บทที่ 3
แนวคิด ทฤษฎี และวรรณกรรมที่เกี่ยวข้อง

ในการศึกษาข้อมูลเกี่ยวกับ แนวคิด ทฤษฎี และวรรณกรรมที่เกี่ยวข้องกับการพัฒนาระบบพัฒนาระบบการคำนวณหาวัตถุดิบเพื่อใช้ในการผลิตเครื่องประดับ ด้วยดอทเน็ตเทคโนโลยีนั้นผู้ศึกษาพบว่ามีแนวคิด ทฤษฎีและงานวิจัยต่าง ๆ ที่เกี่ยวข้องประกอบด้วยดังรายละเอียดตามลำดับดังนี้

3.1 การคำนวณหาวัตถุดิบ

โกศล ดีศีลธรรม 2547 ได้เขียนไว้ในหนังสือ การบริหารสินค้าคลังจากคลังสินค้าสู่ศูนย์กระจายสินค้าไว้ว่า การวางแผนความต้องการวัสดุ (MRP : Material Requirement Planning) คือการใช้ระบบคอมพิวเตอร์เพื่อช่วยในการควบคุมวัสดุและการวางแผนการผลิต ระบบวางแผนความต้องการวัสดุจะพิจารณาความต้องการวัสดุจนถึงระดับผลิตภัณฑ์ โดยคำนวณความต้องการส่วนประกอบของผลิตภัณฑ์ในแต่ละช่วงเวลา เพื่อจัดการสั่งผลิตหรือสั่งซื้อส่วนประกอบนั้นๆ นอกจากนี้ระบบวางแผนความต้องการวัสดุยังทำหน้าที่เป็นกลไกในการปรับปรุงเปลี่ยนแปลงตารางการผลิตเมื่อมีการทบทวนแผนงาน

MRP ทำเพื่ออะไร?

1) เพื่อระบุความต้องการวัสดุในแต่ละช่วงเวลา

2) เพื่อให้มั่นใจว่ามีวัสดุอย่างพอเพียงเมื่อต้องการ

3) เพื่อรักษาระดับวัสดุคงคลังที่ต่ำที่สุด

ความต้องการผลิตภัณฑ์ ความต้องการส่วนประกอบและ MRP

การจัดการความต้องการวัสดุประเภทส่วนประกอบผลิตภัณฑ์ (Component) มีความแตกต่างจากการจัดการผลิตภัณฑ์ (Finished Goods) กล่าวคือปริมาณความต้องการผลิตภัณฑ์ (Finished Goods) เกี่ยวเนื่องกับความต้องการของตลาด ตัวอย่างเช่น ปริมาณความต้องการของโต๊ะขึ้นอยู่กับคำสั่งซื้อของลูกค้า แต่ปริมาณความต้องการวัสดุประเภทส่วนประกอบผลิตภัณฑ์สามารถคำนวณได้จากปริมาณความต้องการผลิตภัณฑ์

ตัวอย่างเช่นโต๊ะ 1 ตัว ประกอบด้วยขาโต๊ะ 4 ขา เพราะฉะนั้น ถ้ามีความต้องการโต๊ะ 100 ตัว ปริมาณขาโต๊ะที่ต้องการเท่ากับ 4 * 100 = 400 ขา

ความต้องการผลิตภัณฑ์จะเป็นความต้องการชนิดอิสระ (Dependent Demand) ความต้องการของผลิตภัณฑ์แต่ละรายการไม่มีความสัมพันธ์กัน ส่วนความต้องการ

ส่วนประกอบผลิตภัณฑ์เป็นความต้องการชนิดไม่อิสระ (Independent Demand) ซึ่งจะขึ้นกับปริมาณความต้องการของผลิตภัณฑ์ที่ส่วนประกอบนั้นๆประกอบอยู่

เนื่องจาก MRP วางแผนความต้องการวัสดุถึงระดับส่วนประกอบผลิตภัณฑ์ ฉะนั้นก่อนทำ MRP ต้องทราบถึงส่วนประกอบของผลิตภัณฑ์นั้นๆ เพื่อให้สามารถคำนวณจำนวนความต้องการของทุกส่วนประกอบเมื่อมีความต้องการผลิตภัณฑ์ได้ ตัวอย่างเช่น ถ้ามีความต้องการโต๊ะ 100 ตัวในเช้าวันศุกร์ และเวลาที่ต้องการใช้ในการประกอบโต๊ะคือ3 วัน ฉะนั้นเช้าวันอังคารจะต้องมีพื้นโต๊ะ 100 ชิ้นและขาโต๊ะ 4 * 100 = 400 ขา เพื่อรอการประกอบ

เมื่อไรจึงจะใช้ MRP

1) MRP เหมาะกับการวางแผนการผลิตและการควบคุมวัสดุสำหรับผลิตภัณฑ์ที่มีความซับซ้อน มีส่วนประกอบหลายส่วน เนื่องจากระบบสามารถคำนวณหาความต้องการของส่วนประกอบผลิตภัณฑ์แต่ละส่วนในแต่ละช่วงเวลาได้

2) ระบบออกแบบเพื่อรองรับความต้องการวัสดุที่ไม่ต่อเนื่อง (Discrete) ตัวอย่างเช่น มีความต้องการโต๊ะ 100 ตัวในเช้าวันศุกร์ จะเห็นว่าโต๊ะ 100 ตัวต้องการพร้อมกันในเช้าวันศุกร์ ความต้องการไม่ได้มีอย่างต่อเนื่อง

3) ระบบเหมาะสำหรับการควบคุมวัสดุที่ใช้ในการวางแผนวางแผนการผลิตแบบทำตามสั่ง (Job shop) รวมทั้งการประกอบผลิตภัณฑ์ตามสั่ง (Assembly to order)
MRP การวางแผนความต้องการวัสดุ การบริหารการผลิต นิยามธุรกิจ

MRP : Material Requirement Planning ต้องการข้อมูล 3 อย่างที่ต้องมีความถูกต้องเพื่อที่จะคำนวณหา ชนิด และปริมาณของวัสดุทั้งวัตถุดิบ และบรรจุภัณฑ์ที่ใช้ในการผลิตที่ตรงกับความต้องอย่างพอดี ไม่มากไม่น้อยเกินไป อาจมีเผื่อของเสียได้บ้าง 3 อย่างที่ต้องมีความแม่นยำคือ

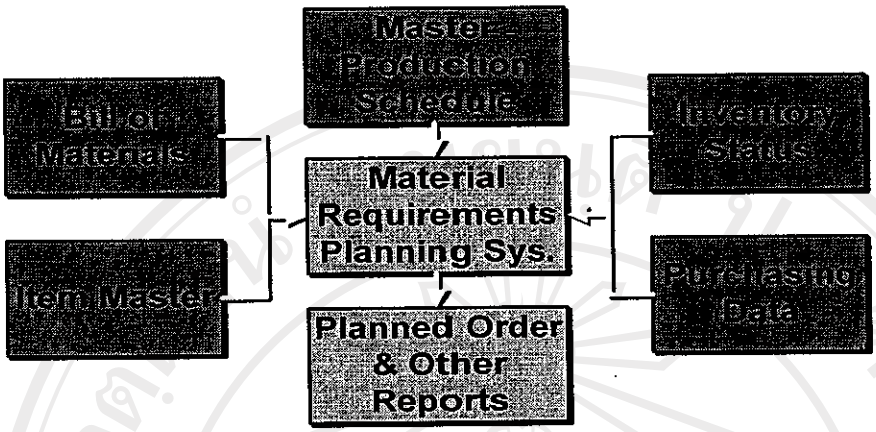
1) ยอดสินค้าคงคลัง (On Hand) คือการทำให้ Physical Stock : Logical Stock ซึ่งต้องเป็นแบบ Real Time แปลง่ายๆ คือ ทำให้ยอดของคงคลังที่นับได้จริง ตรงกับ ยอดของคงคลังในระบบสต๊อคการ์ด หรือในคอมพิวเตอร์ให้ตรงกันในทันทีทันใด (Real Time) หลังจากที่มีการโยกย้าย รับจ่ายวัสดุนั้นๆ บางแห่งใช้ระบบอัตโนมัติควบคุมของคงคลัง (Inventory) ก็จะได้ Real time ที่สมบูรณ์ บางแห่งใช้คนคีย์ป้อนเข้าคอมพิวเตอร์ ก็ขึ้นกับคนาจะป้อนเข้าไปเมื่อไหร่ และป้อนถูกไหม ยอดของคงคลัง ดูเหมือนจะเป็นเรื่องง่ายๆ แต่

ในบางโรงงานที่มีการจัดการไม่ดี ความแม่นยำของ Physical Stock ต่ำมากเมื่อเทียบกับ Logical Stock จนถึงขนาดที่แผนกวางแผนผลิตบอกว่าเวลา Run MRP จะคิดว่ายอดของคงคลังทั้งหมดเป็นศูนย์ เวลาคำนวณความต้องการวัสดุจากสูตรได้เท่าไหร่ ก็จะสั่งซื้อเท่านั้น โดยไม่ไปลบออกจากยอดของคงคลังเลย ที่คลังดังกล่าวมีของอยู่ 1000 ชนิด มีขนาด 2500 pallets ของมากองที่พื้นล้นแล้วล้นอีก ทำงานก็ยาก FIFO ก็ยาก ตอนนี้ต้องสร้างหลังคานอกชานเพิ่มแบบฉุกเฉินเพราะของวางนอกโกดังและเข้าหน้าฝนแล้ว ไม่มีใครสนใจทำให้ยอดของคงคลังให้ตรงเสียเงินเสียทองโดยไม่จำเป็น

2) BOM หรือ Bill of Material หรือรายการโครงสร้างวัสดุ ไม่เรียกว่าสูตรเพราะการผลิตสินค้าไม่ได้มีเฉพาะตัวสินค้าแต่ยังต้องมีบรรจุภัณฑ์ หรือ Processing aids หรือวัสดุที่ช่วยในการผลิต แต่ไม่ได้อยู่ใน Finished Goods หรือสินค้าที่สำเร็จที่พร้อมส่งมอบให้ลูกค้า

3) MPS : Master Plan Scheduling หรือแผนหลักของการผลิต ซึ่งได้มาจาก Demand Plan ที่ได้มาจาก Forcast และ Order ของลูกค้า เป็นการยากที่สุดที่ทำให้ MPS แม่นยำ โดยเฉพาะถ้าผลิตแบบ Make to Stock เพราะต้องได้จากการคาดการณ์ความต้องการของสินค้า (Forcast) เช่นสินค้าที่เป็นอุปโภค และบริโภคแต่ถ้าใช้คอมพิวเตอร์ทำ MRP ก็สามารถทำการคำนวณ และประมวลผลได้ไว ก็จะลดความเบื่อหน่ายของการวางแผนซ้ำ หรือเปลี่ยนแผนได้บ้าง แต่ถ้าแผนกวางแผนกำหนดแผนไปแล้ว ยืนยันไม่เปลี่ยนแผนกะทันหันระหว่างผลิต ก็จะไม่มีปัญหาเรื่องชนิด และจำนวนที่จะต้องใช้ในการผลิตว่าจะขาดแคลน บางโรงงานที่มีแผนฉุกเฉินเป็นประจำก็จะใช้วิธีมีจำยอดของคงคลังของวัสดุแต่ละตัวให้สูงหน่อยตลอดเวลา (Safety Stock หรือ Buffer Stock) เพราะเครื่องอาจเสีย Supplier ส่งของไม่ทันลูกค้าเปลี่ยนใจกระทันหัน

แต่การที่มี Safety Stock มากไปก็มีผลเสีย เพราะต้องเสียเงินที่ไปจัดซื้อมาเสียพื้นที่ในการเก็บของเสียหายเพราะเก็บไม่ดีหมดอายุเพราะเก็บนาย FIFO ไม่ดีเพราะของมากคลังเต็ม



รูปที่ 3.1 Input and Output ของระบบ MRP

MRP มีจุดประสงค์คือจัดซื้อวัสดุให้พอใช้ในการผลิตอย่างพอดี ไม่มากไปไม่น้อยไป ต้องการลดยอดของคงคลังให้ลดลงครับ

3.2 เริ่มต้นรู้จักกับ ASP.NET

"ASP .net" [Online] Available http://www.aspthai.net/aspnet/default.asp (20 มกราคม 2549) กล่าวว่า โลกในยุคปัจจุบันนี้กำลังพัฒนาไปสู่โลกแห่งการสื่อสารไร้พรมแดนในโลกแห่งเทคโนโลยีปัจจุบันนี้ หลายๆท่านคงคุ้นเคยหรือได้ยินเกี่ยวกับเทคโนโลยีและการบริการพิเศษต่างๆ มาบ้างแล้ว ไม่ว่าจากสื่อโฆษณา โทรทัศน์ บางท่านอาจจะเป็นผู้ใช้เทคโนโลยีนี้อย่างไม่รู้ตัว การเช็คเมลล์ เล่นอินเตอร์เน็ต รวมทั้งจองตั๋วชมภาพยนตร์ผ่านทางโทรศัพท์มือถือ สิ่งเหล่านี้ล้วนเป็นเทคโนโลยีที่ทันสมัยและน่าทึ่งอย่างยิ่ง แล้วตัวคุณจะรู้หรือไม่ว่าสิ่งเหล่านี้เกิดขึ้นได้อย่างไรและใช้สิ่งใดในการสร้าง

.NET แนวคิดแห่งโลกเทคโนโลยีสมัยใหม่

.NET (อ่านว่า ด็อทเนต) คือแนวคิดหนึ่งที่ไมโครซอฟท์ภูมิใจนำเสนอ โดย .NET ตัวนี้ไม่ได้เกี่ยวข้องกับโดเมนเนมของเว็บใดๆ ทั้งสิ้น แต่ .NET ตัวนี้ หมายถึง การนำเอาอุปกรณ์ทุกอย่างบนโลกมาเชื่อมโยงต่อกันเหมือนตาข่าย (net = ตาข่าย) ซึ่งหากว่าทำสำเร็จแล้วไม่ต้องนึกเลยว่าไมโครซอฟท์จะได้เป็นเจ้าแห่งเทคโนโลยีอย่างไม่ต้องสงสัย แต่เรื่องมันไม่ง่ายอย่างที่คิด เพราะอุปกรณ์ต่างๆเหล่านั้นล้วนถูกออกแบบมาต่างๆกัน การที่มันจะ

ติดต่อสื่อสารกันรู้เรื่องนั้น ย่อมเป็นเรื่องที่เป็นไปได้ยาก ไมโครซอฟท์เล็งเห็นจุดนี้ จึงได้พยายามที่จะคิดค้นสิ่ง ที่เป็นมาตรฐานขึ้น เพื่อให้อุปกรณ์ทุกๆชนิดทั่วโลกติดต่อสื่อสารกันได้อย่างรู้เรื่อง จึงไม่ใช่เรื่องแปลกถ้าในอนาคตเราเปิด Website เล่นอินเตอร์เน็ตด้วยอุปกรณ์อื่นๆนอกเหนือจากคอมพิวเตอร์

ASP Web Services คืออะไร

หลายๆคนอาจเคยได้ยินคำๆนี้มากันบ้างแล้ว แต่ก็มีหลายคนสงสัยว่ามันคืออะไรกันแน่ ซึ่งชื่อ Web Service นั้นแปลตามตัวก็คือการใช้บริการต่างๆ ผ่านทางเว็บนั่นเอง ซึ่งก็ตรงตามจุดประสงค์ของไมโครซอฟท์ ที่ต้องการจะทำให้ทุกอย่างสามารถใช้งานเชื่อต่อกับอินเตอร์เน็ตได้นั่นเอง คำๆนี้จึงกลายเป็นคำฮิตติดปากมาจนถึงทุกวันนี้ แต่จริงๆแล้ว Web Service ก็คือการเรียกใช้งานชุดคำสั่งในระยะไกล ซึ่งชุดคำสั่งเหล่านี้ไม่ใช่แค่ชุดคำสั่งธรรมดา แต่เป็นชุดคำสั่งที่เขียนขึ้นมาเป็นโปรแกรมเหมือนกับการใช้งานของ Client เลยทีเดียว ทำให้เราไม่ต้องไปติดตั้งโปรแกรมต่างๆภายในเครื่องคอมพิวเตอร์ของเราเลย เมื่อใดอยากใช้งานก็ต่ออินเตอร์เน็ตเข้าไปใช้บริการในเว็บไซท์ผู้ผลิตได้ทันที โดยอาจมีการเรียกเก็บค่าบริการเป็นครั้งๆไป ซึ่งจะช่วยลดปัญหาในการละเมิดลิขสิทธิ์ และชุดคำสั่งเหล่านี้จะทำให้ ASP.NET มีบทบาทมากทีเดียว

What's .NET Framework?

อย่างที่ได้กล่าวไปแล้วในตอนต้นว่าไมโครซอฟท์ต้องการที่จะสร้างอะไรที่เป็นมาตรฐานขึ้นมา เพื่อให้ทุกสิ่งทุกอย่าง เพื่อให้ทุกสิ่งทุกอย่างสามารถติดต่อสื่อสารกันได้หมด โดยคิดค้นระบบซึ่งหมายมั่นปั้นมือว่าจะให้เป็นระบบมาตรฐาน ระบบนี้คือ .NET Framework ซึ่งระบบนี้ไม่ใช่ระบบปฏิบัติการ (OS) แต่เปรียบเสมือนโปรแกรมหนึ่งที่จะสามารถสร้างสภาวะแวดล้อมหนึ่ง ซึ่งสามารถทำงานในระบบ .NET นี้ได้ ในอนาคตไมโครซอฟท์ก็หวังที่จะนำเอาระบบนี้ไปติดตั้งลงบนอุปกรณ์ทุกชนิด เพื่อทำให้อุปกรณ์ทุกอย่างมีระบบๆหนึ่งที่เหมือนกันหมด โดย .NET Framework นั้นมีส่วนประกอบภายในแบ่งออกเป็น 3 ชั้นใหญ่ๆ คือ

1) Programming Language เป็นรูปแบบของภาษาที่ออกแบบมาเพื่อให้สามารถทำงานใน สภาวะที่เป็น .NET ได้โดยที่ทางไมโครซอฟท์ได้เปิดตัวภาษาหลักๆที่จะใช้พัฒนาบน .NET นี้ 3ภาษา

1.C# เป็นภาษาใหม่ที่ไมโครซอฟท์พัฒนามาจาก C++ กับ Java เป็น

หลัก

2. VB.NET เป็นภาษาที่พัฒนามาจาก Visual Basic ในเวอร์ชั่น 6.0

3. JScript.net เป็นภาษาที่พัฒนามาจาก JScript ซึ่งเป็น JavaScript ใน เวอร์ชั่นของ ไมโครซอฟท์

2) Base Classes Library Library นั้นเปรียบเสมือนชุดคำสั่งสำเร็จรูปย่อยๆที่เพิ่มเข้ามา ซึ่งส่วนใหญ่จะเป็นชุดคำสั่งที่ต้องใช้งานอยู่เป็นประจำ ดังนั้นจึงมีผู้คิดค้นเพื่ออำนวยความสะดวกในการเขียนโปรแกรม ซึ่ง library ในภาษาต่างๆส่วนใหญ่จะอยู่ในรูปแบบไฟล์ incould แต่ถ้าเป็น ASP สิ่งที่เป็น library ก็คือ คอมโพรเนนท์ต่างๆ นั่นเอง ซึ่งภายในระบบ .NET จะสร้างสิ่งที่เรียกว่าเป็น library พื้นฐานขึ้น ทำให้ไม่ว่าจะใช้ภาษาใดในการพัฒนาโปรแกรมก็สามารถที่จะเรียกใช้ library ที่เป็นตัวเดียวกันได้หมด

3) Common Language Runtime (CLR) นับเป็นสิ่งสำคัญแทบจะที่สุดของระบบ .NET นี้ก็ว่าได้ เพราะ CLR ที่ว่านี้มีหน้าที่ทำให้โปรแกรมที่เขียนขึ้นมาด้วยภาษาต่างๆกันกลายเป็นภาษารูปแบบมาตรฐานเดียวกัน ทั้งหมด เราเรียกภาษาที่ว่านี้ว่า Intermediate language (IL) ซึ่งเมื่อต้องการที่จะรันโปรแกรมใด CLR ที่ว่านี้จะตรวจสอบเครื่องที่รันว่ามีสภาวะแวดล้อมการทำงานเช่นใดหลังจากนั้นก็จะคอมไพล์เป็นโปรแกรมที่เหมาะสมต่อการทำงานของเครื่องนั้น ทำให้เราสามารถใช้งานโปรแกรม ต่างๆได้อย่างมีประสิทธิภาพสูงสุดในแต่ละเครื่อง

Dot NET Framework มีดีตรงไหน

ประโยชน์และข้อดีของ Dot NET Framework นั้นพอจะสรุปออกมาได้เป็นข้อๆดังนี้

1) เป็นระบบที่มีไลบรารีที่เป็นมาตรฐานเดียวกัน เนื่องจากมีไลบรารีที่เป็นมาตรฐานเดียวกัน ทั้งหมดทำให้เราไม่ต้องกังวล ว่าภาษาที่ใช้เขียนนั้นมีไลบรารีตัวนั้นตัวนี้หรือไม่ รวมทั้งไม่ต้องคอยกังวลว่าถ้าใช้ไลบรารีของภาษาหนึ่งแล้วอีกภาษาหนึ่งจะไม่มีไลบรารีตัวนั้น

2) ไม่ขึ้นกับระบบปฏิบัติการ (OS) เนื่องจากระบบปฏิบัติการ ที่แต่ละบุคคลหรือองค์กรใช้นั้นย่อมไม่เหมือนกัน แต่ภายใน .NET Framework จะไม่มีปัญหานี้ของเพียงแค่มีระบบ .NET Framework ก็จะทำให้สามารถใช้งานโปรแกรมต่างๆได้ ซึ่งเป็นข้อดีตรงที่เราจะสามารถใช้โปรแกรมต่างๆได้ทุกระบบปฏิบัติการ

3)ใช้ในการพัฒนาได้ทุกภาษา ทำให้เราไม่ต้องคอยมาศึกษาภาษาใหม่ๆเมื่อต้องการสร้างโปรแกรมในแต่ละครั้ง นอกจากนั้นเรายังสามารถเลือก ใช้ภาษาที่เราถนัด

ที่สุดในการพัฒนาโปรแกรมต่างๆได้ด้วย

4) มีการควบคุมสิ่งแวดล้อมในการทำงานเป็นอย่างดี เนื่องจากเป็นระบบที่เป็นมาตรฐานทำให้การควบคุมจัดสรรระบบต่างๆ ทำได้ง่ายขึ้น ไม่ว่าจะเป็นการจัดสรรหน่วยความจำ ด้านการใช้งานเครื่องก็มีความรวดเร็วมากขึ้น ลดโอกาสที่เครื่องจะแฮงก์ได้เป็นอย่างดี

5) ความปลอดภัยที่มีมากขึ้น Dot NET Framework สามรถกำหนดสิทธิการใช้งานหรือ Permission ของผู้ใช้งานได้มากขึ้นทำให้สมารถกำหนดว่า จะให้โปรแกรมในส่วนใดใช้งานได้หรือไม่ได้ แล้วแต่เฉพาะบุคคล

ทั้งหมดนี้เป็นเพียงประโยชน์ส่วนหนึ่งในแนวคิดของไมโครซอฟท์ที่กำลังจะพัฒนาให้สำเร็จเท่านั้น บางข้อทำได้สำเร็จแล้ว แต่บางข้อก็ยังไม่สำเร็จดังนั้นจึงต้องคอยรอดูว่า ฝันของไมโครซอฟท์จะเป็นจริงและสำเร็จได้เมื่อไร

ASP.NET ภาษาแห่งอนาคตกับ Dot NET

ASP .NET หรืออีกชื่อหนึ่งว่า ASP+ ซึ่งเป็นชื่อที่ไมโครซอฟท์ใช้เรียกในตอนแรก ถือว่าเป็นASP เวอร์ชั่นล่าสุดต่อจาก ASP 3.0 แต่คงไม่สามารถพูดได้เต็มปากว่า ASP.NET พัฒนามาจาก ASP เพราะรูปแบบ และไวยากรณ์ต่างๆ และภาษาที่นำมาใช้งานนั้นต่างจากเดิมแทบทั้งสิ้น แทบจะเรียกได้ว่ายกเครื่องใหม่เลยทีเดียว น่าจะพูดได้ว่า ASP.NET เป็นอีก Generation หนึ่งของ ASP มากกว่า เรามาลองดูกันว่าใน ASP.NET นั้นมีอะไรที่แตกต่างจาก ASP รุ่นก่อน ๆบ้าง

1) ใช้ภาษาใดๆ ในการเขียนสคริปต์ก็ได้ : จากเดิมที่เราสามรถใช้ได้เฉพาะภาษาที่เป็นสคริปต์ของ VBScript และ JScript แต่ใน ASP.NET เราสามารถที่จะใช้ภาษาที่มีรูปแบบของภาษาเต็มๆ ซึ่ง ในเบื้องต้น มี 3 ภาษาคือ C#, VB.NET และ JScript.Net ที่ออกมาเป็นมาตรฐาน แต่ในอนาคตไมโครซอฟท์มีแผนที่จะเพิ่มตัวแปลภาษาให้ครบทุกภาษา

2) มีความยืดหยุ่นในการเขียนโปรแกรมมากขึ้น โดยที่เราสามารถใช้ภาษาในการเขียน ASP.NET ได้มากกว่า 1 ภาษาภายในไฟล์เดียวกัน ทำให้สามารถเลือกรูปแบบของภาษาที่ง่ายที่สุดต่อการเขียน ในแต่ละส่วนได้

3) ลักษณะการแปลภาษาและนามสกุลไฟล์เปลี่ยนไปใน ASP เวอร์ชั่นก่อนๆ มีลักษณะการแปลภาษาเป็นแบบอินเตอร์พรีเตอร์ (Interpreter) คือการจะทำคำสั่งใดค่อยแปลคำสั่งนั้น แต่ในเวอร์ชั่น .NET นี้จะมี ลักษณะเป็นคอมไพเลอร์ (Compiler) คือการ

แปลคำสั่งรวมทั้งโปรแกรม นอกจากนี้นามสกุลของไฟล์ก็มีการเปลี่ยนแปลงจากเดิมที่ใช้นามสกุลไฟล์เป็น "*.asp" เป็น "*.aspx "

4) รูปแบบและการใช้งานคอมโพเนนต์ที่ง่ายขึ้น รูปแบบของคอมโพเนนต์จะเน้นไปที่ XML มากที่สุด และที่สำคัญคือการใช้งานคอมโพเนนต์ใน ASP.NET นั้นเราสามารถอัพโหลดไฟล์ไปไว้ในไดเร็คตอรีที่ผู้ดูแลเซิร์ฟเวอร์ (Admin) กำหนดหลังจากนั้นคอมโพเนนต์จะติดตั้งตัวเองโดยอัตโนมัติ ลดปัญหาที่เกิดจาก ASP เวอร์ชั่นก่อนๆได้เป็นอย่างดี เนื่องจากใน ASP เวอร์ชั่นก่อนนั้นการติดตั้งคอมโพเนนต์กระทำได้เพียงผู้ดูแลเซิร์ฟเวอร์เพียงคดเดียวเท่านั้น ทำให้เวลาต้องการใช้คอมโพเนนต์ต่างๆที่เซิร์ฟเวอร์ไม่มี จึงเป็นเรื่องที่ลำบาก

5) มีไลบรารีให้เลือกใช้ได้มากขึ้น ใน ASP เวอร์ชั่นก่อนๆนั้นแอพพลิเคชั่นบางอย่างสร้างได้ไม่สะดวกนัก ต้องอาศัยคอมโพเนนต์ต่างๆมากมาย แต่ใน ASP.NET นั้นได้เพิ่มไลบรารีในส่วนเหล่านี้ให้กลายเป็นพื้นฐานของการใช้งาน

6) มีคอนโทรลทำให้การใช้งานในบางสิ่งง่ายขึ้น เป็นส่วนพิเศษที่เพิ่มเติมมาจาก ASP รุ่นก่อนๆที่ไม่มีส่วนที่เรียกว่า คอนโทรล ซึ่งคอนโทรลนี้จะช่วยให้เราสามารถสร้างเว็บไซท์ได้อย่างง่ายและมีประสิทธิภาพมากขึ้น จึงไม่ต้องกังวลว่าบราวเซอร์รุ่นนั้นรุ่นนี้จะรองรับกับภาษาที่เราเขียนหรือไม่

7) สามารถร้องขอข้อมูลจากเซิร์ฟเวอร์ได้ ใน ASP เวอร์ชั่นก่อนๆเซิร์ฟเวอร์สามารถเรียกขอข้อมูลได้จากเครื่องผู้ใช้เท่านั้นแต่ใน ASP.NET เครื่องเซิร์ฟเวอร์สามารถเรียกขอข้อมูลจากเครื่องเซอร์เวอร์ด้วยกันได้

8) ไม่ต้องต่อ Hardware เนื่องจากเป็นระบบใน .NET Framework ดังนั้นจึงมีคุณสมบัติของ Common Languare Runtime (CLR) ทำให้มีการคอมไพล์โปรแกรมเป็นภาษามาตรฐานที่เรียกว่า IL ก่อน ดังนั้นไม่ว่าคุณจะเล่นเครื่องปาล์มหรือโน้ตบุ๊ค PDA ก็ไม่เกิดปัญหา

9) ง่ายต่อการหาจุดผิดพลาดในการเขียนโปรแกรม หากเป็น ASP รุ่นก่อนเวลาเกิดความผิดพลาด (Error) เครื่องจะบอกแค่ว่าเป็นความผิดพลาดชนิดใดบรรทัดไหน แต่ใน ASP.NET นี้เครื่องจะแสดงรายละเอียดที่มากขึ้น พร้อมแนวทางแก้ไข

10) มีการตรวจสอบเหตุการณ์ต่างๆได้ภายในเว็บเพจ มีการตรวจสอบเหตุการณ์ต่างๆ ตั้งแต่โหลดหน้าเว็บเพจไปจนถึงปิดหน้าเว็บเพจลง ทำให้เราสามารถเขียนโปรแกรมกำหนดเหตุการณ์ต่างๆได้ง่ายขึ้น

11) แยกส่วนที่เป็น HTML กับ ASP ออกมาอย่างชัดเจน ในเวอร์ชั่นก่อนๆส่วนที่เป็น HTML กับ ASP จะเขียนปนกันไปมา แต่ในเวอร์ชั่นนี้จะแยกส่วนกันอย่างชัดเจนว่าส่วนไหนเป็น HTML และส่วนไหนเป็น ASP

นอกจากนี้ยังมีประโยชน์อีกมากมายที่ยังไม่ได้เขียนเอาไว้ แต่ประโยชน์เพียงเท่านี้ก็คงเพียงพอที่จะทำให้ทุกท่านอยากจะลองใช้งานดูว่าสมคำล่ำลือหรือไม่

3.3 การเขียนโปรแกรมเชิงวัตถุ

"การเขียนโปรแกรมแบบ OOP" [Online] Available http://www.mobi2you.com/J2ME/j2me_03.htm (21 มกราคม 2549) กล่าวว่า ในความคิดแบบเชิงวัตถุ (Object Oreinted Thinking) นั้น "Object" จะหมายถึงอะไรก็ได้ที่ปรากฏอยู่รอบๆ ตัวเรา ภายใต้สภาพของโลกแห่งความเป็นจริงที่เราอาศัยอยู่ ซึ่งอาจเป็นจริงได้ทั้ง "สิ่ง" ที่อยู่ในลักษณะจับต้องได้ ซึ่งสอดคล้องกับสิ่งที่มีลักษณะเชิงกายภาพ เช่น ปากกา เครื่องคอมพิวเตอร์ คน รถยนต์ หรืออยู่ในลักษณะที่จับต้องไม่ได้ที่สามารถแบ่งออกได้ดังนี้

1) บทบาท (Role) ของคนหรือองค์กร เช่น ผู้ป่วย พนักงาน ลูกค้า เป็นต้น

2) สิ่งที่อุบัติขึ้น (Incident) หรือเหตุการณ์ที่เกิดขึ้นของ "Object" เหล่านี้จะมีการทำธุรกรรม (Transaction) หรือการติดต่อกัน (Contract) ระหว่าง "Object" ตั้งแต่ 2 "Object" ขึ้นไปภายในแบบจำลองระบบ (System Model) ที่พัฒนา เช่น การซื้อจะมีความสัมพันธ์ระหว่างผู้ซื้อ ผู้ขาย สิ่งของที่ซื้อ การแต่งงาน จะเป็นความสัมพันธ์ระหว่างผู้ชายกับผู้หญิง

หลักการพื้นฐานของการโปรแกรมเชิงวัตถุ

1) ออบเจ็กต์ (Object) จะต้องประกอบด้วยสองสิ่งต่อไปนี้เสมอคือ คุณลักษณะ (Attributes) และพฤติกรรม (Behaviors) เมื่อมองจากภายนอก หรืออาจจะเปรียบเสมือนกล่องๆ หนึ่ง ซึ่งภายในกล่องมีข้อมูล และเมธอดรวมกันอยู่ และออบเจ็กต์นี้จะทำการรับและส่งข้อมูลระหว่างออบเจ็กต์ เพื่อทำการใช้งานข้อมูลและเมธอดที่อยู่ภายใน

2) เมธอด (Method) เป็นการทำงานที่สำคัญหรือเป็นวิธีการกระทำที่สามารถทำงานกับออบเจ็กต์ได้ แต่ละคลาสจะมีเมธอดของตัวเอง โดยการทำงานจะเริ่มจากการส่ง Message ไปยังออบเจ็กต์ที่ต้องการ (Receiver Object) เพื่อเรียกใช้เมธอดที่อยู่ภายในออบเจ็กต์นั้นๆ

3) การถ่ายทอดคุณสมบัติ (Inheritance) คือ การสร้างคลาสใหม่ขึ้นมา โดยมีการสืบทอดคุณสมบัติพื้นฐานที่มาจากคลาสเดิม แต่จะมีข้อมูล หรือเมธอดเพิ่มขึ้นจากคลาส

เดิม ถ้าคลาส B เป็นสับคลาสของคลาส A ออบเจ็กต์ที่เป็นอินสแตนซ์ของคลาส B จะมีคุณสมบัติพิเศษกว่าอินสแตนซ์ของคลาส A แต่อย่างน้อยที่สุด จะต้องมีคุณสมบัติเหมือนอินสแตนซ์ของคลาส A

4) การห่อหุ้ม (Encapsulation) คือการรวมกันของโครงสร้างข้อมูล กับฟังก์ชันเกิดเป็นออบเจ็กต์ใหม่ที่มีความสามารถในการซ่อนข้อมูลจากระบบภายนอกได้ ทำให้ข้อมูลมีความปลอดภัยมากขึ้น

5) โพลิมอร์ฟิซึม (Polymorphism) คือการที่เราส่ง Message ที่เหมือนกัน ไปในออบเจ็กต์ที่ต่างกัน แต่ละออบเจ็กต์จะตอบสนองออกมาไม่เหมือนกัน ตามแต่ชนิดและหน้าที่ของออบเจ็กต์ ความสามารถที่ใช้ Message เหมือนกัน สำหรับการกระทำที่เหมือนกัน ไปยังออบเจ็กต์ต่างชนิดกัน มีลักษณะเหมือนกับการที่มนุษย์คิดในการแก้ปัญหาหนึ่งๆ

6) ไดนามิบายดิง (Dynamic Binding) คือ การนำโปรแกรมย่อยๆ มาประกอบให้ใช้งานในขณะ Run Time โดยในขณะ Complie Time นั่นจะเก็บโปรแกรมในรูปแบบของคลาสต่างๆ ไว้ เพื่อไม่ให้เกิดความยุ่งยาก ซับซ้อน ต่อจากนั้น เมื่อนำมาใช้ในขณะ Run Time เมื่อมีการเรียกใช้คลาสนั้นๆ จะทำการนำคลาสนั้นมาไว้ในส่วนของโปรแกรม และเมื่อใช้งานเสร็จแล้วจะถูกลบออกจากหน่วยความจำ

7) โอเวอร์ไรด์ (Override) คือการที่เราสร้างซับคลาสขึ้นมาใหม่ และมีการสร้างเมธอดที่ซ้ำกับเมธอดที่เป็นของซุปเปอร์คลาส

8) โอเวอร์โหลด (Overload) คือการที่เราสร้างเมธอดชื่อเหมือนกัน แต่รับ Parameter ต่างกัน ภายในคลาสเดียวกัน

คุณสมบัติของ OOP

1) การห่อหุ้ม (Encapsulation / Information Hiding) เป็นการนำเอาข้อมูลที่ได้จากการจำกัดขอบเขตซึ่งมองเฉพาะสิ่งที่เกี่ยวข้องมาเก็บรวมกับฟังก์ชันการทำงานที่จะใช้กับข้อมูลนั้น โดยข้อมูลที่อยู่ในออบเจ็กต์จะถูกเรียกใช้จากฟังก์ชันที่อยู่ในออบเจ็กต์เดียวกันเท่านั้น เปรียบเหมือนกับว่าทั้งสองข้อมูลและฟังก์ชั่นการทำงานถูกห่อหุ้มไว้ด้วยกัน เพื่อเป็นการปกป้องข้อมูลโดยไม่ให้ออบเจ็กต์อื่นเข้ามาทำการเปลี่ยนแปลงข้อมูลก่อนได้รับอนุญาต

ซึ่งช่วยแก้ปัญหาการเขียนโปรแกรมแบบโครงสร้าง ซึ่งหลายฟังก์ชันจะมีการใช้ข้อมูลร่วมกันทำให้การทดสอบและการแก้ไขโปรแกรมทำได้ลำบาก เนื่องจากข้อมูลอาจ

ถูกเปลี่ยนแปลงโดยฟังก์ชันอื่นได้ง่าย และเมื่อจำเป็นต้องแก้ไขโครงสร้างข้อมูลก็อาจต้องแก้ไขโปรแกรมเกือบทั้งหมด ซึ่งต่างจากหลักการของโปรแกรมเชิงวัตถุ ที่มีคุณสมบัติของ Encapsulation ทำให้ข้อมูลและฟังก์ชันการทำงานถูกผนึกเข้าด้วยกัน ข้อมูลจะถูกเปลี่ยนแปลงจากฟังก์ชันการทำงานที่อยู่ภายในออบเจ็กต์เท่านั้น ทำให้การตรวจสอบความผิดพลาดทำได้ง่าย เนื่องจากเราจะทำเฉพาะภายในออบเจ็กต์นั้นเพียงอย่างเดียว และเมื่อมีการแก้ไขโครงสร้างข้อมูลก็ไม่มีผลกระทบต่อส่วนอื่นของโปรแกรม เพราะผู้ที่ทำการแก้ไขจะทำเฉพาะภายในออบเจ็กต์เท่านั้น

2) การถ่ายทอดคุณสมบัติ (Inheritance) เป็นการสร้างคลาสใหม่ซึ่งได้รับการถ่ายทอดข้อมูลและฟังก์ชันการทำงานจากคลาสเดิม โดยเรียกคลาสเดิมว่าซุปเปอร์คลาส (Super class) หรือ Parent class และเรียกคลาสใหม่ว่าซับคลาส (Sub Class) ซึ่งซับคลาสอาจจะมีข้อมูลและฟังก์ชันเพิ่มเติมจากซุปเปอร์คลาสก็ได้ ซึ่งมีข้อดีคือเป็นการเพิ่มความถูกต้องของข้อมูล (Consistancy) ให้กับคลาสเพียงแต่เปลี่ยนข้อมูลหรือฟังก์ชันที่ซุปเปอร์คลาส ก็จะมีผลให้ค่าที่ซับคลาสเปลี่ยนไปด้วยและเป็นการส่งเสริมการนำออบเจ็กต์กลับมาใช้ใหม่

3) โพลิมอร์ฟีซึม (Polymorphism) เป็นการส่ง Message เดียวกันให้กับออบเจ็กต์ที่ต่างกันเป็นผลทำให้ออบเจ็กต์แสดงพฤติกรรมที่แตกต่างกัน มีข้อดีคือ เป็นการสนับสนุนการนำโปรแกรมกลับมาใช้ใหม่ และสามารถเปลี่ยนแปลงโปรแกรมได้ในระหว่างการพัฒนา รูปแบบของโพลิมอร์ฟีซึมมีอยู่ 2 แบบคือ

1. ในคลาสที่ต่างคลาสกัน มีชื่อฟังก์ชันการทำงาน (Operation) ชื่อเดียวกัน ขึ้นอยู่กับการส่งข้อความให้กับออบเจ็กต์ใด ซึ่งการมีชื่อฟังก์ชันการทำงานเหมือนกันไม่จำเป็นต้องมีการทำงานที่เหมือนกัน

2. การรับการสืบทอดสามารถนำออบเจ็กต์มาดัดแปลงแก้ไขได้ โดยใช้ชื่อฟังก์ชันการทำงานเดิม

ข้อดีของการเขียนโปรแกรมแบบ OOP

1) เป็นการสร้างแนวคิดในขั้นตอนวิเคราะห์ ออกแบบตลอดจนการติดตั้งระบบของซอฟต์แวร์

2) ทำให้ขั้นตอนการวิเคราะห์ระบบ ออกแบบ จนถึงการติดตั้งระบบของซอฟต์แวร์เป็นไปแนวทางเดียวกันทั้งหมด ผู้ใช้งานสามารถเปลี่ยนแปลงความต้องการ (Requirement) ได้ทุกจุดโดยไม่ต้องเริ่มต้นทำใหม่

3) เป็นการสนับสนุนการนำกลับมาใช้ใหม่ และเพิ่มความสามารถในการทำงานของซอฟต์แวร์

4) เป็นการลดระยะเวลาในการพัฒนาเนื่องจากช่วยแก้ปัญหาระบบที่ซับซ้อน

3.4 การเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML

UML-Unified Modeling Language" [Online] Available http://www.itmelody.com/tu/uml1.htm (5 มกราคม 2549) ในการออกแบบโปรแกรมเชิงวัตถุเครื่องมือที่ใช้ในการวิเคราะห์ระบบงานในการวิจัยครั้งนี้ จะใช้ UML (Unified Modeling Language) ในการพัฒนาโมเดล

เป้าหมายของโมเดลที่ใช้ในการพัฒนาซอฟต์แวร์ คือ

1) โมเดลช่วยให้เราสามารถมองเห็นภาพของระบบงานได้ชัดเจนขึ้นว่าระบบงานจออกมาในลักษณะไหน หรือมองเห็นภาพว่าเราต้องการให้ระบบออกมาในทิศทางใด

2) โมเดลจะทำให้เราสามารถระบุถึงโครงสร้างและพฤติกรรมของระบบงานที่จะพัฒนาได้

3) โมเดลเป็นเหมือนเทมเพล็ตให้แก่การสร้างระบบงานจริงๆ

4) โมเดลช่วยให้เราสามารถทำให้การตัดสินใจในเรื่องต่างๆ อยู่ในรูปของเอกสารอ้างอิงได้ง่ายขึ้น

UML คืออะไร

UML (Unified Modeling Language) คือ โมเดลมาตรฐานที่ใช้หลักการออกแบบ OOP (Object Oriented Programming) รูปแบบของภาษา UML จะมี Notation ซึ่งเป็นสัญลักษณ์ที่นำไปใช้ใน Model ต่างๆ UML จะมีข้อกำหนดกฎระเบียบต่างๆ ในการโปรแกรม โดยกฎระเบียบต่างๆ จะมีความหมายต่อการเขียนโปรแกรม (Coding) ดังนั้นการใช้ UML จะต้องทราบความหมายของ Notation ต่างๆ เช่น Generalize Association Dependency Class และ Package สิ่งเหล่านี้มีความจำเป็นยิ่งต่อการตีความหมายขอการออกแบบและ Design ระบบก่อนนำไป Implement ระบบงานจริง ในปัจจุบันมีเครื่องมือมากมายที่สามารถแปลง Model UML เป็น Code ภาษาต่างๆ ยกตัวอย่างเช่น ภาษา Java Power Builder และ VB

ทำไมต้องใช้ UML

1) UML ได้รวมข้อดีของโมเดลต่างๆ เอาไว้ได้แก่

1. Data Model ซึ่งนำมาจากโมเดล OMT ของ James Rambaugh ซึ่ง Rambaugh เน้นหนักมากในเรื่องของข้อมูลโดยเอาแนวความคิดมาจาก ER-Diagram

2. Business Model หรือเวิร์กโฟลว์ (Work Flow) คล้ายกับ Data Flow Diagram (DFD) แต่ดีกว่าในเรื่องของ Sequence Loop Check if Condition

3. Object Model คือสามารถที่จะสร้างออปเจ็กต์ในแบบต่างๆ ได้

4. Component Model เป็นโมเดลที่มีแนวความคิดว่าทำอย่างไรจึงจะผลิตซอฟต์แวร์ให้เหมือนการผลิตฮาร์ดแวร์นั่นคือสามารถที่จะปั๊มแต่ละส่วนออกมาประกอบกันในลักษณะของคอมโพเนนต์ได้ โดยเริ่มจากชิ้นส่วนที่เล็กที่สุดมาประกอบกันให้เป็นชิ้นใหญ่ขึ้นเรื่อยๆ เมื่อเทียบกับการผลิตฮาร์แวร์แล้วก็เหมือนกับการผลิตไอซี (IC) เป็นตัวๆ แล้วนำมาประกอบใช้งานตามแต่ผู้พัฒนาต้องการ

การนำเอาข้อดีของโมเดลต่างๆ มารวมกันนั้นทำให้ภาษา UML เป็นวิธีการที่มีช่องโหว่น้อยกว่าวิธีการอื่นๆ อย่างไรก็ตาม การทำเช่นนั้นก็เป็นการเพิ่มความซับซ้อนในการศึกษาและการใช้งานให้มากขึ้นตามไปด้วย ซึ่งเป็นสาเหตุหนึ่งสำหรับผู้ที่ไม่คุ้นเคยกับภาษาสัญลักษณ์อาจจะต้องใช้เวลานานพอสมควรในการศึกษาจนสามารถนำไปใช้งานจริงได้

2) เป็นภาษาที่เป็นมาตรฐานเปิด (Open Standard) ของทุกภาษาในปัจจุบันไม่ว่าจะเป็น Java VisualAge ผลิตภัณฑ์ใหม่ๆ ของ Microsoft ล้วนแต่สนับสนุน UML

3) ภาษา UML คลอบคลุมทุกส่วนในวงจรชีวิต (Life Cycle) ของการพัฒนาระบบตั้งแต่ขั้นตอนของการหาความต้องการของระบบ (Requirement) การออกแบบระบบ (Design) การนำไปใช้งานจริง (Implementation) การติดตั้งระบบ (Installation) ไปจนถึงขั้นตอนของการจัดทำเอกสาร (Documentation) และถึงว่าระบบงานนั้นจะมีการใช้เทคโนโลยีหลายๆ อย่างร่วมกันก็ยังคงสามารถนำภาษา UML ไปประยุกต์ใช้งานได้

4) เป็นภาษาที่มีความสมดุลในแง่ของความเรียบง่ายและความซับซ้อน คือไม่ยากเกินไปที่จะเรียนรู้และนำไปใช้งานจริง และก็สามารถนำไปใช้กับงานที่ซับซ้อนมากๆ ได้ด้วย

5) บริษัทชั้นนำและอุตสาหกรรมต่างๆ ให้การยอมรับและให้การสนับสนุน ไม่ว่าจะเป็น Rational Software Corporation HP รวมทั้งโปรดักส์ต่างๆ เช่น Websphere VisualAge ของบริษัทไอบีเอ็ม เป็นต้น กระทั่งบริษัทไมโครซอฟต์ ถึงกับลงทุนซื้อส่วน

หนึ่งของ UML มาไว้ใน VisualBasic ที่เรียกว่า VisualModeller โดยเน้นหนักในส่วนของการออกแบบระบบ (Design) และการพัฒนาระบบไปใช้งานจริง (Implementation)

Use Case Driven

กระบวนการการพัฒนาโครงการใดๆ นั้น มีวัตถุประสงค์เพื่อทำการสนับสนุนการทำงานของผู้ใช้เป็นหลัก ทั้งนี้ผู้ใช้ไม่ได้หมายถึงคนเพียงอย่างเดียว อาจจะหมายถึงระบบอื่นๆ ที่อยู่ภายนอกระบบก็ได้ ผู้ใช้จะกระทำกิจกรรมใดๆ ต่อระบบ ระบบจะส่งผลลัพธ์ให้ผู้ใช้ กระบวนการตอบสนองของระบบอย่างมีลำดับเพื่อให้ได้งานตามต้องการในลักษณะนี้เรียกว่า Use Case Use Case จึงเป็น Function การทำงานของระบบที่ทำหน้าที่ให้ผลลัพธ์หรืองานตามที่ผู้ใช้ต้องการ นักพัฒนาระบบจะนำ Use Case มาสร้างเป็น Model ที่สามารถอธิบาย Function การทำงานที่สมบูรณ์ของระบบต่อไป

Iterative and Incremental Development

ในโลกแห่งความเป็นจริง การทำงานไม่สามารถทำให้เสร็จสมบูรณ์ได้ในขั้นตอนเดียว และวิธีการทำงานโดย Water fall model ซึ่งเป็นวิธีการทำงานแบบดั้งเดิมจะต้องทำงานให้เสร็จในรอบเดียว นับว่าเป็นความเสี่ยงอย่างยิ่งต่อความล้มเหลวของโครงการ เนื่องจากการทำงานในยุคปัจจุบันมักมีการเปลี่ยนแปลง Environment ต่างๆ เกิดขึ้นเสมอ เช่น Requirement change Technology change เป็นต้น

ดังนั้นจะเห็นได้ว่าในสภาพการทำงานจริงๆ มักมีการทำงานแบบแก้ไขใหม่วนซ้ำเรื่อยๆ ซึ่งก็คือ Iteration นั้นเอง นอกจากนี้ยังพบว่าการทำงานโดยการค่อยๆ เพิ่มงานเข้าไปในงานเดิมที่ทำเสร็จแล้วเรื่อยๆ จนหมดทั้งโครงการก็จะเป็นการลดความเสี่ยงงานได้ด้วย เนื่องจากงานที่ทำเสร็จเป็นช่วงๆ โดยทำการแบ่งซอยงานใหญ่ๆ ออกเป็นงานย่อยๆ เมื่อเกิดความผิดพลาดขึ้นจะมีผลกระทบต่องานเพียงส่วนย่อยที่กำลังดำเนินการอยู่เท่านั้น ไม่ใช่กระทบทั้งโครงการ วิธีการทำงานเช่นนี้เรียกการทำงานแบบ Incremental จะเห็นว่าการทำงานแบบ Incremental & Iteration จะให้ผลดีกว่าการทำงานแบบดั้งเดิมหรือ Water fall model ดังนั้นในยุคปัจจุบันจึงนิยมที่จะทำงานโดยอาศัยหลัก Incremental & Iteration

การทำงานในปัจจุบันมักพบว่าโปรเจคต่างๆ จะมีขนาดใหญ่ ดังนั้นจึงนิยมทำการแบ่งงานออกเป็นโปรเจคย่อยๆ หลายๆ ส่วน แต่ละส่วนก็จะมีการทำการวนซ้ำ และเพิ่มเติมเข้าไปในรายละเอียดของงานเดิม การแบ่งส่วนงานออกเป็นงานย่อยๆ จะส่งผลดีคือ

1. การแบ่งส่วนงานจะสามารถทำงานร่วมกับ Use Case จำนวนมากได้

2. ลดความเสี่ยงในการทำงาน

Symbol in Analysis Model

1) Boundary Class (ส่วนติดต่อผู้ใช้)

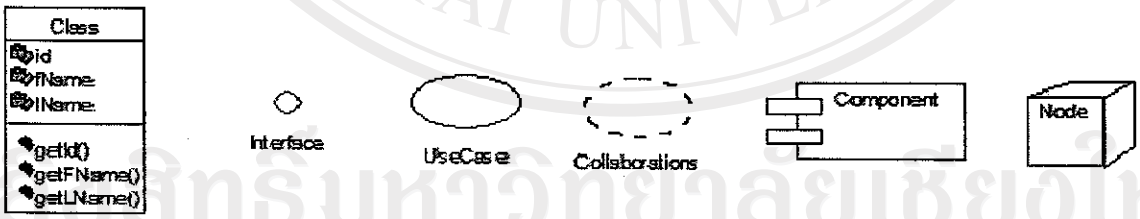
2) Control Class (ส่วนควบคุม)

3) Entity Class (ใช้เก็บข้อมูล)

Building Block of the UML : องค์ประกอบของ UML แบ่งออกเป็น 3 กลุ่ม

1) Thing คือสัญลักษณ์หรือสิ่งต่างๆ ที่นำมาใช้สร้าง Diagram UML แบ่งออกเป็น 4 หมวด คือ

1. Structural Things หรือหมวดโครงสร้าง เป็นคำนามใช้สำหรับ UML ส่วนใหญ่จะเป็นส่วน Static ได้แก่ Use case Interface Class Collaboration Component Note



รูปที่ 3.2 Use case Interface Class Collaboration Component Note

2. Behavioral Things หรือหมวดพฤติกรรม ได้แก่ส่วนที่เป็น Dynamic แสดงถึงพฤติกรรมของระบบ ประกอบด้วย 2 ส่วนใหญ่ๆ คือ

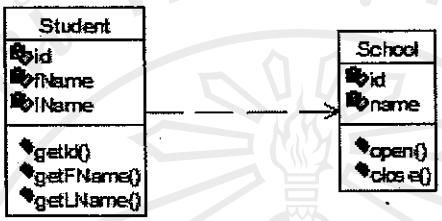
- Interaction
- State Machine

3. Grouping Things หรือหมวดการจัดกลุ่มหมู่ได้แก่ Package

4. Annotation Things หรือหมวดคำอธิบายได้แก่ Note

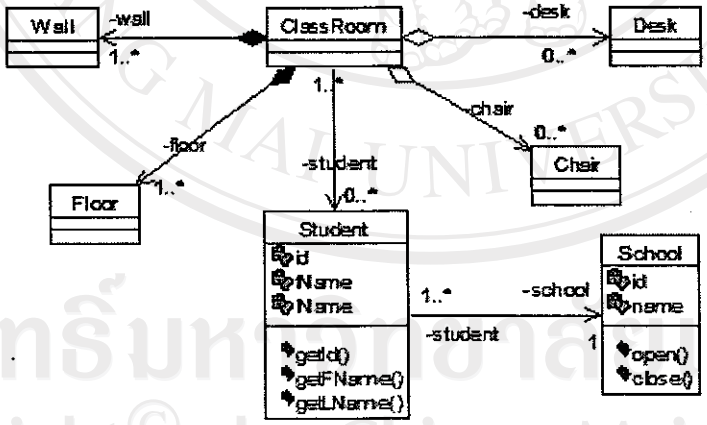
2) Relationships ความสัมพันธ์ใน UML ประกอบด้วย

1. Dependency หรือความขึ้นอยู่ต่อกันจะให้ความหมายว่าเมื่อเกิดการเปลี่ยนแปลงที่ส่วนหนึ่งแล้วจะส่งผลกระทบถึงอีกส่วนหนึ่งที่ลากเส้นมาสัมพันธ์กัน เช่น การเปลี่ยนแปลงของโรงเรียนจะมีผลกระทบของนักเรียนเป็นต้น



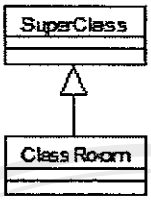
รูปที่ 3.3 Relationships Dependency

2. Association หรือความสัมพันธ์จะแสดงถึงความสัมพันธ์ระหว่าง Object ความสัมพันธ์ยังสามารถแยกออกเป็นความสัมพันธ์แบบธรรมดา ความสัมพันธ์แบบ Aggregration Composite ทิศทางความสัมพันธ์ นอกจากนั้นยังสามารถกำหนด Multiplicity ให้กับความสัมพันธ์ได้อีกด้วย



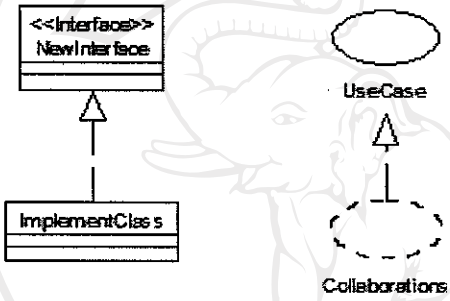
รูปที่ 3.4 Relationships Association

3. Generalization หรือ การสืบทอดคุณสมบัติ หรือ Inheritance นั่นเอง



รูปที่ 3.5 Relationships Generalization

4. Realization หรือ การทำให้ทำงานได้จริง เช่น Interface ถูก Realize โดย Class Use case ถูก Realize โดย Collaboration



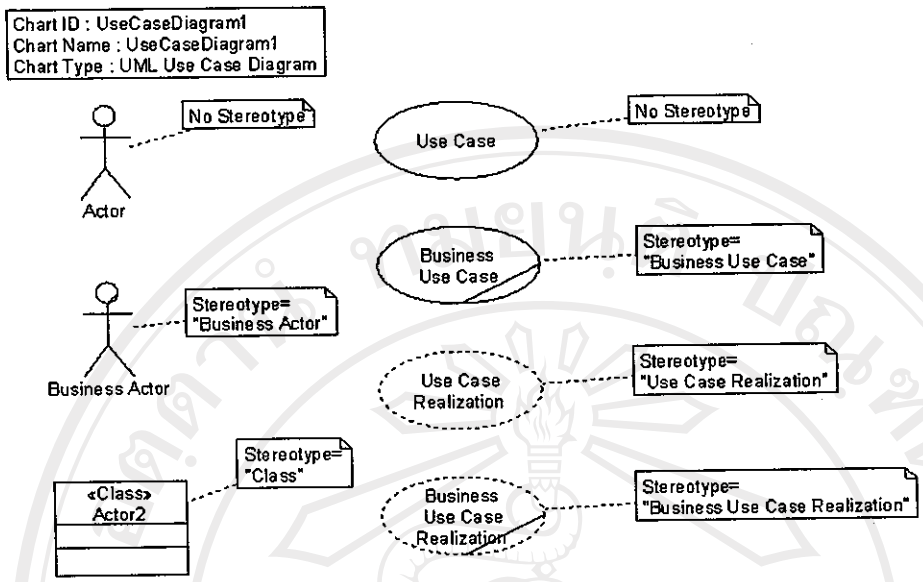
รูปที่ 3.6 Relationships Realization

3) Diagrams : Diagrams ของ UML แบ่งออกได้เป็น 9 Diagrams หลักดังต่อไปนี้คือ

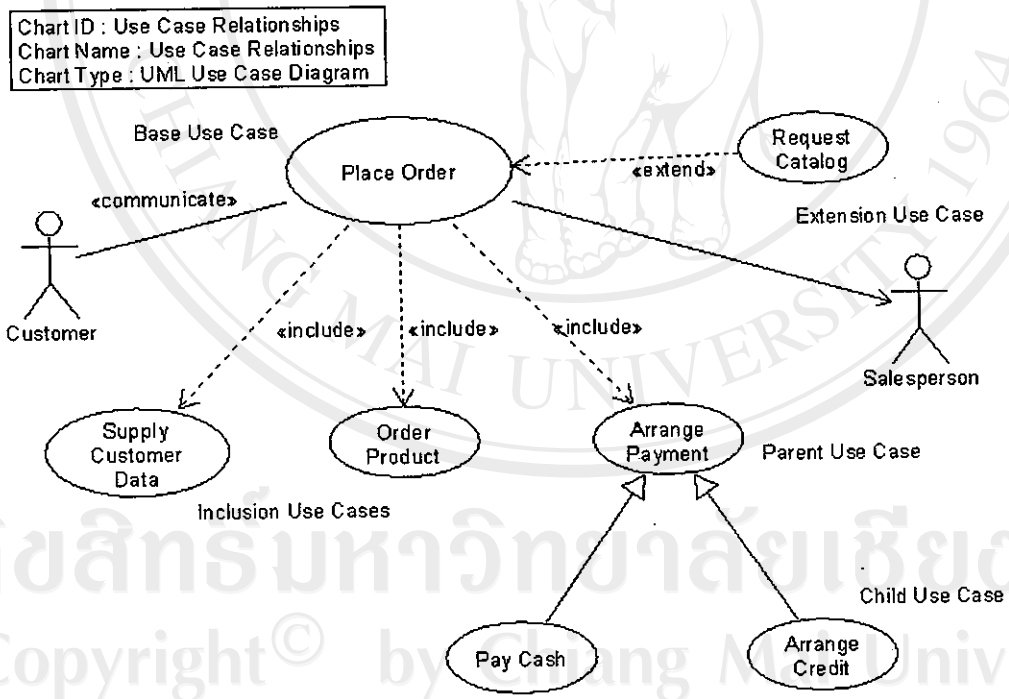
1. Use case Diagram ในการพัฒนาระบบงานใดๆ นั้น การเก็บรวบรวมความต้องการขอ

ผู้ใช้มีความสำคัญมาก และจะทำในระยะแรกๆ ของการพัฒนาระบบงานเสมอ Use case diagram เป็น Diagram ที่ทำหน้าที่ Capture requirement

- เป็นเทคนิคในการสร้างแบบจำลองเพื่อใช้อธิบายหน้าที่ของระบบใหม่หรือระบบปัจจุบัน
- กระบวนการสร้าง Use case เป็นแบบ Iteration
- ความต้องการของระบบจะได้จาก ลูกค้า/ผู้ใช้ + ผู้พัฒนาระบบ
- องค์ประกอบจะมี Use case Actor Use case Relation และ System

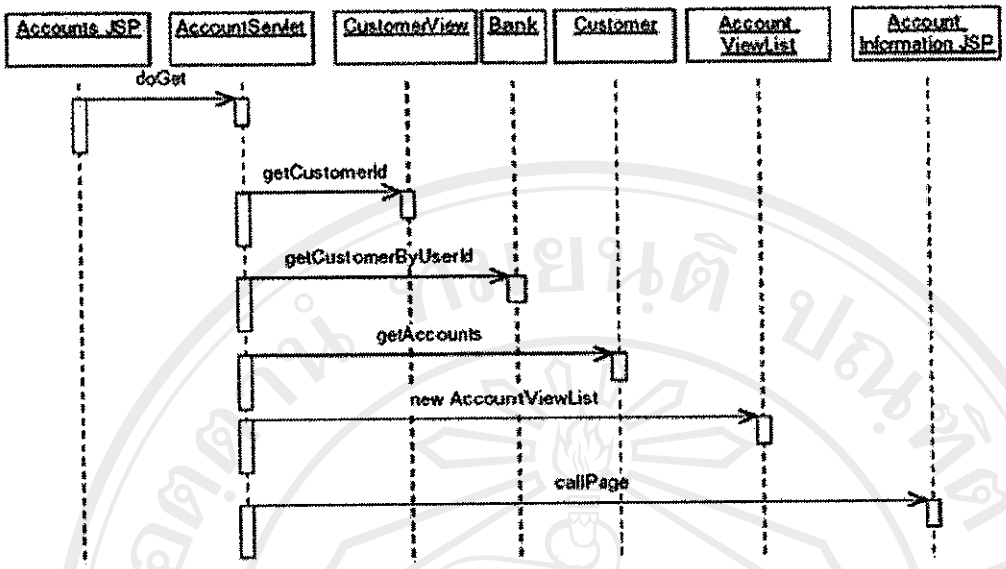


รูปที่ 3.7 องค์ประกอบของ Use Case



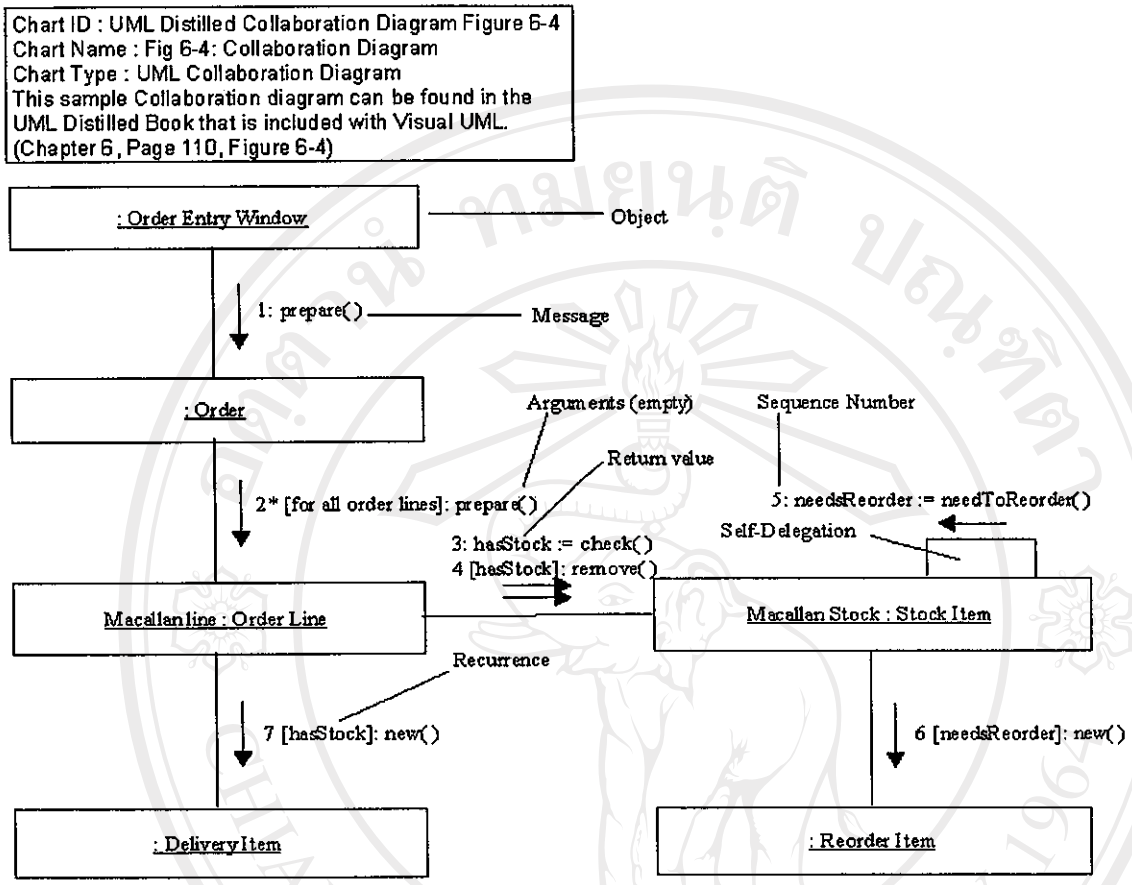
รูปที่ 3.8 Use Case และ ความสัมพันธ์

2. Sequence Diagram จ ะบอกลำดับการทำงานของระบบ โดยมี Object และ เวลา เป็นตัวกำหนดลำดับของงาน



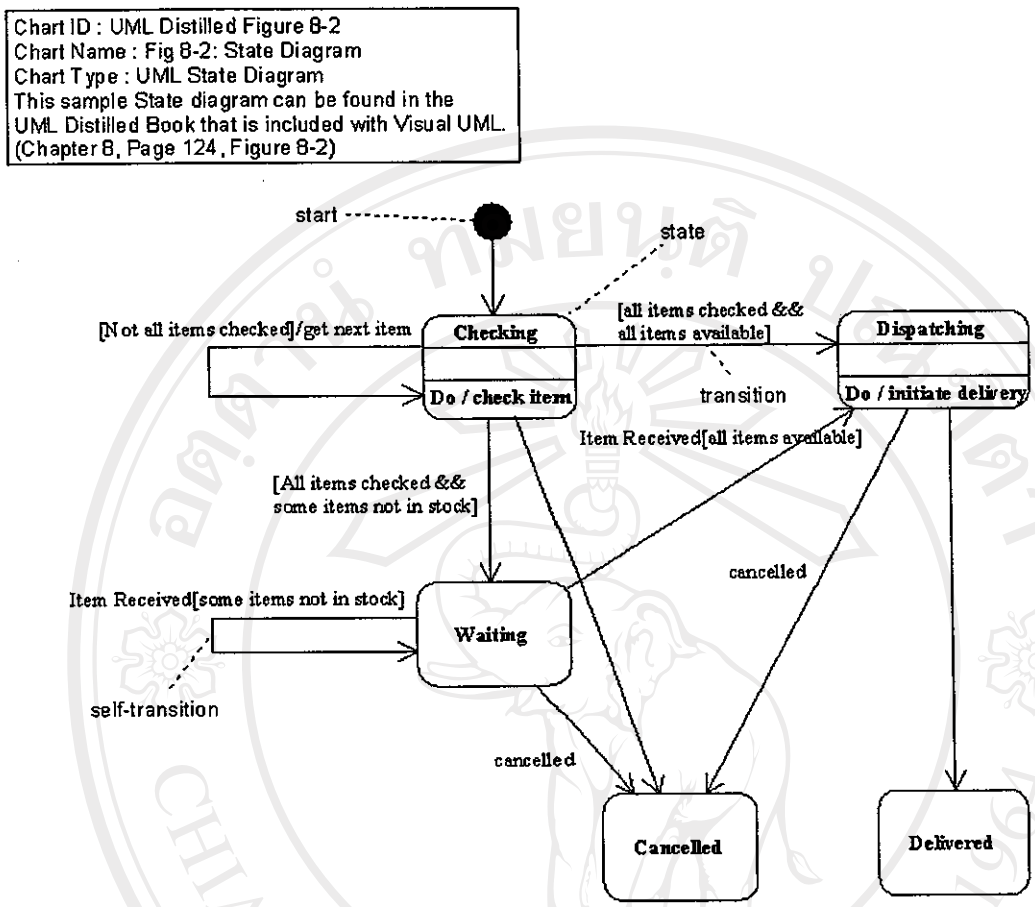
รูปที่ 3.9 Sequence Diagram ในขั้นตอนของการ Design

3. Collaboration Diagram ทำหน้าที่เช่นเดียวกับ Sequence diagram แต่รูปแบบและลักษณะการเขียนจะต่างกัน หรือ อาจกล่าวได้ว่า Collaboration diagram ก็คือรูปอีกรูปแบบหนึ่งของ Sequence diagram เมื่อได้ Sequence diagram แล้ว Tools บางชนิดสามารถ Generate Collboration diagram ให้ได้เลย หรือ ในทางกลับกันเมื่อสร้าง Collaboration diagram เสร็จแล้ว ก็จะสามารถ Generate Sequence diagram ได้ โดยอัตโนมัติ ซึ่งถือว่าทั้ง 2 Diagram สะท้อนภาพกันและกันอยู่นั่นเอง



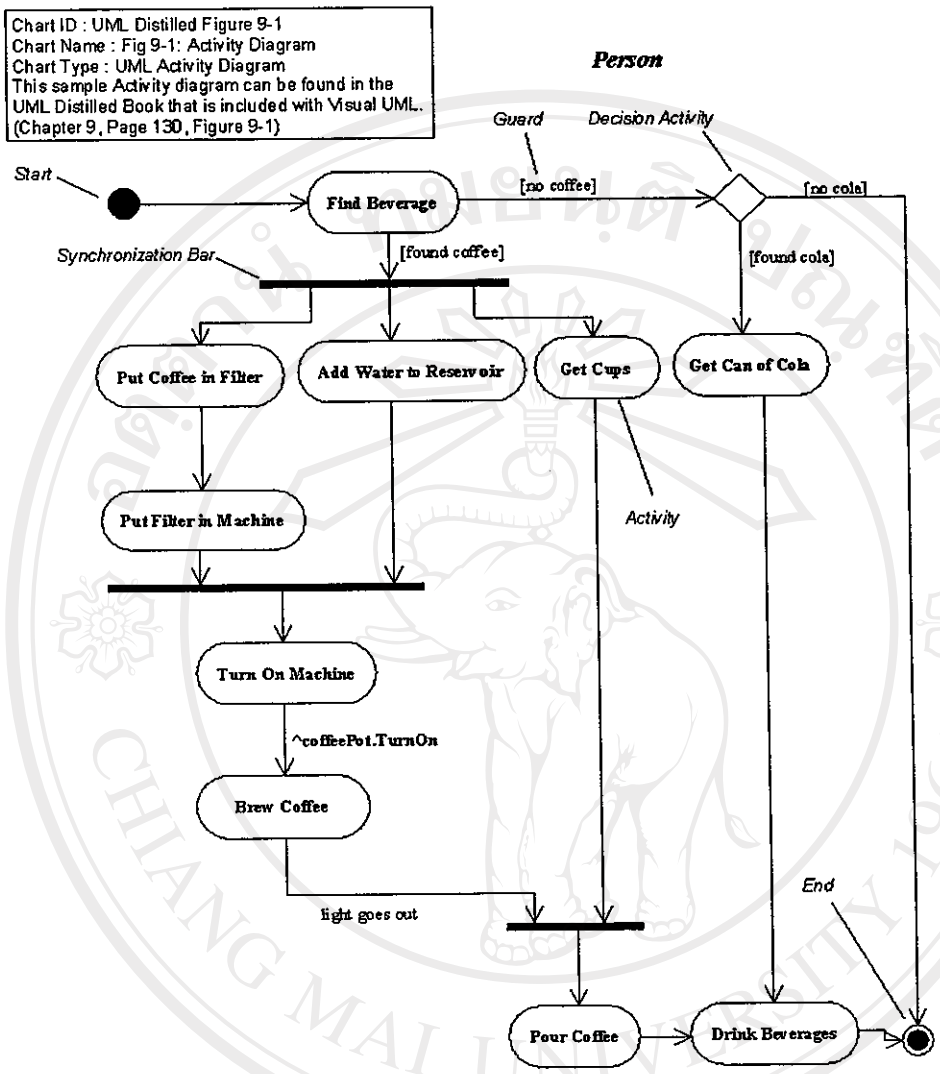
รูปที่ 3.10 ตัวอย่าง Collaboration diagram

4. State Diagram ประกอบด้วย State ต่างๆ ของ Object และเหตุการณ์ต่างๆ ที่ทำให้สถานะของ Object เปลี่ยนและการกระทำที่เกิดขึ้นเมื่อสถานะของระบบเปลี่ยนไป สามารถบอกสถานะของ Object ได้ โดยจะให้ความสนใจว่า ณ เวลาใดๆ Object นั้นมี Status เป็นแบบใด



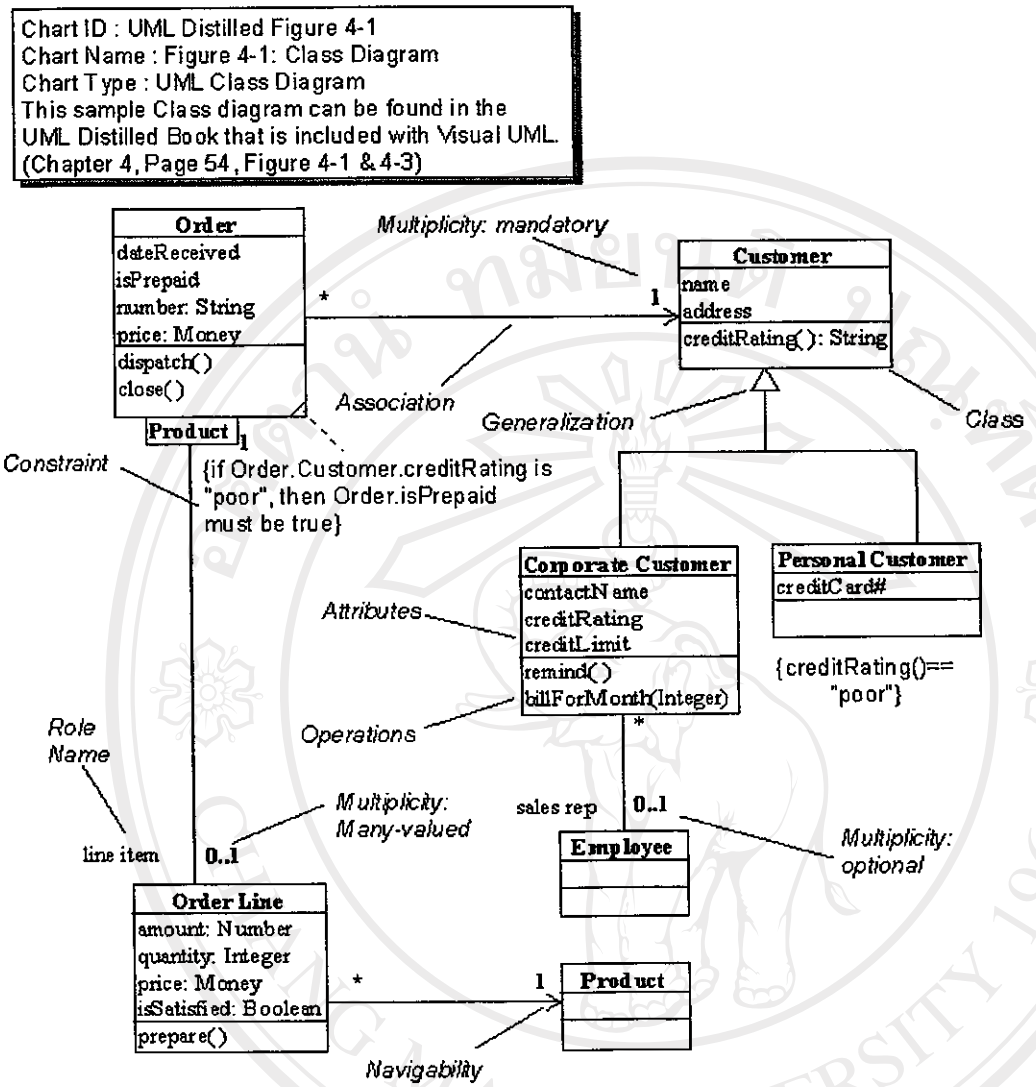
รูปที่ 3.11 ตัวอย่าง State diagram

5. Activities diagram แสดงลำดับกิจกรรมของการทำงาน (Flow) สามารถแสดงทางเลือกที่เกิดขึ้นได้ Activity diagram จะแสดงขั้นตอนการทำงานในการปฏิบัติการ โดยประกอบไปด้วยสถานะต่างๆ ที่เกิดขึ้นระหว่างการทำงาน และผลจากการทำงานในขั้นตอนต่างๆ



รูปที่ 3.12 ตัวอย่าง Activity diagram

6. Class Diagram ประกอยด้วย Class และความสัมพันธ์ต่างๆ ระหว่าง Class เช่น Dependency Generalization Association เป็นต้น Class diagram ยังสามารถทำการแสดงรายละเอียดภายใน Class แต่ละ Class ได้ว่ามี Method อะไรบ้าง Field และ Attribute เป็นอย่างไร



รูปที่ 3.13 ตัวอย่าง Class diagram

7. Object Diagram ประกอบด้วย Object และ Relation ระหว่าง Object โดยแต่ละ Object จะแสดง Instance ของแต่ละ Class ที่มีในระบบ และความสัมพันธ์ต่างๆ ระหว่าง Class เช่น Dependency Generalization Association จะมีลักษณะเช่นเดียวกับใน Class diagram

8. Component diagram เป็น Diagram ซึ่งแสดงโครงสร้างทางกายภาพของ Software โดยจะประกอบด้วยองค์ประกอบซึ่งอยู่ในรูปต่างๆ เช่น Binary Text และ Executable ภายใน Component Diagram ก็จะมีความสัมพันธ์แสดงอยู่เช่นเดียวกับ Class diagram Object diagram

9. Deployment diagram เป็นสิ่งที่สามารถทำการแสดงระบบสถาปัตยกรรมของ Hardware/Software ตลอดจนความสัมพันธ์ระหว่าง Hardware/Software