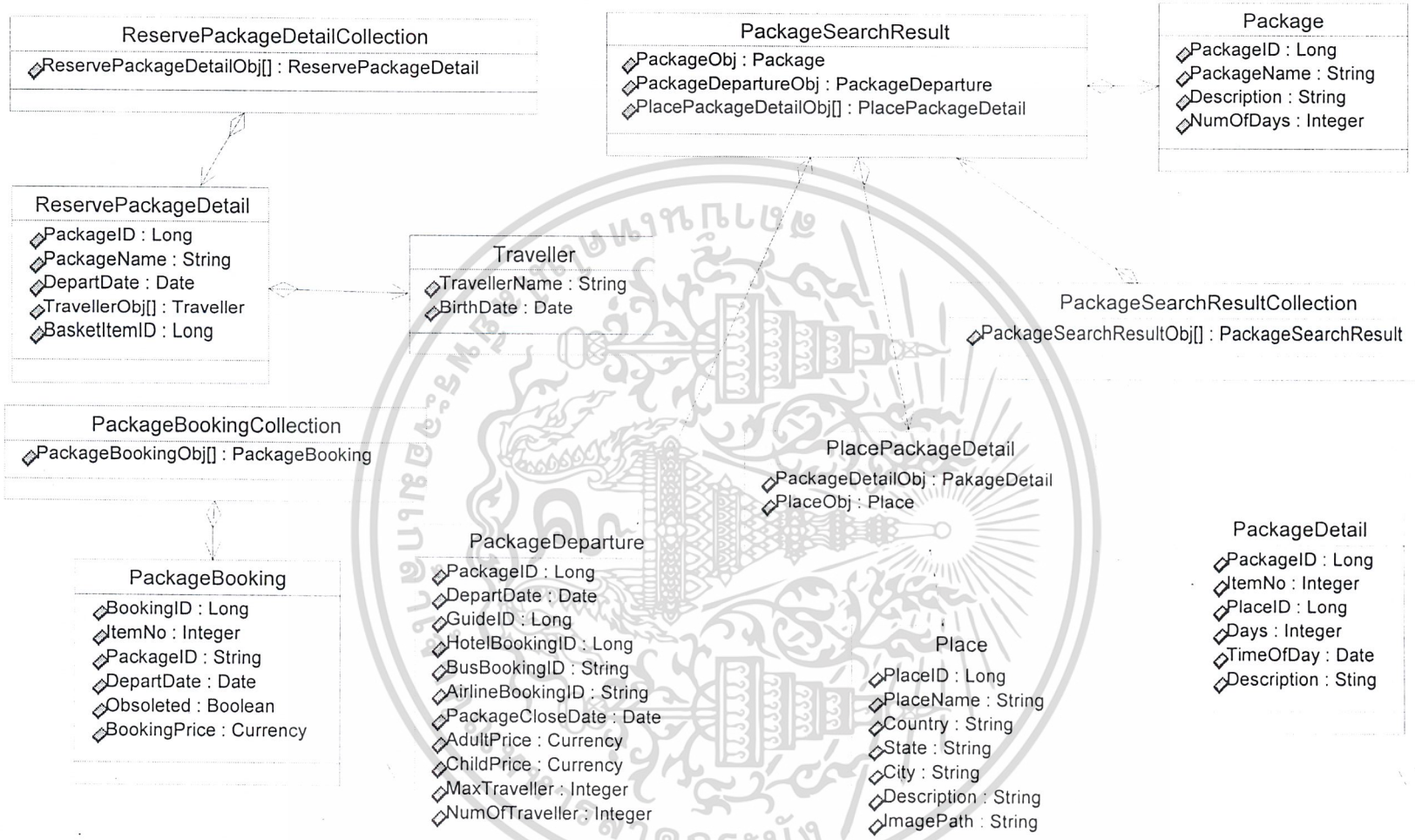


รูปที่ 11-14 แสดง Class Diagram – Data Object 1 ของระบบทัวร์

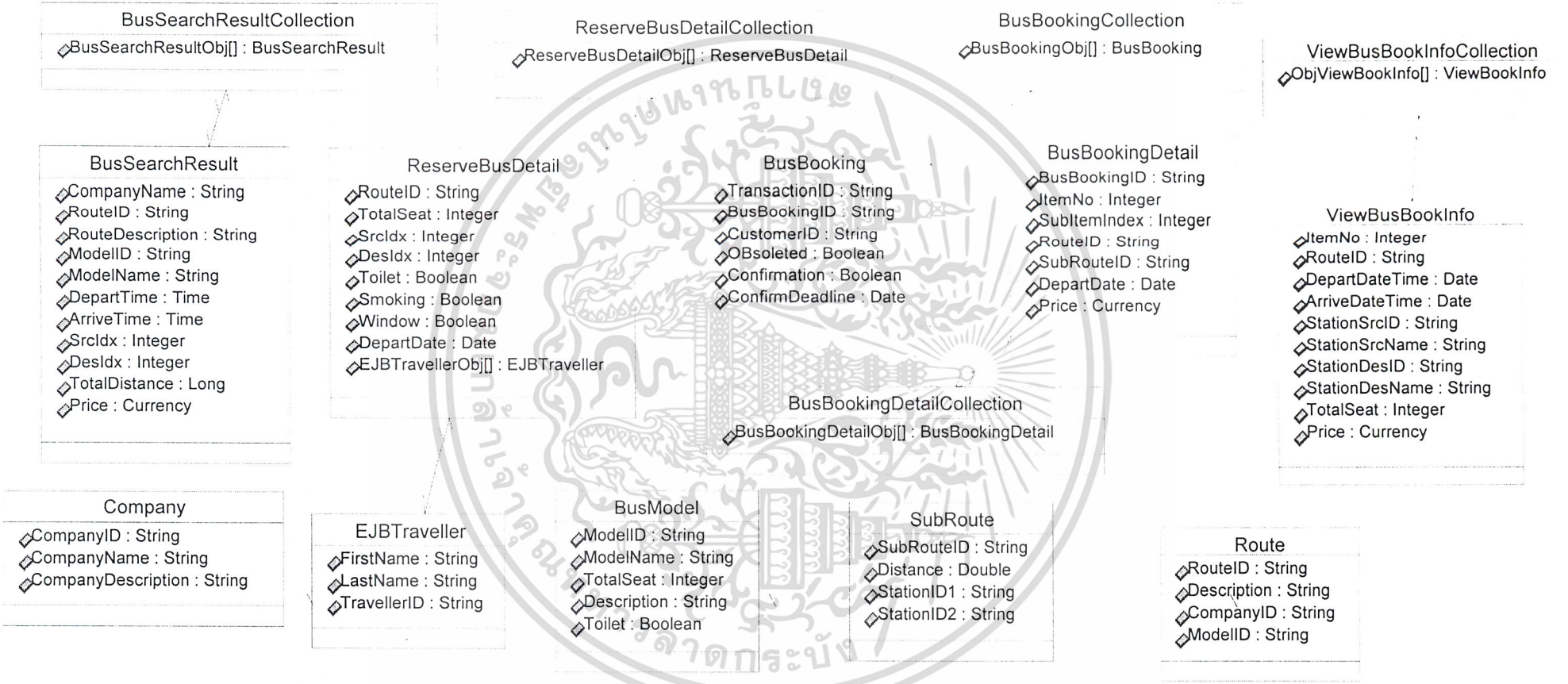
156



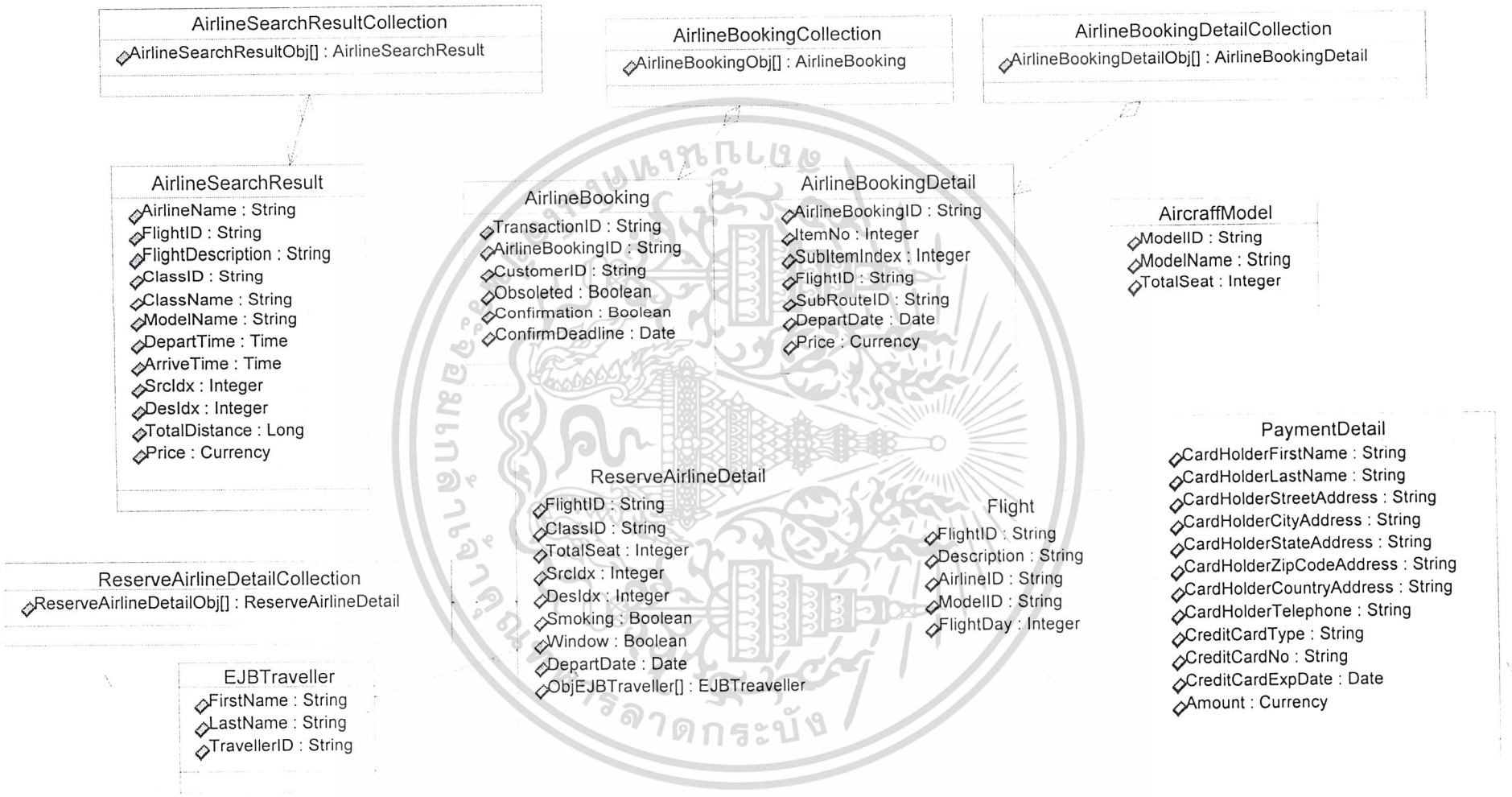
รูปที่11-15 แสดง Class Diagram – Data Object 2 ของระบบทัวร์



รูปที่ 11-16 แสดง Class Diagram – Data Object 3 ของระบบทัวร์



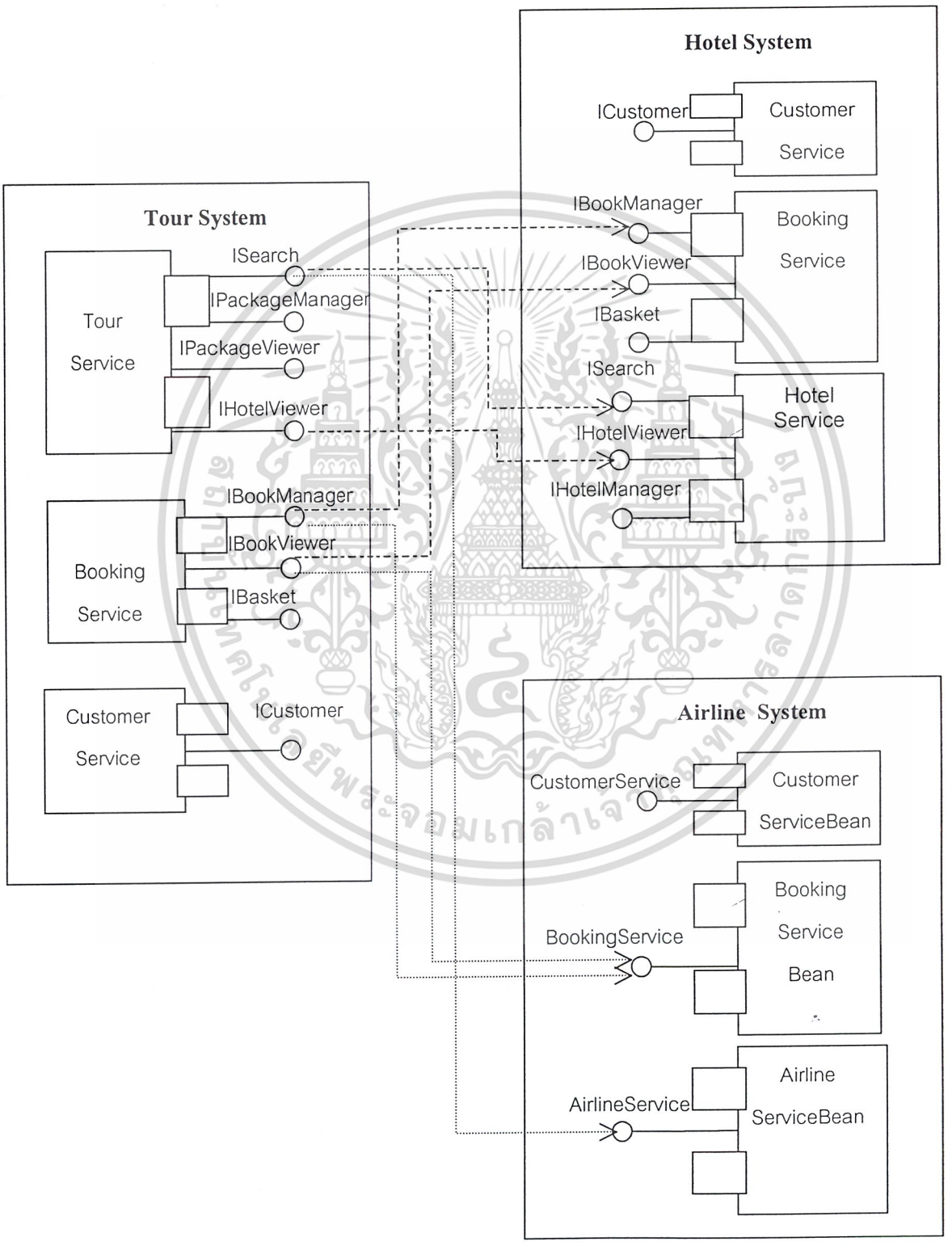
รูปที่ 11-17 แสดง Class Diagram – Data Object 4 ของระบบทัวร์



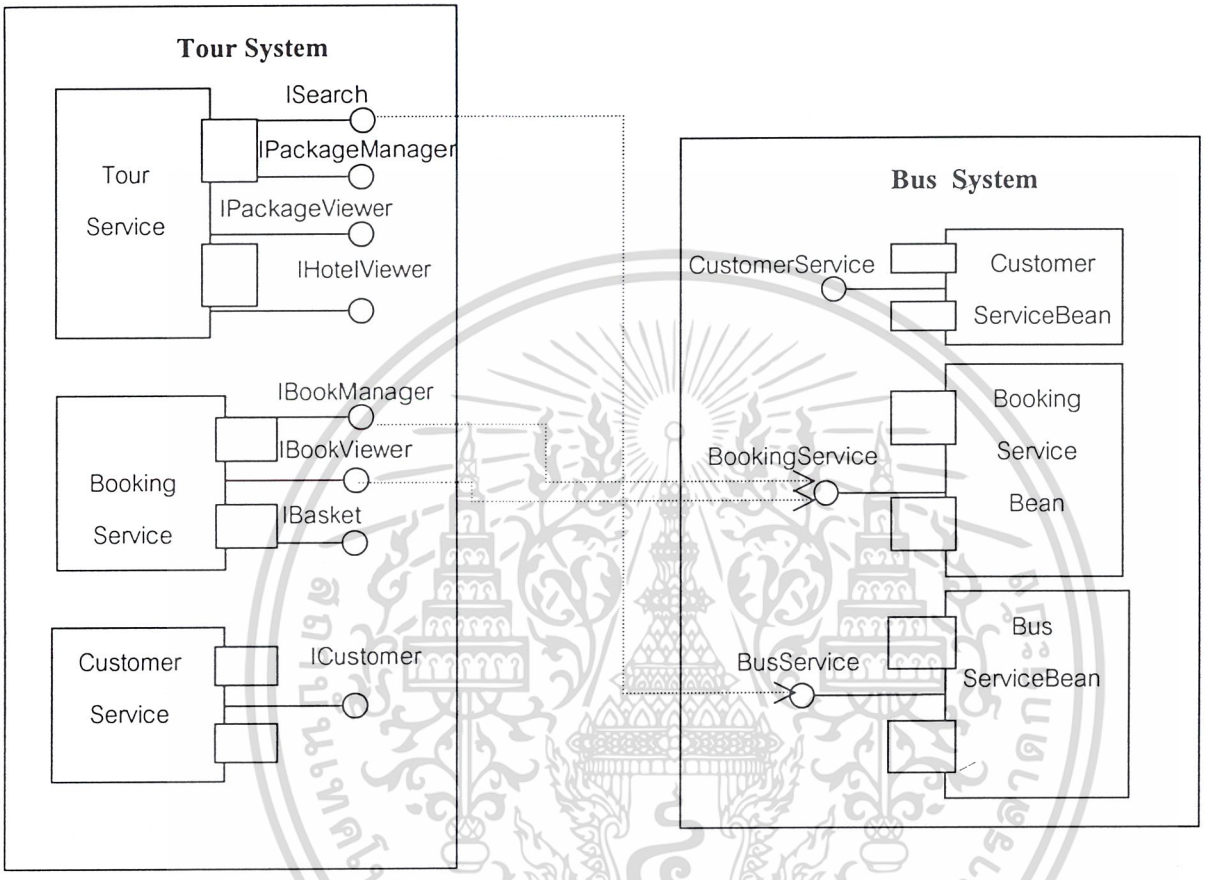
รูปที่ 11-18แสดง Class Diagram – Data Object 5 ของระบบทัวร์

11.8.4 คอมโนเนนต์ไดอะแกรม

การเรียกใช้คอมโพเนนต์ข้ามระบบ



รูปที่ 11-19 แสดง Component Diagram 1 ของระบบทัวร์



รูปที่ 11-20 แสดง Component Diagram 2 ของระบบทัวร์

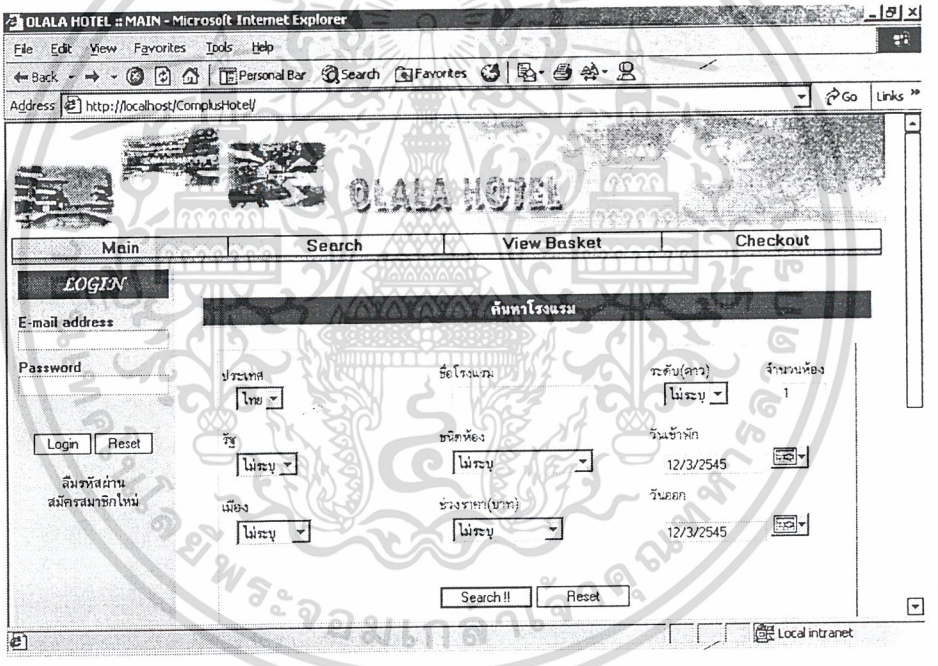
บทที่ 12
ลักษณะแอพพลิเคชัน

12.1 ระบบงาน Olala Hotel บนอินเทอร์เน็ต

ระบบงาน Olala Hotel จะให้บริการในการจองห้องพัก โดยจะให้บริการผ่านทางระบบอินเทอร์เน็ต ลูกค้าของระบบสามารถจองห้องพักได้ โดยเริ่มจากค้นหาห้องพักที่ต้องการ แล้วใส่รายการที่ต้องการในตะกร้า เมื่อได้ครบตามที่ต้องการแล้วก็ทำการจองห้องพัก ที่การจองนี้ ลูกค้ายังไม่ต้องจ่ายเงินให้กับระบบ แต่จะจ่ายเมื่อทำการยืนยันการจองแล้ว โดยลูกค้าที่จองห้องพักแต่ยังไม่ยืนยันการจองจะสามารถแก้ไขข้อมูลการจองห้องพักได้ แต่ถ้าทำการยืนยันการจองแล้ว จะไม่สามารถแก้ไขได้

จากความต้องการของระบบโรงแรม ทำให้ได้งานแอพพลิเคชันออกมา ดังต่อไปนี้

12.1.1 หน้าจอหลัก

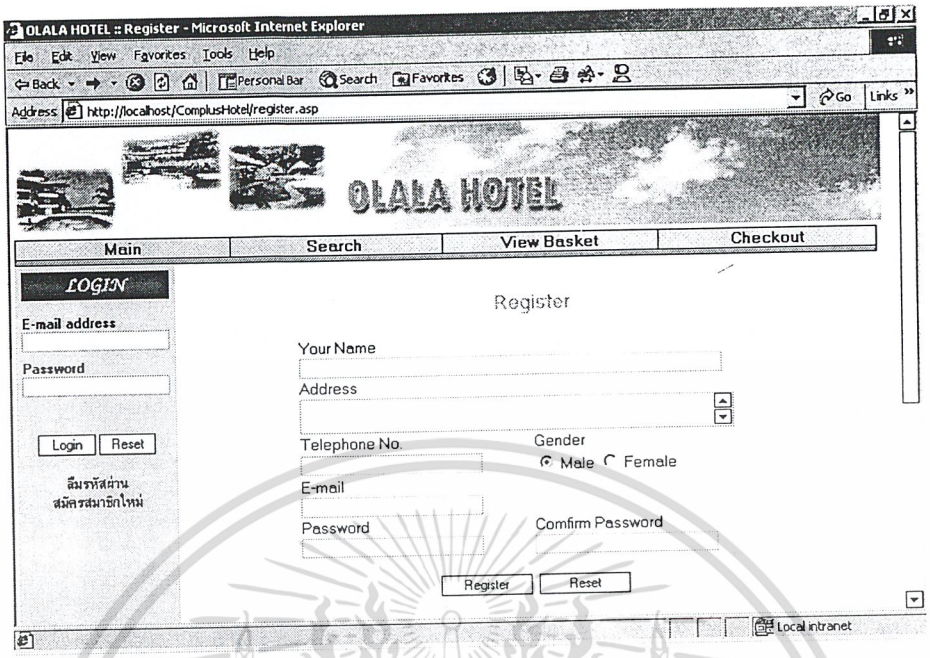


รูปที่ 12-1 แสดงหน้าจอหลักของระบบโรงแรม

ที่หน้าจอหลักจะมีบริการต่าง ๆ คือ

- การสมัครสมาชิกใหม่
- การล็อกอินเข้าระบบ
- การแก้ไขข้อมูลของลูกค้า
- ส่งอีเมล์ไปให้ลูกค้า ในกรณีที่ลืมรหัสผ่าน
- การค้นหาห้องพักที่ต้องการ
- การดูรายการจอง ทั้งที่ยืนยันแล้ว และยังไม่ได้ยืนยัน
- การดูรายการจองในตะกร้า

12.1.2 การสมัครเป็นสมาชิก

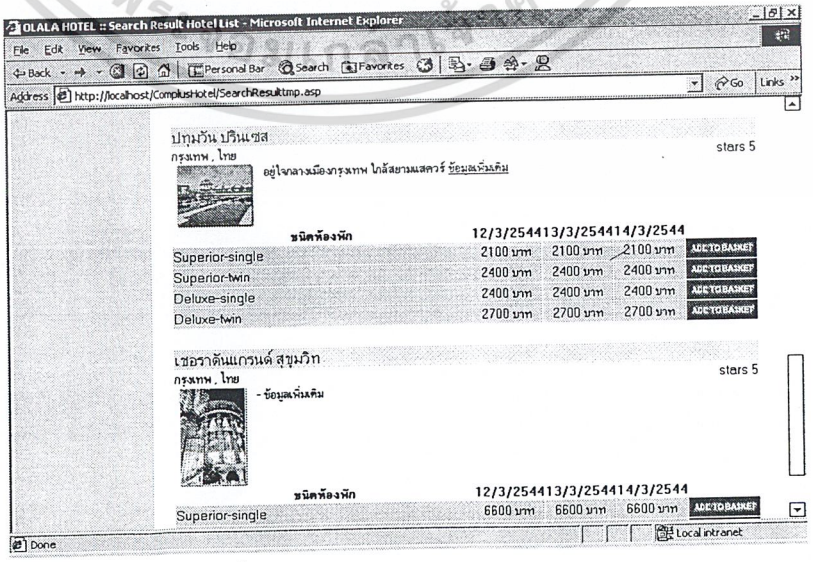


รูปที่ 12-2 แสดงการสมัครเป็นสมาชิกของระบบโรงแรม

การที่ลูกค้าจะจองห้องพักได้นั้น ลูกค้าต้องเป็นสมาชิกของระบบก่อน โดยกรอกข้อมูลดังรูปที่ 12-2 เพื่อสมัครเป็นสมาชิก จากนั้นลูกค้าจะใช้อีเมล์ที่ระบุนี้ในการเข้าใช้บริการในครั้งต่อ ๆ ไป แต่ในกรณีที่ลูกค้าต้องการค้นหาห้องพัก แล้วใส่รายการห้องพักลงตะกร้าสินค้านั้น ยังไม่ต้องเป็นสมาชิกของระบบก็สามารถทำได้ แต่ถ้าต้องการจองรายการห้องพักในตะกร้า จะต้องล็อกอินกับระบบก่อน

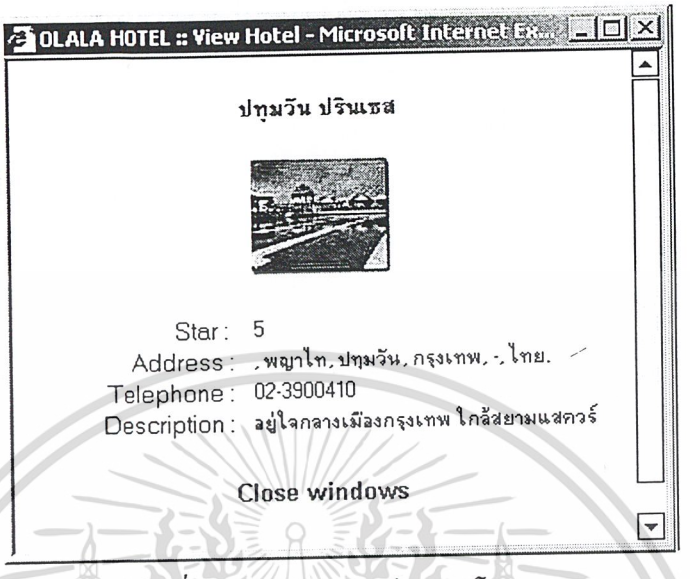
12.1.3 การค้นหาห้องพักและเลือกสินค้าใส่ตะกร้า

ลูกค้าสามารถค้นหาห้องพักได้ตั้งแต่หน้าแรกของเว็บดังรูปที่ 12-1 จากนั้นระบบจะแสดงผลลัพธ์จากการค้นหา ดังรูปที่ 12-3



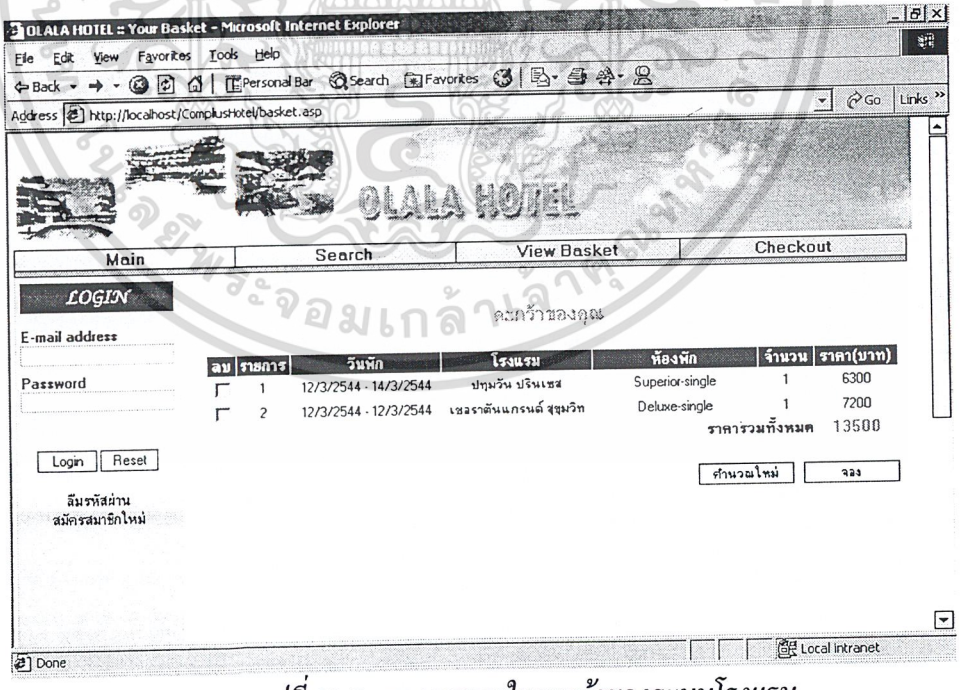
รูปที่ 12-3 แสดงผลลัพธ์จากการค้นหาโรงแรม

จากหน้าเว็บที่แสดงผลจากการค้นหาห้องพัก ลูกค้าสามารถดูรายละเอียดของโรงแรม หรือรายละเอียดของห้องพักในแต่ละโรงแรมได้โดยคลิ้กที่ “ข้อมูลเพิ่มเติม” หรือชื่อของห้องพักของโรงแรมนั้น ๆ จากนั้นจะมีหน้าต่างขึ้นมาแสดงรายละเอียดของโรงแรม ดังรูปที่ 12-4



รูปที่ 12-4 แสดงรายละเอียดของโรงแรม

นอกจากนี้แล้วที่หน้าผลลัพธ์จากการค้นหา ลูกค้าสามารถนำรายการห้องพักมาใส่ในตะกร้าได้โดยคลิ้กที่ “add to cart” จากนั้นจะปรากฏหน้าแสดงรายการในตะกร้า ดังรูปที่ 12-5

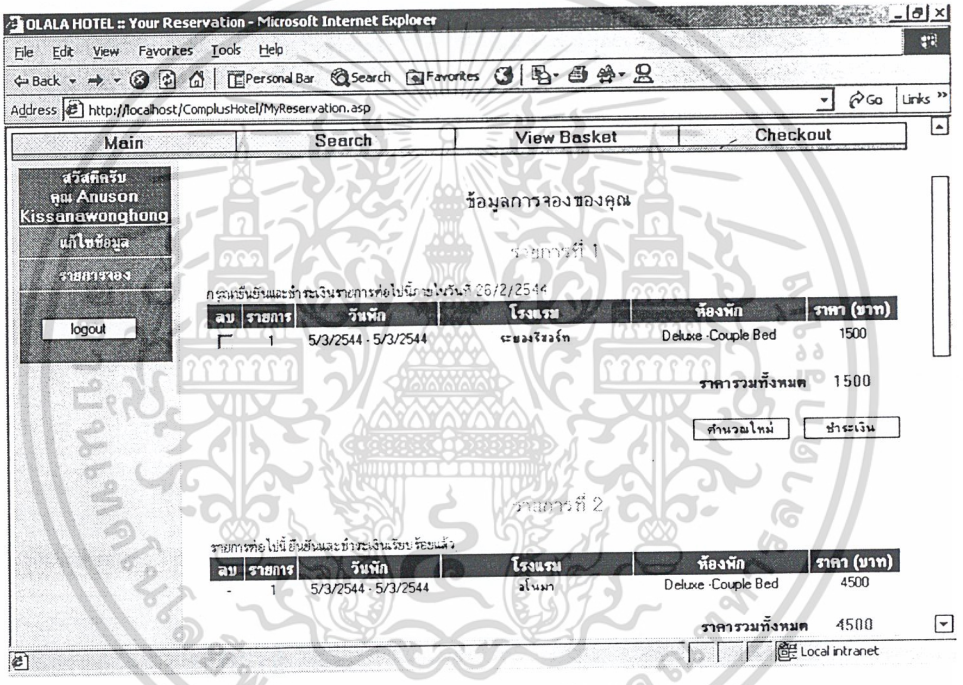


รูปที่ 12-5 แสดงรายการในตะกร้าของระบบโรงแรม

ที่หน้าแสดงรายการห้องพักในตะกร้าลูกค้าสามารถยกเลิก หรือทำการจองรายการห้องพักได้ โดยการยกเลิกนั้นจะยกเลิกเป็นบางรายการหรือยกเลิกทั้งหมดก็ได้ แต่การจองนั้นจะเป็นการจองทุกรายการที่อยู่ในตะกร้านั้น ๆ

12.1.4 การจอง และการยืนยันการจองห้องพักของระบบโรงแรม

ในการจองและยืนยันการจองห้องพักนั้น ลูกค้าจะต้องล็อกอินเข้าสู่ระบบก่อน ถ้ายังไม่ล็อกอินและทำการจองหรือยืนยันการจอง ทางระบบจะแจ้งเตือนให้ล็อกอินก่อน เมื่อล็อกอิน และทำการจองห้องพักแล้ว ทางระบบจะแสดงผลการจอง และแจ้งให้ลูกค้าทราบว่าต้องทำการยืนยันการจองภายในวันที่เท่าไหร่ แต่ถ้าลูกค้าต้องการยืนยันการจองเลยก็สามารถทำได้โดย คลิ๊กที่ "รายการจอง" จากนั้นระบบจะโชว์รายการจองของลูกค้านั้น ๆ ทั้งรายการที่ยืนยันการจองแล้ว และที่ยังไม่ได้ทำการยืนยัน ดังรูปที่ 12-6



รูปที่ 12-6 แสดงรายการจองของลูกค้าในระบบโรงแรม

ที่หน้านี้ ถ้าลูกค้าสามารถยกเลิกรายการจองได้ ถ้ารายการนั้น ๆ ยังไม่ได้ทำการยืนยันการจอง โดยเลือกเช็กบอกซ์รายการที่ต้องการยกเลิก แล้วเลือกปุ่มคำนวณใหม่ หรือถ้าต้องการยืนยันการจองก็เลือกปุ่มชำระเงิน จากนั้นลูกค้าจะกรอกข้อมูลบัตรเครดิตเพื่อทำการโอนเงินให้กับระบบ เมื่อโอนเงินสำเร็จแล้ว ระบบจะแสดงรายการห้องพัก พร้อมเลขห้องของโรงแรมนั้น ๆ ในการเข้าพัก

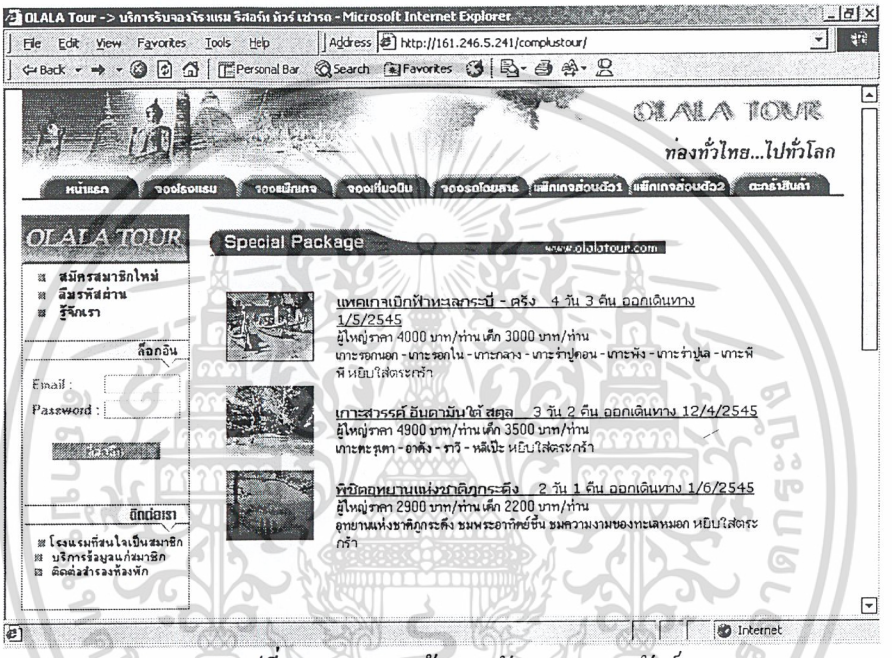
12.2 ระบบงาน Olala Tour บนอินเทอร์เน็ต

ระบบงาน Olala Tour จะให้บริการในการจองแพ็กเกจทัวร์, จองห้องพัก , จองตั๋วรถ และจองตั๋วเครื่องบิน โดยระบบทัวร์นี้จะแสดงให้เห็นถึงแอพพลิเคชันที่มีการเรียกใช้ข้ามระบบ ไม่ว่าจะเป็นการเรียกใช้บริการข้ามระบบแต่อยู่บนแพลตฟอร์มเดียวกัน คือ ระบบทัวร์จะไปเรียกใช้บริการของระบบโรงแรม ซึ่งอิมพลีเมนต์อยู่บนแพลตฟอร์มของวินโดวส์ดีเอ็นเอ หรือ การเรียกข้ามระบบที่อยู่คนละแพลตฟอร์มกัน

คือ ระบบทัวร์ ซึ่งอิมพลีเมนต์อยู่บนแพลตฟอร์มของวินโดวส์ดีเอ็นเอ จะไปเรียกใช้บริการของระบบรถโดยสาร และระบบสายการบิน ซึ่งอิมพลีเมนต์อยู่บนแพลตฟอร์มของจาวา

ลูกค้าของระบบทัวร์สามารถเรียกใช้บริการในการจองแพ็กเกจทัวร์ , ห้องพัก , รถโดยสาร และตั๋วเครื่องบินได้เสมือนกับว่าบริการต่าง ๆ นั้นเป็นบริการของระบบทัวร์เอง โดยลูกค้าสามารถลงทะเบียนเป็นสมาชิกของระบบทัวร์ , ค้นหาแพ็กเกจทัวร์ , ห้องพัก ,ตั๋วรถโดยสาร และตั๋วเครื่องบิน ใส่รายการที่ต้องการลงในตะกร้าและทำการจอง หรือยกเลิกการจองในรายการต่าง ๆ ได้

12.2.1 หน้าจอหลักของระบบแพ็กเกจทัวร์



รูปที่ 12-7 แสดงหน้าจอหลักของระบบทัวร์

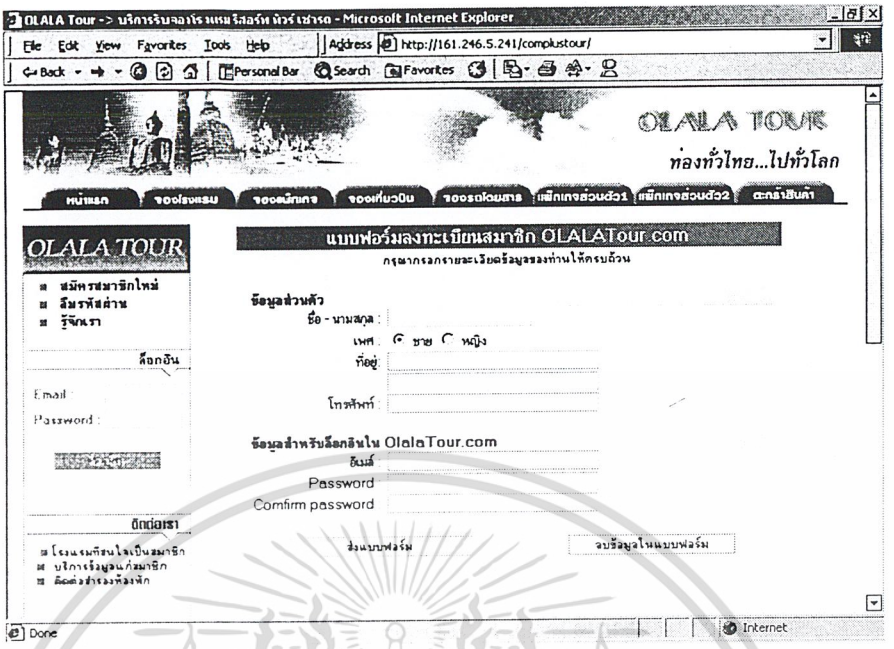
ที่หน้าจอหลักนี้จะมีบริการต่าง ๆ ดังต่อไปนี้

- การสมัครเป็นสมาชิกใหม่ (ลงทะเบียนเป็นสมาชิกของระบบ)
- การล็อกอินเพื่อใช้บริการ
- การส่งอีเมล์ให้ลูกค้าในกรณีที่ลืมรหัสผ่าน
- การค้นหาแพ็กเกจทัวร์
- การค้นหาห้องพักของโรงแรม
- การค้นหาเที่ยวรถโดยสาร
- การค้นหาเที่ยวสายการบิน
- การดูรายการในตะกร้า

ในกรณีที่ลูกค้าทำการล็อกอินแล้วจะมีบริการเพิ่ม คือ

- แก้ไขข้อมูลที่เคยลงทะเบียนไว้
- ดูข้อมูลที่เคยลงทะเบียนไว้
- ดูรายการจองทั้งหมด

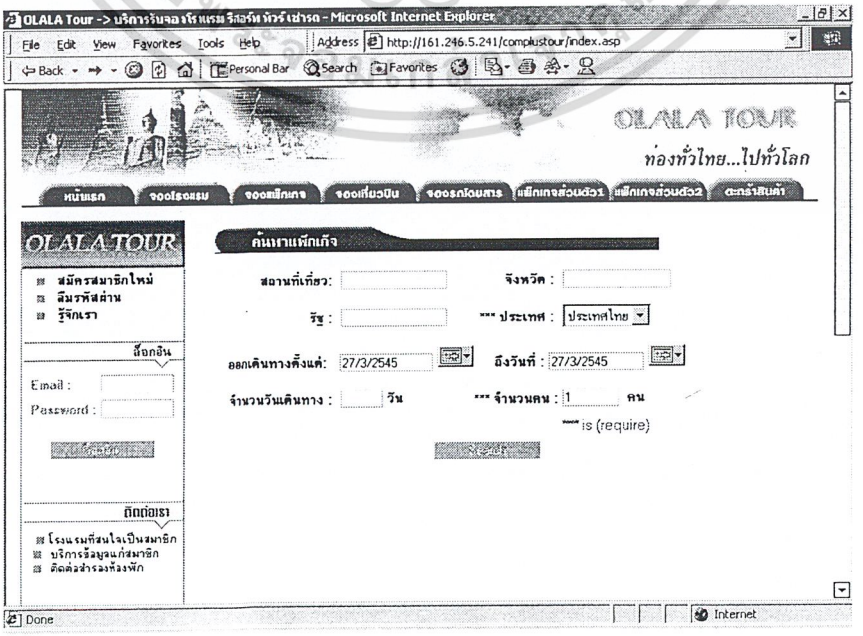
12.2.2 การลงทะเบียนเป็นสมาชิกของระบบทัวร์



รูปที่ 12-8 แสดงหน้าจอการลงทะเบียนของระบบทัวร์

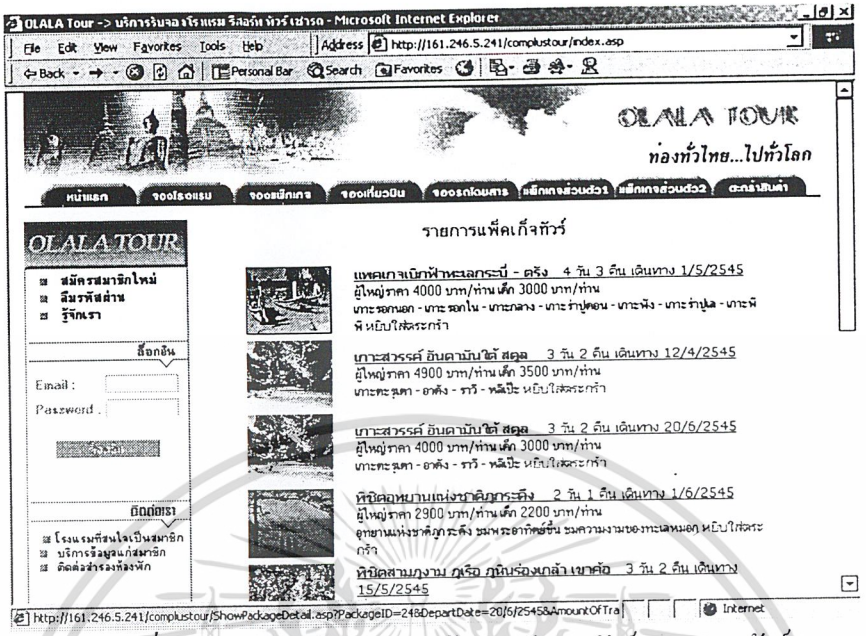
เช่นเดียวกันกับระบบโรงแรม การที่ลูกค้าจะจองแพ็กเกจทัวร์, ห้องพัก , ตั๋วรถโดยสาร หรือ ตั๋วเครื่องบินได้นั้น ลูกค้าต้องลงทะเบียนเป็นสมาชิกกับระบบก่อน โดยกรอกข้อมูลดังรูปที่ 12-8 เพื่อสมัครเป็นสมาชิก จากนั้นลูกค้าจะใช้อีเมล์ที่ระบุนี้ในการเข้าใช้บริการในครั้งต่อ ๆ ไป แต่ในกรณีที่ลูกค้าต้องการค้นหาแพ็กเกจทัวร์, ห้องพัก , ตั๋วรถโดยสาร หรือ ตั๋วเครื่องบิน แล้วใส่รายการที่ต้องการลงตะกร้าสินค้านั้น ยังไม่ต้องเป็นสมาชิกของระบบก็สามารถทำได้ แต่ถ้าต้องการจองรายการที่อยู่ในตะกร้า จะต้องล็อกอินกับระบบก่อน

12.2.3 การค้นหาแพ็กเกจทัวร์



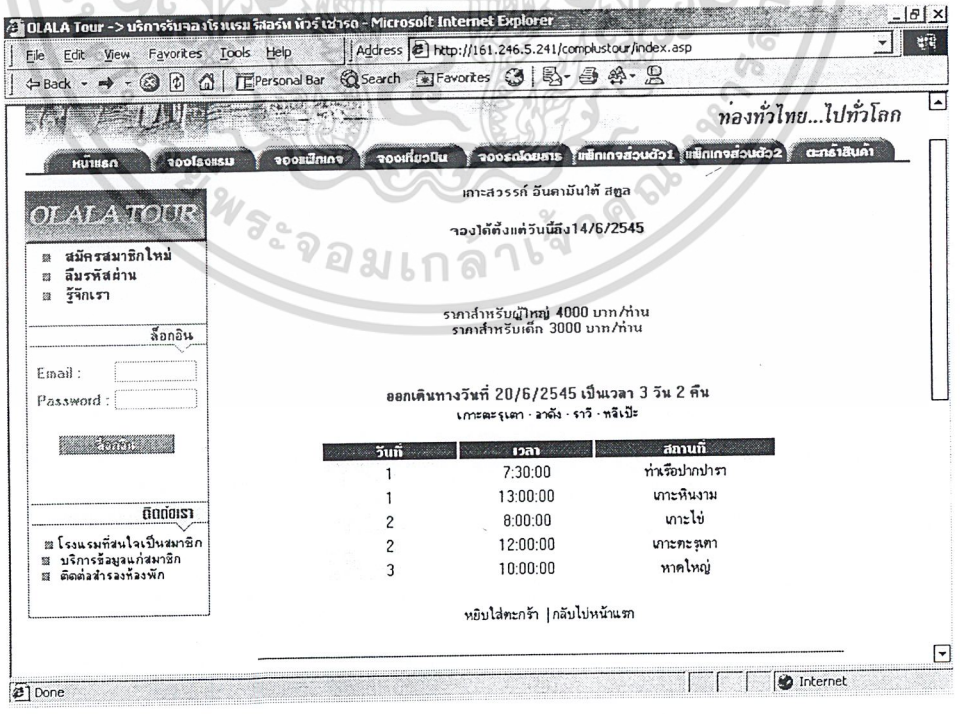
รูปที่ 12-9 แสดงหน้าจอการค้นหาแพ็กเกจทัวร์ของระบบทัวร์

เมื่อลูกค้ากรอกข้อมูลที่ต้องการค้นหาแล้วกดปุ่ม Search ระบบจะโชว์ผลลัพธ์จากการค้นหาตามรูปที่ 12-10



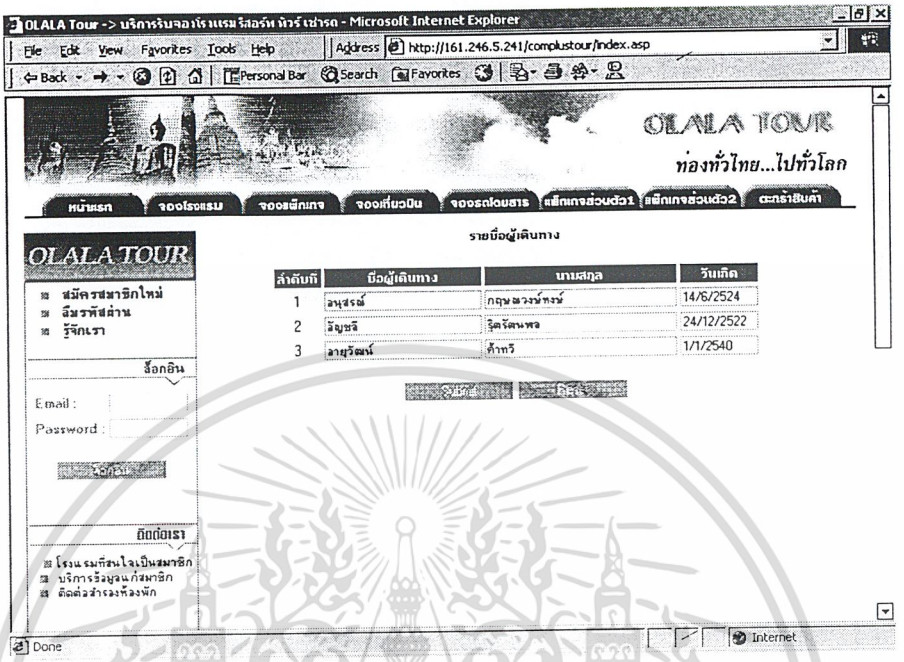
รูปที่ 12-10 แสดงหน้าจอผลการค้นหาแพ็กเกจทัวร์ของระบบทัวร์

ที่หน้าจอผลลัพธ์จากการค้นหาแพ็กเกจทัวร์นี้ ลูกค้าสามารถหยิบใส่ตะกร้า หรือ เลือกดูรายละเอียดของแต่ละแพ็คเกจได้ โดยถ้าต้องการเลือกดูรายละเอียดของแพ็กเกจทัวร์ก็ให้คลิ๊กที่ ชื่อแพ็กเกจ จากนั้นจะปรากฏหน้าต่างแสดงรายละเอียดของแพ็กเกจทัวร์ขึ้นมา ดังรูปที่ 12-11



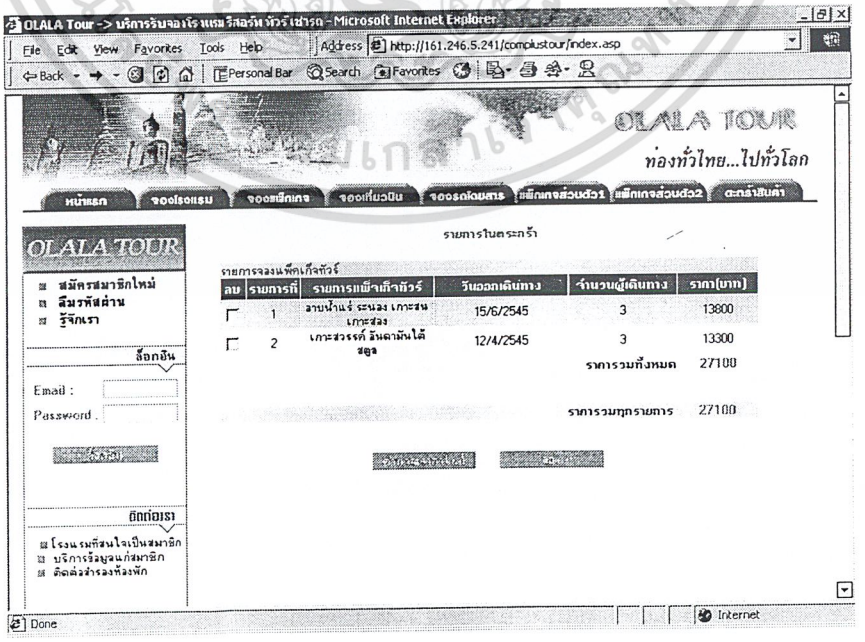
รูปที่ 12-11 แสดงหน้าจอแสดงรายละเอียดของแพ็คเกจทัวร์

ที่หน้าจอดังรูปที่ 12-11 ก็สามารถหยิบแพ็กเกจทัวร์นี้ใส่ตะกร้าได้เช่นเดียวกันกับหน้าจอดังรูปที่ 12-10 เมื่อเลือกหยิบใส่ตะกร้าแล้ว ระบบก็จะให้ใส่รายชื่อผู้เดินทาง พร้อมทั้งวันเกิด ดังแสดงในรูปที่ 12-12



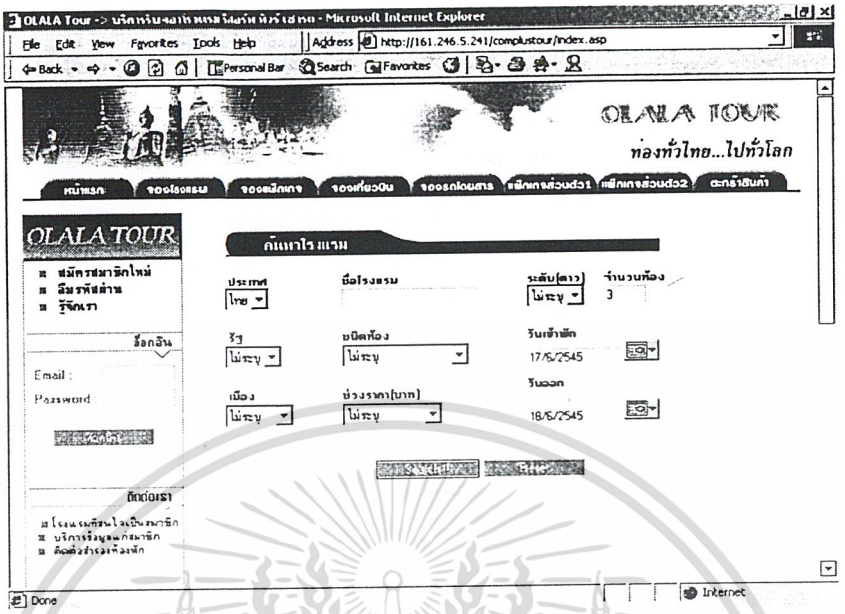
รูปที่ 12-12 แสดงหน้าจอให้ใส่ชื่อผู้เดินทาง

โดยจะขึ้นเท็กซ์บอกซ์ให้ใส่ชื่อผู้เดินทางตามจำนวนผู้เดินทางที่ใส่ไว้ในหน้าค้นหาแพ็กเกจทัวร์ หลังจากใส่ชื่อผู้เดินทางแล้ว เมื่อกดปุ่มซับมิต ระบบก็จะโชว์รายการที่อยู่ในตะกร้าทั้งหมด ดังรูปที่ 12-13



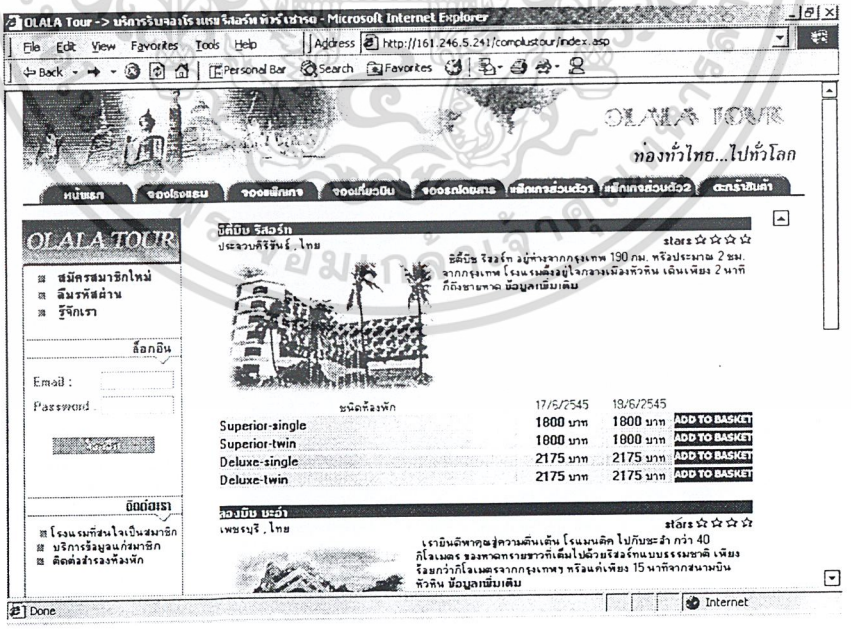
รูปที่ 12-13 แสดงหน้าจอโชว์รายการในตะกร้าของระบบทัวร์

12.2.4 การค้นหาโรงแรมของระบบทัวร์



รูปที่ 12-14 แสดงหน้าจอการค้นหาห้องพักของระบบทัวร์

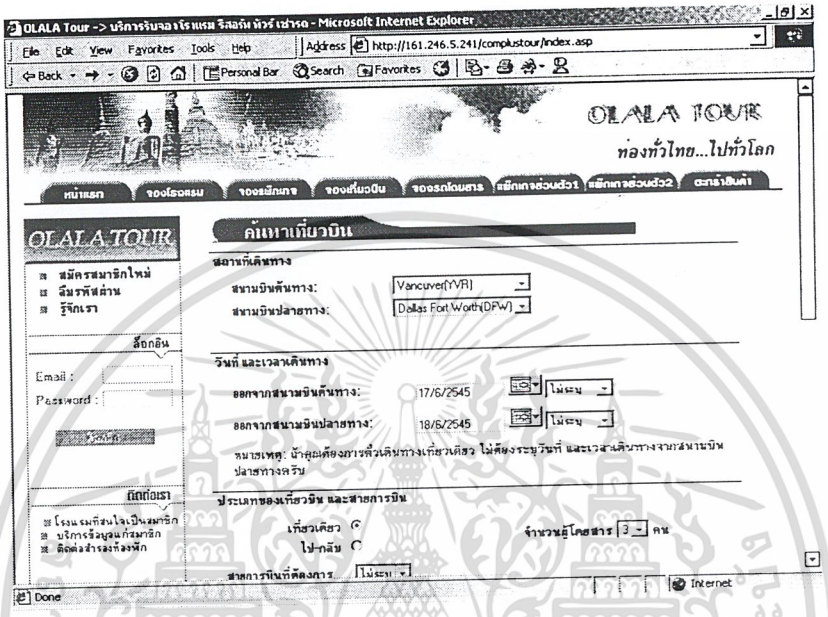
เมื่อลูกค้ากรอกข้อมูลที่ต้องการค้นหาแล้วกดปุ่ม Search ระบบจะแสดงผลลัพธ์จากการค้นหาตามรูปที่ 12-15



รูปที่ 12-15 แสดงหน้าจอผลลัพธ์การค้นหาโรงแรมของระบบทัวร์

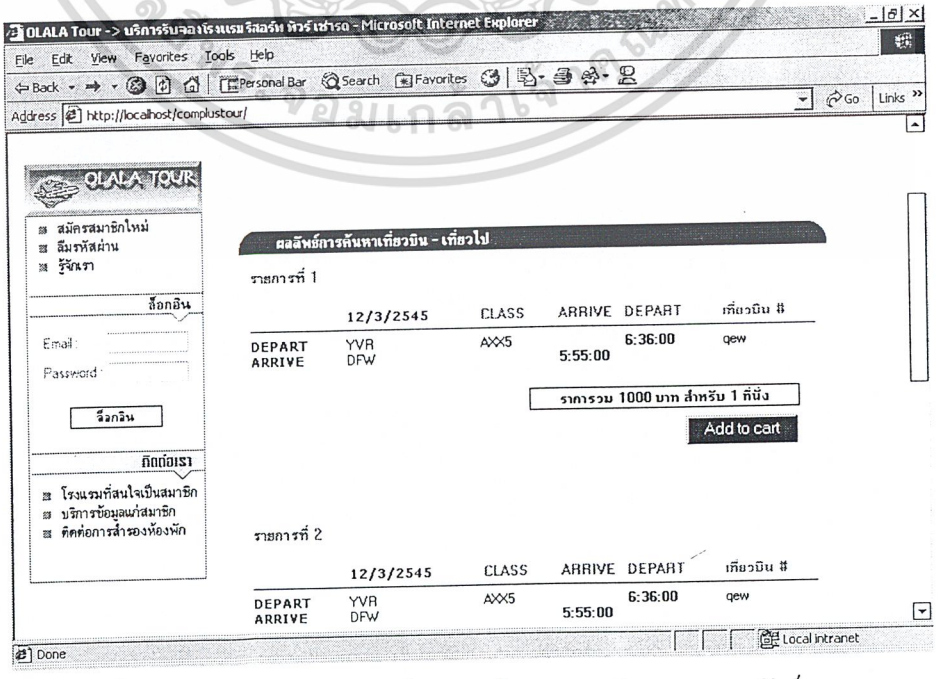
12.2.5 การค้นหาสายการบิน และรถโดยสารของระบบทัวร์

การค้นหาสายการบิน และรถโดยสารของระบบทัวร์นี้ไม่ได้มีการอิมพลีเมนต์ที่ระบบทัวร์โดยตรง แต่ระบบทัวร์จะไปเรียกใช้บริการของระบบสายการบิน และระบบรถโดยสารอีกต่อหนึ่ง ในที่นี้จะยกตัวอย่างให้เห็นในการค้นหาสายการบิน ส่วนการค้นหารถโดยสารนั้นก็จะเหมือนกับสายการบิน



รูปที่ 12-16 แสดงหน้าจอการค้นหาสายการบินของระบบทัวร์

เมื่อลูกค้ากรอกข้อมูลที่ต้องการค้นหาแล้วกดปุ่ม Search ระบบจะแสดงผลลัพธ์จากการค้นหาตามรูปที่ 12-17

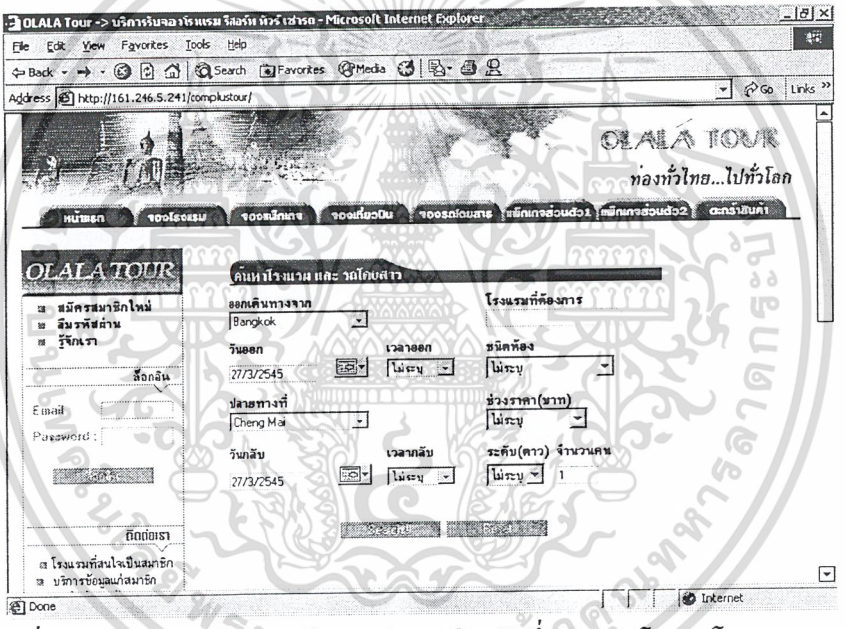


รูปที่ 12-17 แสดงหน้าจอผลลัพธ์จากการค้นสายการบินของระบบทัวร์

หลังจากเลือกหยิบใส่ตะกร้าแล้วจะแสดงหน้าจอให้ใส่ชื่อผู้เดินทางเช่นเดียวกันกับการหยิบใส่ตะกร้าของแพ็กเกจทัวร์ เมื่อใส่รายชื่อผู้เดินทางแล้ว ระบบจะแสดงรายการที่อยู่ในตะกร้าทั้งหมด ซึ่งรายการเหล่านี้ ไม่ว่าจะเป็นแพ็กเกจทัวร์ , ห้องพักโรงแรม , ตั๋วรถโดยสาร และตั๋วสายการบิน จะใส่ลงในตะกร้าใบเดียวกัน และถ้าทำการจองรายการเหล่านี้ก็จะเป็นการจองทุกรายการที่อยู่ในตะกร้านั้น ๆ แต่ถ้าจองรายการใดรายการหนึ่งไม่ได้ ระบบก็จะยกเลิกการจองนั้นทั้งหมด

12.2.6 การสร้างแพ็กเกจส่วนตัวเดินทางโดยรถโดยสาร

ลูกค้าสามารถสร้างแพ็กเกจแบบเดินทางด้วยรถโดยสารตามที่ต้องการได้ โดยระบุต้นทาง , ปลายทาง , จำนวนผู้เดินทาง , ชนิดห้องของโรงแรมที่ต้องการพัก , วันที่ออกเดินทางไป – กลับ ดังแสดงในรูปที่ 12-18

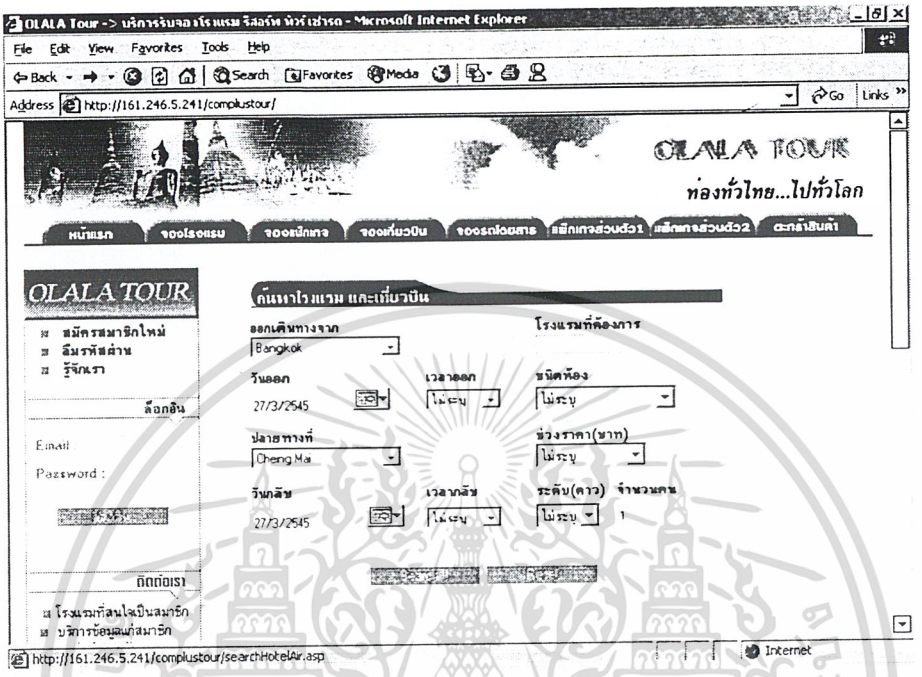


รูปที่ 12-18 แสดงหน้าจอการค้นหาแพ็กเกจส่วนตัวซึ่งเดินทางโดยรถโดยสาร

โดยทางระบบจะค้นหาโรงแรมและเที่ยวรถทั้งไป และกลับ ที่ตรงกับเงื่อนไขให้ แล้วแสดงผลให้ทราบ

12.2.7 การสร้างแพ็กเกจส่วนตัวเดินทางโดยเครื่องบิน

ลูกค้าสามารถสร้างแพ็กเกจแบบเดินทางด้วยเครื่องบินตามที่ต้องการได้ โดยระบุต้นทาง , ปลายทาง , จำนวนผู้เดินทาง , ชนิดห้องของโรงแรมที่ต้องการพัก , วันที่ออกเดินทางไป – กลับ ดังแสดงในรูปที่ 12-19



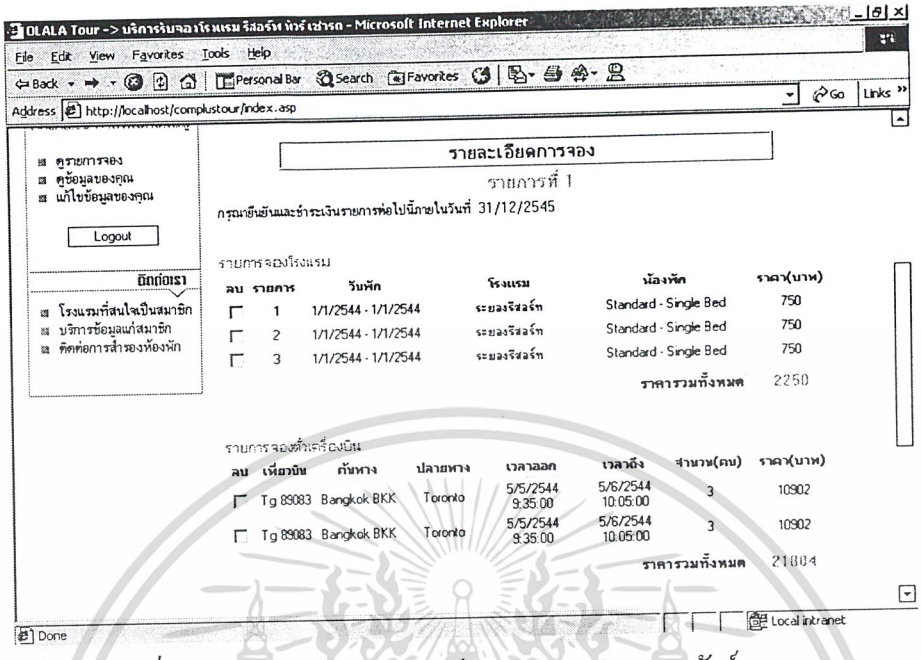
รูปที่ 12-19 แสดงหน้าจอการค้นหาแพ็กเกจส่วนตัวซึ่งเดินทางโดยเครื่องบิน

โดยทางระบบจะค้นหาโรงแรมและเที่ยวรถทั้งไป และกลับ ที่ตรงกับเงื่อนไขให้ แล้วแสดงผลให้ทราบ

12.2.8 การจอง และการยืนยันการจอง

ในการจองและยืนยันการจองห้องพักนั้น ลูกค้าจะต้องล็อกอินเข้าสู่ระบบก่อน ถ้ายังไม่ล็อกอินและทำการจองหรือยืนยันการจอง ทางระบบจะแจ้งเตือนให้ล็อกอินก่อน เมื่อล็อกอิน และทำการจองแล้ว ทางระบบจะแสดงผลการจอง และแจ้งให้ลูกค้าทราบว่าต้องทำการยืนยันการจองภายในวันที่เท่าไหร่ แต่ถ้าลูกค้าต้องการยืนยันการจองเลยก็สามารถทำได้โดย คลิ๊กที่ “รายการจอง” จากนั้นระบบจะโชว์รายการจองของลูกค้านั้น ๆ

ทั้งรายการที่ยืนยันการจองแล้ว และที่ยังไม่ได้ทำการยืนยัน ดังรูปที่ 12-20



รูปที่ 12-20 แสดงหน้าจอรายละเอียดการจองของระบบทัวร์

ที่หน้านี้ ลูกค้าสามารถยกเลิกรายการจองได้ ถ้ารายการนั้น ๆ ยังไม่ได้ทำการยืนยันการจอง โดยเลือกเช็กบอกซ์รายการที่ต้องการยกเลิก แล้วเลือกปุ่มคำนวณใหม่ หรือถ้าต้องการยืนยันการจองก็เลือกกดปุ่มชำระเงิน จากนั้นลูกค้าจะกรอกข้อมูลบัตรเครดิตเพื่อทำการโอนเงินให้กับระบบ เมื่อโอนเงินสำเร็จแล้ว ระบบจะให้หมายเลขกับลูกค้าไว้ใช้ในเดินทางไปในรายการที่ชำระเงินเรียบร้อยแล้ว

บทที่ 13
การจัดวางเครื่องเพื่อทดสอบการกระจายโหลด

13.1 การจัดวางเครื่อง

จากที่กล่าวมาในบทต้นๆ เกี่ยวกับการกระจายโหลดเพื่อเพิ่มประสิทธิภาพการทำงาน ในบทนี้จะกล่าวถึงเรื่องการจัดวางเครื่องคอมพิวเตอร์และปรับแต่งค่าต่างๆ เพื่อทดสอบการกระจายโหลด ซึ่งในการทดสอบนี้จะใช้เครื่องคอมพิวเตอร์ทั้งหมด 5 เครื่อง โดยใช้เครื่อง 2 เครื่องที่พรีเซ็นเตชั่นเทียร์เพื่อทำการกระจายโหลดการเรียกใช้เว็บเพจและใช้เครื่องอีก 2 เครื่องที่บิสิเนสโลจิกเทียร์เพื่อทำการกระจายโหลดการใช้งานคอมโพเน็นต์ ส่วนการทำคลัสเตอริ่งฐานข้อมูลที่ดาต้าเทียร์จะไม่ทำการทดสอบ เนื่องจากต้องการฮาร์ดแวร์พิเศษจำพวก RAID และ SCSI

เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบต้องมีคุณสมบัติดังนี้

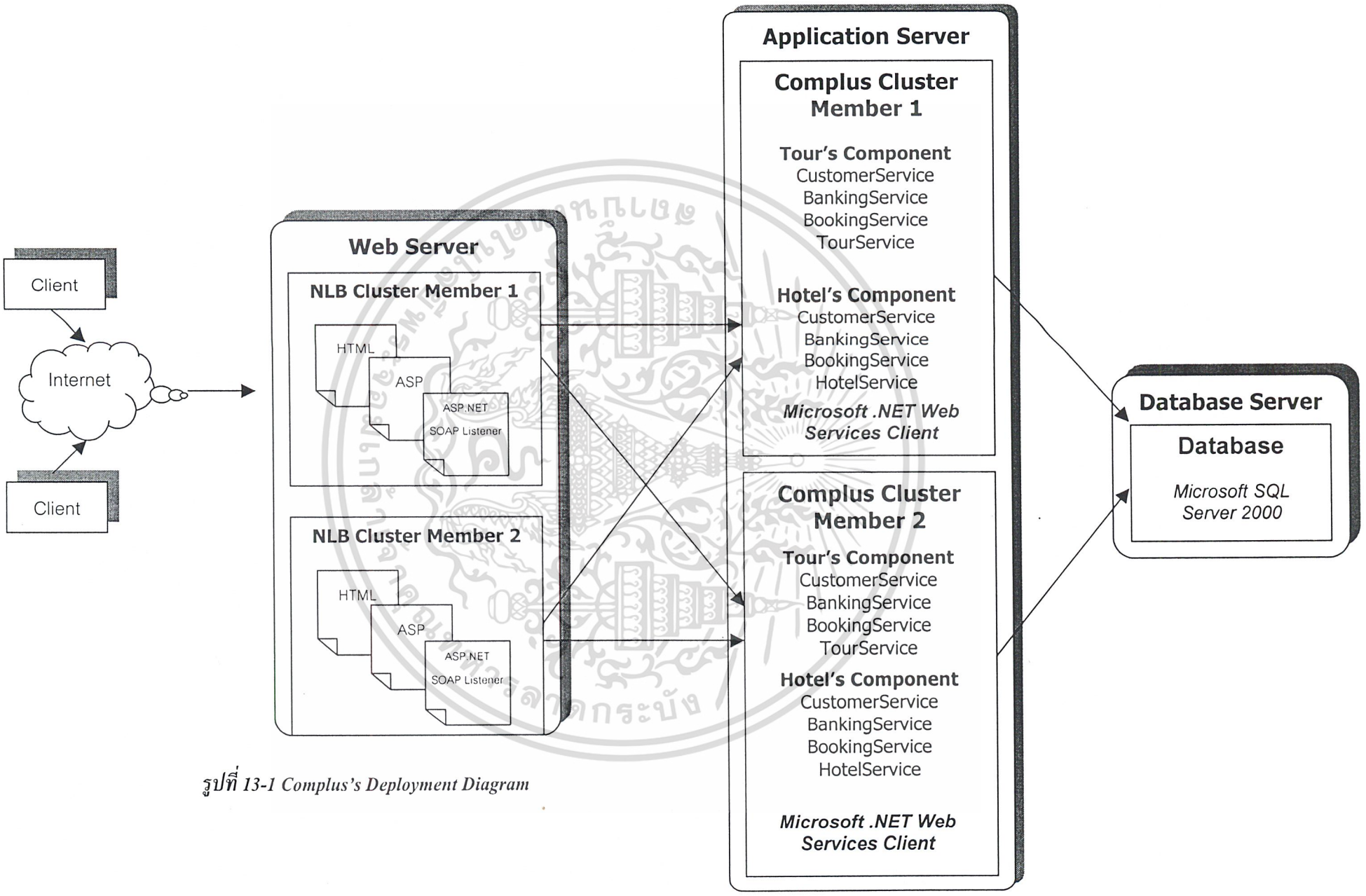
1. ทุกเครื่องที่ใช้ในการทดสอบต้องใช้ระบบปฏิบัติการ Microsoft Windows 2000 Advanced Server หรือ Microsoft Windows 2000 Datacenter

2. เครื่องคอมพิวเตอร์ใช้ในพรีเซ็นเตชั่นเทียร์ทั้ง 2 เครื่อง ต้องทำการสร้างเว็บเซิร์ฟเวอร์และนำไฟล์ ASP ที่ใช้ในระบบทัวร์และระบบโรงแรมมาไว้ จากนั้นนำไฟล์ ASP.NET ที่ใช้ SOAP ในการติดต่อกับระบบสายการบินและระบบรถโดยสารที่พัฒนาโดยฝ่าย EJB มาไว้ด้วย จากนั้นปรับแต่ง Network Load Balancing (NLB) เพื่อทำการกระจายโหลดของการเรียกใช้เว็บเพจ (จะกล่าวโดยละเอียดในภายหลัง)

3. เครื่องคอมพิวเตอร์ใช้ในบิสิเนสโลจิกเทียร์ทั้ง 2 เครื่อง ต้องติดตั้งโปรแกรม Microsoft Application Center 2000 ซึ่งใช้ในการสร้างคลัสเตอร์และจัดการการกระจายโหลดการใช้งานคอมโพเน็นต์ภายในคลัสเตอร์ เมื่อทำการติดตั้งเรียบร้อยแล้วให้ทำการสร้างคลัสเตอร์และเพิ่มเครื่องคอมพิวเตอร์ทั้ง 2 เครื่องให้เป็นสมาชิกของคลัสเตอร์ จากนี้ให้ทำการดีพลอยคอมโพเน็นต์ของระบบทัวร์และระบบโรงแรมทั้ง 2 เครื่อง (จะกล่าวโดยละเอียดในภายหลัง)

4. ที่ดาต้าเทียร์ เราจะติดตั้งโปรแกรม Microsoft SQL Server เพื่อสร้างฐานข้อมูลของระบบทัวร์และระบบโรงแรม

การจัดวางเครื่องดังที่กล่าวมาข้างต้น สามารถวาดได้ดังรูป



รูปที่ 13-1 Complus's Deployment Diagram

13.2 วิธีการปรับแต่ง Network Load Balancing ด้วย NLB ในวินโดวส์ 2000

ทางไมโครซอฟท์ได้สร้างทูลที่เรียกว่า Network Load Balancing (NLB) เพื่อช่วยให้สามารถกระจายโหลดของการเรียกใช้เว็บเพจได้ นอกจากนี้ทูลนี้แล้ว ไมโครซอฟท์ยังได้สร้างโปรแกรมที่เรียกว่า Microsoft Application Center 2000 ซึ่งสามารถใช้ในการกระจายโหลดของการเรียกใช้เว็บเพจได้เช่นกัน แต่ในที่นี้ จะทดสอบโดยใช้ NLB ส่วนโปรแกรม Microsoft Application Center 2000 จะทดสอบในภายหลัง

NLB เป็นคอมโพเน็นต์ตัวหนึ่งที่แถมมากับ Microsoft Window 2000 Advanced Server หรือ Datacenter ดังนั้นในการทดสอบนี้ จะติดตั้ง NLB ในเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการ Microsoft Window 2000 Advanced Server ทั้ง 2 เครื่อง นอกจากนี้ แต่ละเครื่องจะต้องมีการกำหนด IP ไว้คงที่ด้วย (ไม่ใช่ได้รับจาก DHCP) และยังต้องมีการกำหนดค่า Virtual IP เพื่อให้ไคลเอ็นต์ใช้งานคลัสเตอร์ได้โดยผ่าน IP เสมือนเพียง IP เดียว

สรุปค่าการปรับแต่งค่าแต่ละเครื่อง จะเป็นดังนี้

เครื่องที่ 1

ชื่อเครื่อง : OLALA03

ระบบปฏิบัติการ : Microsoft Windows 2000 Advanced Server

Virtual IP: 161.246.5.243

Primary IP Address : 161.246.5.216

เครื่องที่ 2

ชื่อเครื่อง : OLALA06

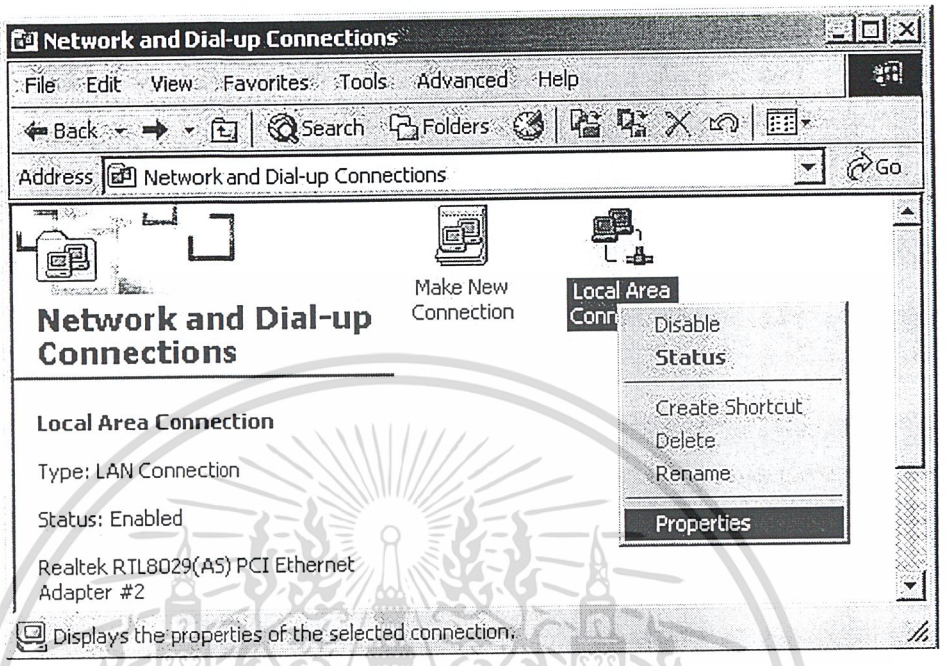
ระบบปฏิบัติการ : Microsoft Windows 2000 Advanced Server

Virtual IP: 161.246.5.243

Primary IP Address : 161.246.5.213

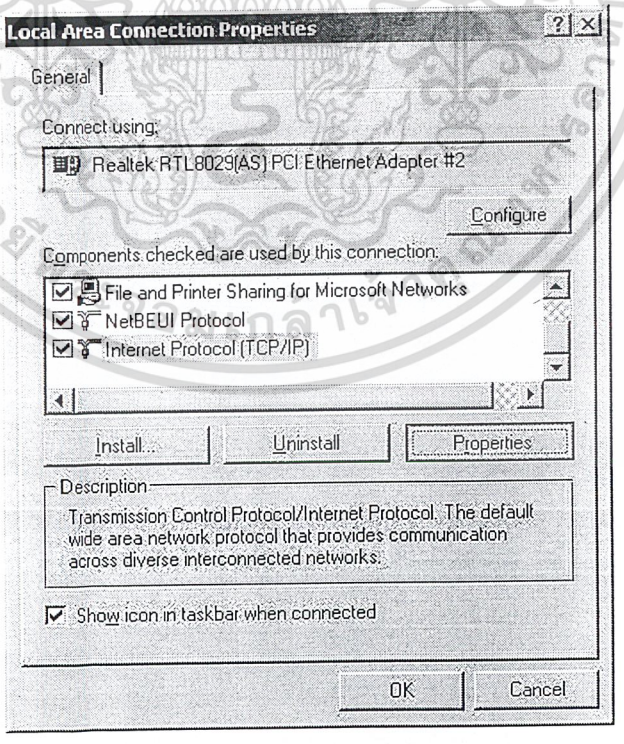
ขั้นตอนในการติดตั้งและปรับแต่ง NLB ทำได้ดังนี้

1. เข้าไปที่ Control Panel แล้วเข้าไปใน Network and Dial-up Connections คลิ๊กขวาที่ Local Area Network เลือก Properties ดังรูป



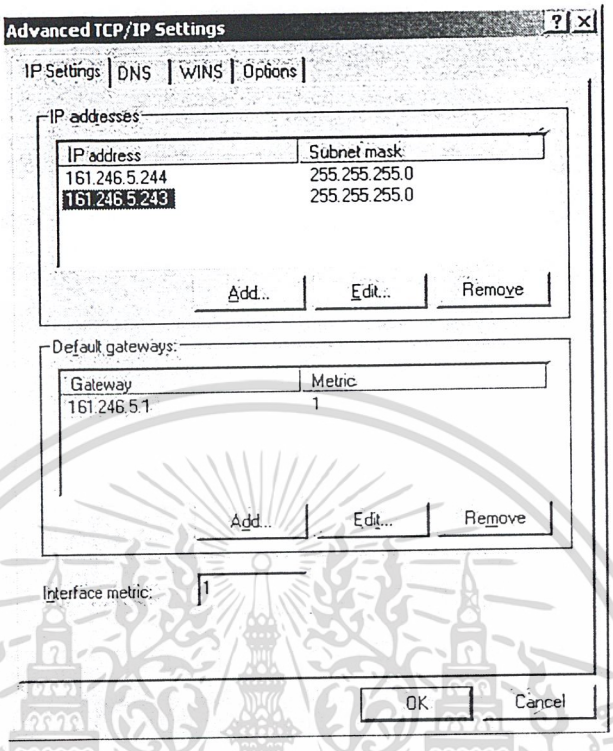
รูปที่ 13-2 เลือกคุณสมบัติของวงแลน

2. เลือกที่ Internet Protocol (TCP/IP) แล้วคลิ๊กที่ปุ่ม Properties ดังรูป



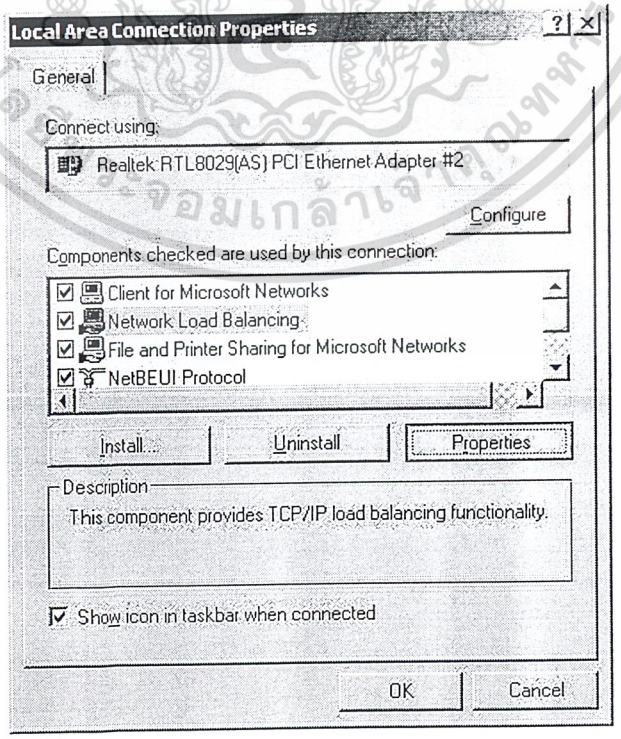
รูปที่ 13-3 เลือกคุณสมบัติของ TCP/IP

3. คลิ๊กเลือกปุ่ม Advance แล้ว เพิ่มค่า Virtual IP Address ลงไป (ในที่นี้ กำหนดให้มีค่าเป็น 161.246.5.243) โดยค่า Subnet Mask ต้องมีค่าเดียวกันด้วย ดังรูป



รูปที่ 13-4 เพิ่ม Virtual IP

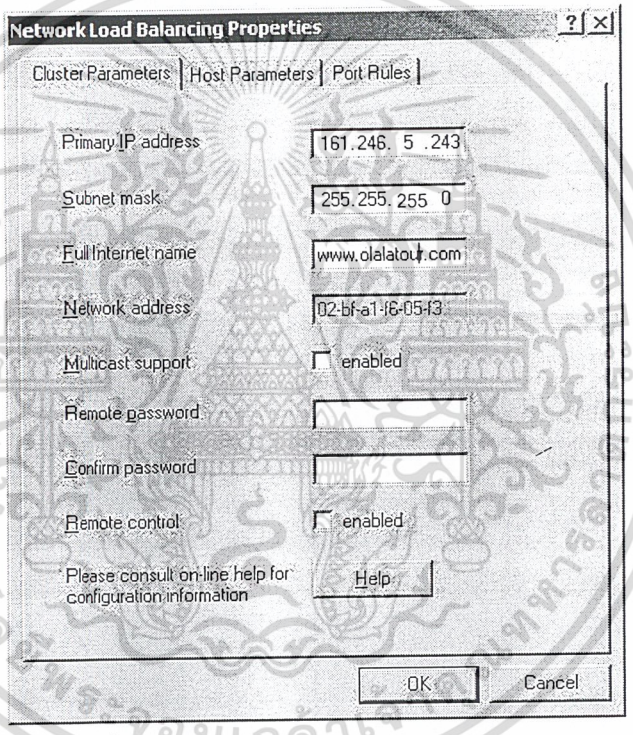
4. เสร็จแล้วคลิ๊ก OK 2 ครั้ง จนกลับมายังหน้า Local Area Connection Properties จากนั้นคลิ๊กให้ตัวเลือก Network Load Balancing มีเครื่องหมายถูกปรากฏ แล้วจึงคลิ๊กที่ปุ่ม Properties ดังรูป



รูปที่ 13-5 เลือกคุณสมบัติของ NLB

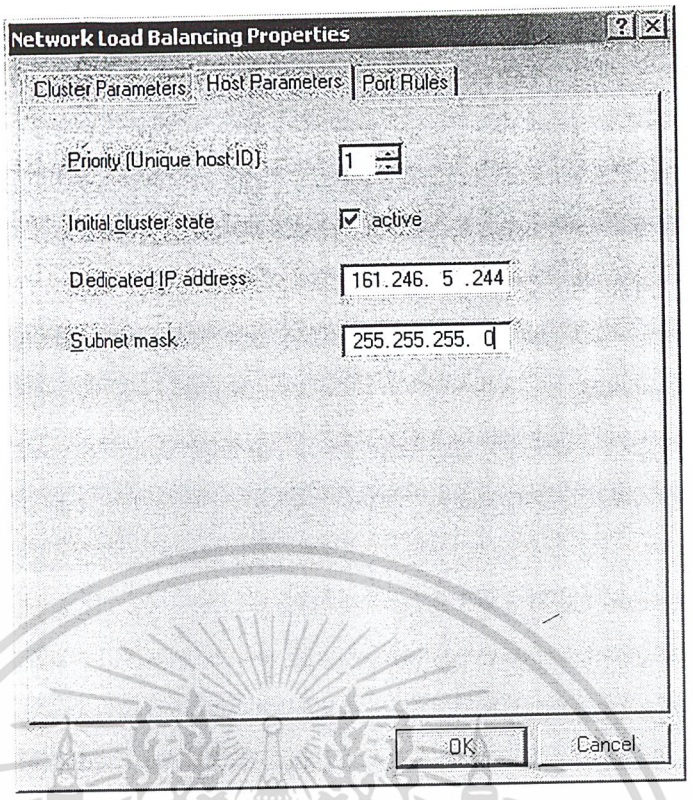
5. จากรูปจะมีแท็ปอยู่ 3 แท็ป คือ Cluster parameters, Host parameters และ Port rules โดยในแท็ปแรกนี้ (Cluster parameter) จะใช้ในการกำหนดข้อมูลเกี่ยวกับคลัสเตอร์ เช่น Virtual IP Address, Subnet mask, การกำหนดว่าจะใช้ Multicast หรือ Singlecast message นอกจากนี้ยังสามารถกำหนดได้อีกว่าจะให้สามารถควบคุมคลัสเตอร์ในระยะไกลผ่าน Command line ได้หรือไม่

NLB จะเรียก Virtual IP ว่าเป็น Primary IP Address ซึ่งค่านี้จะเป็นค่า IP Address ของคลัสเตอร์ และจะถูกใช้ในการเข้าถึงทุกๆ ไซต์ในคลัสเตอร์ ทำให้เราสามารถกำหนดค่า Public DNS ให้แก่ค่า IP นี้ได้ เช่น กำหนดเป็น www.olalatour.com ในส่วนของค่า Full Internet name จะถูกใช้สำหรับการจัดการในระยะไกล (ใช้ในฐานะ ID) ถ้าใช้เน็ตเวิร์กอะแดปเตอร์เพียงใบเดียว จะต้องเลือกตัวเลือก multicast เพื่ออนุญาติให้การ์ดเน็ตเวิร์กสามารถรับรู้ทราฟฟิกทั้งจากคลัสเตอร์และจาก Dedicated IP Address ได้ ในที่นี้ เราจะกำหนดค่าต่างๆ ดังรูป



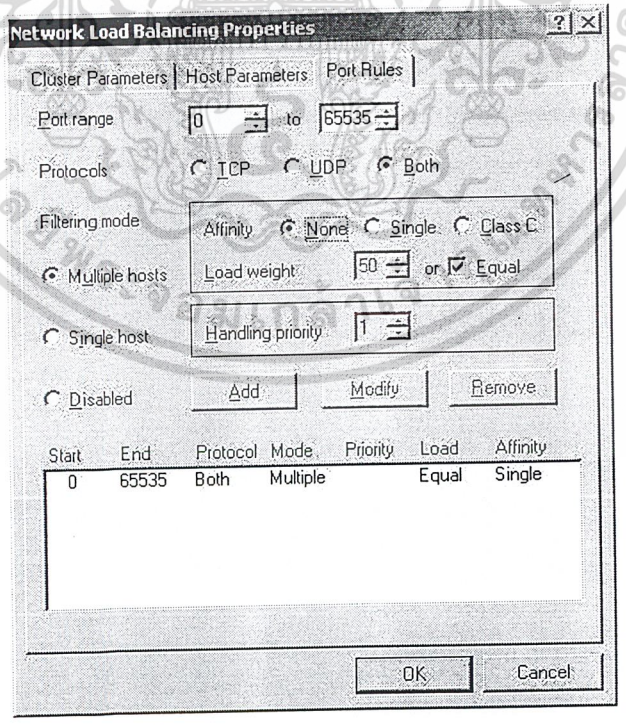
รูปที่ 13-6 ปรับแต่งคลัสเตอร์พารามิเตอร์ของ NLB

6. ในแท็ป Host Parameters เราสามารถปรับแต่งค่า Priority และ Dedicated IP Address ได้ (*อธิบายไว้ในส่วนเนื้อหาของ NLB*) ในที่นี้ เราจะกำหนดค่าต่าง ๆ ดังรูป (อย่าลืมว่า อีกเครื่อง ต้องตั้งค่า Priority เป็น 2 และอาจจะตั้งค่า Dedicated IP Address หรือไม่ก็ได้)



รูปที่ 13-7 ปรับแต่งโฮสต์พารามิเตอร์ของ NLB

7. ในแท็ป Port rules ใช้ในการปรับแต่งว่าคลัสเตอร์โหนดจะแฮนเดิลรีเควสต์อย่างไร (*อธิบายไว้ในส่วนเนื้อหาของ NLB*) ในที่นี้ เราจะกำหนดค่าต่าง ๆ ดังรูป ทั้ง 2 เครื่อง



รูปที่ 13-8 ปรับแต่งพอร์ตรูลของ NLB

8. คลิก OK ออกมา เป็นอันเสร็จ

13.3 การปรับแต่งค่า NLB ด้วย Application Center

ปรับแต่ง network adapter

อย่าเปิดการทำงานของ NLB ยกเว้นถ้าต้องการตั้งค่า NLB เอง เพราะ Microsoft Application Center 2000 จะปรับแต่ง NLB อย่างอัตโนมัติหลังจากเสร็จสิ้นการสร้างคลัสเตอร์

อะแดปเตอร์แรกเรียกว่า front-end มีหน้าที่รับทราฟฟิกจากทั้ง cluster member, Internet และ Intranet client ซึ่ง NLB จะใช้ front end adapter ในการ balancing client request ทั้งหมด เนื่องจากสมาชิกตัวแรกของคลัสเตอร์จะกลายเป็น cluster controller ซึ่งจะต้องมี 2 static IP address

IP address แรกที่กำหนดให้แก่อะแดปเตอร์จะเป็น dedicate IP address (DIP) ซึ่ง DIP จะใช้ติดต่อโดยตรงไปยังแต่ละเซิร์ฟเวอร์ในคลัสเตอร์ใช้ประโยชน์ในการทำ HTTP monitoring (การติดต่อจากเซิร์ฟเวอร์ไปยังเซิร์ฟเวอร์จะทำผ่าน management adapter) สำหรับ IP address ที่สอง จะเป็น cluster IP address หรือที่นิยมเรียกว่า Virtual IP address (VIP) ซึ่ง address นี้จะต้องแมพไปยัง URL ของเว็บไซต์ สำหรับ IP address อื่นๆที่เพิ่มเข้าไปจะกลายเป็น VIP เช่นกัน

อะแดปเตอร์ตัวที่ 2 เรียกว่า back-end จะใช้ในการติดต่อภายในและทราฟฟิกขาออกไปยังเว็บไคลเอ็นต์ซึ่ง application center จะใช้อะแดปเตอร์ตัวนี้ในการทำงานภายในคลัสเตอร์และสนับสนุนความสามารถของ application center เช่น synchronization, heartbeat, ข้อมูลประสิทธิภาพของเครื่อง นอกจากนี้ยังรับผิดชอบการติดต่อกับ database และ component server อีกด้วย

ตัวอย่างการติดตั้งค่า IP address ของ web cluster ที่มีสมาชิกใน cluster จำนวน 2 โหนด

[Web1]	(เป็น cluster controller)
Front End Adapter:	(เรียงตามลำดับการตั้งค่า IP addresses)
- 161.246.5.243	(เป็น DIP ของ Web1)
- 161.246.5.246	(เป็น VIP ของ cluster)
- 161.246.5.247	(option : เป็น VIP เพิ่มเติมของ cluster)

Back End Adapter:

- DHCP IP หรือ Static IP 161.246.5.216

[Web2]	(เป็น cluster member)
Front End Adapter:	(เรียงตามลำดับการตั้งค่า IP addresses)
- 161.246.5.245	(เป็น DIP ของ Web2)

Back End Adapter:

- DHCP IP หรือ Static IP 161.246.5.213

ต่อมาก็สร้าง Web cluster โดยตั้งค่าดังต่อไปนี้

Cluster type = General/Web cluster

Load Balancing = Network Load Balancing (NLB)

Load Balancing Options:

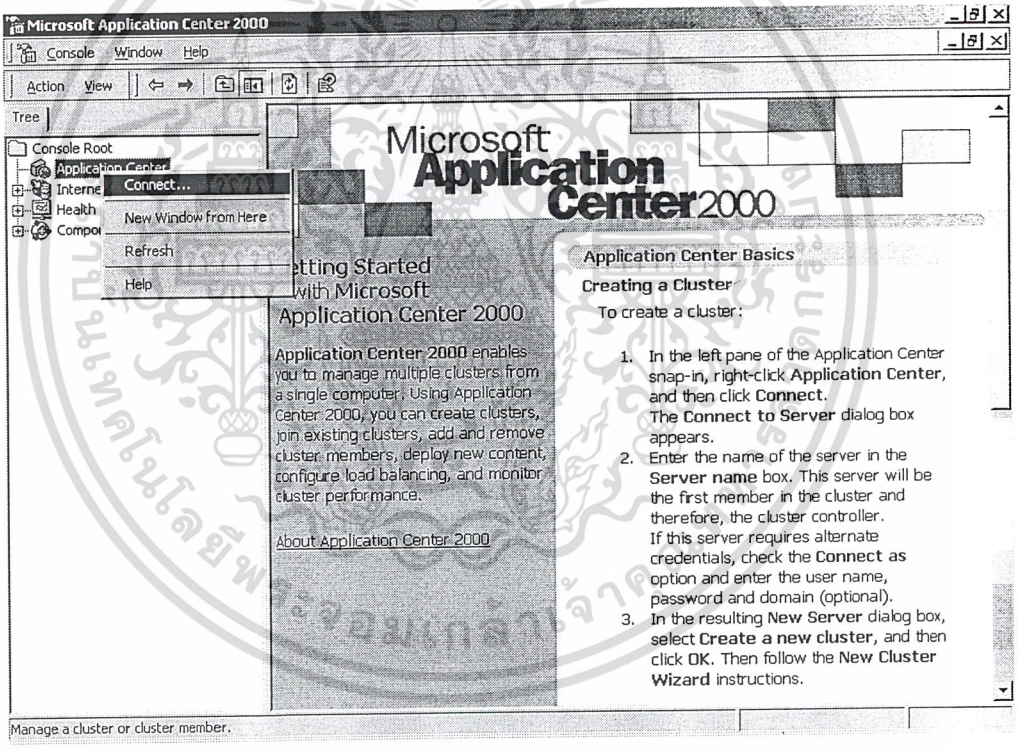
Management traffic network adapter: - <Back End Adapter>

Load balanced network adapter: - <Front End Adapter>

ในหัวข้อ **Load Balancing Options** เป็นการเลือก Network adapter ไม่ใช่ virtual IP สามารถอธิบายได้ตามภาพต่อไปนี้

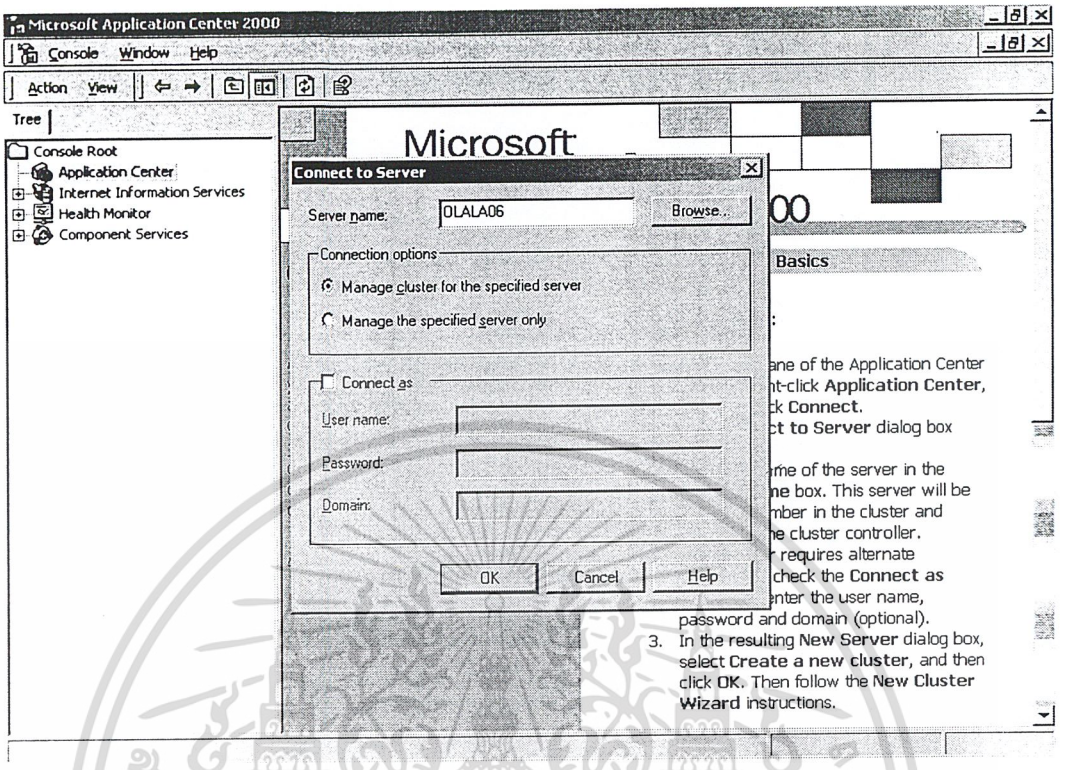
1. สร้าง Web Cluster และ Routing Cluster

   Click ที่ Application Center และเลือก Connect



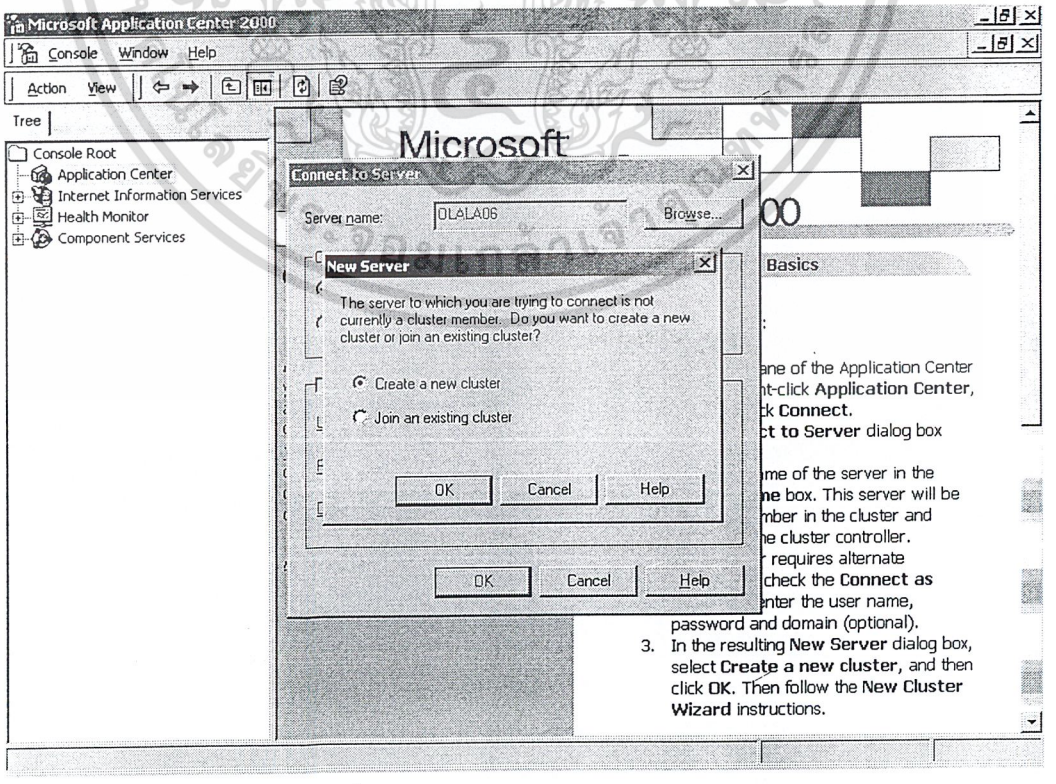
รูปที่ 13-9 ขั้นตอนการสร้าง Cluster

2. ระบุชื่อ Server ที่ต้องการ



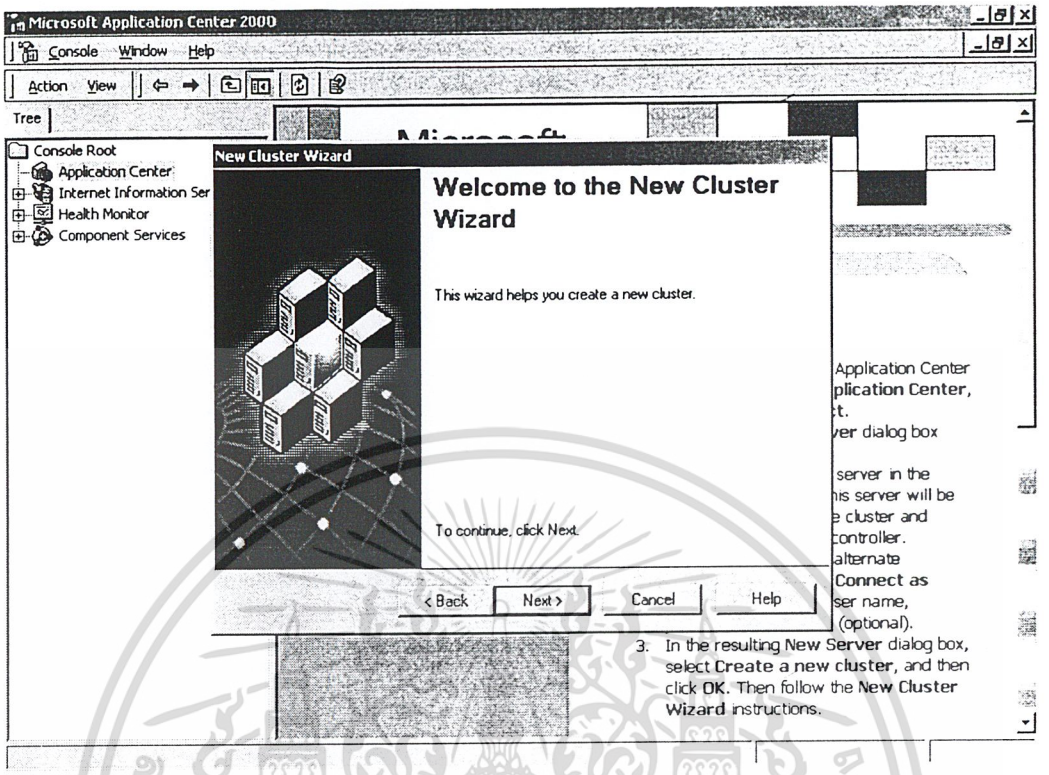
รูปที่ 13-10 การระบุชื่อ Cluster controller

3. เลือกหัวข้อ Create new cluster

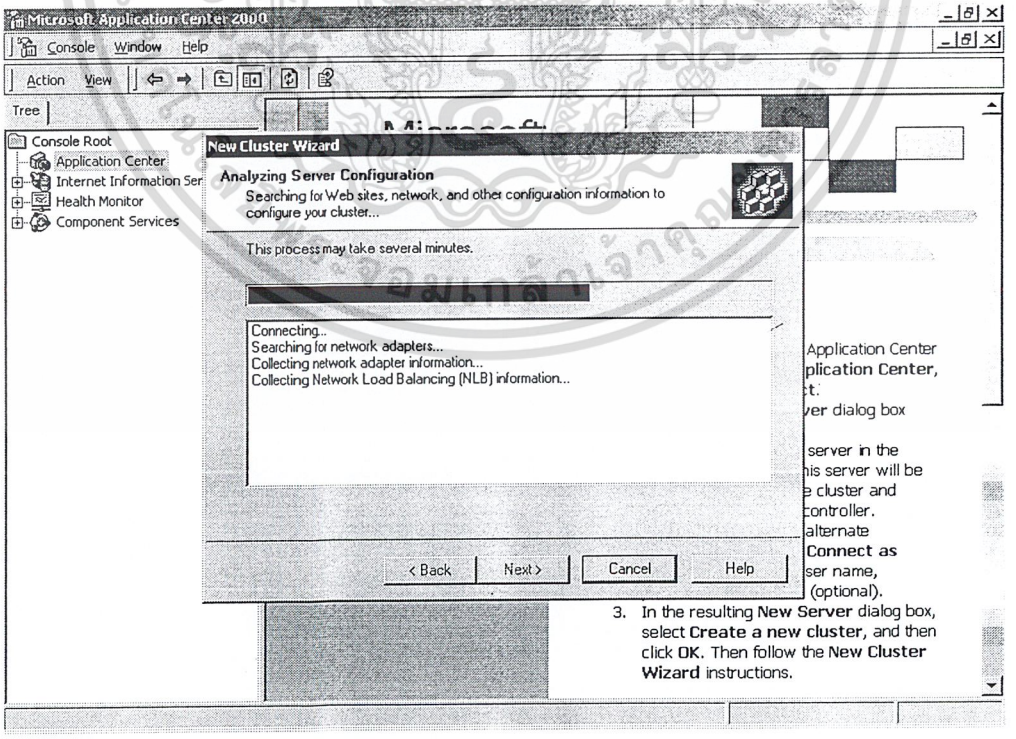


รูปที่ 13-11 การสร้างคลัสเตอร์ใหม่ controller

4. เลือก Next และรอให้โปรแกรมตั้งค่า cluster

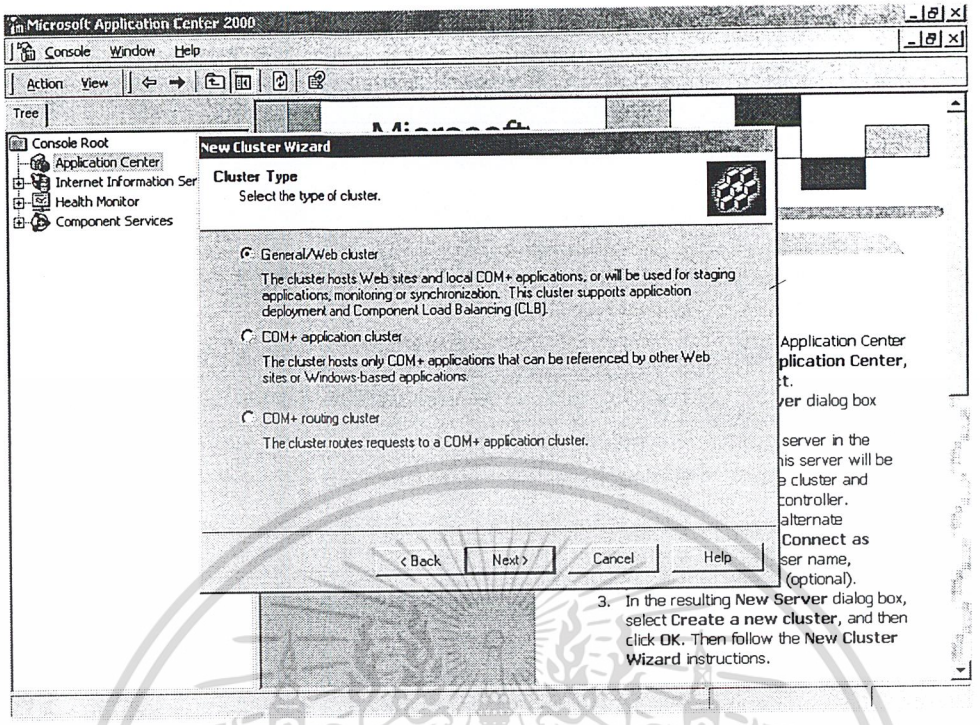


รูปที่ 13-12 แสดงการทำงานของการสร้าง Cluster



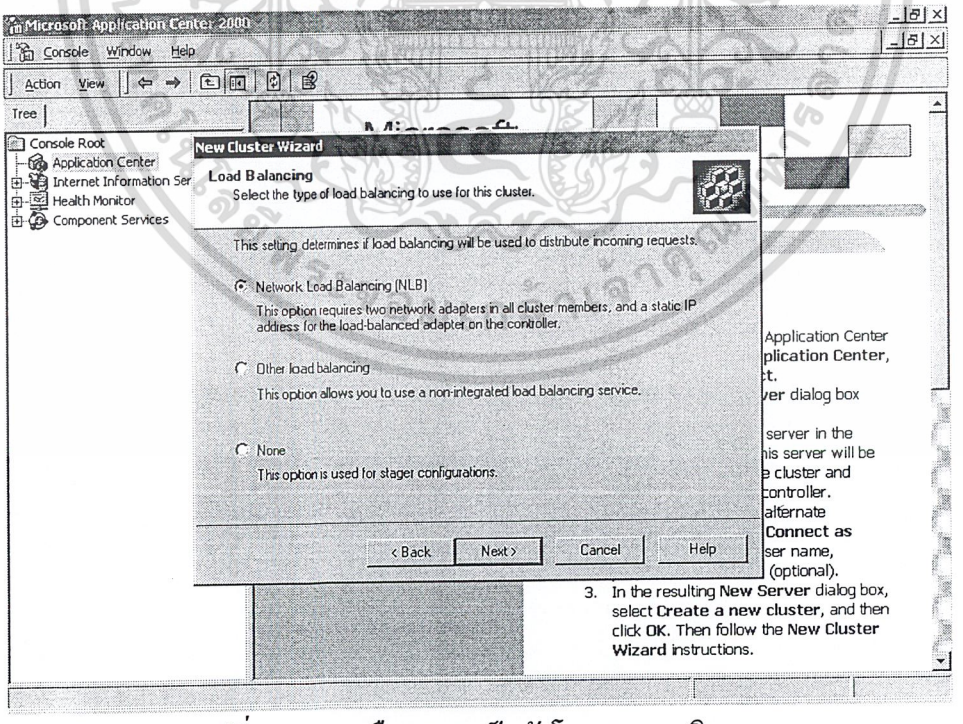
รูปที่ 13-13 การทำงานของการเก็บข้อมูลจาก Server

5. เลือก General/Web cluster



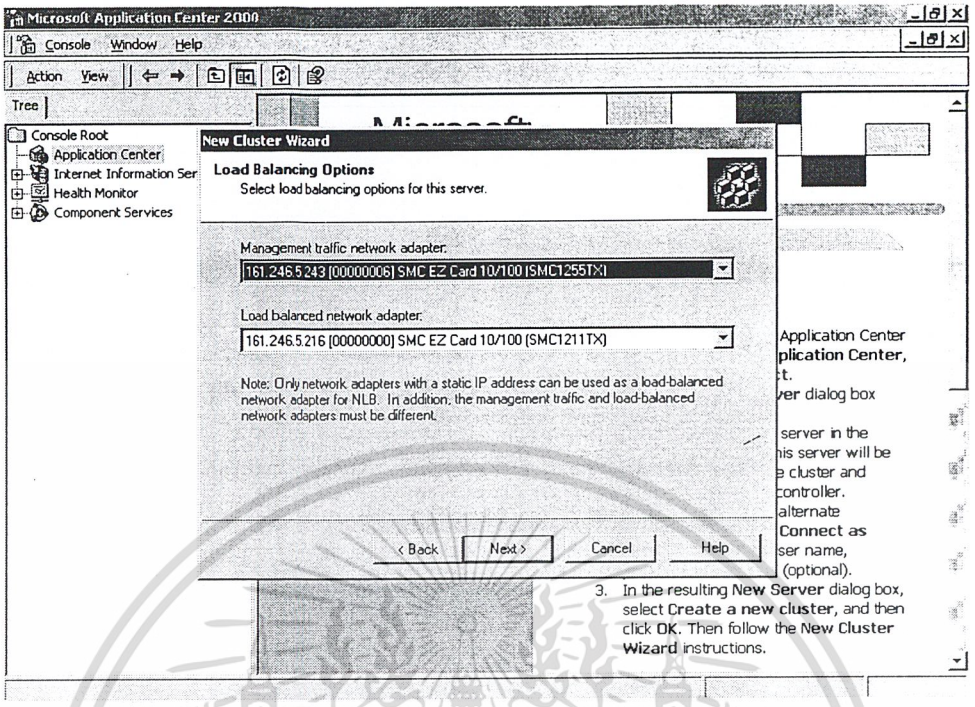
รูปที่ 13-14 การเลือกตัวเลือกในการสร้าง Web cluster

6. เลือก NLB สำหรับการทำ web load balancing



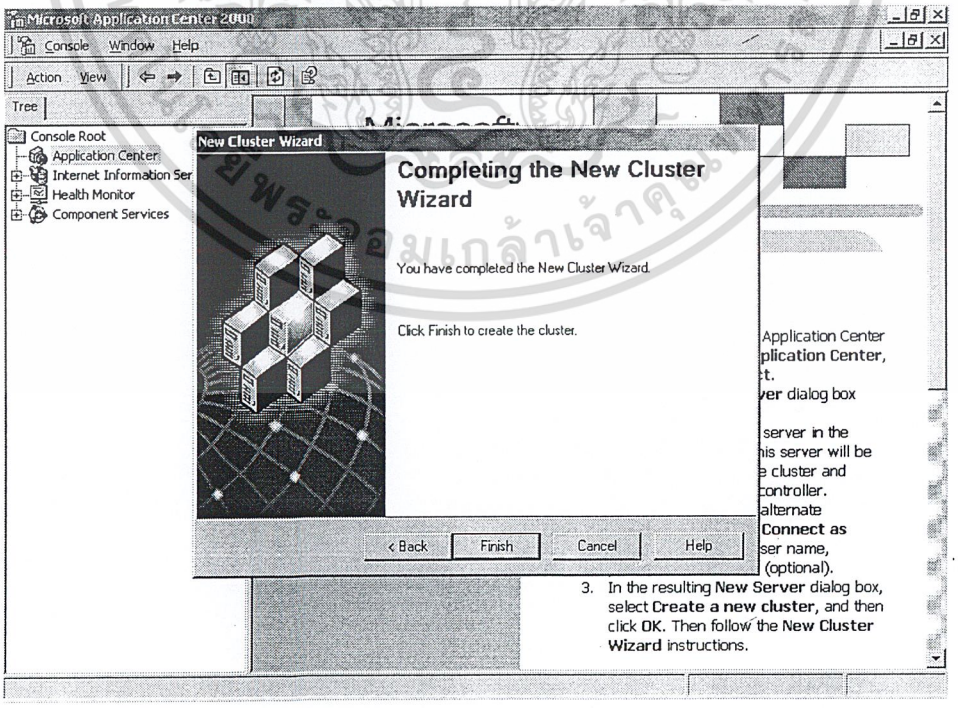
รูปที่ 13-15 การเลือก NLB เป็นตัวโหลดบาลานซิง

7. เลือก LAN card ที่ต้องการให้เป็น management traffic และ load balance



รูปที่ 13-16 แสดงการเลือก LAN card

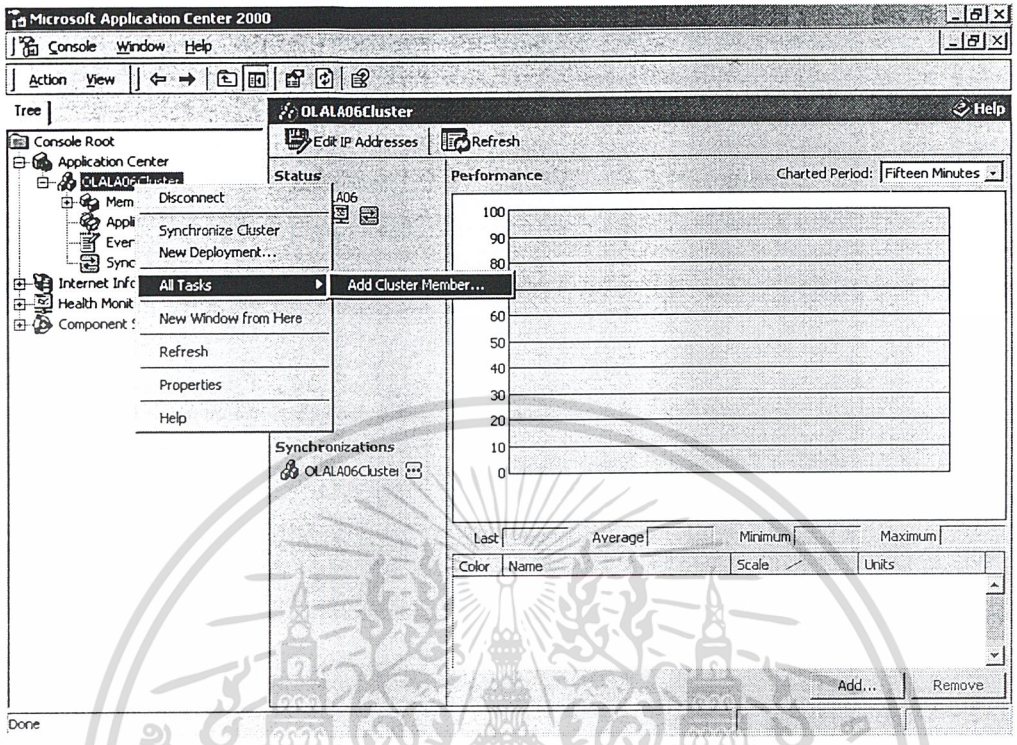
8. Click finish แล้วรอให้โปรแกรมตั้งค่าของ cluster ให้เสร็จ



รูปที่ 13-17 ขั้นตอนสุดท้ายของการสร้าง Cluster

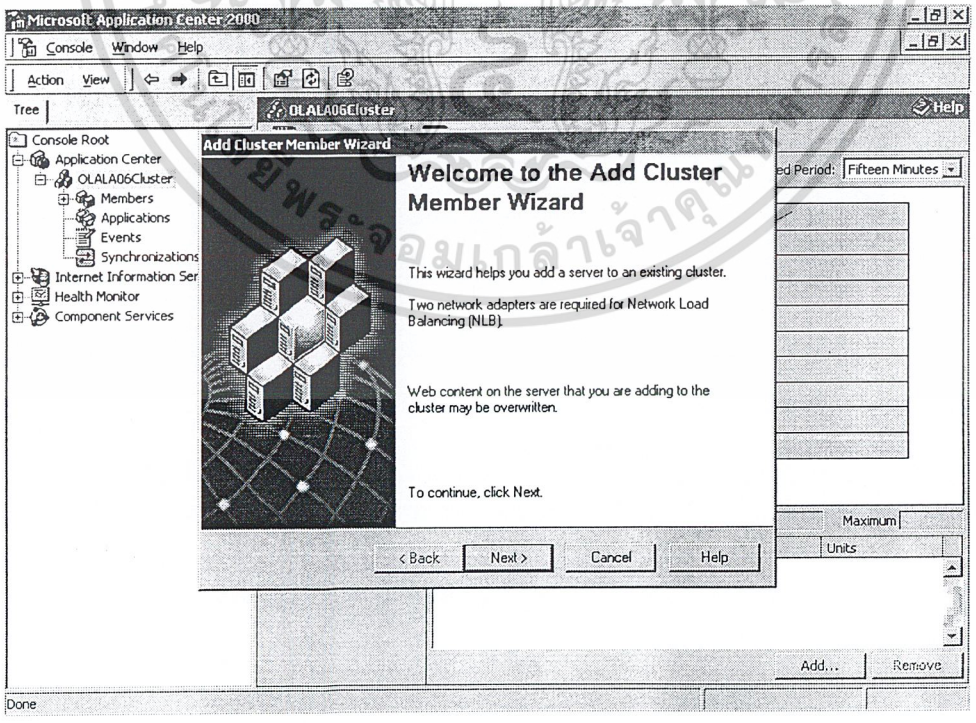
ต่อไปเป็นการเพิ่มสมาชิกเข้าไปใน Web cluster

1. Click ที่ cluster ที่ต้องการจะเพิ่มสมาชิก โดยเลือก All Tasks | Add Cluster Member



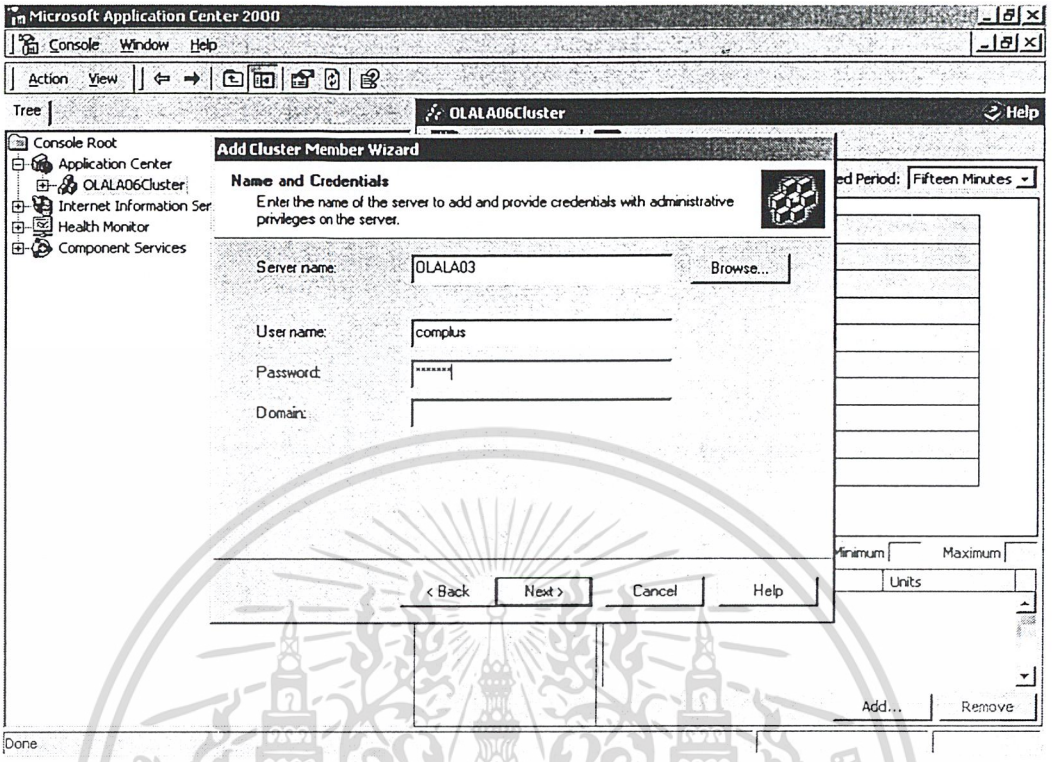
รูปที่ 13-18 การเลือกตัวเลือกในการเพิ่มสมาชิกของ Web cluster

2. Click next

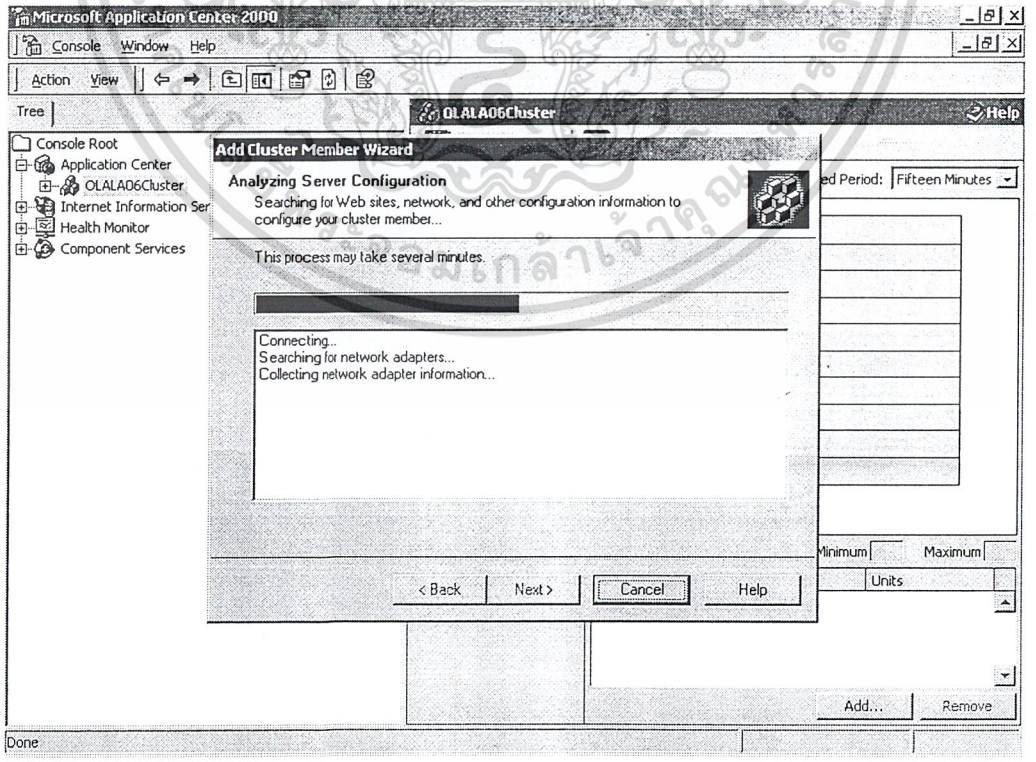


รูปที่ 13-19 การเพิ่มสมาชิกใน Cluster

3.ระบุ Server ที่ต้องการ และใส่ user และ password ในการใช้งานเครื่องด้วย

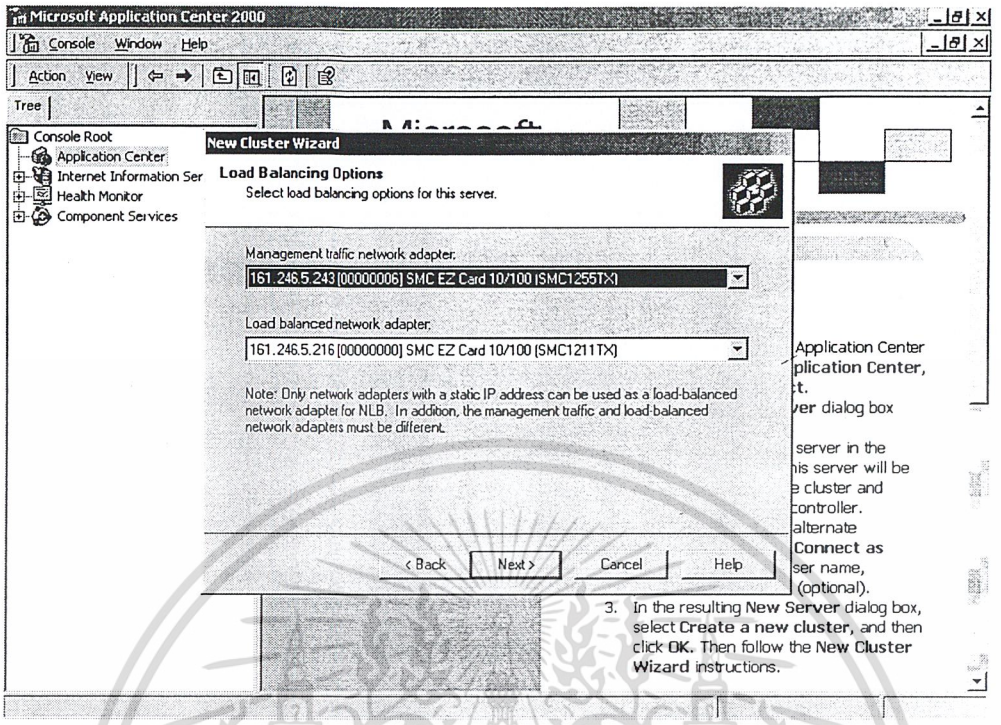


รูปที่ 13-20 การระบุ Server ที่ต้องการให้เป็นสมาชิกของ cluster



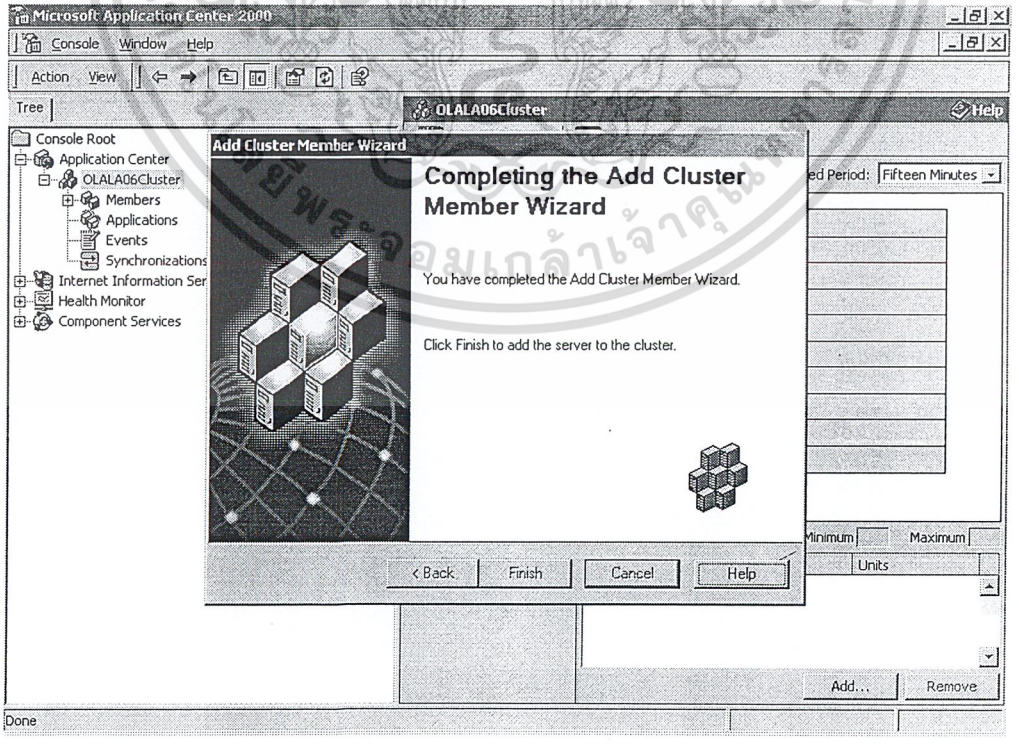
รูปที่ 13-21 การเก็บข้อมูลจาก Server

4. เลือก LAN card ที่ต้องการให้เป็น traffic management และ load balancing



รูปที่ 13-22 การเลือก LAN card

5. รอให้ Application center ปรับแต่งค่าของ cluster ให้เรียบร้อย



รูปที่ 13-23 การเสร็จสิ้นการเพิ่มสมาชิกของ Cluster

13.4 การเตรียม Cluster สำหรับ CLB

ต่อไปเป็นการสร้าง COM+ application cluster

ตัวอย่างการติดตั้งค่า IP address ของ COM+ cluster ที่มีสมาชิกใน cluster จำนวน 2 โหนด

[COM+ server 1]	**(เป็น cluster controller)**
Adapter:	
- 161.246.5.241	(เป็น IP ของ COM+ 1)
[COM+ server 2]	**(เป็น cluster member)**
Adapter:	
- 161.246.5.244	(เป็น IP ของ COM+ 2)

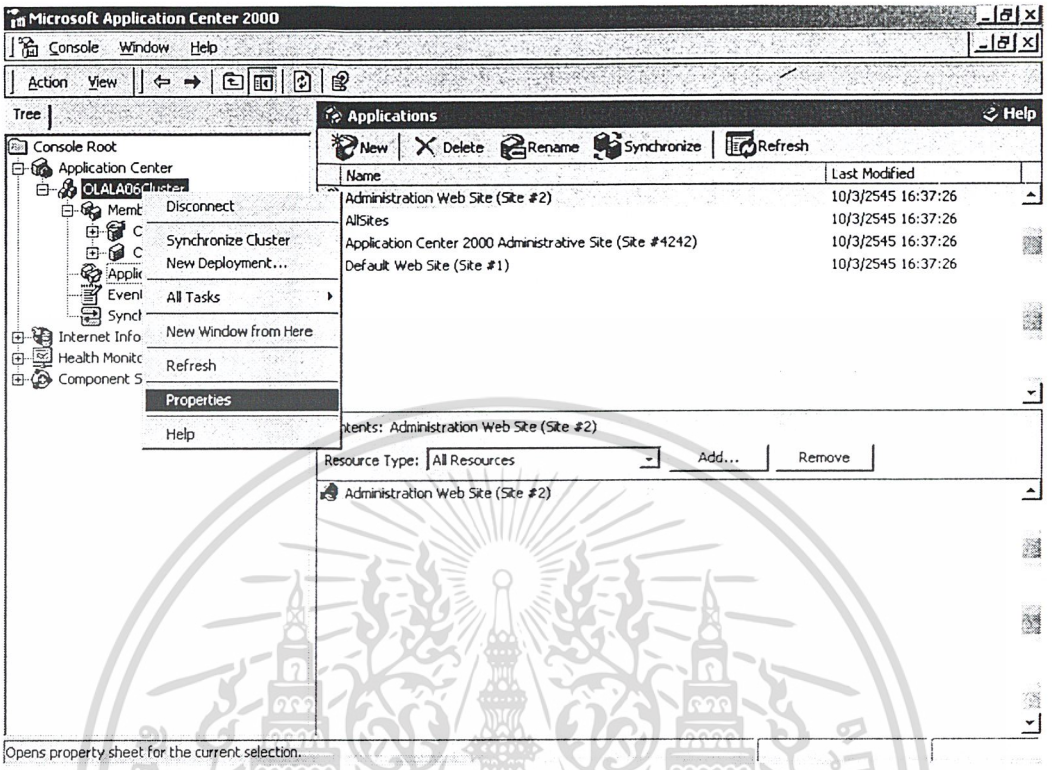
สร้าง COM+ cluster โดยตั้งค่าดังต่อไปนี้

Cluster type = COM+ application cluster

เพิ่มสมาชิกเข้าไปในcluster โดยทำเหมือนกับขั้นตอนที่ผ่านมา

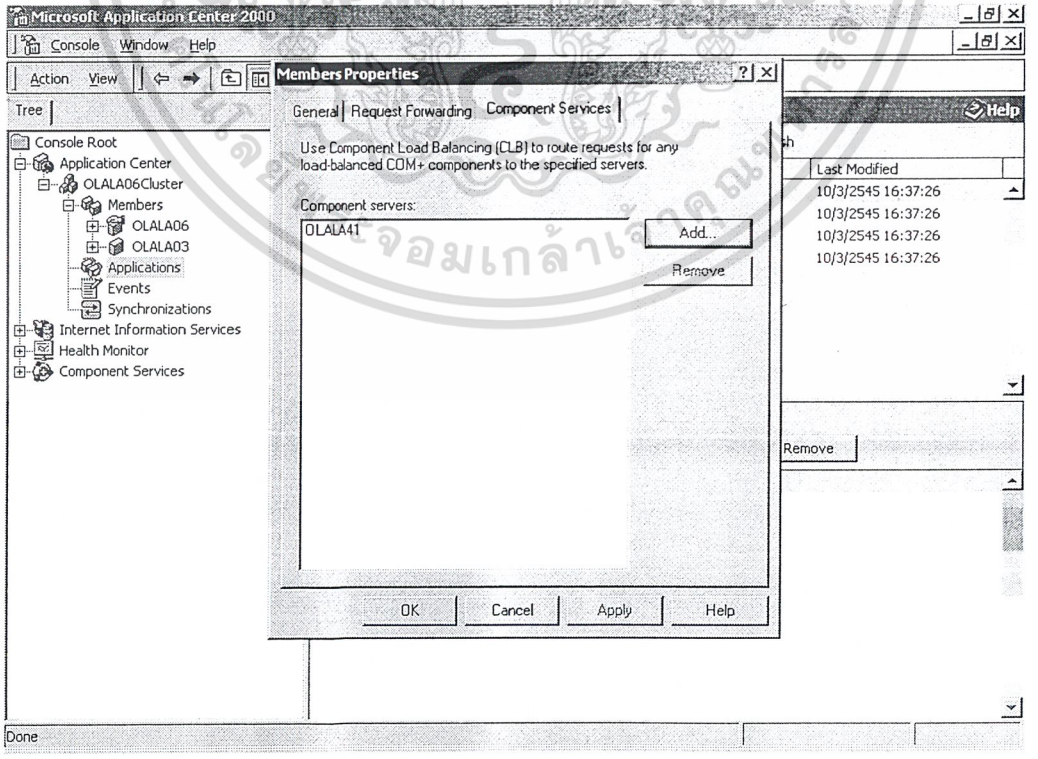
ใส่รายชื่อ server เข้าไปใน routing list ของ web cluster

1. click ขวาที่ web cluster และเลือก properties



รูปที่ 13-24 เลือก Properties ของ web

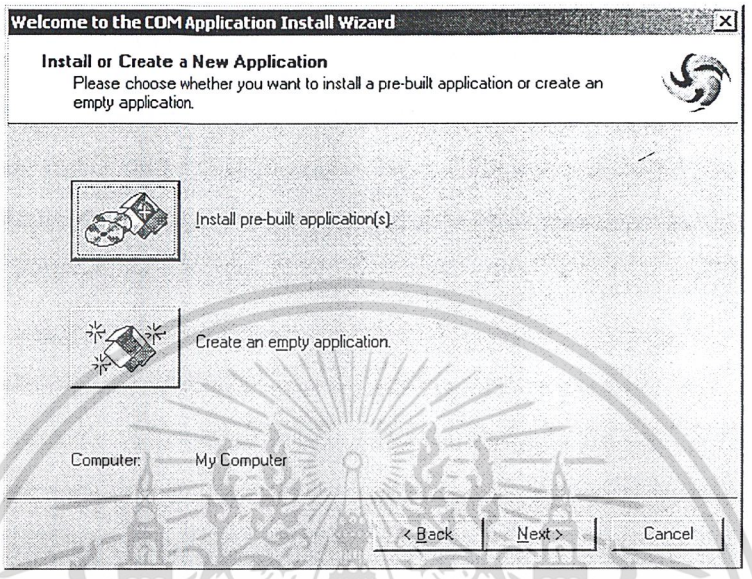
2. เลือก tab component service แล้ว กด Add แล้วเลือกรายชื่อ server ที่ต้องการจะให้อยู่ใน routing lost



รูปที่ 13-25 การเลือกสมาชิกที่ต้องการให้อยู่ใน Routing list

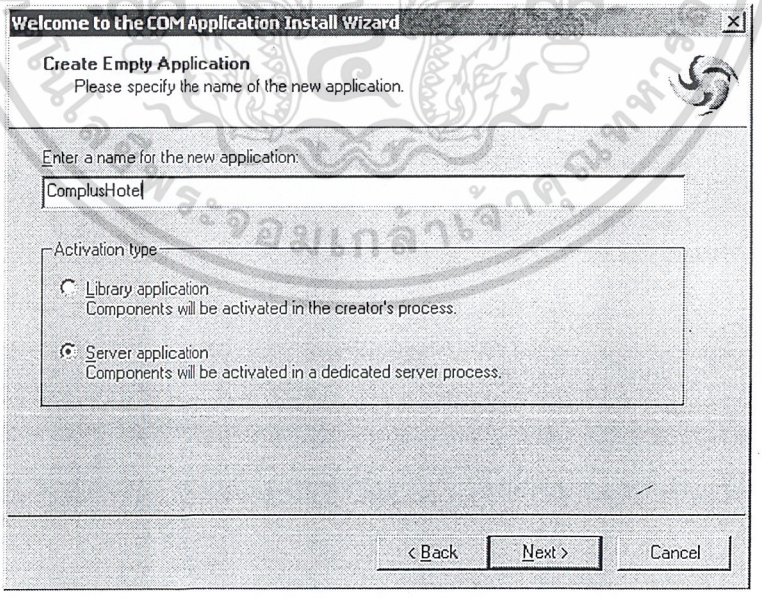
การสร้างตัวติดตั้ง COM+ Application

1. เปิด COM+ Explorer
2. click ขวาที่ COM+ Application เลือก New | Application จะเกิด Dialog Box จากนั้นเลือก Next
3. เลือก Create Empty Application



รูปที่ 13-26 การสร้าง COM+ Application

4. ใส่ชื่อ COM+ Application ที่ต้องการ แล้วเลือก server application แล้ว click next

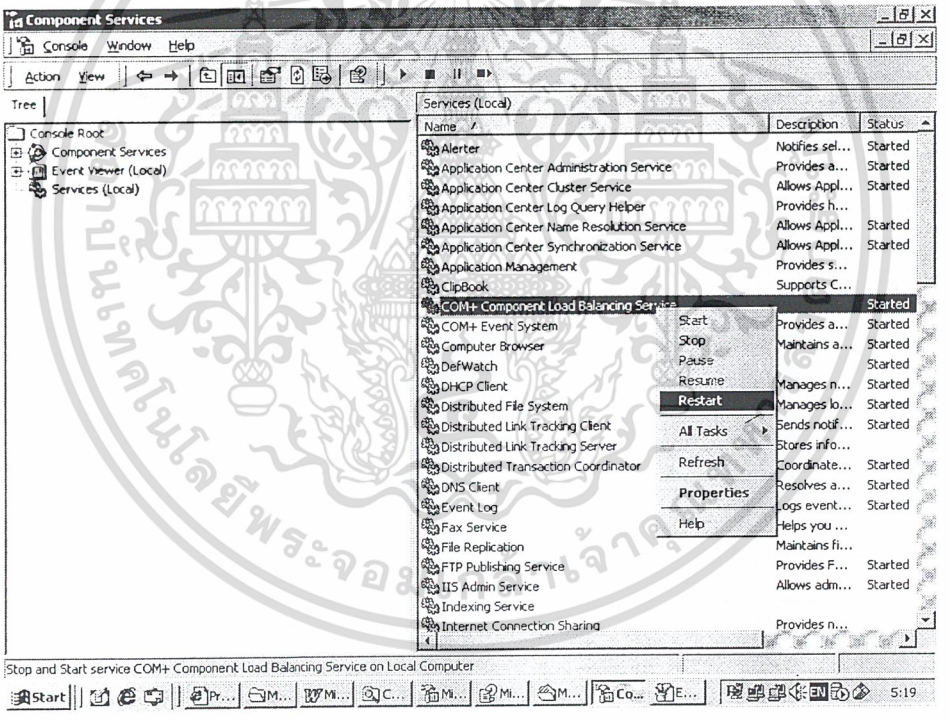


รูปที่ 13-27 การระบุชื่อ COM+ Application ที่ต้องการ

5. ลาก DLL ที่ต้องการมาวางใน Folder Components ของ COM+ Application
6. click ขวาที่ COM+ Application ที่ต้องการจะ Export แล้วเลือก Export
7. ใส่ชื่อไฟล์ MSI ที่ต้องการ , เลือก server application แล้ว กด next
8. จะได้ไฟล์ติดตั้ง COM+ Application ที่มีนามสกุล MSI

เปิดการทำงานของ CLB

1. นำ COM+ Application installer ไปติดตั้งที่ server ทุกเครื่อง ทั้ง Web cluster และ COM+ Application cluster
2. ที่ server ของ Web cluster เปิด COM+ Explorer
   click ขวาที่ Component ที่ต้องการจะทำ CLB จากนั้น click properties
3. เลือก tab Activation
4. เลือก Component supports dynamic load balancing แล้ว OK
5. Restart Component supports dynamic load balancing



รูปที่ 13-28 การ Restart Component supports dynamic load balancing

13.5 การเปรียบเทียบประสิทธิภาพการกระจายโหลดด้วย NLB

การทดสอบนี้ จัดทำขึ้นเพื่อทดสอบการกระจายโหลดด้วยเทคโนโลยี Network Load Balancing (NLB) ซึ่งช่วยกระจายโหลดในชั้นเว็บเทียร์

ในการทดสอบนี้ จะใช้โปรแกรมที่พัฒนาขึ้นมาจากภาษาจาวา เพื่อสร้าง HTTP Request ไปที่ Virtual IP ของคลัสเตอร์ที่เป็น NLB ซึ่งเว็บเพจที่ร้องขอจะคำนวณหาค่าจำนวนเฉพาะ ที่มีค่าอยู่ในช่วง 1-20000 แล้วจึงตอบกลับมา

เครื่องที่ใช้ในการทดสอบทุกเครื่อง ติดตั้งระบบปฏิบัติการไมโครซอฟท์วินโดว์ 2000 แอดวานซ์เซิร์ฟเวอร์ และเปิดใช้บริการ NLB ในเน็ตเวิร์กพรอพเพอตี้ โดยกำหนดให้ทุกเครื่องใช้ Affinity เป็น "none" และ Load Weight เป็น "Equal" แต่ละเครื่องมีสเปก ดังนี้

ชื่อเครื่อง	CPU (MHz)	RAM (MB)	LAN (Mbits)	OS	Affinity	Load Weight
OLALA03	Celeron 450	384	100	Microsoft Windows 2000 Advanced Server	None	Equal
OLALA41	Atlon 700	320	100		None	Equal
OLALA06	Duron 650	384	100		None	Equal
OLALA04	P!!! 600EB	384	100		None	Equal

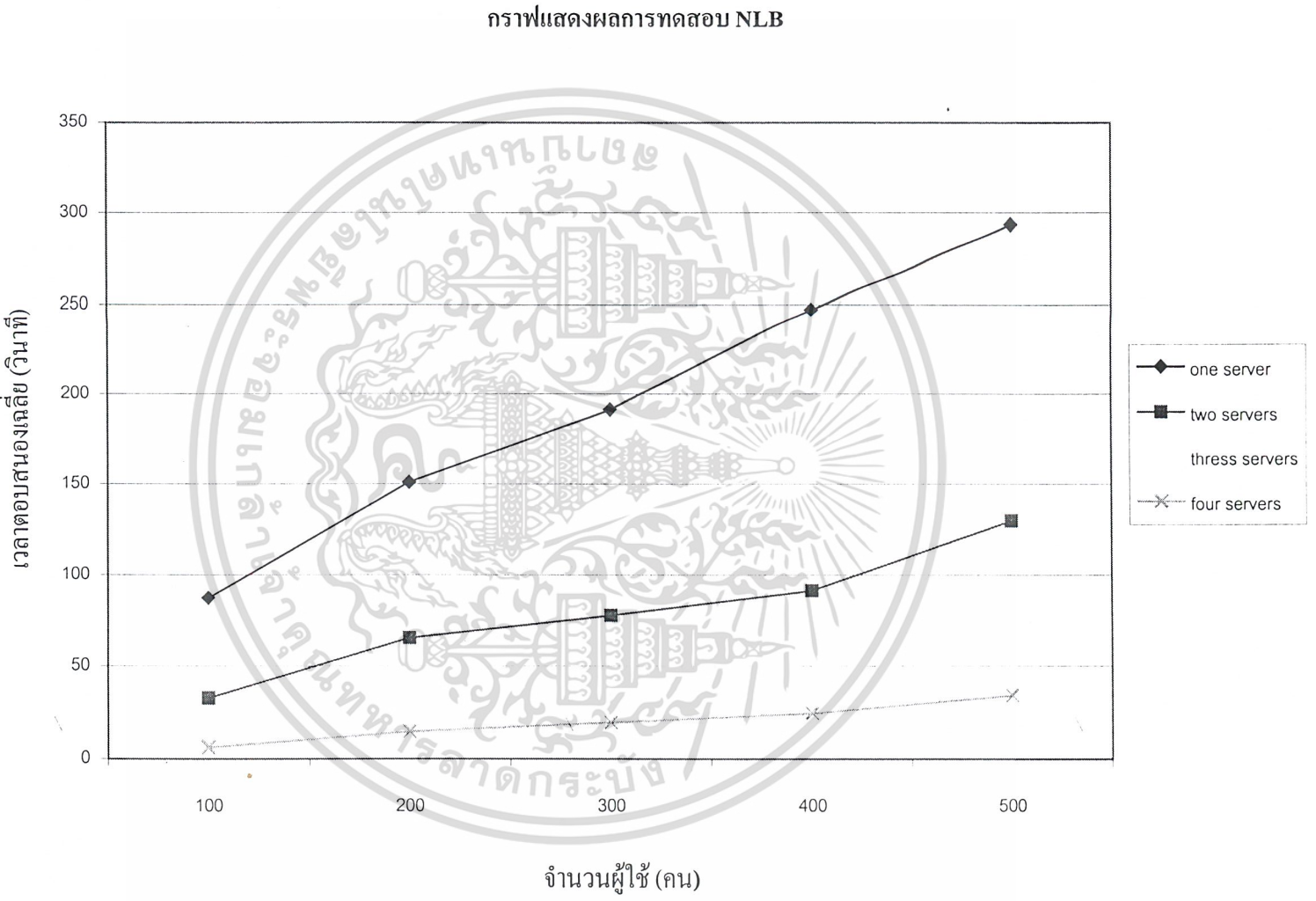
ตารางที่ 13-1 สเปกเครื่องที่ใช้ทดสอบ

การทดสอบการกระจายโหลดนี้ จะเริ่มจากการใช้เครื่องเซิร์ฟเวอร์เพียงเครื่องเดียวในการทดสอบก่อน จากนั้นจึงค่อยๆ เพิ่มเครื่องไปเรื่อยๆ จนกระทั่งครบ 4 เครื่อง (NLB สามารถ เพิ่มเครื่องเซิร์ฟเวอร์ได้สูงสุด 32 เครื่อง) ลำดับการเพิ่มเครื่องเป็นไปตามลำดับ ดังนี้

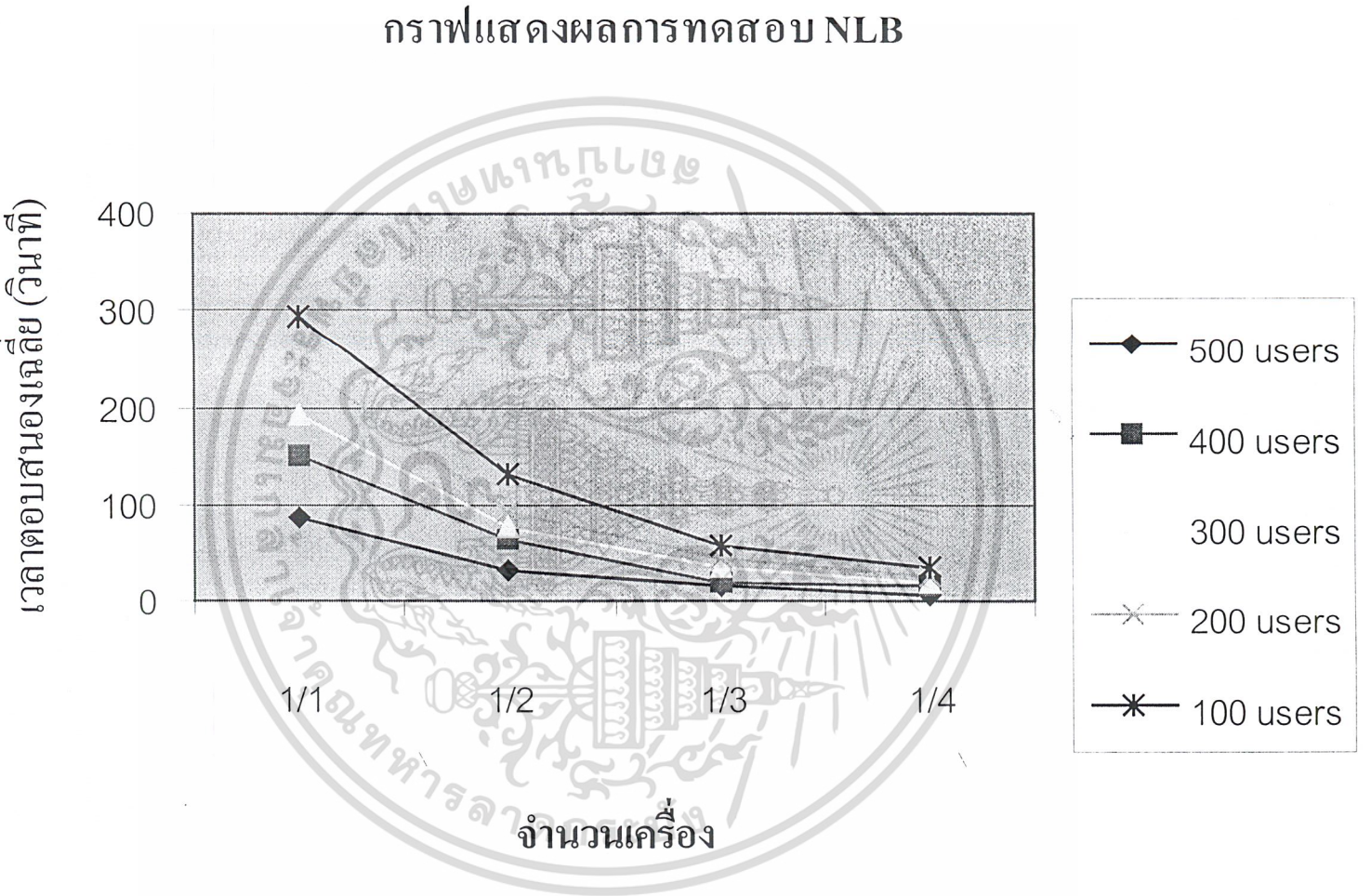
จำนวนเครื่องเซิร์ฟเวอร์ที่ทำการกระจายโหลด (เครื่อง)	ชื่อเครื่อง
1	OLALA03
2	OLALA03 + OLALA04
3	OLALA03 + OLALA41 + OLALA06
4	OLALA03 + OLALA41 + OLALA06 + OLALA04

ตารางที่ 13-2 ลำดับการเพิ่มเครื่องเซิร์ฟเวอร์

จากการทดสอบข้างต้น ได้กราฟแสดงผลการทดสอบ ดังนี้

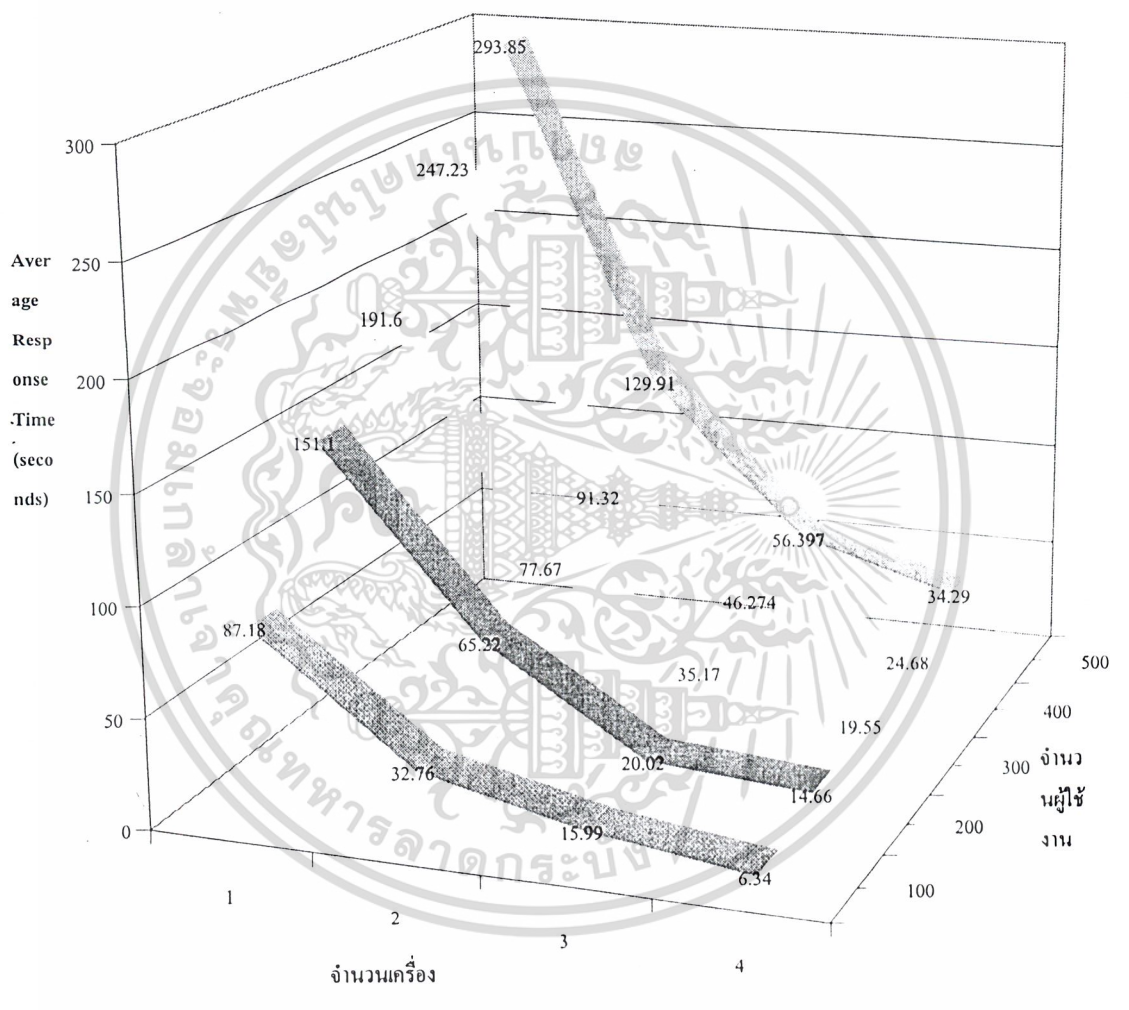


ภาพที่ 13-29 กราฟแสดงผลการทดสอบ NLB (ตามจำนวนเครื่องเซิร์ฟเวอร์)



ภาพที่ 13-30 กราฟแสดงผลการทดสอบ NLB (ตามจำนวนผู้ใช้)

กราฟแสดงผลการทดสอบ NLB



ภาพที่ 13-31 กราฟแสดงผลการทดสอบ NLB (3 แกน)

บทที่ 14
การเปรียบเทียบระหว่าง COM+ และ EJB

การเปรียบเทียบระหว่างเทคโนโลยี COM+ และ EJB นั้นไม่สามารถเปรียบเทียบประสิทธิภาพการทำงานได้ได้โดยตรงเนื่องจากความแตกต่างของแพลตฟอร์ม ดังนั้นจะเปรียบเทียบในด้านความสามารถในการทำงานของทั้งสองเทคโนโลยีมากกว่า สำหรับการเปรียบเทียบในเรื่องการกระจายโหลด จะเปรียบเทียบระหว่างซอฟต์แวร์ของไมโครซอฟท์คือวินโดวส์ 2000 และ แอพพลิเคชัน เซ็นเตอร์ 2000 กับ WebLogic ซึ่งเป็นแอพพลิเคชันเซิร์ฟเวอร์ที่ใช้เทคโนโลยี EJB

14.1 เปรียบเทียบลักษณะทั่วไประหว่าง COM+ และ EJB

Feature	COM+	EJB
Programming Languages	Language Neutrality คือ เขียนคอมโพเนนต์ด้วยภาษาใดก็ได้ เช่น C++, Java, Visual Basic, etc.	Only in Java
Development tools	Microsoft Visual Studio, Delphi, etc.	Visual Café, Visual Age for Java, JBuilder
Middleware Vendors	Microsoft	30+
Legacy Integration	COM TI, MSMQ, OLE DB	RMI/JNDI, CORBA, Connectors
Limited State and Persistence Management	- Stateless component - flushes all object state at the end of every transaction - ต้องจัดการกับ state เอง	Stateless bean, Stateful bean, Entity bean
Method-granularity transactions (configuration)	No	Yes
Middle-tier load balancing	Network load balancing (NLB : เป็นการกระจายโหลดแบบ distribute), Component load balancing (CLB : เป็นการกระจายโหลดแบบ centralize)	ขึ้นกับ Application Server

Portability (Platform)	No. ยึดติดกับ Windows 2000 ไม่สามารถย้ายโค้ดไปที่แพลตฟอร์มอื่นได้	Yes. ถ้าเขียนโค้ดตาม EJB Specification จะสามารถนำไปรันในเครื่องอื่นได้ (Platform independent)
Interoperability	ผ่าน DCOM, ASP CORBA-Bridge	- Deploy ง่ายในระบบที่ต่างกัน (Heterogeneous System) - ผลิตภัณฑ์คนละยี่ห้อสามารถทำงานร่วมกันได้ทั้งใน system-level และ application-level
Scalability	ใน Windows 2000 Advanced Server สนับสนุน 16 โพรเซสเซอร์ และใน Windows 2000 Data Center สนับสนุน 32 โพรเซสเซอร์	สามารถ deploy ได้ใน UNIX และ Mainframe ที่สนับสนุน 256 โพรเซสเซอร์
Security	Integrated with Windows 2000	สนับสนุน CORBA security หรือ SSL หรือ JAS
Reliability	Clustering and load-balancing	สามารถ deploy ได้ในระบบที่มีความเสถียรสูง ขึ้นกับ Application Server
Application Servers based	Microsoft Windows 2000	Many vender เช่น IBM's WebSphere, BEA's WebLogic, Oracle's application server, etc.

ตารางที่ 14-1 เปรียบเทียบลักษณะทั่วไประหว่าง COM+ และ EJB

14.2 เปรียบเทียบเทคโนโลยีในเว็ปเทียร์ระหว่าง NLB และ WebLogic's web-tier

Feature	NLB	WebLogic's web-tier
Load	Load คือ HTTP request	Load คือ HTTP request
การกระจาย load	- NLB Driver แทรกอยู่ระหว่าง TCP/IP กับNetwork Adapter Driver ในทุกเครื่อง - ทุกเครื่องรับ packet พร้อมๆกัน แล้วจึงดู hash table เครื่องที่ถูก	ใช้ proxy software – Round Robin หรือใช้ Hardware Load balancer

	เลือกจะทำงาน เครื่องอื่นจะทิ้ง packet ไป	
Hardware	ไม่ใช้ server เพิ่มเติม	ต้องมี server ทำหน้าที่ proxy กระจาย load
Architecture	Distributed Architecture	Centralized Architecture
Single point of failure problem	ไม่มีปัญหา single point of failure	มีปัญหา single point of failure ที่ proxy
Fail-over	ไม่มี	มี
Cluster members	จำนวน cluster member มีได้มากที่สุด 32 เครื่อง	ไม่จำกัด

ตารางที่ 14-2 เปรียบเทียบเทคโนโลยีในเว็ปเทียร์ระหว่าง NLB และ WebLogic's web-tier

14.3 เปรียบเทียบเทคโนโลยีในคอมโพเน็นต์เทียร์ระหว่าง CLB และ WebLogic's Component-tier

Feature	CLB	WebLogic's Component -tier
Load	Load คือ point of instantiated	Load คือ point of calling method
การกระจาย load	Response time และ Round robin	Combination ใน 3 อัลกอริทึมนี้ - Random - Weight - Round robin
Software	Microsoft Application Center 2000	Web Logic
Architecture	Centralized Architecture	Distributed Architecture
Single point of failure problem	แก้ปัญหา single point of failure ด้วย NLB หรือ Cluster service	ไม่มี
Cluster members	ไม่จำกัด	ไม่จำกัด
Fail-over	ไม่มี	มี

ตารางที่ 14-3 เปรียบเทียบเทคโนโลยี

ในคอมโพเน็นต์เทียร์ ระหว่าง CLB และ WebLogic's Component-tier

บทที่ 15
บทวิจารณ์ และสรุป

15.1 สรุปผลการดำเนินงาน

ในโครงงานนี้ได้แบ่งการดำเนินงานออกเป็น 5 ส่วนหลัก คือ

1. ศึกษาทฤษฎีพื้นฐานของ Windows DNA, COM, COM+, XML, SOAP, Clustering และเทคโนโลยีอื่นๆ ที่เกี่ยวข้อง
2. ออกแบบระบบงานขนาดใหญ่เพื่อทำการทดสอบทฤษฎีต่างๆ ที่ได้ศึกษามา โดยตัวอย่างระบบงานที่สร้างขึ้นมา คือ ระบบการจองแพ็กเกจท่องเที่ยวและระบบการจองโรงแรม
3. ทำการอิมพลีเมนต์ระบบงานที่ได้ออกแบบไว้ โดยใช้ภาษา Visual Basic, ASP, HTML พัฒนาตามมาตรฐานของ COM และอิงตามแพลตฟอร์ม Windows DNA
4. ทำการอิมพลีเมนต์ระบบงานในส่วนที่เรียกใช้ระบบการจองสายการบินและระบบการจองรถโดยสาร ที่พัฒนาจากเทคโนโลยี EJB ผ่านโพรโตคอลมาตรฐานที่เรียกว่า SOAP
5. เพิ่มความน่าเชื่อถือในการใช้งานให้แก่ระบบงานโดยการสร้างคลัสเตอร์และทำการกระจายโหลดภายในคลัสเตอร์ นอกจากนี้ยังใช้บริการต่างๆ ของ COM+ เพื่อเพิ่มความสามารถต่างๆ ให้แก่ระบบงานด้วย

ระบบงานที่ทดลองสร้างขึ้น มีการใช้เทคโนโลยีต่างๆ ที่ได้ศึกษามา โดยได้ทำการสร้างระบบงานเป็น 3 เทียร์ส คือ

1. *พรีเซนเตชั่นเทียร์* มีไคลเอ็นต์เป็นเว็บบราวเซอร์ที่เรียกใช้บริการจากระบบและแอพพลิเคชันที่ทำหน้าที่เป็นเครื่องมือในการจัดการระบบ
2. *บิสิเนสเทียร์* มีเว็บเซิร์ฟเวอร์ IIS และใช้เซิร์ฟเวอร์ไซด์สคริปเป็น ASP นอกจากนี้ยังมีแอพพลิเคชั่นเซิร์ฟเวอร์ ซึ่งเป็น COM+ รันไทม์
3. ดาต้าเทียร์ ใช้ดาตาเบสเซิร์ฟเวอร์เป็น Microsoft SQL Server 2000

15.2 แนวทางการพัฒนาต่อ

ถึงแม้ว่า การทดลองสร้างระบบงานให้สามารถเรียกใช้บริการจากระบบงานอื่นที่พัฒนามาจากเทคโนโลยีที่แตกต่างกันได้ (คือ COM+ และ EJB) โดยผ่านทางโพรโตคอล SOAP นั้น สามารถทำงานได้เป็นอย่างดีก็ตาม แต่เครื่องมือที่ใช้งานทั้ง 2 ฝ่ายในปัจจุบันไม่มีประสิทธิภาพเพียงพอ คือ ใช้ Microsoft SOAP Tookit 2.0 Beta 2 และ Apache SOAP 2.2 ทำให้ต้องมีการปรับแต่งและเขียนโปรแกรมเพิ่มเติมมากมาย ซึ่งปัญหาเหล่านี้ แก้ได้โดยแต่ละฝั่งสร้างเป็นเว็บเซอร์วิส (Web Services) ให้อีกฝ่ายหนึ่งเรียกใช้ ซึ่งอาจสร้างได้โดยใช้ Microsoft Visual Studio.NET นอกจากนี้ ทางไมโครซอฟท์ยังได้พัฒนาเฟรมเวิร์กใหม่ที่เรียกว่า .NET Framework ซึ่งเป็นคอมโพเน็นต์ออบเจ็กต์โมเดลตัวใหม่ ที่ทำให้การพัฒนาซอฟท์แวร์สะดวก รวดเร็วและมีความสามารถเพิ่มขึ้นกว่าเดิมเป็นอย่างมาก และยังเข้ากันได้กับระบบ COM+ อีกด้วย ซึ่งน่าจะเป็นสิ่งที่ควรศึกษาและพัฒนาต่อไป

บรรณานุกรม

[1] David J. Kruglinski, Scot Wingo, George Shepherd : *Programming Microsoft Visual C++ 5th Edition* , 1998.

[2] Jeff Prosise : *Programming Windows with MFC 2nd Edition.* , Microsoft Press, 1999.

[3] Guy Eddon, Henry Eddon : *Inside COM+ Base Services*, Microsoft Press, 1999.

[4] Mary Kirtland : *Designing Component-Based Applications*, Microsoft Press, 1998.

[5] Microsoft Corporation : *Desktop Applications with Microsoft Visual Basic 6.0 : MCSD Training Kit* , Microsoft Press, 1999.

[6] Microsoft Corporation : *Desktop Applications with Microsoft Visual C++ 6.0 : MCSD Training Kit* , Microsoft Press, 1999.

[7] The Mandelbrot Set (International) Limited. : *Advanced Microsoft Visual Basic 6.0 2nd Edition.*, Microsoft Press, 1998.

[8] Robert J. Oberg : *Understanding and programming COM+ :a practical guide to Windows 2000 DNA.* Prentice Hall PTR, 2000.

[9] Ted Pattison : *Programming Distributed Applications with COM and Microsoft Visual Basic 6.0*, Microsoft Press, 1998.