การออกแบบและพัฒนาระบบสารสนเทศเชิงวัตถุโดยใช้ OONIAM/UML; กรณีศึกษาระบบทะเบียนนักศึกษา

The Design and Implementation of an Object-Oriented system using OONIAM/UML; The case study of student Registration system

นาย ณรงเดช เจิมเมือง
นาย ณัฐภัทร อิทธิรัตนสุนทร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

การออกแบบและพัฒนาระบบสารสนเทศเชิงวัตถุ โดยใช้ OONIAM/UML ; กรณีศึกษาระบบทะเบียนนักศึกษา

The Design and Implementation of an Object-Oriented system using OONIAM/UML; The case study of student Registration system

โดย

นาย ณรงเดช เจิมเมือง

นาย ณัฐภัทร อิทธิรัตนสุนทร

อาจารย์ที่ปรึกษา

รศ.ดร. ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบและพัฒนาระบบสารสนเทศเชิงวัตถุโดยใช้ OONIAM/UML;
กรณีศึกษาระบบทะเบียนนักศึกษา

The Design and Implementation of an Object-Oriented system using OONIAM/UML;
The case study of student Registration system

ผู้จัดทำ

1.นาย ณรงเดช เจิมเมือง รหัสประจำตัว 42015301

2.นาย ณัฐภัทร อิทธิรัตนสุนทร รหัสประจำตัว 42015302

อาจารย์ที่ปรึกษา

(รศ.ดร. ศุภมิตร จิตตะยโศธร)

การออกแบบและพัฒนาระบบสารสนเทศเชิงวัตถุโดยใช้ OONIAM/UML;กรณีศึกษาระบบทะเบียนนักศึกษา

นาย ณรงเดช เจิมเมือง42015301
นาย ณัฐภัทร อิทธิรัตนสุนทร42015302
รศ.ดร. ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ในการวิเคราะห์และออกแบบระบบเชิงวัตถุ โดยใช้ภาษาที่เรียกว่า "UML" (Unified Modeling - Language) ในการวิเคราะห์และออกแบบและจะทำการพัฒนาระบบที่ได้รับการออกแบบ มาโดยที่ต้องการพัฒนา ระบบเชิงวัตถุนั้นเป็นระบบที่มีความเป็นกลางซึ่งระบบนั้นจะสามารถที่จะปรับเปลี่ยนตามความต้องการของ ผู้ใช้ได้อย่างไรก็ตามการวิเคราะห์และออกแบบระบบเชิงวัตถุ โดยใช้ยูเอ็มแอลนั้น มีข้อจำกัดบางกรณีในการทำงาน

วิทยานิพนธ์ฉบับนี้นำเสนอ โมเดลOONIAMในการที่จะช่วยวิเคราะห์และออกแบบระบบในบางส่วนที่ยูเอ็มแอลมีข้อจำกัดและเสนอข้อบกพร่องและแนวในการปรับปรุงยูเอ็มแอลเพื่อให้สามารถใช้ยูเอ็มแอลในการวิเคราะห์และออกแบบระบบให้สามารถรองรับการทำงานให้มากยิ่งขึ้น โดยจะนำกรณีศึกษาของระบบลงทะเบียนนักศึกษามาเป็นตัวอย่างในการนำเสนอละจะใช้ระบบจัดการฐานข้อมูลคาเช่และไมโครซอร์ทวิเชล - เบสิค 6.0 เพื่อร่วมใช้ในการพัฒนาระบบที่ได้รับการออกแบบมา

The Design and Implementation of an Object-Oriented system using OONIAM/UML ;
The case study of student Registration system.

Narongdech Jermmung
Nattapat Attiratanasunthron
Assoc. Prof. Dr. Suphamit Chittayasothorn
Advisor

Abstract

Analysis and Design object-oriented system using a language called "UML" (Unified Modeling Language) This project is to analyze, design and develop the system such that the developed object-oriented system must be in general compatibility. The system is able to adapt to meet User's needs. However, there are limitations of system performance in some cases during its operation.

This project presents the OONIAM model in order to solve the UML's limitations in analysis and design of the systems. Also described about obstacles and solutions to improve the UML so as to support more tasks in designing and analyzing the system. The case student of the student registration system is brought to study as an example for presenting such the improvement. The Cache' database managing system and Microsoft Visual Basic 6.0 are also used in developing the designed system

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลายฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษาวิทยานิพนธ์ รวมทั้ง อาจารย์ บัณทิต พัสยา ที่ให้ความเอาใจใส่ ไตร่ถามความก้าวหน้า แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

นาย ณรงเดช เจิมเมือง

นาย ณัฐภัทร อิทธิรัตนสุนทร

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมา

องค์กรในสมัยปัจจุบันนี้ส่วนใหญ่ล้วนมีแข่งขันสูง การที่จะได้เปรียบเหนือคู่แข่งนั้นต้องมีข้อมูลที่เหนือกว่า ทำอย่างไรให้แต่ละองค์กรมีระบบสารสนเทศที่มีความสามารถที่จะรองรับและจัดเก็บข้อมูลต่างๆ เพื่อการแข่งขันในระหว่างองค์กรได้

อย่างไรก็ตามการที่แต่ละองค์กรจะทำการพัฒนาระบบสารสนเทศของตนเองขึ้นมานั้นมิใช่เรื่องง่ายที่ทุกองค์กรจะสามารถจะทำหรือพัฒนาได้ เพราะอาจติดข้อจำกัดหลายหลากประการ เช่น ความรู้ความสามารถของบุคคลในองค์กรรวมทั้งเงินทุนและเวลาในการพัฒนาระบบขององค์กรของตนเองขึ้นมา และแม้ว่าจะมีความรู้ความสามารถของบุคคลในองค์กรและเงินทุนวิจัยพัฒนา แต่ถ้าหากมีระบบสารสนเทศที่มีประสิทธิภาพสูงที่ได้รับการพัฒนาอยู่ก่อนแล้ว ก็จะเป็นการเสียเวลาและทรัพยากรโดยใช่เหตุ หรืออาจเป็นเพราะการที่องค์กรแต่ละองค์กรมีวัตถุประสงค์ที่แตกต่างกัน มีส่วนให้การออกแบบระบบนั้นต้องแตกต่างกัน แต่หากจะมีพื้นฐานหรือองค์ประกอบหลักที่ใช้งานร่วมกันได้

โครงงานนี้ได้นำเสนอกรณีศึกษาโดยใช้แนวความคิดของการที่จะทำการพัฒนาและปรับปรุงระบบสารสนเทศให้มีคุณสมบัติเป็นซอฟท์แวร์กลางได้ โดยหากองค์กรใดที่มีจุดประสงค์ขององค์กรที่คล้ายคลึงกัน หรือหากตรงกันแล้ว ก็จะสามารถนำระบบสารสนเทศนี้ไปใช้งานได้หรือหากที่ข้อแตกต่างของความต้องการในบางส่วนก็จะสามารถเพิ่มและลดองค์ประกอบของระบบสารสนเทศในองค์กรนั้นได้

1.2 วัตถุประสงค์ของงานวิจัย

1. ศึกษาหลักการของแบบจำลองเชิงวัตถุ (Object Oriented Model) ในแนวคิดต่างๆ
2. ศึกษาวิเคราะห์แก้ไขข้อจำกัดในบางส่วนและทำการพัฒนากระบวนการการออกแบบระบบโดยใช้ UML ซึ่งเป็นภาษาที่สนับสนุนแบบจำลองเชิงวัตถุ
3. ศึกษาและวิเคราะห์กระบวนการการออกแบบและออกแบบระบบ โดยใช้ OONIAM
4. ทำการสร้างระบบงานบางส่วนที่ออกแบบโดย OONIAM/UML ที่ออกแบบไว้ เพื่อทดสอบว่าสามารถจะใช้กระบวนการพัฒนาระบบนี้ นำไปพัฒนาระบบในแนวความคิดของซอฟท์แวร์กลางได้

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้นำเสนอการปรับปรุงข้อจำกัดของโมเดลยูเอ็มแอล ทำความเข้าใจปัญหาข้อจำกัดและทำการปรับปรุงแก้ไขข้อจำกัดในการทำงานออกแบบ รวมทั้งยกตัวอย่างกรณีศึกษาของระบบลงทะเบียนนักศึกษาโดยใช้แนวคิดของซอฟท์แวร์กลางได้

1.4 วิธีการดำเนินงาน

1.4.1 ศึกษาระบบฐานข้อมูลในรูปแบบต่างๆ (Literature Review)

1.4.2 ศึกษาการออกแบบระบบโดยใช้ เครื่องมือ UML

1.4.3 ศึกษาระบบฐานข้อมูลเชิงวัตถุ

1.4.4 ศึกษาการออกแบบฐานข้อมูล โดยใช้ OONAIM เพื่อทำการออกแบบระบบได้

1.4.5 ทำการหาข้อจำกัดในการทำงานของ UML และปรับปรุงแก้ไขในบางส่วน

1.4.6 ทำการออกแบบระบบระบบกรณีศึกษาระบบลงทะเบียนนักศึกษาในเชิงซอฟท์แวร์กลางได้

1.4.7 พัฒนาระบบกรณีศึกษาระบบลงทะเบียนนักศึกษาในเชิงซอฟท์แวร์กลางได้

ทำการศึกษาหลักการของแบบจำลองเชิงวัตถุ (Object Oriented Model) ในแนวคิดต่างๆ

1.5 เนื้อหาภายในปริญญานิพนธ์

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้นั้นนำเสนอเป็นส่วนต่างๆ ดังนี้ โดยในบทนี้นั้นจะกล่าวถึงจะทำการออกแบบวิเคราะห์และพัฒนาระบบได้อย่างไรให้สามารถนำไปใช้ได้ในขณะที่สิ่งแวดล้อมและความต้องการของระบบรวมถึงการที่ระบบมีความต้องการ(Requirement) ที่แตกต่างกันไปในบางส่วน และนำเสนอ วิธีการออกแบบเชิงวัตถุในแนวความคิดต่างๆ เพื่อทำการเปรียบเทียบในแต่ละวิธีนั้นว่ามีหลักการออกแบบและวิเคราะห์อย่างไรในการทำงานการวิเคราะห์และออกแบบเชิงวัตถุ (Literature Review) ในบทที่ 2 นั้นจะนำเสนอเครื่องมือในการออกแบบเชิงวัตถุโดยใช้ เครื่องมือยูเอ็มแอลในการทำงาน โดยนำเสนอหลักการและวิธีการออกแบบที่เป็นหลักการของยูเอ็มแอล ในบทถัดมาบทที่ 3 จะนำเสนอเครื่องมือช่วยในการออกแบบเชิงวัตถุอีกตัวหนึ่งคือ OONIAM เป็นเครื่องมือในการที่จะเข้ามาเพื่อช่วยในการแก้ไขข้อจำกัดบางประการของยูเอ็มแอล รวมทั้งหลักการในการสร้างและกฎต่างๆ ภายใน NIAM และ OONIAM และในบทที่ 4 จะนำเสนอระบบจัดการฐานข้อมูลคาเช่ Cache′ DBMS ซึ่งเป็นระบบจัดการฐานข้อมูลเชิงวัตถุตัวหนึ่ง โดยทำการศึกษาหลักการทำงานและภาษาของคาเช่ เพื่อนำไปพัฒนาและปรับปรุงระบบได้ ในบทที่ 5 จะทำการกล่าวถึงข้อจำกัดบางประการในการที่จะใช้ยูเอ็มแอลในการวิเคราะห์และออกแบบระบบ โดยนำกล่าวถึงปัญหาและวิธีที่จะปรับเปลี่ยนได้ในบางส่วน เพื่อที่จะอาจทำให้การออกแบบและวิเคราะห์ระบบโดยใช้ยูเอ็มแอลนั้นมีความสมบูรณ์และทำความเข้าใจในตัวโมเดลได้มากยิ่งขึ้น ในบทที่ 6 นั้นทำการนำการออกแบบจากยูเอ็มแอลและ OONIAM แล้วนั้นไปพัฒนาในแนวความคิดของซอร์ฟแวร์กลางได้ โดยอาศัยหลักการของ Object Oriented ในการทำงาน โดยสามารถสร้างระบบที่มีความเป็นกลางสามารถปรับเปลี่ยนตามสิ่งแวดล้อมได้ และบทสุดท้ายเป็นการสรุปการทำงานและสรุปผลการทดลองตั้งแต่ต้นมา

บทที่ 2

ศึกษาระบบฐานข้อมูลในรูปแบบต่างๆ (Literature Review)

แนวคิดในเชิง Object ของ Bryon K. Ehlmann, Gregory A. Riccardi:

A Notation for Describing Aggregate Relationships in an Object-Oriented Data Model

2.1 ออบเจ็กต์ซับซ้อน (Composite object)

ออบเจ็กต์ซับซ้อน เป็นกลุ่มของออบเจ็กต์ที่มีความสัมพันธ์กันและถูกมองเสมือนว่าเป็น ออบเจ็กต์เพียงตัวเดียว กลุ่มของ ออบเจ็กต์ ที่มีความสัมพันธ์กันนั้นคือ ออบเจ็กต์ ตัวหนึ่งที่มีสมาชิกเป็น ออบเจ็กต์อีกตัวหนึ่ง(IS-PART-OF relationship)ในงานหลายงานเราต้องการโมเดล (Model) ที่จะนิยามและจัดการกับกลุ่มของ ออบเจ็กต์ เสมือนว่าเป็น ออบเจ็กต์ เพียงตัวเดียวและต้องมีการ อ่านและจัดเก็บข้อมูลที่ ง่าย และมีประสิทธิภาพ

ออบเจ็กต์ซับซ้อนได้ถูกนำเสนอขึ้นมาและนำไปใช้ในการพัฒนาโครงการORION(Kimet al.,1987) ซึ่งจากผลการทดสอบงานหลายงานในโครงการ ORION แสดงให้เห็นว่าแนวคิดเรื่อง ออบเจ็กต์ซับซ้อน นั้นมีประโยชน์มาก แต่ว่าแนวคิดเรื่อง ออบเจ็กต์ซับซ้อน นั้นมีจุดอ่อนอยู่บ้าง คือ

ประการแรก คือ เนื่องจาก ส่วนประกอบของ ออบเจ็กต์ซับซ้อน(Component object) จะมีคลาสแม่(Parent or Root Class)ได้เพียงตัวเดียวตามคุณสมบัติของ Exclusivity(exclusive dependent composite reference) ข้อจำกัดอันนี้ทำให้ ในงานบางงานไม่สามารถที่จะนำ Model นี้ไปใช้ประยุกต์ใช้ได้ ยกตัวอย่างเช่น ระบบการจัดการกับ Hypertextซึ่งข้อความจากส่วนหนึ่งในหนังสือเล่มหนึ่งอาจจะมีในหนังสืออีกเล่มหรือหลายเล่มก็ได้

ปัญหาประการที่สองคือ Model นั้นออกแบบมาเพื่อต้องการให้ ออบเจ็กต์ซับซ้อน นั้นถูกออกแบบแบบบนลงล่าง (Top-Down) ยกตัวอย่างเช่น Component ลูก จะไม่สามารถถูกสร้างขึ้นมาได้ถ้า Component พ่อถูกสร้างขึ้นมาก่อน ข้อจำนี้ทำให้เราไม่สามารถพัฒนาโครงการแบบ ล่างขึ้นบน (Bottom-Up) จุดอ่อนประการสุดท้ายคือ เนื่องจากแนวคิดเรื่อง ออบเจ็กต์ซับซ้อน นั้น ออบเจ็กต์ มีความสัมพันธ์เป็นเจ้าของกัน (Dependent objects) และเมื่อ ออบเจ็กต์ ตัวใดตัวหนึ่งถูกลบ Component object ของออบเจ็กต์ ตัวที่ถูกลบนั้นจะต้องถูกลบไปด้วย

ดังนั้น Model แบบที่สองจึงได้ถูกพัฒนาขึ้นเพื่อกำจัดข้อบกพร่องที่เกิดขึ้นที่เกิดจากการพัฒนาโครงการ ORION (Kim et al., 1989a) Model นี้ได้กำหนดความสัมพันธ์ขึ้นมาสองชนิดคือ weak และ composite ความสัมพันธ์แบบ weak นั้น เป็นความสัมพันธ์แบบปกติระหว่าง ออบเจ็กต์ คือ เมื่อ ออบเจ็กต์พ่อ อ้างถึง ออบเจ็กต์ ลูก เพื่อเป็นการใช้ค่าใน Attribute นั้นเฉยๆ และความสัมพันธ์แบบ Composite จะเป็นความสัมพันธ์แบบเป็นส่วนหนึ่งของ (IS-PART-OF Relationship) ความสัมพันธ์แบบ Composite แบ่งออกเป็น exclusive และ shared เนื่องจากงานในยุคแรกนั้นความซับซ้อน และชนิดของข้อมูลก็ไม่ยุ่งยาก ทำให้ ออบเจ็กต์ แต่ละตัวเป็น ออบเจ็กต์ ลูกของแม่ตัวเดียว แต่ในขณะ นี้ข้อมูลมีความซับซ้อนมากขึ้น ออบเจ็กต์ หนึ่งตัวอาจ ออบเจ็กต์ แม่ที่มากกว่าหนึ่งตัว และนอกจากนี้ความสัมพันธ์แบบ Composite ยังแยกออกเป็น dependent และ independect ในยุคแรกนั้น ออบเจ็กต์ มักจะมีความสัมพันธ์แบบ

dependent กับ ออบเจ็กต์ แม่ของมัน แต่ในขณะนี้ ก็จะมีความสัมพันธ์แบบ independent ด้วยดังนั้นจึงมีความสัมพันธ์แบบ composite แบ่งเป็น 4 แบบดังนี้

(1) exclusive dependent composite reference

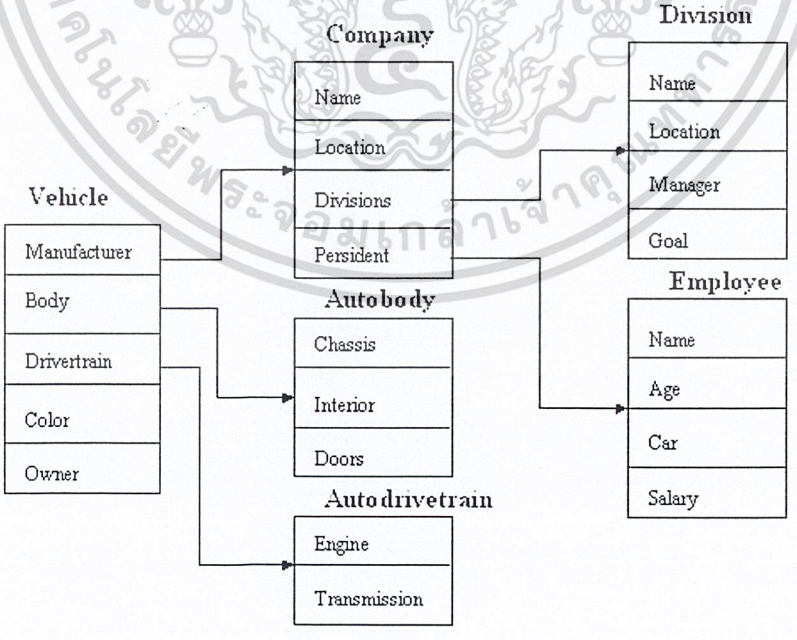
(2) exclusive independent composite reference

(3) shared dependent composite reference

(4) shared independent composite reference

ในขณะที่ความสัมพันธ์แบบที่ (1) เราได้รู้จักกันบ้างแล้วดังข้างต้น (Kim et al., 1989a) และความสัมพันธ์แบบที่ (3) ออบเจ็กต์ แต่ละตัวมี ออบเจ็กต์ แม่ได้มากกว่า หนึ่งตัว เมื่อเป็นเช่นนี้แล้วหมายความว่าการที่ลบ ออบเจ็กต์ซับซ้อน เป็นผลให้ส่วนที่ shared (shared component object) จะถูกลบก็ต่อเมื่อออบเจ็กต์ พ่อของ shared componet object ทุกตัวถูกลบออกหมด ใน Kim et al. (1989a) กฎการลบจะต้องถูกกำหนดขึ้นว่าการลบและเงื่อนไขการลบเป็นอย่างไร เมื่อ component object ของ ออบเจ็กต์ซับซ้อนถูกสร้างขึ้น

เพื่อเป็นตัวอย่างลองพิจารณา คลาส เอกสาร electronic สมมติ ว่าเอกสารประกอบด้วยชื่อเรื่อง(title) ผู้แต่งหนึ่งคนหรือมากกว่า มีหนึ่งส่วน(section)หรือมากกว่า แต่ละส่วนประกอบด้วยหลายย่อหน้า ส่วนและย่อหน้าสามารถถูก shared ให้เอกสารต่างๆ ได้ และให้สามารถเพิ่ม คำอธิบายประกอบเข้าไปในเอกสารได้ คำอธิบายประกอบจะเป็นข้อมูลส่วนตัวของเอกสารฉบับนั้นฉบับเดียว สุดท้ายเอกสารสามารถที่จะมีรูปประกอบซึ่งนำมาจาก แฟ้มข้อมูลที่มีอยู่แล้ว ดังนั้นเราสามารถแยกความสัมพันธ์ออกได้เป็น sections และย่อหน้า มีความสัมพันธ์แบบ shared dependent components คำอธิบายประกอบ มีความสัมพันธ์แบบ exusive dependent componenst รูป มีความสัมพันธ์แบบ shared independent components



รูปที่ 2-1 การสืบทอดคุณสมบัติของ คลาส

คุณสมบัติของ ออบเจ็กต์ซับซ้อน

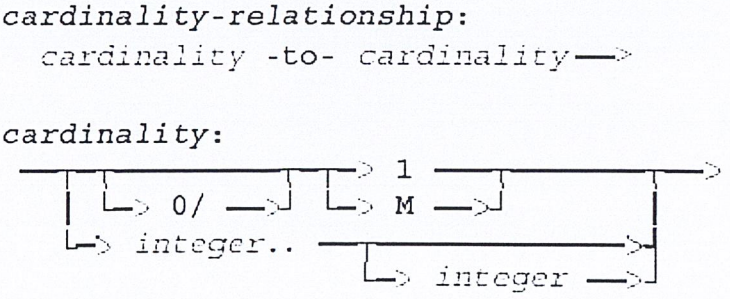
1. คุณสมบัติของ Composite Link ภายใน ออบเจ็กต์ซับซ้อน จะถูกถ่ายทอดไปยัง คลาส ที่สืบทอดคุณสมบัติด้วย(inherite class) ตัวอย่างเช่น class automobile เป็น subclass ของ vehicle มันจะสืบทอดคุณสมบัติ Body จาก vehicle เนื่องจาก Body เป็น composite component ใน คลาส vehicle ดังนั้นใน คลาส automobile จะมี Body เป็น composite component ด้วย
2. Composite component จะเปลี่ยนความสัมพันธ์กับ ออบเจ็กต์ แม่ (composite link)เป็นแบบ noncomposite component ได้แต่ noncomposite component จะเปลี่ยนความสัมพันธ์กับ ออบเจ็กต์ แม่ เป็น composite component ไม่ได้

2.2 A Notation for Describing Aggregate Relationships in an Object-Oriented Data Model โดยมีรายละเอียดดังนี้

ความสำคัญของ ในระบบ Object-Oriented Database OODBs นั้นมีความสัมพันธ์กันในแบบ “is a” และแบบ “is a part of” ซึ่งเกี่ยวเนื่องกับระบบฐานข้อมูล โดยจะนำแนวคิดของ Object-Oriented กับระบบการจัดการฐานข้อมูลเข้าไว้ด้วยกัน โดยในส่วนของแนวคิด Object-Oriented นั้นจะมีส่วนของ รหัสเฉพาะของแต่ละออบเจ็กต์ (Object Identity : OID), Abstract Data types หรือ คลาส , การถ่ายทอดคุณสมบัติ (Inheritance) และโพลีมอร์ฟีซึม (Polymorphism) โดยจะนำเสนอและชี้เฉพาะไปในส่วนชนิดความสัมพันธ์และสนับสนุน OODB System ได้อย่างไร

Cardinality Relationship

เครื่องหมายที่ใช้แทน Cardinality Relationship มีรูปแบบที่เป็นไปได้ดังแสดงตามรูปที่ 1. Cardinality ด้านหน้า –to- ระบุชื่อ Cardinality สำหรับ Subject class และ. Cardinality ด้านหลัง –to- ระบุชื่อ Cardinality สำหรับ Relative class จำนวนตัวเลขที่ระบุตรง Relative class cardinality นั้นหมายถึงจำนวนของ ออบเจ็กต์ ที่เกิดจาก Relative class ต่อจำนวน ออบเจ็กต์ ของ Subject class หนึ่งตัว และในทางกลับกัน จำนวนตัวเลขที่ระบุตรง Subject class cardinality นั้นหมายถึงจำนวนของ ออบเจ็กต์ ที่เกิดจาก Subject class ต่อจำนวน ออบเจ็กต์ ของ Relative class หนึ่งตัว เครื่องหมาย “/” ใช้แทนคำว่า “หรือ” M(many) หมายถึงมากกว่าหรือเท่ากับหนึ่ง



รูปที่ 2-2 แสดงความสัมพันธ์แบบ Cardinality

จะขอยกตัวอย่างเพื่อความเข้าใจ คือ จะเขียน 1-to-0/M แทนความสัมพันธ์ระหว่างแผนกงานและพนักงานที่ทำงานในแผนกนั้น โดยจะมีความหมายว่า แต่ละแผนกจะมีพนักงาน ได้ศูนย์คนหรือมากกว่า และพนักงานทุกคนจะต้องสังกัดแผนกหนึ่งแผนกเสมอ ไม่มีแผนกไม่ได้จะเห็นว่าตัวเลขของสัญญาลักษณ์ "1" นั้นหมายถึงจำนวนของ Subject class object ที่ Relative class object หนึ่งตัวสามารถมีความสัมพันธ์ได้ และตัวเลขของสัญญาลักษณ์ "0/M" นั้นหมายถึงจำนวนของ Relative class object ที่ Subject class object หนึ่งตัวสามารถมีความสัมพันธ์ได้ ยังมีสัญญาลักษณ์ตัวเลขที่แสดงจำนวนของ ออบเจ็กต์ คือ integer..,integer..integer เช่น 2.. หมายถึงสอง ออบเจ็กต์ หรือมากกว่า 2..10 หมายถึงสอง ออบเจ็กต์ ถึงสิบ ออบเจ็กต์ แต่จะมีเงื่อนไขที่ตัวเลขจะต้องเป็นจำนวนเต็มบวกหรือจำนวนเต็มศูนย์ และตัวเลขตัวที่สองต้องไม่น้อยกว่าจำนวนเต็มตัวแรก

ยังมีอีกแง่มุมหนึ่งที่เราควรคำนึงถึง คือ intra-class relationship (ความสัมพันธ์ของ คลาส คลาสเดียวกัน) ซึ่งจะมีความสัมพันธ์แบบ 0/1-to-1, 0/1-to-M หรือ 1-to-M สัญญาลักษณ์ความสัมพันธ์แบบนี้จะไม่สามารถเป็นไปได้ในความสัมพันธ์แบบ intra-class relationship

การกำหนดสัญญาลักษณ์เหล่านี้จะทำให้เกิดเงื่อนไขใน Database ดังต่อไปนี้

-Add a class object

-Delete a class object

-Create a relationship between two class object

-Destroy a ralationship between two class object

-Change a relation for a class object so it relate to a different class object

ทีนี้เราจะแยกดูความหมายของสัญญาลักษณ์แต่ละส่วนคือ

-to-1 กำหนดให้ความสัมพันธ์ R ที่มีกับ relative class คือ หนึ่ง หมายความว่า แต่ละ Subject class object ที่จะประกาศตัวแปรได้นั้นจะต้องมีความสัมพันธ์กับ Relative class object ด้วย คือ Subject object ไม่สามารถถูกรวมเข้าฐานข้อมูลได้หากไม่มีหากไม่มีความสัมพันธ์ R ไปยัง Relative class object และ Subject object จะต้องถูกลบออกถ้า การลบของ Relative object ได้รับการยินยอมให้ทำการลบออกไปได้

ตัวอย่างคือ ความสัมพันธ์ 0/M-to-1 ระหว่าง พนักงานและแผนกงานหมายความว่าพนักงานแต่ละคนจะต้องสังกัดแผนกงานหนึ่งแผนกงานเสมอ การที่จะลบแผนกงาน ของพนักงานจะทำลายความสัมพันธ์ที่ว่าพนักงานแต่ละคนจะต้องสังกัดแผนกงานหนึ่งแผนกงานเสมอ เพราะแผนกงานถูกลบออกไปทำให้พนักงานไม่แผนกงานสังกัด และถ้าแผนกได้รับความยินยอมให้ถูกลบได้ พนักงานที่สังกัดแผนกนั้นจะต้องถูกลบออกด้วยเพื่อทำให้เงื่อนไขของความสัมพันธ์ถูกต้องเสมอ

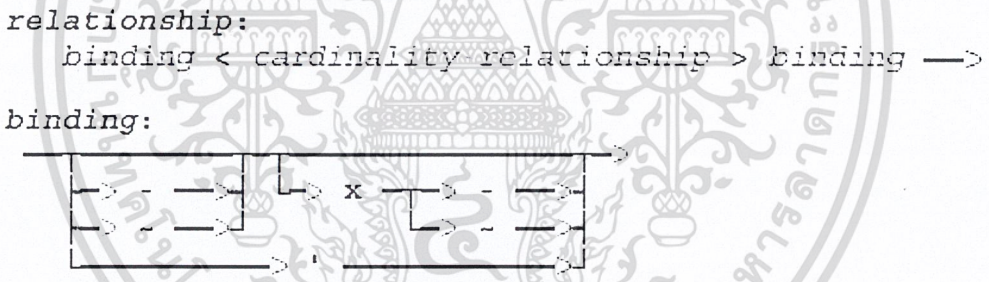
-to-M กำหนดให้ความสัมพันธ์ R ที่มีกับ relative class คือ หนึ่งหรือมากกว่า หมายความว่า แต่ละ Subject class object ที่จะประกาศตัวแปรได้นั้นจะต้องมีความสัมพันธ์กับ Relative class object ด้วย คือ Subject object ไม่สามารถถูกรวมเข้าฐานข้อมูลได้หากไม่มีหากไม่มีความสัมพันธ์ R ไปยัง Relative class object อย่างน้อยหนึ่งตัว และ Subject object จะต้องถูกลบออกถ้า การลบของ Relative object ตัวสุดท้ายได้รับการยินยอมให้ทำการลบออกไปได้

-to-0/1 กำหนดให้ความสัมพันธ์ R ที่มีกับ relative class คือ หนึ่งหรือไม่มีความสัมพันธ์ หมายความว่า แต่ละ Subject class object ที่จะประกาศตัวแปรได้นั้นจะต้องไม่จำเป็นต้องมีความสัมพันธ์กับ Relative class object คือ Relative class object เป็นแค่เพียงตัวที่แสดงคุณสมบัติของ ออบเจ็กต์ เฉยๆ Subject object ถูกลบออกไปได้โดยไม่มีเงื่อนไข

-to-0/M กำหนดให้ความสัมพันธ์ R ที่มีกับ relative class คือ หนึ่งหรือมากกว่า หรือไม่มีความสัมพันธ์ หมายความว่า แต่ละ Subject class object ที่จะประกาศตัวแปรได้นั้นจะต้องไม่จำเป็นต้องมีความสัมพันธ์กับ Relative class object คือ Relative class object เป็นแค่เพียงตัวที่แสดงคุณสมบัติของ ออบเจ็กต์ เฉยๆ Subject object ถูกลบออกไปได้โดยไม่มีเงื่อนไข

Relationship

Relationship ในรูปแบบที่เป็นไปได้ถูกแสดงดังรูปที่ 2-2 ซึ่งจะเป็นตัวกำหนดความสัมพันธ์ระหว่าง ออบเจ็กต์ ที่ให้ลายละเอียดของความความสัมพันธ์ขึ้นมาอีกระดับหนึ่งซึ่งก็เหมือนเรากำหนดกฎเกณฑ์(Constraint)ต่างๆ เข้าไปในข้อมูล หรือเรียกว่า Bindings ซึ่ง Bindings จะเป็นตัวกำหนดคุณสมบัติของการถูกทำลายโดยตรงและโดยแฝง(implicit and explicit) และจะเป็นตัวบอกว่าจะเกิดมีผลการลบที่ทำลายกฎความสัมพันธ์ระหว่าง ออบเจ็กต์ หรือไม่ ความสัมพันธ์ทั้งหมดที่ปรากฏอยู่นั้นจะบอกว่าออบเจ็กต์ ที่เกี่ยวข้องจะถูกทำลายตามกันไปด้วยหรือไม่



รูปที่ 2-3 แสดงการ Binding หรือการกำหนดกฎเกณฑ์เข้าไปในข้อมูล

Default Binding คือ การลบ ออบเจ็กต์ ใดๆ สามารถที่จะลบได้เลย ถ้าหากการลบ ออบเจ็กต์ เหล่านั้นไม่ละเมิดกฎ ของ Cadinality relationship ไม่ว่าจะเป็นการลบโดยทางตรงหรือทางแฝงถ้าเกิดการละเมิดกฎเกิดขึ้นการลบนั้นจะไม่ถูกอนุญาติ ตัวอย่างเช่น ความสัมพันธ์ระหว่าง พนักงาน และแผนกงาน เป็น <0/1-to-M> เมื่อลบพนักงานออกหมดกฎก็ยังไม่ถูกละเมิดเนื่องจาก แผนกงานไม่จำเป็นต้องมีพนักงานก็ได้ แต่เราจะไม่สามารถลบแผนกงานออกโดยตรงไม่ได้เลยหากแผนกงานมีพนักงานสังกัดอยู่ การลบทำให้เกิดการละเมิดกฎทำให้ไม่อนุญาติให้มีการลบได้

Implicit Destructibility Binding มีสัญญาลักษณ์ที่เพิ่มมาสองชนิดคือ “-” และ ”~” การ binding แบบ “-” จะกำหนดว่า ออบเจ็กต์ ที่ binding แบบนี้ไม่สามารถทำการลบออกได้ ตัวอย่างเช่น ความสัมพันธ์ระหว่างพนักงานและแผนกงาน <0/1-to-0/M>- คือพนักงานสามารถที่จะไม่มีแผนกได้ อย่างไรก็ตามถ้าพนักงานสังกัดแผนกแล้วจะไม่สามารถลบพนักงานคนนั้นออกได้ การ binding แบบ “~” จะเกิดเฉพาะความสัมพันธ์แบบ 1 หรือ M เท่านั้น นิยามคือ ใน Relationship R ใดๆจะถูกทำลายโดยแฝงจากการ

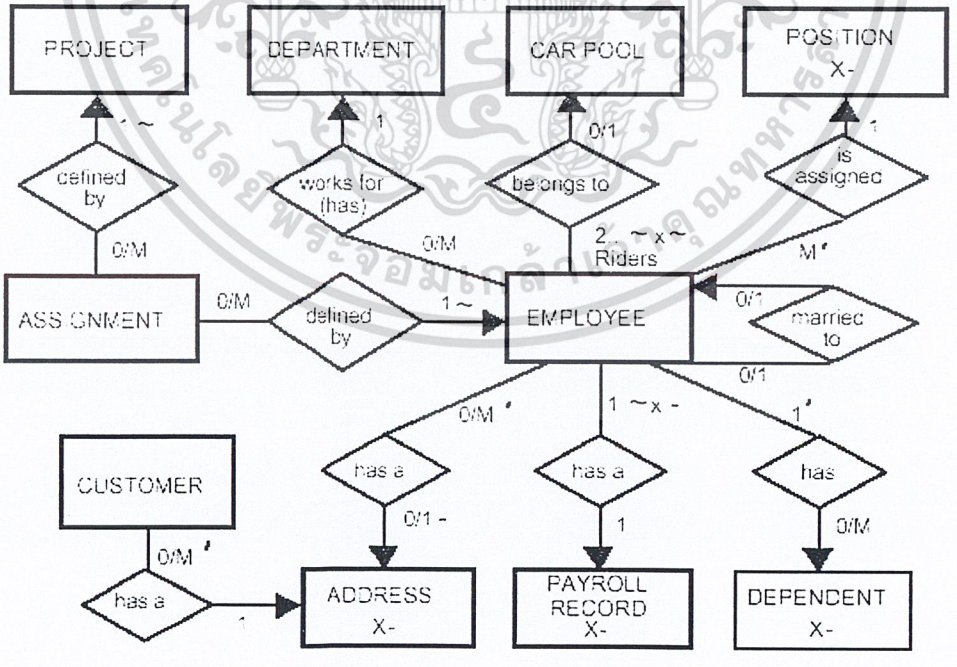
ทำลาย ออบเจ็กต์ C เมื่อการปฏิบัติงานละเมิดกฎ R ทำให้ ออบเจ็กต์ ที่เกี่ยวข้องถูกทำลายไปด้วย จะขอยกตัวอย่างระหว่าง แผนกงานและพนักงาน <1-to-0/M>~ คือเมื่อการลบพนักงานคนสุดท้ายเกิดขึ้นกฎถูกละเมิดจึงต้องลบแผนกออกเพื่อรักษากฎ

Explicit Destructibility Binding มีการ Binding แบบ x- และ x~ ซึ่งการ Binding แบบ x- คือ จะไม่สามารถถูกลบได้โดยตรงแต่จะถูกลบโดยแฝงได้ ตัวอย่างเช่น ความสัมพันธ์ระหว่างพนักงานและแผนกงานเป็น <0/1-to-0/M>x- เมื่อถูกสร้างขึ้นแล้วไม่สามารถลบหรือทำลายโดยตรงได้แต่ระถูกทำลายได้เมื่อพนักงานถูกลบออกหมด การ Binding แบบ x~ คือ จะเกิดเฉพาะความสัมพันธ์แบบ 1 หรือ M เท่านั้น นิยามคือ ใน Relationship R ใดๆ ออบเจ็กต์ จะถูกทำลายโดยตรงได้หากการทำลายละเมิดกฎ ทำให้ออบเจ็กต์ ที่เกี่ยวข้องถูกทำลายไปด้วย จะขอยกตัวอย่างระหว่าง แผนกงานและพนักงาน x~<1-to-0/M>x~ คือเมื่อการลบพนักงานคนสุดท้ายเกิดขึ้นกฎถูกละเมิดจึงต้องลบแผนกออกด้วย

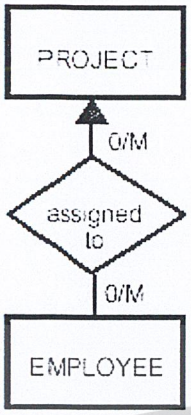
Prime Bindibg มี (‘) เป็นสัญญาลักษณ์ จะมีลักษณะเหมือน ออบเจ็กต์ซับซ้อน คือ ออบเจ็กต์ O’ ถูกลบออกแล้ว ออบเจ็กต์ ที่เกี่ยวข้องจะต้องถูกลบออกด้วย

Object-Relational Diagram(ORD)

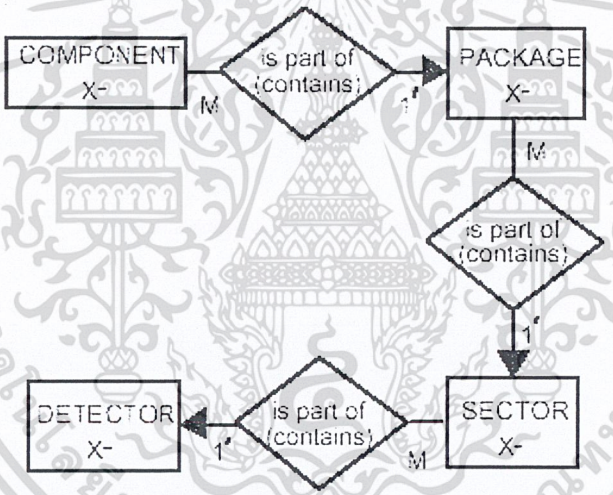
Object -Relational Diagram มีพื้นฐานมาจาก Entity-Relationship Diagram(ERD) แต่ก็จะมีข้อแตกต่างบ้างยกตัวอย่าง เช่น มันจะบอกความสัมพันธ์และเกี่ยวข้องกันระหว่าง ออบเจ็กต์ เมื่อดูตามรูปที่ 2-3 จะแสดงแบบของฐานข้อมูลของบริษัทแห่งหนึ่ง รูปที่ 2-4 และรูปที่ 2-5 แสดงแบบของฐานข้อมูลของโครงการทดลอง นิวเคลียร์



รูปที่ 2-4 แสดงแบบของฐานข้อมูลของบริษัทแห่งหนึ่ง



รูปที่ 2-5 แสดงแบบของฐานข้อมูลของโครงการทดลอง นิวเคลียร์



รูปที่ 2-6 แสดงแบบของฐานข้อมูลของ โครงการทดลอง นิวเคลียร์

โดย ORD นั้นเป็นจุดเริ่มต้นของการพัฒนาและเป็นเครื่องมือในการออกแบบฐานข้อมูลเป็น OODB โดยมีสัญลักษณ์และเครื่องมือที่หลากหลายมากที่จะเป็นตัวแสดงโดยจะมีส่วนร่วมกับ ER diagram ซึ่งความหมายของสัญลักษณ์ต่างๆ มีดังนี้

- สัญลักษณ์สี่เหลี่ยม : แทน Object Class ซึ่งจะเป็นตัวบ่งบอก ออบเจ็กต์ ของ คลาส โดยตรง
- X- : แทนตัวระบุว่า Object Class นั้นไม่สามารถลบได้โดยตรง ซึ่งจะกำหนดเป็นค่า Default ของ X-
- สัญลักษณ์แบบเพชร : แทนการต่อเชื่อมกันของความสัมพันธ์ระหว่างคลาส
- สัญลักษณ์ลูกศร : แทนการแสดงความสัมพนธ์ระหว่างคลาส โดยอาจทำการกลับความสัมพันธ์ได้โดยกลับหัวลูกศร

โดยสัญลักษณ์สี่เหลี่ยมก่อนหัวลูกศรจะเรียกว่า Subject Class และหลังหัวลูกศรจะเรียกว่า Relative Class และชื่อนั้นจะมีความสัมพันธ์กันอยู่ในคลาส ออบเจ็กต์

ความสัมพันธ์แบบ “is a” ใน ORD เป็นตัวระบุชี้ว่า Subject Class เป็น Subclassของ Relative Class โดยจะเรียกคลาสนี้ว่า Super Classและ ออบเจ็กต์ ของ Subject Class เป็น ออบเจ็กต์ ของ Relative Class. โดย Subclass นั้นสามารถที่จะเป็นเจ้าของ attribute Behavior และมีการถ่ายทอดคุณสมบัติที่ชัดเจนของความสัมพันธ์จาก Superclass(es) อย่างไรก็ตาม Cardinality นั้นไม่กำหนดความสัมพันธ์แบบ “is a”

ระดับขั้นของ OID นั้นพิจารณาจากความสัมพันธ์ระหว่างคลาส ตัวอย่างเช่นความสัมพันธ์ระหว่างคลาส Employees และ คลาส Car pool โดยเป็นแบบ ~ X~ <2..- to – 0/1> ถ้าต้องการที่จะ Binding สิ่งแรกที่ต้องทำคือกำหนดค่าเริ่มต้นให้เป็น Left off ORD ซึ่งก็จะได้เป็น Level ที่ 1 ของ ORD และภายหลังทำการวิเคราะห์และออกแบบต่อ ซึ่งก็จะได้เป็น Level ที่ 2 ของ ORD

บทที่ 3

UML (Unified Modeling Language)

3.1 Unified Modeling Language

Unified Modeling Language เป็นการสร้างโมเดลจำลอง(Abstract Model) ที่มีสัญลักษณ์ (notation) ที่มีประสิทธิภาพซึ่งจะช่วยในส่วนของการวิเคราะห์ (Analysis) และออกแบบ(Design)ซึ่งมีประโยชน์ในหลายด้านสำหรับใช้ในการออกแบบระบบต่างๆโดยจะเป็นในรูปแบบของกสรออกแบบข้อมูลแบบออบเจ็กต์โอเรียนเท็ดซึ่งการออกแบบจะทำให้เราสามารถวิเคราะห์ส่วนประกอบโดยรวมของระบบและยังสามารถที่จะรองรับภาษาที่จะใช้ในการพัฒนาได้หลากหลายภาษามากและยังหลีกเลี่ยงความซ้ำซ้อนและเฉพาะเจาะจงเกินไป ซึ่งการออกแบบโดยวิธี UML นี้สามารถที่จะแสดงออกมาได้ในรูปแบบของไดอะแกรม (diagram) ต่างๆ

การพัฒนาซอฟต์แวร์ในปัจจุบันนั้นเกิดปัญหาขึ้นมากมายเช่น ระบบที่พัฒนามาไม่ตรงตามความต้องการของลูกค้าหรืออาจใช้เวลามากเกินไป ซึ่งก็ได้มีการพัฒนาแก้ไขปัญหาโดยใช้เทคนิคใหม่ ๆ เข้ามาช่วยแต่ก็ยังคงเกิดปัญหาขึ้นอยู่ดี ทางแก้ของปัญหาต่าง ๆ นี้คือต้องมีการสร้างโมเดลจำลอง ของระบบขึ้นมาก่อน ซึ่งจะช่วยให้มีข้อดีต่างๆ ดังนี้

1. ความถูกต้องตามความต้องการจริงของระบบ
2. ทุก ๆ ส่วนของระบบมีความสอดคล้องกัน
3. ง่ายในการสื่อสารระหว่างผู้พัฒนาและผู้ใช้ระบบ
4. สามารถเปลี่ยนแปลงได้ง่าย
5. สามารถทำความเข้าใจได้ง่ายทั้งผู้พัฒนาและผู้ใช้ระบบ

3.2 ส่วนประกอบของ UML

UML ประกอบด้วยหลาย ๆ ส่วน ดังนี้

View: แสดงมุมมองต่าง ๆ ของระบบที่ถูกออกแบบขึ้นมาโดยจะใช้ไดอะแกรมต่าง ๆ ในการอธิบาย

Diagram: เป็นไดอะแกรมที่ใช้ในการอธิบายส่วนต่าง ๆ ของ view ซึ่งใน UML มีไดอะแกรมทั้งหมด 9 ไดอะแกรม

Model element: เป็นสัญลักษณ์ที่ใช้ในไดอะแกรมเพื่อแสดงหรือเป็นตัวแทนของสิ่งต่างๆ เช่น คลาส (class),ออบเจ็กต์,แมสเซส (message) และความสัมพันธ์ (relationship) เป็นต้น

General mechanism: เป็นส่วนที่แสดงคอมเม้นเพิ่มเติม (extra comment), ข้อมูลอื่น ๆ ที่จำเป็นหรือความหมายของโมเดลเอลิเม้น (model element)

3.2.1 view

ในการออกแบบระบบที่มีขนาดใหญ่และมีความซับซ้อนมาก ๆ นั้นจะทำให้ผู้ออกแบบระบบไม่สามารถที่จะออกแบบระบบได้ครบถ้วน ดังนั้นจึงต้องมีการมองระบบเป็นมุมมองต่างๆ เพื่อทำให้ง่ายในการออกแบบ เช่น มุมมองด้านฟังก์ชั่นนอล , นอนฟังก์ชั่นนอล (Nonfunctional), มุมมองขององค์กร เป็นต้น ดังนั้นระบบจึงมี view ที่ต่างๆ กันซึ่งแต่ละ view จะแสดงมุมมองเฉพาะของระบบซึ่งอธิบายรวมกันเป็นระบบที่สมบูรณ์ ซึ่งจะประกอบด้วย View ต่าง ๆ ดังนี้

3.2.1.1 Use-case View

อธิบายการทำงานต่าง ๆ ของระบบที่ถูกมองจากภายนอกหรือผู้ใช้ระบบ Use-case view ซึ่งอธิบายโดย ยูสเคซไดอะแกรม (use-case diagram) เป็นมุมมองสำหรับลูกค้า , ผู้ออกแบบ , ผู้พัฒนาระบบและผู้ทดสอบระบบ

3.2.1.2 Logical View

อธิบายการทำงานต่าง ๆ ที่ถูกออกแบบไว้ภายในระบบ ว่าระบบจะมีบริการอะไรให้กับผู้ใช้บ้าง โดยจะแสดงโครางสร้างแบบสถิต (Static) เช่น คลาส, ออบเจ็กต์, ความสัมพันธ์และการทำงานร่วมกันแบบไดนามิก (dynamic collaboration) ซึ่งจะเกิดขึ้นเมื่อออบเจ็กต์ ส่งแมสเซส ระหว่างกันในการทำงาน

โครงสร้างแบบสถิตจะอธิบายโดยใช้ คลาสไดอะแกรม (Class diagram) และออบเจ็กต์ไดอะแกรม (object diagram) ส่วนการทำงานร่วมกันแบบไดนามิกจะอธิบายโดยใช้ สเตทไดอะแกรม (state diagram), ซีเควันซ์ไดอะแกรม (sequence diagram), ค็อลแลโบเรชั่นไดอะแกรม (collaboration diagram) และ แอคติวิตี้ไดอะแกรม (activity diagram)

3.2.1.3 Component View

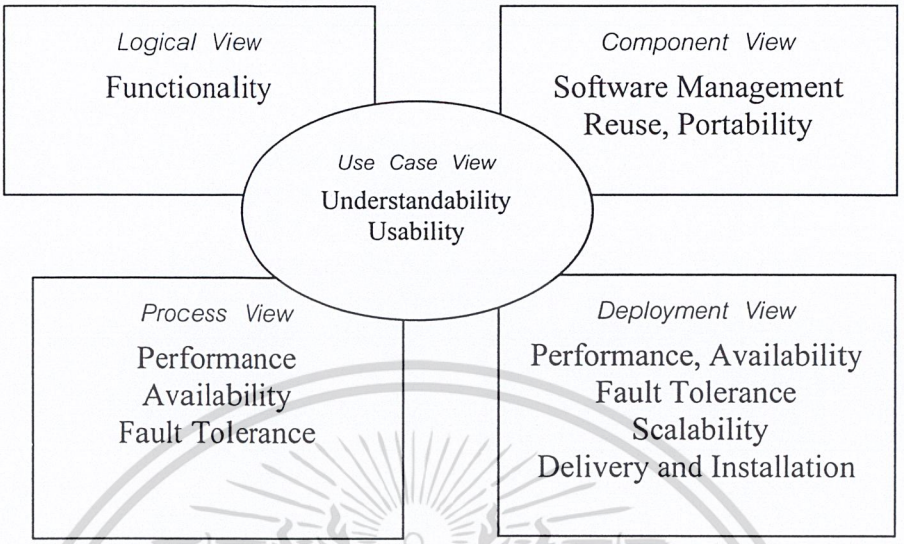
อธิบายการสร้างและความขึ้นต่อกันของโมดูล (Module) ที่ใช้ในการพัฒนาระบบ โดยใช้คอมโพเน้นไดอะแกรม (Component diagram) ในการอธิบาย

3.2.1.4 Deployment View

อธิบายการจัดวางระบบให้เหมาะในด้านกายภาพ (Physical) แสดงด้วยคอมพิวเตอร์และโหนด (nodes) ต่าง ๆ เพื่อให้ระบบมีเสถียรภาพมากขึ้น โดยใช้ ดีพลอยเม้นไดอะแกรม (deployment diagram) ในการอธิบาย

3.2.1.5 Process View

แสดงการทำงานร่วมกันและการติดต่อกันของส่วนต่าง ๆ ในระบบ



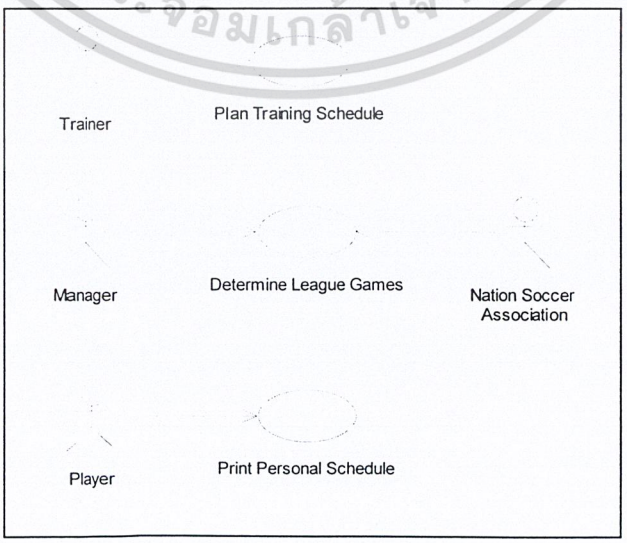
รูปที่ 3-1 สถาปัตยกรรมของ View

3.2.2 Diagram

เป็นไดอะแกรมซึ่งแสดงสัญลักษณ์ที่ถูกจัดเรียงเพื่ออธิบายระบบในมุมมองต่างๆซึ่งในระบบหนึ่ง ๆ จะประกอบด้วยหลาย ๆ ไดอะแกรม และในแต่ละไดอะแกรมยังสามารถมองในหลาย ๆ มุมมองได้ UML มีไดอะแกรมที่ต่างกันอยู่ 9 ไดอะแกรม ดังนี้

3.2.2.1 Use Case Diagram

ยูสเคซไดอะแกรมจะแสดงความสัมพันธ์ระหว่างผู้ใช้งาน (User) กับระบบ โดยใช้แอ็กเตอร์ (actors) แทนผู้ใช้งานและแอ็กเตอร์จะต้องติดต่อกับระบบโดยผ่าน ยูสเคซ (use case) ต่าง ๆ ซึ่งยูสเคซในยูสเคสไดอะแกรม ก็คือการทำงานต่าง ๆ ของระบบที่ผู้ใช้ต้องการ



รูปที่ 3-2 แสดงตัวอย่างยูสเคสไดอะแกรมของการบริหารทีมฟุตบอล

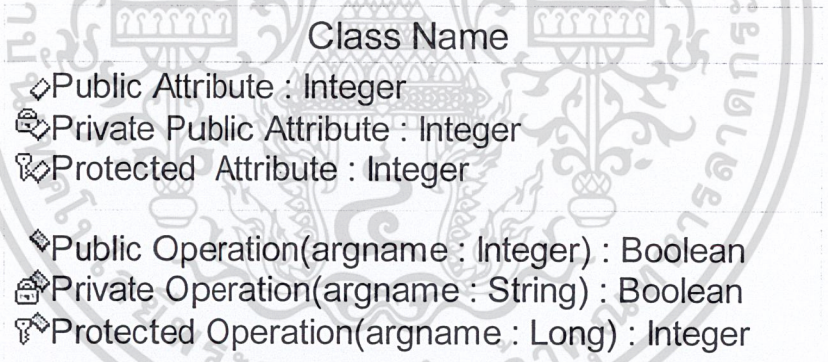
แอ็กเตอร์ไม่ใช่ส่วนประกอบของระบบ แต่จะเป็นส่วนที่ใช้ติดต่อกับระบบ ซึ่งอาจจะเป็นเพียงการป้อนข้อมูลเข้าไปในระบบหรือเป็นเพียงการรับข้อมูลจากระบบ หรืออาจจะเป็นทั้งสองอย่างก็ได้ ในระบบหนึ่งๆ จะต้องมี แอ็กเตอร์ซึ่งจะต้องทำการกำหนดคุณลักษณะของแอ็กเตอร์(identify actor) ให้ได้ก่อน ดังนี้

1. ใครเป็นผู้ใช้งานระบบ
2. ใครที่มีความเหมาะสมที่จะใช้งานระบบ
3. ใครที่มีส่วนสนับสนุนข้อมูลในระบบ ใช้ข้อมูลในระบบและแก้ไขข้อมูลในระบบ
4. ใครที่สนับสนุนการบำรุงรักษาระบบ(Maintenance)
5. ระบบมีการใช้งานกับภายนอกหรือเปล่า

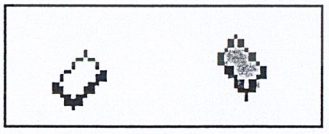
3.2.2.2 Class Diagram

คลาสไดอะแกรมจะแสดงโครงสร้างที่ไม่เปลี่ยนแปลงของระบบในเทอมของคลาส (Static Diagram) และแสดงความสัมพันธ์ระหว่างคลาสต่าง ๆ เหล่านั้น ซึ่งในระบบหนึ่ง ๆ จะประกอบด้วยหลาย ๆ คลาสไดอะแกรม

คลาส คือ กลุ่มของออบเจ็กต์ที่มีคุณสมบัติ(attributes) และพฤติการณ์ (behavior) ร่วมกัน สัญลักษณ์ของคลาส แสดงดังรูปที่ 3-3ส่วนแรกเก็บ ชื่อคลาส ส่วนที่สองเก็บคุณสมบัติ ส่วนที่สามเก็บการทำงาน(operation) โดยที่คลาส 1 คลาส จะต้องมีอย่างน้อย 1 ส่วน คือ ชื่อคลาส



รูปที่ 3-3 สัญลักษณ์ของคลาสไดอะแกรม

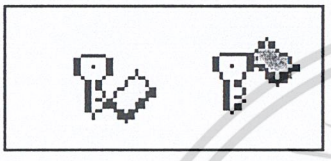


รูปที่ 3-4 สัญลักษณ์ของแอตทริบิวต์และ โอเปอเรชันชนิด Public

สัญลักษณ์ แสดงแอตทริบิวต์และโอเปอเรชันที่มีการมองเห็นแบบ Public คือแอตทริบิวต์และโอเปอเรชันที่สามารถมองเห็นได้จากภายนอก สามารถเข้าไปเปลี่ยนค่า อ่านค่า หรือเรียกใช้งานแอตทริบิวต์และโอเปอเรชันนั้นได้ทันทีโดยอิสระจากภายนอก

รูปที่ 3-5 สัญลักษณ์ของ แอตทริบิวต์ และ โอเปอเรชัน ชนิด Private

สัญลักษณ์ แสดง แอตทริบิวต์s และ โอเปอเรชันที่มี Visibility แบบ Private คือ แอตทริบิวต์ และ โอเปอเรชันที่ไม่สามารถมองเห็นได้จากภายนอกของ Class แต่สามารถมองเห็นได้จากภายในตัว Class เองเท่านั้น หากภายนอกต้องการที่จะเข้ามาเพื่อแก้ไขหรืออ่านค่าหรือเรียกใช้ แอตทริบิวต์ และ โอเปอเรชัน ที่เป็น Visibility นั้นจะทำได้เพียงวิธีเดียวคือทำผ่าน โอเปอเรชัน ที่เกี่ยวข้องสัมพันธ์กับ แอตทริบิวต์ หรือ โอเปอเรชัน ที่เป็น Private ดังกล่าว



รูปที่ 3-6 สัญลักษณ์ของ แอตทริบิวต์ และ โอเปอเรชัน ชนิด Protected

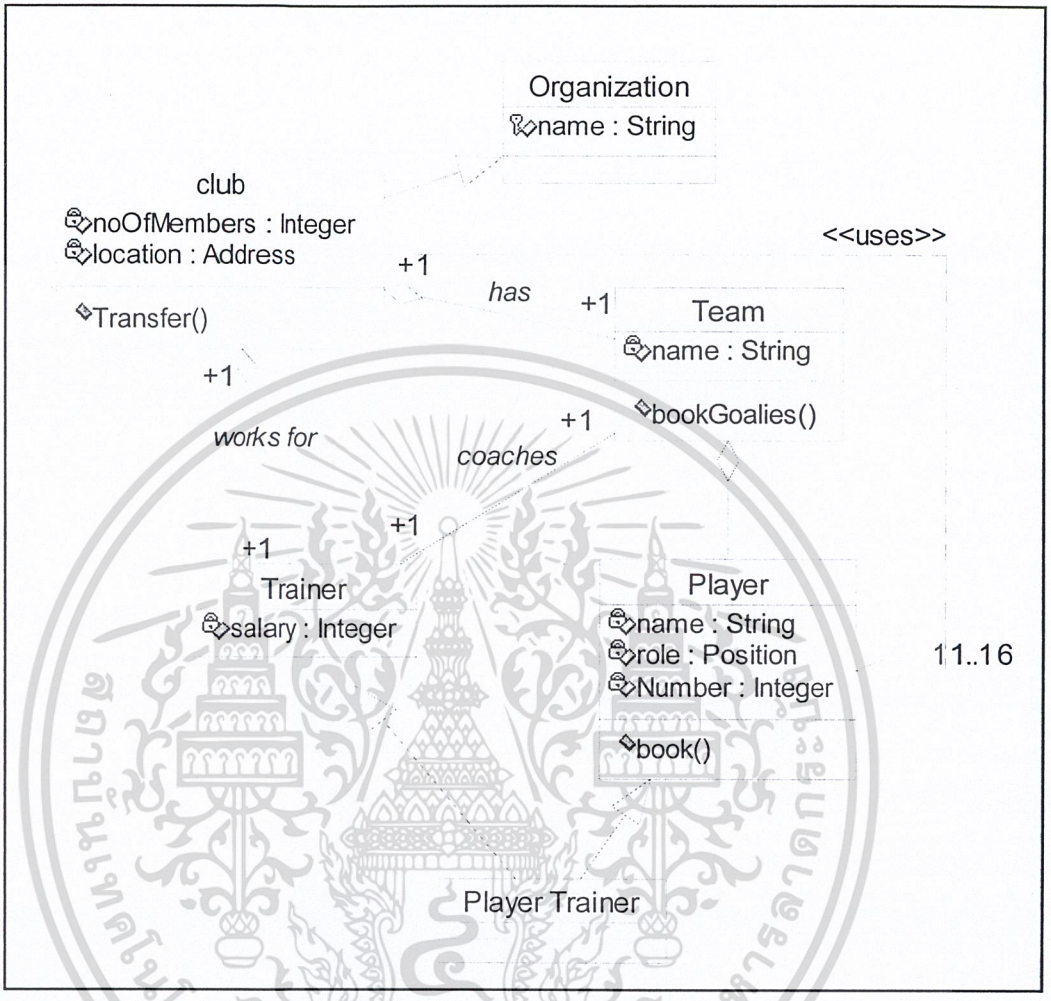
สัญลักษณ์ แสดง แอตทริบิวต์ และ โอเปอเรชันที่มี Visibility แบบ Protected คือ แอตทริบิวต์ และ โอเปอเรชัน ที่สงวนไว้สำหรับการทำ Inheritance (Specialization) โดยเฉพาะ โดย แอตทริบิวต์s และ โอเปอเรชัน เหล่านี้จะเป็นของ Superclass เมื่อทำ Inheritance แล้ว แอตทริบิวต์s และ โอเปอเรชัน ที่มี Visibility แบบ Protected จะกลายไปเป็น Private แอตทริบิวต์/โอเปอเรชัน หรือ Protected แอตทริบิวต์/โอเปอเรชัน ของ SubClass (โดยอาจจะขึ้นอยู่กับข้อกำหนดของภาษาที่ใช้ได้)

สัญลักษณ์ความสัมพันธ์ต่าง ๆ ของคลาสไดอะแกรม มีดังนี้

1	หมายถึงจะมีออบเจ็กต์ในคลาสไดอะแกรมได้หนึ่งออบเจ็กต์เท่านั้น
0..1	หมายถึงจะมีออบเจ็กต์ในคลาสไดอะแกรมได้แค่หนึ่งหรืออาจจะไม่มีก็ได้
M..N	ออบเจ็กต์ในคลาสไดอะแกรมได้ตั้งแต่ M ถึง N(เมื่อ M และ N เป็นจำนวนเต็มบวก)
*	หมายถึงจะมีออบเจ็กต์ในคลาสไดอะแกรมได้ตั้งแต่ศูนย์ขึ้นไป
0..*	หมายถึงจะมีออบเจ็กต์ใน คลาสไดอะแกรมได้ตั้งแต่ศูนย์ขึ้นไป
1..*	หมายถึงจะมีออบเจ็กต์ใน คลาสไดอะแกรมได้ตั้งแต่หนึ่งขึ้นไป

ตาราง 3-1 แสดงสัญลักษณ์ความสัมพันธ์ต่าง ๆ ของคลาสไดอะแกรม

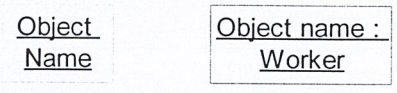
ตัวอย่าง คลาสไดอะแกรมกับ object type สำหรับการจัดการทีมฟุตบอล



รูปที่ 3-7 ตัวอย่างของ คลาสไดอะแกรม สำหรับระบบการจัดการทีมฟุตบอล

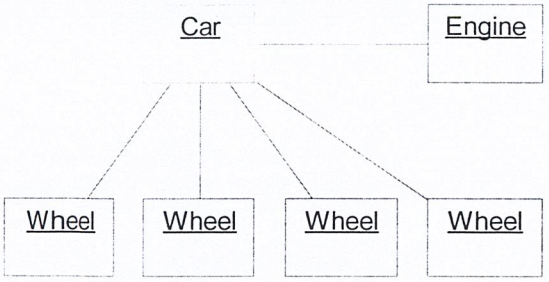
2.2.2.3 Object Diagram

ออบเจ็กต์ไดอะแกรมจะได้มาจากคลาสไดอะแกรม ซึ่งจะเป็นการแสดงการเชื่อมต่อระหว่างอินซแท็นซ์(instances) ต่าง ๆ ที่อยู่ภายในคลาสไดอะแกรมอย่างละเอียด เพื่อใช้เป็นตัวอย่างสำหรับคลาสไดอะแกรมที่มีความซับซ้อนโดยจะนำมาแทนเป็น ออบเจ็กต์ จริงๆ และแสดงความสัมพันธ์ต่างๆ ให้เห็น โดยสัญลักษณ์ของอินซแท็นซ์ในออบเจ็กต์ไดอะแกรม จะใช้การขีดเส้นใต้เป็นสัญลักษณ์ ภายในกรอบสี่เหลี่ยมดังรูปที่ 2-7



รูปที่ 3-8 สัญลักษณ์ของ Instance ใน Object diagram

ตัวอย่าง ออบเจ็กต์ไดอะแกรมอธิบายโครงสร้างของรถยนต์ (Car) ซึ่งรถแต่ละคันจะมีเครื่องยนต์ (engine) หนึ่งเครื่องและมีล้อ 4 ล้อ



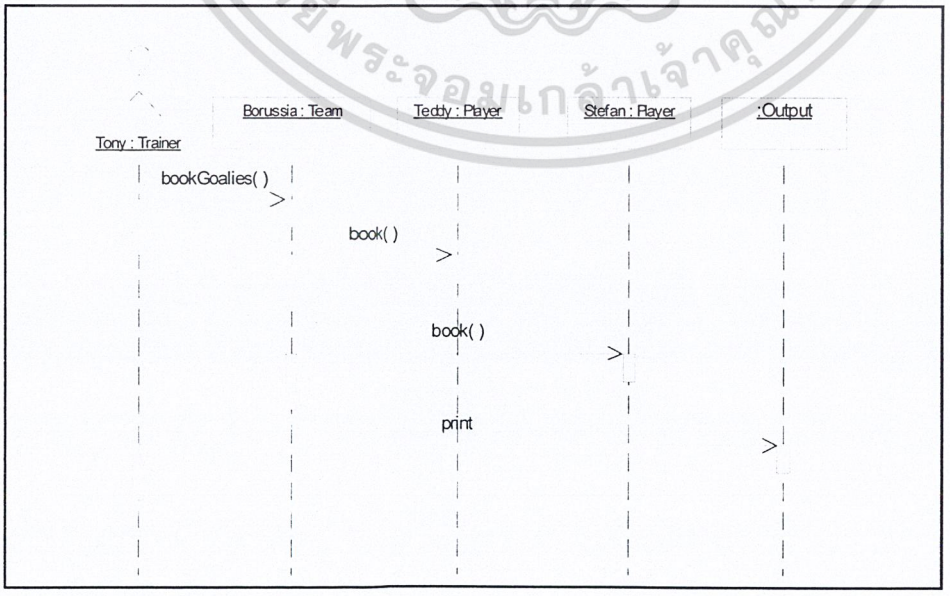
รูปที่ 3-9 Object diagram ของรถยนต์

3.2.2.5 Sequence Diagram

ซีเควันซ์ไดอะแกรมแสดงการติดต่อโต้ตอบระหว่างออบเจ็กต์ ที่เน้นไปที่ลำดับก่อนหลังและเวลามีความสำคัญในการเปลี่ยนแปลง จะมีการส่งแมสเซส เพื่อให้ออบเจ็กต์ ต่าง ๆ ที่อยู่ในระบบทำงาน โดยใช้สัญลักษณ์ที่เรียกว่า ออบเจ็กต์แมสเซส ซีเควันซ์ชาท (Object message sequence chart) ในการแสดงดังรูปที่ 3-10

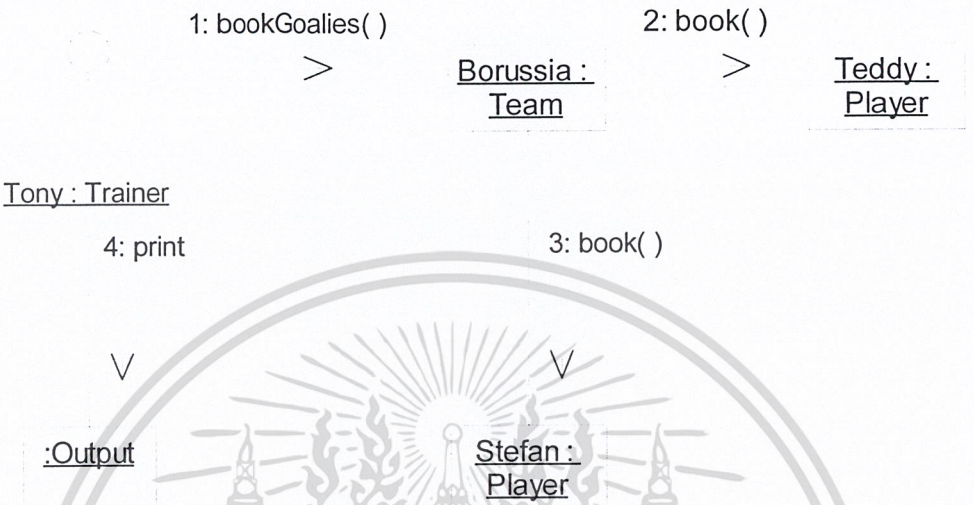
รูปที่ 3-10 object message sequence chart

ตัวอย่าง ซีเควันไดอะแกรม ของการวางแผนทีมฟุตบอล



รูปที่ 3-11 แสดงตัวอย่างซีเควันไดอะแกรมของการบริหารการจัดการทีมฟุตบอล

3.2.2.6 ค็อลแลโบเรชั่นไดอะแกรม ใช้อธิบายการทำงานร่วมกันของออบเจ็กต์ที่มีส่งแมสเซส ระหว่างกัน เพื่อบอกถึงลำดับขั้นตอนการทำงานของระบบ โดยใช้สัญลักษณ์ลูกศรเป็นตัวกำหนดทิศทางซึ่ง ค็อลแล โบเรชั่นไดอะแกรมจะเป็นส่วนอธิบายการทำงานของออบเจ็กต์ไดอะแกรม

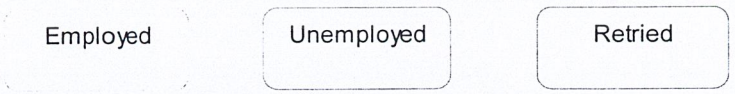


รูปที่ 3-12 แสดงค็อลแลโบเรชั่นไดอะแกรม สำหรับการสนทนาทางโทรศัพท์

จากรูปที่ 3-12 จะมีการติดต่อกันระหว่าง ออบเจ็กต์ โดยมีลำดับก่อนหลังในการส่งแมสเซส ให้แต่ละออบเจ็กต์ทำงาน

3.2.2.6 State Diagram

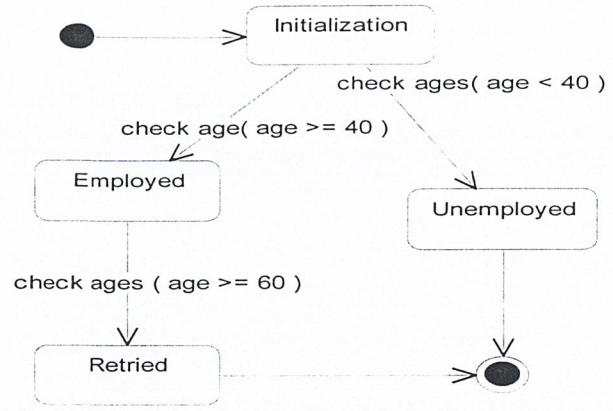
สเตทไดอะแกรมใช้อธิบายคลาสต่างๆ ในระบบโดยจะแสดงทุก ๆ สถานะที่เป็นไปได้และเหตุการณ์ที่ทำให้ออบเจ็กต์เหล่านั้นเกิดการเปลี่ยนแปลง โดยเหตุการณ์ที่ทำให้เกิดการเปลี่ยนแปลงอาจเกิดจากออบเจ็กต์อื่นส่งแมสเซสมา การเปลี่ยนแปลงสถานะเรียกว่า ทรานซิชั่น (Transitions) จากรูปที่ 2.10 แสดงสเตทของบุคคล (person) อาจจะเป็นได้ทั้ง 3 สเตท คือ ไม่ถูกจ้าง (Unemployed), ถูกจ้าง(Employed) และ ถูกปลด (Retired)



รูปที่ 3-13 แสดงสเตทต่างๆ ในไดอะแกรม

สเตทของบุคคลจะขึ้นอยู่กับเงื่อนไขต่าง ๆ ที่บริษัทกำหนด เช่น บริษัทจะรับลูกจ้างอายุ 40 ปี ขึ้นไปและจะถูกปลดเมื่ออายุ 60 ปี

โดยสามารถนำเอาลักษณะของสถานะต่างๆที่เรากำหนดมาแสดงเป็นสเตทไดอะแกรมของบุคคล(person) ได้ดังนี้



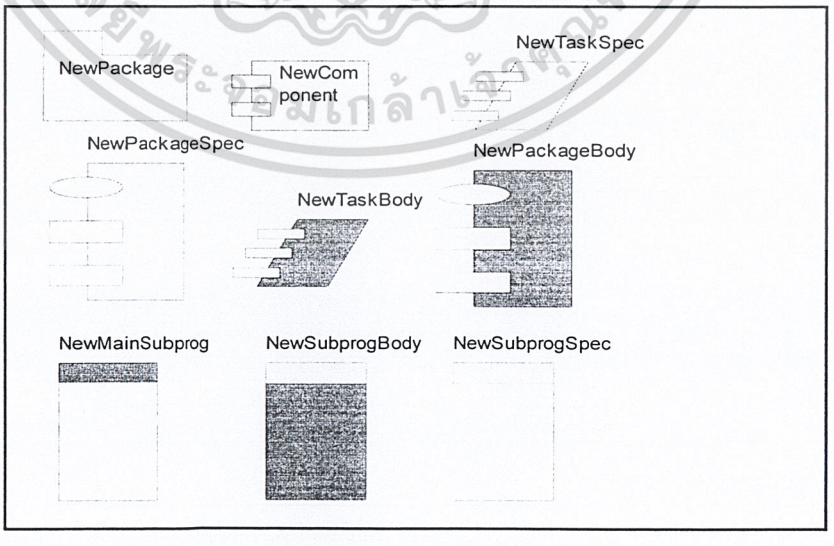
รูปที่ 3-14 แสดง สเตทไดอะแกรมของการทำงานของบุคคล โดยอาจพิจรณามาจากเงื่อนไขกฎข้อบังคับของระบบ

3.2.2.7 Activity diagram

แอคติวิตี้ไดอะแกรมแสดงลำดับการไหลของกิจกรรม(Activity) ต่าง ๆ โดยจะอธิบายกิจกรรมต่าง ๆ ในลักษณะของการกระทำในไดอะแกรม จะแสดงเป็นสถานะการกระทำ(Action State) ซึ่งสถานะจะเปลี่ยนไปเมื่อเกิดการกระทำอย่างใดอย่างหนึ่งเกิดขึ้นตามเงื่อนไขหรือการตัดสินใจที่กำหนดไว้เพื่อควบคุมการไหลของกิจกรรมรวมถึงสามารถมีแมสเซสที่รับ – ส่งระหว่างแต่ละกิจกรรม

3.2.2.8 Component diagram

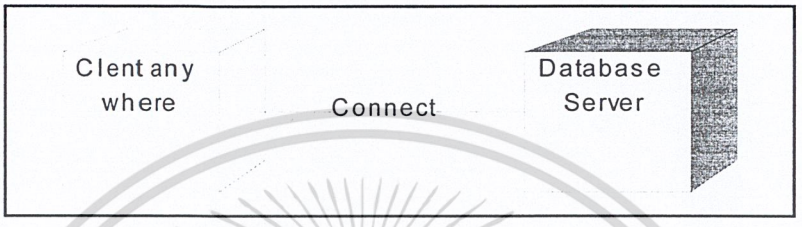
ค็อมโพเน็นท์ไดอะแกรมแสดงโครงสร้างทางกายภาพของโค้ดในรูปค็อมโพเน็นท์ของโค้ด (Code) อาจเป็นส่วนประกอบของซอสโค้ด (source code) คอมโพเน้นจะประกอบด้วยข้อมูลเกี่ยวกับ ลอจิคัล คลาส (logical class) ของคอมโพเน้นในไดอะแกรม มีการแสดงความสัมพันธ์หรือความพึ่งพากันของค็อมโพเน็นท์ให้ช่วยในการวิเคราะห์ว่าเมื่อเกิดการเปลี่ยนแปลงของ คอมโพเน้นหนึ่งจะมีผลต่อคอมโพเน้นอื่น ๆ อย่างไร ซึ่งช่วยในการโปรแกรม



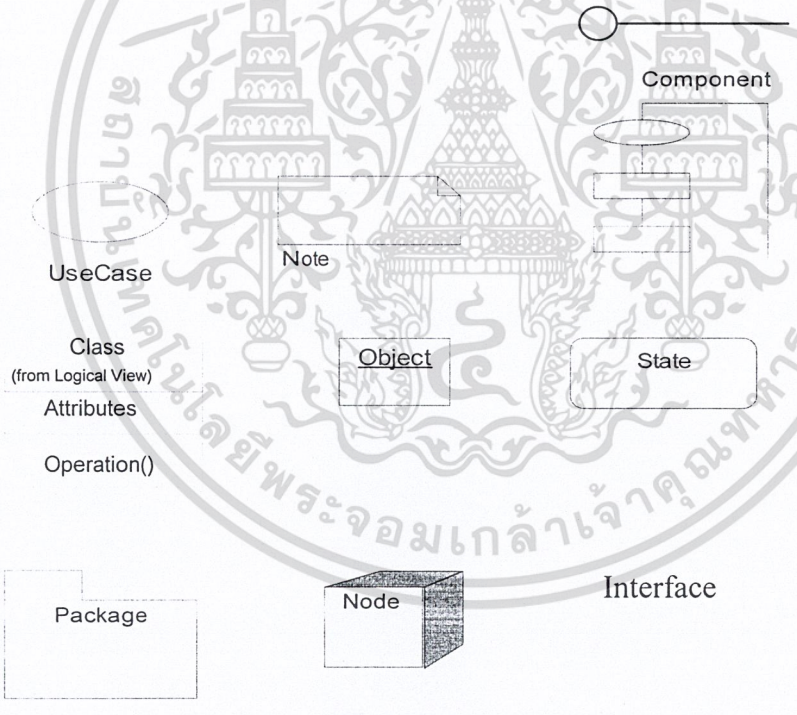
รูปที่ 3-15 แสดงสัญลักษณ์ของคอมโพเน้นไดอะแกรม

3.2.2.9 **Deployment diagram**

ดีพลอยเม้นไดอะแกรมแสดงสถาปัตยกรรมทางกายภาพของส่วนประกอบต่างๆของฮาร์ดแวร์(hardware) ซึ่งถูกเรียกว่า โหนด (Node) ในระบบซึ่งสามารถแสดงเป็นโหนดและการเชื่อมต่อระหว่างกัน ชนิดของการเชื่อมต่อ นอกจากนี้ภายในโหนด ยังสามารถมีคอมโพเน้น หรือออบเจ็กต์ที่สามารถปฏิบัติกับโหนดเพื่อแสดงว่าโปรแกรมส่วนใดถูกปฏิบัติบนโหนดใดและความสัมพันธ์ระหว่างคีอมโพเน็นท์ที่อยู่บนโหนด ซึ่งโหนดจะถูกแทนด้วยสัญลักษณ์ของสี่เหลี่ยมลูกบาศก์ดังรูปที่ 2-15



รูปที่ 3-16 แสดงดีพลอยเม้นไดอะแกรมของ การเชื่อมต่อระหว่าง Client ในที่ต่าง ๆ และ Database



รูปที่ 3-17 แสดง model element ที่ใช้ร่วมกันในไดอะแกรม

3.2.3 Model element

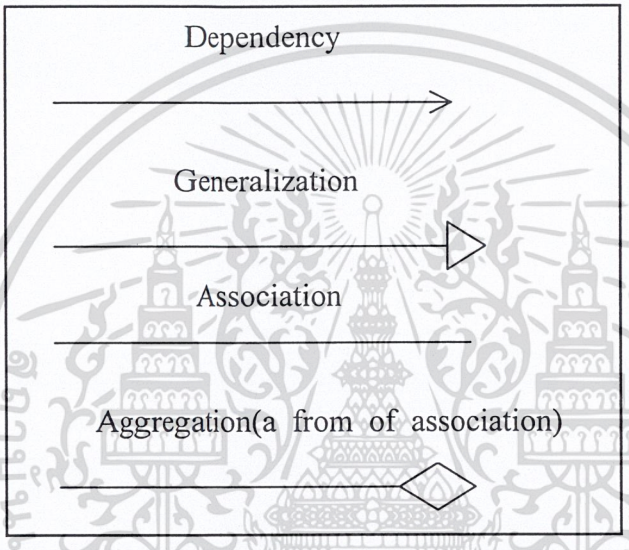
เป็นสัญลักษณ์ที่ใช้ในไดอะแกรมเพื่อแสดงหรือเป็นตัวแทนของสิ่งต่าง ๆ เช่น คลาส ,ออบเจ็กต์ แมสเซส และความสัมพันธ์ ต่าง ๆ ที่มี เช่น

Association ใช้เชื่อมต่อระหว่างอุปกรณ์ที่มีความสัมพันธ์กัน

Generalization บางครั้งเรียกว่า อินเฮอริแท้น (inheritance) ซึ่งเป็นความสัมพันธ์แบบพิเศษระหว่างอุปกรณ์ หนึ่งที่มีต่ออีกอุปกรณ์หนึ่ง

Dependency การพึ่งพากันระหว่างอุปกรณ์ในทางใดทางหนึ่ง

Aggregation เป็นรูปแบบการแสดงอุปกรณ์ที่ประกอบจากอุปกรณ์อื่นหลายอุปกรณ์



รูปที่ 3-18 ตัวอย่างของความสัมพันธ์แบบต่าง ๆ

3.3 Object Constraint Language OCL เป็นส่วนหนึ่งภายใน UML และสามารถใช้กำหนดความต้องการและกำหนดกฎข้อบังคับในไดอะแกรมต่างๆ ได้ โดย OCL นั้นประกอบไปด้วยคุณสมบัติของ Expression Language, Modeling Language และ Formal Language

3.3.1 Expression Language โดยเมื่อใช้ Expression Language ภายใน OCL นั้นจะสามารถยอมรับได้ว่าจะไม่มีผลต่อการทำงานหรือผลประทบใดๆ ภายในไดอะแกรมได้ รวมทั้งค่าทั้งหมดของออบเจ็กต์ และการส่งค่าต่างจะไม่มีการเปลี่ยนแปลงไป

3.3.2 Modeling Language โดยภายใน OCL นั้นมีคุณสมบัติของ Modeling Language คือจะไม่สามารถที่จะเขียนโปรแกรมควบคุมการทำงานภายใน OCL ได้ ไม่สามารถร้องของ Process และไม่สามารถทำการ Query Operation ได้

3.3.3 Formal Language โดยพื้นฐานของ Formal Language นั้นจะมีส่วนประกอบจากภาษาคณิตศาสตร์ ทำให้สามารถเข้าใจเป็นในทางเดียวกันได้ แต่มีข้อเสียคือเป็นการยากในการใช้งานสำหรับการออกแบบในเชิงธุรกิจและนำเสนอ

ตัวอย่างของ OCL

```
An interface can only contain operations
self.allFeatures->forall (f |f.oclIsKindOf(Operation))
```

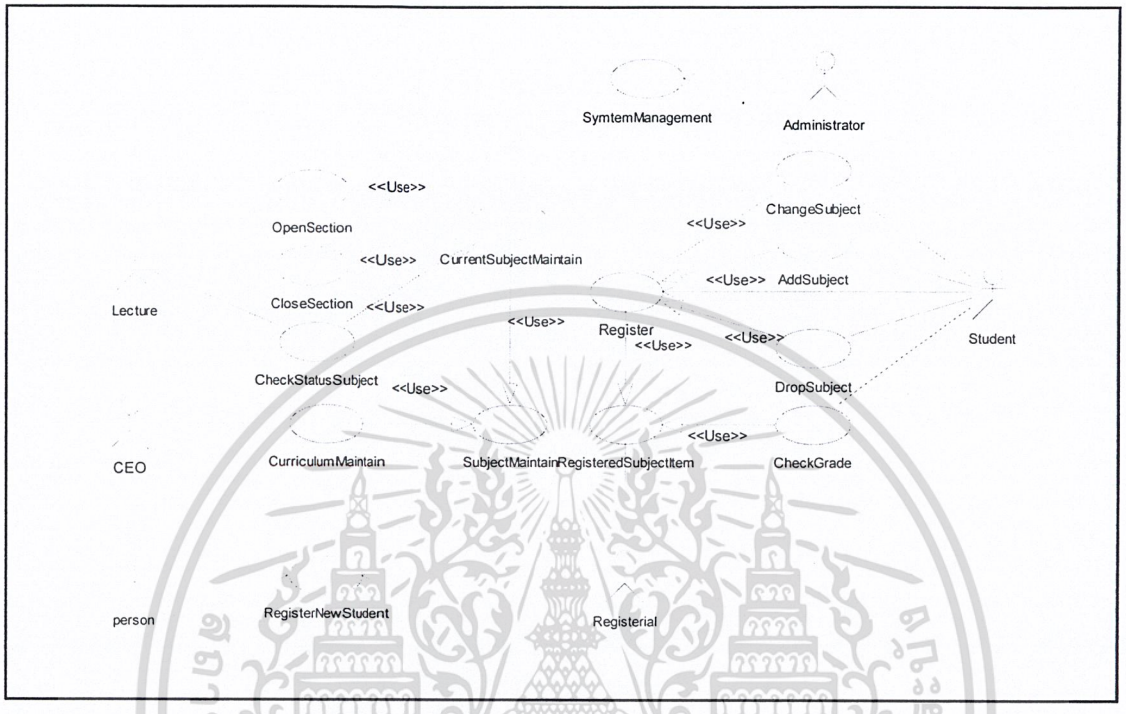
รูปที่ 3-19 แสดงตัวอย่างการกำหนด OCL

```
All features defined in an interface are public
Self.allFeatures->forall (f | f.visibility = #public)
```

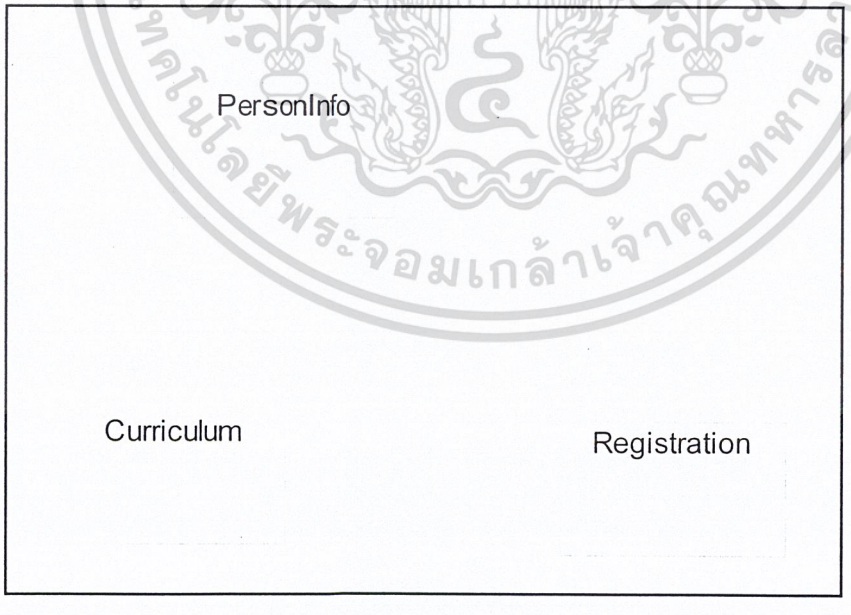
รูปที่ 3-20 แสดงตัวอย่างการกำหนด OCL

ทำการออกแบบระบบกรณีศึกษาระบบลงทะเบียนนักศึกษา

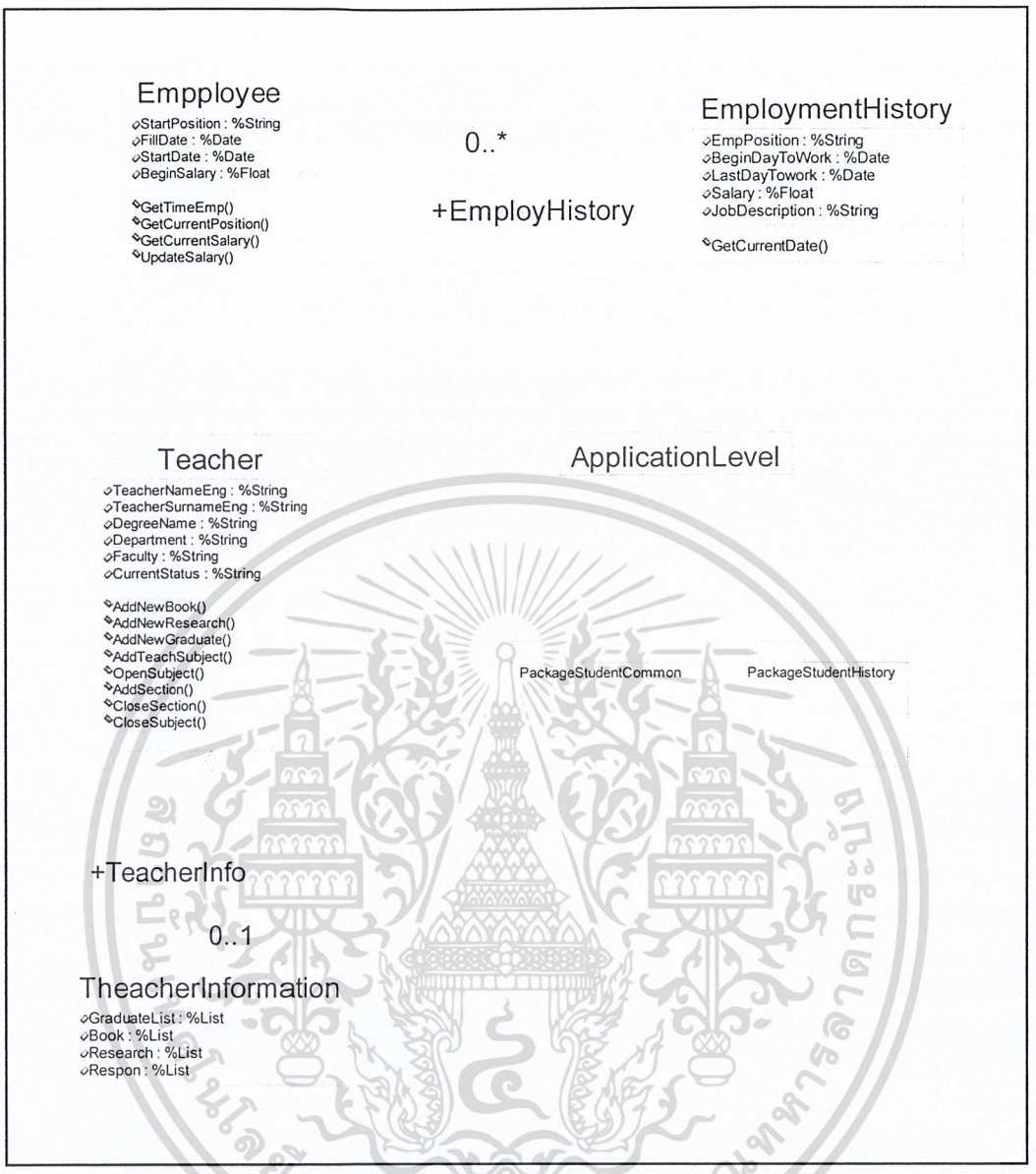
แสดงยูสเคซของระบบลงทะเบียนนักศึกษา



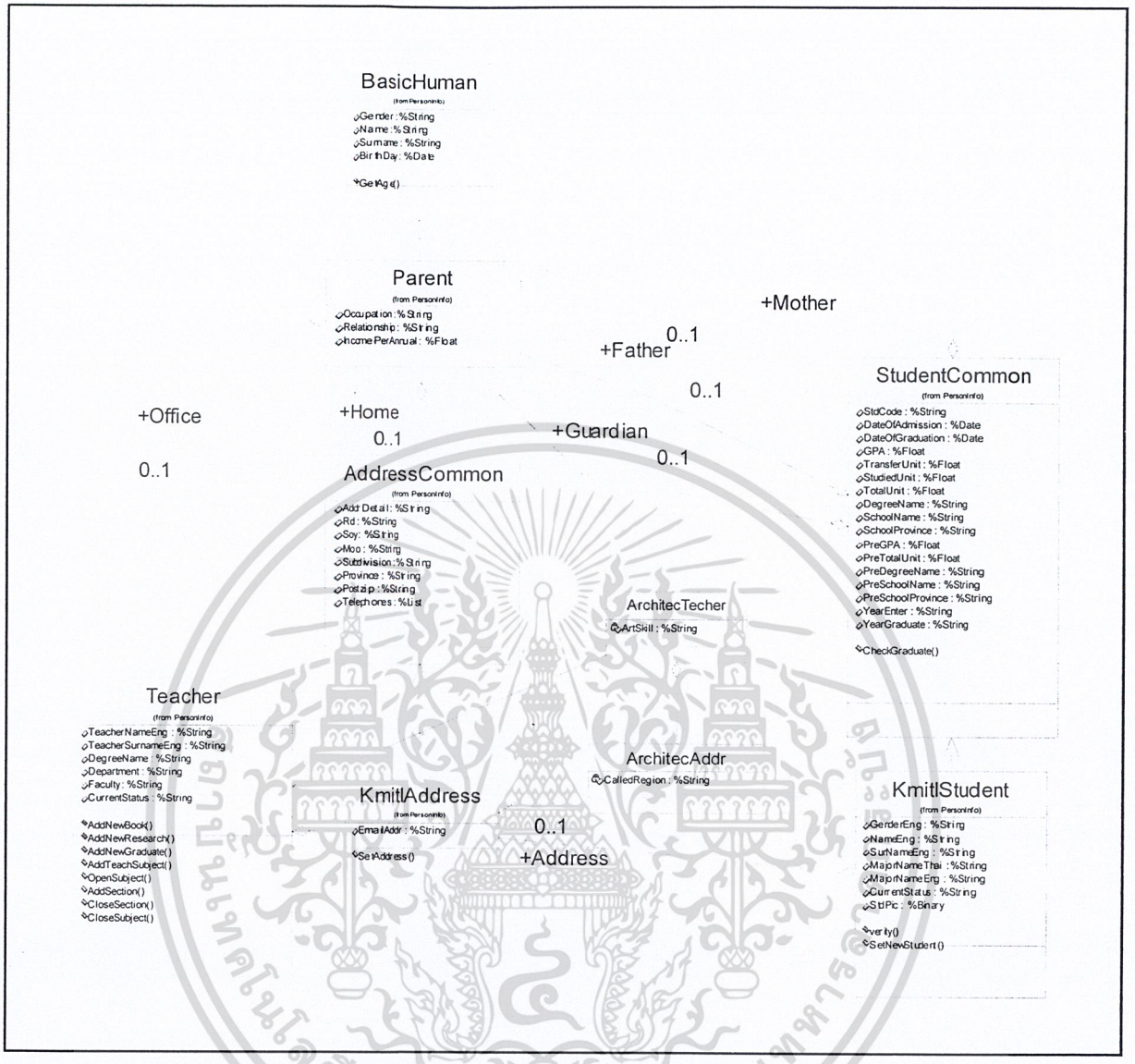
รูปที่ 3-21 แสดงยูสเคซของระบบลงทะเบียนนักศึกษา



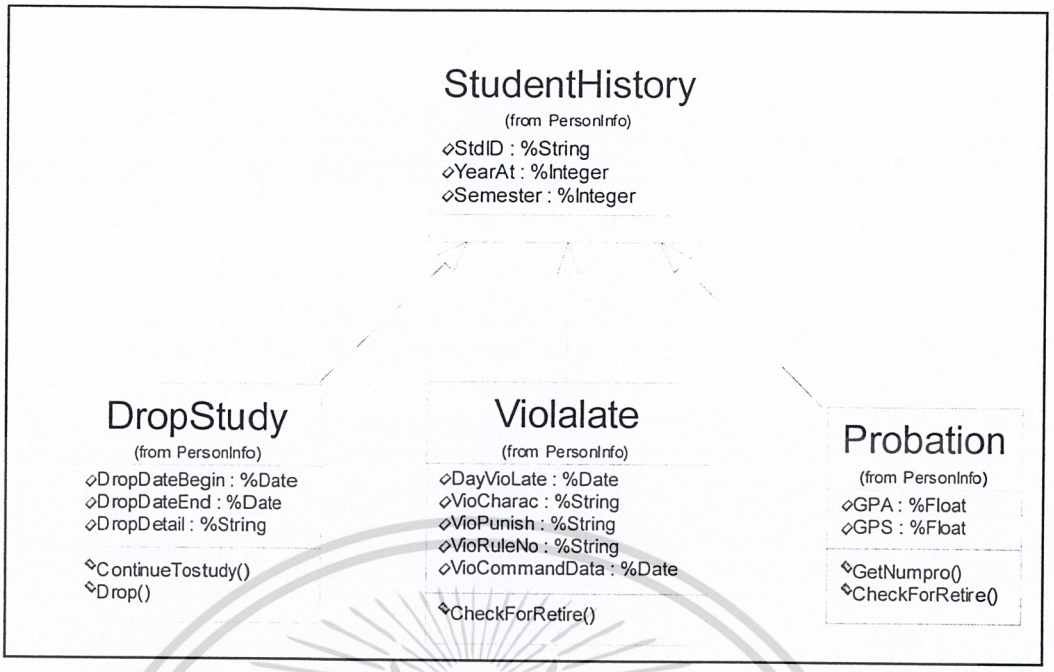
รูปที่3-22 แสดง Logical View Package โดยรวมของระบบ



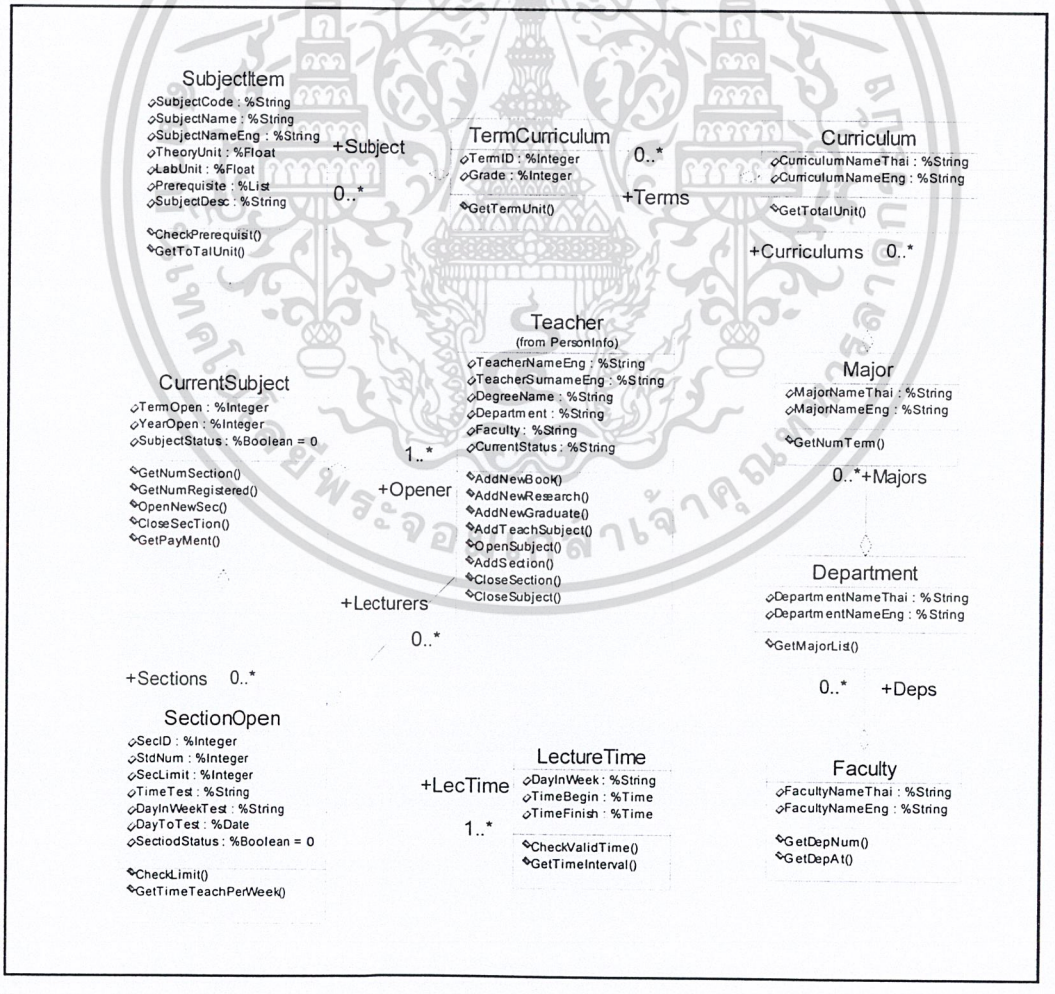
รูปที่ 3-23 แสดง คลาสไดอะแกรม ภายใน Package PersonInfo



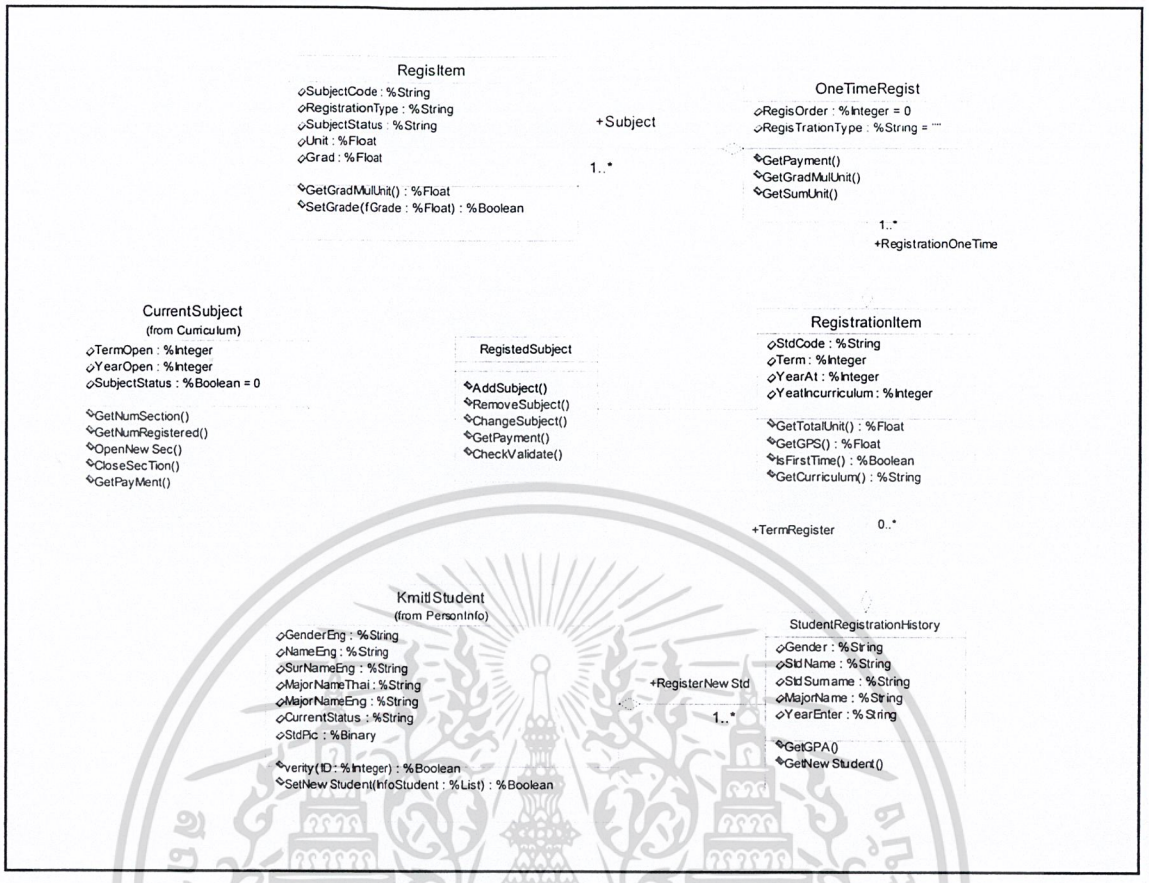
รูปที่3-24 แสดง คลาสไดอะแกรม ภายใน PackageStudentCommon



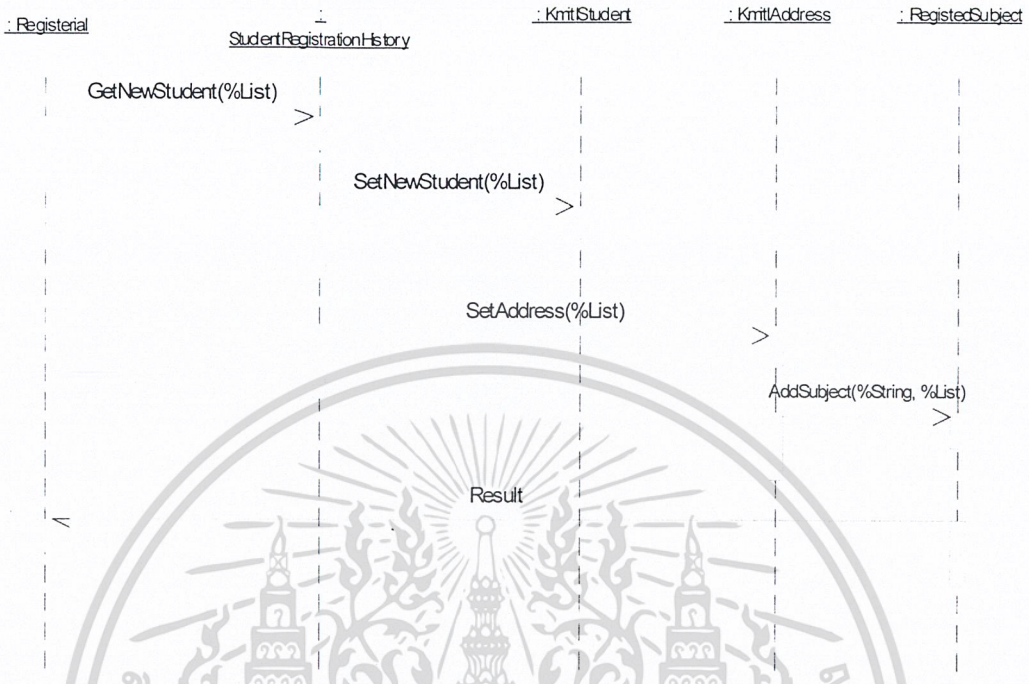
รูปที่ 3-25 แสดงคลาสไดอะแกรม Package StudentHistory



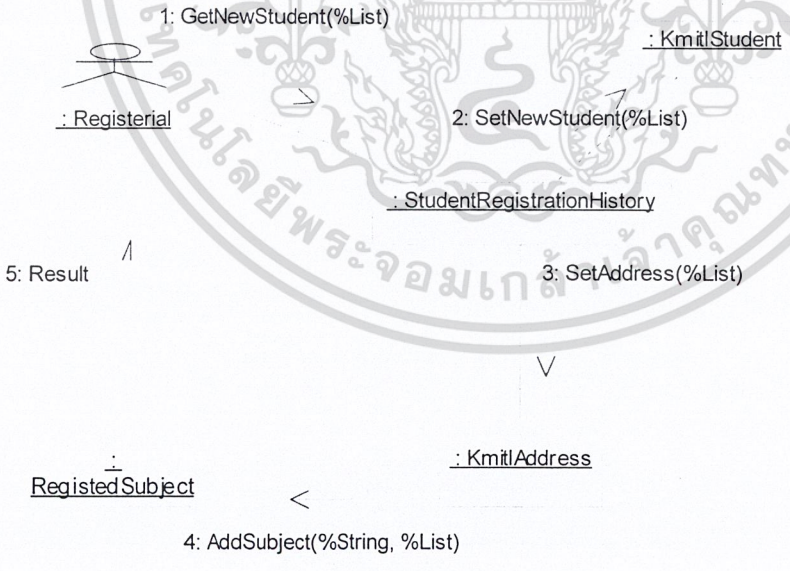
รูปที่ 3-26 แสดงคลาสไดอะแกรมของคลาส Curriculum



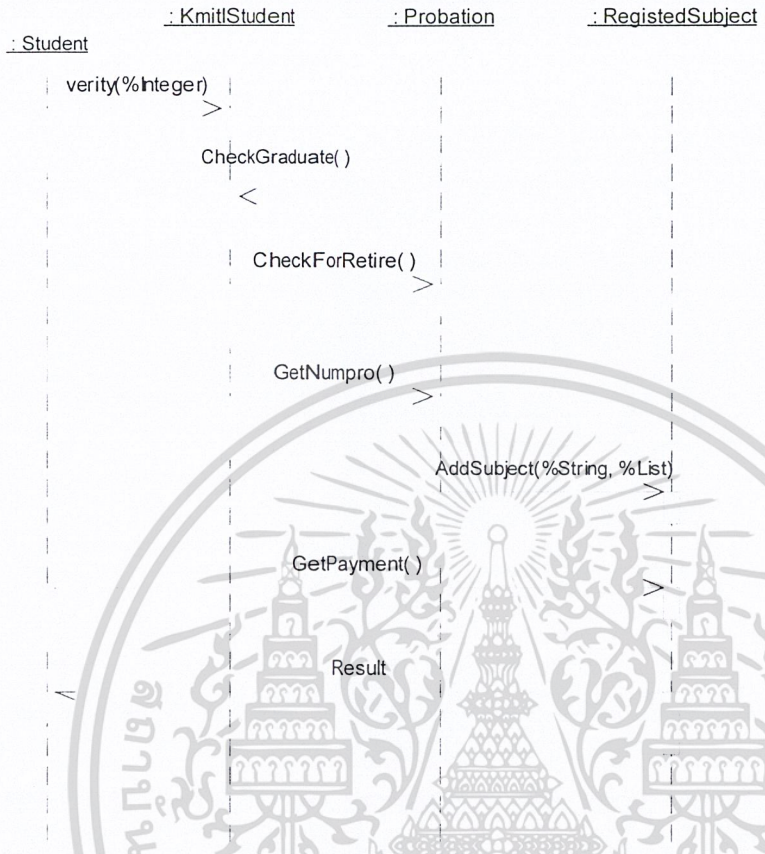
รูปที่ 3-27 แสดงคลาสไดอะแกรมของคลาส Registration



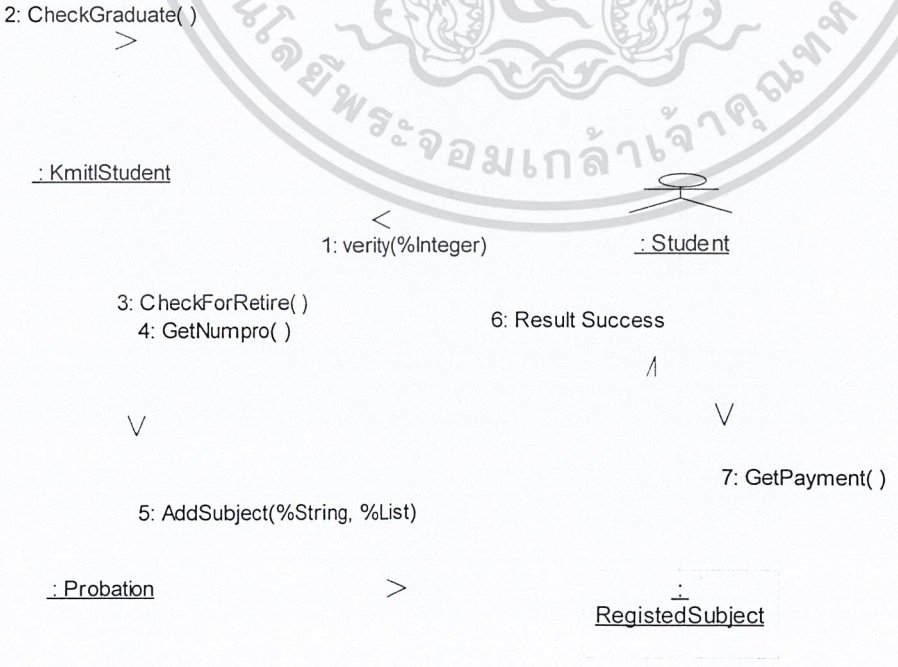
รูปที่ 3-28 แสดงซีเคว้นไดอะแกรม ของ RegistrationSubject



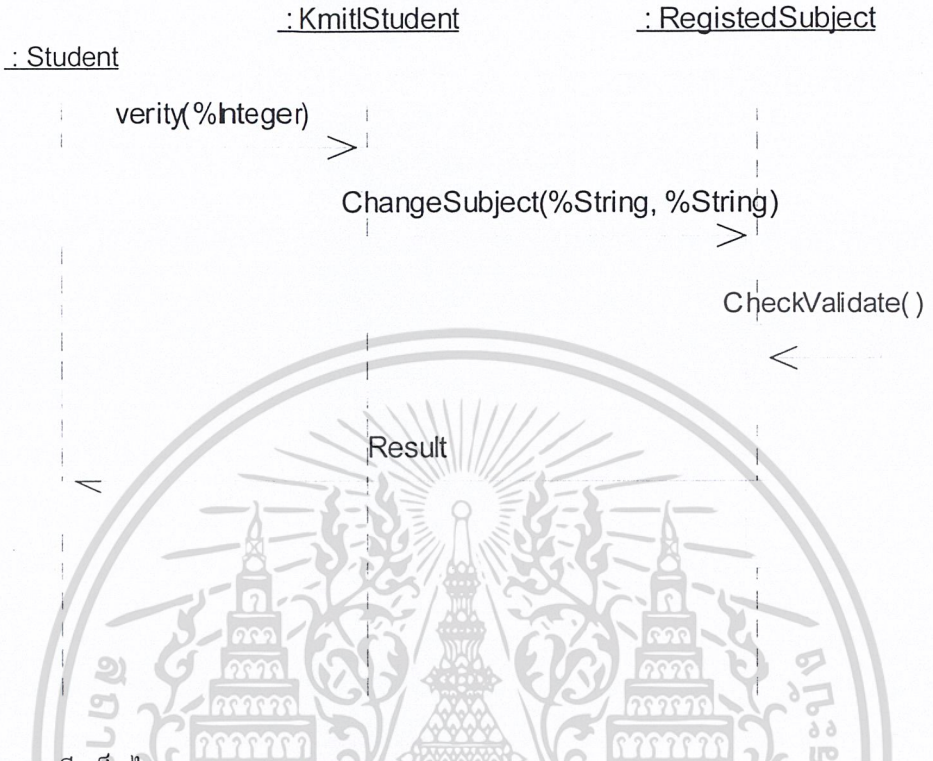
รูปที่ 3-29 แสดง ค็อลแลโบเรชั่นไดอะแกรม ของ RegistrationSubject



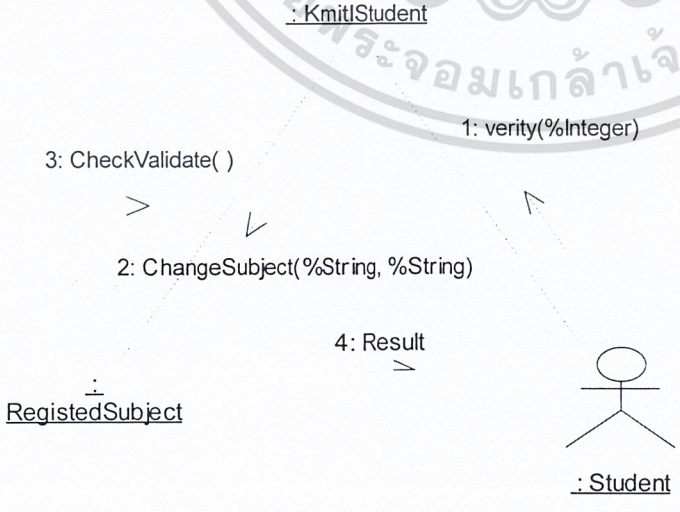
รูปที่ 3-30 แสดง ซีเคว็นไดอะแกรมของการ AddSubject



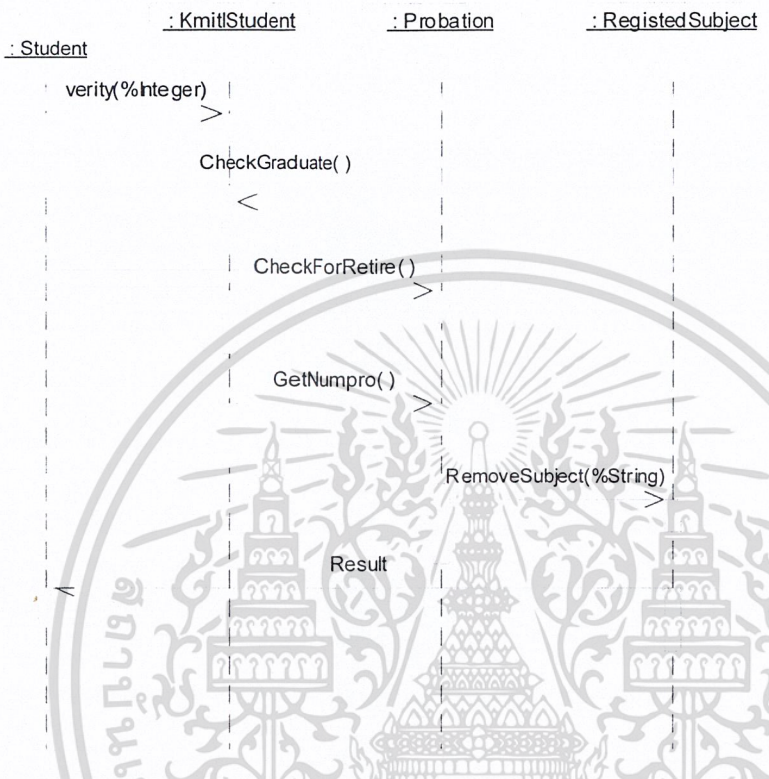
รูปที่ 3-31 แสดง ค็อลแลโบเรชั่นไดอะแกรม AddSubject



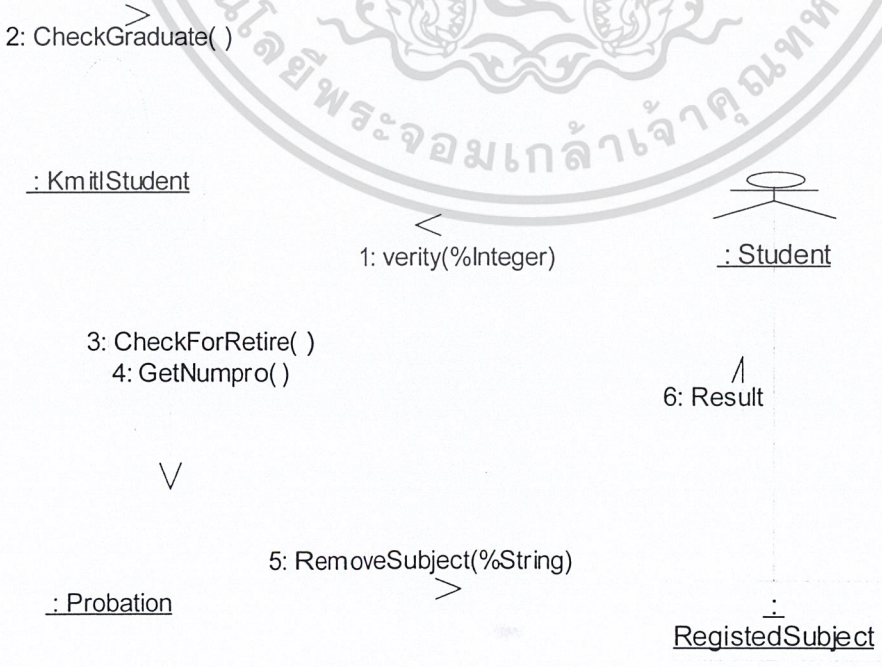
รูปที่ 3-32 แสดง ซีเควันไดอะแกรมของ ChangeSubject



รูปที่ 3-33 แสดง คือลแลโบเรชั่นไดอะแกรมของ ChangeSubject

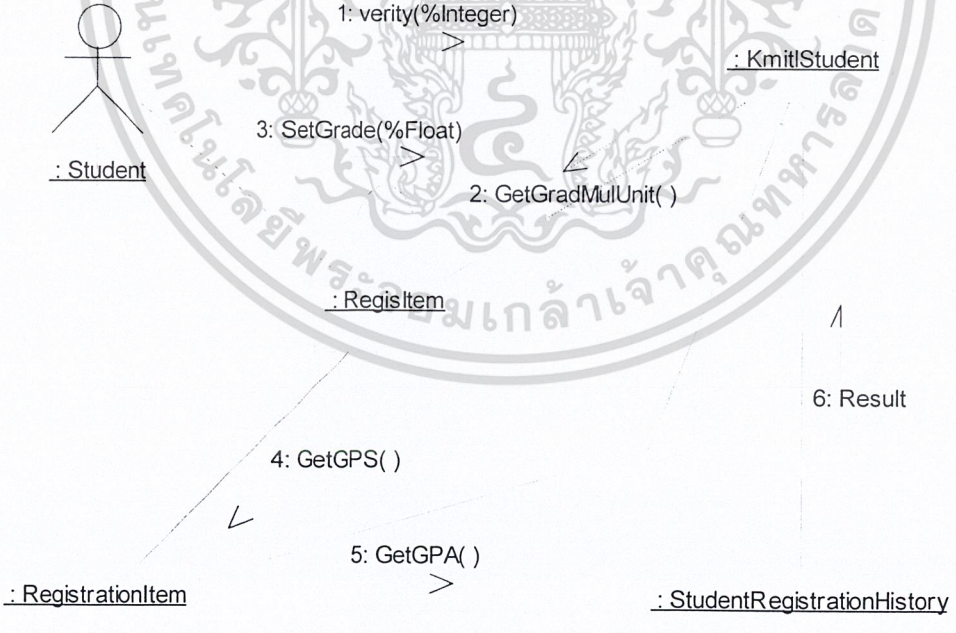


รูปที่ 3-34 แสดง ซีเควิ่นไดอะแกรม RemoveSubject

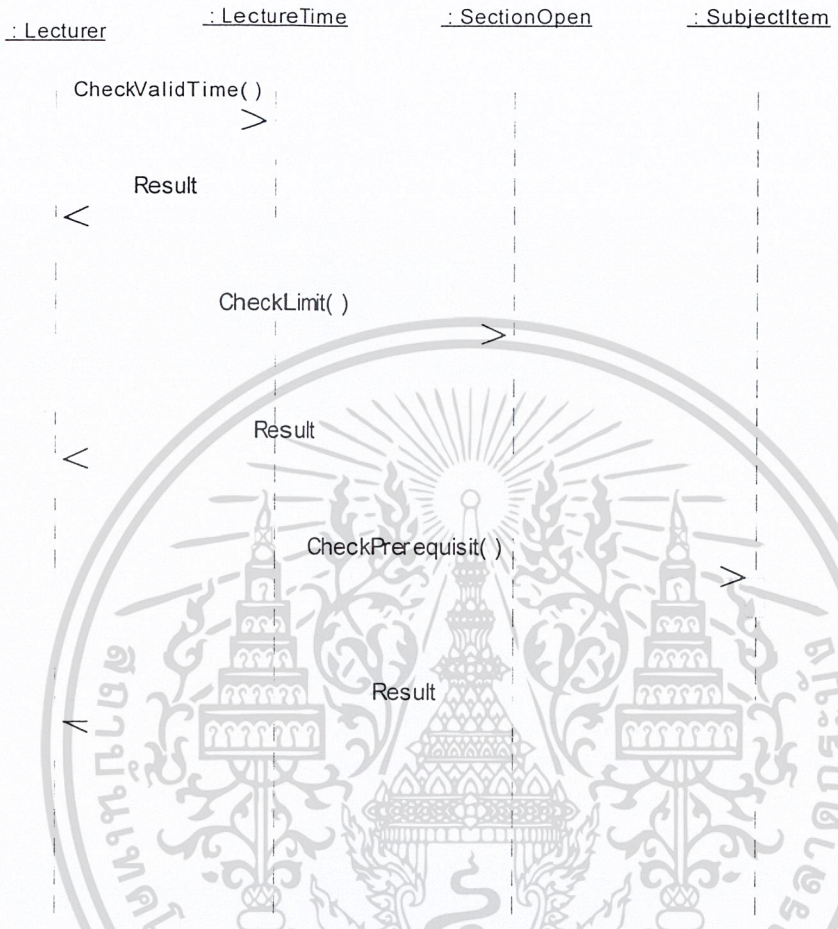


รูปที่ 3-35 แสดง ค็อลแลโบเรชั่นไดอะแกรมของ RemoveSubject

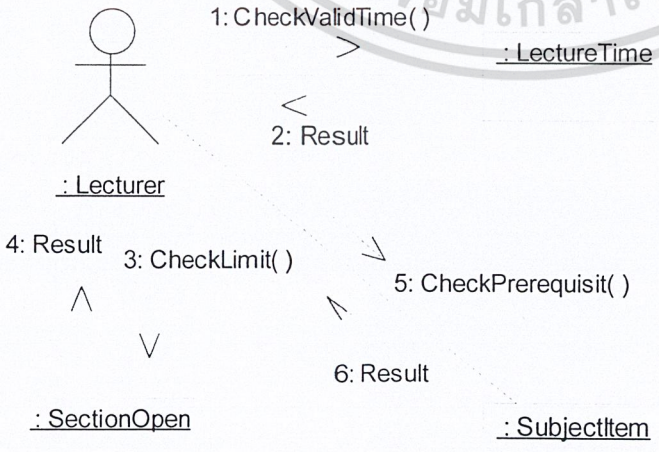
รูปที่ 3-36 แสดง ซีเควันไดอะแกรมของ ViewGradeStudent



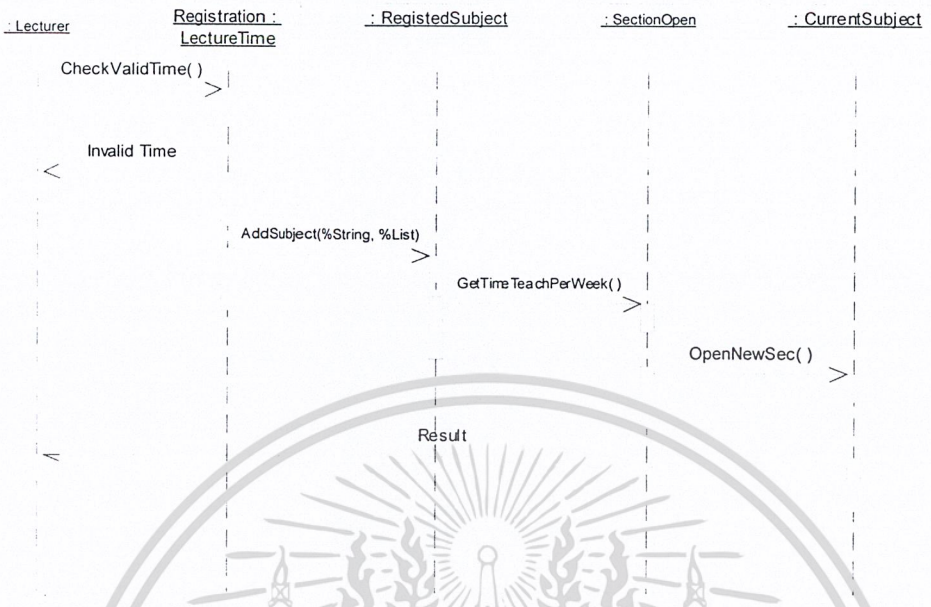
รูปที่ 3-37 แสดง ค็อลแลโบเรชั่นไดอะแกรมของ ViewGradeStudent



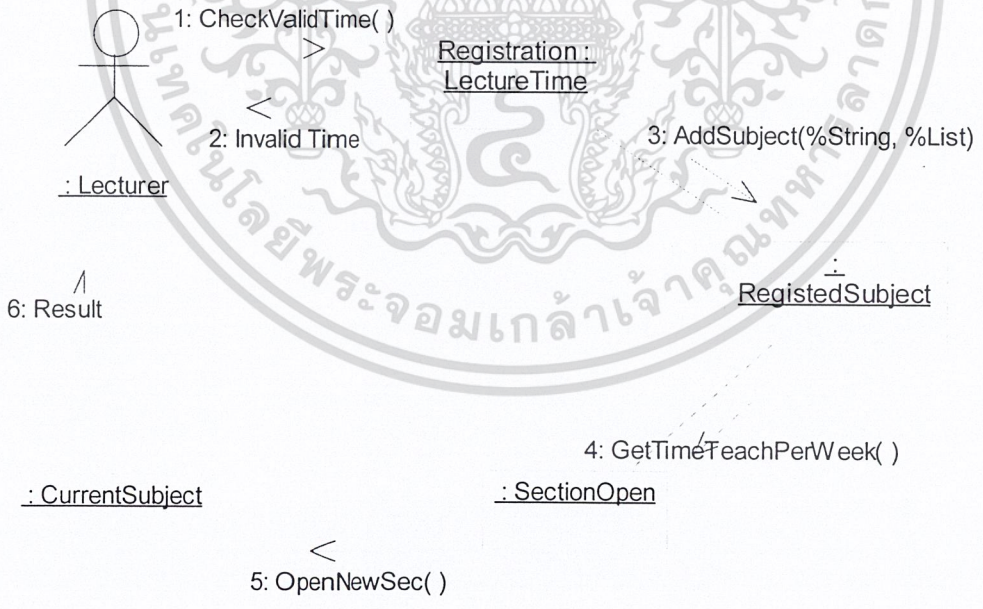
รูปที่ 3-38 แสดง ซีเควันไดอะแกรมของ CheckStatusSection



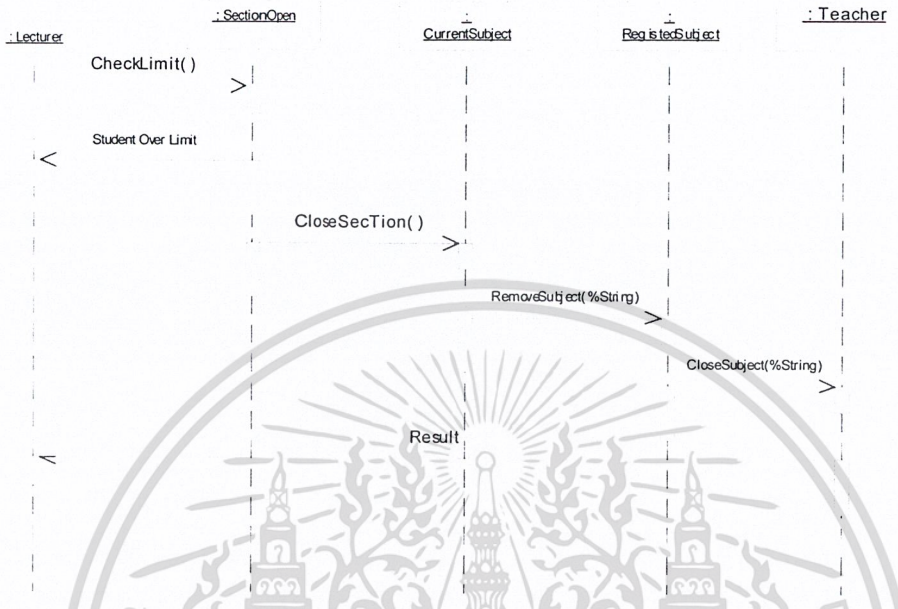
รูปที่ 3-39 แสดง คอลแลโบเรชั่นไดอะแกรม CheckStatusSection



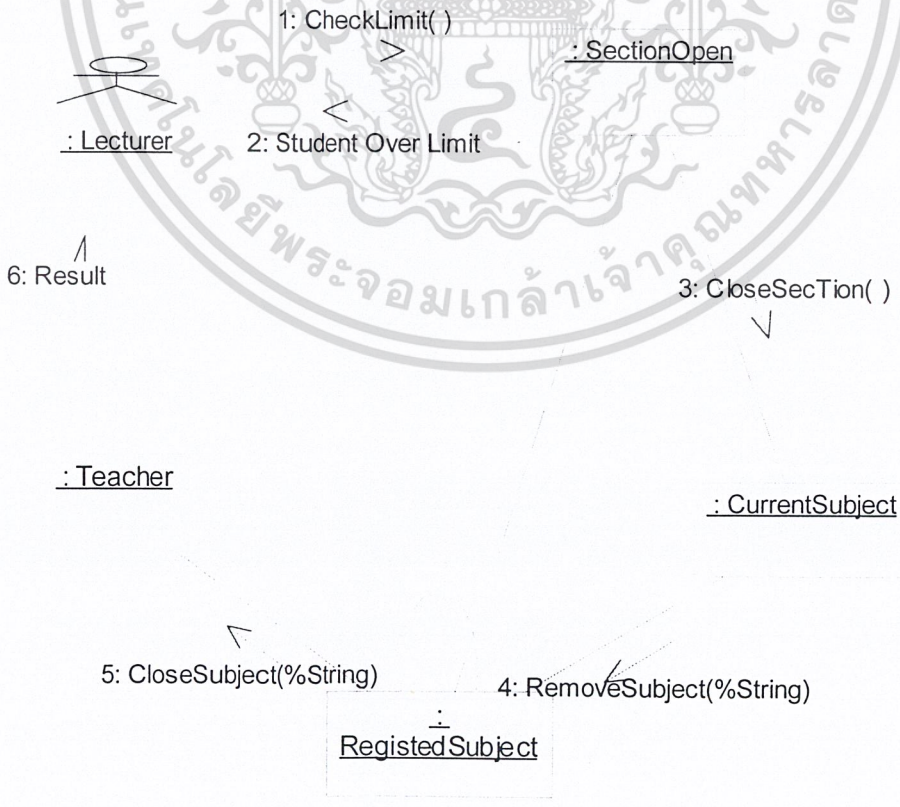
รูปที่ 3-40 แสดง ซีเควันไดอะแกรมของ OpenSection



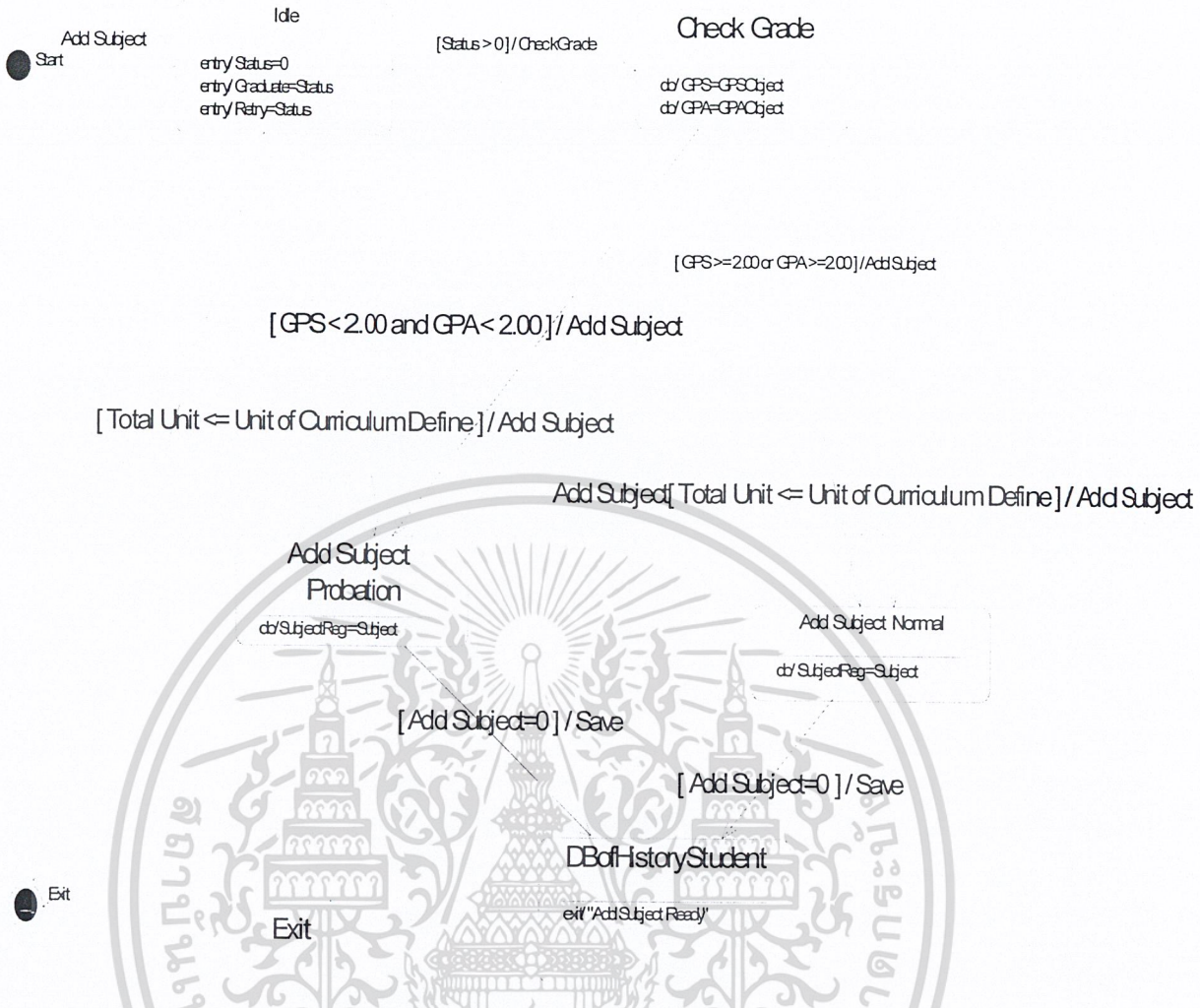
รูปที่ 3-41 แสดง คือลแลโบเรชั่นไดอะแกรมของ OpenSection



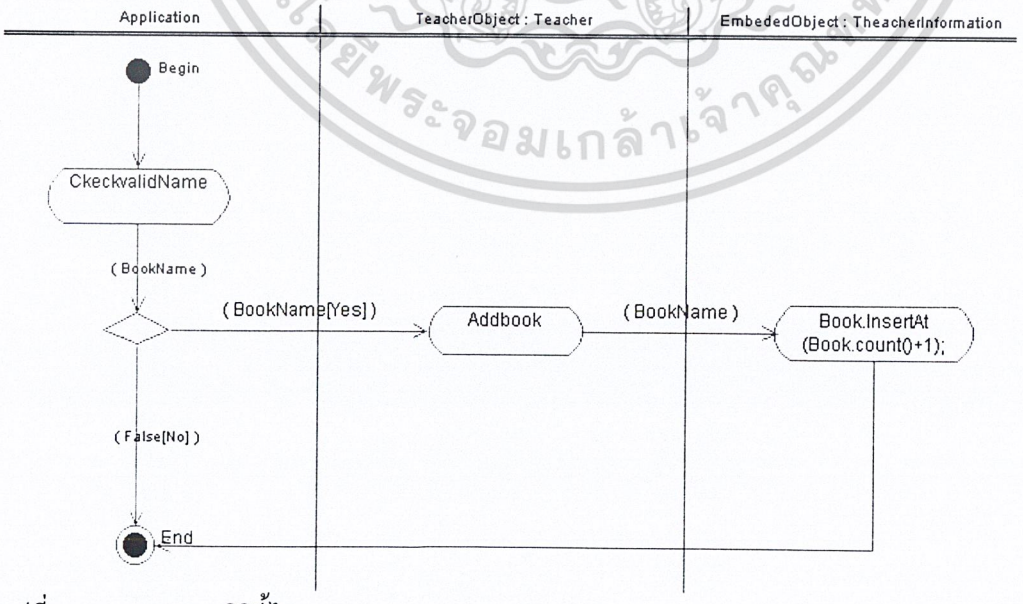
รูปที่ 3-42 แสดง ซีเควิ่นไดอะแกรมของ CloseSection



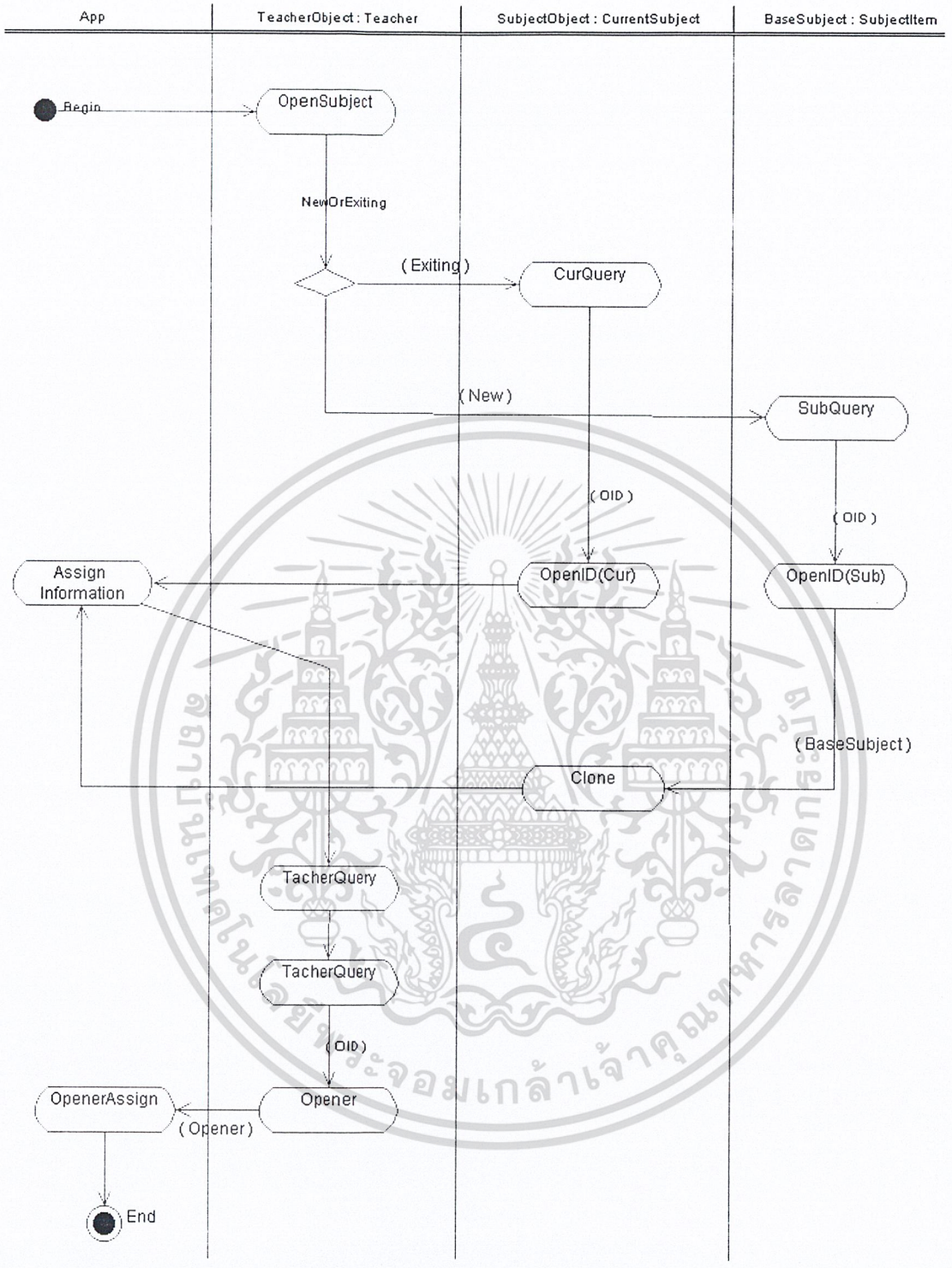
รูปที่ 3-43 แสดง ค็อลแลโบเรชั่นไดอะแกรมของ CloseSection



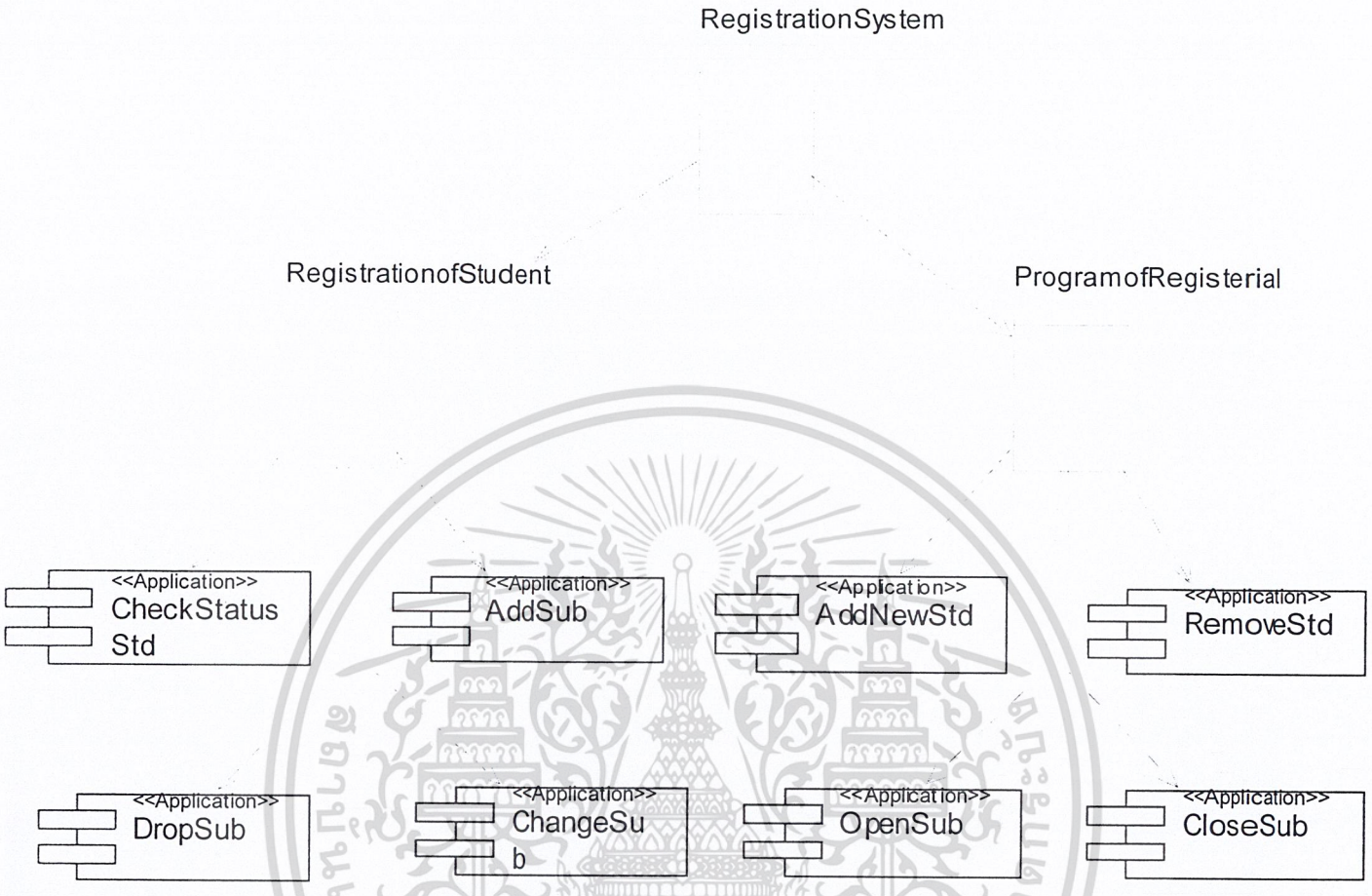
รูปที่ 3-44 แสดง State Diagram ของการลงทะเบียนโดยมีการตรวจสอบเกรด



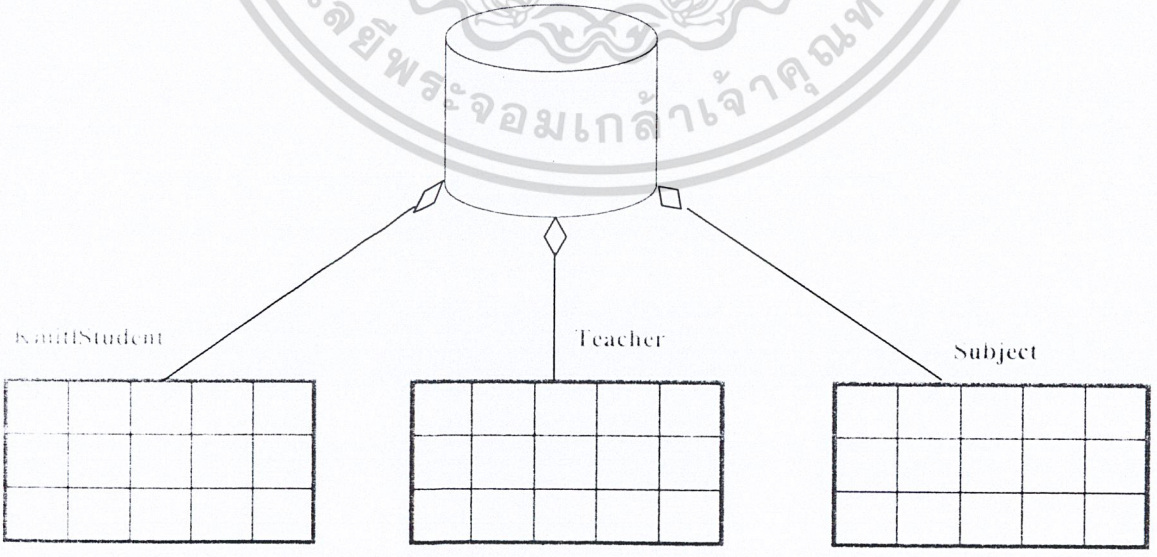
รูปที่ 3-45 แสดง แอคติวิตี้ไดอะแกรม ของการ AddNewBook



รูปที่ 3-46 แสดง แอคติวิตี้ไดอะแกรม ของการ OpenSubject



รูปที่ 3-47 แสดง Presentation Logic Subsystem



รูปที่ 3-48 แสดง Database Logic Subsystem

รูปที่ 3-49 แสดงดีพลอยเม้นไดอะแกรมของระบบ

การออกแบบฐานข้อมูลด้วยเครื่องมือของ Rational Rose ออกมาในรูปแบบของ UML ซึ่งสามารถที่จะเปลี่ยนจากรูปแบบของ UML ออกมาเป็นการเก็บในฐานข้อมูลได้ดังต่อไปนี้ กรณีศึกษาของระบบลงทะเบียนนักศึกษา แสดง Logical View Report Unified Modeling Language Syntaxซึ่งประกอบด้วย Public Attributes And Operations

LOGICAL VIEW REPORT

Logical View

Cache Data Types

%Binary
%Boolean
%CacheString
%Currency
%Date
%DynamicGWQuery
%DynamicObjectQuery
%DynamicSQLQuery
%ExactString
%ExtentSQLQuery
%Float
%Integer
%List
%Name
%Numeric
%ObjectHandle
%ObjectIdentity
%Query
%SQLQuery
%SmallInt
%Status
%Stream
%String
%StringTimeStamp
%Time
%TimeStamp
%TinyInt

Curriculum

CurrentSubject

Derived from SubjectItem

Public Attributes:

TermOpen : %Integer
YearOpen : %Integer
SubjectStatus : %Boolean = 0

Public Operations:

GetNumSection () : %Integer
GetNumRegistered () : %Integer
OpenNewSec () : %Integer
CloseSecTion () : %Integer
GetPayMent () : %Float

Curriculum

Public Attributes:

CurriculumNameThai : %String
CurriculumNameEng : %String

Public Operations:

GetTotalUnit () : %Float

Department

Public Attributes:

DepartmentNameThai : %String
DepartmentNameEng : %String

Public Operations:

GetMajorList () : %List

Faculty

Public Attributes:

FacultyNameThai : %String
FacultyNameEng : %String

Public Operations:

GetDepNum () : %Integer
GetDepAt () : %String

LectureTime

Public Attributes:

DayInWeek : %String
TimeBegin : %Time
TimeFinish : %Time

Public Operations:

CheckValidTime () : %Boolean
GetTimeInterval () : %Time

Major

Public Attributes:

MajorNameThai : %String
MajorNameEng : %String

Public Operations:

GetNumTerm () : %Integer

SectionOpen

Public Attributes:

SecID : %Integer
StdNum : %Integer
SecLimit : %Integer
TimeTest : %String
DayInWeekTest : %String
DayToTest : %Date
SectiodStatus : %Boolean = 0

Public Operations:

CheckLimit () : %Boolean
GetTimeTeachPerWeek () : %Time

SubjectItem

Public Attributes:

SubjectCode : %String
SubjectName : %String
SubjectNameEng : %String
TheoryUnit : %Float
LabUnit : %Float
Prerequisite : %List
SubjectDesc : %String

Public Operations:

CheckPrerequisit () : %Boolean
GetToTalUnit () : %Float

TermCurriculum

Public Attributes:

TermID : %Integer
Grade : %Integer

Public Operations:

GetTermUnit () : %Float

PersonInfo

AddressCommon

Public Attributes:

AddrDetail : %String
Rd : %String
Soy : %String
Moo : %String
Subdivision : %String
Province : %String
Postzip : %String
Telephones : %List

ApplicationLevel

BasicHuman

Public Attributes:

Gender : %String
Name : %String
Surname : %String
BirthDay : %Date

Public Operations:

GetAge () : %Integer

DropStudy

Derived from StudentHistory

Public Attributes:

DropDateBegin : %Date
DropDateEnd : %Date
DropDetail : %String

Public Operations:

ContinueTostudy () : %Boolean
Drop () : %Boolean

EmploymentHistory

Public Attributes:

EmpPosition : %String
BeginDayToWork : %Date
LastDayTowork : %Date
Salary : %Float
JobDescription : %String

Public Operations:

GetCurrentDate () : %Date

Empployee

Public Attributes:

StartPosition : %String
FillDate : %Date
StartDate : %Date
BeginSalary : %Float

Public Operations:

GetTimeEmp () : %Date
GetCurrentPosition () : %String
GetCurrentSalary () : %Float
UpdateSalary (fNewSalary : %Float = 0) : %Float

KmitlAddress

Derived from AddressCommon

Public Attributes:

EmailAddr : %String

Public Operations:

SetAddress (Address : %List) :

KmitlStudent

Derived from StudentCommon

Public Attributes:

GenderEng : %String
NameEng : %String
SurNameEng : %String
MajorNameThai : %String
MajorNameEng : %String
CurrentStatus : %String
StdPic : %Binary

Public Operations:

verity (ID : %Integer) : %Boolean
SetNewStudent (InfoStudent : %List) : %Boolean

Parent

Derived from BasicHuman

Public Attributes:

Occupation : %String
Relationship : %String
IncomePerAnnual : %Float

Probation

Derived from StudentHistory

Public Attributes:

GPA : %Float
GPS : %Float

Public Operations:

GetNumpro () : %Integer
CheckForRetire () : %Boolean

StudentCommon

Public Attributes:

StdCode : %String
DateOfAdmission : %Date
DateOfGraduation : %Date
GPA : %Float
TransferUnit : %Float
StudiedUnit : %Float
TotalUnit : %Float
DegreeName : %String
SchoolName : %String
SchoolProvince : %String
PreGPA : %Float
PreTotalUnit : %Float
PreDegreeName : %String
PreSchoolName : %String
PreSchoolProvince : %String
YearEnter : %String
YearGraduate : %String

Public Operations:

CheckGraduate () : %Boolean

StudentHistory

Public Attributes:

StdID : %String
YearAt : %Integer
Semester : %Integer

Teacher

Derived from Empployee

Public Attributes:

TeacherNameEng : %String
TeacherSurnameEng : %String
DegreeName : %String
Department : %String
Faculty : %String
CurrentStatus : %String

Public Operations:

AddNewBook (BookName : %String) : %Boolean
AddNewResearch (ResearchName : %String) : %Boolean
AddNewGraduate (SchoolName : %String) : %Boolean
AddTeachSubject (SubjectCode : %String) : %Boolean
OpenSubject (SubjectCode : %String) : %Boolean
AddSection (SubjectCode : %String) : %Boolean
CloseSection (SubjectCode : %String, SecID : %Integer) : %Boolean
CloseSubject (SubjectCode : %String) : %Boolean

TheacherInformation

Public Attributes:

GraduateList : %List
Book : %List
Research : %List
Respon : %List

Violalate

Derived from StudentHistory

Public Attributes:

DayVioLate : %Date
VioCharac : %String
VioPunish : %String
VioRuleNo : %String
VioCommandData : %Date

Public Operations:

CheckForRetire () : %Boolean

PackageStudentCommon

ArchitecAddr

Derived from KmitlAddress

ArchitecTecher

Derived from Teacher

PackageStudentHistory

Registration

NewClass

OneTimeRegist

Public Attributes:

RegisOrder : %Integer = 0
RegisTrationType : %String = ""

Public Operations:

GetPayment () : %Float
GetGradMulUnit () : %Float
GetSumUnit () : %Float

RegisItem

Public Attributes:

SubjectCode : %String
RegistrationType : %String
SubjectStatus : %String
Unit : %Float
Grad : %Float

Public Operations:

GetGradMulUnit () : %Float
SetGrade (fGrade : %Float) : %Boolean

RegistedSubject

Public Operations:

AddSubject (SubjectName : %String, FormSubjects : %List) : %Boolean
RemoveSubject (SubjectCode : %String) : %Boolean
ChangeSubject (OldSubject : %String, NewSubject : %String) : %Boolean
GetPayment () : %Float
CheckValidate () : %Boolean

RegistrationItem

Public Attributes:

StdCode : %String
Term : %Integer
YearAt : %Integer
YeatIncurriculum : %Integer

Public Operations:

GetTotalUnit () : %Float
GetGPS () : %Float
IsFirstTime () : %Boolean
GetCurriculum () : %String

StudentRegistrationHistory

Public Attributes:

Gender : %String
StdName : %String
StdSurname : %String
MajorName : %String
YearEnter : %String

Public Operations:

GetGPA () : %Float
GetNewStudent (InfoStudent : %List) : %Boolean

รวมทั้งหมด

8 Logical Packages

66 Classes

LOGICAL PACKAGE STRUCTURE

Logical View

PersonInfo

PackageStudentHistory

PackageStudentCommon

Cache Data Types

Curriculum

Registration

บทที่ 4

ศึกษาและวิเคราะห์กระบวนการการออกแบบและออกแบบระบบโดยใช้ OONIAM

4.1 OODB แนวความคิดเชิงวัตถุ

รายละเอียดหลักการต่างๆ แนวคิดเชิงวัตถุมีดังนี้

4.1.1 ออบเจ็กต์ (Object)

ออบเจ็กต์ (Object) คือ ออบเจ็กต์คือ Abstract Representation ของ Real World Entity ที่มี

- Unique Identity : ออบเจ็กต์ต้องมีสิ่งที่ระบุถึงความแตกต่างจากออบเจ็กต์อื่น
- Embedded Properties
- ความสามารถในการติดต่อสื่อสารกับออบเจ็กต์อื่นและตัวเอง

4.1.2 Object Identity (OID)

Object Identity หรือ Object Identifier (OID) เป็นสิ่งที่ใช้ระบุหรืออ้างอิงออบเจ็กต์ ซึ่งแต่ละออบเจ็กต์จะมี OID ไม่ซ้ำกัน OID จะถูกกำหนดให้โดยระบบตั้งแต่เมื่อออบเจ็กต์ถูกสร้างขึ้นและจะไม่สามารถเปลี่ยนแปลงได้

OID จะแตกต่างจาก Primary (PK) ของระบบที่ใช้ข้อมูลเชิงสัมพันธ์ เพราะ PK จะขึ้นอยู่กับค่าของแอททริบิวท์ (Attribute) ที่ผู้ใช้กำหนด (User-Given Value) ซึ่งจะสามารถเปลี่ยนแปลง ในภายหลังได้ แต่ OID ถูกกำหนดโดยระบบซึ่งจะไม่ขึ้นกับค่าแอททริบิวท์ของออบเจ็กต์และเปลี่ยนแปลงไม่ได้ โดยเมื่อออบเจ็กต์ถูกลบออกไปจากระบบ OID ของออบเจ็กต์นั้นก็จะถูกลบตามไปด้วยและจะไม่มีการนำ OID ที่ถูกลบไปแล้วกลับมาใช้ใหม่ ค่าของ OID จะไม่ผูกติดกับตำแหน่งทางกายภาพ (Physical Address) ของหน่วยความจำ (Permanent Memory) ทำให้ระบบเชิงวัตถุมีความเป็นอิสระของข้อมูลทางกายภาพ (Physical Data Independence)

4.1.3 แอททริบิวท์ (Attributes หรือ Instance Variables)

แอททริบิวท์ (Attribute) หรือ Instance Variable ก็คือ ข้อมูลของออบเจ็กต์ อาจเป็นได้ทั้งข้อมูลชนิดพื้นฐานหรือเป็นออบเจ็กต์ก็ได้

4.1.4 สถานะของออบเจ็กต์ (Object State)

สถานะของออบเจ็กต์ขึ้นกับค่าของแอททริบิวท์ของออบเจ็กต์ ณ เวลาที่กำหนด ถึงแม้ว่าสถานะของออบเจ็กต์จะเปลี่ยนแปลงไปแต่ OID ยังคงเดิม ถ้าต้องการเปลี่ยนสถานะของออบเจ็กต์จะต้องเปลี่ยนค่าแอททริบิวต์โดยจะต้องส่งเมสเซจไปยังออบเจ็กต์ แล้วแมสเซจที่ส่งไปนี้จะเรียกเมธอดที่เกี่ยวข้องให้ทำงาน

4.1.5 เมสเซจและเมธอด (Message and Method)

ตามคุณสมบัติของเอนแคปซูเลชั่นการที่จะกระทำการใดๆ กับออบเจ็กต์ต้องกระทำผ่านเมธอดเท่านั้น เมธอดจะถูกใช้เพื่อเรียกดูหรือเปลี่ยนค่าแอททริบิวท์ของออบเจ็กต์ ในการเรียกใช้เมธอดนั้นจะต้องส่งเมสเซจไปยังออบเจ็กต์ เมสเซจที่ส่งจะต้องระบุออบเจ็กต์ที่รับเมสเซจชื่อเมธอดและ พารามิเตอร์ที่เกี่ยวข้อง

4.1.6 เอนแคปซูเลชัน (Encapsulation)

ออบเจ็กต์จะเป็นการรวมเอาแอททริบิวท์และเมธอดไว้ด้วยกัน ซึ่วการจะเข้าถึงแอททริบิวท์ต้องทำผ่านเมธอดเท่านั่น ซึ่งจะช่วยปกป้องข้อมูลของออบเจ็กต์ได้

4.1.7 คลาส (Class)

ในระบบเชิงวัตถุจะนำออบเจ็กต์ที่มีคุณสมบัติเหมือนกันเข้าไว้ด้วยกันเป็นคลาส หรือคลาสก็คือ Collection ของออบเจ็กต์ที่มีคุณสมบัติเหมือนกันและใช้แอททริบิวท์และเมธอดร่วมกัน

4.1.8 การสืบทอดคุณสมบัติ (Inheritance)

การสืบทอดคุณสมบัติเป็นการสร้างคลาสใหม่ (Subclass) โดยสืบทอดคุณสมบัติ (ทั้งแอททริบิวท์ และเมธอด) จากคลาสที่มีอยู่แล้ว (Superclass) นอกจากนี้ยังสามารถเพิ่มรายละเอียดให้ Subclass ได้อีกด้วย การสืบทอดคุณสมบัตินี้มีประโยชน์ในเรื่องการนำกลับมาใช้ใหม่ โดยแบ่งได้หลักๆ คือ

- Single Inheritance เป็นการสืบทอดคุณสมบัติมาจากคลาสเดียว
- Multiple Inheritance เป็นการสืบทอดคุณสมบัติจากหลายๆ คลาส

4.1.9 Method Overriding and Polymorphism

Subclass สามารถสร้างเมธอดที่ชื่อซ้ำกับเมธอดที่สืบทอดมาจาก Superclass ได้เพื่อทำให้ Superclass นั้นมีความเฉพาะเจาะจงมากขึ้น เมธอดที่ Subclass สร้างขึ้นนี้จะถูกเรียกใช้แทน (Override) เมธอดของ Superclass

พอลิมอร์ฟีซึม (Polymorphism) เป็นการทำให้ออบเจ็กต์ที่ต่างกันสามารถตอบสนองต่อเมสเซจเดียวกันได้ในหลายๆ วิธีการ พอลิมอร์ฟีซึมเป็นคุณสมบัติที่สำคัญของระบบเชิงวัตถุเพราะช่วยให้แต่ละออบเจ็กต์มีความเฉพาะ เจาะจงมากขึ้น โดยในระบบเชิงวัตถุพอลิมอร์ฟีซึมหมายถึง

- เมธอดสามารถใช้ชื่อเดียวกันได้ในหลายๆ คลาส
- ผู้ใช้ส่งเมสเซจเดียวกันไปยังออบเจ็กต์จากคลาสต่างๆ กัน ก็ยังคงให้ผลลัพธ์ที่ถูกต้อง

4.1.10 Abstract Data Type

Abstract Data Type (ADT) เป็นคุณสมบัติที่ใช้สร้างชนิดของข้อมูลชนิดใหม่ขึ้นมา โดยกำหนดโครงสร้างข้อมูลและโอเปอร์ชันที่ใช้จัดการข้อมูลขึ้นมาจากข้อมูลพื้นฐาน

โดย Abstract Data Type และคลาสมีความหมายใกล้เคียงกัน แต่ในระบบเชิงวัตถุสองคำนี้จะมีความหมายที่แตกต่างกัน โดย Type จะหมายถึงโครงสร้างข้อมูลและเมธอดของคลาส และคลาสหมายถึง Collection ของ Object Instance เมื่อกำหนดคลาสใหม่นี้ขึ้นมาก็จะเป็นการกำหนด Type ใหม่ด้วย Type ที่กำหนดขึ้นจะถูกใช้เป็นต้นแบบในการสร้างออบเจ็กต์ใหม่ขึ้นมา ซึ่งจะถูกจัดการโดยคลาสขณะ run-time

โดยคุณสมบัติของ ADT และ Inheritance จะสนับสนุนให้มี Complex Object โดย Complex Object ถูกสร้างขึ้นโดยนำออบเจ็กต์อื่นเข้ามารวมไว้ด้วยกัน ในรูปของ Set ของ Complex Relation

4.2 คุณลักษณะของ Object-Oriented Data Model (OODM)

OODM อย่างน้อยที่สุดควรมีลักษณะดังนี้

1. สนับสนุนการทำ Complex Object ได้
2. Extensible : ต้องมีความสามารถในการกำหนด Data Type และ Operation ที่เกี่ยวข้องขึ้นมาใหม่ได้
3. สนับสนุน Encapsulation : รูปแบบของข้อมูลและการจัดการของเมธอดต้องถูกซ่อนจากภายนอก
4. ต้องมีคุณสมบัติ Inheritance : ออบเจ็กต์ต้องสามารถสืบทอดคุณสมบัติ (ข้อมูลและเมธอด) จากอีอบเจ็กต์อื่นได้
5. ต้องสนับสนุน Object Identity (OID)

4.3 การออกแบบฐานข้อมูลโดย NIAM และ OONIAM

ปัจจุบันวิธีการออกแบบฐานข้อมูลที่รู้จักกันอย่างแพร่หลายได้แก่นอร์มัลไลเซชั่น (Normalization) ซึ่งเป็นวิธีที่มีแนวคิดในการปรับปรุงคุณสมบัติของรีเลชั่นเป็นขั้นตอน อย่างมีระบบเพื่อลดโอกาสที่จะเกิดความผิดพลาดเนื่องจากการเปลี่ยนแปลงข้อมูล การออกแบบฐานข้อมูลด้วยวิธีนอร์มัลไลเซชั่น เริ่มต้นด้วยการ กำหนดยูนิเวอร์แซลรีเลชั่น พร้อมทั้งกำหนดความสัมพันธ์ระหว่างแอตทริบิว (Attribute) ต่าง ๆ ของยูนิเวอร์แซลรีเลชั่น ในรูปของฟังก์ชันแนลดีเพนเดนซ์(Functional Dependency) มัลติเวลลู ดีเพนเดนซ์ (Multivalued Dependency) และจอยน์ดีเพนเดนซ์ซี (Join Dependency) แล้วทำให้ รีเลชั่นในแต่ละขั้นตอนมีคุณสมบัติตรงตามที่กำหนดไว้ ตั้งแต่ขั้นตอนที่ 1 ถึงขั้นตอนที่ 5 ผลที่ได้ในขั้นตอนสุดท้ายจะได้ Fifth Normal Form (5NF) ถึงแม้ว่ามอร์มัลไลเซชั่นนี้จะเป็นวิธีที่มีขั้นตอนที่เป็นระบบแต่ก็เป็นวิธีที่ค่อนข้างยุ่งยากและซับซ้อน โดยเฉพาะอย่างยิ่งเมื่อระบบงานมีขนาดใหญ่ แอตทริบิวมีจำนวนมาก การออกแบบด้วยวิธีนอร์มัลไลเซชั่นจึงเป็นเรื่องลำบาก

ดังนั้นขอเสนอการออกแบบระบบฐานข้อมูลรวมแบบรีเลชั่นแนล (Integrated Relational Database System) โดยใช้ไนแอม (NIAM : Nijssen 's Information System Analysis Method) เป็นเครื่องมือในการออกแบบ เนื่องจากแนวคิดที่ให้ Conceptual Schema มีพื้นฐานมาจากโครงสร้างภาษาธรรมชาติ ใช้รูปประโยคที่มี ประธาน กริยา กรรม วิธีแสดงรูปแบบความสัมพันธ์ เป็นแบบจำลองที่มีความหมายและมีเครื่องหมายแสดงความสัมพันธ์ของข้อมูล และข้อจำกัดของข้อมูลได้อย่างชัดเจน นอกจากนั้น ยังสามารถแปลง Conceptual Schema และ Relational Database Schema ซึ่งจะอยู่ในรูปของ Fifth Normal Form และเนื่องจากวิธีการนี้ใช้รูปสัญญลักษณ์ที่แสดงความสัมพันธ์ของข้อมูลและง่ายต่อการเข้าใจ ดังนี้จึงสะดวกในการออกแบบระบบฐานข้อมูลของระบบงานใหญ่ ๆ

เนื่องจากเป็นที่ยอมรับกันโดยทั่วไปแล้วว่า ระบบฐานข้อมูลรีเลชั่นแนลเป็นเครื่องมือที่เหมาะสม ในการช่วยพัฒนาระบบสารสนเทศ (Information System) โดยเฉพาะอย่างยิ่งระบบสารสนเทศที่มีจุดประสงค์ ในการสนับสนุนการจัดการ (Management Information System :MIS) ดังนั้นการออกแบบฐานข้อมูลรวมสำหรับองค์กรจึงเป็นสิ่งสำคัญอย่างยิ่ง เพราะฐานข้อมูลที่ได้จะเป็นโครงสร้างข้อมูลหลักสำหรับรองรับระบบสารสนเทศ ดังกล่าว

4.3.1 การออกแบบฐานข้อมูลด้วยวิธีไนแอม

step 1 : กำหนดขอบเขตของงาน (Universe of Discourse :UoD) และความจริงที่เกิดขึ้นภายในของเขตของงานที่กำหนดไว้

step 2 : วาด Conceptual Schema Diagram โดยคร่าว ๆ จากความจริงในขอบเขตของงาน

step 3 : จัดรูปของ Schema ให้เป็นระเบียบและหาชนิดความจริงที่ได้รับข้อมูล มาจากชนิดความจริงอื่น

step 4 : เติมสัญลักษณ์แสดง Uniqueness constraints

step 5 : ตรวจสอบหาความถูกต้องของชนิดความจริง

step 6 : เติมสัญลักษณ์แสดง Lexical,Mandatory ,Role,Subtype constraints

step 7 : ตรวจสอบ Unique Identifier ของแต่ละชนิดเอนติตี้

step 8 : เติมสัญลักษณ์แสดง Equality,Exclusion,Subset constraints

step 9 : ตรวจสอบความสมบูรณ์ของ Conceptual Schema ว่าต้องสอดคล้องกับตัวอย่างข้อมูลและไม่มีความซ้ำซ้อนของข้อมูล

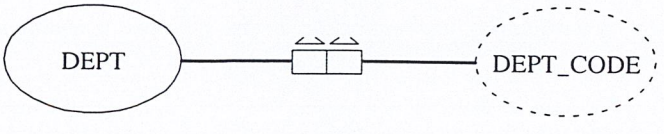
ส่วนประกอบพื้นฐานของไนแอม

- ชนิดเอนติตี้ (Entity Type)
- ชนิดเลเบิล (Lable Type)
- ชนิดความจริง (Fact Type)
- ชนิดอ้างอิง (Reference Type)
- ข้อจำกัดเพื่อความถูกต้องของข้อมูล (Integrity Constraints)

สัญลักษณ์และตัวอย่างการใช้ส่วนประกอบพื้นฐานของแบบจำลองไนแอมแสดงไว้

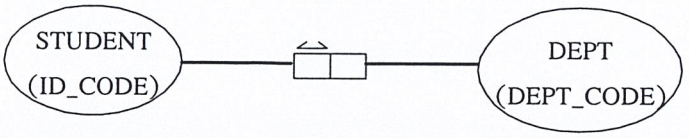
รูปที่ 4-1 แสดงสัญลักษณ์ของชนิดเอนติตี้ภาควิชา

รูปที่ 4-2 แสดงสัญลักษณ์ของชนิดเลเบลรหัสภาควิชา



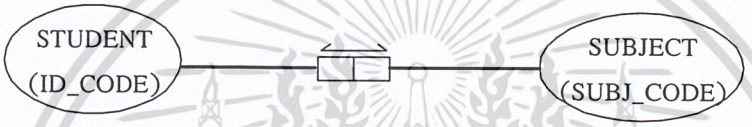
รูปที่ 4-3 แสดงความสัมพันธ์อ้างอิงแบบ one to one ภาควิชาใด ๆ จะมีรหัสภาควิชาเพียงรหัสเดียวเท่านั้นและไม่ซ้ำกับภาควิชาอื่น)

รูปที่ 4-4 แสดงการเขียนความสัมพันธ์อ้างอิงแบบ one to one



รูปที่ 4-5 แสดงความจริงแบบ many to one

(นักศึกษาหนึ่งคนจะสังกัดภาควิชาได้เพียงภาคเดียว แต่ภาควิชาใด ๆ สามารถศึกษาในสังกัดได้มากกว่าหนึ่งคน)



รูปที่ 4-6 แสดงความจริงแบบ many to many

(นักศึกษาหนึ่งคนสามารถลงทะเบียนเรียนได้หลายวิชา และแต่ละวิชาที่เปิดสอน สามารถรับจำนวนนักศึกษาได้มากกว่าหนึ่งคน แต่นักศึกษาที่ลงทะเบียนเรียนวิชาใด ๆ แล้วจะลงทะเบียนซ้ำ วิชาเดิมไม่ได้: ไม่เป็นจริงในทางปฏิบัติ

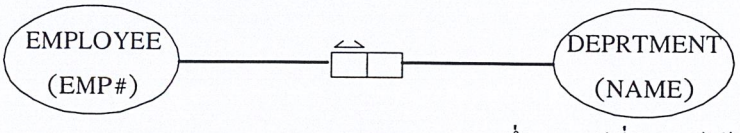
ชนิดเอนติตี้เป็นเซ็ต (Set) ซึ่งมีสมาชิกเป็นตัวอย่างเอนติตี้ (Entity Instance) เช่น ภาควิชา A (ภาควิชาโทรคมนาคม) ภาควิชา B (ภาควิชาวิศวกรรมคอมพิวเตอร์) เป็นตัวอย่างเอนติตี้ของชนิดเอนติตี้ภาควิชา

เครื่องหมายความสัมพันธ์ที่เป็นส่วนเชื่อมโยงระหว่างชนิดเอนติตี้ และชนิดเอนติตี้ หรือชนิดเลเบลนั้น เรียกว่า บทบาท (role) จะเขียนความหมายของบทบาทนั้นไว้ภายในหรือข้าง ๆสัญลักษณ์ของมัน

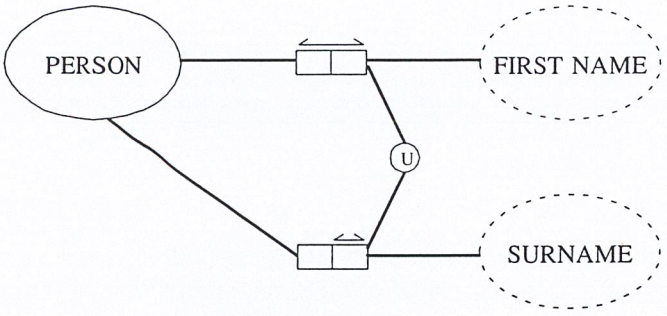
การแปลงข้อมูลที่วิเคราะห์มาให้อยู่ในรูปแบบจำลอง ก่อนอื่นต้องนำข้อมูลมากำหนด เป็นชนิดเอนติตี้และเลเบิลให้เรียบร้อยเสียก่อน จึงนำชนิดเอนติตี้ที่ได้มาเขียนเป็นประโยค ความจริงมูลฐาน (Elementary Fact)

แล้วเอาความจริงทั้งหมดที่ได้มาเขียนเป็นแบบจำลอง และเติมข้อจำกัดต่าง ๆ ลงไปตามความเป็นจริงในขอบเขตของงาน

ตัวอย่างการใช้ข้อจำกัดต่าง ๆ แสดงไว้ดังนี้

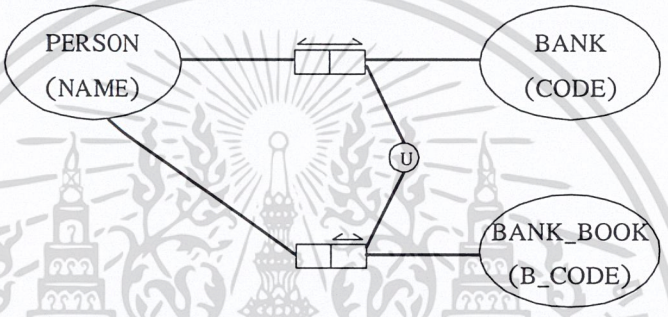


รูปที่ 4-7 แสดงการใช้ intra fact type uniqueness constraint (Employee หนึ่งคนจะมีที่ทำงานได้ที่เดียวเท่านั้น)



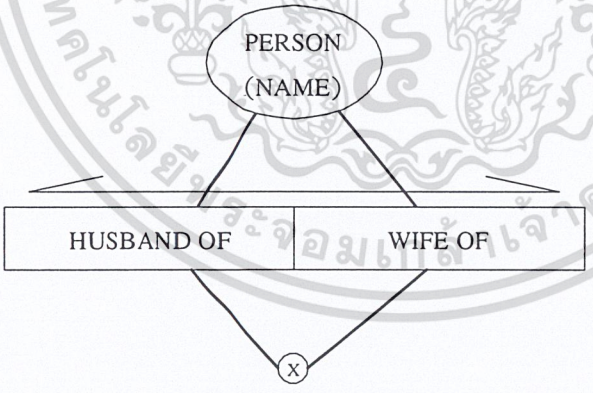
รูปที่ 4-8 แสดงการใช้ inter fact type uniqueness constraint

(บุคคลหนึ่งจะมี ชื่อ 1 ชื่อ นามสกุล 1 นามสกุล ชื่อของบางคนอาจจะซ้ำกันและนามสกุลของบางคนอาจจะซ้ำกัน แต่ชื่อรวมกับนามสกุลจะต้องไม่ซ้ำกัน)



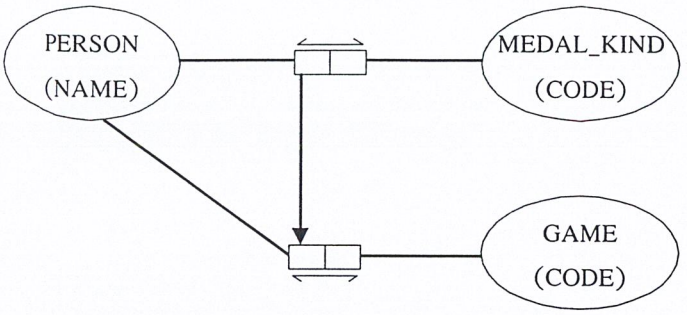
รูปที่ 4-9 แสดงการใช้ equality constraint

(ถ้าบุคคลหนึ่งเป็นลูกค้าของธนาคารใดแล้วบุคคลนั้นต้องมีสมุดบัญชีของธนาคารนั้นด้วย หรือในทางกลับกัน ถ้าบุคคลใดมีสมุดบัญชีของธนาคารใดแล้ว ก็ต้องเป็นลูกค้าของธนาคารนั้นด้วย)



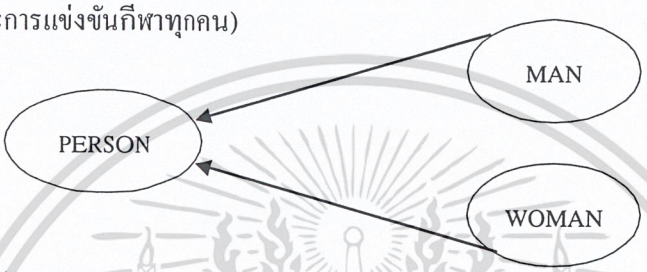
รูปที่ 4-10 แสดงการใช้ exclusion constraint

(บุคคลใดเป็นภรรยาของอีกบุคคลหนึ่งแล้ว บุคคลนั้นต้องไม่เป็นสามีของบุคคลใด ๆ ในทางกลับ บุคคลที่เป็นสามีของบุคคลหนึ่งแล้ว จะต้องไม่เป็นภรรยาของบุคคลใดด้วย)



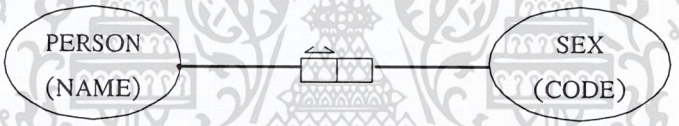
รูปที่ 4-11 แสดงการใช้ subset constraint

(บุคคลใดที่ชนะกีฬาการแข่งขันกีฬาทุกคนจะต้องเป็นบุคคลที่เล่นกีฬา แต่บุคคลที่เล่นกีฬาไม่จำเป็นต้องชนะการแข่งขันกีฬาทุกคน)



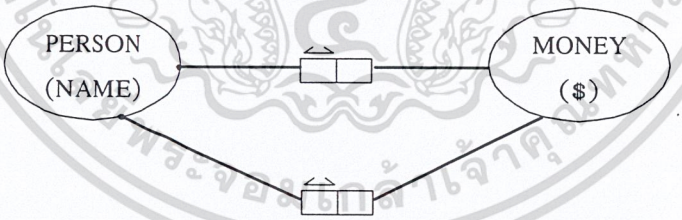
รูปที่ 4-12 แสดงการใช้ subtype constrint

(เอนติตี้ทุกตัวของชนิดเอนติตี้ผู้ชาย และชนิดเอนติตี้ผู้หญิง ต่างก็เป็นสมาชิกของเอนติตี้บุคคล)



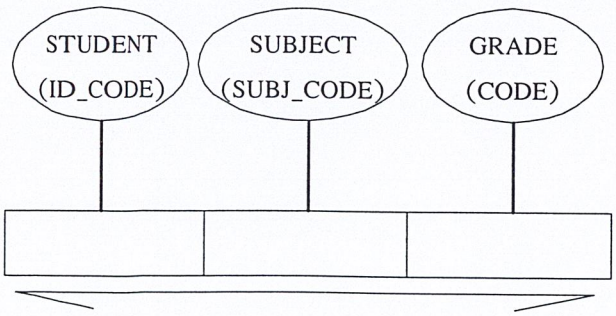
รูปที่ 4-13 แสดงการใช้ mandatory constraint,lexical constraint

(บุคคลทุกคนต้องมีเพศและสมาชิกของเอนติตี้เพศมีเพียง M:Male และF :Female)

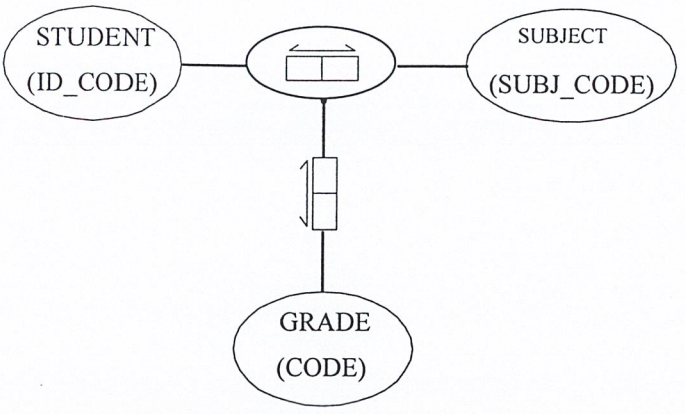


รูปที่ 4-14 แสดงหมายความว่า Employee ทุกคนต้องมีรายได้ และทุกคนต้องเสียภาษี)

นอกจากตัวอย่างความจริงแบบ Binary Fact Type ที่แสดงไว้ข้างต้นแล้วยังมีตัวอย่างชนิดอื่นอีก เช่น



รูปที่ 4-15 แสดงTernary Fact Typ (ข้อมูลการเรียนของนักศึกษาทุกคนจะต้องมีทั้งรหัสวิชาและเกรด)



รูปที่ 4-16 แสดง Nested Fact Type โดยแสดง ข้อมูลการเรียนของนักศึกษาทุกคนจะต้องมีทั้งรหัสวิชาและเกรด

4.4 TOONIAM

โมเดลสำหรับออกแบบฐานข้อมูลเชิงเวลา โดย TOONIAM นี้เป็นการพัฒนามาจาก NIAM ซึ่งเป็นโมเดลในการออกแบบฐานข้อมูลเชิงเวลาและสร้างเครื่องมือในการออกแบบฐานข้อมูลเชิงเวลา โดยใช้ NIAM (Nijssen's Information Analysis Methodology) ซึ่ง NIAM นั้นช่วยในการออกแบบฐานข้อมูลสัมพันธ์ได้ดีกว่า โดยสามารถที่จะแปลงเป็นรูป 5NF (Fifth Normal Form) ได้เลยซึ่งในขณะที่ ER Model จะแปลงได้แค่ 3 NF (Third Normal Form) โดยนอกจากนี้ยังได้ทำการเพิ่มหลักการเชิงวัตถุเข้าไปด้วย เพื่อให้เครื่องมือใหม่นี้สามารถออกแบบฐานข้อมูลเชิงเวลาบนฐานข้อมูลเชิงวัตถุได้ และยังได้สร้างอัลกอริธึมในการแปลงรูปจากแบบจำลองที่สร้างขึ้นให้เป็นภาษานิยมเชิงวัตถุ (Object Definition Language) อีกด้วย

ในการที่ออกแบบฐานข้อมูล โดยใช้เครื่องมือในการออกแบบฐานข้อมูลเชิงวัตถุในแบบอื่นๆ นั้นจะพบปัญหาดังนี้

1. Class Diagram ของ UML (Unified Modeling Language) นั้นไม่สามารถบอกได้ว่าส่วนใดเป็น Embedded Class และส่วนใดเป็น คลาส ที่ต้องสร้างใหม่

2. Class Diagram ไม่ได้นำเสนอ Function Dependency (FD) ซึ่งอาจทำให้ Attribute ซ้ำซ้อนกันได้

3.Class Diagram ไม่สามารถรองรับหลักการเชิงเวลาได้

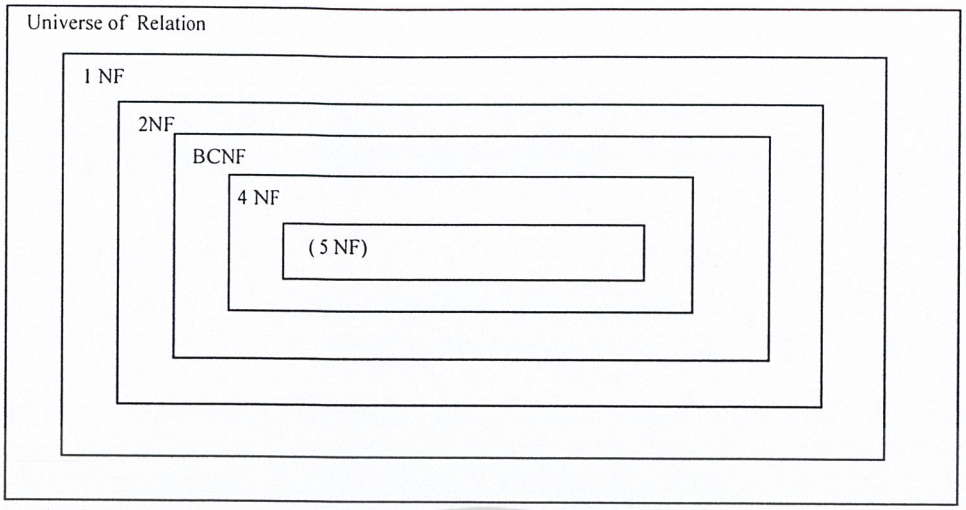
โดยถ้าอาศัยหลักการของ TOONIAM ก็จะแก้ไขปัญหาทั้งสามประการนี้ได้ โดยสำหรับข้อจำกัดทั้งสามประการของ UML ทำการแก้ไขปัญหา 3 ข้อนี้ได้ดังนี้ ?

- เนื่องจาก NIAM สามารถแปลงเป็น 5 NF ได้เลย จากปัญหาในข้อที่ 2 .ที่ว่า "ClassDiagram ไม่ได้นำเสนอ Function Dependency (FD) ซึ่งอาจทำให้ Attribute ซ้ำได้"

โดยจากคุณสมบัติของ 3 NF ที่ว่า 1. 2 NF

2. ไม่มี FD ระหว่าง non-key Attributes

โดย Assumption ว่าถ้าสามารถออกแบบฐานข้อมูลให้แปลงรูปเป็น 5 NF ได้ ก็จะต้องมีคุณสมบัติของ 3 NF ไปโดยปริยาย ซึ่งสามารถแก้ปัญหาข้อที่ 2.ได้



รูปที่ 4-17 แสดง Normal Form

การสร้างแบบจำลองข้อมูลไนแอมจะมีพื้นฐานอยู่บนการกำหนดตัวอย่างข้อมูล และหลังจากที่ผ่านกระบวนการ (Procedure) ที่ได้กำหนดไว้ เราก็จะได้แผนภาพที่มีความหมายในการแทนแบบจำลองข้อมูลดังกล่าวได้ โดยขั้นตอนการออกแบบ NIAM นั้นมีทั้งหมด 9 ขั้นตอน ดังนี้

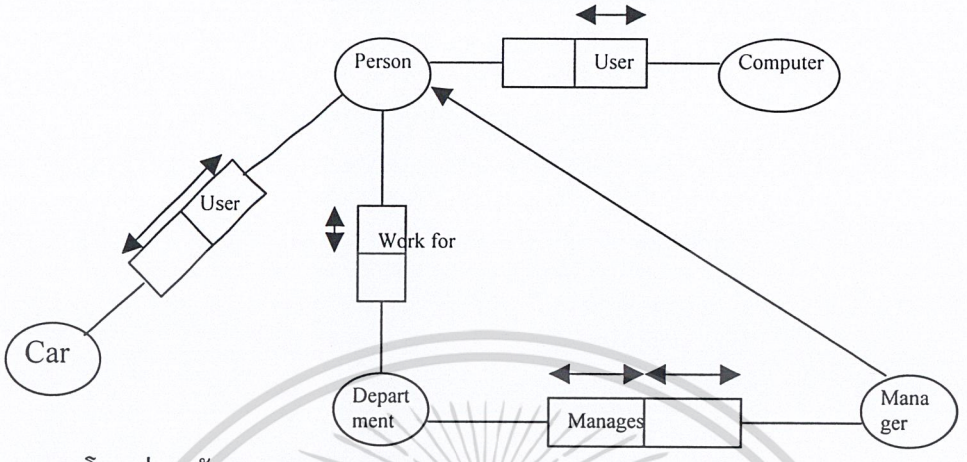
1. From Example to elementary facts
2. First Draft of conceptual schema diagram
3. Trim schema and find derived fact type
4. Uniqueness Constraint
5. Arity Checks
6. More Constraints
7. Entity Identification Schemes
8. Further Constraints
9. Final Check

4.4.1 เหตุผลที่ต้องทำ TOONIAM

การปรับปรุงให้ Model NIAM ที่มีอยู่เดิมให้สามารถรองรับแนวความคิดเชิงวัตถุรวมทั้งเชิงเวลาได้ โดยนำเอาโมเดล OONIAM มาเป็นโครงสร้างพื้นฐาน และปรับปรุงพัฒนามาเป็นโมเดลแบบ TOONIAM

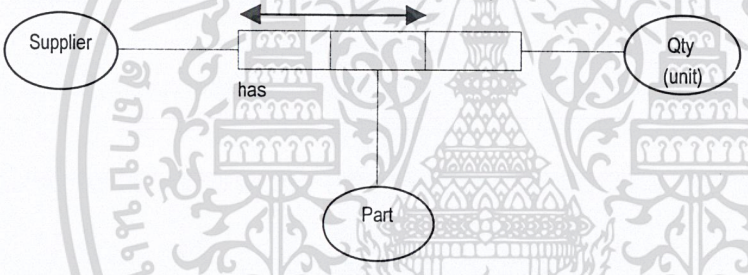
การใช้งาน ER และ NIAM นั้นจะใช้กับระบบฐานข้อมูลเชิงสัมพันธ์ แต่ในปัจจุบันระบบฐานข้อมูลไม่ได้มีเพียงฐานข้อมูลเชิงสัมพันธ์เท่านั้น แต่มีระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ และระบบฐานข้อมูลเชิงวัตถุอีกด้วยซึ่งระบบฐานข้อมูลทั้งสองได้รับการพัฒนาให้มีขีดความสามารถที่สูงขึ้น โดยให้มีแนวความคิดที่สามารถรองรับข้อมูลและกระบวนการของข้อมูลที่สลับซับซ้อนมากขึ้น เมื่อมีระบบฐานข้อมูลใหม่ขึ้นมาแล้ว ก็ต้องมีการออกแบบใหม่ด้วยซึ่งจะเรียกโมเดลใหม่นี้ว่า โมเดลไนแอมเชิงวัตถุ (Object Oriented NIAM Schema Model) หรือเรียกว่า OONIAM โดย OONIAM ได้แบ่งส่วนสำคัญออกเป็นสองส่วนดังนี้

4.4.1.1 โครงร่างหลัก (Main Schema)โครงร่างหลักนี้เป็นภาพของความสัมพันธ์ระหว่างคลาส ดังนั้นในแต่ละคลาสจะไม่มีการแสดงแอตทริบิวต์ และไม่มี Uniqueness Identifier



รูปที่ 4-18 แสดง โครงร่างหลักของคลาสหลายคลาส

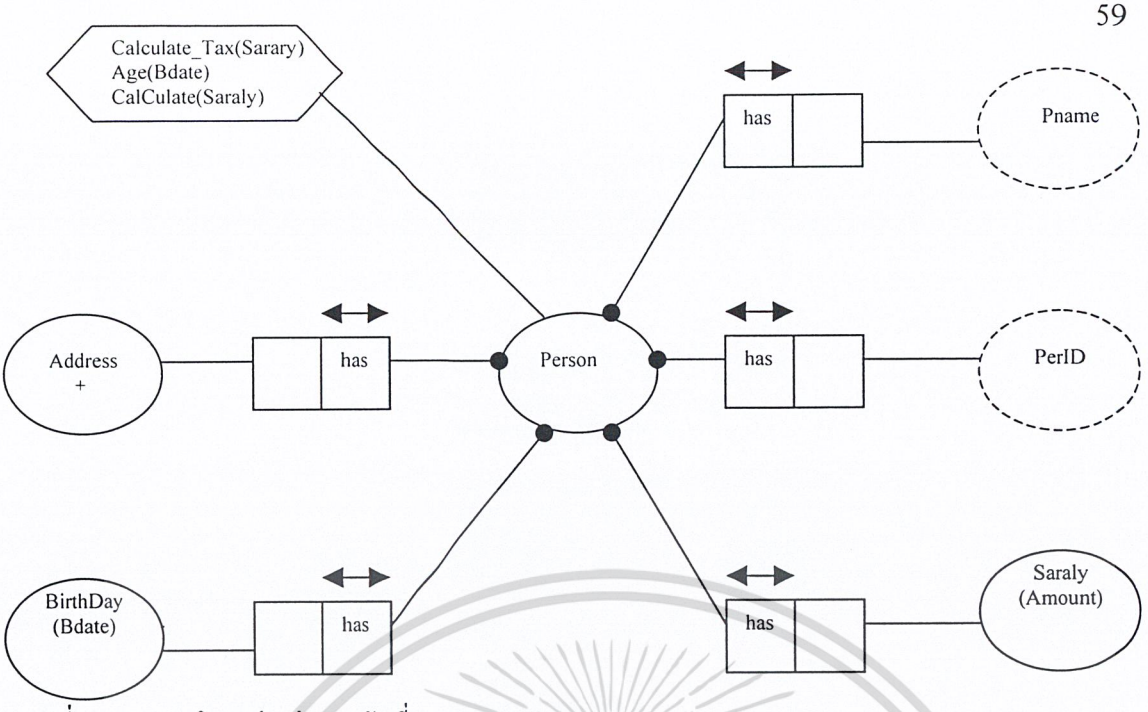
โดยโครงร่างหลัก นี้จะแสดงความสัมพันธ์ระหว่างคลาสแบบหนึ่งต่อหนึ่ง (one-to-one)หรือหนึ่งต่อหลาย (one-to-many) หรืออาจเป็นแบบ(many-to-many) ก็ได้ แล้วแต่การออกแบบคลาสนั้น ๆ



รูปที่ 4-19 แสดงโครงร่างหลักแบบที่มี Uniqueness Identifier ที่ Entity

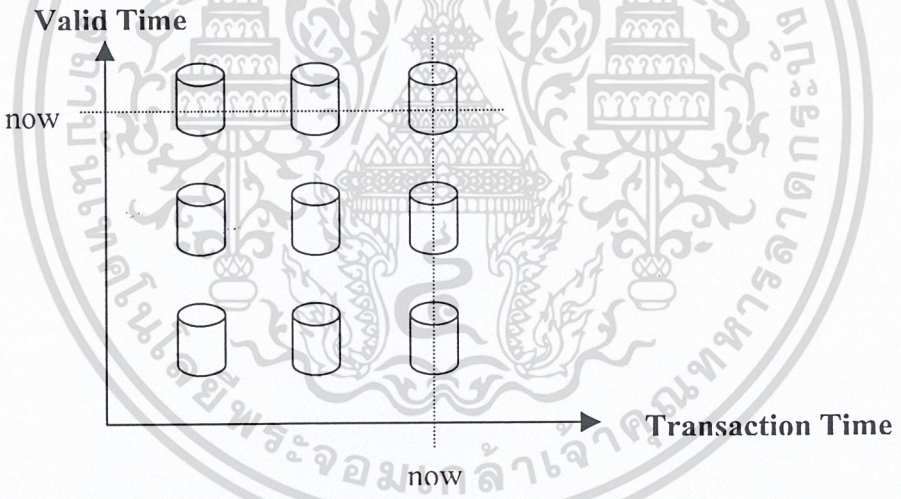
4.4.1.2 โครงร่างย่อย (Sub Schema) โครงร่างย่อยเป็นการแสดงรายละเอียดของแต่ละคลาส ซึ่งแต่ละคลาสนั้นอาจเป็นคลาสซับซ้อน (Complex Object) ก็ได้ กล่าวคือคลาสดังกล่าวอาจมีคลาสอื่นเป็นส่วนประกอบก็ได้ นอกจากนี้ยังได้เพิ่มคุณสมบัติให้สามารถกำหนดได้ในกรณีที่ คลาสนั้นๆ เป็นคลาสย่อย (Embebdded Class) โดยจะใช่เครื่องหมายบวก (+) ไว้ใต้ชื่อคลาส

ซึ่งจากปัญหาข้อที่1. ที่กล่าวว่า “Class Diagram ของ UML (Unified Modeling Language) นั้นไม่สามารถบอกได้ว่าส่วนใดเป็น Embedded Class และส่วนใดเป็น คลาส ที่ต้องสร้างใหม่” จะเห็นได้ว่าสามารถแก้ปัญหาในข้อที่1.ได้



รูปที่ 4-20 แสดงโครงร่างย่อยระดับที่ 1 ของ Class Person

โดยแนวคิดของแบบจำลอง TOONIAM นี้ได้นำเอา OONIAM มาเพิ่มขยายโครงสร้างให้สามารถรองรับเวลาของระบบฐานข้อมูลเชิงเวลาได้

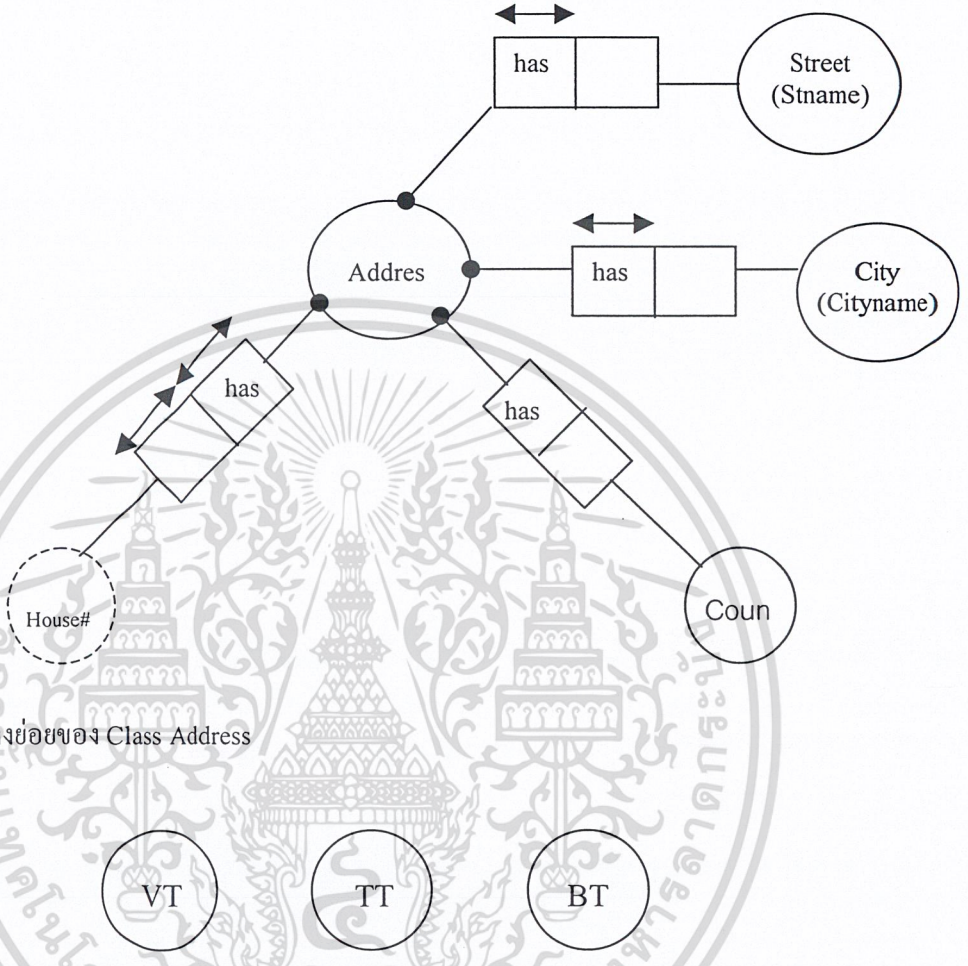


รูปที่ 4-21 แสดงการจัดเก็บข้อมูลของระบบฐานข้อมูลเชิงเวลา

โดยเวลาของระบบฐานข้อมูลเชิงเวลานั้นประกอบไปด้วยเวลา 3 ชนิดคือ

- Transaction Time (TT) เป็นเวลาที่ข้อมูลอยู่ในฐานข้อมูลและเวลาที่ข้อมูลอาจได้รับการแก้ไข โดยเวลาแบบนี้ระบบจะจัดหาให้ ผู้ใช้ไม่ต้องทำการป้อนข้อมูลเอง
- Valid Time (VT) คือเวลาที่ข้อเป็นจริง ซึ่งผู้ใช้งานจะต้องทำการป้อนเวลาที่ข้อมูลนั้นเป็นจริงเข้าไปในระบบ โดยผู้ใช้งานต้องกำหนดเวลาที่ข้อมูลเป็นจริง เวลาดังกล่าวจะระบุหน่วยของเวลาเข้าไปด้วย เช่น ปี เดือน วัน เป็นต้น หรืออาจทำการระบุเป็นช่วงเวลาได้
- Bitemporal Time (BT) คือ จะมีการจัดเก็บทั้ง Transaction Time และ Valid Time เข้าไว้ด้วยกัน สำหรับข้อมูลที่ใช้กับเวลาประเภทนี้คือ ข้อมูลที่ต้องการการจัดเก็บทั้งเวลาที่จัดเก็บและเวลาที่ข้อมูลเป็นจริง

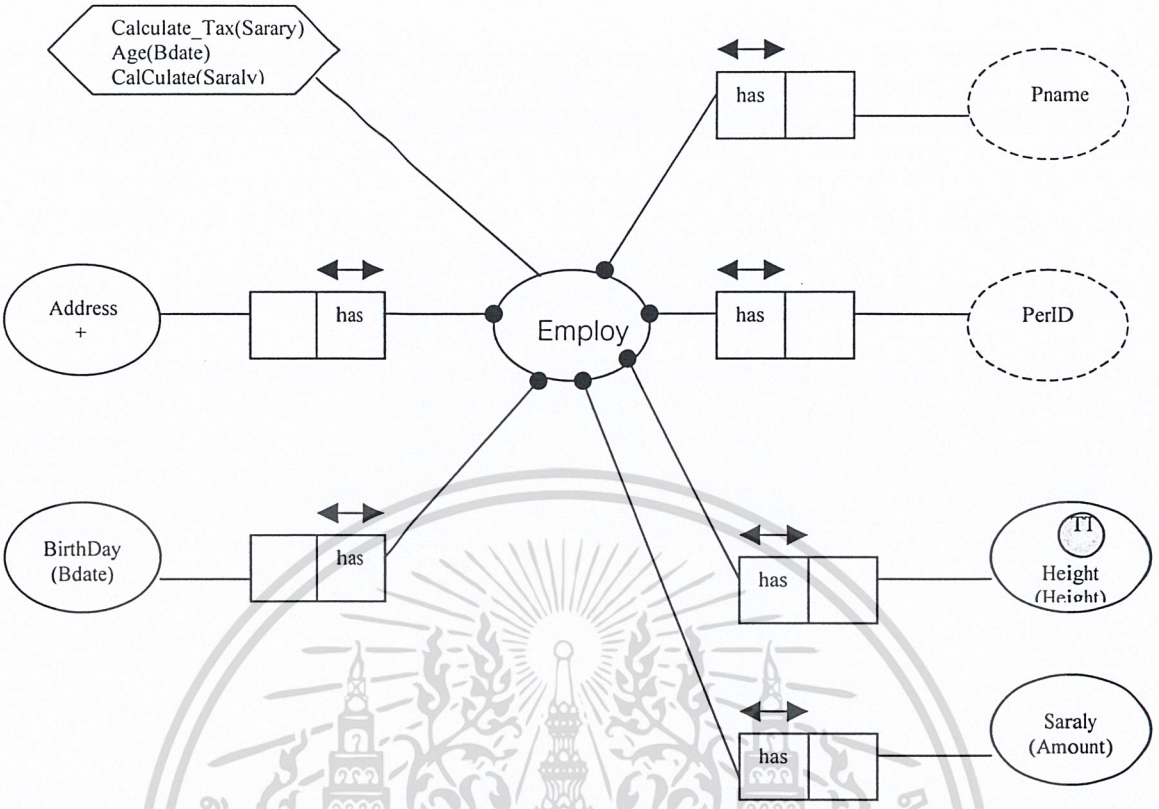
สำหรับปัญหาข้อที่3. ที่ว่า “คลาสไดอะแกรมไม่สามารถรองรับหลักการเชิงเวลาได้” ดังนั้นถ้าเรานำหลังการของ และแนวคิดของแบบจำลอง TOONIAM มาใช้ก็จะสามารถที่จะแก้ปัญหาข้อที่3นี้ได้ เนื่องจาก TOONIAM มีการออกแบบที่ลองรับหลักการเชิงเวลาได้



รูปที่ 4-22 แสดงโครงร่างย่อยของ Class Address

รูปที่ 4-23 แสดง สัญลักษณ์ของเวลาที่ใช่ใน TOONIAM

โดยเราจะนำเวลาที่เพิ่มเข้าไปนั้น จะต้องนำไปเพิ่มในส่วนของโครงร่างหลัก ซึ่งจะเป็นเวลาของความสัมพันธ์ (Relationships with time) โดยหากต้องการให้ความสัมพันธ์ของคลาสมีเวลาแบบใดเข้ามาเกี่ยวข้อง ก็สามารถที่จะนำสัญลักษณ์ของเวลาแบบนั้นๆ ไปใส่ไว้ยังความสัมพันธ์ที่ต้องการได้



รูปที่ 4-24 แสดงโครงร่างย่อยของข้อมูลบุคคลที่จัดเก็บเวลาที่ข้อมูลอยู่ในฐานข้อมูลไว้ด้วย

โดยสิ่งที่เพิ่มเข้าไปนั้นก็คือการจัดการเก็บเวลา Transaction Time เข้าไปที่แอตทริบิวต์ความสูงนั้นหมายความว่าระบบจะมีการจัดเก็บประวัติความสูงของบุคคลลงในฐานข้อมูลในโครงงานนี้จะทำการศึกษาเฉพาะส่วนที่เกี่ยวกับการบริการนักศึกษาเพื่อนำไปใช้เป็นการทำ **Software Overriding** เพื่อเป็นการบริการ โดยมีแนวความคิดที่ว่าเมื่อมีการพัฒนาผลิต Software ชุดหนึ่งซึ่งนำไปใช้กับองค์กรใดแล้วก็จะสามารถนำไปใช้กับองค์กรที่มีส่วนเกี่ยวข้องหรือมีความประสงค์และหน้าที่ที่คล้ายคลึงกันก็จะสามารถนำผลิตภัณฑ์ หรือ Software ที่ได้รับการพัฒนาแล้วนั้นไปใช้ต่อในองค์กรนั้นๆ ได้ทันที โดยสามารถปรับเปลี่ยนหรือแลกคุณสมบัติ ที่ต้องมีหรือความจำเป็นที่ต้องใช้ function หน้าที่ตามองค์กรนั้นๆ ได้

โดยในโครงงานนี้นำระบบลงทะเบียนนักศึกษาในส่วนของที่เกี่ยวกับการบริการนักศึกษามาทำการพัฒนา Software Overriding โดยระบบลงทะเบียนที่จะทำการพัฒนานี้จะวิเคราะห์ในส่วนการทำงานหลักๆ ดังนี้

ฝ่ายรับเข้าศึกษาและทะเบียนประวัติ ดำเนินงานต่าง ๆ ดังนี้ (เฉพาะที่จะทำและพัฒนาเป็น Software Overriding ได้)

<u>งานสอบคัดเลือกและรับเข้าศึกษา</u>

- สถาบันต่างกันมีหลักสูตรในการรับคัดเลือกนักศึกษาที่ต่างกัน เช่น หลักสูตรปริญญาตรี หลักสูตรปริญญาโท และปริญญาเอก โดยต่างหลักสูตรกัน แล้วแต่สถาบัน อาจมีทุนหรือโควต้าต่างๆ ที่ต่างกัน

งานการรายงานตัวและลงทะเบียนแรกเข้า

- การจัดการลงทะเบียนเรียนและชำระเงินนักศึกษาใหม่ทุกประเภท โดยนักศึกษาต่างหลักสูตรกัน เช่น หลักสูตรปริญญาตรี หลักสูตรปริญญาโท และปริญญาเอก

งานกำหนดรหัสประจำตัวนักศึกษาและการออกบัตรประจำตัว

- สถาบันต่างกันการกำหนดรหัสประจำตัวนักศึกษาแรกเข้า แยกตามระดับการศึกษาหรือตามสถาบัน

งานทะเบียนประวัตินักศึกษา

- การวิเคราะห์ข้อมูลนักศึกษาใหม่ แบ่งตามหลักสูตร หลักสูตรปริญญาตรี หลักสูตรปริญญาโท และปริญญาเอก

งานลงทะเบียนเรียนและชำระเงิน

- จัดการลงทะเบียนเรียนของนักศึกษาปัจจุบันทุกประเภท โดยหน่วยกิจต่างกันตามหลักสูตร โดยบางคณะมีกำหนดให้ลงกี่หน่วยหรือบางที ก็อาจจะไม่มีกำหนดให้นักศึกษาลงได้ไม่จำกัด แต่บางที่ก็จะยอมให้ลงเกินกำหนดถ้านักศึกษาผู้นั้นขอจบการศึกษาในภาคการศึกษานั้น
- การรับลงทะเบียน และชำระเงินเพื่อการรักษาสภาพนักศึกษา โดยอาจคิดอัตราค่าลงทะเบียนตามหลักสูตร ตามสถาบัน

งานเพิ่ม เปลี่ยน ถอนวิชาเรียน

- งานรับลงทะเบียนเพิ่มวิชาเรียนของนักศึกษาตามหลักสูตร
- งานลงทะเบียนการถอนวิชาเรียน โดยกฎ และ ระเบียบต่างๆ อาจแตกต่างกันตามสถาบัน
- งานตรวจสอบรายวิชาเรียนของนักศึกษาภายหลังการเพิ่มเปลี่ยน และถอนวิชาเรียน โดยกฎและ ระเบียบต่างๆ อาจแตกต่างกันตามสถาบัน

งานลาพักการศึกษาและพ้นสภาพ

- การจัดทำแบบฟอร์ม การขอลาพักการศึกษา ต่างสถาบันกัน
- การตรวจสอบระยะการลาพักการศึกษา ให้เป็นไปตามระเบียบการลาของสถาบันฯ ตามกฎสถาบัน
- การประกาศรายชื่อนักศึกษาพ้นสภาพ ของแต่ละภาคการศึกษา โดยกฎตามสถาบัน โดยอาจคิดตามหลักสูตร

งานบันทึกข้อมูลและตรวจสอบผลการศึกษา(อาจเหมือนกันทุกสถาบันได้)

- การบันทึกข้อมูล ผลการศึกษาของนักศึกษาทุกระดับเข้าเครื่องคอมพิวเตอร์
- การบันทึกผลการสอบตามรายวิชาของนักศึกษาทุกระดับเข้าเครื่องคอมพิวเตอร์
- การคิดค่าระดับคะแนนเฉลี่ยสะสมประจำภาค และค่าระดับคะแนนเฉลี่ยสะสม

งานตรวจสอบการสำเร็จการศึกษา

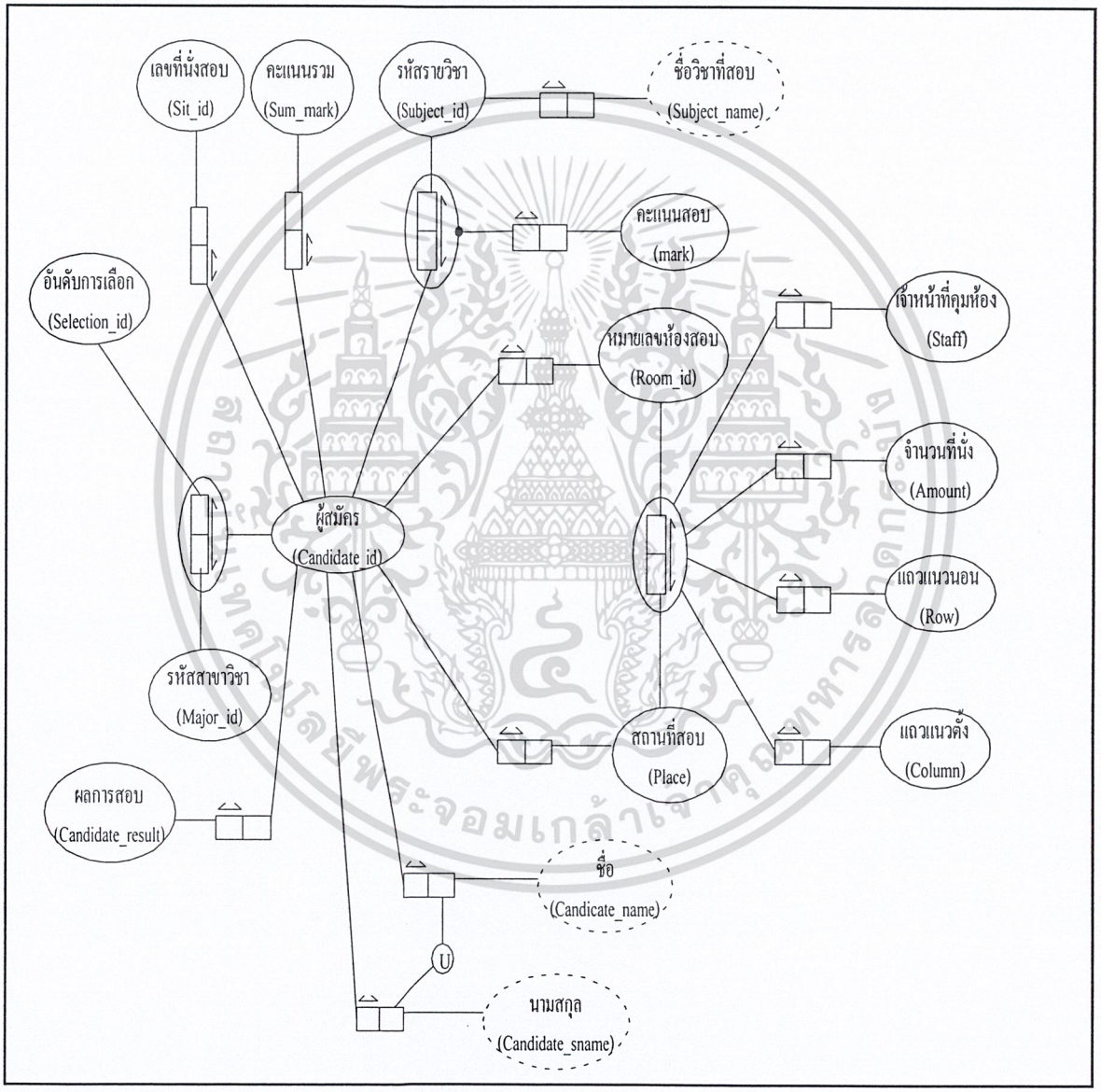
– การตรวจสอบรายวิชาที่ลงทะเบียนเรียน ตามหลักสูตรการศึกษาทุกระดับ โดยการเสนอขออนุมัติสำเร็จการศึกษา ตามกฎและระเบียบของสถาบัน โดยกฎและระเบียบแตกต่างตามสถาบัน

งานทะเบียนผู้สำเร็จการศึกษา

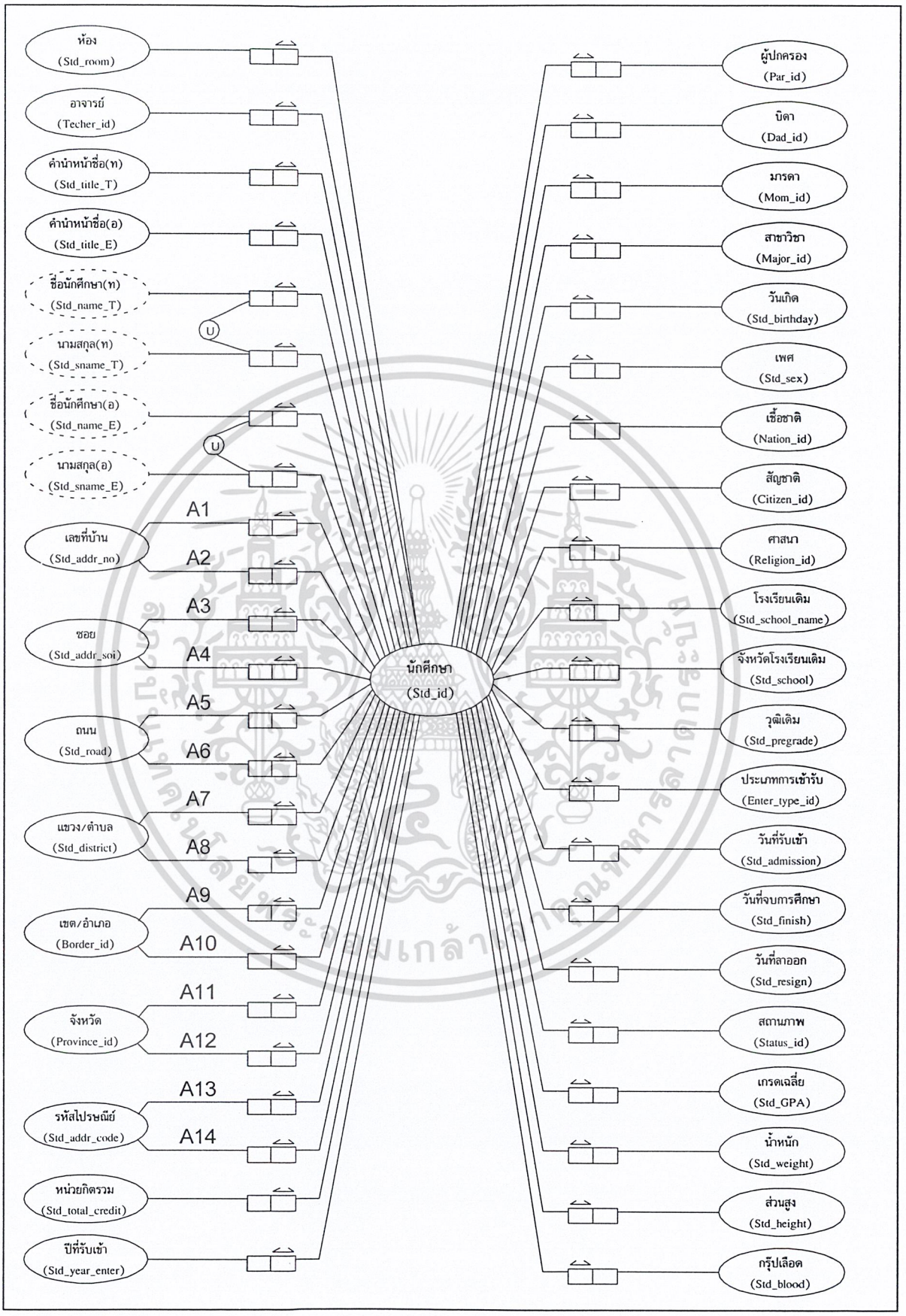
- การจัดทำแฟ้มทะเบียนผู้สำเร็จการศึกษา แยกตามสาขาวิชา คณะ หลักสูตร และระดับการศึกษา โดยกฎการอนุมัติจบตามสถาบัน

ในโครงงานนี้ได้ขอนำส่วนงานสอบคัดเลือกและรับเข้าศึกษา,งานทะเบียนประวัตินักศึกษาและงานเพิ่มเปลี่ยนถอนวิชาเรียนนำมาปรับปรุงโดยการแปลงรูปแบบจาก NIAM ไปสู่ OONIAM โดยรูปแบบเก่าของโครงร่าง NIAM มีดังนี้

แผนภาพแสดงความสัมพันธ์ของข้อมูล (NIAM) (Nijssen's Information Analysis Methodology)



รูปที่ 4-25 แสดงแผนภาพไนแอมของการรับเข้าศึกษา



รูปที่ 4-26 แสดงแผนภาพไนแอมของงานทะเบียนประวัตินักศึกษา

หมายเหตุ :

A1 = นิสิตรหัส...บ้านเลขที่....ตามทะเบียนบ้าน

A2 = นิสิตรหัส...บ้านเลขที่....ตามที่อยู่ปัจจุบัน

A3 = นิสิตรหัส...ซอย....ตามทะเบียนบ้าน

A4 = นิสิตรหัส...ซอย....ตามที่อยู่ปัจจุบัน

A5 = นิสิตรหัส...ถนน....ตามทะเบียนบ้าน

A6 = นิสิตรหัส...ถนน....ตามที่อยู่ปัจจุบัน

A7 = นิสิตรหัส...ตำบล....ตามทะเบียนบ้าน

A8 = นิสิตรหัส...ตำบล....ตามที่อยู่ปัจจุบัน

A9 = นิสิตรหัส...อำเภอ....ตามทะเบียนบ้าน

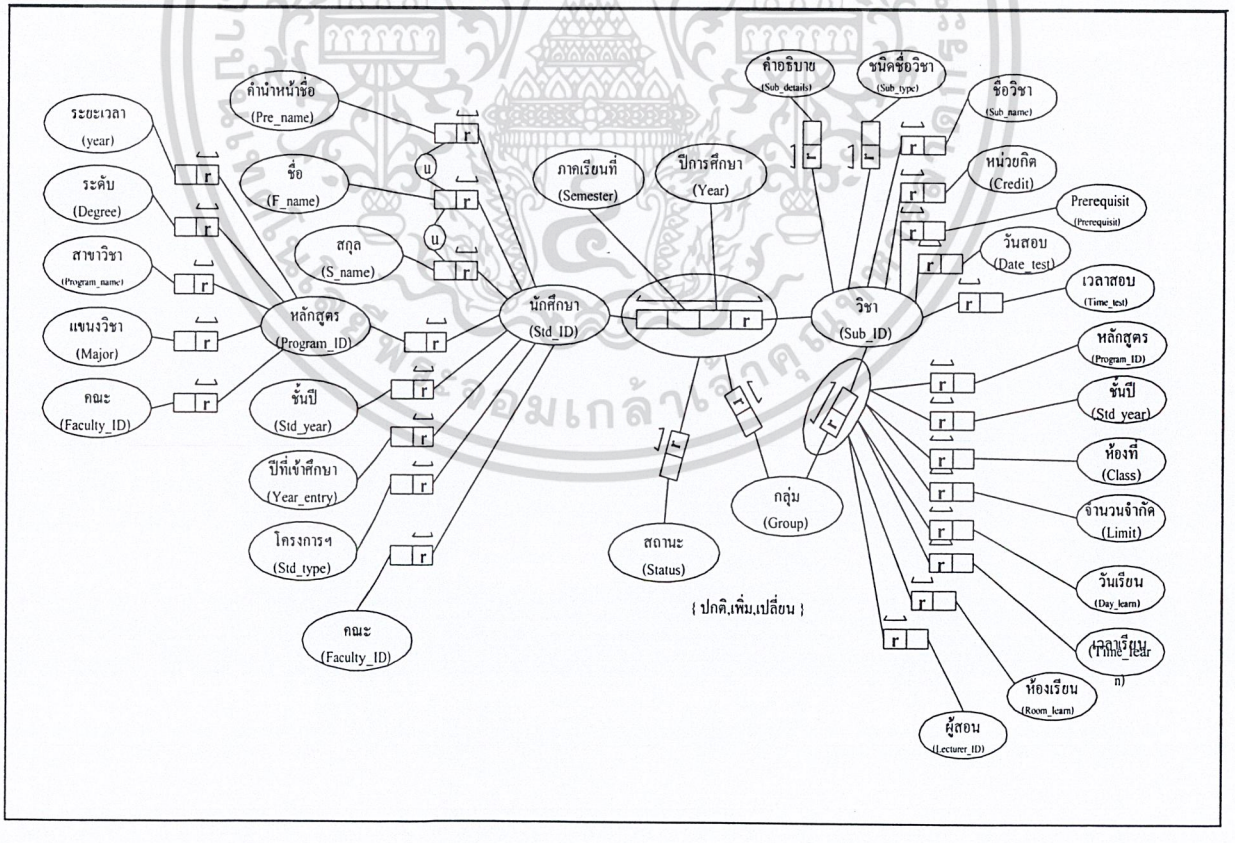
A10 = นิสิตรหัส...อำเภอ....ตามที่อยู่ปัจจุบัน

A11 = นิสิตรหัส...จังหวัด....ตามทะเบียนบ้าน

A12 = นิสิตรหัส...จังหวัด....ตามที่อยู่ปัจจุบัน

A13 = นิสิตรหัส...รหัสไปรษณีย์....ตามทะเบียนบ้าน

A14 = นิสิตรหัส...รหัสไปรษณีย์....ตามที่อยู่ปัจจุบัน

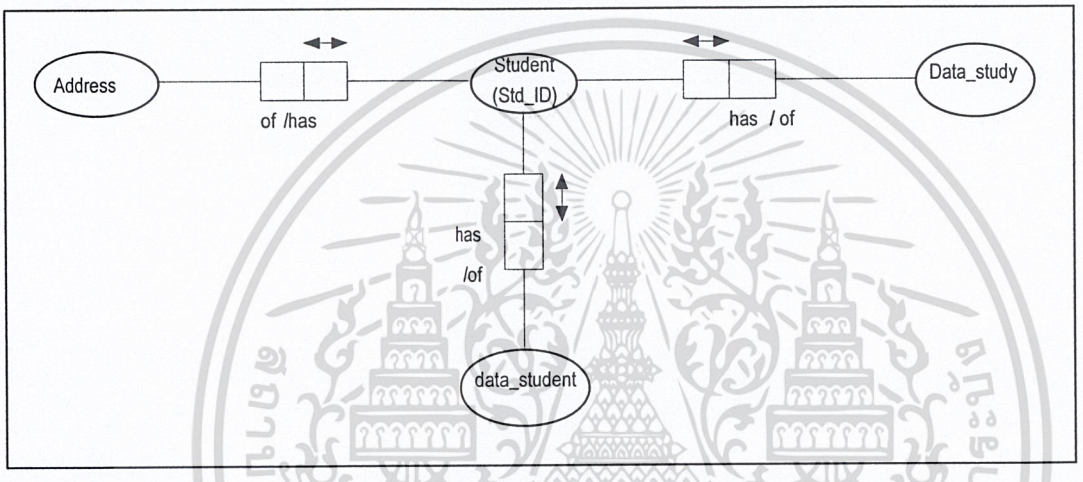


รูปที่ 4-27 แสดงแผนภาพไนแอมของระบบลงทะเบียน

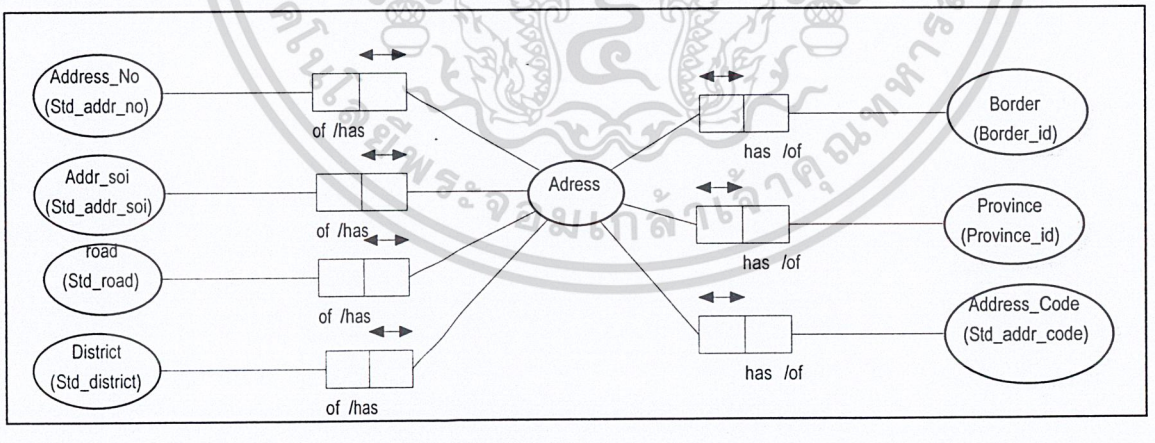
ในความสัมพันธ์ของการลงทะเบียนเรียนของนักศึกษา นักศึกษาจะต้องแจ้งรหัสประจำตัว ภาคเรียนที่ ปีการศึกษา รหัสวิชา และกลุ่มที่เรียน จากนั้นสถานะการลงทะเบียนจะถูกกำหนดด้วยโปรแกรมเพื่อการจัดการการลงทะเบียนรายวิชาปกติ ลงทะเบียนรายวิชาเพิ่ม ลงทะเบียนเปลี่ยนรายวิชา

จากแผนภาพแสดงความสัมพันธ์ของข้อมูล (NIAM) (Nijssen's Information Analysis Methodology) ของการ แสดงแผนภาพไนแอมของการรับเข้าศึกษา,แสดงแผนภาพไนแอมของงานทะเบียนประวัตินักศึกษาและ NIAM แสดงความสัมพันธ์ของข้อมูลนักศึกษากับการลงทะเบียน

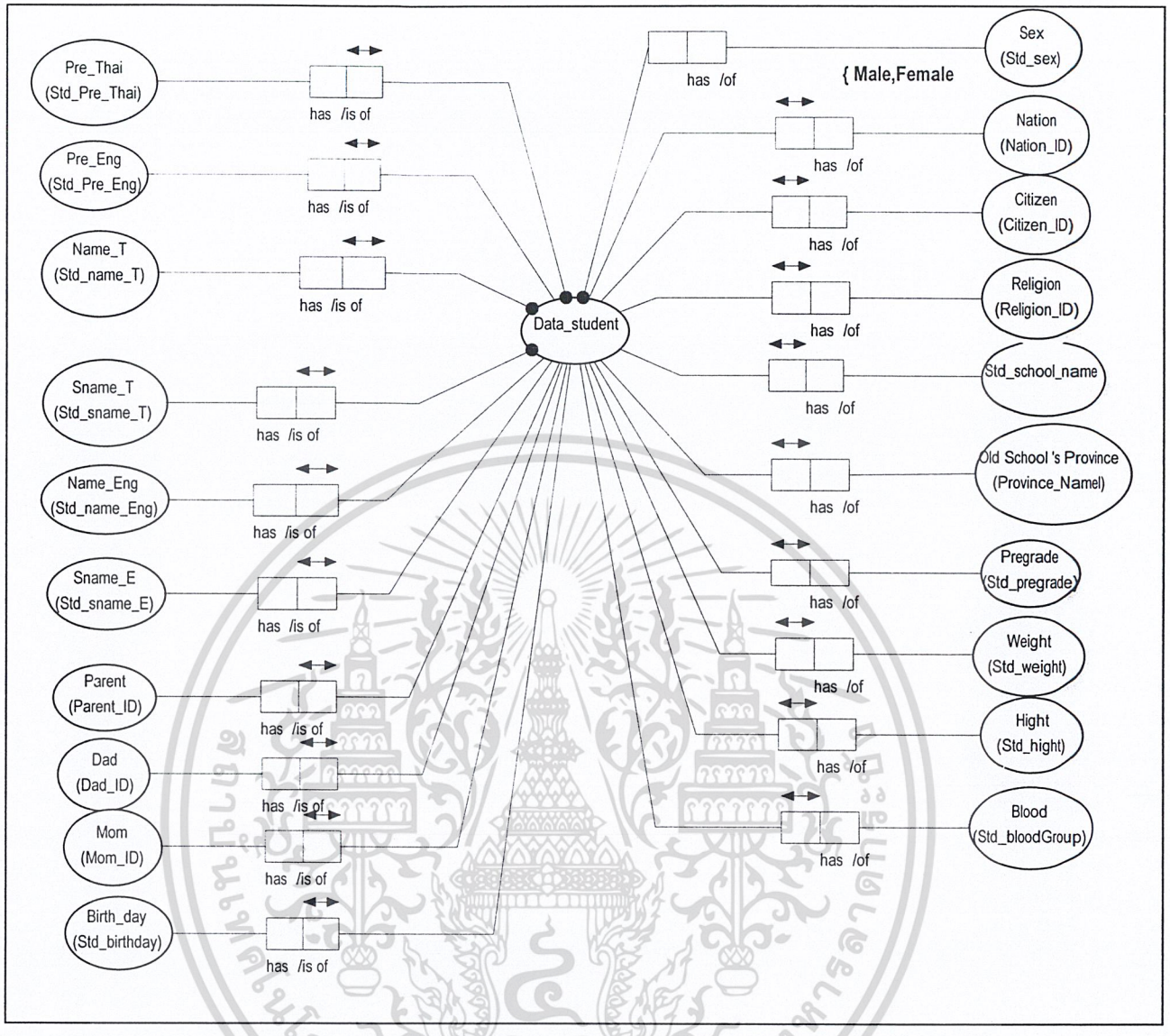
ทำการแปลงรูปแบบไนแอมดังกล่าวไปเป็น OONIAM โดยแสดงโครงร่างใหม่ที่แสดงเป็น OONIAM ได้ดังนี้



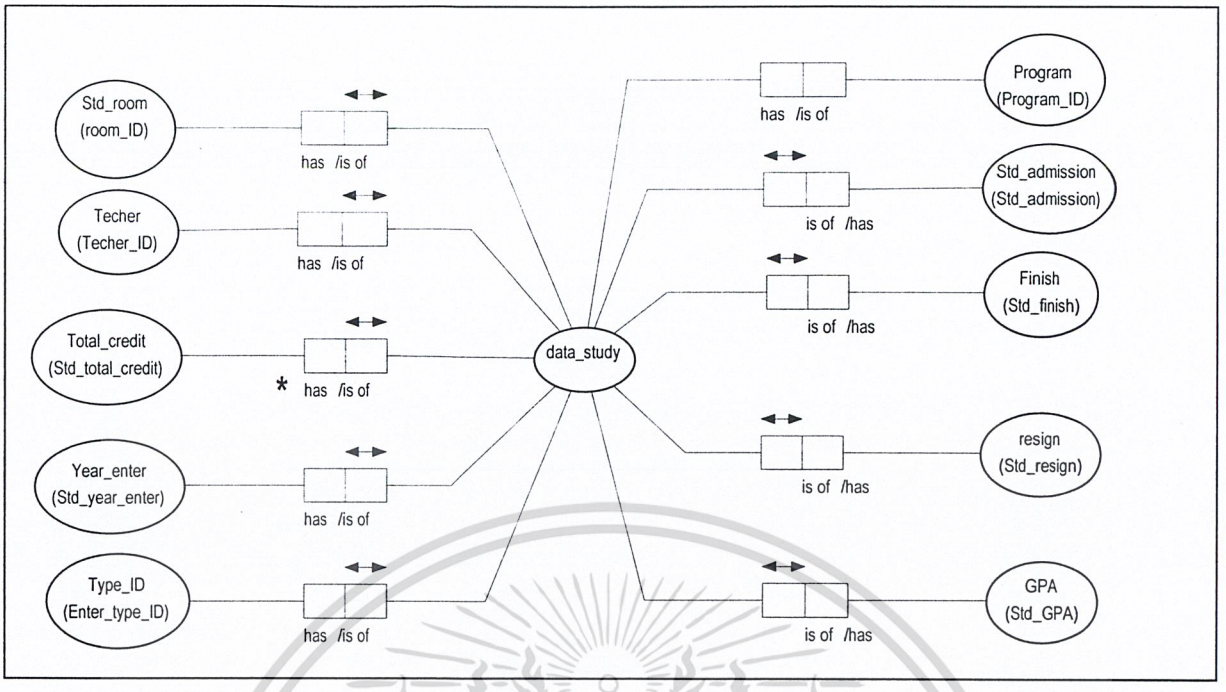
รูปที่ 4-28 แสดงโครงร่างหลักของคลาสงานทะเบียนประวัตินักศึกษา



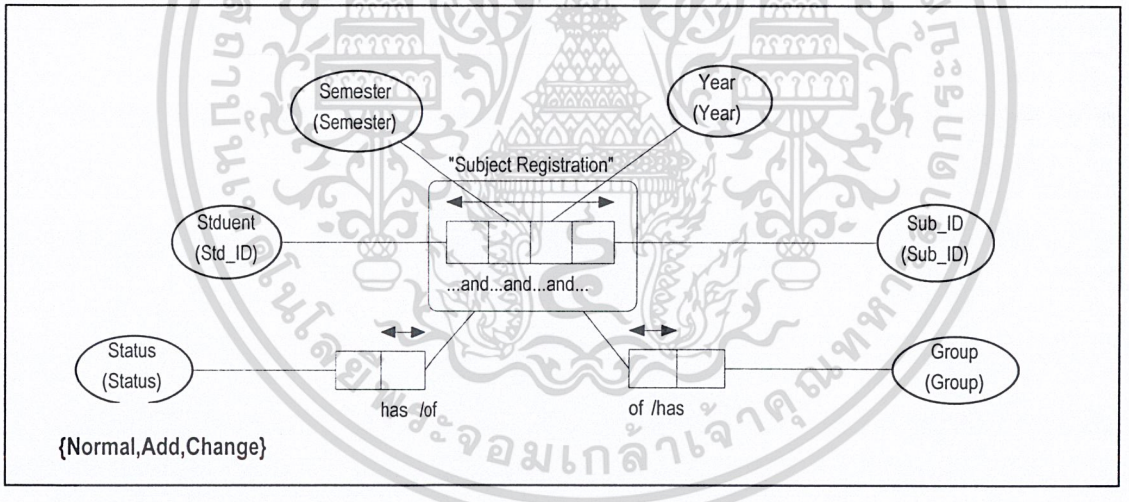
รูปที่ 4-29 แสดงโครงร่างย่อยของคลาสแอดเดรส



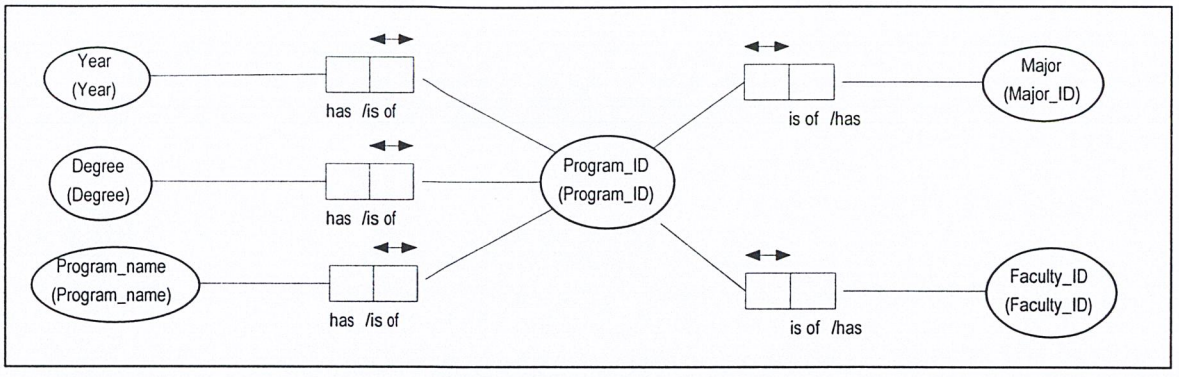
รูปที่ 4-30 แสดงโครงร่างย่อยของคลาสข้อมูลนักศึกษา



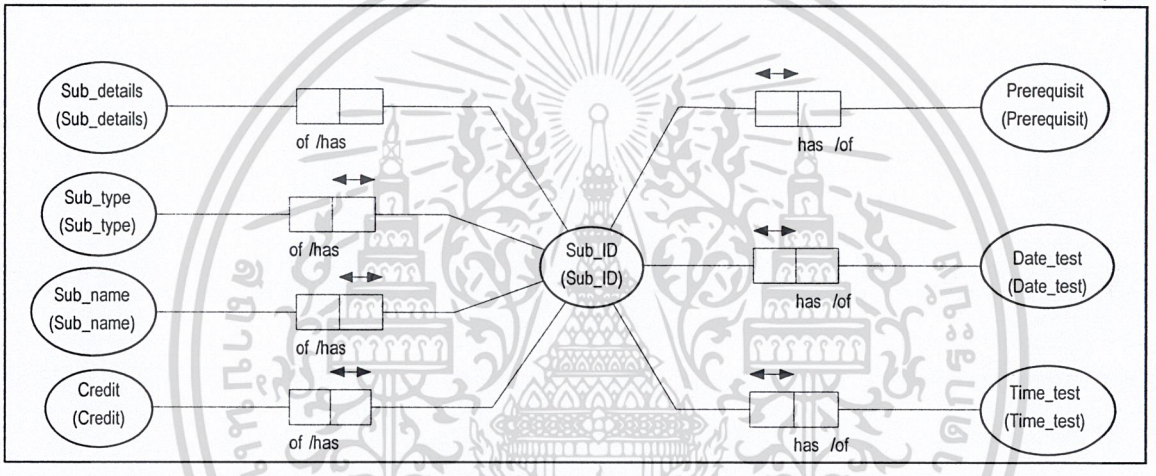
รูปที่ 4-31 แสดงโครงร่างย่อยของคลาสข้อมูลการศึกษา



รูปที่ 4-32 แสดงโครงร่างหลักของคลาสการลงทะบียน



รูปที่ 4-33 แสดงโครงร่างย่อยของคลาสหลักสูตรของนักศึกษา



รูปที่ 4-34 แสดงโครงร่างย่อยของคลาสรายวิชาที่ศึกษา

โดยจะนำทั้งโครงร่างหลักและโครงร่างย่อยที่ได้ทำการออกแบบนี้นำไปพัฒนาเป็นระบบสารสนเทศระบบลงทะเบียนนักศึกษาต่อไป

บทที่ 5

ระบบการจัดการฐานข้อมูลคาเช่

ระบบจัดการฐานข้อมูลคาเช่ (Cache′DBMS) จะประกอบด้วยส่วนองค์ประกอบย่อยๆ เพื่อเป็นเครื่องมือช่วยอำนวยความสะดวก ให้นักพัฒนาระบบฐานข้อมูล หรือผู้ดูแลระบบฐานข้อมูล ดังต่อไปนี้

1.**Object Architect** เป็นเครื่องมือที่เป็นแบบ GUI สำหรับพัฒนาระบบฐานข้อมูลโดย สามารถที่จะสร้างคลาส แก้ไขเพิ่มเติมโครงสร้างของคลาส แก้ไข Method code ลบคลาส และ Compile คลาส

2.Studio จะเป็น อิดิเตอร์เพื่อใช้เขียนโปรแกรมภาษา Cache′ Object Script หรือภาษา HTML

3.**Explorer** เป็นเครื่องมือที่ใช้ดูข้อมูล ในระบบระบบทั้งหมดเช่น คลาส Global Routine

4. SQL Manager เป็นเครื่องมือที่ใช้ดูข้อมูลในรูปแบบ ของ Relational Database และประกอบด้วยนิยามของตาราง และจะแสดงข้อมูลในระบบฐานข้อมูลในรูปแบบตาราง นอกจากนี้ยังมีความสามารถที่จะรับประมวลผลการทำงานของคำสั่งภาษา SQL

5. Control Panel เป็นเครื่องมือที่ใช้ในการบริหารระบบเช่น ระบบคามปลอดภัย ทำการสำรองข้อมูล ฯลฯ

6. Configuration Manager เป็นเครื่องมือที่ใช้สร้าง ลบ หรือแก้ไข Name Space หรือ ฐานข้อมูล และ ตั้งค่าการติดต่อผ่านระบบเครือข่ายคอมพิวเตอร์

7. System Viewer เป็นเครื่องมือที่ใช้ในการดู Processes การทำงานของระบบ ประสิทธิภาพของระบบ

8.Terminal เป็นเครื่องมือที่ใช้ รับสั่ง Cache′ Object Script โดยตรง กล่าวคือ Object Architect ทำหน้าที่เสมือน Graphic Interface ของ Terminal

5.1 Cache′ Object Architect

Cache′ Object Architect เป็นเครื่องมือในการพัฒนาระบบฐานข้อมูลที่มีรูปแบบการทำงานแบบกราฟิก ช่วยให้นักพัฒนาสามารถที่จะสร้าง คลาสโดยได้ไม่ยากเกินไปนัก หน้าที่หลักๆ ของ Cache′ object Architect นั้นคือ เรื่องที่เกี่ยวกับการจัด การคลาส เช่น สร้างแก้ไขและคอมไพล คลาส

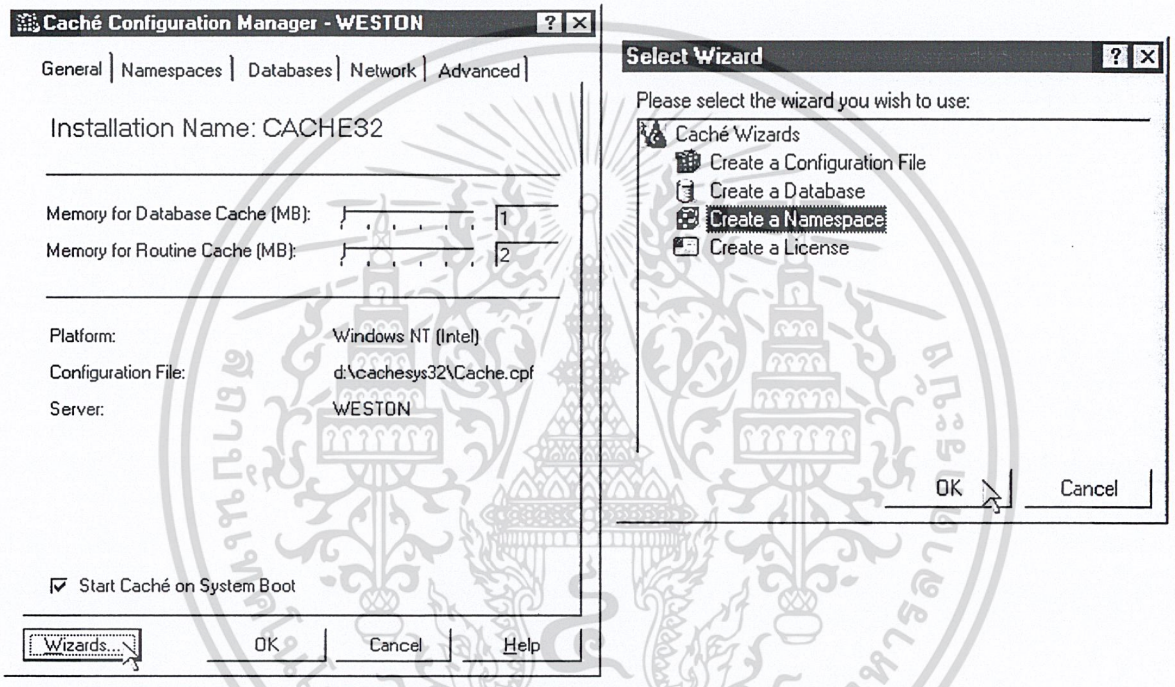
ขั้นตอนการสร้างระบบงานฐานข้อมูลโดยใช้ Cache′ Object Architect นั้นมีลำดับดังต่อไปนี้

5.1.1 สร้างฐานข้อมูลและกำหนดค่าต่างๆให้กับ Namespace และฐานข้อมูล

ลำดับการพัฒนาระบบงานนั้นเมื่อเราทราบความต้องการของระบบงานซึ่งก็คือระบบทะเบียนนักศึกษา และนำความต้องการของระบบงานมาออกแบบโดยใช้ OONIAM และ UML มาช่วยออกแบบเมื่อเสร็จจากขั้นตอนการออกแบบ ก็จะนำแบบที่ได้มาสร้างคลาสก่อนที่จะสร้างคลาสของโปรแกรมได้นั้น จำเป็นที่จะต้องกำหนดค่าของระบบแวดล้อม การทำงานขึ้นก่อน

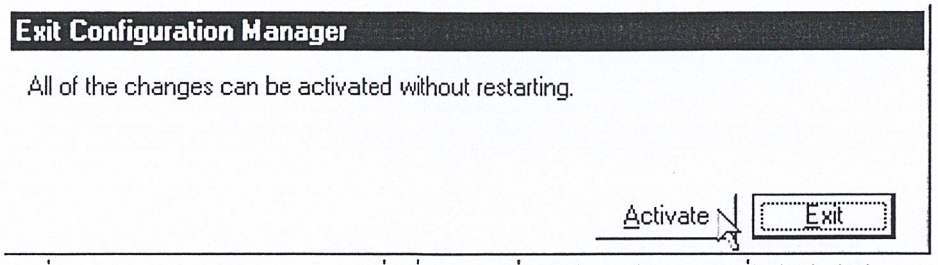
การสร้างและกำหนดต่างๆให้กับ Namespace และฐานข้อมูล คาเช่ นั้นจะมอง Namespace เสมือนข้อมูล เสมือนของตัวข้อมูลเท่านั้น กล่าวคือ Namespace จะทำหน้าที่เสมือนเป็นที่ที่เก็บ pointers ที่ชี้ไปยังพื้นที่ที่ใช้เก็บข้อมูลที่หนึ่ง หรือมากกว่า และ Namespace นั้นสามารถที่จะอ้างข้อมูลที่อยู่ใน ฐานข้อมูล หลายที่ด้วย Namespace เดียว Namespace นั้นยังยอมให้เราสร้างกลุ่มของข้อมูลที่มาจากหลาย ฐานข้อมูล ไม่ว่า

ฐานข้อมูล นั้นจะอยู่ที่ Computer เครื่องเดียวกันหรือไม่ ยกตัวอย่างเช่น มี ฐานข้อมูล ของลูกค้าและ ฐานข้อมูล ของผู้ขายซึ่งอยู่ต่าง ฐานข้อมูล กันแต่เราสามารถที่จะสร้าง Namespace โดยที่จะอ้างถึง ข้อมูลโดยไม่จำเป็นต้องสนใจเลยว่าข้อมูลที่เราอ้างถึงนั้นอยู่ใน ฐานข้อมูล ใดเพราะเราจะใช้ข้อมูลโดยอ้างผ่าน Namespace และกลไกของ Namespace จะทำหน้าที่นำข้อมูลมาให้เราเอง เมื่อทราบประโยชน์ของ Namespace พอสมควรแล้วและต่อไปนี้คือขั้นตอนของการสร้าง Namespace เริ่มขั้น ตอนแรกด้วยการสร้าง ฐานข้อมูลและ Namespace โดยเลือก Configuration Manager จากเมนู Cache′ Cube และเลือกปุ่ม Wizards เพื่อ สร้างฐานข้อมูลและ Namespace



รูปที่ 5-1 แสดงการ Cache′ Configulation Manager และ แสดงการ Create a Namespace

จากรายการของข้อมูลที่ปรากฏดังรูปเพื่อสร้าง Namespace ใหม่ในกล่องข้อความให้ใส่ชื่อ Namespace ตามที่ต้องการ ขั้นตอนต่อมาคือการกำหนดว่าจะใช้ฐานข้อมูลที่มีอยู่แล้วอันไหนกับ Namespace นี้หรือจะสร้างฐานข้อมูลใหม่ ถ้าเลือกที่จะสร้างฐานข้อมูลใหม่จะมี Database wizard สำหรับสร้างฐานข้อมูลใหม่ และใส่ชื่อฐานข้อมูล ลงไปในกล่องข้อความจะเห็นว่าการกำหนดชื่อของ Namespace และฐานข้อมูลจะมีชื่อคล้ายกันแต่ชื่อของฐานข้อมูลควรที่จะมีอะไรเป็นตัวบอกว่าเป็นฐานข้อมูล หรือ Namespace ถ้าเป็นการกำหนดฐานข้อมูลขึ้นมาใหม่ จะมี Wizard การเลือก Directory ที่ใช้เก็บฐานข้อมูล ขั้นตอนสุดท้ายจะตั้งค่าว่าฐานข้อมูลจะมีขนาดใหญ่เท่าไร แต่ไม่จำเป็นที่จะต้องกังวลกับค่าที่ตั้งไว้มากนัก เพราะฐานข้อมูลของ คาเช่ สามารถขยายได้ตามต้องการ เลือก OK เพื่อบันทึกผลการเปลี่ยนแปลง



รูปที่ 5-2 แสดงการเลือก Activate เพื่อที่สามารถที่จะใช้ฐานข้อมูลใหม่ที่ได้สร้างได้ทันที

5.1.2 สร้างคลาส ในระบบงาน

5.1.2.1 ชนิดคลาสของ คาเช่ มีด้วยกันหลายชนิดดังต่อไปนี้คือ

ชนิดคลาสของ คาเช่ มีด้วยกันหลายชนิดดังต่อไปนี้คือ

Registered Classes เป็นคลาสที่ประกอบไปด้วยฟังก์ชันพื้นฐานต่างๆ ที่ใช้ในการพฤติกรรมของมันที่มีต่อหน่วยความจำ

Persistent Classes เป็นคลาสที่มีคุณสมบัติครบและจะมี OID เป็นของตัวเอง จะมีฟังก์ชันพื้นฐานต่างๆ มาให้ เช่นระบบจะจองหน่วยความจำ เพื่อใช้เป็นพื้นที่เก็บค่าของ Object ให้สนับสนุน Swizzling โดยอัตโนมัติ สนับสนุน Polymorphism

Embeddable Classes นั้นจะมีคุณสมบัติคือ object สามารถที่จะถูกนำมาเก็บไว้ในหน่วยความจำได้อย่างอิสระ แต่การเก็บ object ของ Embeddable Classes นั้นมันจะถูกเก็บไว้ใน persistent object กล่าวคือ Embeddable object นั้นเปรียบเสมือนเป็นตัวแปรตัวหนึ่งของ persistent object นั่นเองEmbedded objects จะมีความแตกต่างกัน ระหว่างตอนที่อยู่ในหน่วยความจำ กับขณะที่ถูกบันทึกบนแผ่นดิสก์ ในหน่วยความจำนั้น embedded object จะถูกแยกออกมาเป็น object ตัวหนึ่งและจะใช้ OREF ซึ่งเปรียบเสมือนเป็น pointer ที่ชี้ไปยัง object นั้น บนแผ่นดิสก์ ตัวแปรชนิด embedded object ถูกเก็บเสมือนว่าเป็นส่วนหนึ่งของ object อีกตัวหนึ่ง และมันจะไม่มี OID เป็นของตัวเองดังนั้นทำให้ object ชนิดนี้จึงไม่สามารถที่จะถูกอ้างถึงได้โดย object ตัวอื่นเลยนอกจาก object ตัวที่เป็นเจ้าของ embedded object นั้นเท่านั้น ตามรูปที่แสดงดังด้านล่างนั้นแสดงให้เห็นวิธีการเก็บและอ้างถึง embeddable object ในหน่วยความจำและบนแผ่นดิสก์

Non-Registered Classes จะเป็นคลาสที่ไม่ได้กำหนด ฟังก์ชันพื้นฐานมาให้เท่าไรนักดังนั้นการที่จะสร้างคลาสชนิดนี้ขึ้น ต้องการการเขียนโปรแกรมค่อนข้างมากเหมาะสำหรับผู้ที่มีความชำนาญสูงต้องการที่จะสร้างคลาสที่มีคุณสมบัติตามที่ต้องการ และ Non-Registered Classes จะมีขีดจำกัดดังนี้คือ ระบบจะไม่จองหน่วยความจำ เพื่อใช้เป็นพื้นที่เก็บค่าของ object ให้ ไม่สนับสนุน Swizzling โดยอัตโนมัติ ไม่สนับสนุน Polymorphism

Data Type Classes จะเป็นคลาสจำพวกตัวแปรพื้นฐาน เช่น string,Integer

5.1.2.2 Property

Property จะเปรียบแล้วกับ Relational Database ก็จะเป็นการกำหนด แอตทริบิวต์ โดยที่สามารถกำหนดว่าใน คลาสของ คาเช่ นั้นประกอบด้วย แอตทริบิวต์ อะไรบ้าง ชื่ออะไร เป็นชนิดใดและมีคุณสมบัติอย่างไรบ้าง แอตทริบิวต์ สามารถที่จะมีชนิดได้เป็นไปตาม ชนิดข้อมูลที่ คาเช่ ได้จัดเตรียมไว้ หรือ

มีชนิดเป็น คลาสที่ได้กำหนดไว้ก่อนหน้าแล้ว และ ตัวอย่างชนิดข้อมูลพื้นฐานที่ คาเช่ ได้จัดเตรียมไว้ได้แก่

%Binary	แทนข้อมูลชนิด binary
%Boolean	แทนข้อมูลชนิด boolean
%Currency	แทนข้อมูลชนิด currency
%Date	แทนข้อมูลชนิด date
%Float	แทนข้อมูลชนิด เลขทศนิยม(floating point)
%Integer	แทนข้อมูลชนิด เลขจำนวนเต็ม(integer)
%List	แทนข้อมูลชนิด ลิงค์ลิส
%Name	แทนข้อมูลชนิด ที่ใช้เก็บในรูปแบบ “นามสกุล,ชื่อ”
%Numeric	แทนข้อมูลชนิด numeric ที่มีความแม่นยำสูง
%Status	แทนข้อมูลชนิด ที่ใช้เก็บสถานะของ error
%String	แทนข้อมูลชนิด ข้อความ
%Time	แทนข้อมูลชนิด time
%TimeStamp	แทนข้อมูลชนิด %Time กับ %Date

ตารางที่ 5-1 ชื่อชนิดและความหมายตัวแปรที่ใช้ในคาเช่

คุณลักษณะของตัวแปร Property มีความหมายดังต่อไปนี้

Private ระบุว่าเป็นตัวแปรที่ใช้ส่วนตัวตามหลัก Object oriented โปรแกรมมิ่ง ทั่วไป ถ้าไม่กำหนดจะเป็นแบบ public โดยปริยาย และถ้าคลาสนี้ถูกสืบทอดคุณสมบัติไปคลาสลูกไม่สามารถที่จะเรียกใช้ได้

Calculated ระบุว่าเป็นตัวแปรที่ไม่มีการจองหน่วยความจำเพื่อใช้เก็บค่าของตัวแปรชนิดนี้ และจะมีชีวิตอยู่เพียงชั่วขณะ ถ้าไม่กำหนดจะเป็นแบบ not calculated โดยปริยาย และถ้าคลาสนี้ถูกสืบทอดคุณสมบัติไปคลาสลูกไม่สามารถที่จะเรียกใช้ได้

Required ระบุว่าเป็นตัวแปร จะต้องมีค่าอยู่ที่ตัวมันก่อนที่จะถูกบันทึกลงดิสก์

Final ระบุว่าเป็นตัวแปรคลาสลูกไม่สามารถที่จะเรียกใช้ได้

5.1.2.3 เมธอด

เมธอด ที่ให้นักพัฒนาระบบเขียนโปรแกรมเหมือนหลักการการเขียนโปรแกรม ตามหลักการ Object Oriented Programming โดยทั่วไป Method มีโดยทั่วไป ชนิดดังต่อไปนี้

Code Methods

Code Methods เป็น method ที่เขียนขึ้นด้วย Cache′ object script ยกตัวอย่างเช่น method ของคลาส Dog

```
CLASS Dog {
        SUPER = %Persistent;
        PERSISTENT;
        METHOD Speak() {
        CODE = { QUIT "Woof, Woof" }
        }}
```

รูปที่ 5-3 แสดงคลาส Dog

Expression method

Expression method จะให้ผลการทำงานคืนค่ามาที่ตัวแปร EXPREESION ตัวอย่างการเขียน Expression method

```
METHOD Speak() {
        EXPRESSION = "Woof, Woof";
}
```

รูปที่ 5-4 แสดงโค้ดการ Expression method

สมมติให้ dog เป็น object ชนิด dog เมื่อ ประมวลผลคำสั่งจะได้

WRITE dog.Speak() ➔ WRITE "Woof, Woof"

5.1.2.4 Query

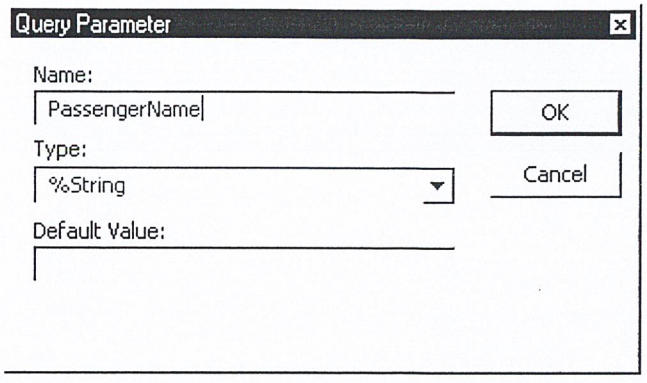
Query จัดเป็นส่วนประกอบที่สำคัญ ของคลาสของ คาเช่ มีไว้เพื่อค้นหาข้อมูลที่ต้องการในฐานข้อมูล

5.1.3 สร้าง Property ในแต่ละคลาส

การสร้าง Property ในแต่ละคลาส สามารถทำได้โดยเลือก New Property และกำหนดชนิดของ Property ตาม ชนิดข้อมูลพื้นฐานของ คาเช่' หรือมีชนิดตามคลาสที่เราได้สร้างขึ้นก่อนหน้านี้และทำการคุณสมบัติของ Property เช่น Require จะหมายความว่าคลาสนั้นจะต้องมีค่าของ Property นี้จึงจะทำการบันลงไปในระบบฐานข้อมูลได้เป็นต้น

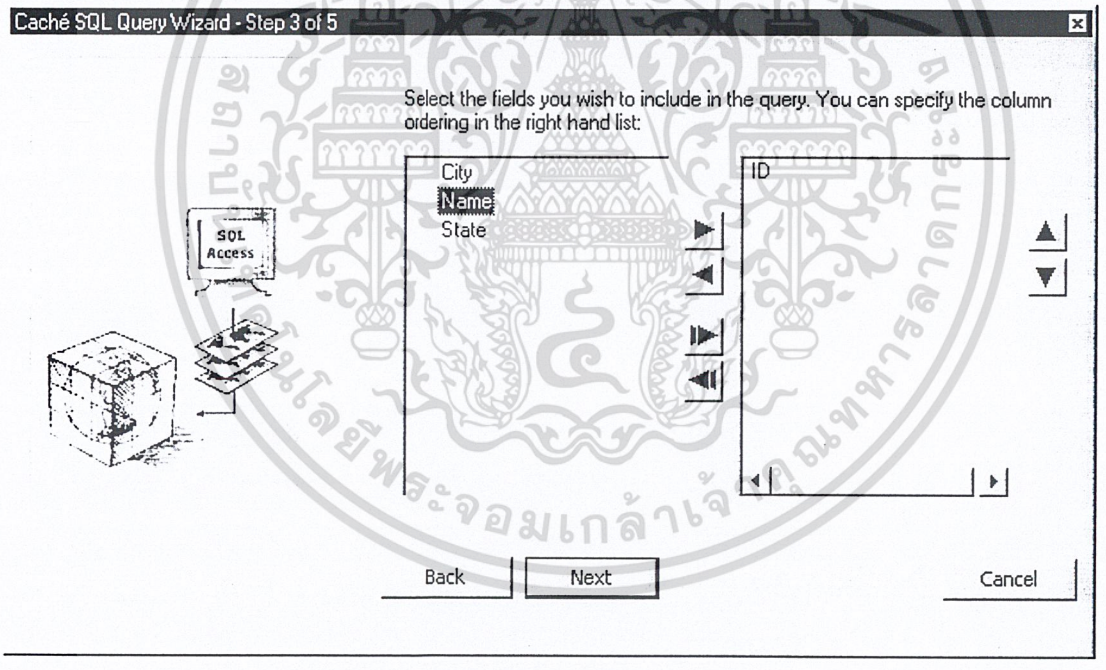
5.4 สร้างเมธอด และเขียนโปรแกรมลงในเมธอดนั้นๆ

5.5 สร้าง Query การ สร้างQuery สามารถใช้เครื่องมือช่วยและจะมีลำดับการทำงาน ดังต่อไปนี้ เลือกที่แถบเครื่องมือ New Query กำหนดชื่อและชนิดของตัวแปรแต่ละตัวที่ผ่านค่ามาเพื่อใช้ Query โดยจำนวนตัวแปรสามารถกำหนดได้โดยเลือกจากคุณสมบัติ ของ Query



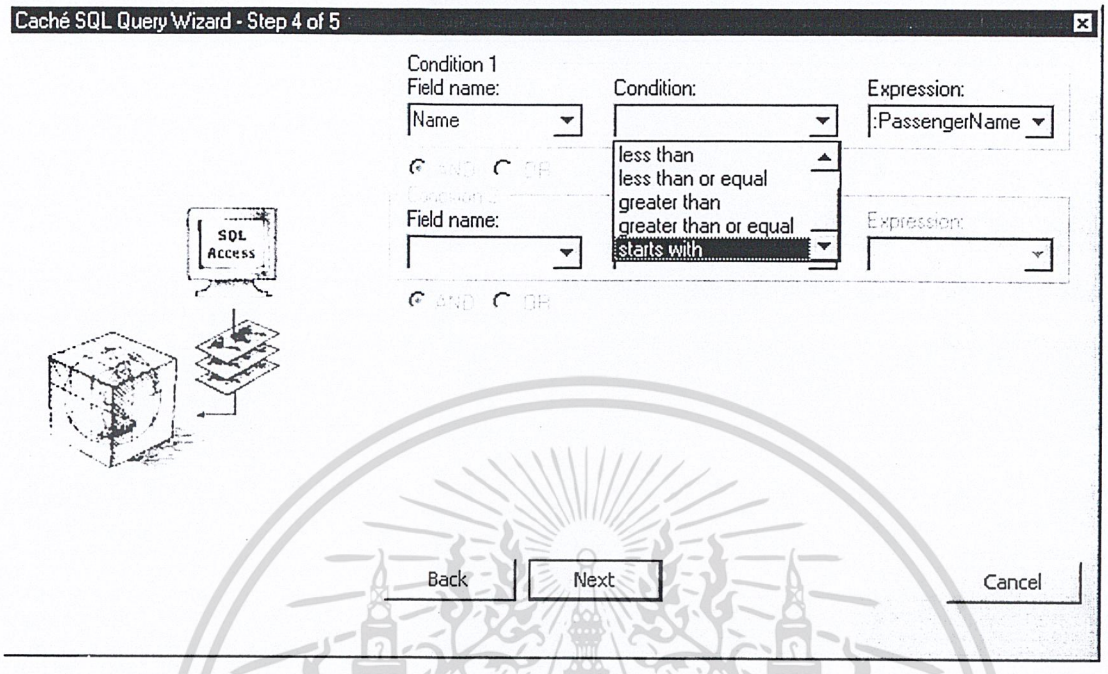
รูปที่ 5-5 แสดงการกำหนดชื่อและชนิดตัวแปรที่ส่งข้ามาเพื่อใช้ Query

ระบุว่าการ Query นั้นต้องการอะไรบ้าง(คำสั่ง Select) โดยที่ด้านขวานั้นจะแสดง Property ทั้งหมดของคลาส คลาสนั้นและจะมี ID เพิ่มเข้ามา โดยที่ ID ระบบจะสร้างให้เราเองโดยอัตโนมัต เมื่อเราทำการสร้างออบเจ็กต์ และเมื่อเราต้องการจะโหลดออบเจ็กต์ บันทึกการเปลี่ยนแปลง หรือทำการลบออบเจ็กต์เราต้องใช้ ID เสมอ อีกประการหนึ่งก็คือ ระบบจะสร้าง ID ให้เฉพาะ คลาสชนิด Persistent เท่านั้น



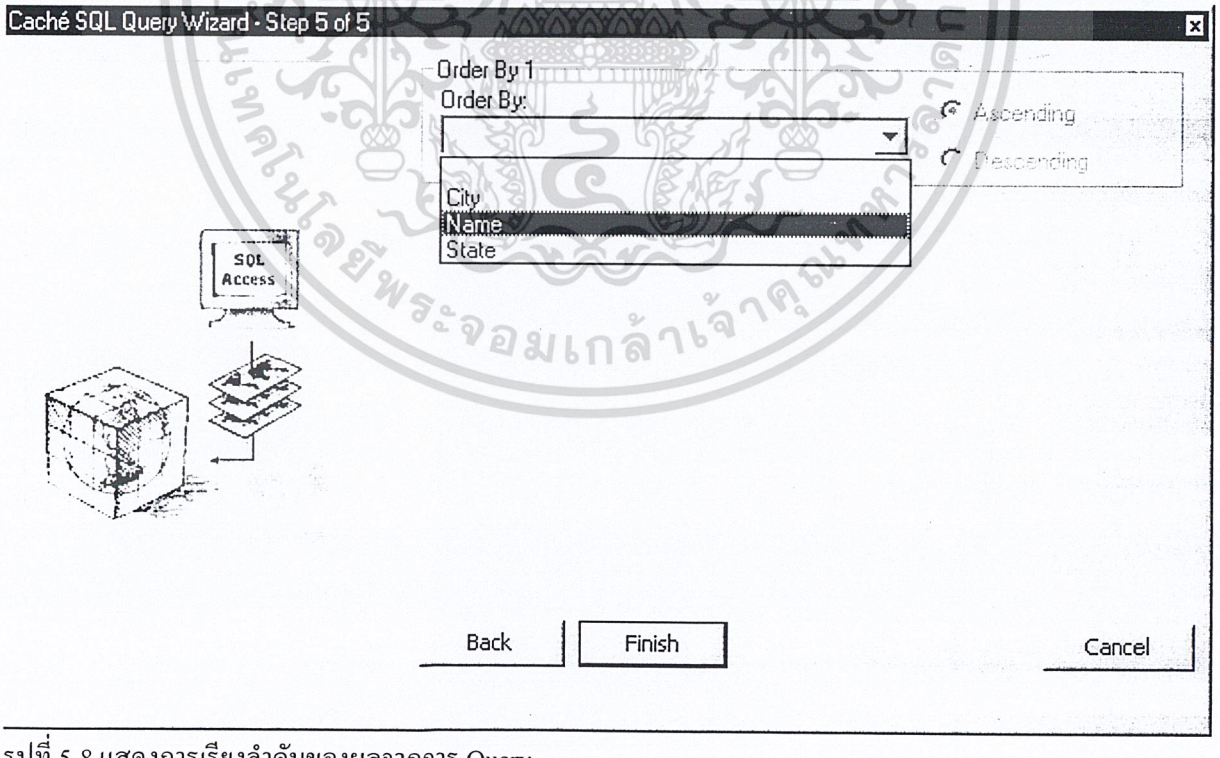
รูปที่ 5-6 แสดงการเลือก แอตทริบิวต์ ของคลาสที่ต้องการ

กำหนดเงื่อนไขการเปรียบเทียบกับค่าที่ส่งเข้ามา กับ แอตทริบิวต์ ของออบเจ็กต์ หรือค่าคงที่กับ แอตทริบิวต์ ของออบเจ็กต์ จริงๆ แล้วก็คือก็คือส่วนของเงื่อนไขใน Where ในภาษา SQL นั่นเอง



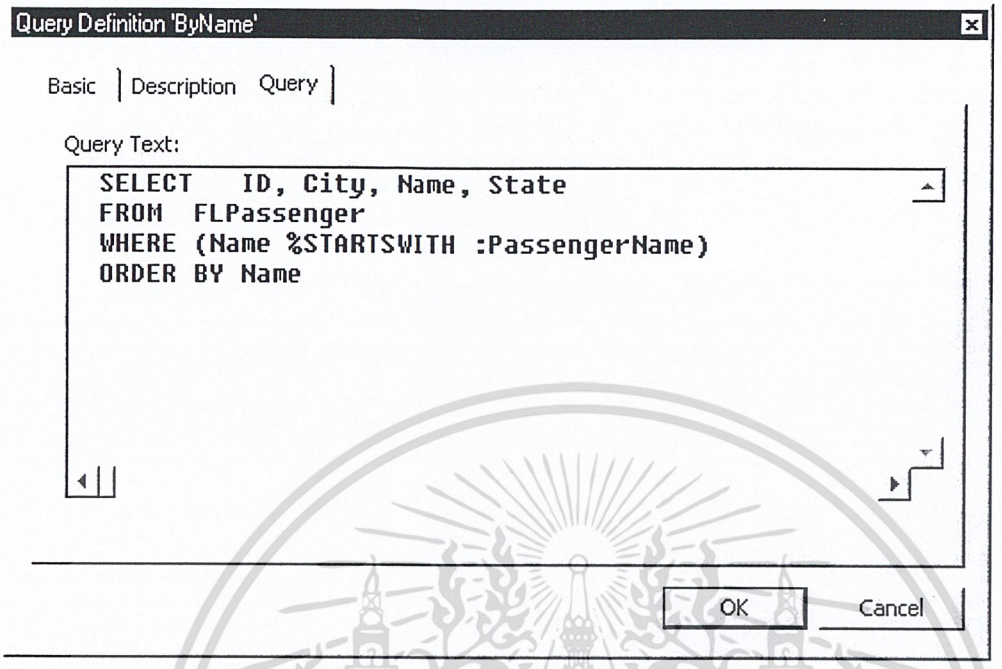
รูปที่ 5-7 แสดงการกำหนดเงื่อนไขของการ Query Object

เลือกการเรียงลำดับว่าจะเรียงตาม แอตทริบิวต์ อะไร และเรียงจากมากไปน้อยหรือจากน้อยไปมาก



รูปที่ 5-8 แสดงการเรียงลำดับของผลจากการ Query

เมื่อกด Finish แล้วจะเห็นว่าเมื่อดับเบิลคลิกที่ Query จะได้คำสั่ง SQL ที่เราได้ใช้ Wizard สร้างดังด้านบน ซึ่งสามารถแก้ไขและเปลี่ยนแปลงจากจุดนี้ได้เช่นกัน



รูปที่ 5-9 แสดงโค้ด SQL จากการ Query wizard

5.2 การเขียนโปรแกรมโดยใช้ Cache′ ObjectScript

ระบบจัดการฐานข้อมูล คาเช่นั้นได้มี Cache′ ObjectScript เพื่อให้นักพัฒนาโปรแกรมได้ใช้เขียนโปรแกรมติดต่อกับฐานข้อมูล ใช้สำหรับการประมวลผลข้อมูล หรืออาจนำมาใช้ในการพัฒนาโปรแกรมที่ทำงานในสภาพแวดล้อมของคาเช่เพื่อให้เข้าใจ โปรดดูตัวอย่างรูปด้านล่าง

```
SAMPLES>do ^RightTriangle
Compute the area and hypotenuse of a right triangle
given the lengths of its two sides.
First, choose a unit of measurement.
(i)nches, (f)eet, (m)iles, (c)entimeters, m(e)ters, (k)ilometers: i
 Length of side 1: 3  Accepted.
Length of side 2: 4  Accepted.
 The area of this triangle is 6.00 square inches.
 The hypotenuse is 5.00 inches.
SAMPLES>
```

รูปที่ 5-10 แสดงการเรียกใช้งานติดต่อกับฐานข้อมูล

จากรูปด้านบนแสดง command line ของ คาเช่ หากจะเปรียบเทียบก็เหมือนกับ Dos prompt ของระบบปฏิบัติการของวินโดว์ แต่จะต่างกัน ก็ตรง Command line นี้จะใช้ประมวลผล Cache′ ObjectScript

ที่คำสั่ง “do” เป็นคำสั่งของ Cache′ ObjectScript ที่สั่งให้เรียกแฟ้มข้อมูล RightTriangle มาประมวลผล และเครื่องหมาย “^” คาเช่ จะใช้แทน extrinsic functions ก็คือการทำงานที่ผู้ใช้ได้เขียนการประมวลผลขึ้นเอง และยังมีฟังก์ชัน อีกประเภทหนึ่งคือ ฟังก์ชันของระบบ(intrinsic functions) จะขึ้นต้นด้วย “$$ “ จะเห็นว่าเมื่อเรียก Cache′ ObjectScript Procedure จะทำงาน โดยการแสดงผลข้อความออกมาเพื่อให้ผู้ใช้เข้าใจว่า Cache′ ObjectScript นี้จะทำงานอย่างไร จากนั้นผู้ใช้จะทำการป้อนค่าตัวเลข สองค่าเข้าไปจากนั้น ก็ทำการประมวลผล หาพื้นที่ของสามเหลี่ยม และหาด้านตรงข้ามมุมฉากของสามเหลี่ยมมุมฉาก จากข้างต้นเราจะเห็นคำสั่ง “do” ยังคงมีคำสั่งที่ใช้บ่อยๆ ได้แก่

คำสั่ง Write

```
SAMPLES>write "Hello World!"
Hello World!
SAMPLES>write !, "This", !, "Is", !, "A", !, "Multi-line", !, "Message!", !
This
Is
A
Multi-line
Message!
SAMPLES>write !, "This", ?10, "Is", ?20, "A", ?30, "Columnar", ?40, "Message!", !
This    Is      A       Columnar  Message!
SAMPLES>write 7.95 * 1.15
9.1425
```

รูปที่ 5-11 แสดงตัวอย่างแสดงการใช้งานคำสั่ง write

คำสั่ง Read

```
SAMPLES>read ?30, "Enter your name: ", n
                        Enter your name: Alexander
SAMPLES>read !, "You have 5 seconds to respond: ", x:5
You have 5 seconds to respond: I typed this in 5 seconds
SAMPLES>read !, "Type 5 characters and don't press <Return>: ", z#5
Type 5 characters and don't press <Return>: abcde
SAMPLES>
```

รูปที่ 5-12 แสดงตัวอย่างแสดงการใช้งานคำสั่ง read

คำสั่ง Set

มักจะใช้กับการกำหนดค่าคงที่ให้กับตัวแปร หรือค่าที่ได้จากการประมวล

Operator	Operation	Example (Result)
+	Addition. When used in front of an expression as a unary operator, it forces a numeric interpretation of the expression.	set x = 4 + 2 (x=6) set z = "546-FRJ" set y = +z (y=546)
-	Subtraction. When used in front of an expression as a unary operator, it forces a numeric interpretation of the expression, and reverses the sign.	set x = 4 - 2 (x=2) set z = "-10 degrees" set y = -z (y=10)
*	Multiplication.	set x = 4 * 2 (x=8)
/	Division.	set x = 5 / 2 (x=2.5)
**	Exponentiation.	set x = 5 ** 2 (x=25)
\	Integer Division. This divides the first operand by the second and produces the integer portion of the result, without the fractional portion.	set x = 5 \ 2 (x=2)
#	Modulo. This divides the first operand by the second and produces the integer remainder of the result only.	set x = 5 # 2 (x=1)

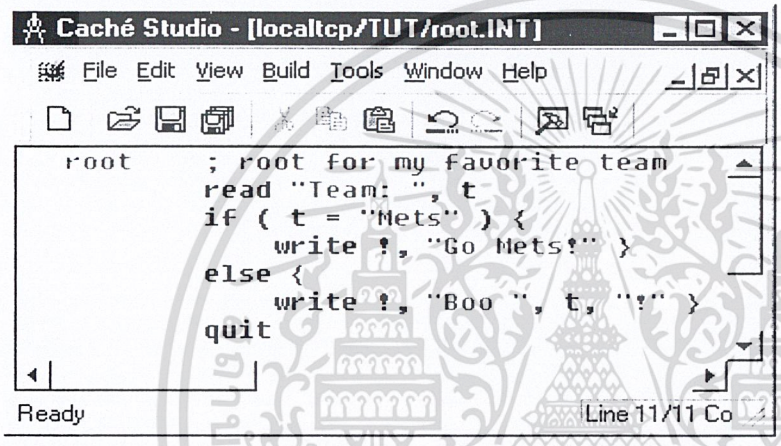
ตารางที่ 5-2 แสดงตัวอย่างแสดงการใช้งานคำสั่ง set

```
SAMPLES>do ^%SQROOT
Square root of: 100 is: 10
Square root of:
SAMPLES>set %X = 25  do INT^%SQROOT  write %Y
5
SAMPLES>
```

รูปที่ 5-13 แสดงตัวอย่างแสดงการใช้งานคำสั่ง do มีลักษณะเหมือนคำสั่ง call

```
if
SAMPLES>do ^root
Team: Mets
Go Mets!
SAMPLES>do ^root
Team: Blue Sox
Boo Blue Sox!
SAMPLES>
```

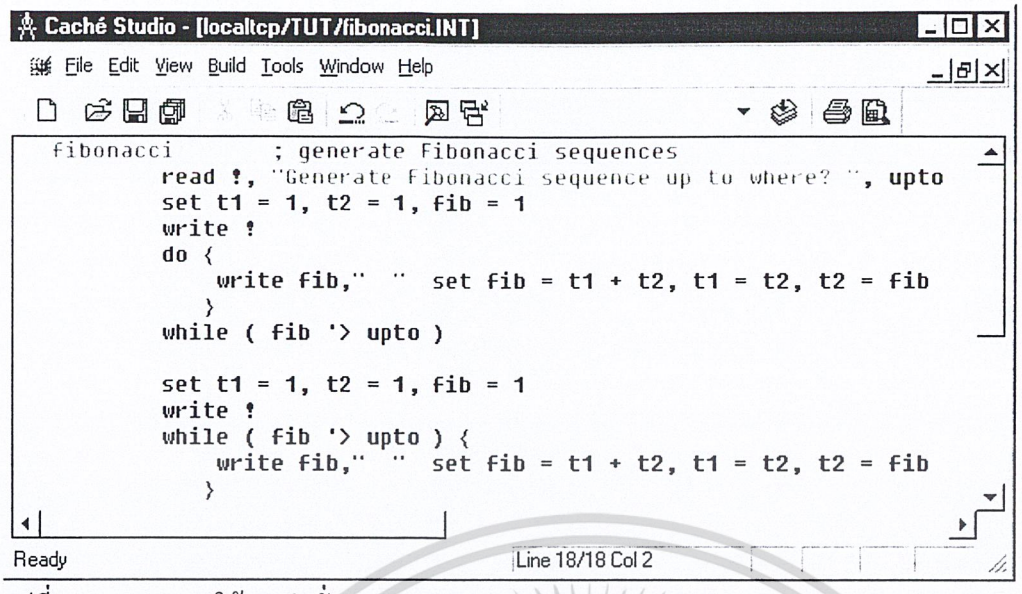
รูปที่ 5-14 แสดงการใช้งานคำสั่ง if



รูปที่ 5-15 แสดงการใช้งานคำสั่ง if

```
SAMPLES>do ^fibonacci
Generate Fibonacci sequence up to where? 100
1 2 3 5 8 13 21 34 55 89
1 2 3 5 8 13 21 34 55 89
SAMPLES>
```

รูปที่ 5-16 แสดงลำดับ Fibonacci โดยเป็นผลจากการทำงาน



รูปที่ 5-17 แสดงการใช้งานคำสั่ง do/while

```
SAMPLES>write $length("how long is this?")
17
SAMPLES>write $extract("there's gold in them hills", 9, 12)
gold
SAMPLES>write $find("there's gold in them hills", "gold")
13
SAMPLES>write $piece("slice of pizza", " ", 3)
pizza
SAMPLES>write $justify(3.1415, 10, 3)
    3.142
SAMPLES>
```

รูปที่ 5-18 แสดงการใช้งานสตริงฟังก์ชันโดยทั่วไป

คำสั่ง $length (."string long") เป็นคำสั่งที่ใช้หาความยาวของสตริง

คำสั่ง $extract (."string ",begin,end) เป็นคำสั่งที่ใช้ในการนำค่าสตริงย่อย โดยเริ่มจาก begin จนถึง end

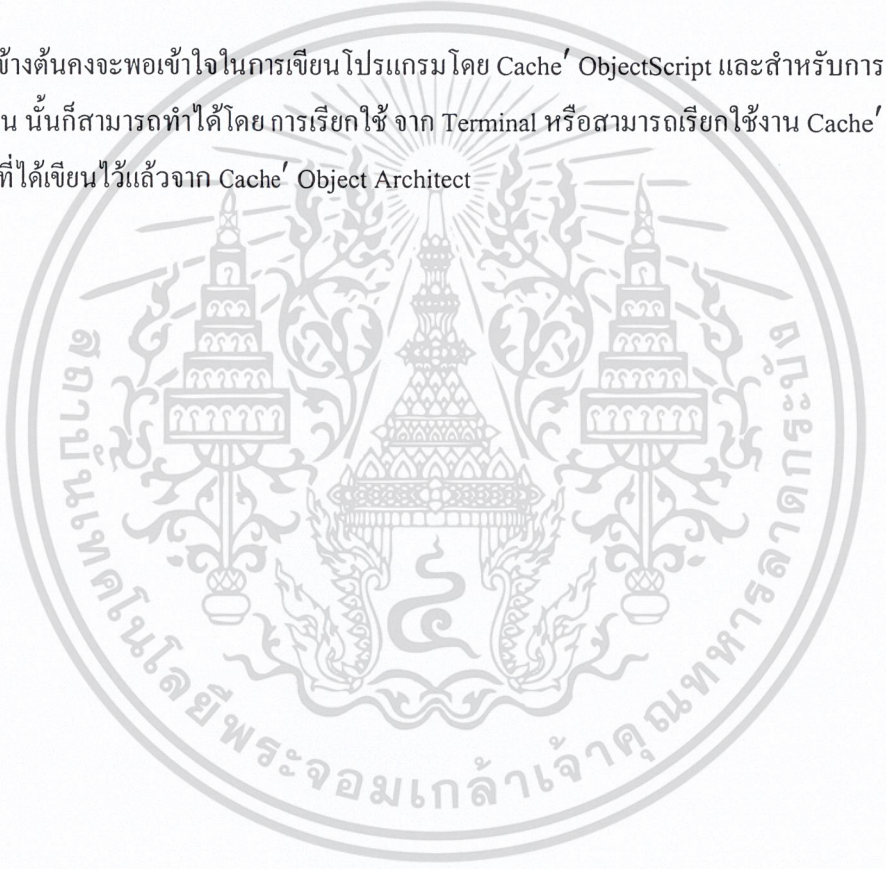
คำสั่ง $find("string","find") เป็นคำสั่งที่ใช้ในการค้นหาสตริงย่อย โดยจะคืนค่า Index character ของสตริงที่ต้องการค้นหาตัวสุดท้ายกับมา

คำสั่ง $justify(3.1415, 10, 3) เป็นคำสั่งที่ใช้ในการปัดเศษตัวเลขโดย ตัวแรกคือเลขที่ต้องการปัดเศษตัวที่สองคือบอกว่าเป็นเลขฐานอะไร และตัวที่สามคือจำนวนจุดทศนิยม

Operator	Operation	Examples of true conditions
=	Equals.	if (2 = 2) { write "equal" }
>	Greater than.	if (4 > 3) { write "greater than" }
<	Less than.	if (3 < 4) { write "less than" }

ตาราง 5-3 แสดง relational operator ของ คาเช่

จากข้างต้นคงจะพอเข้าใจในการเขียนโปรแกรมโดย Cache′ ObjectScript และสำหรับการประยุกต์ใช้งาน นั้นก็สามารถทำได้โดย การเรียกใช้ จาก Terminal หรือสามารถเรียกใช้งาน Cache′ ObjectScript ที่ได้เขียนไว้แล้วจาก Cache′ Object Architect



บทที่ 6
การปรับปรุงข้อจำกัดบางประการของยูเอ็มแอล

เครื่องมือช่วยออกแบบเชิงวัตถุ(Object Oriented Modeling)

เนื่องจากในการออกแบบฐานข้อมูลเชิงวัตถุ ต้องใช้เครื่องมือในการออกแบบเชิงวัตถุ ซึ่งในที่นี้ผู้จัดทำได้ใช้ UML และ OONIAM ช่วยในการออกแบบ จึงได้ทำการศึกษาและทดลองใช้เครื่องมือดู ว่าการนำเครื่องมือช่วยออกแบบ นั้นมีความเหมาะสมและสมบูรณ์ หรือไม่ที่จะนำมาใช้กับ ฐานข้อมูลเชิงวัตถุ

คุณสมบัติที่ UML ไม่สนับสนุนการออกแบบฐานข้อมูลเชิงวัตถุ และข้อจำกัดบางประการในการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ

ปัญหา 6.1 เนื่องจากคุณสมบัติของ คลาสในฐานข้อมูลเชิงวัตถุนั้นมีสองประเภทหลักๆ คือ เอมเบตคลาส (Embeded class) กับเพอร์ซิแทนคลาส (Persistance class) จากคลาสไดอะแกรม(Class diagram) ไม่สามารถแยกได้ว่า คลาสใดเป็น แบบ Embeded หรือ Persistance class

รูปที่ 6-1 แสดงคลาสไดอะแกรมกรณีที่ต้องการเป็น Embeded Class

ปัญหา 6.2 ในการออกแบบ จำเป็น ที่จะต้องมีทั้ง Class diagram ของฐานข้อมูลและ Class diagram ของ Application ไม่สามารถแยกได้ว่า คลาสใดเป็น Database class หรือ Application class

ทำการออกแบบคลาสในการกรณี Embeded Class ได้ อาจทำการออกแบบดังรูป 6-1 หรืออาจใช้ Stereotype ก็ได้ โดยกำหนดเป็น Entity Class หรือคลาสชนิดใดก็ได้ตามกำหนด เช่น Database Class หรือ Application Class ก็ได้

ปัญหา 6.3 Attribute ของ Database class บางตัวไม่สามารถนำมาแสดงได้ จะต้องมีการแปลงรูปแบบก่อนจึงจะจัดเก็บได้ หรืออาจจะไม่จำเป็นที่จะต้องนำมาแสดงในงานบางประเภท ไม่สามารถแยกแยะได้ว่าต้องนำ Attribute ใดบ้างมาแสดงผล ในโปรแกรมบ้าง

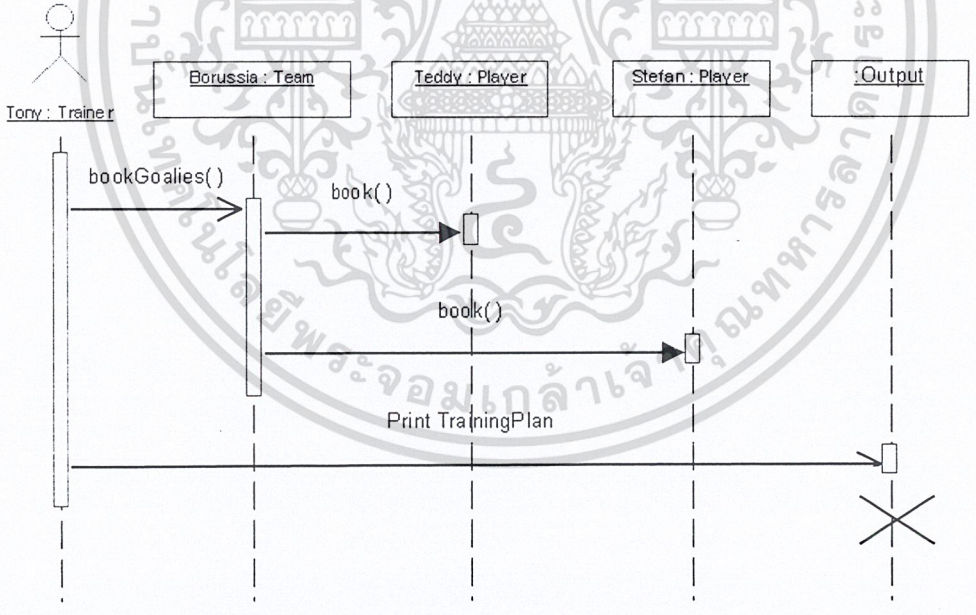
โดยหากใช้ โมเดลของ OONIAM ก็จะสามารถแก้ปัญหา 6.1-6.3 ได้คือ สามารถบอกได้ว่า คลาสไหนเป็น Persistance class คลาสไหนควรเป็น Embeded โดยหากต้องการให้เป็น Embeded Class ก็

กำหนดคลาสไว้ภายใต้ Sub Schema โดยในส่วนของ Main Schema ก็กำหนดเป็นส่วนของ Persistance Class ไปนั้นเอง (จากบทที่ 4 การสร้างและออกแบบ OONIAM)

ปัญหา 6.4 ในการออกแบบ Database class ตามหลัก Object Oriented Database มักมีคลาสที่ สืบทอดคุณสมบัติจาก คลาสอื่น ๆ ต่างจาก การออกแบบฐานข้อมูลแบบ รีเลชั่นแนล ที่จะมีตารางเท่าใดก็จะนำไปสร้างเท่านั้น แต่การออกแบบฐานข้อมูลเชิงวัตถุ อาจจะมีบางคลาสไม่จำเป็นที่จะต้อง นำมาใช้ในโปรแกรม จึงไม่อาจจะทราบได้ว่า ต้องนำคลาสใดบ้างมาพัฒนาระบบ

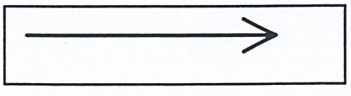
ปัญหา 6.5 ภายใน Class Diagram นั้นไม่สามารถบอกได้ว่าส่วนใดเป็นกฎข้อบังคับภายในฐานข้อมูล ตัวอย่างเช่น คลาสของการยืมหนังสือหากมีกฎ ห้ามนักศึกษายืมหนังสือเกินคนหละ 5 เล่ม กฎในส่วนนี้ไม่สามารถบอกได้ภายในคลาสไดอะแกรม แต่หากแก้ไขโดยให้User Define ไปที่คลาสว่า ห้ามเกิน 5 เล่ม เป็นการบอกกฎกำหนดกฎไว้ที่ตัวคลาสไดอะแกรมเลยก็ได้ (หากใช้ State Diagram เป็นตัวช่วยในการบอกว่าส่วนใดเป็นกฎข้อบังคับก็ได้)

ปัญหา 6.6 ในการออกแบบสร้าง Sequence Diagram นั้นการติดต่อกันของ Object ภายใน Class นั้นจะต้องทำการเรียกผ่าน Operation แต่ภายในโมเดล UML นั้นไม่ได้ทำการกำหนดว่าการติดต่อระหว่าง Object นั้นเป็นแบบใดบ้าง (แต่สามารถทำการกำหนดความสัมพันธ์ได้ โดยใน โปรแกรมของ Rational Rose 2000 ทำการกำหนด สัญลักษณ์ เหล่านี้ได้) โดยหาก Rational Rose นั้นไม่ใช้ยูเอ็มแอลแต่เป็นเครื่องมือที่สนับสนุนการออกแบบและพัฒนาระบบโดยรองรับมาตรฐานของยูเอ็มแอล



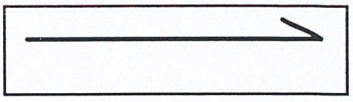
รูปที่ 6-2 แสดง Sequence Diagrams โดยสร้างจาก Rational Rose

โดยที่ Message นั้นเป็นการที่ส่งจาก Object หนึ่งไปยังอีก Object หนึ่งโดยเป็นการแทนโดยใช้หัวลูกศร ซึ่งมีชื่อที่มาจาก Operation ของ Target Object



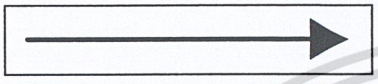
รูป 6-3 แสดง synchronous Message

โดยหัวลูกศรที่มีหัวแบบสมบูรณ์แสดงเป็นแบบ synchronous Message จากตัวอย่างข้างบน bookGoalies เป็น synchronous Message ที่ Trainer ของ goalkeepers ที่ส่งไปยัง team object



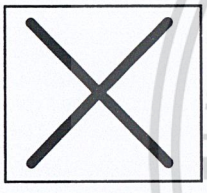
รูป 6-4 แสดง Asynchronous message // block

ส่วน Print TrainPlan เป็น Asynchronous message เนื่องจาก Trainer ต้องการทำงานต่อในขณะที่ output process บน printer



รูป 6-5 แสดง Procedure calls

ลูกศรแบบเต็มแสดงเป็นแบบ Local message ที่เป็น Procedure call เช่น book message ที่ term object ส่งไปยัง 2 player ซึ่งเป็น goalkeeper

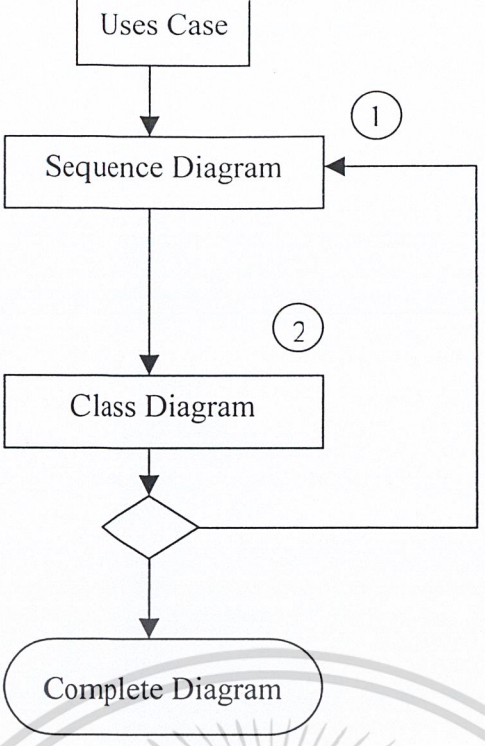


รูป 6-6 แสดง Object Deactivated

โดยสัญลักษณ์กากบาทแสดงเมื่อสิ้นสุดการใช้งาน หรือทำการเลิกใช้ Object นั้นแล้ว

ปัญหา 6.7 การออกแบบ Sequence นั้นไม่มีความชัดเจนว่าจะทำการสร้าง Sequence Diagram โดยดูจาก Use Case หรือดูจาก Class Diagram โดยอาจทำการแก้ไขได้ดังนี้

การออกแบบนั้นให้แบ่งการออกแบบ เป็นรอบๆ โดยในรอบแรกนั้น Sequence Diagram จะทำการออกแบบตามแบบ Uses case Diagram โดยมองหน้าที่และลำดับเหตุการณ์การทำงานตามที่ออกแบบใน Use case Diagram หลังจากนั้นทำการออกแบบ Class Diagram โดยนำเอาชื่อการทำงานใน Sequence Diagram ที่ได้รับการออกแบบมาในรอบแรกมาเปลี่ยนให้อยู่ในชื่อของ Operation ของใน Class Diagram และนำชื่อของสิ่งที่ใช้ใน Sequence Diagram มาเป็น Attributes ใน Class



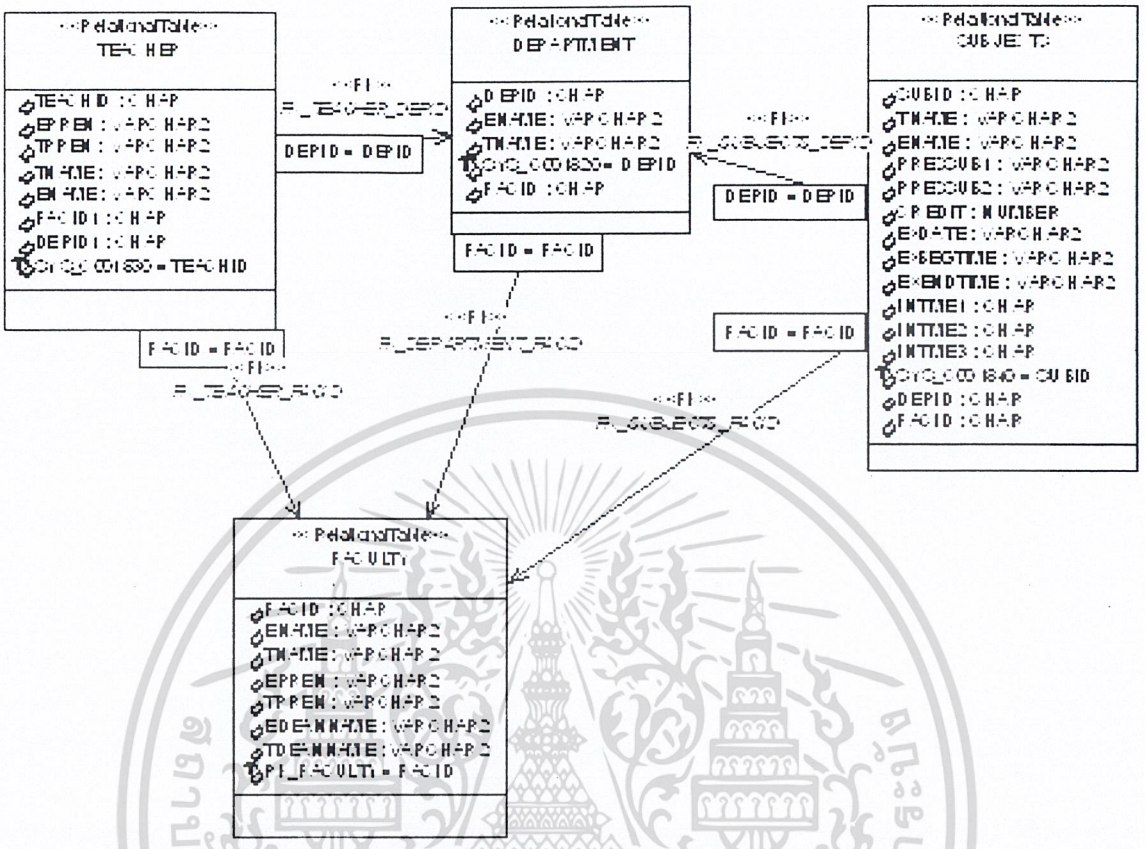
รูปที่ 6-7 การวนลูปเพื่อแก้ปัญหาการสร้าง Sequence Diagram

ในขั้นตอนถัดไป ให้ทำการ Refinement Sequence Diagram อีกครั้งหนึ่ง โดยทำการเปลี่ยนแปลงชื่อและหน้าที่การทำงานของลำดับเหตุการณ์ใน Sequence Diagram ตามชื่อ Operation ภายใน Class Diagram โดยภายใน Sequence Diagram กลไกที่ทำให้ระบบมีกิจกรรมก็คือการส่ง Message นั้นการส่ง Message ระหว่างกันนั้น ทุกๆ Message ภายใน Sequence Diagram จะต้องทำการเรียกที่ผ่าน Operation เสมอ ดังนั้นชื่อและหน้าที่ที่ลำดับเหตุการณ์ภายใน Sequence Diagram ที่ไม่ใช้ Operation ให้ทำการเพิ่มชื่อนั้นให้เป็น Operation ภายใน Class Diagram นั้น และทำการแปลงชื่อและหน้าที่ภายในลำดับเหตุการณ์นั้น เป็น Operation อีกครั้งหนึ่ง

ปัญหา 6.8 โมเดลยูเอ็มแอลนั้นไม่มีไดอะแกรมได้ที่สามารถทำการแปลงจาก Application ไปเป็นไดอะแกรมที่สมบูรณ์ เหมือนครั้งที่มีไดอะแกรมแล้วให้โปรแกรมเมอร์สร้างเป็น Application ได้ คือไม่สามารถแปลงไปมาระหว่างกันได้ระหว่างไดอะแกรมและ Application

ปัญหา 6.9 ทำการออกแบบระบบฐานข้อมูลเชิงสัมพันธ์ Relational Model (รีเลชันนัลโมเดล) โดยใช้ยูเอ็มแอลนั้นซึ่งหากจะทำการออกแบบฐานข้อมูลแบบโดย ซึ่งสามารถทำการออกแบบได้โดยการสร้างคลาสไดอะแกรมโดยแสดงเพียงแค่ Attribute ภายในคลาสเท่านั้น โดยทำการกำหนดPrimary Key (คีย์หลัก) และ Foreign Key (ฟอร์รินคีย์) ภายในคลาสและความสัมพันธ์ระหว่างคลาส

โดยทำกาสร้างคลาสและกำหนดความสัมพันธ์แสดงได้โดย



รูปที่ 6-8 แสดงการออกแบบฐานข้อมูลด้วยเครื่องมือของ Rational Rose

เป็นการออกแบบฐานข้อมูลด้วยเครื่องมือของ Rational Rose ออกมาในรูปแบบของ UML ซึ่งสามารถที่จะเปลี่ยนจากรูปแบบของ UML ออกมาเป็นตารางเก็บในฐานข้อมูลได้ดังต่อไปนี้

ตาราง Faculty ประกอบด้วยรายละเอียดของคณะดังต่อไปนี้

ชื่อคอลัมน์	ประเภท	ขนาด	คำอธิบาย	หมายเหตุ
FACID	Char	2	รหัสคณะ	P.K.
ENAME	Varchar2	60	ชื่อคณะภาษาอังกฤษ	
TNAME	Varchar2	60	ชื่อคณะภาษาไทย	
EPREN	Varchar2	15	คำนำหน้าชื่อภาษาอังกฤษ	
TPREN	Varchar2	15	คำนำหน้าชื่อภาษาไทย	
eDeanNAME	Varchar2	50	ชื่อคณบดีภาษาอังกฤษ	
tDeanNAME	Varchar2	50	ชื่อคณบดีภาษาไทย	

ตารางที่ 6-1 แสดงรายละเอียดของข้อมูลในตาราง Faculty

ชื่อคอลัมน์	ประเภท	ขนาด	คำอธิบาย	หมายเหตุ
TECHID	Char	4	รหัสอาจารย์	P.K.
EPREN	Varchar2	15	คำนำหน้าชื่อภาษาอังกฤษ	
TPREN	Varchar2	15	คำนำหน้าชื่อภาษาไทย	
ENAME	Varchar2	60	ชื่ออาจารย์ภาษาอังกฤษ	
TNAME	Varchar2	60	ชื่ออาจารย์ภาษาไทย	
DEPID	Char	2	ภาควิชาที่สังกัด	F.K.
FACID	Char	2	คณะที่สังกัด	F.K.

ตารางที่ 6-2 แสดงรายละเอียดของข้อมูลในตาราง Teacher

ตาราง Department ประกอบด้วยรายละเอียดของภาควิชาดังต่อไปนี้

ชื่อคอลัมน์	ประเภท	ขนาด	คำอธิบาย	หมายเหตุ
DEPID	Char	2	รหัสภาควิชา	P.K.
FACID	Char	2	รหัสคณะ	P.K.
ENAME	Varchar2	60	ชื่อภาควิชาภาษาไทย	
TNAME	Varchar2	60	ชื่อภาควิชาภาษาอังกฤษ	

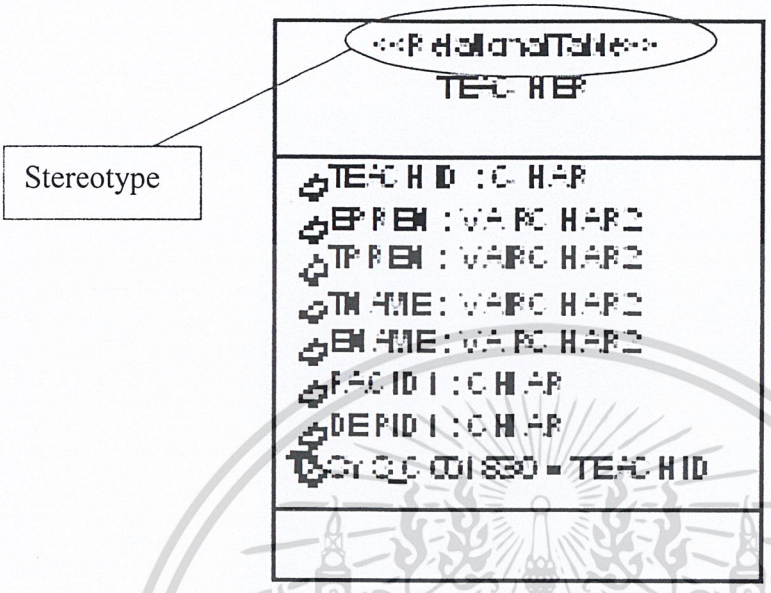
ตารางที่ 6-3 รายละเอียดของข้อมูลในตาราง Department

ตาราง Subject ประกอบด้วยรายละเอียดของแต่ละวิชาดังต่อไปนี้

ชื่อคอลัมน์	ประเภท	ขนาด	คำอธิบาย	หมายเหตุ
SUBID	Char	8	รหัสวิชา	P.K.
TNAME	Varchar2	80	ชื่อวิชาภาษาไทย	
ENAME	Varchar2	80	ชื่อวิชาภาษาอังกฤษ	
DEPID	Char	2	รหัสภาควิชา	F.K.
FACID	Char	2	รหัสคณะ	F.K.
PRESSUB1	Char	8	วิชาบังคับก่อน 1	
PRESSUB2	Char	8	วิชาบังคับก่อน 2	
CREDIT	Number	2	หน่วยกิต	
EXDATE	Varcahr2	10	วันที่สอบ	
ExBegTime	Varchar2	5	เวลาที่เริ่มสอบ	
ExEndTime	Varchar2	5	เวลาที่สิ้นสุดการสอบ	
InTerm1	Char	1	เปิดสอนภาคต้น	
InTerm2	Char	1	เปิดสอนภาคปลาย	
InTerm3	Char	1	เปิดสอนภาคฤดูร้อน	

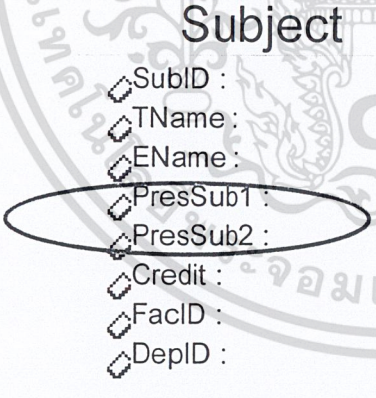
ตารางที่ 6-4 แสดงรายละเอียดของข้อมูลในตาราง Subject

จากรูป คลาสไดอะแกรม เป็นการสร้าง*สำหรับคลาสของฐานข้อมูล แล้วความแตกต่างของคลาสที่เป็นคลาสฐานข้อมูลกับแอพพลิเคชั่นคลาส* (Database class and Application class) ซึ่งสามารถแยกหรือระบุได้โดยการใช้ Stereotype เป็นตัวกำหนดได้

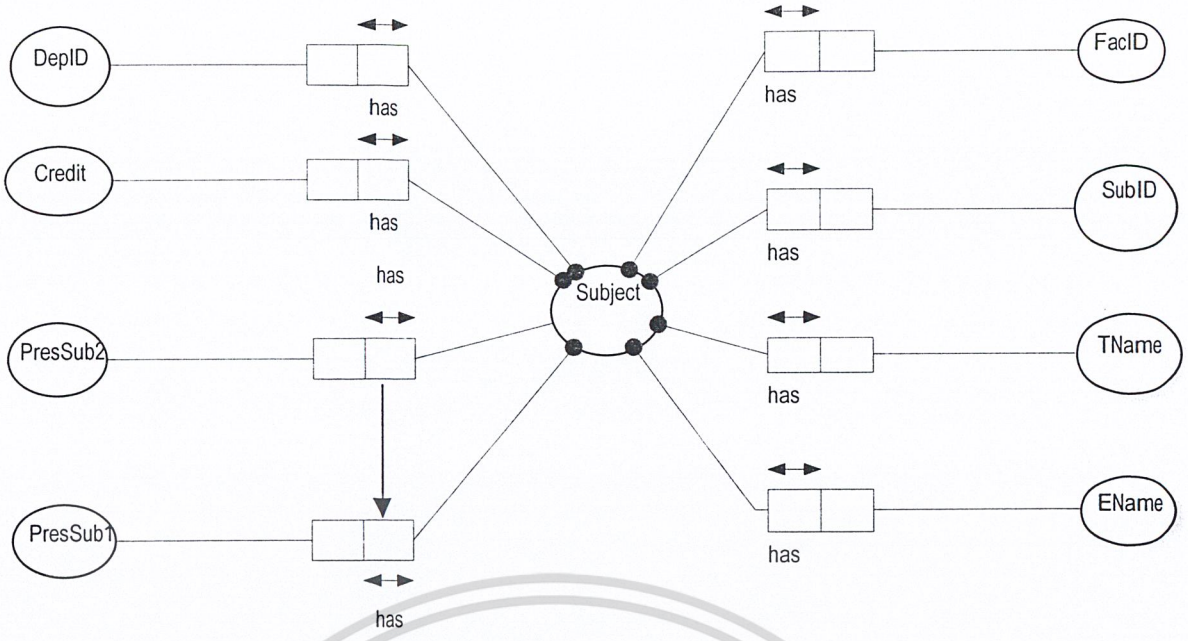


รูปที่ 6-9 แสดงคลาส TEACHER โดยระบุเป็นคลาสของฐานข้อมูล

โดยภายในโมเดลยูเอ็มแอลจะทำให้ทราบได้อย่างไรว่า Constrain ที่เก็บในฐานข้อมูลนั้นถูกเก็บอยู่ที่ใดภายในคลาสไดอะแกรม โดยที่จะสามารถรับทราบได้โดยพิจราณาจากไดอะแกรม



รูปที่ 6-10 แสดงคลาส Subject โดยพิจรณาที่ Attribute PresSub1และ PresSub2



รูปที่ 6-11 แสดง OONIAM ในการที่เปลี่ยนมาจากคลาส Subject

โดยจากรูปที่ 6-9 และ 6-10 แสดงการแก้ปัญหาภายในคลาสไดอะแกรมทำการระบุกฎข้อบังคับภายในคลาสไดอะแกรมได้โดยใช้ OONIAM จากรูป 6-10 เป็นกฎข้อบังคับชนิด Subset Constrain

จากปัญหาต่างที่กล่าวมาข้างต้น บางปัญหาก็มีทางที่จะแก้ไขแล้วแต่ยังมีอีกหลากหลายปัญหาที่ยังไม่สามารถแก้ได้โดยเป็นข้อจำกัดทางเวลาและความสามารถของผู้เขียน แต่ทั้งหมดนี้เป็นเพราะหากมีผู้ใดต้องการออกแบบและพัฒนาระบบงานขึ้นมาโดยใช้ยูเอ็มแอลจะได้เข้าใจถึงปัญหาและช่วยลดเวลาในการพัฒนาได้ และอาจหาหนทางลดข้อจำกัดในโมเดลต่อไปได้ในอนาคต

บทที่ 7

การออกแบบโปรแกรมกลาง (Common SoftWare)

หลักการการออกแบบ โปรแกรมคอมพิวเตอร์ที่สามารถนำโปรแกรมที่ได้เขียนไว้แล้วและนำกลับมาใช้ใหม่ หรือนำมาปรับปรุงเพิ่มเติม ได้มีมานานพอสมควรแล้วโดยหลักการ การโปรแกรมเชิงวัตถุ(Object Oriented Programming) ระบบของฐานข้อมูลก็ได้เริ่มมีการใช้ ฐานข้อมูลเชิงวัตถุ(Object Oriented Database)มากขึ้นตามลำดับ แต่นักพัฒนาโปรแกรมก็ยังไม่สามารถใช้ความสามารถของฐานข้อมูลได้อย่างเต็มประสิทธิภาพ ผู้จัดทำจึงได้แนวความคิดว่าจะใช้หลักการ การโปรแกรมเชิงวัตถุมาออกแบบฐานข้อมูล และหาความต้องการพื้นฐานของงานนั้นๆ (Common business logic) เมื่อได้ความต้องการพื้นฐานแล้ว ก็นำมาออกแบบ โดยใช้ UML(Unified Modeling Language) และ OONIAM (Object Oriented Niam) มาช่วยในการออกแบบ เมื่อความต้องการพื้นฐานได้ออกแบบ เสร็จแล้วจึงได้นำมาพัฒนาโปรแกรมใช้งาน (Application Program) จากนั้นจึงออกแบบเพิ่มเติม เพื่อให้นำไปใช้กับระบบงานที่แท้จริง โดยใช้หลักการนำโค้ดของโปรแกรม นำกลับมาใช้ใหม่ โดยสืบทอด(Inheritance) ระบบงานจาก Common business logic และเปรียบเทียบ ว่าแนวคิด ในการออกแบบ มีประมากกว่า การออกแบบระบบใหม่ทั้งหมด หรือไม่ เพื่อวิเคราะห์ ถึงข้อได้เปรียบฐานข้อมูลเชิงวัตถุ ว่าสามารถนำมาใช้ในการออกแบบและสร้างระบบงานได้จริงหรือไม่

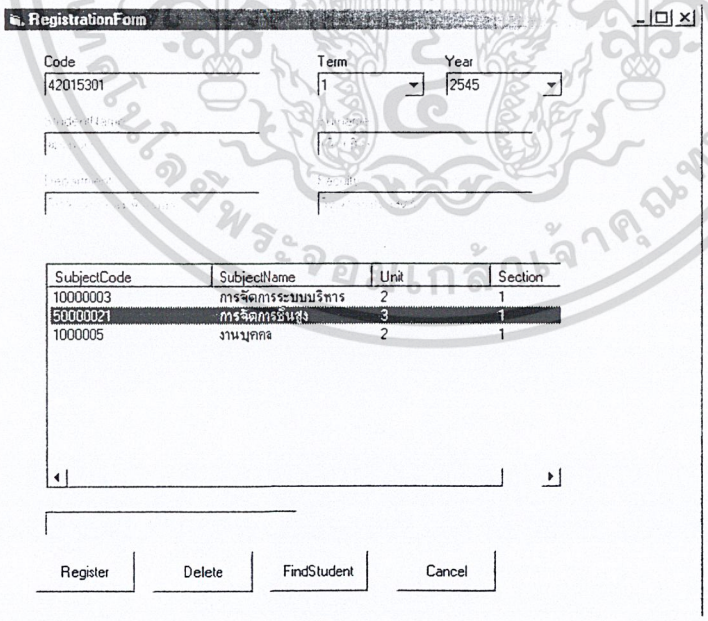
จุดเด่นจากการใช้ฐานข้อมูลเชิงวัตถุ

สามารถพัฒนาระบบที่มี ลักษณะงานคล้ายกันหลาย ๆ ระบบได้รวดเร็ว โดยอาศัยการการนำโค้ดโปรแกรมนำกลับมาใช้ใหม่ และคุณสมบัติ Polymorphism (Overide method and Overide Attribute)ได้ระบบที่มีความสอดคล้อง(Data Consistency) เนื่องจากระบบอาจจะต้องมีการปรับปรุงเปลี่ยนแปลง เช่น ในระบบฐานข้อมูลการทะเบียน ต้องการแยกข้อมูล นักศึกษามาทำเอง ก็สามารถแยกได้โดยอาศัย คุณสมบัติการสืบทอดคุณสมบัติ ออกมาเป็น คลาสลูก และเนื่องจาก คลาสลูก จะมีคุณสมบัติทุกประการเหมือนกับ คลาสที่คลาสลูกได้สืบทอดมาทุกประการ ทำให้คลาสแม่สามารถ เปิดคลาสที่สืบทอดคุณสมบัติ จากมันได้ และทำการปรับ ปรุงแก้ไข ข้อมูลทำให้ข้อมูลมีความสอดคล้องกัน การที่สามารถทำการ Overide method ได้นั้นทำให้ ระบบงานที่ใหญ่ มีหลาย หน่วยงาน และอาจจะใช้ข้อมูลร่วมกัน แต่กฎความถูกต้องของข้อมูล(Data intrigrity) มีความแตกต่างกัน อาศัยคุณสมบัติ Polymorphism ทำการ Overide method เพื่อให้ระบบที่มีความแตกต่างกัน ทำงานร่วมกันได้

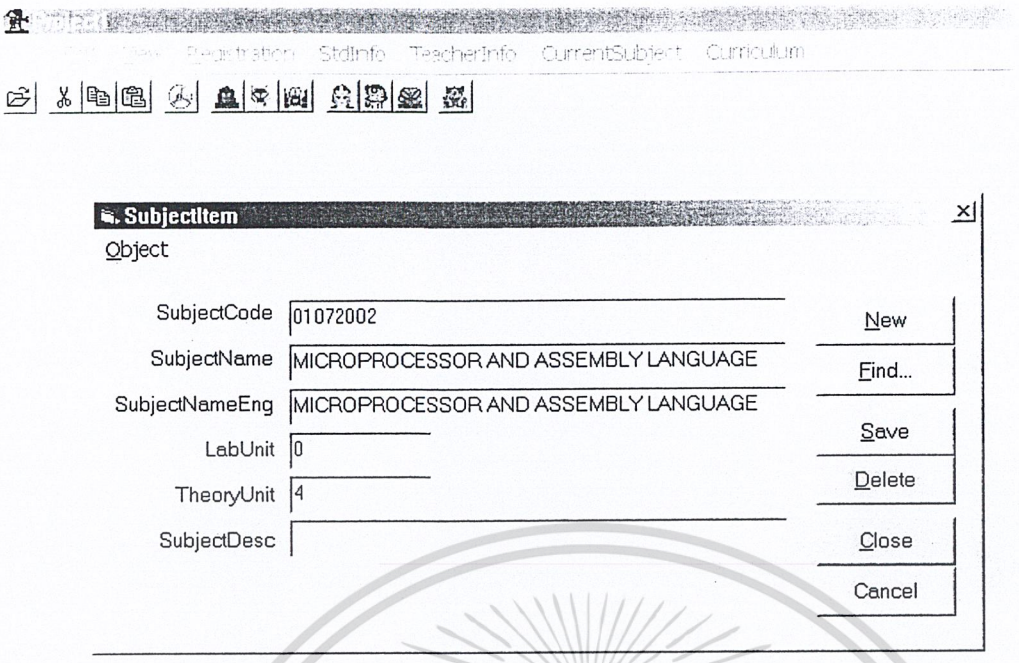
จากการศึกษาและออกแบบระบบและทำการพัฒนาระบบงานฐานข้อมูลนักเรียน นักศึกษา และระบบลงทะเบียนโดยอาศัยคุณสมบัติของฐานข้อมูลเชิงวัตถุทำการ Override Method เพื่อให้เกิดการทำงานแบบ Polymorphism เนื่องจากกฎการลงทะเบียนของแต่ละคณะวิชามีข้อแตกต่างกัน โดยขอยกตัวอย่างเช่นบางคณะวิชาอนุญาติให้นักศึกษาที่ติดทัณฑ์บนลงทะเบียนได้ไม่เกิน 15 หน่วย หรือบางคณะอาจจะไม่มีกฎข้อบังคับในส่วนนี้ และในส่วนของข้อมูล เช่นข้อมูลครู อาจารย์ของแต่ละคณะอาจมีข้อมูลไม่เหมือนกัน หากจะแยกฐานข้อมูล อาจทำให้เกิดปัญหาการไม่สอดคล้องกันของข้อมูล (Data

Inconsistency) แต่เนื่องด้วยระบบฐานข้อมูลที่ทำขึ้นเป็นฐานข้อมูลเชิงวัตถุ จึงอาศัยคุณสมบัติของฐานข้อมูล (Inheritance) มาแก้ไขปัญหาในส่วนนี้

รูปที่ 7-1 แสดงหน้าต่างหลักของระบบลงทะเบียนนักศึกษา

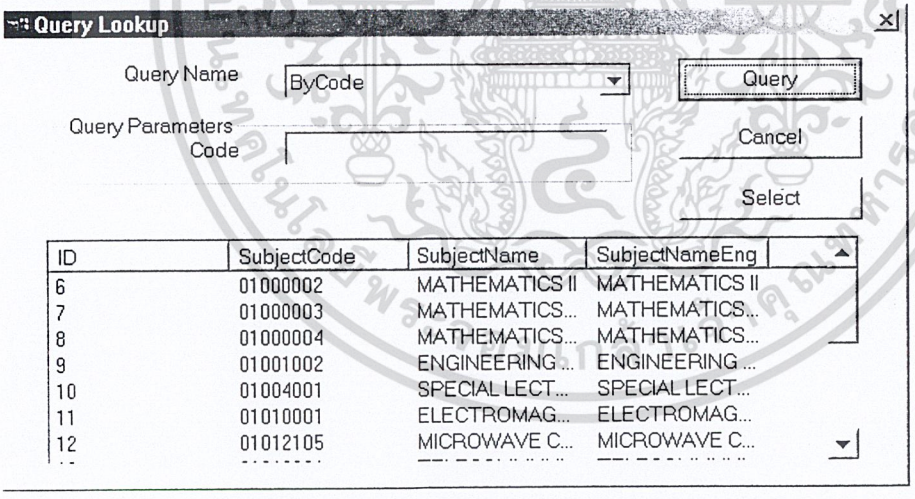


รูปที่ 7-2 แสดงหน้าต่างระบบลงทะเบียน



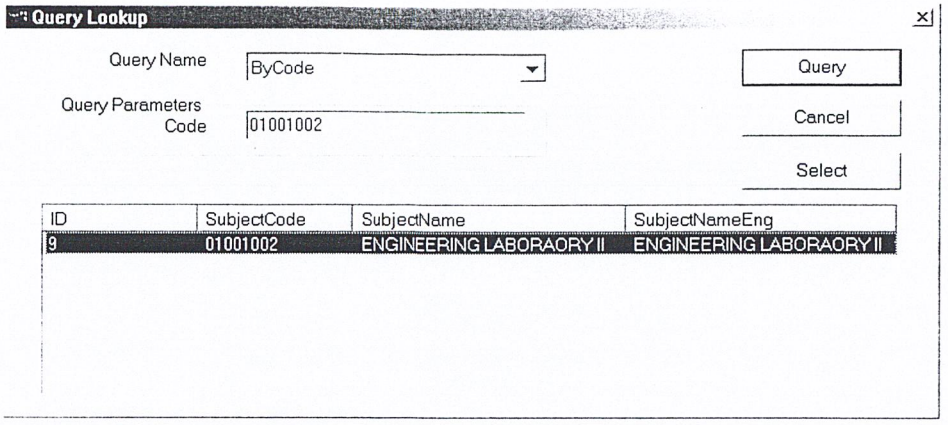
รูปที่ 7-3 แสดง การเพิ่มข้อมูลรายวิชา

จากรูปที่ 7-3 แสดงการเพิ่มข้อมูลรายวิชา แต่หากต้องการลบข้อมูลรายวิชาสามารถทำได้โดยเลือกคำสั่ง Delete ได้ และทำการเพิ่มข้อมูลโดยคำสั่ง save

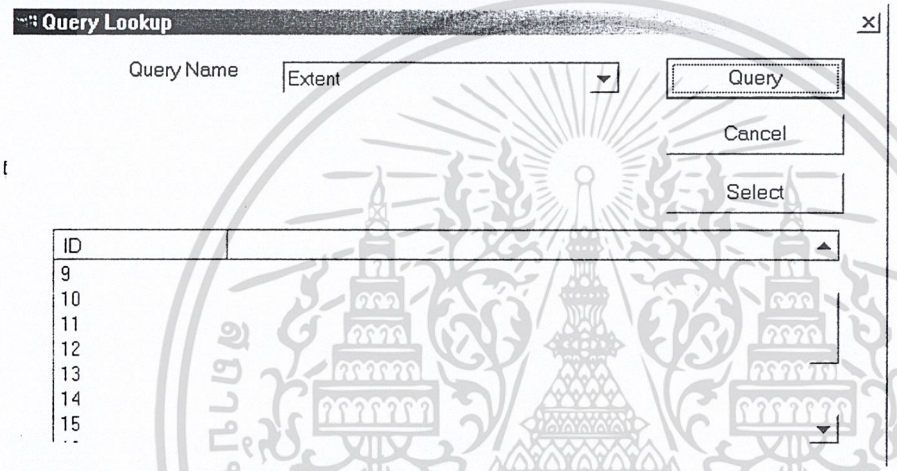


รูปที่ 7-4 แสดงการค้นหารายชื่อข้อมูลรายวิชา

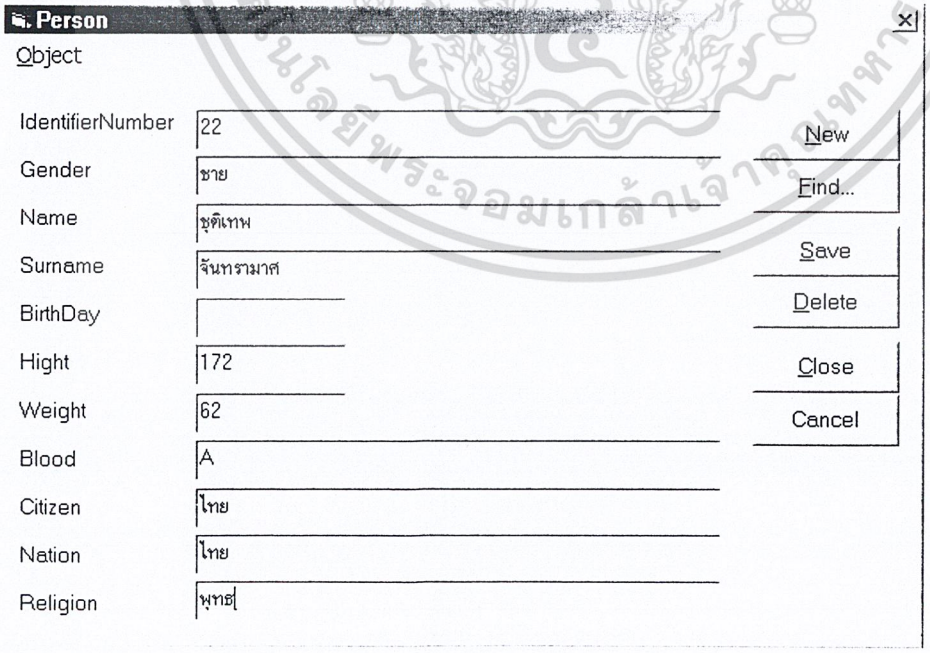
จากรูปที่ 7-4 แสดงการค้นหารายชื่อข้อมูลรายวิชาข้อมูลรายวิชา โดยทำการค้นหาเรียงลำดับตามลำดับของ Subject Code และสามารถดูรายละเอียดข้อมูลได้โดยเลือกคำสั่ง Select



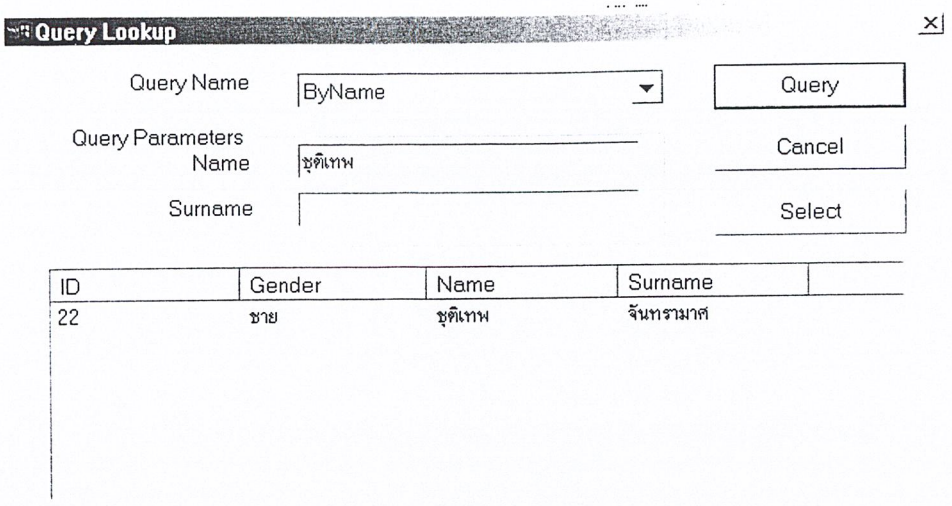
รูปที่ 7-5 แสดงการค้นหารายชื่อข้อมูลรายวิชาโดยค้นหาตามรหัสวิชา



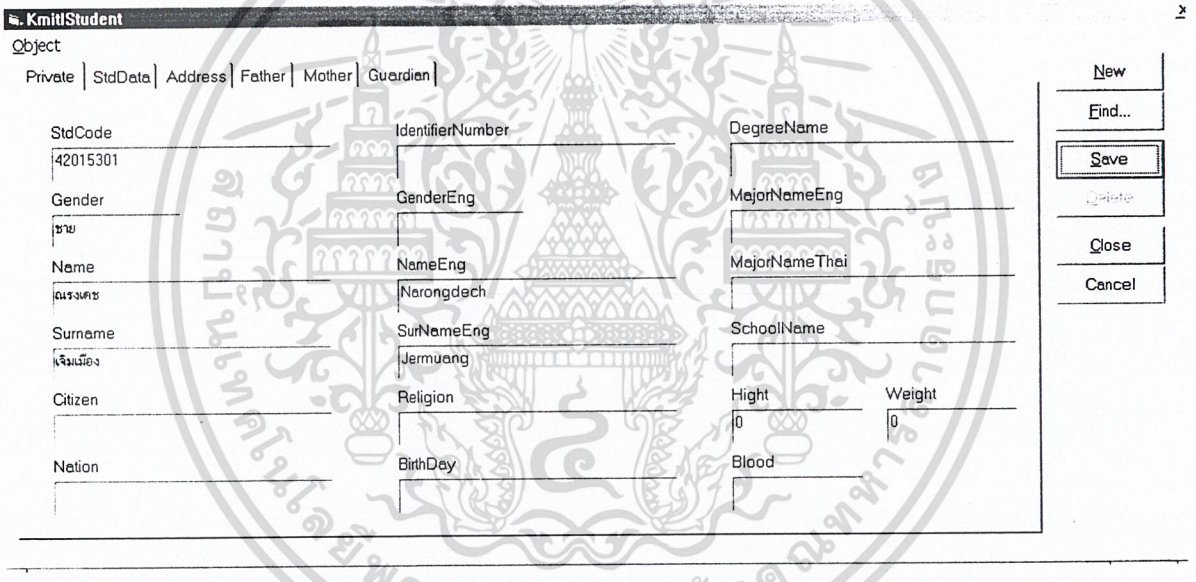
รูปที่ 7-6 แสดงการค้นหาโดยตาม Object ID



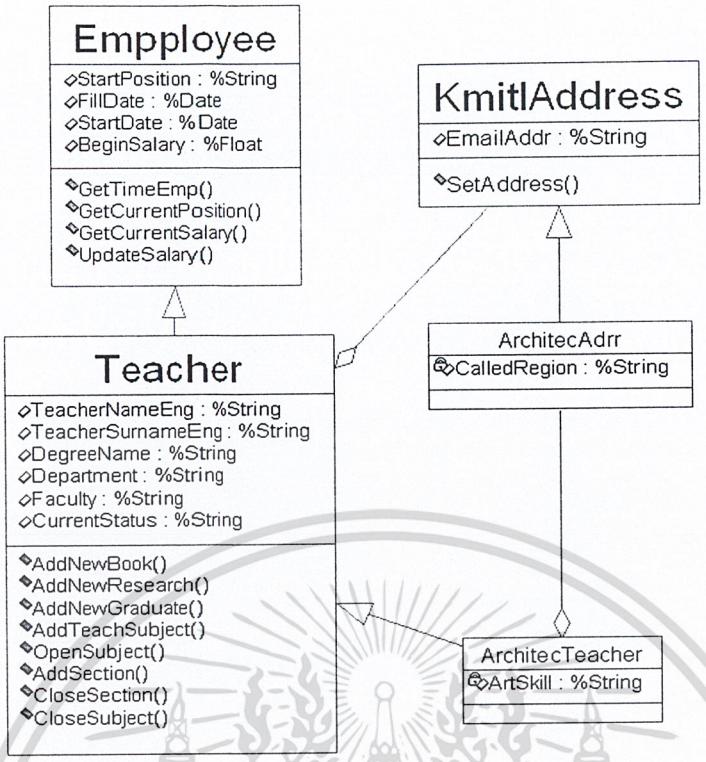
รูปที่ 7-7 แสดง การเพิ่มรายชื่อบุคคลภายในระบบ



รูปที่ 7-8 แสดงการค้นหารายชื่อบุคคล โดยค้นหาตามชื่อจริง

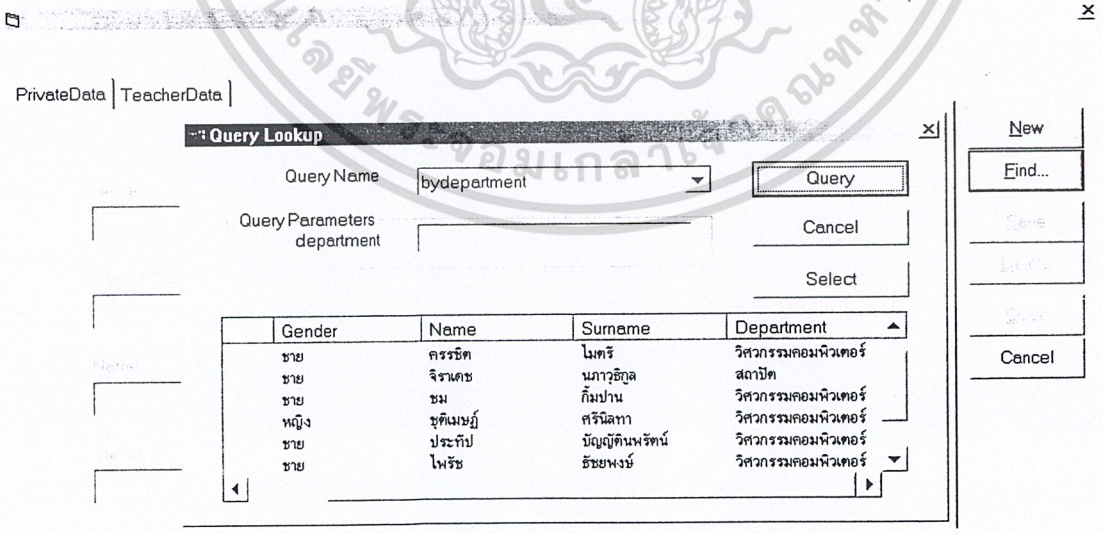


รูปที่ 7-9 แสดงการเพิ่มข้อมูลนักศึกษา



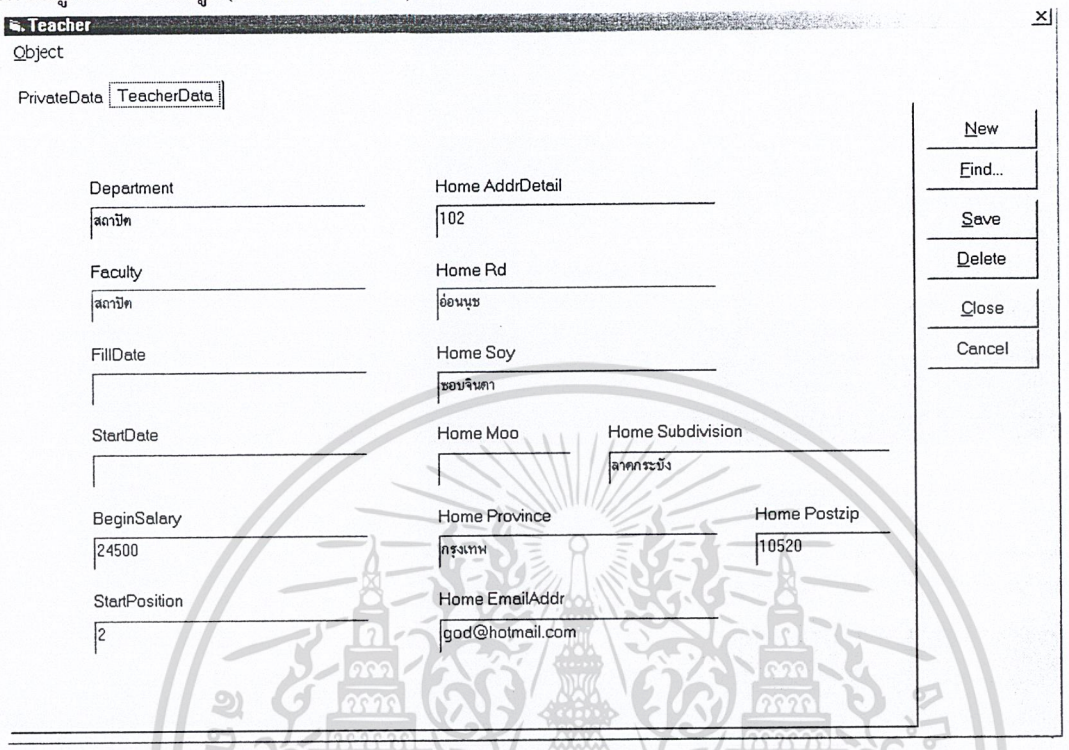
รูปที่ 7-10 แสดงคลาสไดอะแกรมที่ได้ทำการ Override Attribute

จากรูปจะเห็นได้ว่าคลาส Teacher จะมี Attribute เป็น KmitlAddress และคลาส ArchitecTeacher ได้รับการสืบทอดคุณสมบัติจากคลาส Teacher โดยเพิ่ม Attribute ชื่อว่า “ArtSkill” และทำการ Override Attribute จาก KmitAddress ไปเป็น ArchitecAdrr



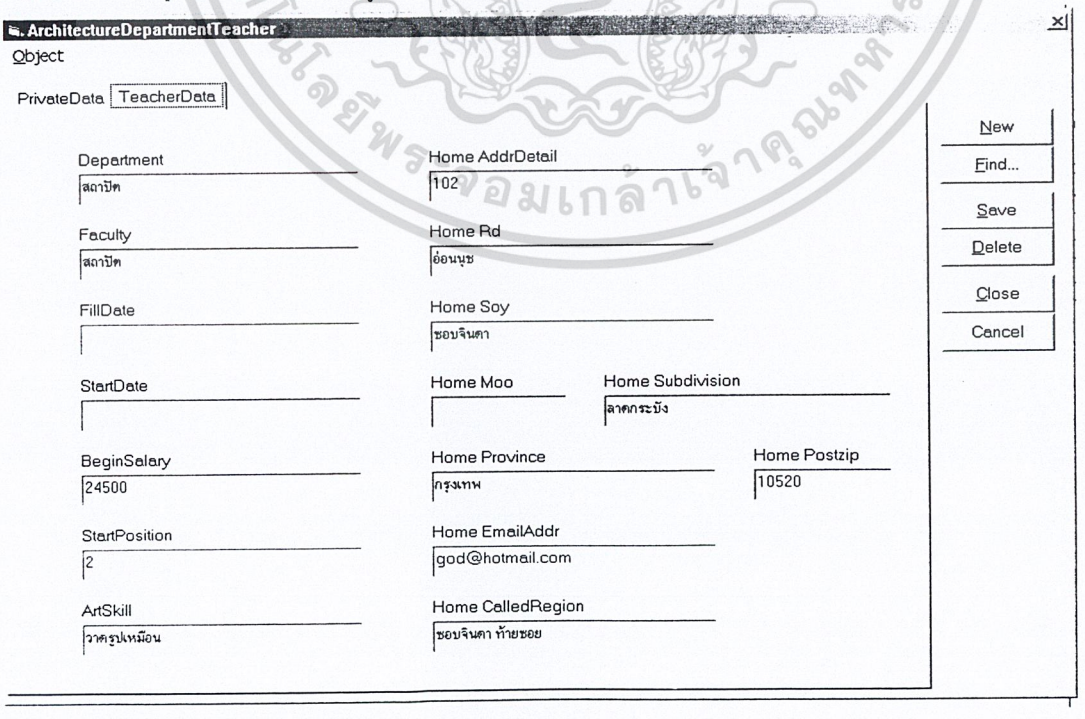
รูปที่ 7-11 แสดงผลที่ได้ลัพธ์จากการ Query คลาส Teacher

จากผลการ Query ของคลาส Teacher จะเห็นได้ว่าจะสามารถเห็น Object ทั้ง Teacher และ ArchitecTeacher ทั้งนี้เนื่องมาจากคุณสมบัติของการ Inheritance ทำให้คลาสแม่สามารถเห็นและทำการเปิดข้อมูลของ คลาสลูก(ArchitecTeacher)ได้

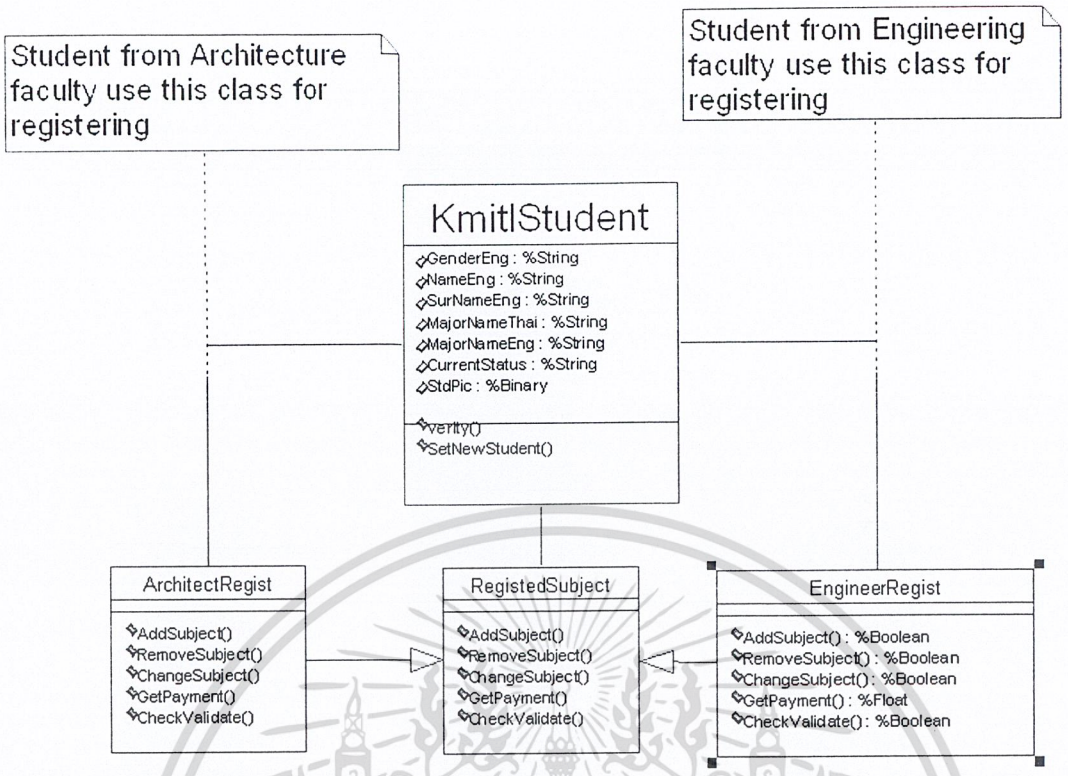


รูปที่ 7-12 แสดงการเปิด Object ของคลาส Teacher

จากรูปที่ 7-12 แสดงให้เห็นว่าคลาสแม่นั้นสามารถเข้าถึง Attribute และ Operation ของคลาสลูกได้ ยกเว้น Attribute และ Operation ของคลาสลูก



รูปที่ 7-13 แสดงการเปิด Object ของคลาส ArchitecTeacher



รูปที่ 7-14 แสดงคลาสไดอะแกรมของการ Override Method ของระบบลงทะเบียน

จากรูปที่ 7-14 กฎข้อบังคับของการลงทะเบียนแต่ละคณะวิชานั้นมีข้อบังคับและกฎของการลงทะเบียนที่มีความแตกต่างกันได้

บทที่ 8

สรุปและวิจารณ์ผลการทดลอง

8.1 ทำการศึกษาหลักการของแบบจำลองเชิงวัตถุ (Object Oriented Model) ในแนวคิดต่างๆจึงได้เกิดแนวความคิดการพัฒนาระบบฐานข้อมูล โดยการออกแบบโปรแกรมให้มีลักษณะแบบโปรแกรมกลาง(Common SoftWare)ที่เป็นหลักการการออกแบบ โปรแกรมคอมพิวเตอร์ที่สามารถนำโปรแกรมที่ได้เขียนไว้แล้วและนำกลับมาใช้ใหม่ หรือนำมาปรับปรุงเพิ่มเติมได้ ตามหลักการ การโปรแกรมเชิงวัตถุ(Object Oriented Programming) และนำมาใช้กับการออกแบบ ฐานข้อมูลเชิงวัตถุ(Object Oriented Database)เพื่อใช้ความสามารถของฐานข้อมูลได้อย่างเต็มประสิทธิภาพ ผู้จัดทำจึงได้แนวความคิดว่าจะใช้หลักการ การโปรแกรมเชิงวัตถุมาออกแบบฐานข้อมูล และหาความต้องการพื้นฐานของงานนั้นๆ (Common business logic) และเมื่อได้ความต้องการพื้นฐานแล้ว ก็นำมาออกแบบ โดยใช้ UML(Unified Modeling Language) และ OONIAM (Object Oriented Niam) มาช่วยในการออกแบบ เมื่อความต้องการพื้นฐานได้ออกแบบ เสร็จแล้วจึงได้นำมา พัฒนาโปรแกรมระบบลงทะเบียนและข้อมูลนักศึกษาจากนั้นจึงออกแบบเพิ่มเติม เพื่อให้นำไปใช้กับระบบงานที่แท้จริง โดยใช้หลักการนำโค้ดของโปรแกรม นำกลับมาใช้ใหม่ โดยสืบทอด(Inheritance) ระบบงานจาก Common business logic และได้ทดสอบว่าแนวคิดในการออกแบบระบบโดยนำโค้ดโปรแกรมมาใช้ใหม่ มีข้อได้เปรียบในการออกแบบ ระบบงานได้จริง

8.2 ทำการศึกษาและวิเคราะห์กระบวนการการออกแบบระบบโดยใช้ UML

8.3 ทำการแก้ไขข้อบกพร่องและทำการปรับปรุงแก้ไข UML ในการออกแบบระบบให้สามารถรองรับการทำงานได้มากยิ่งขึ้นในบางส่วน

8.4 ศึกษาและวิเคราะห์กระบวนการการออกแบบและออกแบบระบบ โดยใช้ OONIAM

8.5 ทำการพัฒนาระบบที่ได้รับการออกแบบไว้ พัฒนาโดยใช้ระบบจัดการฐานข้อมูล Cache' 4.0 และ Visual Basic 6.0

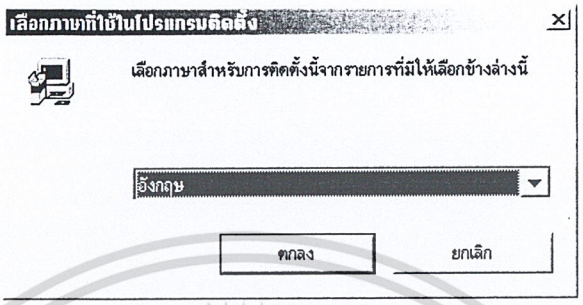
ภาคผนวก ก.

การติดตั้งเริ่มต้นใช้งาน Cache′ 4.0

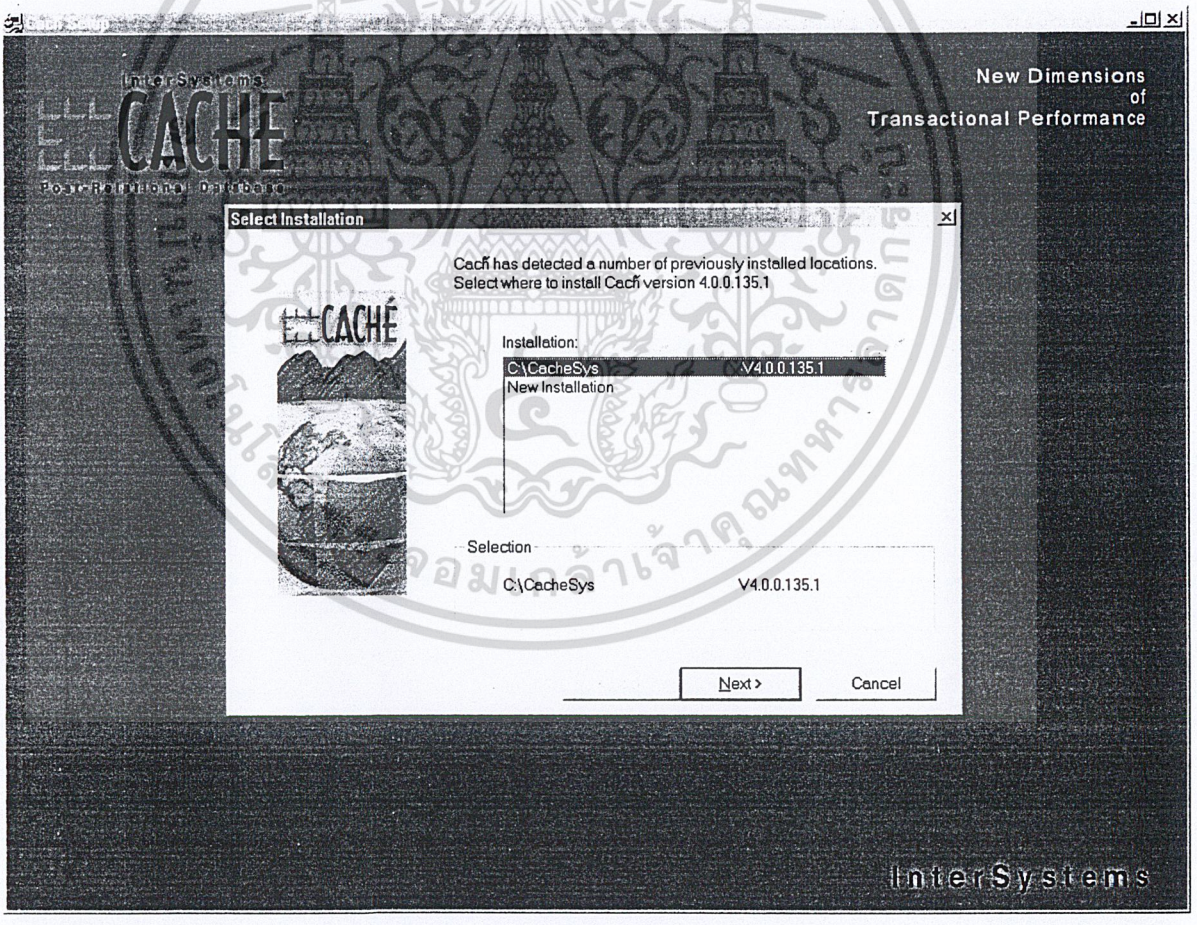
การ Install โปรแกรม Cache′ DBMS

เริ่มแรกเรียก ไฟล์ setup.exe จากแผ่นติดตั้ง Cache′ 4.0

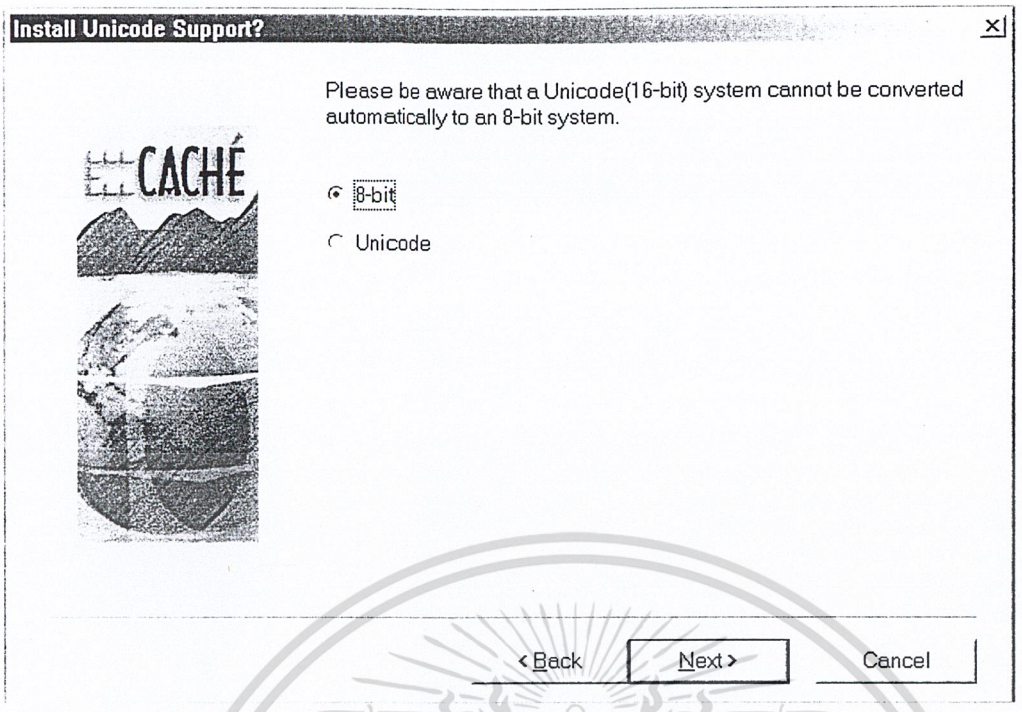
Install Cache′ DBMS



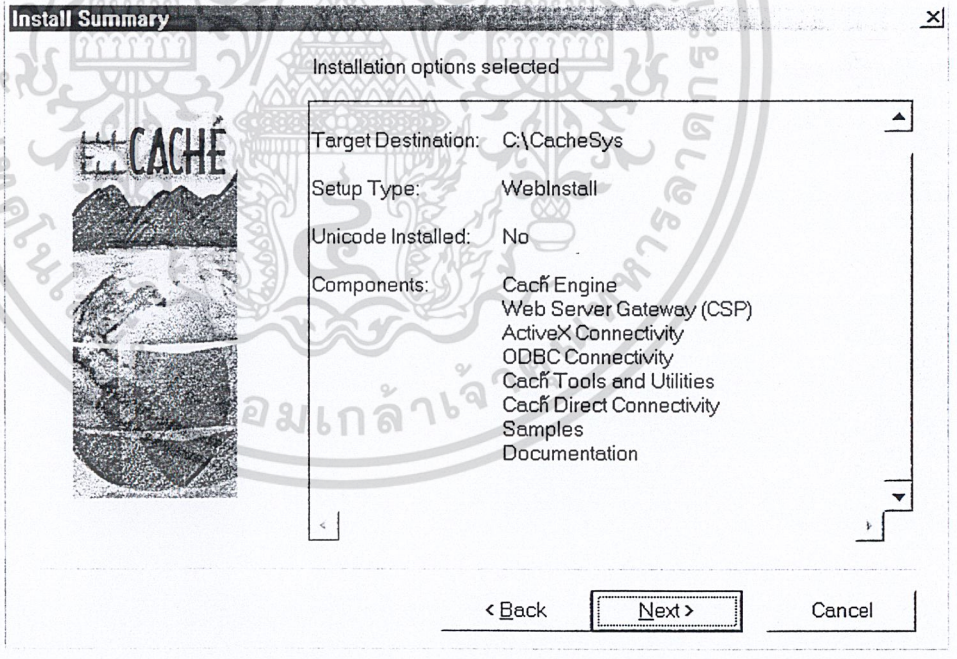
รูปที่ ก-1 ทำการเลือกภาษาสำหรับการติดตั้ง



รูปที่ ก-2 ทำการคลิ๊กที่ Next เพื่อทำงานขั้นต่อไป



รูปที่ ก-3 ทำการเลือก 8-bit และทำการคลิ๊กที่คำสั่ง Next



รูปที่ ก-4 แสดงรายละเอียดของส่วนประกอบที่ทำการติดตั้งและเลือกคำสั่ง Next เพื่อทำงานต่อไป

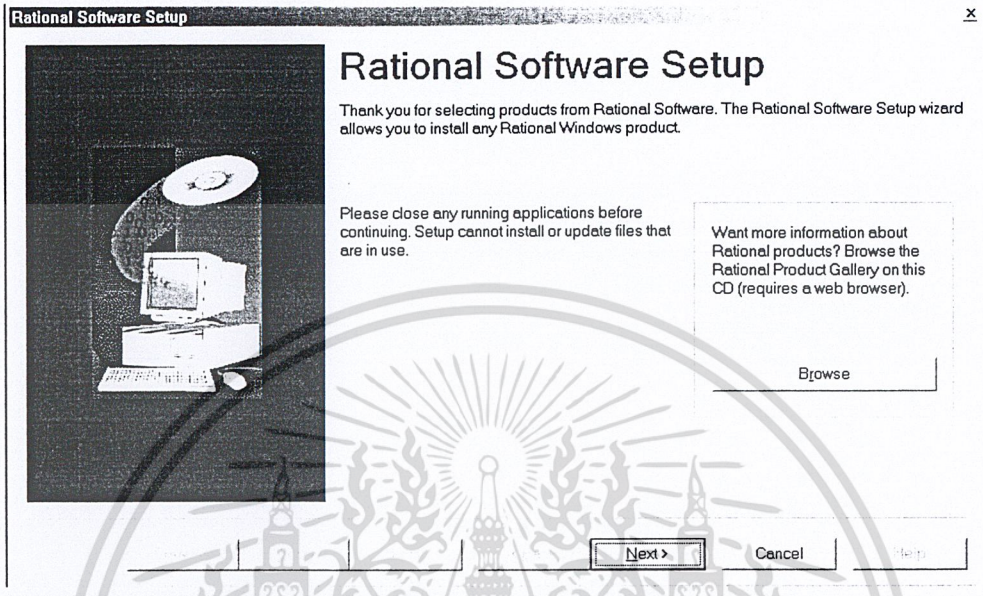
รูปที่ ก-5 แสดงการติดตั้งเสร็จสิ้น ทำการเลือก Finish เพื่อทำงานขั้นต่อไป

ภาคผนวก ข.

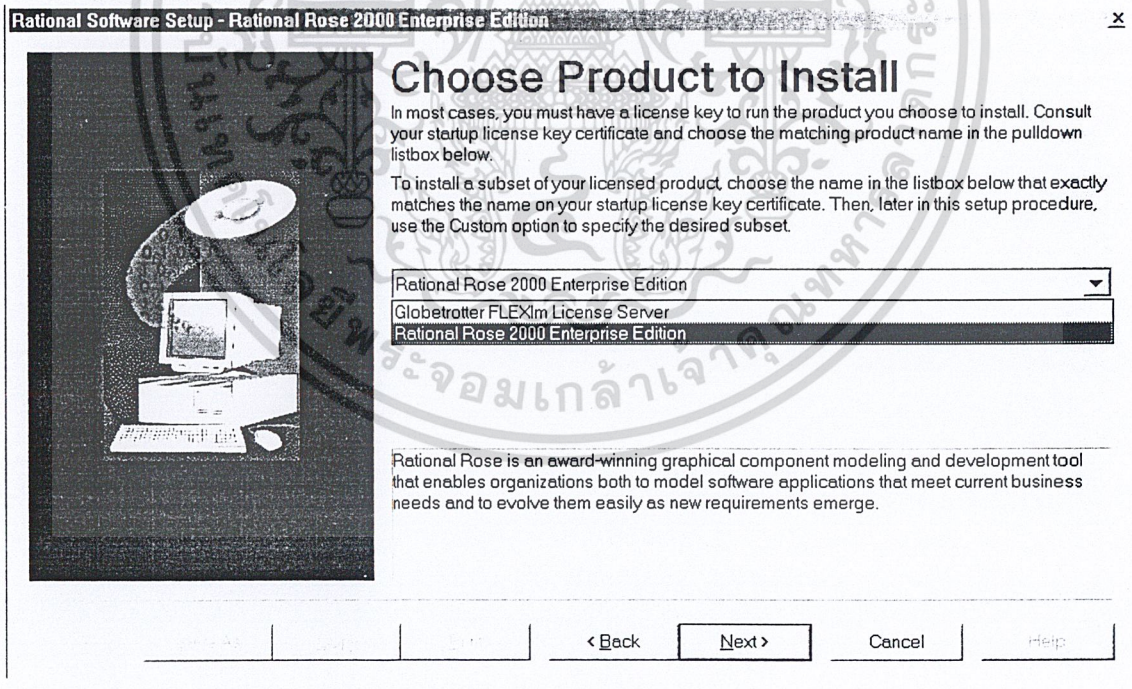
การติดตั้งเริ่มต้นใช้งาน Rose 2000

การ Install โปรแกรม Rose 2000

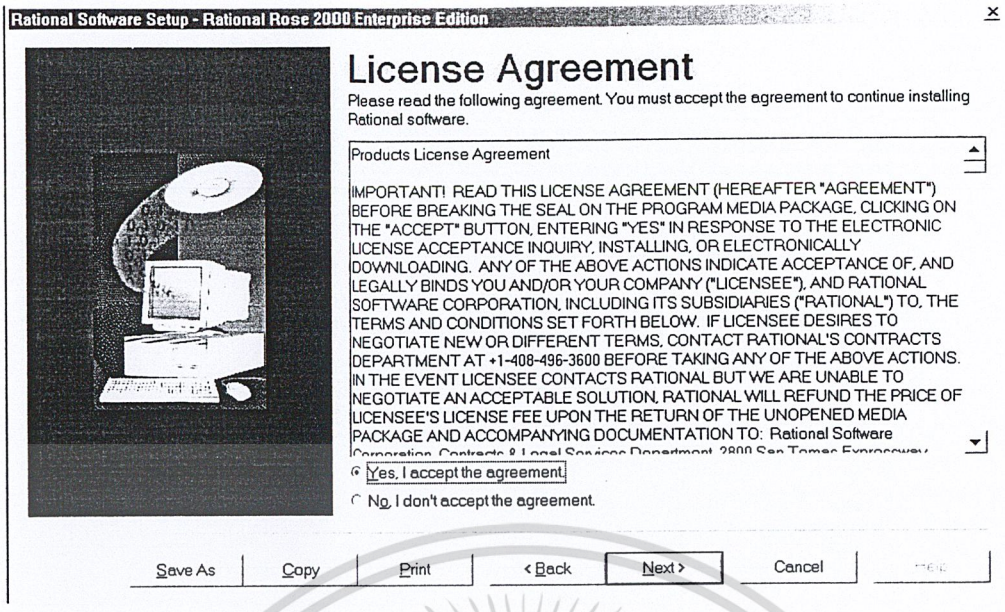
เริ่มแรกเรียก ไฟล์ setup.exe จากแผ่นติดตั้ง Rational Rose 2000



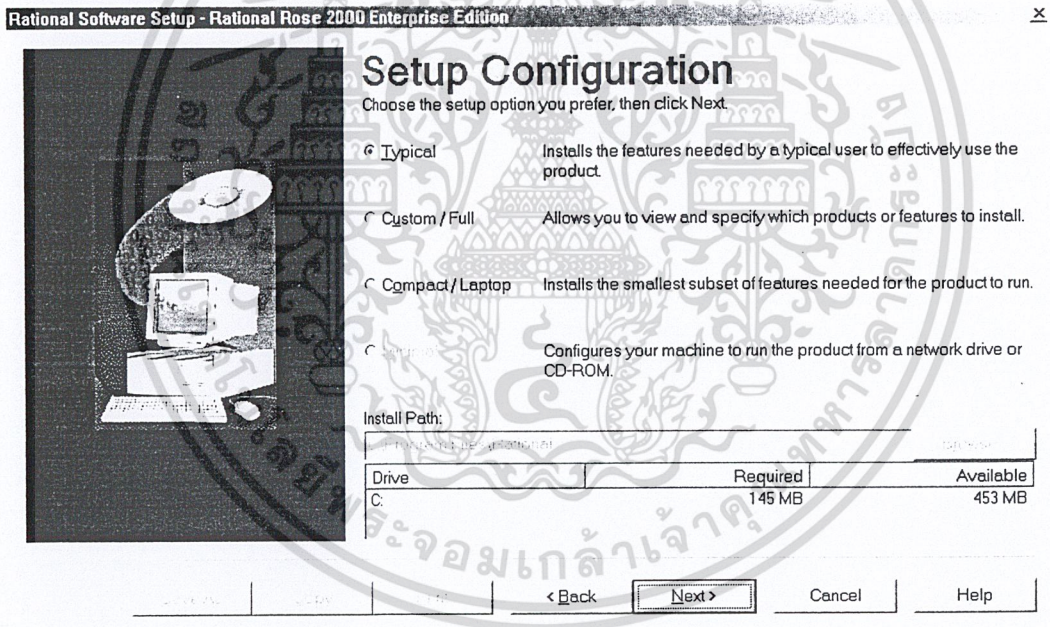
รูปที่ ข-1 แสดงการเริ่มต้นการติดตั้ง Rational Rose ทำการเลือก Next เพื่อทำงานต่อไป



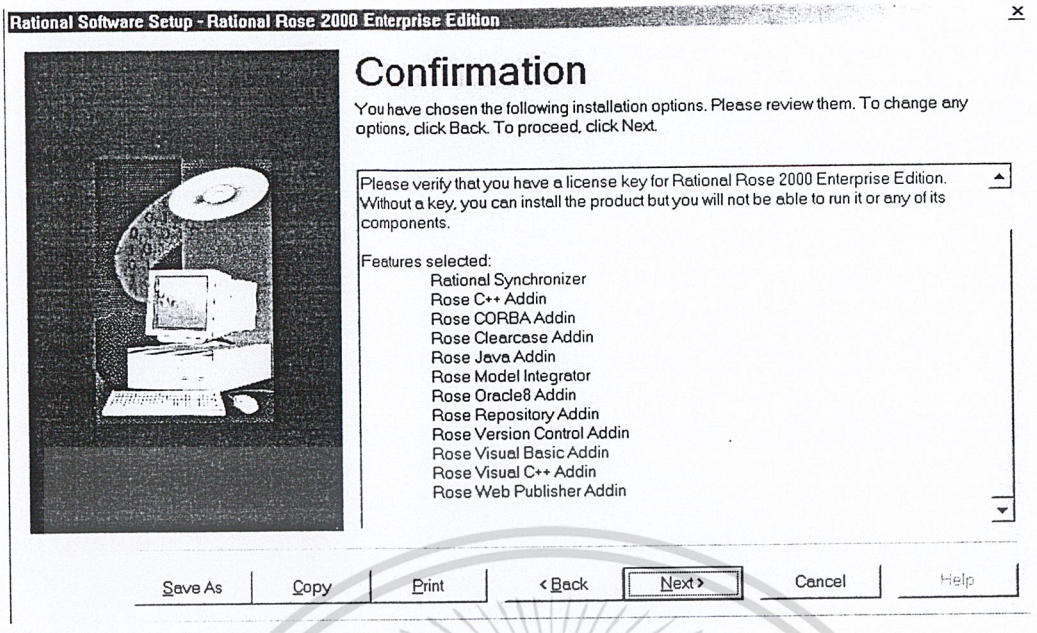
รูปที่ ข-2 ทำการเลือกที่ Rational Rose 2000 ภายใน list และคลิ๊กที่ Next



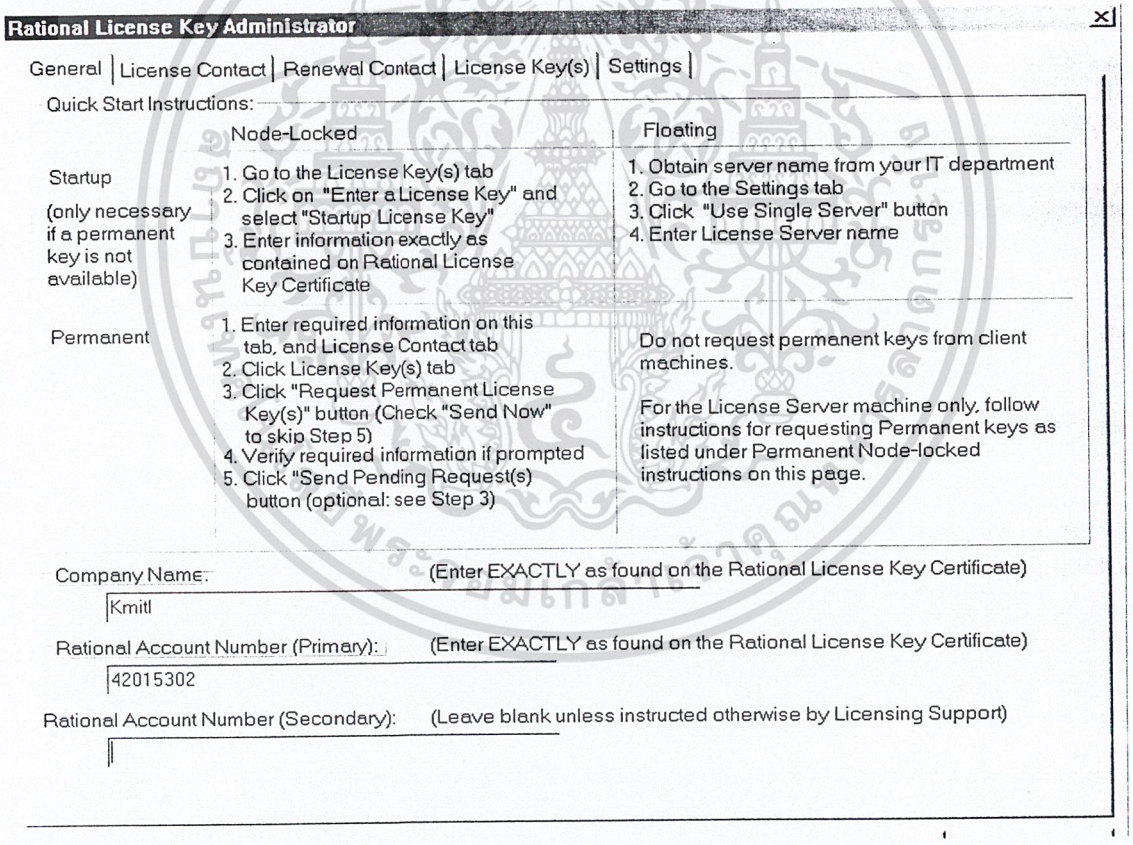
รูปที่ ข-3 เลือกที่ Yes และทำการคลิ๊กที่ Next



รูปที่ ข-4 ทำการเลือกการติดตั้งตามความเหมาะสมและการใช้งาน และทำการคลิ๊กที่ Next



รูปที่ ข-5 แสดงส่วนประกอบที่ทำการติดตั้งและคลิ๊ก Next เพื่อทำงานขั้นต่อไป



รูปที่ ข-6 เมื่อทำติดตั้งเสร็จสิ้นทำการลงข้อมูลให้ครบถ้วนเพื่อเรียกใช้งานโปรแกรม

หลังจากติดตั้งและกรอกข้อมูลเสร็จสิ้นแล้วมาที่ไดเร็กทรอรี่

C:\Program Files\Rational\RSSetup

รูป ข-3 แสดง ไฟล์ rssetup.exe

ทำการเลือกและดับเบิ้ลคลิ๊กที่ RSSetup.exe หลังจากนั้นทำการเรียกใช้ Rational Rose 2000 ได้ตามความต้องการ

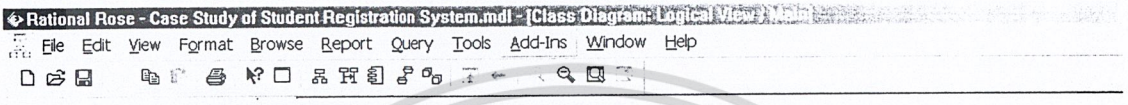
ในที่นี้ใช้ Rational Rose 2000 ถ้าหากต้องการ version ใหม่ก็สามารถทำการ download ได้จาก http://www.rational.com

ภาคผนวก ค

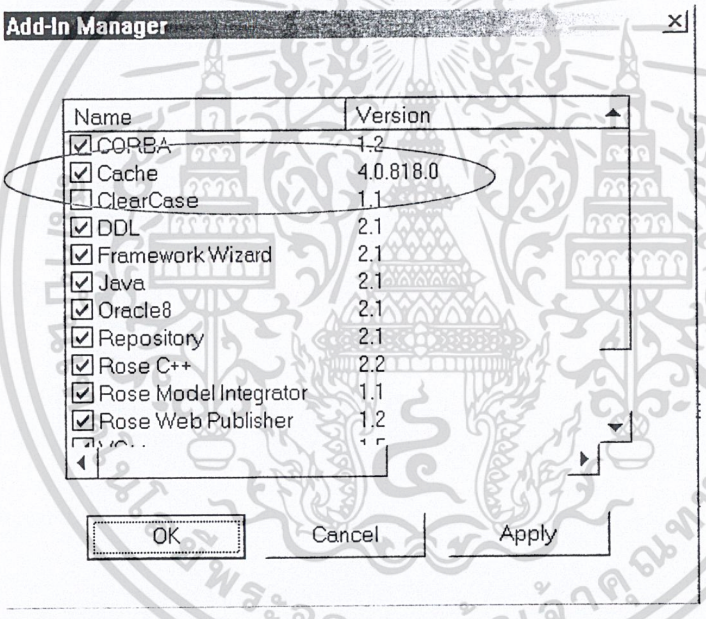
การติดต่อระหว่าง Cache′ 4.0 และ Rational Rose 2000

การติดตั้งถ้าคุณทำการติดตั้ง Rational Rose ก่อนที่คุณทำการติดตั้ง Cache′ 4.0 โปรแกรม Rational Rose อาจมองไม่เห็นโปรแกรม Cache′ ให้คุณทำการ Setup Cache′ 4.0 อีกครั้งหนึ่ง

หลังจากทำการออกแบบคลาสไดอะแกรมตามความต้องการเสร็จสิ้นแล้ว จะสามารถทำการ Export Class ที่ทำการออกแบบไว้แล้วนั้นเข้าสู่ Namespace Database ภายใน Cache′ ได้โดยเริ่มต้นนั้นทำการเลือกที่คำสั่ง Add-Ins ภายใน Rational Rose

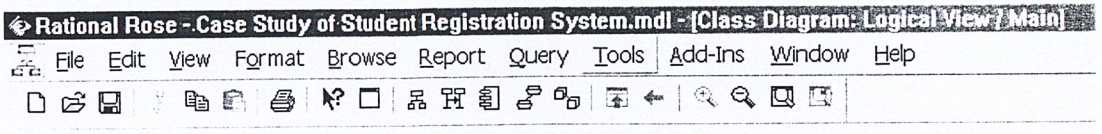


รูปที่ ค-1 แสดงเมนูหลักของ Rational Rose ทำการคลิ๊กที่ Add-Ins



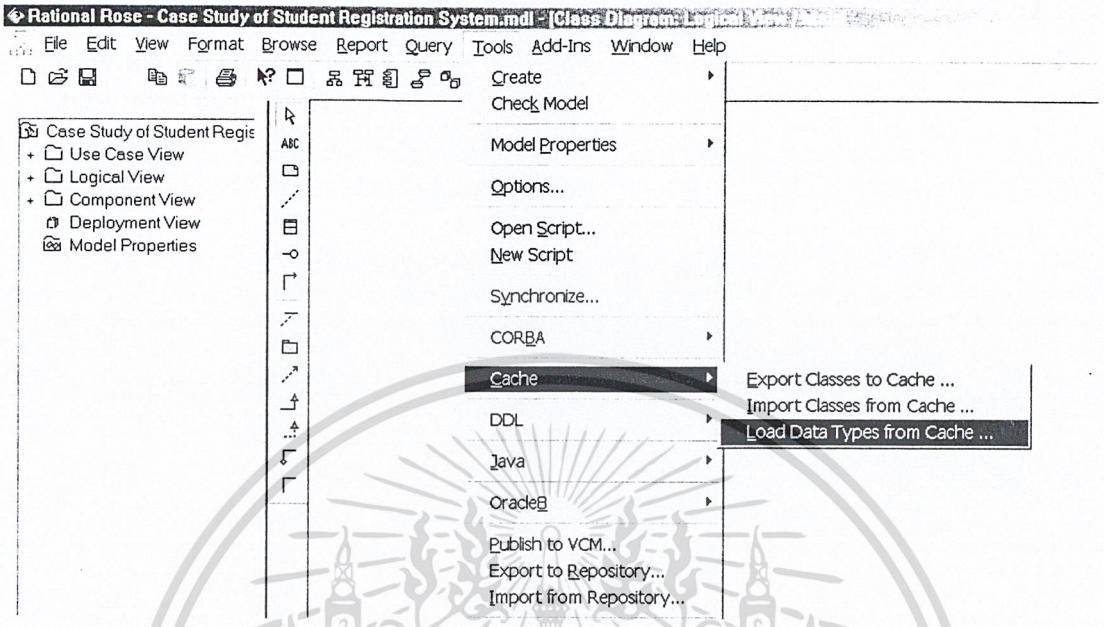
รูปที่ ค-2 แสดงเมนู Add-In Manager ทำการเลือกที่ Cache 4.0.818.0 เพื่อทำการเลือกเมนู Export และ Import ไปสู่ Cache′ 4.0

ขั้นต่อไปทำการเลือกที่เมนู Tool

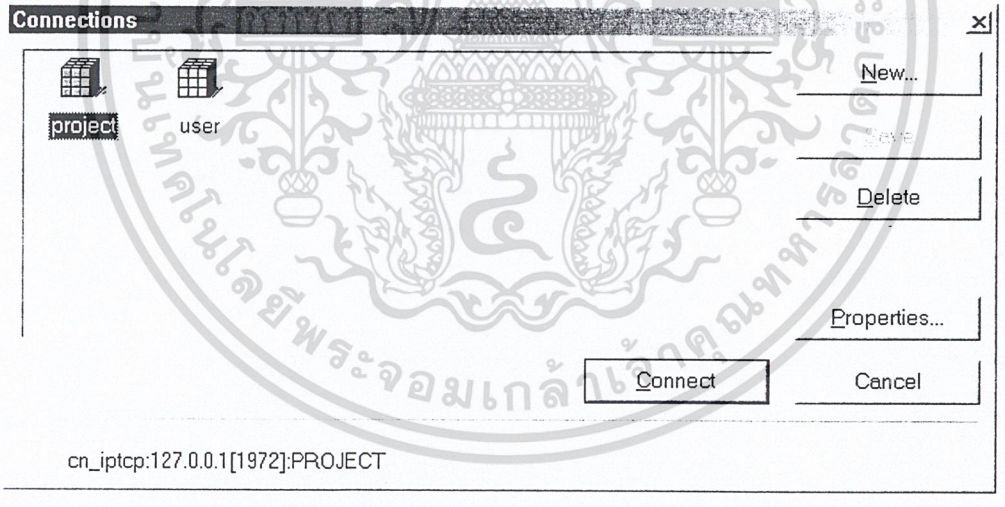


รูปที่ ค-3 แสดงเมนูหลักของ Rational Rose ทำการคลิ๊กที่ Tools

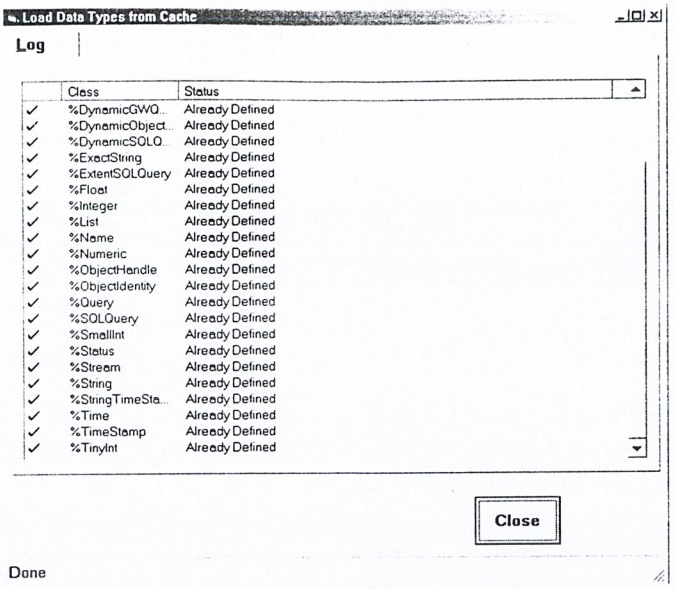
ก่อนอื่นต้องทำการ Load Data Types จาก Cache' เข้ามาก่อนเพื่อเลือกชนิดของ Attribute หรือการ Return Type ภายในคลาสไดอะแกรมที่ทำการออกแบบให้ตรงกับ Types ภายใน Cache' แสดงได้โดย



รูปที่ ค-4 เลือกที่ Tools และทำการคลิ๊กเลือกที่ Load Data Type from Cache…

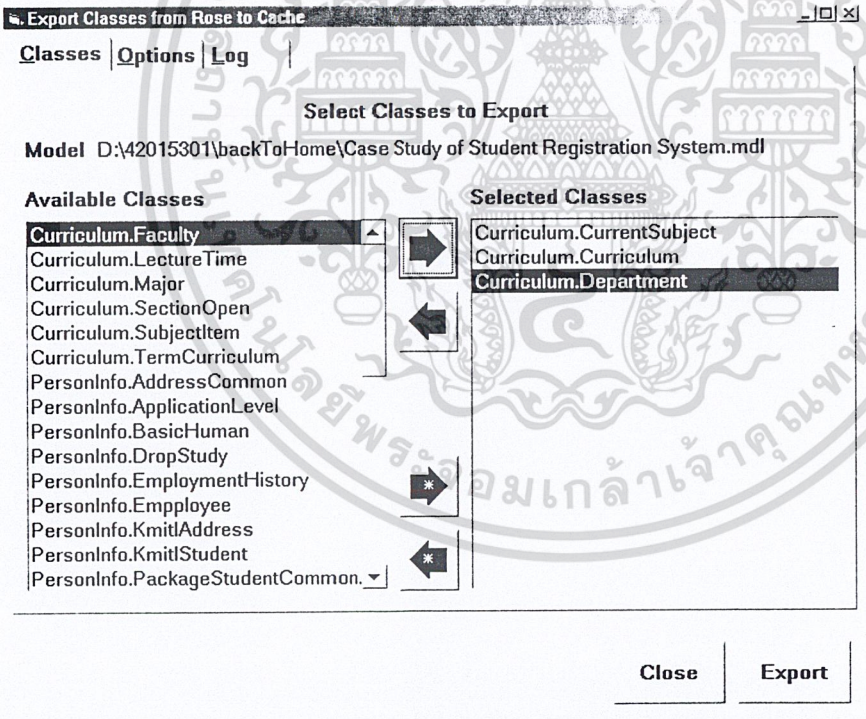


รูปที่ ค-5 ทำการเลือก Connect เพื่อเข้าไปยัง Database Name Space User



รูปที่ ค-6 แสดงการ Load Data Types เสร็จสิ้น และทำการคลิ๊กที่ Close

เมื่อทำการจัดการออกแบบภายในคลาสไดอะแกรมเสร็จสิ้นแล้วทำการ Export คลาสไปสู่ Cache′ โดยเลือกที่คำสั่ง Tools และ Export Classes to Cache..
และทำการเลือกคลาสที่ต้องการไปสู่ Cache′



รูปที่ ค-7 แสดงการเลือกคลาสที่ต้องการและคลิ๊ก Export เพื่อทำงานต่อไป

เมื่อทำการ Export ต้องทำการเลือก Connect เพื่อเข้าไปยัง Database Name Space ที่ต้องการ โดยเมื่อ Export เสร็จสิ้นทำการตรวจเช็คได้โดยดูได้จาก
Cache′ เลือกที่เมนู Cache′ Object Architect และทำการ connect เข้าไปยัง Database space ที่ทำการ Export เข้าไป

บรรณานุกรม

[1] Bryon K. Ehlmann, Gregory A. Riccardi: A Notation for Describing Aggregate Relationships in an Object-Oriented Data Model. ADB 1994: 62-77

[2] Wolfgang Emmerich : Engineering Distributed Objects. : 34-42

[3] Bundit Pasaya,Suphamit Chittayasothorn : A Temporal Object Oriented Conceptual Schema Model.

[4] Bundit Pasaya : The TOONIAM conceptual schema model and a transformation to an object oriented database schema

[5] Apinetr Unakul : Object-Oriented Analysis And Disign.

[6] Cache′ Post-Relational Database Advanced System Management Guide Version 4.0,Intersystem Corporation,USA,1999.

[7] Terry Quatrani. (1998) : "*Visual Modeling with Rational Rose and UML*", Addison Wesley.

[8] ศุภชัย สมพานิช. Database Programming ด้วย Visual Basic ฉบับมืออาชีพ กรุงเทพฯ : อินโฟเพรส, 2543.

[9] http://www.rational.com

[10] http://www-3.ibm.com/software/ad/library/standards/ocl.html