ระบบสารสนเทศทะเบียนนักศึกษาด้วยวิธีการเชิงวัตถุ

The Registrar's Information Using Object Oriented approach

นาย เอกสกุล ศรีจอมขวัญ

นางสาว จิตรลดา ตุรงค์สมบูรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ระบบสารสนเทศทะเบียนนักศึกษาด้วยวิธีการเชิงวัตถุ

The Registrar's Information Using Object Oriented approach

โดย

นาย เอกสกุล ศรีจอมขวัญ

นางสาว จิตรลดา ตุรงค์สมบูรณ์

อาจารย์ที่ปรึกษา

อาจารย์ วิบูลย์ พร้อมพานิชย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบสารสนเทศทะเบียนนักศึกษาด้วยวิธีการเชิงวัตถุ

(The Registrar's Information Using Object Oriented approach)

ผู้จัดทำ

1.นาย เอกสกุล ศรีจอมขวัญ รหัสประจำตัว 38014673

2.นางสาว จิตรลดา ตุรงค์สมบูรณ์ รหัสประจำตัว 38014068

..... อาจารย์ที่ปรึกษา

(อาจารย์ วิบูลย์ พร้อมพานิชย์)

ระบบสารสนเทศทะเบียนนักศึกษาด้วยวิธีการเชิงวัตถุ

นาย เอกสกุล ศรีจอมขวัญ 38014673
นางสาว จิตรลดา ตุรงค์สมบูรณ์ 38014068
อาจารย์ วิบูลย์ พร้อมพานิชย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เรียบเรียงขึ้นจากผลงานที่ได้พัฒนาระบบทะเบียนสารสนเทศ ซึ่งถูกออกแบบโดยวิธีการเชิงวัตถุ (Object Oriented) และนำไปประยุกต์ใช้กับฐานข้อมูลเชิงสัมพันธ์ (Relational Database) โดยในการดำเนินงานในขั้นตอนของการออกแบบจะใช้เครื่องมือช่วยในการออกแบบชื่อ Rational Rose ซึ่งใช้ภาษาทางโมเดลลิ่งที่มีชื่อว่า UML (Unified Modeling Language)

ในการพัฒนาระบบนี้จะใช้ฐานข้อมูลที่เป็นฐานข้อมูลเชิงสัมพันธ์ ในการจัดเก็บข้อมูลในส่วนของ Application จะใช้โปรแกรมภาษา Visual C++ สำหรับในการออกแบบหน้าจอและนำข้อมูลจากฐานข้อมูลออกมาแสดง สำหรับระบบนี้ได้ทำการจำลองระบบทะเบียนบางส่วนเท่านั้น จากระบบที่ใช้งานจริงซึ่งจะสามารถนำไปใช้งานได้จริงจะต้องมีการพัฒนาต่อไป

The Registrar's Information Using Object Oriented approach

Akesakul Srijomkwan 38014673

Jitlada Turongsomboon 38014068

Mr.Wiboon Prompanich Advisor

Year 1998

Abstract

This thesis is an application of Object Oriented approach for the Registrar's Information and apply to use for the Relational Database . In the design phase of the development, the system use the tool of Rational Software Corp.,which is the modeling language named UML (Unified Modeling Language).

This system uses Relational Database. In the application, the User Interface and retrieve the data from database use Visual C++ programming. The system simulate the registrar's information in only some part , The complete system needs development additionally.

กิตติกรรมประกาศ

กว่าจะมาเป็นวิทยานิพนธ์ฉบับนี้ ผู้จัดทำต้องฝ่าฟันกับอุปสรรคมากมาย แต่ก็ได้รับความช่วยเหลือและกำลังใจจากบุคคลต่างๆมากมาย บุคคลท่านแรกที่ช่วยให้วิทยานิพนธ์นี้เสร็จเรียบร้อยก็คือ อาจารย์ที่ปรึกษาอาจารย์ วิบูลย์ พร้อมพานิชย์ ที่ให้คำแนะนำและแก้ไขข้อมูลที่ไม่ถูกต้องเกี่ยวกับวิทยานิพนธ์ และในส่วนของตัวโครงงาน ข้อมูลต่างๆที่จะต้องใช้ ซึ่งต้องขอขอบพระคุณมา ณ ที่นี้ และต้องขอขอบใจเพื่อนๆที่คอยถามสารทุกข์สุกดิบอยู่อย่างสม่ำเสมอ

บุคคลที่สำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ก็คือ บิดา มารดา อันเป็นที่รักยิ่งของผู้จัดทำ หากไม่มีท่านทั้งสี่ ที่คอยให้โอกาสไม่ว่าจะเป็นด้านการศึกษาหรือด้านอื่นๆที่จำเป็นต่อการดำรงชีวิต และให้กำลังใจอย่างสม่ำเสมอ ก็คงไม่มีวิยานิพนธ์ฉบับนี้เกิดขึ้น ต้องขอระลึกถึงและกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

เอกสกุล ศรีจอมขวัญ
จิตรลดา ตุรงค์สมบูรณ์

สารบัญ

บทคัดย่อภาษาไทย I
บทคัดย่อภาษาอังกฤษ II
กิตติกรรมประกาศ III
สารบัญ IV
สารบัญรูปภาพ VI
บทที่1 บทนำ 1
1.1 ความสำคัญและความเป็นมาในการพัฒนาระบบ 1
1.2 วัตถุประสงค์ 1
บทที่ 2 Unified Modeling Language (UML) 2
2.1 Unified Modeling Language 2
2.2 ส่วนประกอบของ UML 3
2.2.1 view 3
2.2.1.1 use case view 4
2.2.1.2 logical view 4
2.2.1.3 component view 5
2.2.1.4 concurrency view 5
2.2.1.5 Deployment view 5
2.2.2 diagrams 5
2.2.2.1 use case diagram 6
2.2.2.2 class diagram 6
2.2.2.3 object diagram 7
2.2.2.4 state diagram 7
2.2.2.5 sequence diagram 7
2.2.2.6 collaboration diagram 8
2.2.2.7 activity diagram 8
2.2.2.8 component diagram 9
2.2.2.9 deployment diagram 9
2.2.3 model element 9
2.2.4 general mechanisms 10
2.2.4.1 adornment 10
2.2.4.2 notes 10
2.2.4.3 specification 10
2.3 rational objectory process 11
บทที่ 3 Object Oriented Programming 13

3.1 การโปรแกรมเชิงวัตถุ 13
3.1.1 ความหมายของการโปรแกรมเชิงวัตถุ 13
3.1.2 ลักษณะที่สำคัญของการเขียนโปรแกรมตามแนวคิดแบบออบเจ็กต์ 14
3.1.2.1 การเข้าถึงข้อมูลภายในคลาส 14
3.1.2.2 การสืบทอดของคลาส (Inheritance) 15
3.1.2.3 คอนสตรัคเตอร์และดีสตรัคเตอร์ (Constructor & Destructor) 16
3.2 การรวมกันระหว่างการโปรแกรมเชิงวัตถุกับข้อมูลเชิงสัมพันธ์ 18
3.2.1 ระบบที่เป็นโมเดลเชิงสัมพันธ์และโมเดลเชิงวัตถุ 18
3.2.2 ความแตกต่างกันของคาต้าเอ็นติตี้ 19
3.2.3 เทคนิคการรวมข้อมูลเชิงสัมพันธ์กับการโปรแกรมเชิงวัตถุด้วยภาษา C++ 19
3.2.3.1 แม็ปตารางไปเป็นคลาส 19
3.2.3.2 การสร้างออบเจ็กต์-รีเลชั่นนัล เพอร์ซิสแทนส์เฟรมเวิร์ก 19
บทที่ 4 การดำเนินงานและการพัฒนาระบบ 21
4.1 ศึกษาระบบและวิเคราะห์ระบบ 21
4.1.1 Use-case Diagram 21
4.1.2 Sequence Diagrams 34
4.2 ออกแบบระบบ 49
บทที่ 5 การทดลอง ผลการทดลอง 67
5.1 ฮาร์ดแวร์และซอฟท์แวร์ที่ใช้ในการทดลอง 67
5.2 การทดลอง 67
- หน้าจอแสดงการทำงาน 69
บทที่ 6 สรุปและวิจารณ์ผลการทดลอง 77
6.1 สรุป 77
6.2 แนวทางการพัฒนาต่อไป 77
บรรณานุกรม 78

สารบัญรูปภาพ

รูปที่ 2.1 แสดง view ต่างๆใน UML 4

รูปที่ 2.2 Use case diagram ของธุรกิจประกันภัย 6

รูปที่ 2.3 แสดงตัวอย่าง class diagram ของการทำการค้า 6

รูปที่ 2.4 state diagram สำหรับลิฟท์ 7

รูปที่ 2.5 sequence diagram สำหรับ Print server 8

รูปที่ 2.6 collaboration diagram สำหรับ Print server 8

รูปที่ 2.7 แสดง model element ที่ใช้ร่วมกันใน diagram ต่างๆ 9

รูปที่ 2.8 ตัวอย่างของความสัมพันธ์แบบต่างๆ 10

รูปที่ 2.9 แสดงการเพิ่มความหมายให้กับสัญลักษณ์ของ element 10

รูปที่ 2.10 แสดงการเพิ่มรายละเอียดของ Operation TheoPrice() โดยใช้ Notes 11

รูปที่ 2.11กระบวนการพัฒนา 11

รูปที่ 3.1 แสดงการแม็ประหว่าง C++ Class กับรีเลชันนัลเทเบิล 19

รูปที่ 3.2 แสดง Object-Relational persistence framework 20

รูปที่ 4.1 Use case Diagram ของระบบทะเบียน 21

รูปที่ 4.2 Sequence Diagram ของ Use case Register for course 34

รูปที่ 4.3 Sequence Diagram ของ Use case Change Register 35

รูปที่ 4.4 Sequence Diagram ของ Use case Verify Result-grade 35

รูปที่ 4.5 Sequence Diagram ของ Use case Maintain Student Information 37

รูปที่ 4.6 Sequence Diagram ของ Use case Maintain Teacher Information 38

รูปที่ 4.7 Sequence Diagram ของ Use case Maintain Curriculum 40

รูปที่ 4.8 Sequence Diagram ของ Use case Maintain Course Information 41

รูปที่ 4.9 Sequence Diagram ของ Use case Maintain Course Information 43

รูปที่ 4.10 Sequence Diagram ของ Use case Maintain Department Information 44

รูปที่ 4.11 Sequence Diagram ของ Use case Maintain Faculty Information 46

รูปที่ 4.12 Sequence Diagram ของ Use case List Student Name 46

รูปที่ 4.13 Sequence Diagram ของ Use case Input Grade 47

รูปที่ 4.14 Sequence Diagram ของ Use case Issue Document 48

รูปที่ 4.15 Class Diagram (Main) แสดงความสัมพันธ์ต่างๆของคลาสของระบบที่ออกแบบไว้ 49

รูปที่ 4.16 แสดง sequence diagram ของ register for course function 50

รูปที่ 4.17 แสดง collaboration diagram ของ register for course function 50

รูปที่ 4.18 แสดง sequence diagram ของ change register function 51

รูปที่ 4.19 แสดง sequence diagram ของ add student function 51

รูปที่ 4.20 แสดง sequence diagram ของ update student function 52
รูปที่ 4.21 แสดง sequence diagram ของ add teacher function 52
รูปที่ 4.22 แสดง sequence diagram ของ update teacher function 53
รูปที่ 4.23 แสดง sequence diagram ของ add cirriculum function 53
รูปที่ 4.24 แสดง sequence diagram ของ update cirriculum function 54
รูปที่ 4.25 แสดง sequence diagram ของ add department function 54
รูปที่ 4.26 แสดง sequence diagram ของ update department function 55
รูปที่ 4.27 แสดง sequence diagram ของ add faculty function 55
รูปที่ 4.28 แสดง sequence diagram ของ update faculty function 56
รูปที่ 4.29 แสดง sequence diagram ของ add course function 56
รูปที่ 4.30 แสดง sequence diagram ของ update course function 57
รูปที่ 4.31 แสดง sequence diagram ของ add room function 57
รูปที่ 4.32 แสดง sequence diagram ของ update room function 58
รูปที่ 4.33 แสดง sequence diagram ของ input result grade function 58
รูปที่ 4.34 แสดง sequence diagram ของ issue document function 59
รูปที่ 4.35 แสดง sequence diagram ของ list student name function 59
รูปที่ 4.36 แสดง sequence diagram ของ verify result grade function 60
รูปที่ 4.37 Class Diagram :Interface/Register Form Class แสดงความสัมพันธ์ของคลาส Register Form กับคลาสอื่นๆ 61
รูปที่ 4.38 Class Diagram :Interface/List Student Form Class แสดงความสัมพันธ์ของคลาส List student Form กับคลาสอื่นๆ 62
รูปที่ 4.39 Class Diagram :Interface/Teacher Form Class แสดงความสัมพันธ์ของคลาส teacher Form กับคลาสอื่นๆ 63
รูปที่ 4.40 Class Diagram :Interface/Course Form Class แสดงความสัมพันธ์ของคลาส Course Form กับคลาสอื่นๆ 64
รูปที่ 4.41 Class Diagram :Interface/Cirriculum Form Class แสดงความสัมพันธ์ของคลาส Cirriculum Form กับคลาสอื่นๆ 65
รูปที่ 4.42Class Diagram :Interface/Input Grade แสดงความสัมพันธ์ของคลาส Input Grade Form กับคลาสอื่นๆ 66

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมาในการพัฒนาระบบ

ในปัจจุบันเป็นที่ยอมรับกันว่าออบเจคเทคโนโลยี (Object Technology) เป็นเทคโนโลยีที่ทำให้อุตสาหกรรมซอฟท์แวร์ มีความก้าวหน้ามากขึ้น ช่วยให้เพิ่มผลผลิต (productivity) ของการพัฒนาซอฟท์แวร์ ยังสามารถช่วยในการเพิ่มคุณภาพของซอฟท์แวร์ไปด้วย กระบวนการพัฒนาซอฟท์แวร์ที่เป็น object oriented นั้นเริ่มมีมาตรฐานในการพัฒนา มีชื่อว่า UML (Unified Modeling Language) ซึ่งจะช่วยให้ การพัฒนาระบบแบบ object oriented เป็นไปอย่างมีมาตรฐานมากขึ้น และยังมีเครื่องมือช่วยในการพัฒนาที่รองรับจากหลายบริษัททำให้การพัฒนามีความรวดเร็วขึ้น

โครงงานนี้ได้ทำการศึกษากระบวนการพัฒนาซอฟท์แวร์ที่เป็น object oriented ด้วยมาตรฐานดังกล่าว และประยุกต์ใช้กับระบบทะเบียนสารสนเทศ เพื่อที่จะเป็นแนวทางในการพัฒนาระบบเพื่อใช้งานจริงต่อไป และเป็นแนวทางในการศึกษาและพัฒนาระบบอื่นๆโดยใช้ Unified Modeling Language (UML)

1.2 วัตถุประสงค์

1. ศึกษาหลักการของแบบจำลองเชิงวัตถุ (Object Oriented Model)
2. ศึกษากระบวนการพัฒนาซอฟท์แวร์ โดยใช้ภาษาที่สนับสนุนแบบจำลองเชิงวัตถุ
3. ออกแบบระบบทะเบียนสารสนเทศ โดยใช้ UML ซึ่งเป็นภาษาที่สนับสนุนแบบจำลองเชิงวัตถุ
4. ทำการสร้างระบบงานบางส่วนที่ออกแบบไว้เพื่อทดสอบว่าสามารถจะใช้กระบวนการพัฒนาซอฟท์แวร์ โดยใช้แบบจำลองเชิงวัตถุพัฒนาระบบงานจริงได้

บทที่ 2

Unified Modeling Language (UML)

2.1 Unified Modeling Language

ในอุตสาหกรรมการพัฒนาซอฟท์แวร์ในปัจจุบันนั้น เกิดปัญหาเกี่ยวกับการล้มเหลวของโปรเจ็กต์การพัฒนาซอฟท์แวร์ เนื่องมาจากระบบที่ได้พัฒนามาไม่ตรงกับความต้องการของลูกค้า หรือมีการใช้เงิน และเวลามากเกินไป ซึ่งก็ได้มีการพยายามแก้ปัญหาโดยใช้เทคนิคใหม่ ๆ เข้ามาช่วยเพิ่ม ผลิตผลจากการพัฒนาซอฟท์แวร์ได้เพิ่มขึ้น เช่น การโปรแกรมเชิงวัตถุ (Object Oriented Programming), การโปรแกรมแบบวิชวล (Visual Programming) และ advanced development environment แต่จากการใช้เทคนิคเหล่านี้ ก็ยังคงมีปัญหาในกรณีของการใส่ใจในการโปรแกรม (programming) มากเกินไป ไม่ค่อยสนใจในเฟสอื่น ๆ ของการพัฒนาซอฟท์แวร์ การขาดความเข้าใจในโพรเซสการพัฒนาซอฟท์แวร์ ทางออกในการแก้ปัญหานี้คือการใช้ การสร้าง Abstract Model ของระบบขึ้นมา แต่การที่จะให้โปรแกรมเมอร์ ซึ่งมีความเคยชินกับการเขียนโค้ดมากกว่า แต่การสร้าง Abstract Model จะมีข้อดีดังต่อไปนี้

1. Accurate ความถูกต้องตามความต้องการจริงของระบบ
2. Consistent ทุกๆส่วนของระบบมีความสอดคล้องกัน
3. Easy to Communicate to Other ง่ายในการสื่อสารระหว่างผู้พัฒนาและผู้ใช้ระบบ
4. Easy to Change สามารถเปลี่ยนแปลงได้ง่าย
5. Understandable สามารถทำความเข้าใจได้ง่ายทั้งผู้พัฒนาและผู้ใช้ระบบ

ในการสร้าง Abstract Model นั้นมีหลายวิธีการ ซึ่งสร้างขึ้นมาเพื่อแก้ปัญหานั้น แต่ละวิธีการจะมีสัญลักษณ์ (notation) และเครื่องมือ (Tool) ที่แตกต่างกันทำให้ผู้พัฒนาเกิดความสับสน เช่น Object Oriented method มีมากมายหลายวิธี ซึ่งแต่ละวิธีก็มีข้อดีข้อเสียของแต่ละวิธีกันไป

ทางออกของอุตสาหกรรมซอฟท์แวร์จะมุ่งไปทางการสร้างแบบจำลอง (Model) ของระบบซึ่งเป็นมาตรฐาน และครอบคลุมถึงระบบทุก ๆ ระบบที่จะสร้างขึ้น และมีการรวมเอา Modeling และ Programming เข้าไว้ด้วยกัน มีความสามารถในการทำ Business Engineering ซึ่งคือความสามารถในการ Model กระบวนการทางธุรกิจ ก่อนที่จะทำการ Implement ระบบ

Unified Modeling language (UML) เป็น Abstract Model ที่มีสัญลักษณ์ที่มีประสิทธิภาพ ซึ่งมีการใช้มาตั้งแต่ ขั้นตอนการ analysis จนกระทั่งถึงการ Design

Unified Modeling language เป็น Abstract Model ที่มีความเป็นไปได้สูงที่จะกลายมาเป็นมาตรฐานทั้งทางด้าน Formal และ Defacto Standard ซึ่งก่อนหน้าที่จะมีความคิดในการสร้าง UML (Unified Modeling language) นั้นได้มีการคิดค้น Model เกี่ยวกับ Object Oriented มากมาย โดยเริ่มในปี 1990 โดยมีหลายแนวความคิดดังนี้

- Booch

- OMT The Object Modeling Technique
- OOSE / Objectory
- Fusion
- Coad / Yordon

โดยแต่ละวิธีการ (Method) จะมีเครื่องหมาย, สัญลักษณ์ในการวาด Object Model, Process (กระบวนการในแต่ละขั้นของการพัฒนาซอฟท์แวร์) และเครื่องมือที่แตกต่างกัน ซึ่งแต่ละวิธีการก็จะมีข้อดีและข้อเสียที่ต่างกันไป UML เกิดขึ้นได้โดย Garady Booch, James Rumbaugh และ Ivar Jacabsen ซึ่งพยายามสร้างวิธีการใหม่ที่เรียกว่า "Unified Method" ซึ่งจะเป็นการรวมข้อดีของแต่ละวิธีการเข้ามาใช้ด้วยกัน โดยสร้างเป็น Modeling Language มาตรฐานและตั้งชื่อว่า "Unified Modeling Language" และออก Version 1.0 มา

UML ได้รับการยอมรับโดยได้มีการรวมกลุ่มบริษัทขึ้นเป็นกลุ่มที่มีชื่อว่า "Object Management Group" (OMG) ประกอบด้วยบริษัทดังต่อไปนี้ Digital Equipment Corporation, Hewlet Packard, I-Logix, Intellicorp, IBM, Texas Instrument, ICON Computing, Microsoft, Oracle, MCI System Software และ Unisys ที่เป็นสมาชิกของกลุ่มและยอมรับมาตรฐานของ UML มาใช้

2.2 ส่วนประกอบของ UML

UML มีประโยชน์ในหลายด้าน สามารถใช้สำหรับการออกแบบในเชิงธุรกิจ, วิศวกรรมซอฟท์แวร์ (Software Engineering) ซึ่งสามารถใช้ได้ในทุกๆเฟสของการพัฒนาซอฟท์แวร์ และกับระบบทุกชนิด,การออกแบบทั่วไป ซึ่งมีทั้งส่วนที่เป็น โครงสร้างแบบสถิตย์ (static structure) และ dynamic behavior ในความสามารถหลากหลายนี้ ภาษาจะต้องถูกกำหนดให้ใช้งานได้ครอบคลุมและใช้งานได้กว้างๆ สำหรับงานออกแบบของระบบที่หลากหลายและหลีกเลี่ยงความซับซ้อนและเฉพาะเจาะจงเกินไป

UML ประกอบด้วยหลายๆส่วน ดังนี้

View : แสดงมุมมองต่างๆของระบบที่ถูกออกแบบขึ้นมา view ไม่ใช่กราฟแต่จะใช้อ้างถึง diagram ต่างๆ

Diagrams : เป็นกราฟซึ่งอธิบายส่วนต่างๆของ view ซึ่งใน UML มี diagram ที่ต่างๆกันอยู่ 9 diagrams

Model element : เป็นสัญลักษณ์ที่ใช้ใน diagram เพื่อแสดงหรือเป็นตัวแทนของสิ่งต่างๆ เช่น คลาส (class) ,ออบเจ็กต์ (object) ,แมสเซส (message) และความสัมพันธ์ (relationship) เป็นต้น

General mechanism เป็นส่วนที่แสดงคอมเม้นท์เพิ่มเติม (extra comment) , ข้อมูลอื่นๆที่จำเป็น หรือความหมายของ model element

2.2.1 View

ในการออกแบบระบบที่ซับซ้อนเป็นงานที่กว้าง ระบบทั้งหมดไม่สามารถถูกแสดงออกมาให้ชัดเจนและเข้าใจง่ายภายในกราฟเดียว เนื่องจากไม่สามารถใส่ข้อมูลทั้งหมดที่ต้องการและจำเป็นลงไปได้หมด

ระบบจะต้องอธิบายด้วยมุมมองที่ต่างๆกัน เช่น มุมมองด้าน Functional, nonfunctional, มุมมองขององค์กร เป็นต้น ดังนั้น ระบบจึงมี view ที่ต่างๆกันซึ่งแต่ละ view แสดงมุมมองเฉพาะของระบบซึ่ง

อธิบายรวมกันเป็นระบบที่สมบูรณ์ แต่ละ view ที่แสดงมุมมองต่างๆของระบบมีส่วนที่เกี่ยวข้องกันบ้าง ดังนั้น diagram หนึ่งๆสามารถเป็นส่วนหนึ่งของ view มากกว่า 1 view

View ต่างๆ ใน UML มีดังนี้

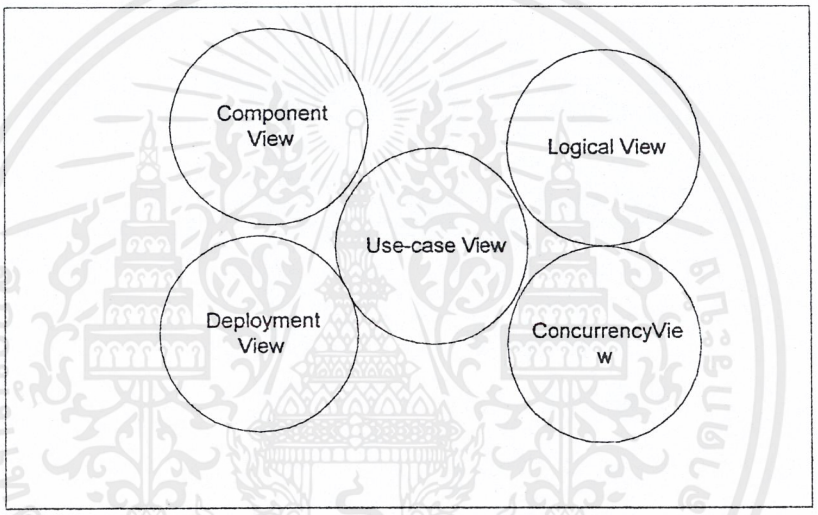
Use-case view แสดงการทำงานต่างๆของระบบที่ถูกมองจากภายนอกหรือผู้ใช้ระบบ

Logical view แสดงการทำงานต่างๆที่ถูกออกแบบไว้ภายในระบบ ทั้งที่เป็นโครงสร้างแบบสถิต และ dynamic behavior

Component view แสดงองค์ประกอบต่างๆของคอมโพเน้นท์ (component) ในส่วนของโค้ด (code)

Concurrency view แสดงการทำงานร่วมกันและการติดต่อกันของส่วนต่างๆในระบบ

Deployment view แสดงการจัดวางระบบให้เหมาะสมในด้านกายภาพ (physical) แสดงด้วย คอมพิวเตอร์ และโหนด (nodes) ต่างๆ



รูปที่ 2.1 แสดง view ต่างๆใน UML

2.2.1.1Use-case view

อธิบายการทำงานของระบบ แสดงสิ่งที่ได้รับจากระบบซึ่งมองโดยภายนอกหรือผู้ใช้ระบบซึ่งบางครั้งอาจเป็นระบบอื่นๆที่เรียกใช้บริการจากระบบนี้ Use-case view ซึ่งอธิบายโดย use-case diagram หรือบางครั้งโดย activity diagram เป็นมุมมองสำหรับลูกค้า,ผู้ออกแบบ,ผู้พัฒนาระบบและผู้ทดสอบระบบ use case ใน use-case diagram คือสิ่งที่เป็นการทำงานของระบบที่ผู้ใช้ต้องการ(function requested)

Use-case diagram เป็นตัวกำหนดเป้าหมายของระบบ จึงเป็นส่วนกลางของ view อื่นๆที่จะต้องจัดหาการทำงานต่างๆที่จะให้ระบบทำงานได้ตามนี้ และสามารถใช้ในการทดสอบระบบทั้ง validate และ verify

2.2.1.2 Logical view

ใช้อธิบายว่าจะสามารถจัดหาการทำงานของระบบให้เป็นตามที่ต้องการได้อย่างไร ซึ่งเป็นหลักที่สำคัญของผู้ออกแบบและพัฒนาระบบ Logical view จะตรงข้ามกับ use-case view เนื่องจากเป็นการ

มองภายในระบบโดยจะแสดงทั้ง โครงสร้างแบสถิต เช่น คลาส , ออบเจ็กต์, ความสัมพันธ์ (relationship) และ dynamic collaboration ซึ่งจะเกิดขึ้นเมื่อออบเจ็กต์ส่งแมสเซส (message) ระหว่างกันในการทำงาน รวมถึงลักษณะต่างๆของระบบ เช่น ความต่อเนื่อง,การทำงานร่วมกัน การเชื่อมต่อ และโครงสร้างภายในของคลาส

โครงสร้างแบบสถิตจะอธิบายโดยคลาสและออบเจ็กต์ diagram ส่วนการทำงานแบบ dynamic จะอธิบายโดย state diagram, sequence diagram, collaboration diagram และ activity diagram

2.2.1.3 Component view

Component view บอกถึง วิธีการสร้างและความขึ้นต่อกันของโมดูล (module) ซึ่งเป็นหลักของผู้พัฒนาระบบและสอดคล้องกับ component diagram ในแต่ละคอมโพเน้นท์จะประกอบด้วยโครงสร้างและความขึ้นต่อกันตลอดจนข้อมูลต่างๆเช่นความต้องการทรัพยากรของ คอมโพเน้นท์ นั้น

2.2.1.4Concurrency view

ใน Concurrency view ระบบจะถูกแบ่งออกเป็นโพรเซสย่อยๆและผู้ที่ดำเนินการกับโพรเซสนั้น โดยจุดประสงค์เพื่อตรวจสอบ nonfunctional ของระบบ เป็นการจัดการให้มีการใช้ทรัพยากรของระบบให้มีประสิทธิภาพ การทำงานแบบขนาน และการจัดการเกี่ยวกับเหตุการณ์ที่เกิดไม่พร้อมกัน โดยการแบ่งระบบเป็นส่วนที่สามารถทำพร้อมกันได้ภายใต้ thread control และจัดการเกี่ยวกับการติดต่อและความสอดคล้องกันระหว่าง thread ต่างๆนี้

2.2.1.5Concurrency view

มีไว้สำหรับผู้พัฒนาระบบและผู้ที่ทำการรวมระบบย่อยเข้าเป็นระบบใหญ่ ซึ่งจะสอดคล้องกับ dynamic diagram (ได้แก่ state diagram, sequence diagram, collaboration diagram, activity diagram) และ implementation diagram (ได้แก่ component diagram, deployment diagram)

2.2.1.6 Deployment view

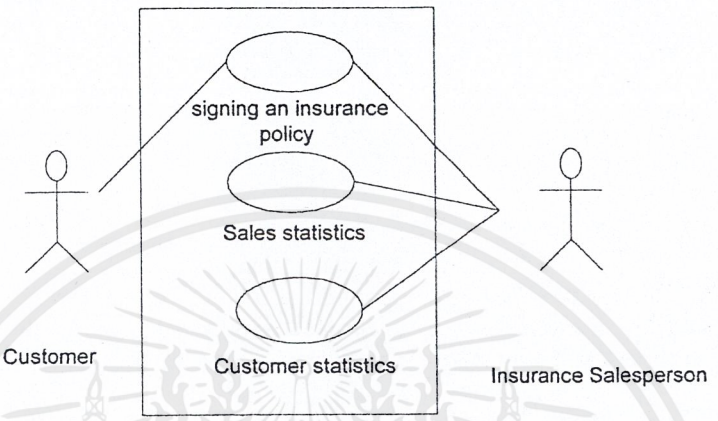
เป็นการแสดงการจัดการระบบในระดับกายภาพ (Physical) ให้เหมาะสม เช่น การเชื่อมต่อระหว่างคอมพิวเตอร์และโหนดต่างๆ และรวมการแสดงการแม็ป (map)คอมโพเน้นท์ต่างๆในระดับ physical architecture เช่น ลำดับการ execute ของ object หรือโปรแกรมบนแต่ละคอมพิวเตอร์ใช้สำหรับผู้พัฒนาระบบ, ผู้รวมระบบ,ผู้ทดสอบระบบ โดยจะแสดงโดย deployment diagram

2.2.2 Diagrams

Diagrams เป็นกราฟซึ่งแสดงสัญลักษณ์ที่ถูกจัดเรียงเพื่อแสดงระบบในมุมมองต่างๆ ในระบบหนึ่งๆจะประกอบด้วยหลายๆ diagram ซึ่งในแต่ละ diagram ยังสามารถมองในหลายๆมุมมองได้ ใน UML มี diagram ต่างๆที่หลากหลาย ดังนี้

2.2.2.1 Use-case diagram

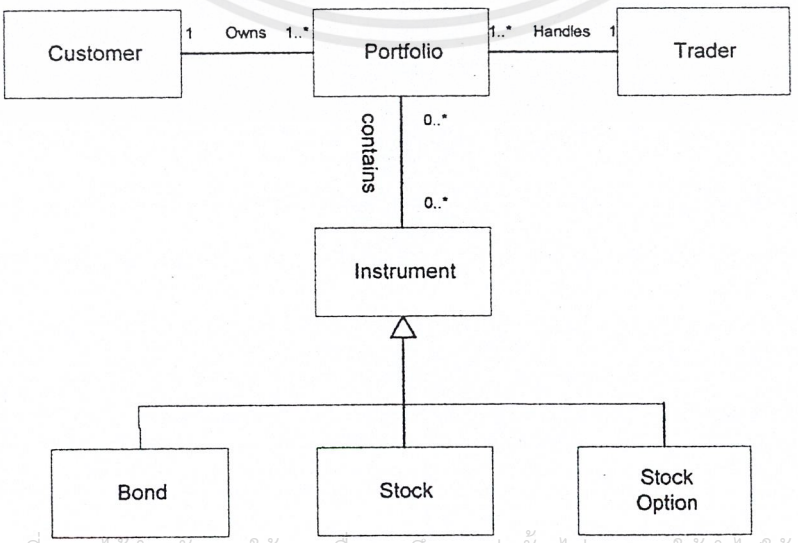
ใน Use-case diagram จะแสดงจำนวนของผู้กระทำต่อระบบซึ่งอยู่ภายนอกระบบ และแสดงการติดต่อระหว่างผู้ใช้กับระบบโดยติดต่อกับ use case ต่างๆ ซึ่งเป็นลักษณะของการใช้งานระบบที่มาจากความต้องการจากผู้ใช้หรือลูกค้า โดยอธิบายลักษณะของงานด้วยคำง่ายๆเป็น ชื่อของ use case แทนด้วยสัญลักษณ์ วงรี มีชื่อของ use case อยู่ภายใน รูปแบบของ use-case diagram แสดงดังรูปที่2



รูปที่ 2.2 Use-case diagram ของธุรกิจประกันภัย

2.2.2.2 Class diagram

จะแสดงโครงสร้างที่ไม่เปลี่ยนแปลงประกอบด้วยคลาสต่างๆในระบบ โดยคลาสจะแทนสิ่งต่างๆในระบบ ซึ่งสามารถมีความสัมพันธ์ระหว่างกันได้โดยหลายวิธี ได้แก่ association (เชื่อมต่อระหว่างกัน), dependent (การพึ่งพาหรือเรียกใช้ class อื่นๆ), specialized (ความเป็นลักษณะเฉพาะของ class อื่น), package (รวมเป็นหน่วย) ความสัมพันธ์เหล่านี้ถูกแสดงโดย class diagram โดยรวมเข้าเป็นโครงสร้างภายในของคลาสในรูปแบบของแอ็ททริบิวต์ (attribute) และ โอเปอร์เรชั่น (operation) ซึ่งในระบบหนึ่งๆจะประกอบด้วยหลายๆ class diagram



รูปที่2.3 แสดงตัวอย่าง class diagram ของการขายสินค้า

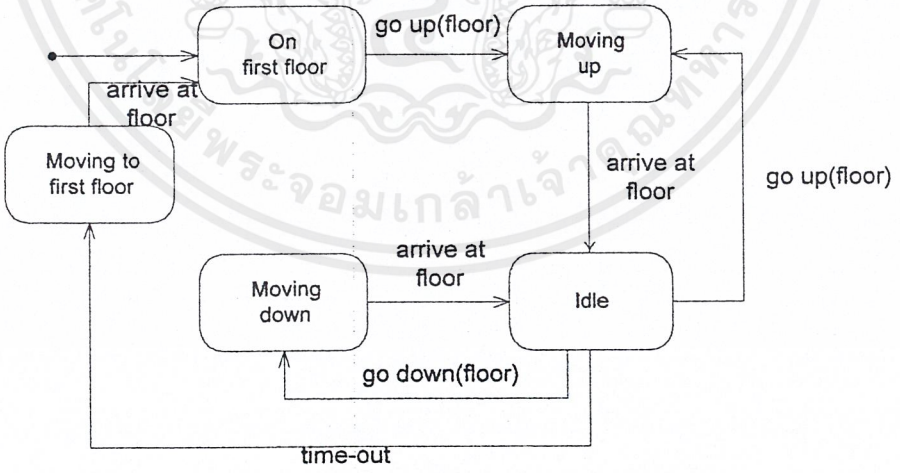
2.2.2.3 Object diagram

Object diagram แตกออกมาจาก class diagram และมีสัญลักษณ์โดยส่วนใหญ่เหมือนกัน ความแตกต่างของ 2 diagram นี้ คือ object diagram จะแสดงออบเจ็กต์ต่างๆซึ่งเป็นตัวแทนของคลาสแทนที่จะเป็นคลาสจริงๆ ดังนั้น object diagram จึงเป็นตัวแทนหรือตัวอย่างของ class diagram ซึ่งแสดงความเป็นไปได้ขณะระบบถูกใช้งาน สัญลักษณ์ต่างๆที่ใช้ใน 2 diagram จะเหมือนกัน สิ่งที่แตกต่างคือ ชื่อจะเป็นชื่อ object และขีดเส้นใต้และมีการแทนความสัมพันธ์ต่างๆ

Object diagram จะไม่มีความสำคัญเหมือน class diagram แต่มีประโยชน์ในการใช้เป็นตัวอย่างสำหรับ class diagram ที่มีความซับซ้อน โดยจะนำมาแทนเป็นออบเจ็กต์จริงๆและแสดงความสัมพันธ์ต่างๆให้เห็น และยังใช้เป็นส่วนของ collaboration diagram ซึ่งแสดงการเปลี่ยนแปลงเมื่อออบเจ็กต์ต่างๆมีการติดต่อกัน

2.2.2.4 State diagram

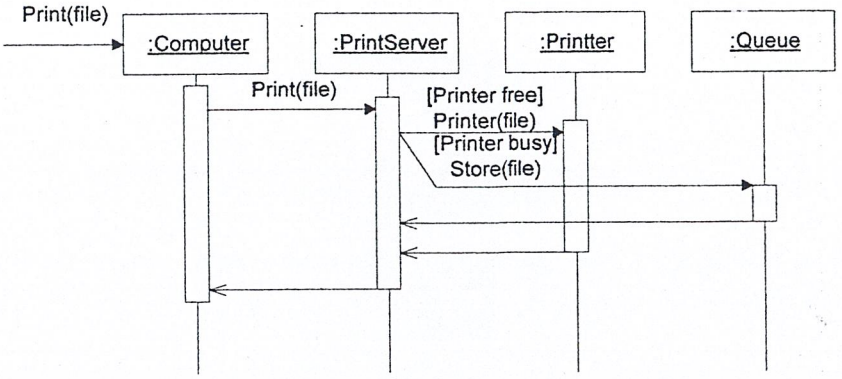
State diagram เป็นส่วนที่ใช้ประกอบคำอธิบายคลาสโดยจะแสดงทุกๆสถานะที่เป็นไปได้และเหตุการณ์ที่ทำให้ออบเจ็กต์ ต่างๆในคลาสเกิดการเปลี่ยนแปลง โดยเหตุการณ์ที่ทำให้เปลี่ยนแปลงอาจเกิดจากออบเจ็กต์อื่นๆส่งแมสเซสมา เช่น เวลาเปลี่ยนไปหรือมีเงื่อนไขใดๆเกิดขึ้น การเปลี่ยนแปลงสถานะเรียกว่าทรานซิชั่น (transition) ซึ่งทรานซิชั่นสามารถมีการกระทำ (action) เฉพาะในแต่ละทรานซิชั่น State diagram ไม่จำเป็นต้องวาดขึ้นสำหรับทุกคลาสแต่จะใช้สำหรับคลาสซึ่งมีการกำหนดสถานะไว้ชัดเจนและเมื่อการกระทำของคลาสได้รับผลหรือถูกเปลี่ยนแปลงไปในสถานะต่างๆตัวอย่างแสดงในรูปที่ 4



รูปที่ 2.4 state diagram สำหรับลิฟต์

2.2.2.5 Sequence diagram

แสดงการทำงานแบบ Dynamic ของออบเจ็กต์ต่างๆ หรือแสดงแมสเซสที่ส่งระหว่างออบเจ็กต์และการกระทำระหว่างกันของออบเจ็กต์มีลักษณะดังรูปที่ 2.5

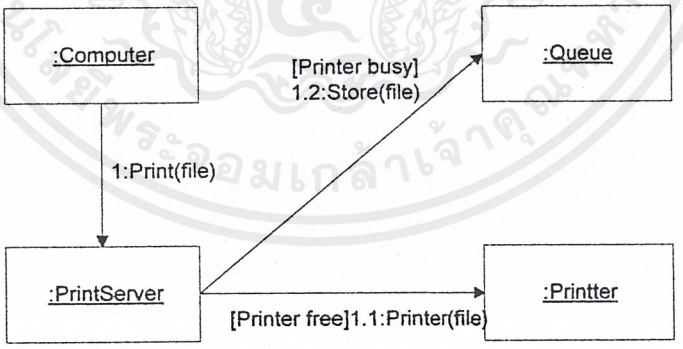


รูปที่ 2.5 sequence diagram สำหรับ Print server

2.2.2.6 Collaboration diagram

แสดงการทำงานร่วมกันแบบ Dynamic เหมือนกับ sequence diagram ในการเลือกว่าจะแสดงเป็น diagram แบบใดจะตัดสินใจโดยดูว่า ถ้าลำดับหรือเวลามีความสำคัญในการเปลี่ยนแปลงให้ใช้ sequence diagram ถ้าความสัมพันธ์หรือบางครั้งเรียกว่า context ระหว่าง object มีความสำคัญมากให้ใช้ collaboration diagram

Collaboration diagram จะวาดเหมือน object diagram โดยจะแสดงออบเจ็กต์ และความสัมพันธ์ระหว่างออบเจ็กต์ ลูกศรแสดงแมสเซสที่ส่งระหว่างออบเจ็กต์จะแสดงการไหลของแมสเซสโดยมีคำสั่งที่แมสเซส ถูกส่งกำกับ รวมทั้งเงื่อนไข,ค่าส่งกลับ โดยต้องเป็นไปตามไวยากรณ์ ซึ่งผู้พัฒนาจะต้องสามารถอ่านเข้าใจได้ ในหนึ่ง diagram สามารถมีออบเจ็กต์ที่ซึ่งถูกกระทำ (active object) พร้อมกันโดยออบเจ็กต์ที่ซึ่งถูกกระทำตัวอื่นๆ



รูปที่ 2.6 collaboration diagram สำหรับ Print server

2.2.2.7 Activity diagram

แสดงลำดับการไหลของกิจกรรม (Activity) ต่างๆ โดยจะอธิบายกิจกรรมต่างๆในลักษณะของการกระทำ (operation) ใน diagram จะแสดงเป็นสถานะกระทำ (action state) ซึ่งสถานะจะเปลี่ยนไปเมื่อ

เกิดการกระทำอย่างใดอย่างหนึ่งเกิดขึ้นตามเงื่อนไขหรือการตัดสินใจที่กำหนดไว้เพื่อควบคุมการไหลของกิจกรรมรวมถึงสามารถมีแมสเซสที่รับ-ส่งระหว่างแต่ละกิจกรรม

2.2.2.8 Component diagram

แสดงโครงสร้างทางกายภาพของโค้ดในรูปคอมโพเน้นท์ของโค้ดอาจเป็นส่วนประกอบของซอร์สโค้ด (source code component), binary component หรือ executable component

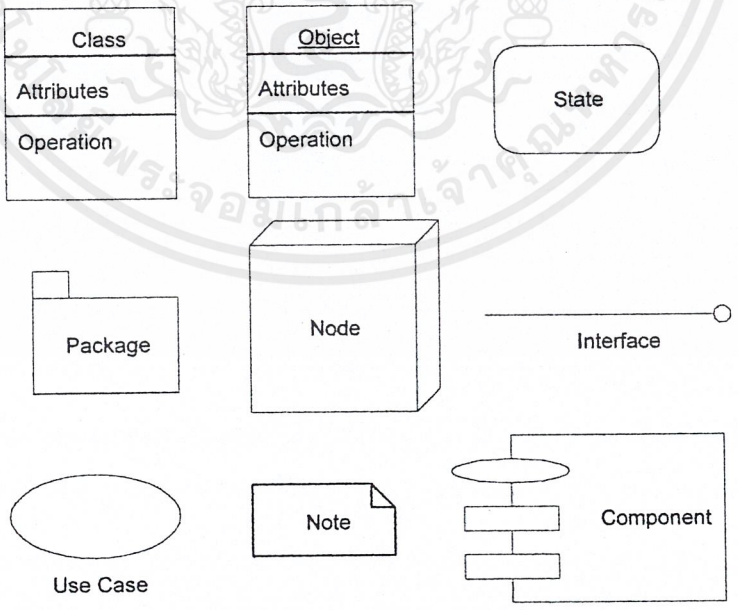
คอมโพเน้นท์จะประกอบด้วย ข้อมูลเกี่ยวกับ logical class ของ component นั้น ใน diagram มีการแสดงความสัมพันธ์หรือความพึ่งพากันของ คอมโพเน้นท์ทำให้ช่วยในการวิเคราะห์ว่าเมื่อเกิดการเปลี่ยนแปลงของ คอมโพเน้นท์หนึ่งจะมีผลต่อ คอมโพเน้นท์ อื่นๆอย่างไร ซึ่งช่วยในการโปรแกรม

2.2.2.9 Deployment diagram

แสดงสถาปัตยกรรมทางกายภาพของฮาร์ดแวร์ (Hardware) และ ซอฟท์แวร์ (software) ในระบบซึ่งสามารถแสดงเป็นคอมพิวเตอร์และโหนดและการเชื่อมต่อระหว่างกัน ชนิดของการเชื่อมต่อ นอกจากนี้ภายในโหนดยังสามารถมีคอมโพเน้นท์ หรือออบเจ็กต์ที่สามารถปฏิบัติกับโหนดเพื่อแสดงว่าโปรแกรมส่วนใดถูกปฏิบัติบนโหนดใด และความสัมพันธ์ระหว่างคอมโพเน้นท์ที่อยู่บนโหนด

2.2.3 Model element

เป็นแนวความคิดที่นำมาใช้ใน Diagram ซึ่งเป็นสัญลักษณ์ที่เป็นรูปภาพแสดงอุปกรณ์ต่างๆซึ่งมีกฎเกณฑ์ ในการใช้ของแต่ละอุปกรณ์ (element)



รูปที่ 2.7 แสดง model element ที่ใช้ร่วมกันใน diagram

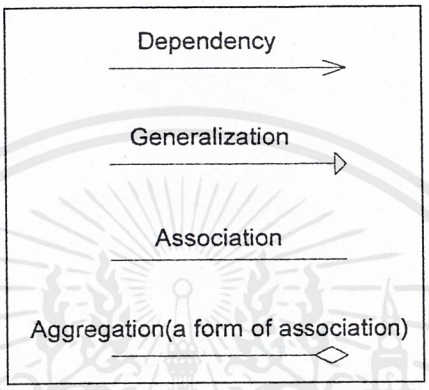
แต่ละอุปกรณ์ต้องมีความสัมพันธ์ระหว่างกันของอุปกรณ์เหล่านั้น ความสัมพันธ์ต่างๆที่มี เช่น

Association ใช้เชื่อมต่อระหว่างอุปกรณ์ที่มีความสัมพันธ์กัน

Generalization บางครั้งเรียกว่า inheritance ซึ่งเป็นความสัมพันธ์แบบพิเศษระหว่างอุปกรณ์ หนึ่งที่มีต่ออีกอุปกรณ์หนึ่ง

Dependency การพึ่งพากันระหว่างอุปกรณ์ในทางใดทางหนึ่ง

Aggregation เป็นรูปแบบการแสดงอุปกรณ์ที่ประกอบจากอุปกรณ์อื่นหลายอุปกรณ์



รูปที่ 2.8 ตัวอย่างของความสัมพันธ์แบบต่างๆ

2.2.4 General mechanisms

เป็นวิธีการที่ใช้ในการรวมเอาข้อมูลเกี่ยวกับอุปกรณ์ ต่างๆเข้าไว้ใน diagram โดยไม่สามารถแสดงไว้ได้ด้วยความสามารถพื้นฐานของโมเดลอิลิเม้นท์ (model element) มีด้วยกัน 3 วิธี

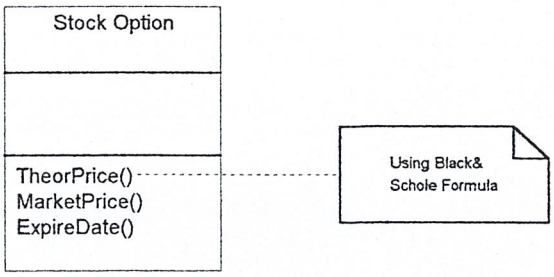
2.2.4.1 Adornment

ใช้การประดับตกแต่งในการเพิ่มความหมายเข้าไปในอุปกรณ์เช่นการแทนออบเจ็กต์ด้วยการขีดเส้นใต้ชื่อออบเจ็กต์และคลาสของออบเจ็กต์เพื่อแยกความแตกต่างระหว่างออบเจ็กต์และแทนคลาสด้วยชื่อคลาสเป็นตัวหนา เป็นต้น

รูปที่ 2.9 แสดง การเพิ่มความหมายให้กับสัญลักษณ์ของ element
โดยให้ตัวหนาแสดง class และขีดเส้นใต้แสดง object

2.2.4.2 Notes

เป็นการเพิ่มเติมข้อมูลของอุปกรณ์เข้าไว้ใน diagram notes สามารถใส่ไว้ได้ทุกๆที่ใน diagram และสามารถแสดงข้อมูลทุกชนิดได้เนื่องจากเป็นสตริง (string) ที่จะไม่ถูกแปลใน UML โดยสามารถเพิ่มเติมข้อมูลได้ด้วยเส้นประ (dash-line) ที่เชื่อมมาจากอุปกรณ์นั้น



รูป 2.10 แสดงการเพิ่มรายละเอียดของ Operation TheorPrice() โดยใช้ Note

2.2.4.3 Specification

เป็นการเก็บข้อมูลต่างๆเช่น คุณสมบัติ หน้าที่และความสามารถของอุปกรณ์นั้น โดยการเก็บจะเก็บแยกจาก diagram สนับสนุนโดยเครื่องมือ (tool)

2.3 Rational Objectory Process

บริษัทผู้ออกแบบ UML ได้พัฒนากระบวนการพัฒนา ที่เรียกว่า Rational Objectory Process ซึ่งเป็นกระบวนการในระดับ มหัพภาค (Marco) คือมีจุดประสงค์ที่จะพิจารณาทั้งการบริหารงานและงานด้านเทคนิค โดยภายในกระบวนการจะมีการทำซ้ำและค่อยๆเพิ่มขึ้นเรื่อยๆ งานที่ได้ออกมาจะมีความเปลี่ยนแปลงไม่มากนักโดยในส่วน Construction Phase จะมีการแบ่งการทำซ้ำหลายๆครั้งเพื่อให้ได้ผลตาม subset ของ requirement ของโปรเจ็กต์ โดยทั่วไปแล้วแต่ละรอบของการทำซ้ำมักจะมีการทำทั้ง analysis,design,implementation และ testing รวมกันอยู่ ดังรูป

รูปที่ 2.11 กระบวนการพัฒนา

เฟสของการพัฒนามีดังต่อไปนี้

- Inception กำหนดขอบเขตและจุดมุ่งหมายของโครงงาน
- Elaboration หาความเป็นไปได้ของโปรเจ็กต์ วางแผนในรูปของการกำหนดงานต่างๆและทรัพยากรที่ต้องการ การกำหนดฟังก์ชั่นและโครงสร้างพื้นฐาน
- Construction ทำการพัฒนาในรายละเอียด โดยทำการทำซ้ำ ทั้ง analysis ,design และ programming
- Transition ทำการส่งระบบไปยัง end users

2.3.1 Inception Phase

ขั้นตอนในการเริ่มต้นซึ่งจะมีการเก็บข้อมูลพื้นฐานเกี่ยวกับผลิตภัณฑ์ โดยจะมีการพูดถึงฟังก์ชั่นการทำงานต่าง ๆ , ความสามารถ ,ประสิทธิภาพ ,เทคโนโลยีที่ใช้ และ คุณสมบัติอื่น ๆ ใน Inception phase นั้นจะเป็นการกำหนดแนวความคิดเพิ่มเติมและแสดงวิธีที่จะใช้ในการพัฒนาในขั้นตอนต่อไป และแสดงวิธีการที่จะทำให้ระบบ มีความสามารถเพิ่มขึ้น

ผลลัพธ์ที่ได้จาก Inception Phase นั้นจะอยู่ในรูปของแผนงาน ซึ่งแสดงว่าจะต้องสร้างอะไรขึ้นมาบ้าง โดนผลลัพธ์ที่ได้นั้นจะได้จากการทำการศึกษา และกำหนดว่าจะสร้างอย่างไรและทำงานได้อย่างไร (feasibility of the project) ใน Inception phase นั้นควรจะเก็บพื้นฐานของการ วิเคราะห์ระบบ และฟังก์ชั่นการทำงานพื้นฐานและ actor ซึ่งอธิบายใน Use cases ทั้งยังมีการวางแผนด้านงบประมาณ ค่าใช้จ่าย และความสามารถในด้านการตลาด การวิเคราะห์ความเสี่ยง และผลิตภัณฑ์ของคู่แข่งด้วย

2.3.2 Elaboration Phase

Elaboration Phase จะประกอบด้วยรายละเอียดของการวิเคราะห์ระบบ การสร้างระบบโดยการกำหนดและวาง แผนก่อนทำงานในขั้นตอนต่าง ๆ ทั้งในส่วนของการทำงานและขอบเขตของปัญหา โดยการใช้ Use Case, Class diagram และ Dynamic diagram และระบบพื้นฐานของระบบ

การวางแผนในขั้นตอนต่อไปจะมีการกำหนดขั้นตอนการทำงานเป็นข้อๆ ในแต่ละการ iteration และจะมีการกำหนด schedule และ resource ต่างๆ ที่จำเป็นต้องใช้

2.3.3 Construction Phase

Construction Phase จะเป็นขั้นตอนการทำซ้ำ ๆ เป็นแบบ Iteration ซึ่งในแต่ละ Iteration จะประกอบด้วย ขั้นตอนการ analysis , design และ programming ในขั้นตอนนี้จะใช้ผลลัพธ์จากขั้นตอนก่อนหน้านี้เพื่อมาทำการกำหนดรายละเอียดลงไปในแต่ละส่วน และจะมีรายละเอียดเพิ่มขึ้นในแต่ละ Iteration ผลลัพธ์ที่ได้จาก Phase นี้จะได้คือจะได้ระบบที่ได้สร้างขึ้น

2.3.4 Transition Phase

Transition Phase ในขั้นตอนนี้จะทำการส่งผลิตภัณฑ์ไปยัง ผู้ใช้ งานจริงและรวมถึงการทำการหาตลาด packaging และ maintenance

บทที่ 3

Object Oriented Programming

3.1 การโปรแกรมเชิงวัตถุ (Object-Oriented Programming)

เนื่องมาจากการใช้วิธีการเขียนโปรแกรมแบบโครงสร้าง (Structure Programming) ซึ่งมีการใช้งานกันอย่างแพร่หลาย แต่ในปัจจุบันได้มีแนวความคิดเกี่ยวกับการเขียนโปรแกรมในวิธีการใหม่ ที่เรียกว่า การเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming) ซึ่งจะพบความสามารถในการทำงานที่มีประสิทธิภาพมากกว่าการเขียนโปรแกรมแบบเดิมเช่น มีความสามารถในการเรียกใช้ได้หลายครั้ง และมีความเชื่อถือได้สูง และยังพบว่าสามรถช่วยลดความซับซ้อนของ โปรแกรมที่มีขนาดมากกว่า 10,000 บรรทัดได้ด้วย

3.1.1 ความหมายของ การโปรแกรมเชิงวัตถุ

Object-Oriented Programming เป็นวิธีการเขียนโปรแกรมซึ่งจัดให้โปรแกรมดำเนินการกับกลุ่มของออบเจ็กต์ที่มีอยู่ในโปรแกรม

ออบเจ็กต์เป็นชนิดของข้อมูลซึ่งประกอบด้วยกลุ่มของข้อมูลและกลุ่มของฟังก์ชัน โดยการใช้ข้อมูลและฟังก์ชันที่มีอยู่เหล่านี้แต่ละออบเจ็กต์ จะทำงานหนึ่งงานได้เสร็จสมบูรณ์ การเขียนโปรแกรมเชิงวัตถุ คือการสร้างและการเรียกใช้ออบเจ็กต์ให้ทำงานตามที่เราต้องการ ในการเรียกใช้ออบเจ็กต์เราจะสนใจเฉพาะการทำงานของออบเจ็กต์จะไม่ต้องสนใจรายละเอียดภายในออบเจ็กต์

เพื่อให้การโปรแกรมเชิงวัตถุมีความสามารถตามที่กล่าวมาแล้วข้างต้น จึงต้องมีคุณสมบัติดังต่อไปนี้

- Encapsulation (การรวมข้อมูลเข้ากับฟังก์ชัน) คือการรวมกันของโครงสร้างข้อมูล (Data Structure) กับฟังก์ชันที่เกี่ยวข้องกับการเรียกใช้ข้อมูลนั้น (เรียกว่า Method หรือ Action) ทำให้เกิด Object ที่สามารถซ่อนข้อมูลจากภายนอกได้ โดยใน C++ จะเรียก Object นี้ว่า Class นั่นเอง การเข้าถึงข้อมูลภายใน class ได้มีการกำหนดระดับเอาไว้เพื่อใช้สำหรับรักษาค่าหรือข้อมูลภายใน class
- Inheritance(การสืบทอด) คุณสมบัติต่าง ๆ ของ ออบเจ็กต์ หนึ่งที่เป็นบรรพบุรุษ (ancestor) ไปยังออบเจ็กต์ที่เป็น ลูกหลาน (descendant) ได้หลาย ออบเจ็กต์ ทำให้เกิดความสัมพันธ์เรียงตามลำดับชั้น ซึ่งจะกล่าวในรายละเอียดต่อไป
- Polymorphism(การมีหลายรูปแบบ) คือการที่ ออบเจ็กต์ ต่างๆ เมื่อได้รับคำสั่งจากโปรแกรมแล้ว แต่ละออบเจ็กต์ จะทำงานตามแบบของตนเองซึ่งทำให้ได้ ผลลัพธ์ที่แตกต่างกัน

3.1.2 ลักษณะที่สำคัญของการเขียนโปรแกรมตามแนวคิดแบบออบเจ็กต์ (OOP)

3.1.2.1 การเข้าถึงข้อมูลภายในคลาส

ในการรักษาค่าหรือข้อมูลภายใน class จะมีการกำหนดระดับในการเข้าถึงข้อมูลไว้ 3 ระดับของสมาชิก 3 แบบ ดังนี้

1. การกำหนดระดับแบบ privateการประกาศตัวแปรหรือฟังก์ชั่นให้เป็นแบบ private นี้ เป็นการป้องกันไม่ให้กระบวนการใดๆที่อยู่ภายนอก class เรียกใช้ได้ จะยอมให้แต่กระบวนการที่อยู่ภายใน class ปัจจุบันเท่านั้นที่สามารถเรียกเข้าถึงสมาชิกพวกนี้ได้ การกำหนดระดับแบบ private สามารถเขียนโค้ดได้ดังนี้

```
class Cpencil                          class Cpencil{
{                                      private:
int    length;                             int    lentgh;
void   write(char *what);                  void  write(char *what);
};                                     };
```

ตัวแปร length และฟังก์ชั่น write() ได้ถูกประกาศให้เป็นแบบ private เมื่อเราสร้างออบเจ็กต์ของคลาสนี้ เราจะไม่สามารถเรียกใช้ตัวแปร length และฟังก์ชั่น write() ได้เพราะออบเจ็กต์เป็นกระบวนการที่อยู่ภายนอกคลาสดังนี้

```
Cpencil1  pencil1;
pencil1.length = 10;                    //ไม่สามารถเรียกใช้ได้
pencil1.write(10);              //ไม่สามารถเรียกใช้ได้
```

2. การกำหนดระดับแบบ public

การประกาศตัวแปรหรือฟังก์ชั่นของคลาสแบบ public นี้ จะตรงข้ามกับแบบ private นั่นคือ กระบวนการที่อยู่ภายในคลาสและภายนอกคลาสทั้งหมดรวมทั้งออบเจ็กต์ของคลาสจะสามารถเข้าถึงได้ทั้งหมด หรือเรียกอีกอย่างหนึ่งว่า การประกาศสมาชิกแบบสาธารณะ

```
class    Cpencil {
int    length;
public:
void  SetLength(int how)  {length = how;}
};
```

ถ้าต้องการกำหนดค่าให้กับตัวแปร length ภายในคลาสนี้ เราจะไม่สามารถใช้ออบเจ็กต์กำหนดค่าให้กับมันได้ เพราะตัวแปร length ยังเป็นแบบ private ออบเจ็กต์และกระบวนการภายนอกคลาสไม่สามารถเข้าถึงตัวแปรนี้ได้แต่เมื่อเราดูที่ฟังก์ชั่น Cpencil::SetLength() ซึ่งสามารถเรียกใช้ตัวแปร length ได้นั่นเป็นเพราะว่าฟังก์ชั่น SetLength() เป็นกระบวนการที่ทำอยู่ภายในคลาส

ดังนั้นเราจึงสามารถกำหนดค่าให้กับตัวแปร length ภายในคลาส Cpencil ซึ่งเป็นแบบ private ได้โดยการเรียกผ่านฟังก์ชั่น SetLength() อีกทีหนึ่ง

```
pencil1.SetLength( 10 );
```

เมื่อฟังก์ชั่น SetLength() รับค่าพารามิเตอร์ 10 เข้าไปกระบวนการภายในคลาสก็จะเริ่มขึ้น ฟังก์ชั่น SetLength() ก็จะกำหนดค่าพารามิเตอร์ที่อยู่ในตัวแปร how ให้กับตัวแปร length ที่เป็นแบบ private ได้

3.การกำหนดระดับแบบ Protected

การประกาศสมาชิกคลาสแบบ Protected นี้จะมีลักษณะเช่นเดียวกับแบบ private แต่จะต่างกันตรงที่ตัวแปรแบบ protected จะเพิ่มสิทธิให้กับสมาชิกภายในคลาสลูกสามารถเข้าถึงตัวแปรแบบ protected นี้ได้ด้วย นั่นคือกระบวนการที่เกิดขึ้นภายในคลาสลูกก็จะสามารถเข้าถึงข้อมูลแบบ protected ในคลาสแม่ได้

```
class    Cpencil{
protected:
int   length;
};
class    CpushPencil:public  Cpencil{
public:
void    UseIt() {length=10;}
};
```

จากโค้ดข้างต้นเป็นการประกาศคลาส Cpencil และประกาศตัวแปร length เป็นแบบ protected เมื่อทำการสืบทอดคลาสลูกมาเป็นคลาส CpushPencil ตัวแปร length ซึ่งเป็นแบบprotected ก็สามารถถูกเรียกใช้ได้โดยกระบวนการภายในคลาสลูกนี้ โดยใช้ฟังก์ชั่น UseIt() นั่นเอง

เมื่อเราประกาศออบเจ็กต์ของคลาส CpushPencil ออบเจ็กต์จะไม่สามารถเข้าถึงตัวแปร length ได้โดยตรง แต่สามารถเรียกใช้ฟังก์ชั่น UseIt() เพื่อกำหนดค่าให้กับตัวแปรได้เพราะฟังก์ชั่น UseIt() เป็นกระบวนการภายในของคลาส

```
CpushPencil   pencil1;
Pencil1.UseIt();
```

3.1.2.2 การสืบทอดของคลาส (Inheritance)

ลักษณะการสืบทอดคลาสหมายความว่าคลาสแต่ละคลาส(คลาสแม่) สามารถสืบทอดมาเป็นคลาสใหม่ได้ (คลาสลูก) โดยคลาสใหม่ที่สืบทอดมานี้จะยังคงมีคุณสมบัติเหมือนกับคลาสเดิมทุกประการ

```
class    Cpencil{
int       length;
void      write(char  *what);
};
class    CpushPencil:Cpencil{
};
```

จากโปรแกรมข้างต้นเป็นการสร้างคลาสใหม่ที่ชื่อ CpushPencil โดยให้คลาสใหม่นี้สืบทอดคุณสมบัติมาจากคลาส Cpencil ซึ่งสมาชิกที่อยู่ในคลาส Cpencil ก็จะปรากฏอยู่ในคลาสใหม่ CpushPencil ด้วยและยังสามารถเพิ่มคุณสมบัติใหม่ๆลงไปได้เช่นเพิ่มตัวแปรหรือ method ใหม่ๆเข้าไป

3.1.2.3 คอนสตรัคเตอร์และดีคอนสตรัคเตอร์(Constructor and Destructor)

การเขียนโปรแกรมแบบ OOP เราสามารถใช้คุณสมบัติของฟังก์ชั่นคอนสตรัคเตอร์และดีสตรัคเตอร์เพื่อเพิ่มความยืดหยุ่นให้กับโปรแกรมได้ ในการสร้างคลาสแต่ละครั้งเราไม่จำเป็นต้องสร้างคอนสตรัคเตอร์และดีสตรัคเตอร์ทุกครั้งไป การสร้างคอนสตรัคเตอร์และดีสตรัคเตอร์ขึ้นอยู่กับความจำเป็นในแต่ละสถานการณ์และการดำเนินงานของคลาส

ในคลาสหนึ่งๆสามารถมีคอนสตรัคเตอร์หรือมีดีสตรัคเตอร์เพียงตัวใดตัวหนึ่งก็ได้หรือมีทั้งสองตัวก็ได้ หรือไม่มีทั้งสองตัวก็ได้ หน้าที่ของฟังก์ชั่นคอนสตรัคเตอร์และดีสตรัคเตอร์ ก็คือ

- **ฟังก์ชั่นคอนสตรัคเตอร์** เป็นฟังก์ชั่นที่จะถูกเรียกให้ทำงาน เมื่อมีการสร้างออบเจ็กต์ของคลาสเกิดขึ้น
- **ฟังก์ชั่นดีสตรัคเตอร์** เป็นฟังก์ชั่นที่จะถูกเรียกให้ทำงาน เมื่อสิ้นสุดการทำงานของออบเจ็กต์ในคลาสนั้นๆ เช่น จบโปรแกรมหรือมีการลบออบเจ็กต์ออกจากหน่วยความจำ เป็นต้น ลักษณะของคอนสตรัคเตอร์จะเป็นฟังก์ชั่นหนึ่งที่อยู่ภายในคลาสโดยมีชื่อฟังก์ชั่นเหมือนกับชื่อคลาส และดีสตรัคเตอร์ก็เป็นฟังก์ชั่นหนึ่งที่ใช้ชื่อเดียวกับชื่อคลาสแต่จะมีเครื่องหมาย ~ นำหน้าฟังก์ชั่น ดังตัวอย่างดังต่อไปนี้

```
class  Cpencil {
int     length;
public:
```

```
CPencil(){length=10;}
~Cpencil(){length=0;}
};
```

จากโปรแกรมในข้างต้นเป็นการประกาศคลาส Cpencil ประกอบด้วยสมาชิก length และคอนสตรัคเตอร์กับดีสตรัคเตอร์ เมื่อใดที่ออบเจ็กต์ของคลาส Cpencil นี้ถูกสร้างขึ้น ฟังก์ชั่นคอนสตรัคเตอร์ Cpencil() ก็จะถูกเรียกทันที บรรทัด length=10; ก็จะถูกกระทำ และเมื่อสิ้นสุดการทำงานของออบเจ็กต์ ฟังก์ชั่นดีสตรัคเตอร์ ~Cpencil() ก็จะถูกเรียกและบรรทัด length=0; ที่อยู่ภายในฟังก์ชั่นก็จะถูกกระทำ

นอกจากนี้คอนสตรัคเตอร์และดีสตรัคเตอร์ยังสามารถนำไปใช้ในการจองหน่วยความจำและคืนหน่วยความจำได้อีกด้วย ซึ่งเป็นที่นิยมมาก เพราะผู้ใช้ไม่จำเป็นต้องเขียนโค้ดโปรแกรมเพื่อคืนหน่วยความจำเมื่อถึงเวลาที่กำหนด โค้ดของการจองหน่วยความจำเราจะใส่ไว้ที่คอนสตรัคเตอร์และโค้ดของการคืนหน่วยความจำเราจะใส่ไว้ที่ดีสตรัคเตอร์

```
class      Cpencil{
Cobject   *I;
Public:
Cpencil() {i = new Cobject(); }
~Cpencil() {delete i; }
};
```

การโอเวอร์โหลดคอนสตรัคเตอร์(Constructor Overloading)

โอเวอร์โหลดใน OOP คือการที่มีฟังก์ชั่นคอนสตรัคเตอร์หลายๆตัวใน 1 คลาส ซึ่งปกติแล้วเราอาจคิดว่าในคลาสหนึ่งๆจะสามารถมีฟังก์ชั่นคอนสตรัคเตอร์ได้เพียงตัวเดียว แต่จริงๆแล้วเราสามารถสร้างฟังก์ชั่นคอนสตรัคเตอร์ขึ้นมาหลายตัว โดยที่ใช้ชื่อเดียวกัน แต่จะต่างกันตรงที่การกำหนดพารามิเตอร์และค่าที่รีเทิร์นให้กับฟังก์ชั่น เพื่อความยืดหยุ่นในการใช้งานคลาส ลักษณะนี้เราเรียกว่า การโอเวอร์โหลดฟังก์ชั่นคอนสตรัคเตอร์

ฟังก์ชั่นเวอร์ชวล(Virtual Function)

ฟังก์ชั่นเวอร์ชวลเป็นฟังก์ชั่นหนึ่งที่คลาสแม่อนุญาตให้คลาสลูกประกาศใช้ชื่อและรูปแบบฟังก์ชั่นที่เหมือนกัน เช่น โค้ดโปรแกรมต่อไปนี้ แสดงการประกาศคลาส Cpencil และฟังก์ชั่นเวอร์ชวล Write()

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
class   CPencil{
public:
```

```
visual  void Write(){printf("pencil write\n");}
};
class  CpushPencil:public  Cpencil{
public:
void Write(){printf("PushPencil write\n");}
};
```

คลาสลูก CpushPencil ก็มีการประกาศฟังก์ชั่น Write() โดยใช้รูปแบบของฟังก์ชั่นที่เหมือนกันกับในคลาสแม่ทุกอย่าง ในเวลาใช้งานคลาส เมื่อเราสร้างออบเจ็กต์ของคลาส CpushPencil และเรียกใช้ฟังก์ชั่น Write() เราจะต้องเขียนโค้ดโปรแกรมดังนี้

```
CpushPencil  p1;
P1.Write();
```

ในการประกาศฟังก์ชั่นเวอร์ชวลนี้ จะช่วยให้สามารถสร้างฟังก์ชั่นที่มีชื่อและรูปแบบซ้ำกันกับฟังก์ชั่นในคลาสแม่ได้ โดยใส่คำว่า virtual นำหน้าฟังก์ชั่นที่ต้องการในคลาสแม่ เราเรียกการประกาศในลักษณะนี้ว่า การโอเวอร์ไรดิ้งฟังก์ชั่น(Function Overriding) หรือการเขียนฟังก์ชั่นซ้ำนั่นเอง

3.2 การรวมกันระหว่างการโปรแกรมเชิงวัตถุกับข้อมูลเชิงสัมพันธ์

ในปัจจุบันเป็นที่ยอมรับกันถึงความสามารถในการจัดการฐานข้อมูลของข้อมูลเชิงสัมพันธ์ (Relational Database) และความสามารถของการโปรแกรมเชิงวัตถุ (Object Oriented Programming) การรวมความสามารถของทั้งสองวิธีการจะทำได้โดยการรวมโมเดลเชิงวัตถุ (Object Model) และโมเดลเชิงสัมพันธ์ (Relational Model) แต่การที่จะรวมทั้งสองอย่างนี้มีข้อแตกต่างกันระหว่างโมเดลเชิงวัตถุและโมเดลเชิงสัมพันธ์ ดังต่อไปนี้

3.2.1 ระบบที่เป็นโมเดลเชิงสัมพันธ์และโมเดลเชิงวัตถุ

ระบบที่ใช้ข้อมูลเชิงสัมพันธ์ (Relational Database System)

- ข้อมูลถูกเก็บในรูปแบบของตาราง 2 มิติ
- ความสัมพันธ์ระหว่างข้อมูลจะแสดงในรูปของตาราง
- ใช้ SQL ในการสร้างความสัมพันธ์ระหว่างข้อมูล
- ความสัมพันธ์ระหว่างเอ็นติตี้ไม่จำเป็นต้องกำหนดตอนสร้างฐานข้อมูล (Database)

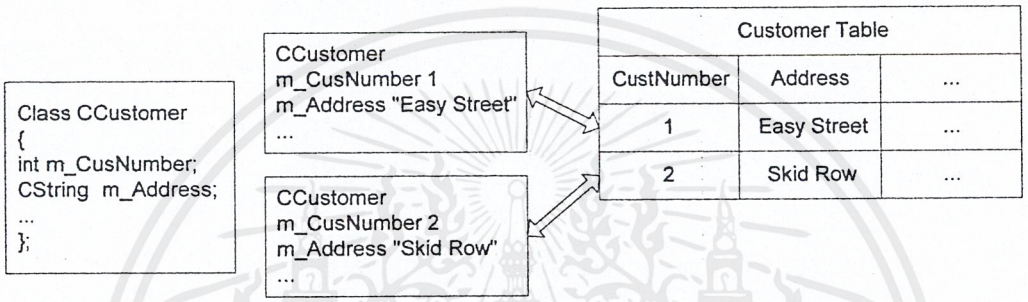
ระบบเชิงวัตถุ (Object Oriented System)

- ข้อมูลจะอยู่ในรูปของออบเจ็กต์ที่เก็บทั้งข้อมูลและกระบวนการที่กระทำกับข้อมูล (Method)
- ความสัมพันธ์ระหว่างออบเจ็กต์มีความซับซ้อน เช่น มีคุณสมบัติของการสืบทอด (Inheritance) การซ่อนข้อมูล (Encapsulation)
- ออบเจ็กต์จะเป็นชนิดของข้อมูล (Data Type) ที่กำหนดโดยโปรแกรมเมอร์เมื่อมีการสร้างระบบ

- แต่ละออบเจ็กต์มักจะถูกเก็บในคอลเลกชั่น (Collection) เพื่อทำให้ใช้งานง่าย

3.2.2 ความแตกต่างกันของดาต้าเอ็นติตี้ (Data Entities)

ในการโปรแกรมเชิงวัตถุนั้นดาต้าเอ็นติตี้คือ คลาส และ ดาต้าเมมเบอร์ แต่ใน โมเดลเชิงสัมพันธ์นั้น ดาต้าเอ็นติตี้คือ ฟิลด์และตาราง ดังแสดงในรูป 3.1 เป็นการแม็ประหว่างโมเดลเชิงวัตถุและโมเดลเชิงสัมพันธ์ คลาสในโมเดลเชิงวัตถุ จะเปรียบเทียบได้กับ รีเลชันนัลเทเบิล ของโมเดลเชิงสัมพันธ์ และดาต้าเมมเบอร์ (Data member) ของคลาสจะเทียบได้กับฟิลด์ในเทเบิล และแต่ละ Instance ของคลาสจะเกี่ยวข้องกับเรคอร์ดในเทเบิล



รูป 3.1 แสดงการแม็ประหว่าง C++ Class กับรีเลชันนัลเทเบิล

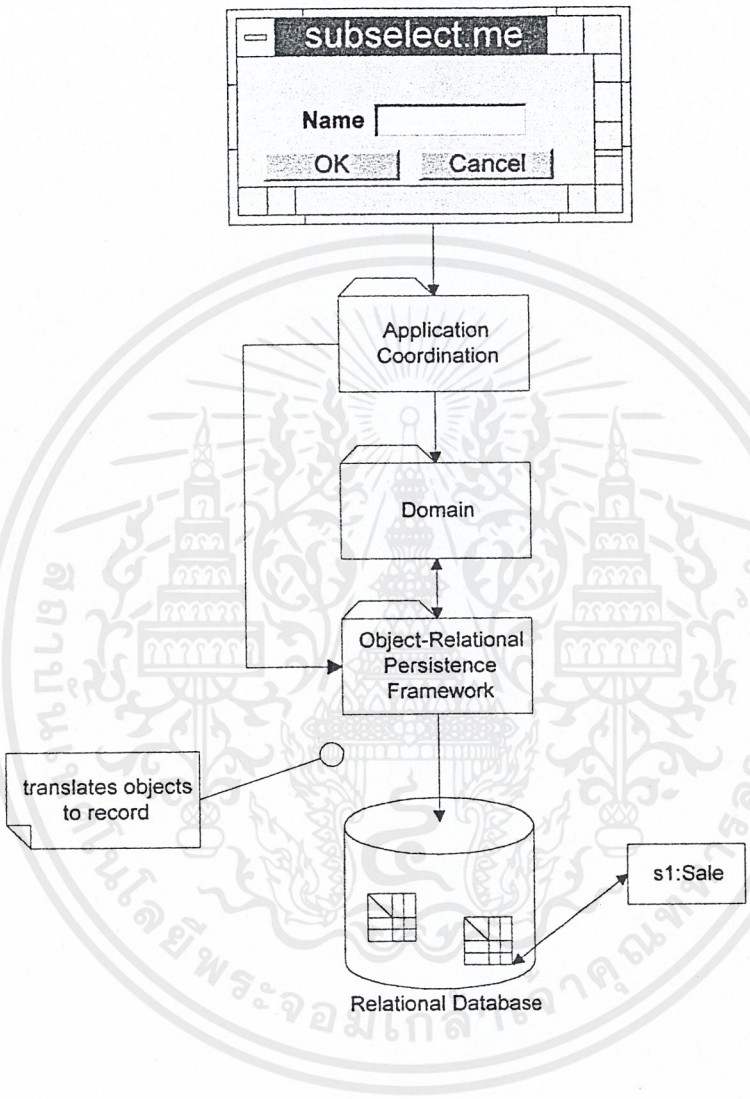
3.2.3 เทคนิคการรวมข้อมูลเชิงสัมพันธ์กับการโปรแกรมเชิงวัตถุด้วยภาษา C++

3.2.3.1 แม็ปตารางไปเป็นคลาส

การแม็ประหว่างคลาสและตารางในฐานข้อมูลและการแม็ประหว่าง ออบเจ็กแอททริบิวต์(Object Attribute) และฟิวด์ในเรคอร์ด นั้นจะมีการ ใช้ตัวชี้ออบเจ็กต์ (Object Identifier) เพื่อให้ง่ายในการอ้างอิงระหว่างเรคอร์ดกับออปเจกต์ เพื่อให้ไม่เกิดความซ้ำซ้อนระหว่าง เรคอร์ดกับออบเจกต์โดยในทางปฏิบัติแล้วเราจะใช้ คีย์หลัก (Primary key) ในตาราง เป็นตัวชี้ออบเจ็กต์ ดังรูป 3.1 แสดงการแม็ประหว่าง C++ Class กับรีเลชันนัลเทเบิล ในการแม็ปเราจะใช้ CustNumber ซึ่งเป็นคีย์หลักในตาราง Customer เป็นตัวชี้ออบเจ็กต์คือดาต้าเมมเบอร์ m_CusNumber ในแต่ละออบเจ็กต์อินสแตนท์

3.2.3.2 การสร้างออบเจ็กต์-รีเลชันนัล เพอร์ซิสแทนส์เฟรมเวิร์ก (Object-Relational persistence framework)

การสร้างเฟรมเวิร์ก (sub-system เพื่อให้บริการที่เกี่ยวข้องกัน) โดยเฟรมเวิร์กสามารถนำกลับมาใช้ได้ใหม่ (Reusable)และสามารถขยาย (Extendable) การสร้างเฟรมเวิร์กดังแสดงในรูป 3.2 โดยเพอร์ซิสแทนส์เฟรมเวิร์ก จะประกอบด้วยรูทีนต่างๆเพื่อทำการแปลงระหว่างออบเจ็กต์ให้เป็นเรคอร์ดในตารางเพื่อใช้ในการเก็บข้อมูล และแปลงจากเรคอร์ดไปเป็นออบเจ็กต์เพื่อที่จะนำข้อมูลกลับออกมา



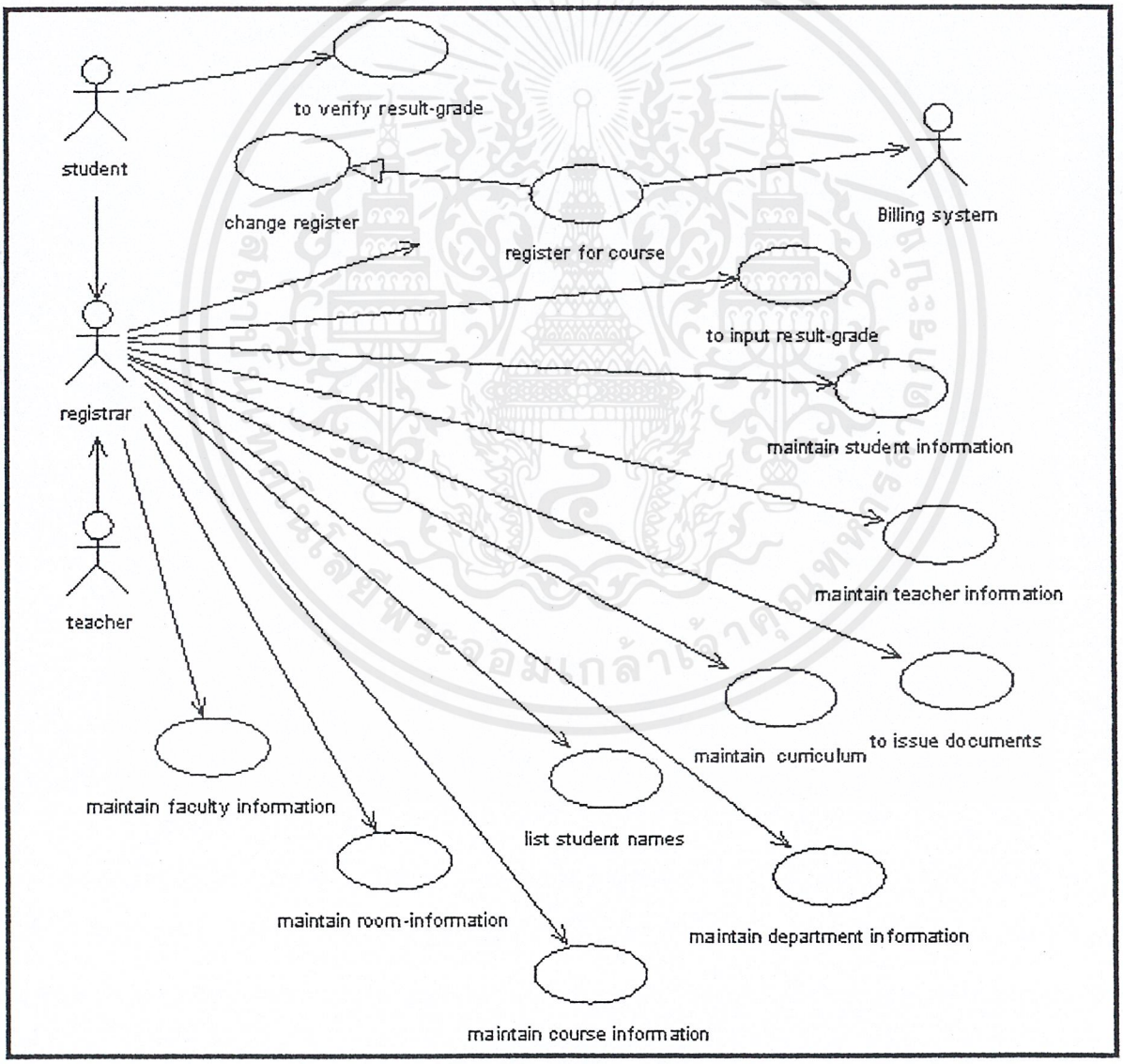
รูป 3.2 แสดง Object-Relational persistence framework

บทที่ 4
การดำเนินงานและการพัฒนาระบบ

4.1 ศึกษาและวิเคราะห์ระบบ

ทำการศึกษาการทำงานของระบบ รวบรวมความต้องการของระบบและแสดงในรูปของ Use-case Diagram โดยการกำหนดการทำงานการทำงานของระบบแสดงโดย use-case ซึ่งแต่ละ use-case จะแสดงถึงการทำงานของระบบเพื่อให้บรรลุเป้าหมายหนึ่งๆ (goal) และสิ่งที่ติดต่อกับระบบได้แก่ end user หรือระบบอื่นๆซึ่งอยู่ภายนอก และกำหนดความสัมพันธ์ระหว่าง actor กับ use case ,use case กับ use case แสดงโดย use case diagram

4.1.1 Use-case Diagram ของระบบทะเบียนที่ทำการพัฒนาระบบ



รูปที่ 4.1 Use-case Diagram ของระบบทะเบียน

แต่ละ use case สามารถมีคำอธิบาย อยู่ในรูปของ document file ซึ่งเชื่อมต่อกับ use case โดย document จะแสดงการไหลของเหตุการณ์ต่างๆ (flow of events) ที่เกิดขึ้นภายใน use case ดังตัวอย่างต่อไปนี้

1.0 Flow of Event for the register for course Use Case

1.1 Preconditions

จะต้องมีการสร้าง time table ก่อนที่จะทำ register for course

1.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะ prompt ให้เลือกใส่ student ID (E-2) และข้อมูลปีการศึกษา,เทอมที่จะลง ระบบจะแสดงหน้าจอลงทะเบียน โดยแสดงข้อมูลเกี่ยวกับตัวนักศึกษา ฟิลด์ให้กรอกรายละเอียดวิชาและ list วิชาที่ลงทะเบียน

-การเพิ่มวิชาที่จะลงทะเบียน ทำตาม S-1

-การลบวิชาที่ลงทะเบียนผิดพลาด ทำตาม S-2

วิชาที่ถูกเลือกจะแสดงไว้ใน list ของวิชาที่เลือก เมื่อเลือกเสร็จแล้วทำการยืนยันการลงทะเบียน (E-3) ตอบตกลงเพื่อกลับไปที่หน้าจอหลักการลงทะเบียน เริ่ม use case นี้อีกครั้ง

1.3 Subflows

S-1: กรอกรหัสวิชาที่ต้องการลงทะเบียนระบบจะแสดงรายละเอียดของวิชานั้นในฟิลด์ต่างๆ กดปุ่มเพิ่มวิชา รายวิชานั้นจะปรากฏใน list

S-2: click ที่วิชาต้องการลบออกจาก list ระบบจะแสดงรายละเอียดของวิชานั้น กดปุ่มลบวิชา วิชานั้นจะถูกลบออกจาก list

1.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: student ID ผิด ให้ใส่ใหม่

E-3: หากมีวิชาซ้ำซ้อนหรือหน่วยกิตรวมเกิน 22 หน่วยกิต ทำการแก้ไข

E-4: ใส่รหัสวิชาผิด ให้ใส่รหัสวิชาใหม่

2.0 Flow of Event for the **change register** *Use Case*

2.1 Preconditions

จะต้องมีการลงทะเบียน (Register for course) เสร็จก่อนที่จะมีการเปลี่ยนแปลงได้ ซึ่งจะต้องทำภายในระยะเวลาที่กำหนดไว้ (ยกเว้นกรณีพิเศษ)

2.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะ prompt เพื่อรับ student ID (E-2) ของนักศึกษาที่จะทำการเปลี่ยนแปลงวิชาที่ลงทะเบียนไปแล้ว กดปุ่มเปลี่ยนแปลงการลงทะเบียน ระบบจะแสดงหน้าจอการเปลี่ยนแปลงการลงทะเบียน ประกอบด้วยรายละเอียดของนักศึกษา list ของวิชาที่นักศึกษาได้ทำการลงทะเบียนไปแล้ว และส่วนของฟิลด์ต่างๆแสดงรายละเอียดของวิชา ให้เลือกทำการเปลี่ยนแปลงวิชาที่ลงทะเบียน ดังนี้

- เพิ่มวิชาเรียน ทำตาม S-1
- ถอนวิชาเรียน ทำตาม S-2
- เปลี่ยนวิชาเรียน ทำตาม S-3

เมื่อทำการเปลี่ยนแปลงการลงทะเบียนเรียบร้อยแล้วทำการยืนยันการแก้ไขที่ทำไป (E-3) ตอบตกลงเพื่อยืนยันการเปลี่ยนแปลงการลงทะเบียน กลับสู่หน้าจอหลัก เริ่ม use case นี้อีกครั้ง

2.3 Subflows

S-1: กรอกรหัสวิชาที่ต้องการลงทะเบียนเพิ่ม ระบบจะแสดงรายละเอียดของวิชานั้นในฟิลด์ต่างๆ กดปุ่มเพิ่มวิชา รายวิชานั้นจะปรากฏใน list

S-2 click ที่วิชาต้องการลบออกจาก list ระบบจะแสดงรายละเอียดของวิชานั้น กดปุ่มลบวิชา วิชานั้นจะถูกลบออกจาก list

S-3 การเปลี่ยนวิชา ให้ทำการลบวิชาที่ไม่ต้องการลงทะเบียนออกตาม S-2 และทำการเพิ่มวิชาที่ต้องการลงไปแทน ตาม S-1

2.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: student ID ผิด ให้ใส่ใหม่

E-3: หากมีวิชาซ้ำซ้อนหรือหน่วยกิตรวมเกิน 22 หน่วยกิต ทำการแก้ไข

E-4: ใส่รหัสวิชาผิดให้ใส่ใหม่

3.0Flow of Event for the **to input result-grade** *Use Case*

3.1 Preconditions

จะต้องมีการทำ list student names ก่อน

3.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะ prompt เพื่อรับรหัสวิชา (E-2) ระบบจะแสดงรายชื่อของนักศึกษาและ student ID และรอรับ grade ที่เจ้าหน้าที่จะใส่ในช่องสำหรับใส่ grade (E-3) ไปจนหมดรายชื่อของนักศึกษาที่ลงทะเบียนในวิชานั้น เริ่ม use case นี้อีกครั้ง

3.3 Subflows

-

3.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: ใส่รหัสวิชา ผิด ให้ใส่ใหม่

E-3: ใส่ invalid grade ให้ใส่ grade ที่ถูกใหม่

4.0 Flow of Event for the **to issue documents** *Use Case*

4.1 Preconditions

-

4.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะให้เลือกว่าต้องการจะออกเอกสารอะไร โดยเลือกจาก

- ออกใบเกรด ทำตาม S-1
- ออกใบ transcript ทำตาม S-2

4.3 Subflows

S-1: ระบบจะแสดงหน้าจอให้เลือกออกตามภาควิชาหรือออกตาม student ID จากนั้นก็ใส่ภาคการศึกษา (E-2) ที่จะทำการออกใบเกรด เลือกทำการพิมพ์เพื่อพิมพ์ใบเกรดออกมา

S-2: ระบบจะแสดงหน้าจอและ prompt ให้ใส่รหัสนักศึกษา (E-3) เลือกทำการพิมพ์เพื่อพิมพ์ใบ transcript ออกมา

4.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: ใส่ภาคการศึกษา ผิด ให้ใส่ใหม่

E-3: ใส่รหัสนักศึกษาผิดให้ใส่ใหม่

5.0 Flow of Event for the* maintain student history *Use Case

5.1 Preconditions

-

5.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะขึ้น prompt มาให้เจ้าหน้าที่เลือกว่าจะเพิ่มประวัติ,ตรวจสอบประวัติ,แก้ไขและลบประวัติ

- เพิ่มประวัติ ทำตาม S-1
- ดูประวัติ ทำตาม S-2
- แก้ไขและลบประวัติ ทำตาม S-3

เริ่ม use case นี้อีกครั้ง

5.3 Subflows

S-1: ระบบจะแสดงหน้าจอรับข้อมูลประวัติ โดยจะมีให้กรอกชื่อ-นามสกุล รหัสประจำตัวนักศึกษา (E-2) เพศ เชื้อชาติ สัญชาติ ศาสนา ที่อยู่ปัจจุบัน ที่อยู่ตามทะเบียนบ้าน เบอร์โทรศัพท์ และข้อมูลเกี่ยวกับการเรียนได้แก่ ภาควิชา,วิชาเอก,วิชาโท,หลักสูตร,รหัสอาจารย์ที่ปรึกษา (E-3) จากนั้นเจ้าหน้าที่จะทำการกดปุ่มตกลง

S-2: ระบบจะแสดงหน้าจอดูประวัติรอรับการเลือกการเรียงลำดับ ได้แก่

- เรียงตามคณะ ภาควิชา
- เรียงตามชั้นปี
- เรียงตามชื่อ-นามสกุล
- เฉพาะนักศึกษารหัส____(E-4)

ระบบจะแสดง list รายชื่อนักศึกษาเรียงตามที่เลือกไว้ หากเจ้าหน้าที่ต้องการดูประวัตินักศึกษาคนใดก็ทำการ double click ที่ชื่อนักศึกษา ระบบจะแสดงประวัติของนักศึกษาคนนั้นโดยที่ไม่สามารถเปลี่ยนแปลงแก้ไขได้

S-3: ระบบจะแสดงหน้าจอการแก้ไข/ลบประวัติรอรับการเลือกการเรียงลำดับ ได้แก่

- เรียงตามคณะ ภาควิชา
- เรียงตามชั้นปี
- เรียงตามชื่อ-นามสกุล
- เฉพาะนักศึกษารหัส____(E-5)

ระบบจะแสดง list ของรายชื่อนักศึกษาตามที่เลือกไว้ หากต้องการแก้ไข/ลบประวัตินักศึกษาคนใดก็ทำการ double click ที่ชื่อนักศึกษาคนนั้น ระบบจะแสดงประวัตินักศึกษาคนนั้น เจ้าหน้าที่สามารถทำการแก้ไขประวัตินักศึกษาคนนั้นได้และสามารถทำการลบ โดยกดปุ่ม ลบ ระบบจะให้ยืนยันการลบเมื่อตอบตกลงก็จะทำการลบประวัตินักศึกษาคนนั้น

5.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: student ID ผิด ให้ใส่ใหม่

E-3: เจ้าหน้าที่ใส่รหัสอาจารย์ที่ปรึกษาผิดพลาด ให้ใส่ใหม่

E-4: ใส่ student ID ที่ต้องการดูประวัติผิดพลาด ให้ใส่ใหม่

6.0 Flow of Event for the* maintain teacher history *Use Case

6.1 Preconditions

-

6.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะขึ้น prompt มาให้เจ้าหน้าที่เลือกว่าจะเพิ่มประวัติ,ตรวจสอบประวัติ,แก้ไขและลบประวัติ

- เพิ่มประวัติ ทำตาม S-1
- ดูประวัติ ทำตาม S-2
- แก้ไขและลบประวัติ ทำตาม S-3

เริ่ม use case นี้อีกครั้ง

6.3 Subflows

S-1: ระบบจะแสดงหน้าจอรับข้อมูลประวัติอาจารย์ โดยจะมีให้กรอกคำนำหน้าชื่อ,ชื่อ-นามสกุล เป็นภาษาไทยและภาษาอังกฤษ,รหัสประจำตัวอาจารย์ (E-2) ตำแหน่ง,ภาควิชา,คณะ, ที่อยู่ปัจจุบัน ที่อยู่ตามทะเบียนบ้าน เบอร์โทรศัพท์ จากนั้นเจ้าหน้าที่จะทำการกดปุ่มตกลง

S-2: ระบบจะแสดงหน้าจอดูประวัติรอรับการเลือกการเรียงลำดับ ได้แก่

- เรียงตามคณะ / ตำแหน่ง
- เรียงตามชื่อ-นามสกุล
- เฉพาะอาจารย์รหัส____(E-3)

ระบบจะแสดง list รายชื่ออาจารย์เรียงตามที่เลือกไว้ หากเจ้าหน้าที่ต้องการดูประวัติอาจารย์ท่านใดก็ทำการ double click ที่ชื่ออาจารย์ ระบบจะแสดงประวัติของอาจารย์ท่านนั้นโดยที่ไม่สามารถเปลี่ยนแปลงแก้ไขได้

S-3: ระบบจะแสดงหน้าจอการแก้ไข/ลบประวัติรอรับการเลือกการเรียงลำดับ ได้แก่

- เรียงตามคณะ / ตำแหน่ง
- เรียงตามชื่อ-นามสกุล
- เฉพาะอาจารย์รหัส____(E-4)

ระบบจะแสดง list รายชื่ออาจารย์เรียงตามที่เลือกไว้ หากเจ้าหน้าที่ต้องการดูประวัติอาจารย์ท่านใดก็ทำการ double click ที่ชื่ออาจารย์ ระบบจะแสดงประวัติของอาจารย์ท่านนั้นเจ้าหน้าที่สามารถทำการแก้ไขประวัติอาจารย์ท่านนั้นได้และสามารถทำการลบ โดยกดปุ่ม ลบ ระบบจะให้ยืนยันการลบเมื่อตอบตกลงก็จะทำการลบประวัติอาจารย์ท่านนั้น

6.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: เจ้าหน้าที่ใส่รหัสอาจารย์ที่ต้องการเพิ่มประวัติผิดพลาด ให้ใส่ใหม่

E-3: เจ้าหน้าที่ใส่รหัสอาจารย์ที่ต้องการดูประวัติผิดพลาด ให้ใส่ใหม่

E-4: เจ้าหน้าที่ใส่รหัสอาจารย์ที่ต้องการแก้ไข/ประวัติผิดพลาด ให้ใส่ใหม่

7.0 Flow of Event for the* to List Student Names *Use Case

7.1 Preconditions

จะต้องมีการลงทะเบียนและแก้ไขวิชาเรียนเสร็จเรียบร้อยแล้ว

7.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) ระบบจะแสดงหน้าจอเพื่อรอรับวิชาที่ต้องการ จะใช้รายชื่อนักศึกษา โดยจะรอรับการใสรหัสวิชา (E-2) เมื่อใส่รหัสวิชาที่ต้องการลงไปแล้ว ก็จะแสดงหน้าจอการกระทำให้เลือกดังต่อไปนี้

-แสดงรายชื่อนักศึกษาออกทางหน้าจอ ทำตาม S-1

-พิมพ์รายชื่อนักศึกษาออกทางเครื่องพิมพ์ ทำตาม S-2

7.3 Subflows

S-1:ระบบจะทำการแสดงรายชื่อนักศึกษาที่ทางหน้าจอโดยจะแสดง ชื่อนักศึกษา,รหัสนักศึกษา,ภาควิชา, คณะ ของนักศึกษาที่ลงทะเบียนในวิชาที่เจ้าหน้าที่ต้องการ และแสดงจำนวนนักศึกษาทั้งหมดที่ลงทะเบียนในวิชานั้น (E-3)

S-2:ถ้าเจ้าหน้าที่เลือกที่จะพิมพ์รายชื่อนักศึกษา ระบบจะทำการพิมพ์ ชื่อนักศึกษา,รหัสนักศึกษา, ภาควิชา, คณะ และจำนวนรวม ของนักศึกษาที่ลงทะเบียนเรียนในวิชานั้น โดยจะมีการแสดงรายละเอียดของ ชื่อวิชา,รหัสวิชา, ภาคการศึกษา ,ปีการศึกษา และอาจารย์ผู้สอนอยู่บนหัวรายชื่อด้วย

7.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: ใส่รหัสวิชา ผิด ให้ใส่ใหม่

E-3: ไม่มีนักศึกษาลงทะเบียนในรายวิชานั้น แจ้งว่าไม่มีนักศึกษาลงทะเบียน และเริ่ม UseCase ใหม่

E-4: ไม่สามารถพิมพ์ได้ ให้สั่งพิมพ์ใหม่ หรือยกเลิกการพิมพ์

8.0 Flow of Event for the* Maintain Curriculum Information *Use Case

8.1 Preconditions

8.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) ระบบจะแสดงหน้าจอ Maintain Curriculum Information ให้เจ้าหน้าที่เลือกการทำงานดังต่อไปนี้

-เพิ่มหลักสูตร ทำตาม S-1

-ลบหลักสูตร ทำตาม S-2

-แก้ไขหลักสูตร ทำตาม S-3

-เพิ่มวิชาในหลักสูตร S-4

-ลบวิชาออกจากหลักสูตร S-5

8.3 Subflows

S-1: เพิ่มหลักสูตร:ระบบจะแสดงหน้าจอเพิ่มหลักสูตร และรอเจ้าหน้าที่ใส่ข้อมูลของหลักสูตรที่ต้องการเพิ่ม ได้แก่ รหัสหลักสูตร,ชื่อหลักสูตร, รหัสคณะ, รหัสภาควิชา และทำการใส่รหัสวิชาจนครบทุกวิชาในหลักสูตร (E-2) โดยมีรายละเอียดของวิชาดังนี้ รหัสวิชา, ชื่อวิชา, กลุ่มวิชา และจำนวนหน่วยกิจ, ชั้นปี/เทอมที่เรียน และ คำอธิบายวิชา (E-3)

S-2: ลบหลักสูตร: ระบบจะแสดงหน้าจอการลบหลักสูตร และรอรับรหัสหลักสูตร (E-4) ที่ต้องการลบ ระบบจะแสดงขอมูลหลักสูตรนั้นและรอรับการยืนยันการลบ ถ้าตอบตกลงจะทำการลบหลักสูตรนั้น

S-3: แก้ไขหลักสูตร:ระบบจะแสดงหน้าจอแก้ไขหลักสูตร เพื่อรอให้เจ้าหน้าที่ทำการใส่รหัสหลักสูตร (E-5) ที่ต้องการแก้ไข เมื่อเจ้าหน้าที่ใส่รหัสหลักสูตรและตอบตกลง ระบบจะแสดงข้อมูลของหลักสูตรนั้น

และรอให้เจ้าหน้าที่ทำการแก้ไข โดยสามารถแก้ไข ข้อมูลทั่วไป ได้แก้ รหัสหลักสูตร, ชื่อหลักสูตร, รหัสคณะ, รหัสภาควิชา และสามารถเข้าไปแก้ไขในช่องต่าง ๆ ได้ (E-6) เมื่อทีการแก้ไขเสร็จสิ้น กดปุ่มตกลงที่หน้าจอหลักของการแก้ไข เพื่อกลับสู่หน้าจอแรก

S-4: ทำการเพิ่มวิชาในหลักสูตร ทำโดยเจ้าหน้าที่เพิ่มวิชาในหลักสูตร ทำโดยเจ้าหน้าที่กดปุ่มเพิ่มบนหน้าจอซึ่งให้ใส่รายละเอียดของวิชาที่จะทำการเพิ่ม รหัสวิชา, ชื่อวิชา, กลุ่มวิชา และจำนวนหน่วยกิจ, ชั้นปี/เทอมที่เรียน และ คำอธิบายวิชา ระบบจะทำการเพิ่มวิชาเข้าไปในหลักสูตร (E-7)

S-5: ทำการลบวิชาในหลักสูตร เจ้าหน้าที่จะ Click ไปในชื่อวิชาในหลักสูตรที่ต้องการจะลบ ระบบจะแสดงรายละเอียดของวิชานั้นและ เมื่อเจ้าหน้าที่ยืนยันการลบ แล้วระบบจะทำการลบวิชานั้นออกไป และกลับไปยังหน้าจอแรก

8.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: ใส่รายละเอียดของหลักสูตรผิด ให้ใส่ใหม่

E-3: ใส่รายละเอียดของวิชาผิด ให้ใส่ใหม่

E-4: ใส่รหัสของหลักสูตรผิด ให้ใส่ใหม่

E-5: ใส่รหัสของหลักสูตรผิด ให้ใส่ใหม่

E-6: ใส่ข้อมูลทั่วไปของหลักสูตรผิด ให้ใส่ใหม่

E-7: ใส่รายละเอียดของวิชาผิด ให้ใส่ใหม่

9.0Flow of Event for the* Maintain Room Information *Use Case

9.1 Preconditions

9.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะขึ้น prompt มาให้เจ้าหน้าที่เลือกว่าจะเพิ่มรายละเอียดห้องเรียน,ตรวจสอบรายละเอียดห้องเรียน,แก้ไขและลบ รายละเอียดห้องเรียน

เพิ่มรายละเอียดห้องเรียน ทำตาม S-1

ตรวจสอบรายละเอียดห้องเรียน ทำตาม S-2

แก้ไขและลบ รายละเอียดห้องเรียน ทำตาม S-3

เริ่ม UseCase นี้อีกครั้ง

9.3 Subflows

S-1:เพิ่มรายละเอียดห้องเรียน: ระบบจะแสดงหน้าจอเพื่อรอรับข้อมูลห้องเรียนซึ่งประกอบด้วย ชื่อห้อง, ชนิดของห้อง, ชื่ออาคาร, ความจุของห้องสถานที่ตั้งของห้อง และ อุปกรณ์การสอน (E-2)

S-2: ตรวจสอบรายละเอียดห้องเรียน:ระบบจะรอรับชื่อห้องเรียน (E-3) จากเจ้าหน้า หลังจากได้รับชื่อห้องเรียนแล้วจะแสดงรายละเอียดของห้องดังต่อไปนี้ชื่อห้อง, ชนิดของห้อง, ชื่ออาคาร, ความจุของห้องสถานที่ตั้งของห้อง และ อุปกรณ์การสอน

S-3: แก้ไขและลบ รายละเอียดห้องเรียน:ระบบจะรอรับชื่อห้องเรียน (E-4) จากเจ้าหน้า หลังจากได้รับชื่อห้องเรียนแล้วจะแสดงรายละเอียดของห้องดังต่อไปนี้ชื่อห้อง, ชนิดของห้อง, ชื่ออาคาร, ความจุของห้องสถานที่ตั้งของห้อง และ อุปกรณ์การสอน และสามารถทำการแก้ไขข้อมูลทุก ๆ อย่างได้ และมีปุ่มสำหรับทำการลบ ข้อมูลห้องที่แสดงอยู่ (E-5)

9.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: ใส่ข้อมูลของห้องผิด แสดง Error ที่ผิดและให้ใส่ใหม่

E-3: ใส่ชื่อห้องผิด ให้ใส่ใหม่

E-4: ใส่ชื่อห้องผิด ให้ใส่ใหม่

E-5: ใส่ข้อมูลของห้องที่ทำการแก้ไขผิด แสดง Error ที่ผิดและให้ใส่ใหม่

10.0Flow of Event for the* maintain course information *Use Case

10.1Preconditions

จะต้องทีการใส่วิชาเรียน Course อาจารย์ และ ห้อง เรียบร้อยแล้ว

10.2Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯและใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) ระบบจะรอรับการเลือกการทำงานของเจ้าหน้าที่

10.3 Subflows

S-1 : เพิ่มข้อมูลตารางสอนขตารางสอบ โดยระบบจะรอรับข้อมูลดังต่อไปนี้ รหัสวิชา,ชั้นเรียน,วัน-เวลาเรียน,ห้องเรียน,รหัสอาจารย์ผู้สอน,วันเวลาสอบ,และนักศึกษาโดยประมาณ (E-2)

S-2 : ตรวจสอบตารางการใช้ห้องเรียน : ระบบจะรอรับชื่อห้อง (E-3) และระบบจะแสดงการใช้ห้องนั้น

S-3 : ตรวจสอบตารางสอน : ระบบจะรอรับรหัสอาจารย์ (E-4) และระบบจะแสดงตารางสอนของอาจารย์

10.4Alternative Flows

E-1 : ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2 : ใส่ข้อมูลตารางสอนผิด ให้ใส่ใหม่

E-3 : ใส่ชื่อห้องผิด ให้ใส่ใหม่

E-4 : ใส่รหัสอาจารย์ผิดให้ใส่ใหม่

11.0Flow of Event for the **Maintain Faculty Information** *Use Case*

11.1 Preconditions

11.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะ แสดงหน้าจอ Faculty Information ให้เจ้าหน้าที่เลือกการทำงานดังต่อไปนี้

- เพิ่มข้อมูลคณะ ทำตาม S-1
- แก้ไข ข้อมูลคณะ ทำตาม S-2
- ลบข้อมูลคณะ ทำตาม S-3

เริ่ม use case นี้อีกครั้ง

11.3 Subflows

S-1: เพิ่มข้อมูลคณะ: ระบบจะแสดงหน้าจอ เพิ่มข้อมูล โดยหน้าจอจะแสดง รายชื่อคณะเดิม และมีช่องให้กรอกข้อมูลในกรณีที่ต้องการเพิ่มคณะ เมื่อกรอกข้อมูล รหัสคณะ(E-2), ชื่อคณะทั้งภาษาไทยและภาษาอังกฤษ ตอบตกลง ชื่อคณะใหม่จะไปอยู่ในรายชื่อคณะบนหน้าจอ ทำการยืนยันการเพิ่มข้อมูลคณะ ก่อนออกจากหน้าจอนี้

S-2: แก้ไข ข้อมูลคณะ: ระบบจะแสดงหน้าจอแก้ไขข้อมูล โดยมีรายชื่อคณะที่มีอยู่ไว้ใน listการแก้ไขให้เลื่อนแถบสีไปยังคณะที่ต้องการ กดปุ่มแก้ไข หรือ Click ที่คณะ จะปรากฏข้อมูลของคณะและสามารถทำการแก้ไขได้ ตอบตกลง ยืนยันการแก้ไขเพื่อกลับสู่หน้าจอหลัก

S-3: ลบข้อมูลคณะ: ระบบจะแสดงหน้าจอ ลบข้อมูล โดยจะแสดงรายชื่อคณะเดิม การลบข้อมูลให้เลื่อนแถบสีไปที่คณะที่ต้องการลบ กดปุ่ม enter หรือ click ที่คณะที่ต้องการ จะปรากฏข้อมูลของคณะกดปุ่มลบเพื่อทำการลบข้อมูล ทำการยืนยันการลบก่อนออกจากหน้าจอนี้

11.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

E-2: ใส่ รหัสคณะผิด ให้ใส่ใหม่

12.0 Flow of Event for the **to verify result-grade** *Use Case*

12.1 Preconditions

นักศึกษาจะต้องทำการลงทะเบียน (Register for cause usecase)

12.2 Main Flow

Use case นี้เริ่มโดยระบบจะแสดงหน้าจอดูผลการเรียนและ รอรับการใส่รหัสนักศึกษา(E-1) และระบบจะรอรับภาคการศึกษาที่ต้องการดู(E-2) ระบบจะแสดงผลการเรียนในรายวิชาที่ลงทะเบียนในเทอมนั้น และคำนวณเกรดในเทอมนั้นและ GPA (คิดถึงเทอมปัจจุบัน)(E-4)

12.3 Subflows

12.4 Alternative Flows

E-1: ใส่รหัสนักศึกษาผิด ให้ใส่ใหม่

E-2: ใส่ภาคการศึกษาผิด ให้ใส่ใหม่

E-3: กรณีที่ ผลการเรียนของภาคการศึกษาล่าสุดยังไม่เรียบร้อย ระบบจะแสดงเฉพาะผลการเรียนที่ผ่านการพิจารณาแล้ว และเกรดเฉลี่ย คิดเฉพาะผลการเรียนที่ผ่านการพิจารณาแล้ว

13.0Flow of Event for the **Maintain Department Information** *Use Case*

13.1 Preconditions

13.2 Main Flow

Use case นี้เริ่มเมื่อเจ้าหน้าที่ log on มายังระบบทะเบียนฯ และใส่ password ระบบจะตรวจสอบว่า password ถูกต้อง (E-1) จะ แสดงหน้าจอ department Information ให้เจ้าหน้าที่ใส่รหัสของคณะที่ต้องการเปลี่ยนแปลงข้อมูลภาควิชา (E-2) และเลือกการทำงานดังต่อไปนี้

- เพิ่มข้อมูลภาควิชา ทำตาม S-1
- แก้ไข ข้อมูลภาควิชา ทำตาม S-2
- ลบข้อมูลภาควิชา ทำตาม S-3

เริ่ม use case นี้อีกครั้ง

13.3 Subflows

S-1: เพิ่มข้อมูลภาควิชา: ระบบจะแสดงหน้าจอ เพิ่มข้อมูล โดยหน้าจอจะแสดง รายชื่อภาควิชาเดิม และมีช่องให้กรอกข้อมูลในกรณีที่ต้องการเพิ่มภาควิชา เมื่อกรอกข้อมูล รหัสภาควิชา, ชื่อภาควิชาทั้งภาษาไทยและภาษาอังกฤษ ตอบตกลง ชื่อภาควิชาใหม่จะไปอยู่ในรายชื่อภาควิชาบนหน้าจอ ทำการยืนยันการเพิ่มข้อมูลภาควิชา ก่อนออกจากหน้าจอนี้

S-2: แก้ไข ข้อมูลภาควิชา: ระบบจะแสดงหน้าจอแก้ไขข้อมูล โดยมีรายชื่อภาควิชาที่มีอยู่ไว้ใน listการแก้ไขให้เลื่อนแถบสีไปยังภาควิชาที่ต้องการกดปุ่มแก้ไข หรือ Click ที่ภาควิชา จะปรากฏข้อมูลของภาควิชาและสามารถทำการแก้ไขได้ ตอบตกลง ยืนยันการแก้ไขเพื่อกลับสู่หน้าจอหลัก

S-3: ลบข้อมูลภาควิชา: ระบบจะแสดงหน้าจอ ลบข้อมูล โดยจะแสดงรายชื่อภาควิชาเดิม การลบข้อมูลให้เลื่อนแถบสีไปที่ภาควิชาที่ต้องการลบ กดปุ่ม enter หรือ click ที่ภาควิชาที่ต้องการ จะปรากฏข้อมูลของภาควิชากดปุ่มลบ เพื่อทำการลบข้อมูล ทำการยืนยันการลบก่อนออกจากหน้าจอนี้

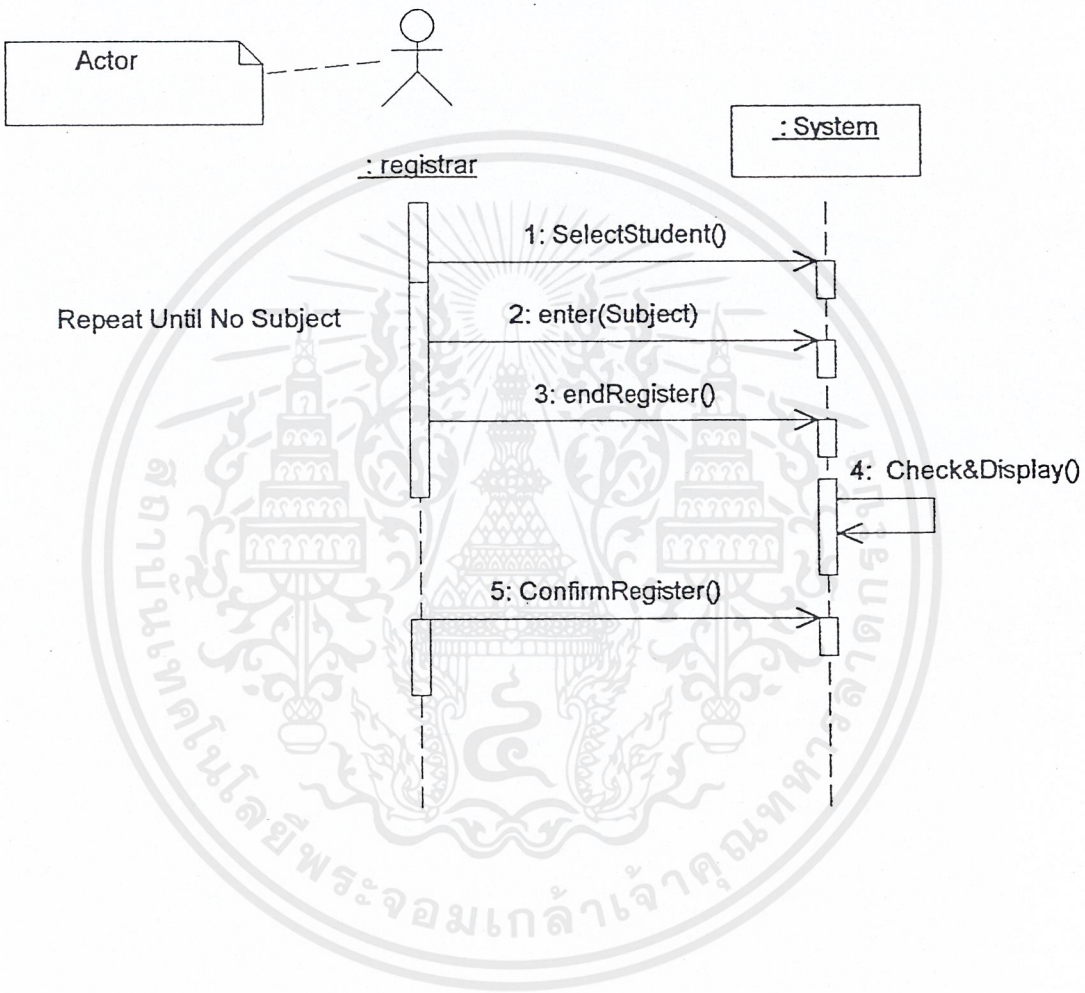
13.4 Alternative Flows

E-1: ใส่ password หรือ ID ผิด ให้ใส่ใหม่

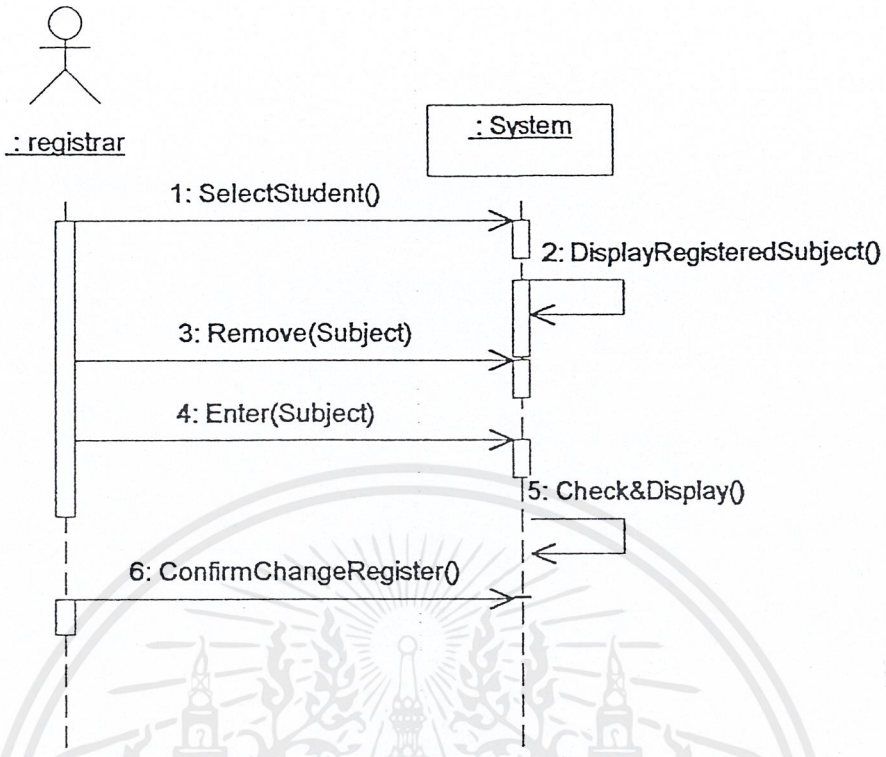
E-2: ใส่รหัสคณะผิด ให้ใส่ใหม่

4.1.2 Sequence Diagrams

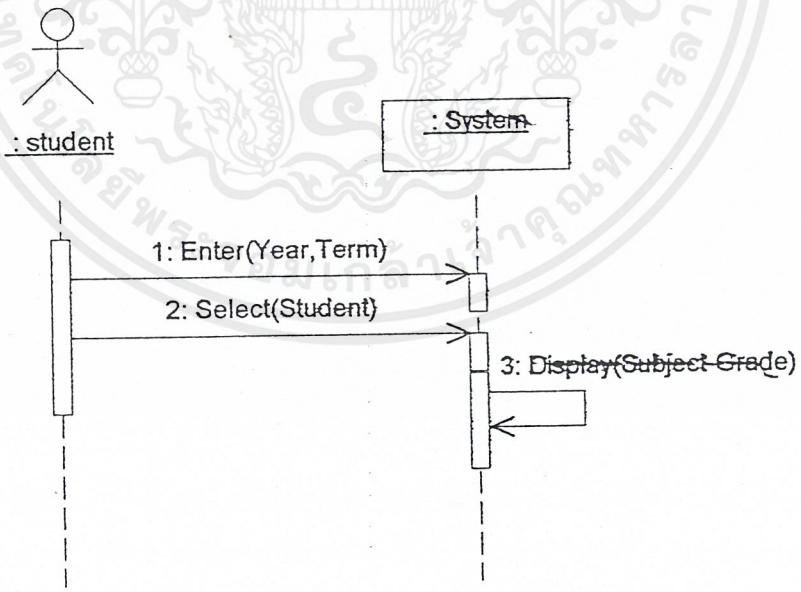
Sequence diagram ในขั้นนี้จะมองระบบเป็น black block และอธิบายว่า actor จะทำอะไรกับระบบได้บ้าง ซึ่งสอดคล้องกับ use case diagram



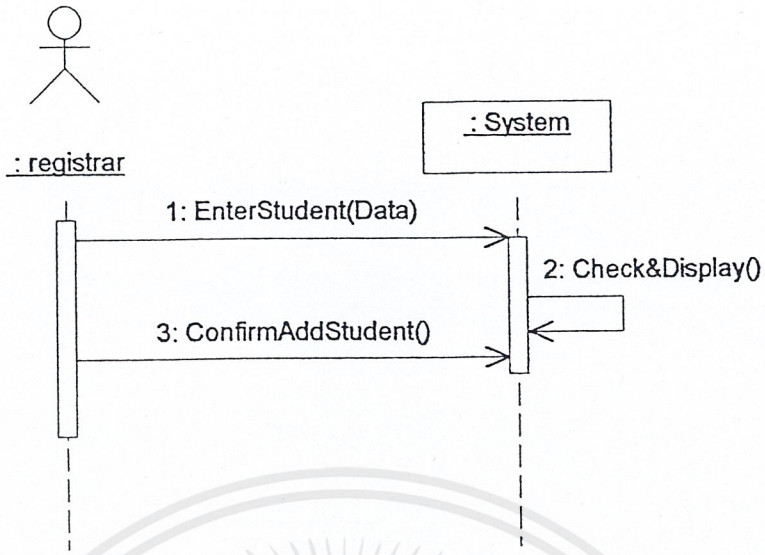
รูปที่ 4.2 Sequence Diagram ของ Use case Register for course

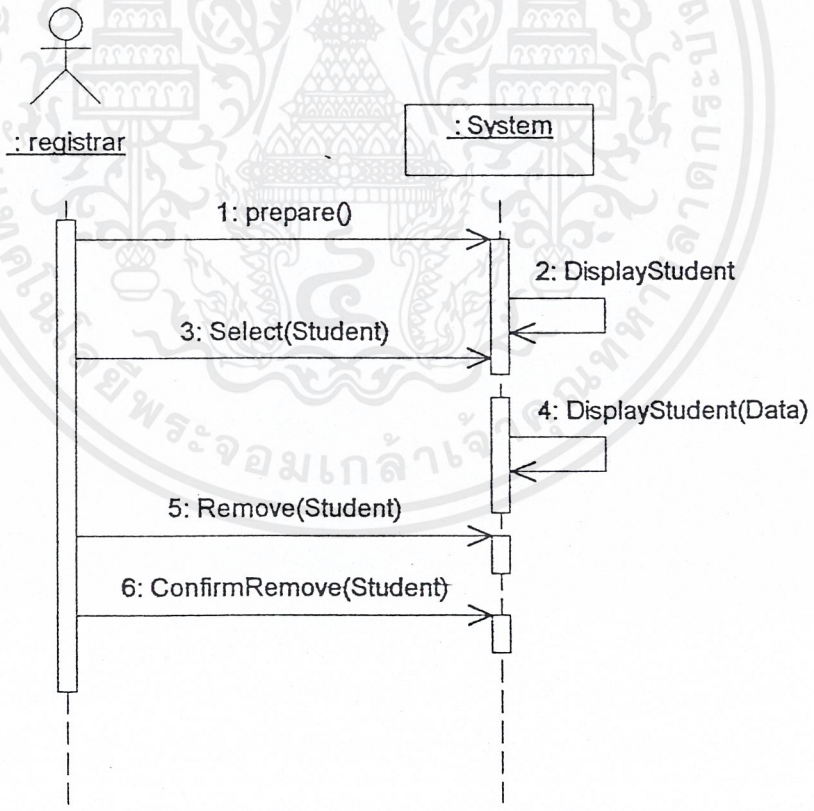


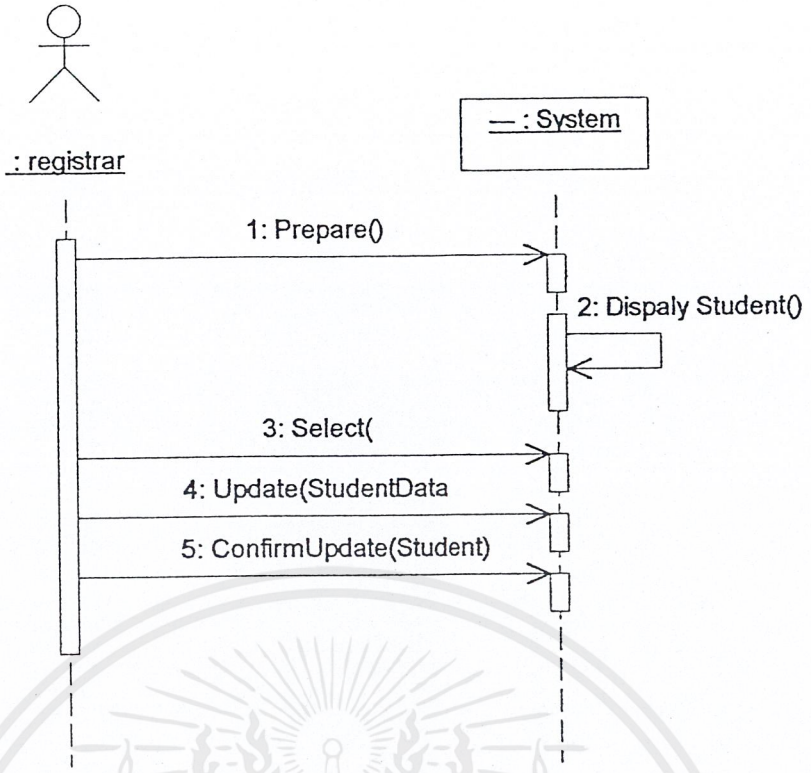
รูปที่ 4.3 Sequence Diagram ~~ที่~~ Use case Change Register



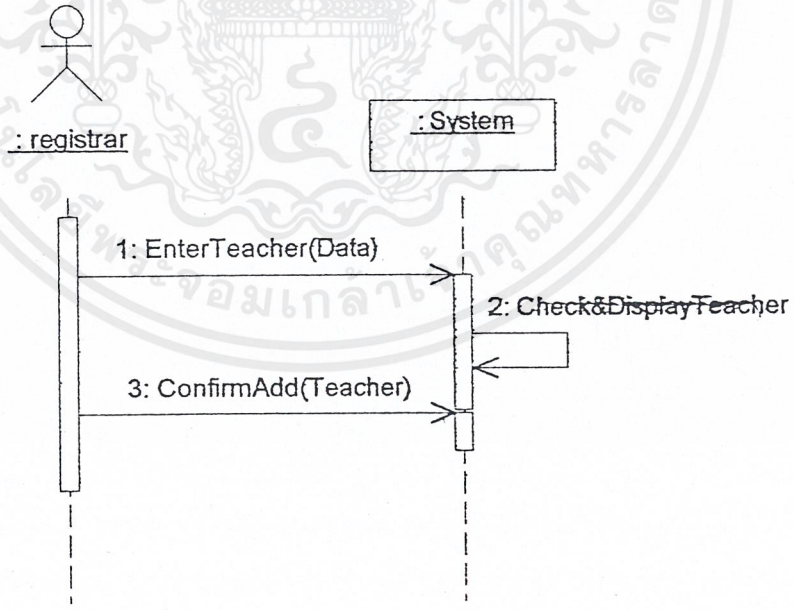
รูปที่ 4.4 Sequence Diagram ~~ที่~~ Use case Verify result-grade

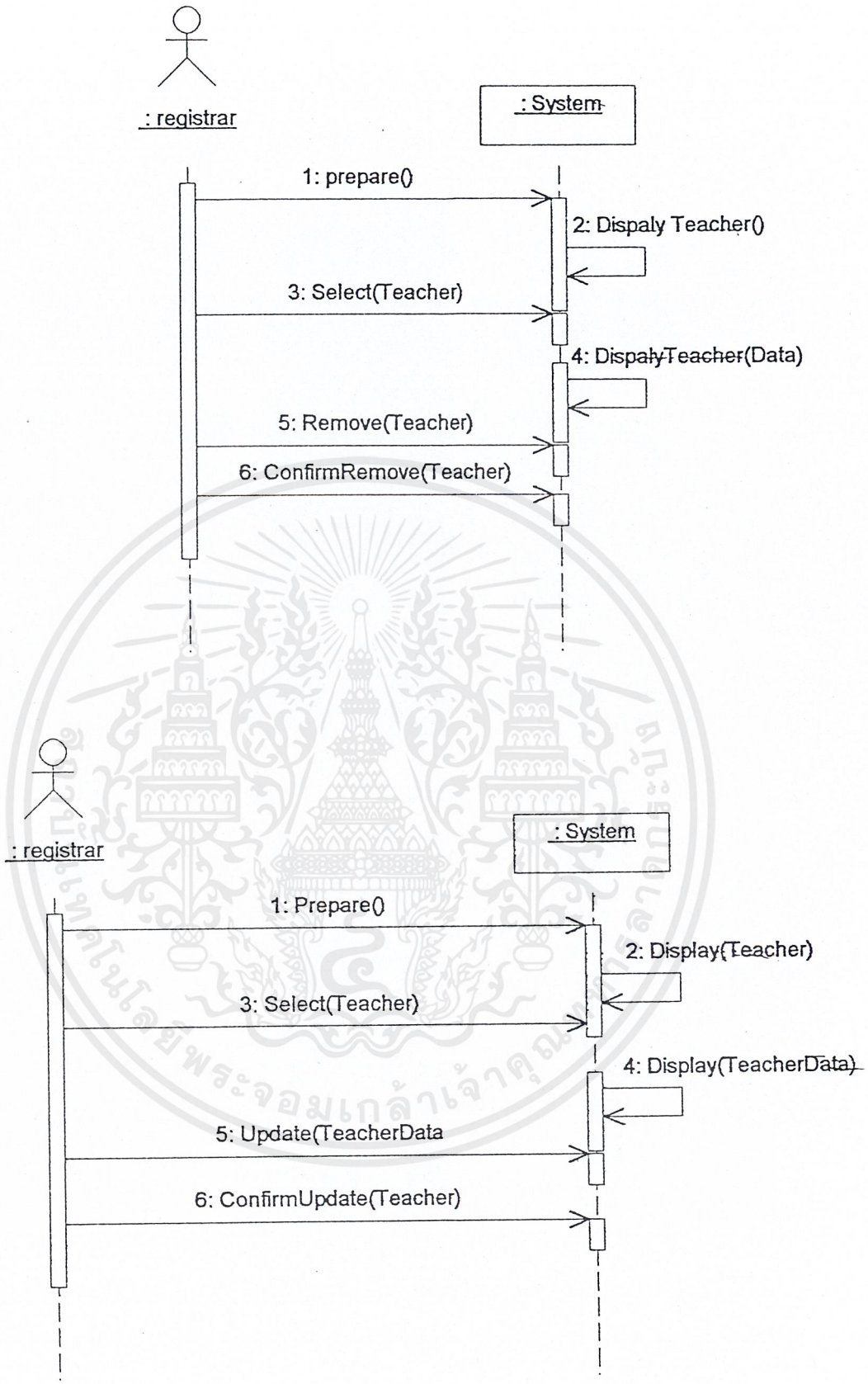




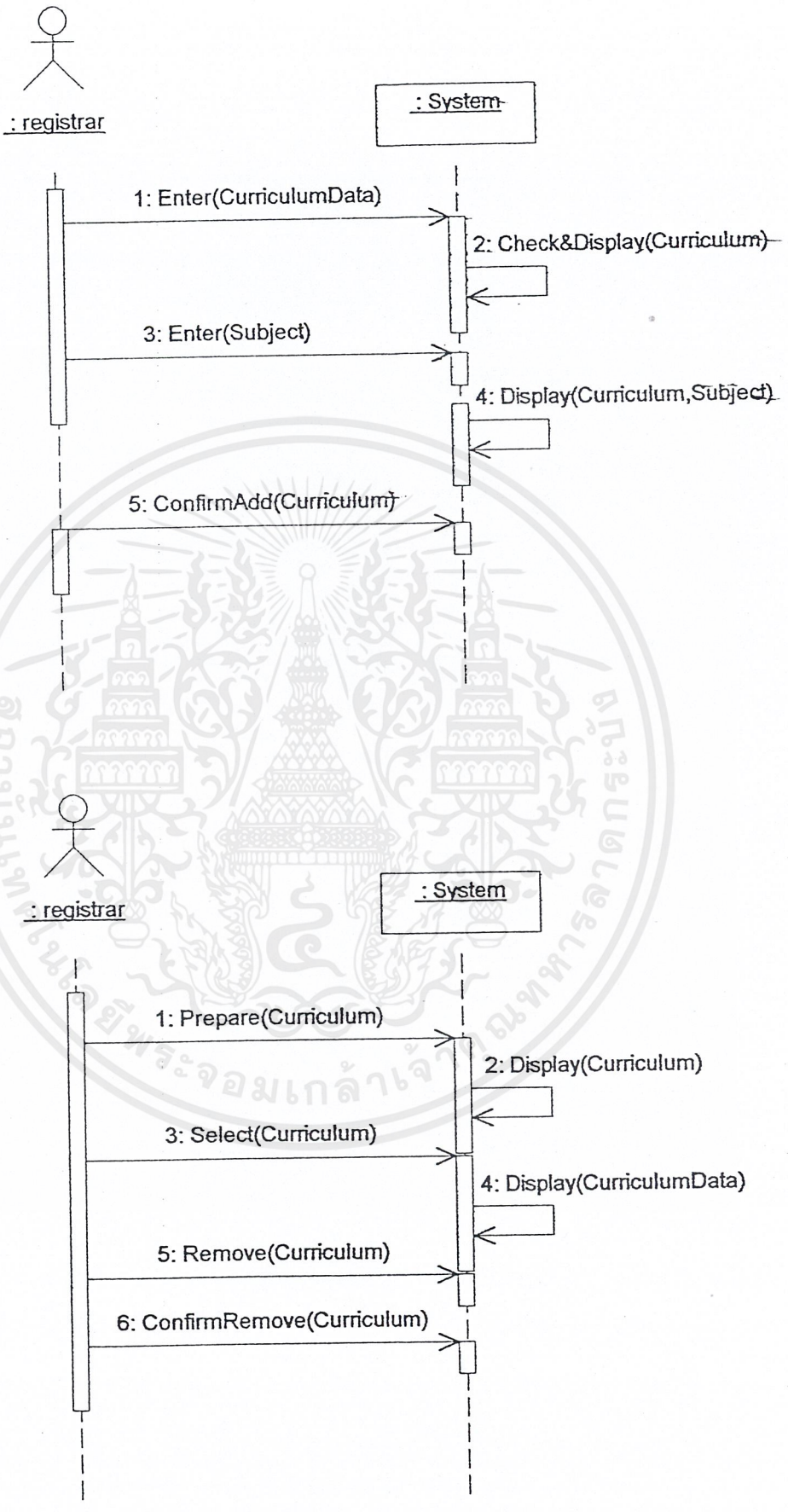


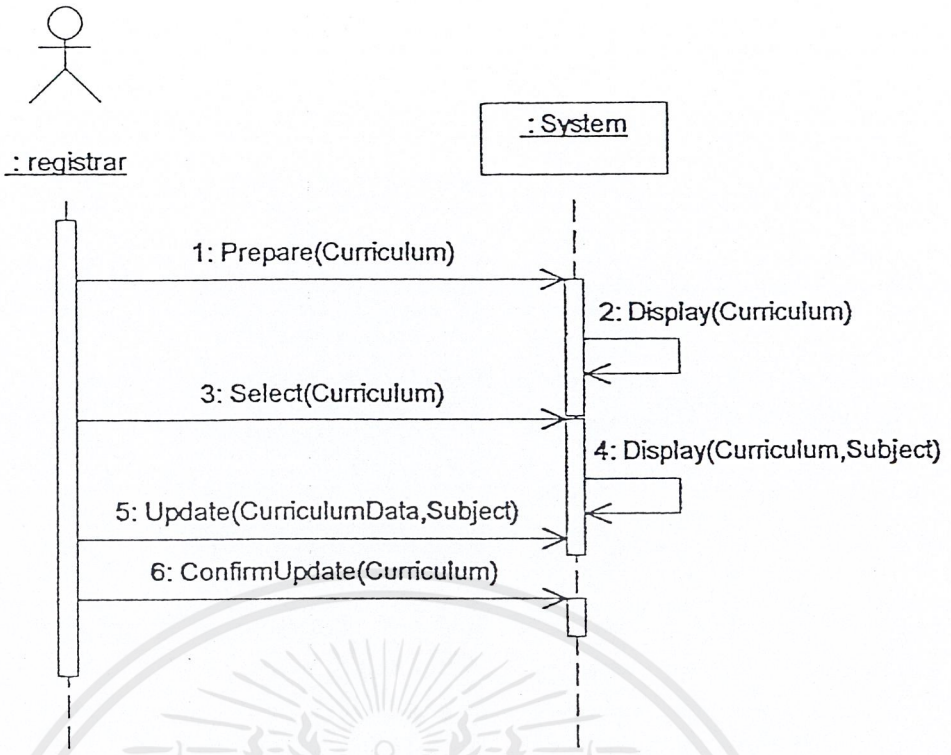
รูปที่ 4.5 Sequence Diagram ของ Use case Maintain Student Information



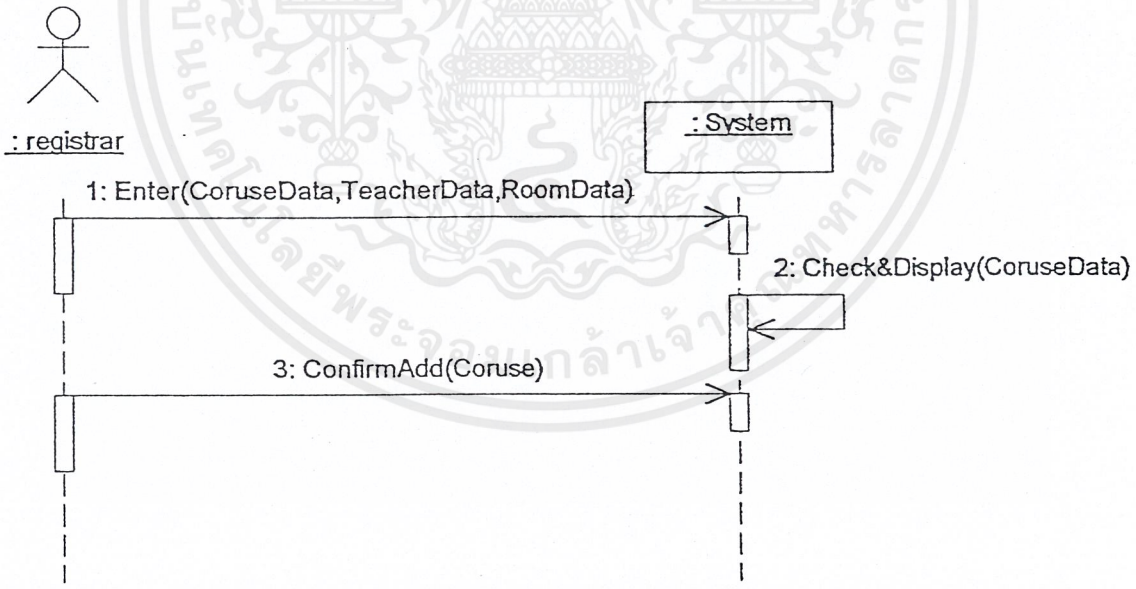


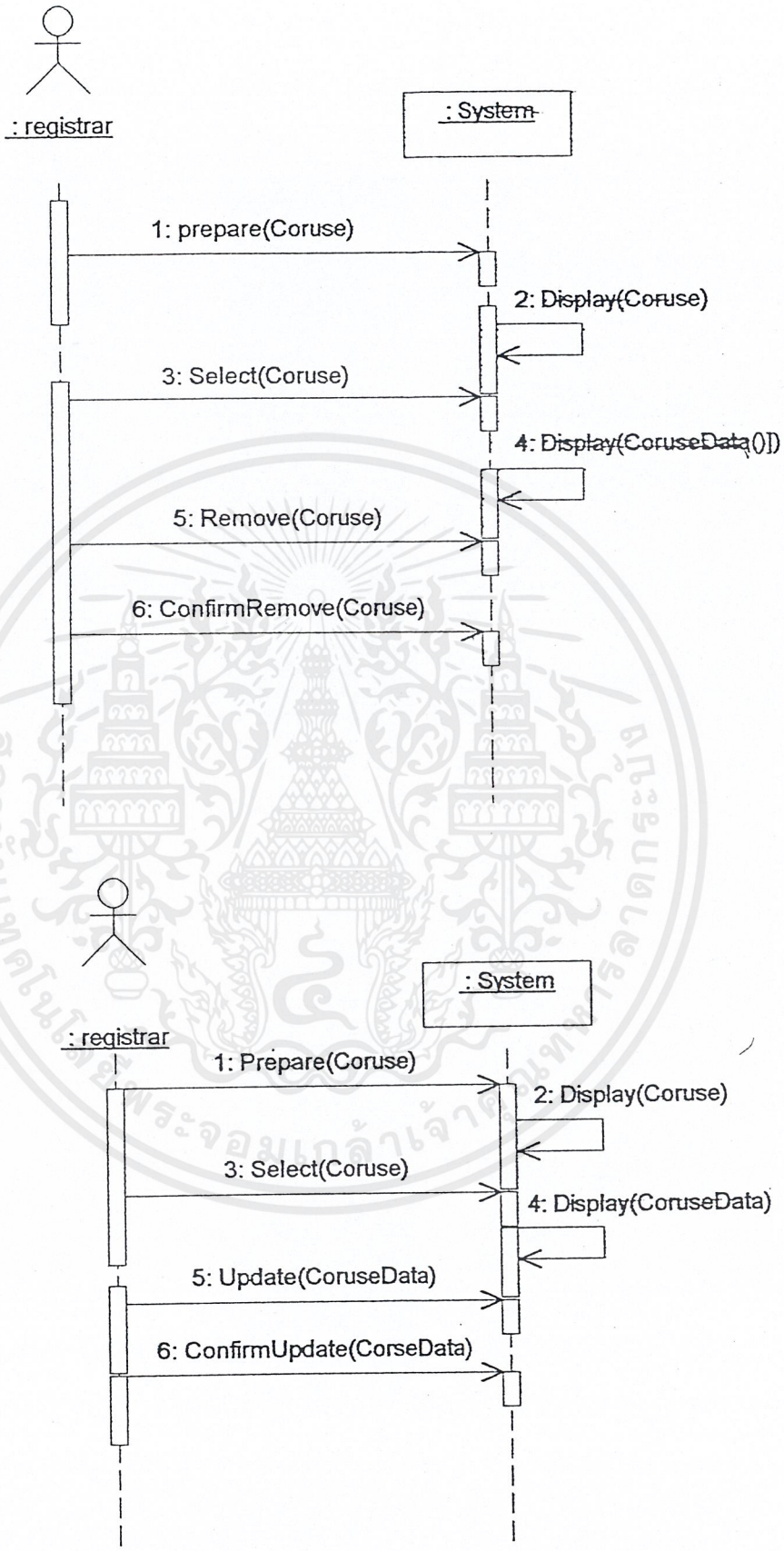
รูปที่ 4.6 Sequence Diagram ของ Use case Maintain Teacher Information



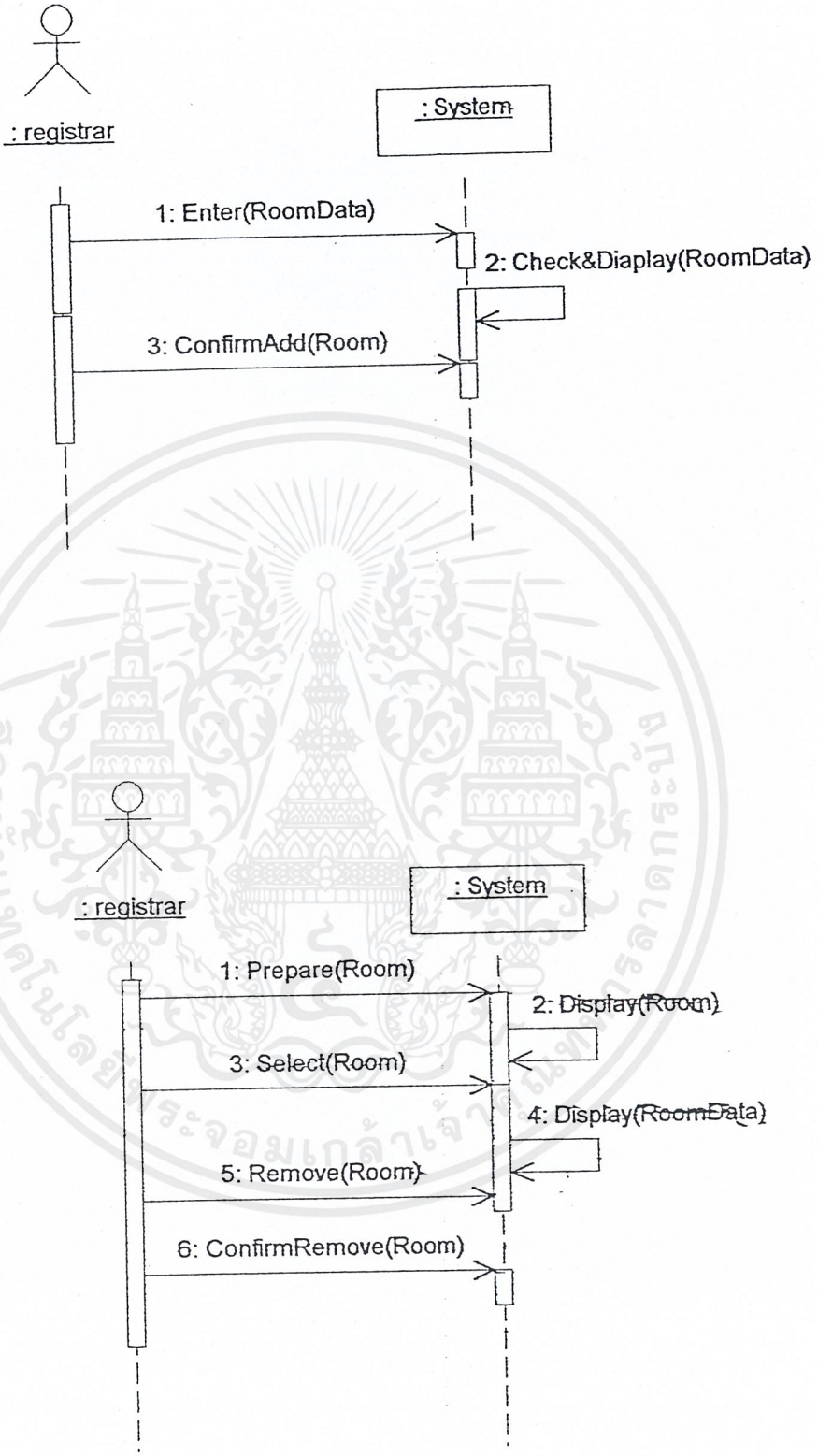


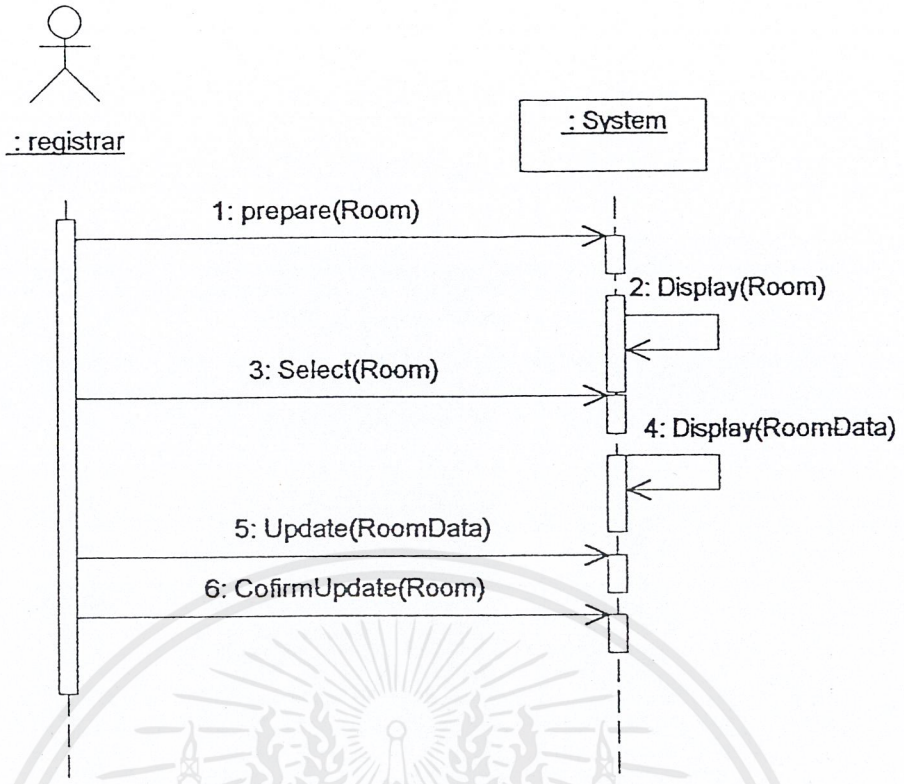
รูปที่ 4.7 Sequence Diagram ของ Use case Maintain Curriculum



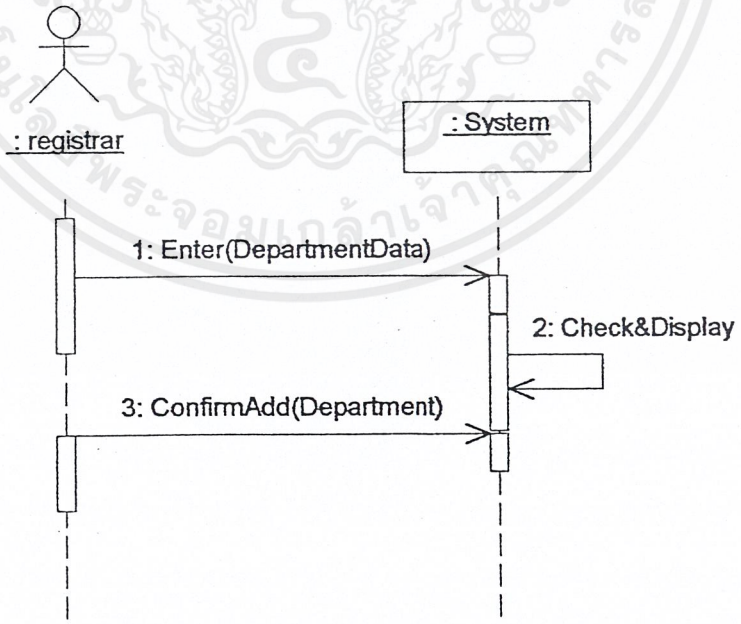


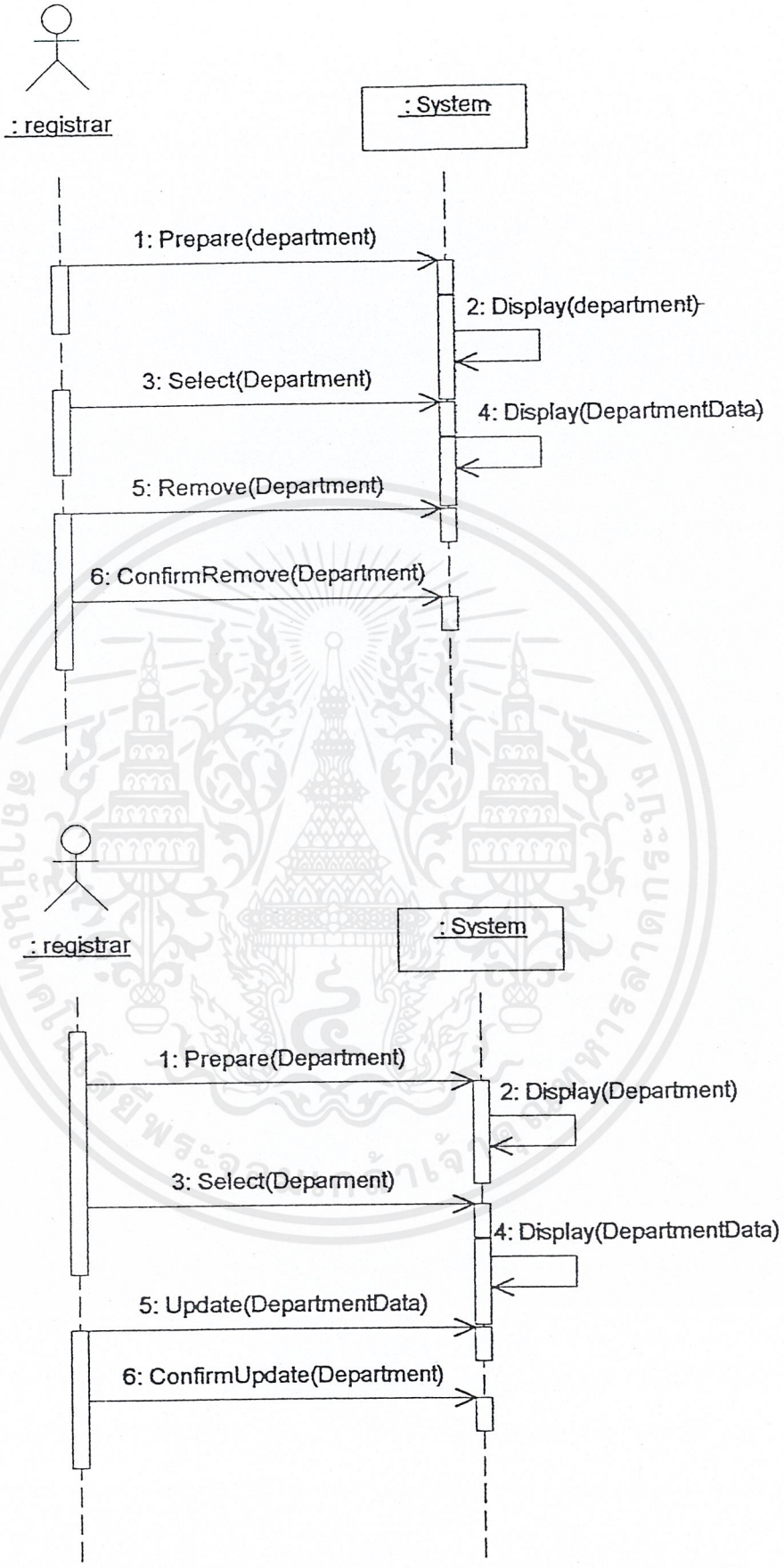
รูปที่ 4.8 Sequence Diagram ของ Use case Maintain Course Information



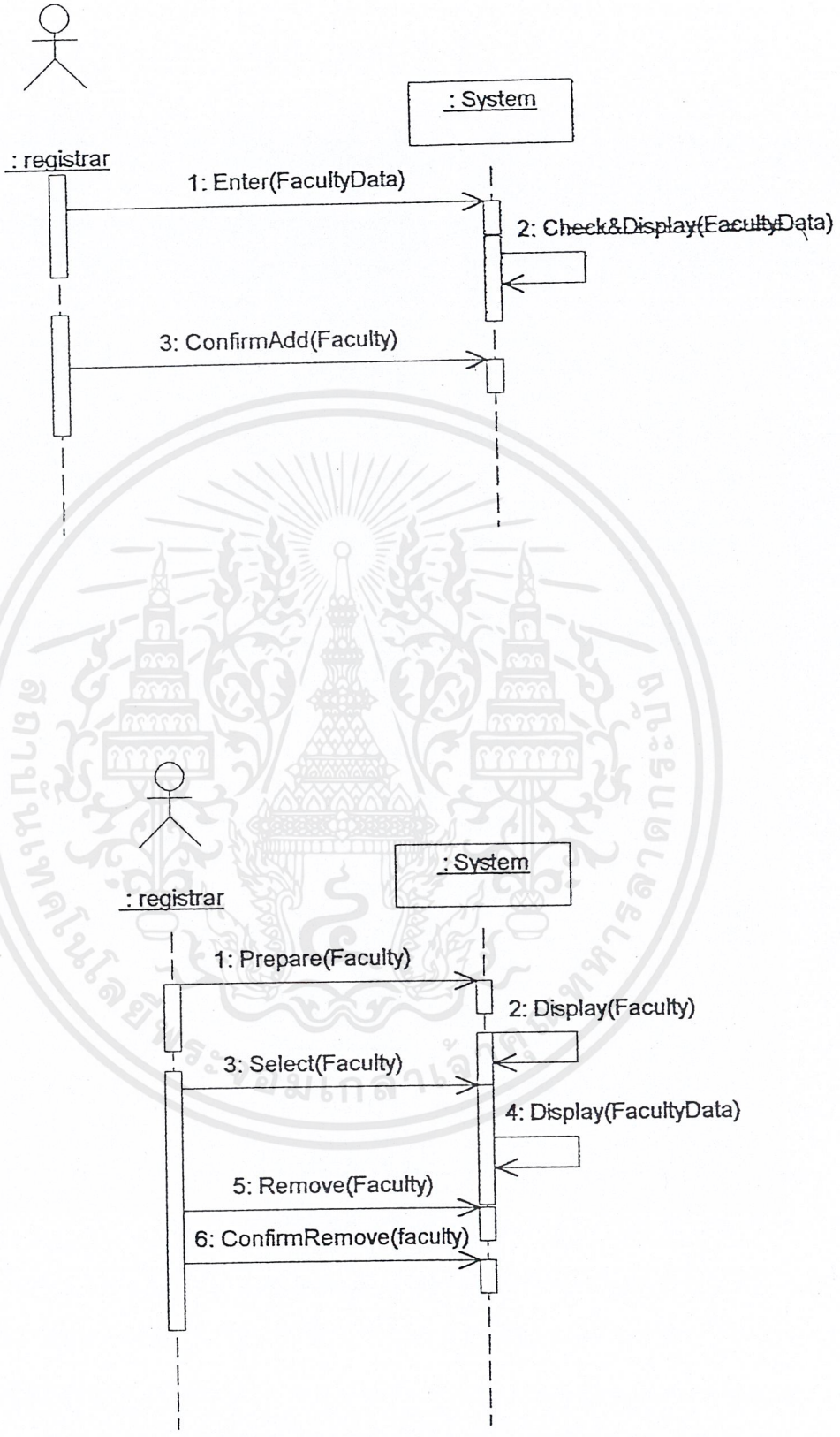


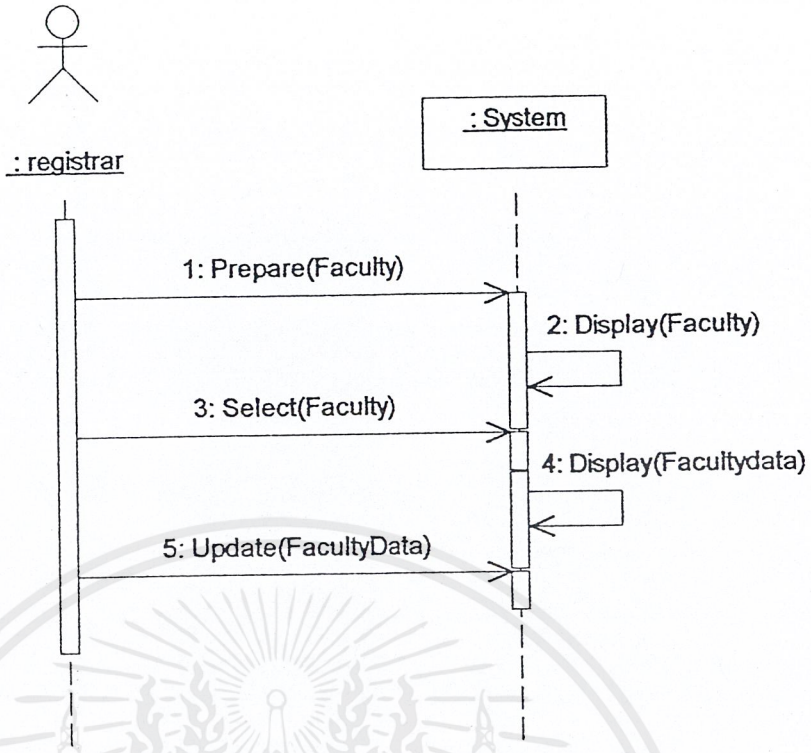
รูปที่ 4.9 Sequence Diagram ของ Use case Maintain Room Information



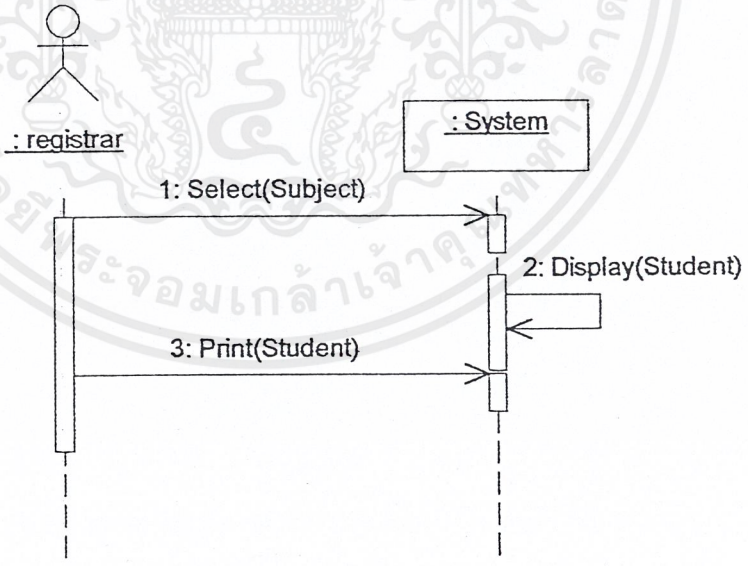


รูปที่ 4.10 Sequence Diagram ~~ของ~~ Use case Maintain Department Information

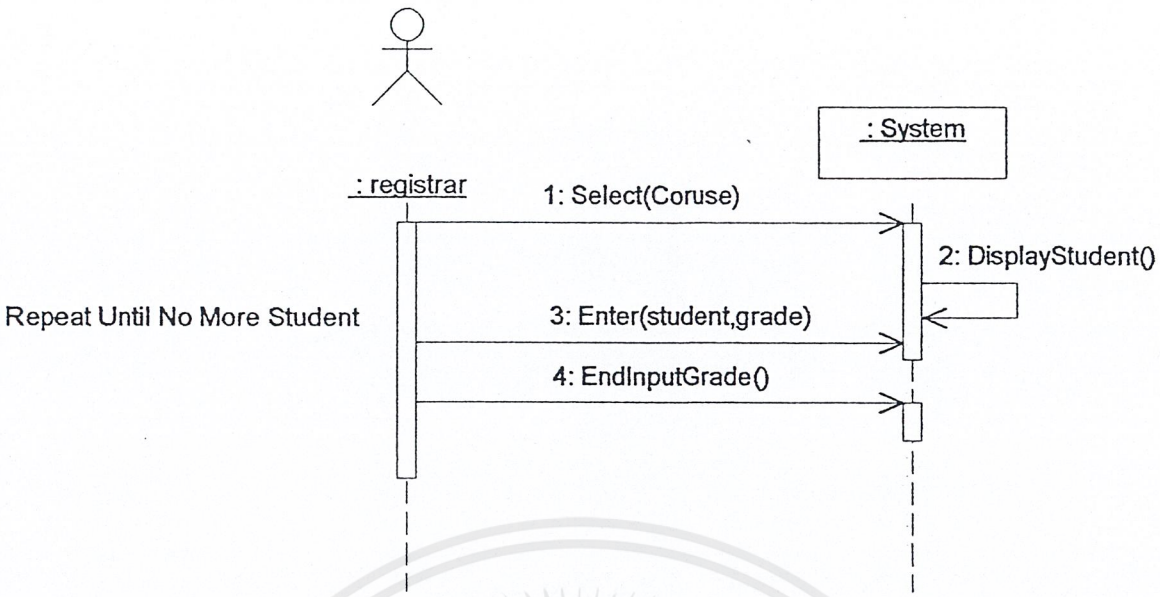




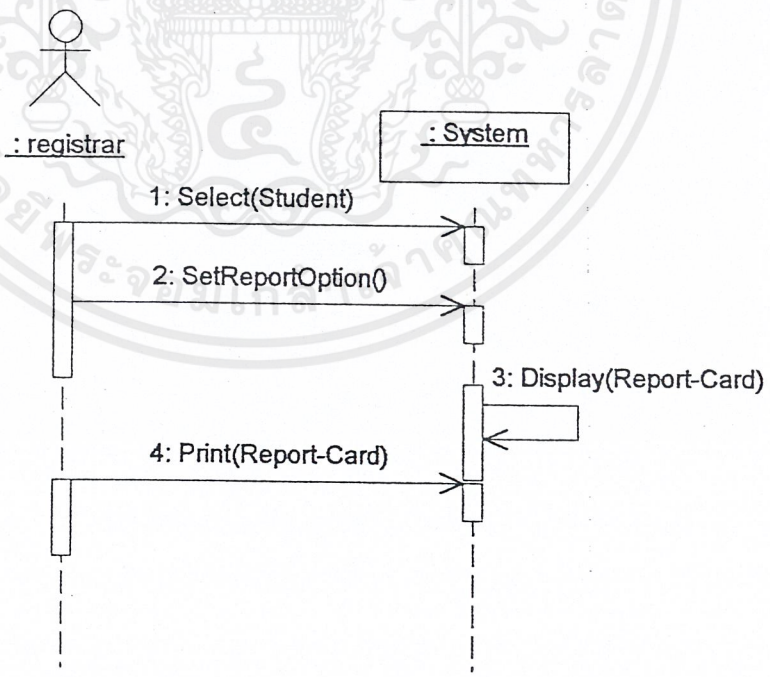
รูปที่ 4.11 Sequence Diagram ของ Use case Maintain Faculty Information

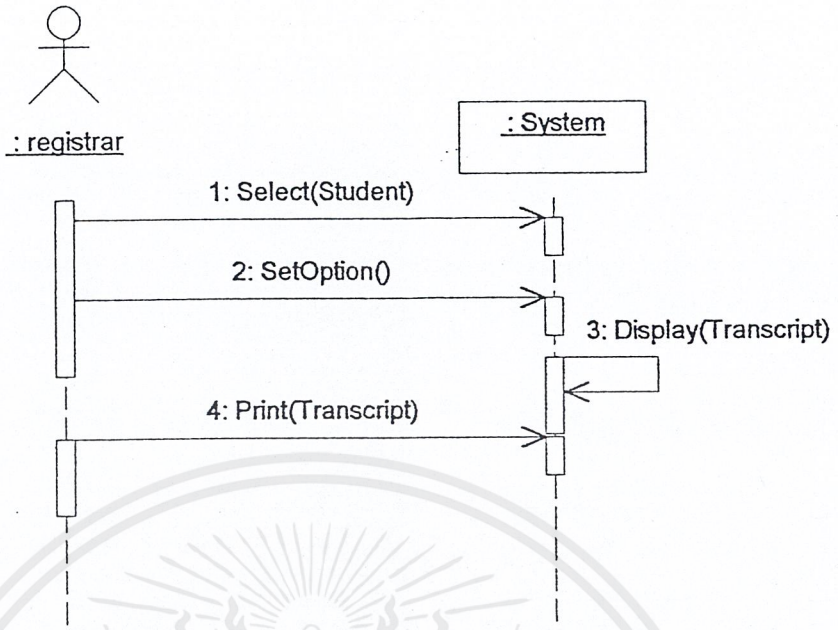


รูปที่ 4.12 Sequence Diagram ของ Use case List Student Name



รูปที่ 4.13 Sequence Diagram ของ Use case Input Grade





รูปที่ 4.14 Sequence Diagram ของ Use case Issue Document

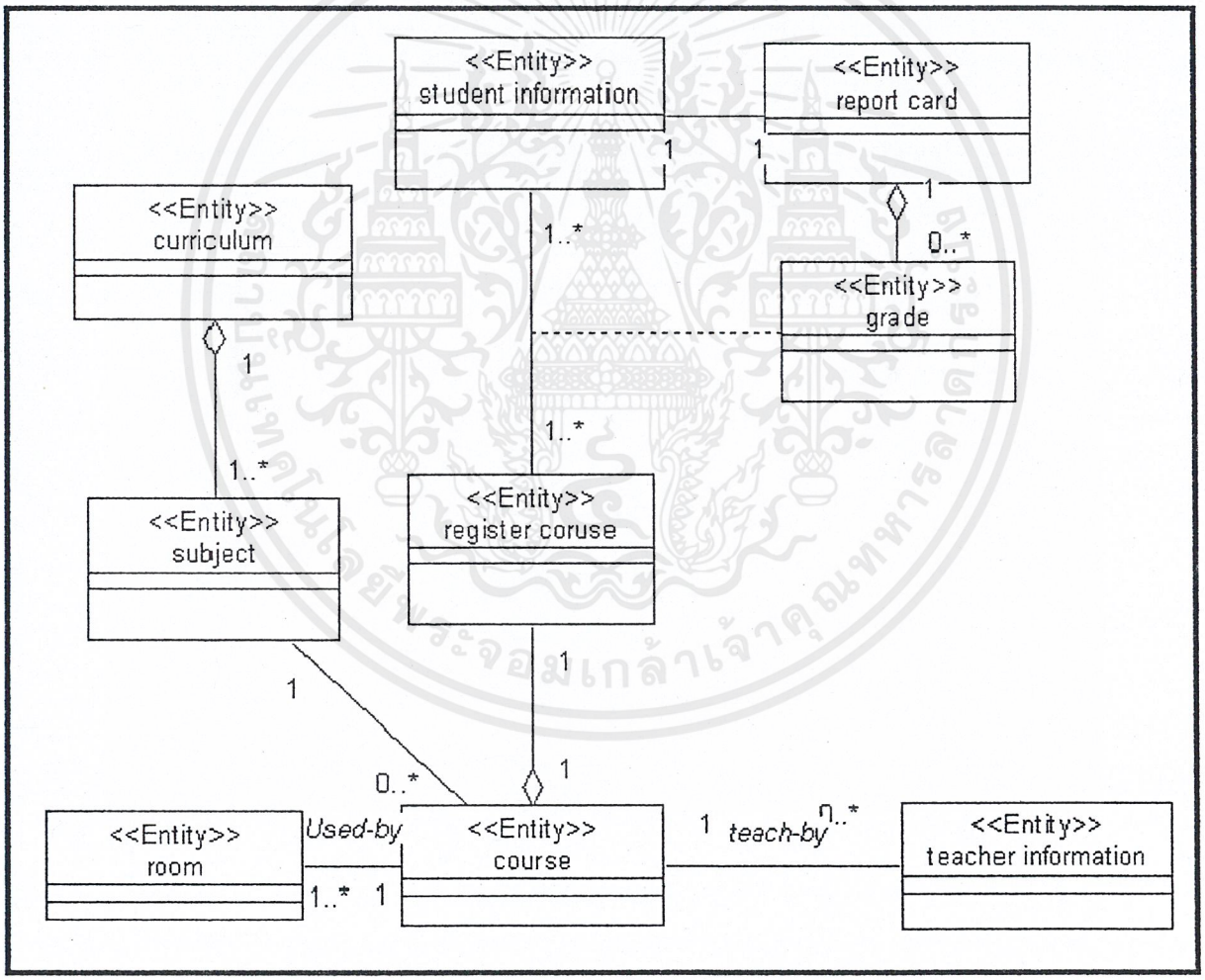
4.2 ออกแบบระบบ

ในขั้นตอนการออกแบบ สิ่งที่สำคัญที่ต้องสร้าง คือ interaction diagram ซึ่งจะเป็น diagram ที่แสดงว่าออบเจ็กต์ต่างๆมีการติดต่อหรือสัมพันธ์กันอย่างไร

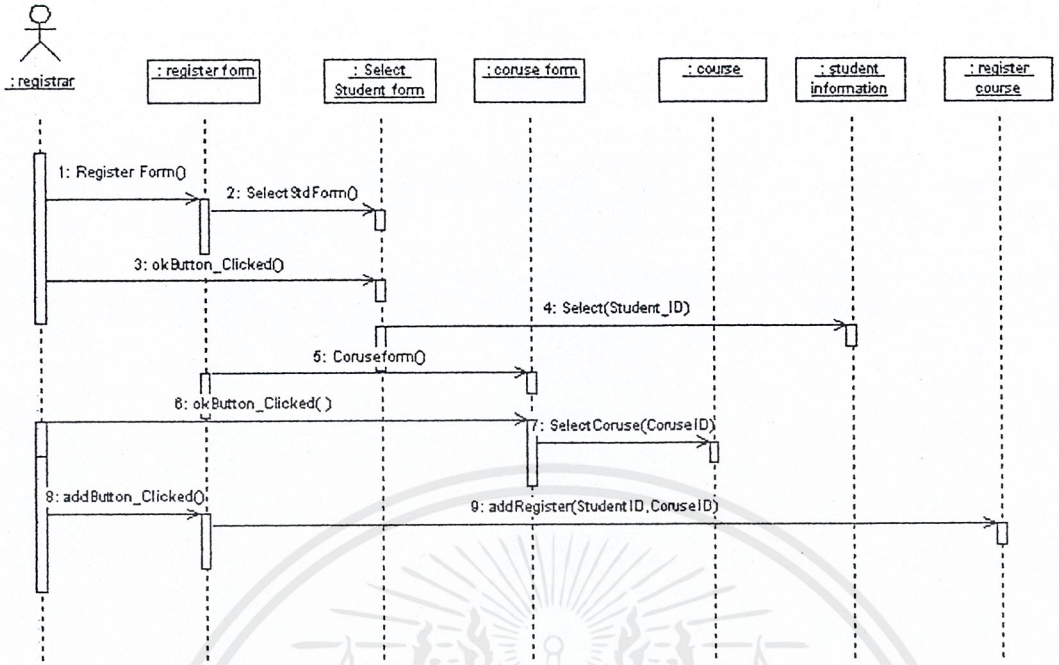
ใน UML กำหนดชนิดของ interaction diagram ไว้ 2 ชนิด ได้แก่ collaboration diagram และ sequence diagram ทั้ง 2 ชนิดแสดงความสัมพันธ์ของออบเจ็กต์ต่างๆเหมือนกันแต่ collaboration diagram จะแสดงความสัมพันธ์ในรูปของ network ส่วน sequence diagram แสดงในรูปของความสัมพันธ์ระหว่าง 2 ออบเจ็กต์ sequence diagram ในขั้นนี้จะต่างกับในขั้นที่แล้ว คือ จะแสดงการโต้ตอบกันระหว่างออบเจ็กต์ในระบบ

หลังจากออกแบบ interaction diagram แล้ว ก็จะสามารถวาด class diagrams ซึ่งสามารถกำหนดรายละเอียดต่างๆภายใน class และ message ต่างๆลงไป

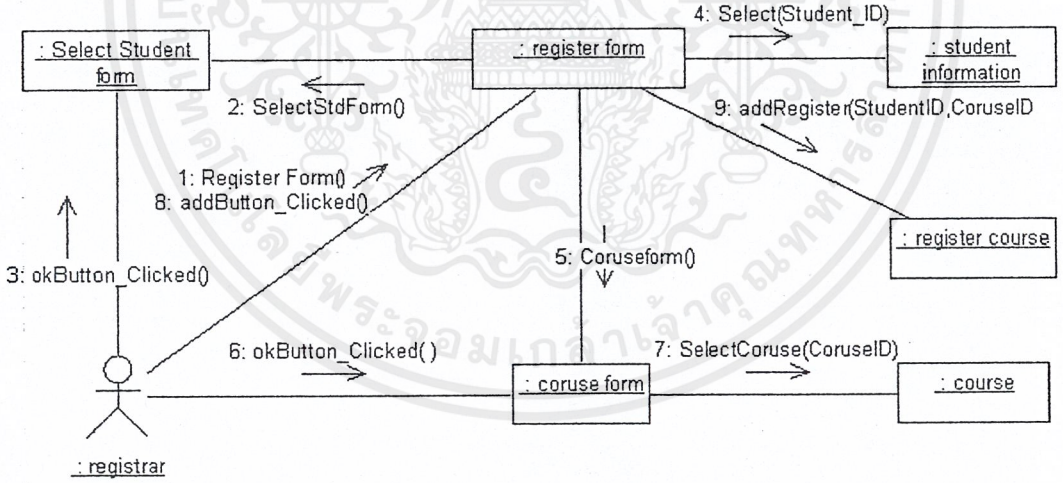
4.2.1 Class Diagram



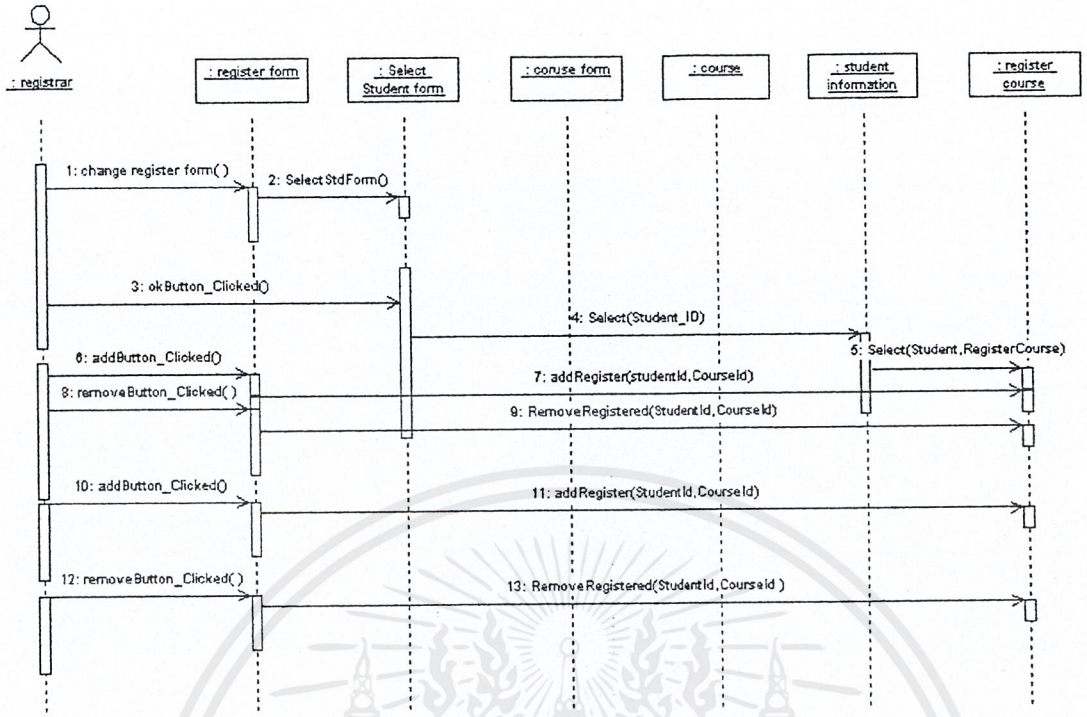
รูปที่ 4.15 แสดง class diagram (main) ของระบบทะเบียนที่ออกแบบไว้



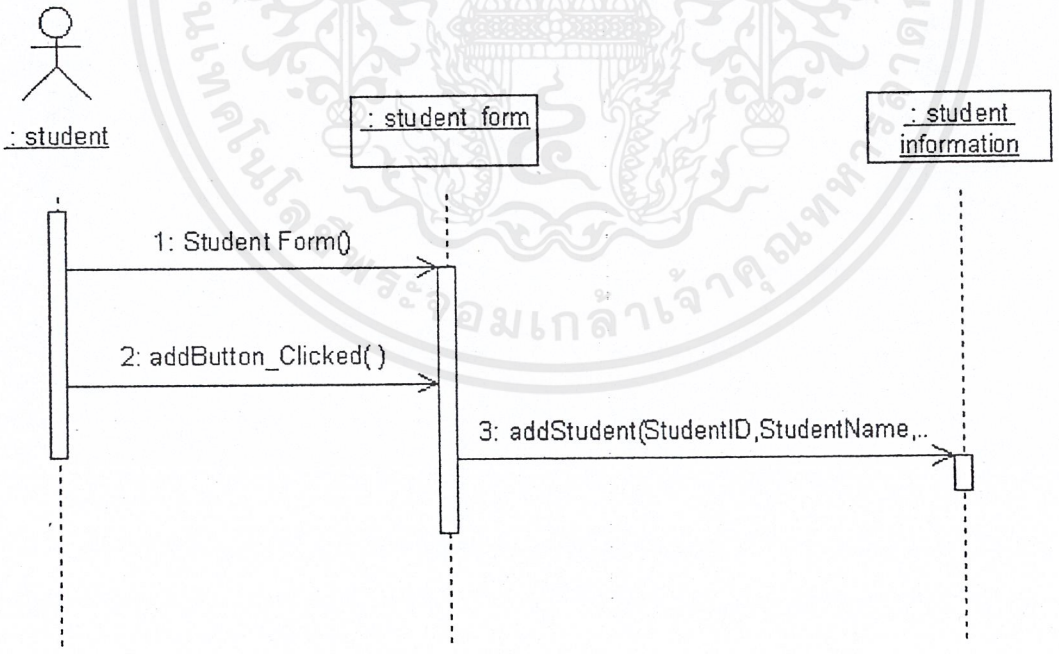
รูปที่ 4.16 แสดง Sequence diagram ของ register for course function



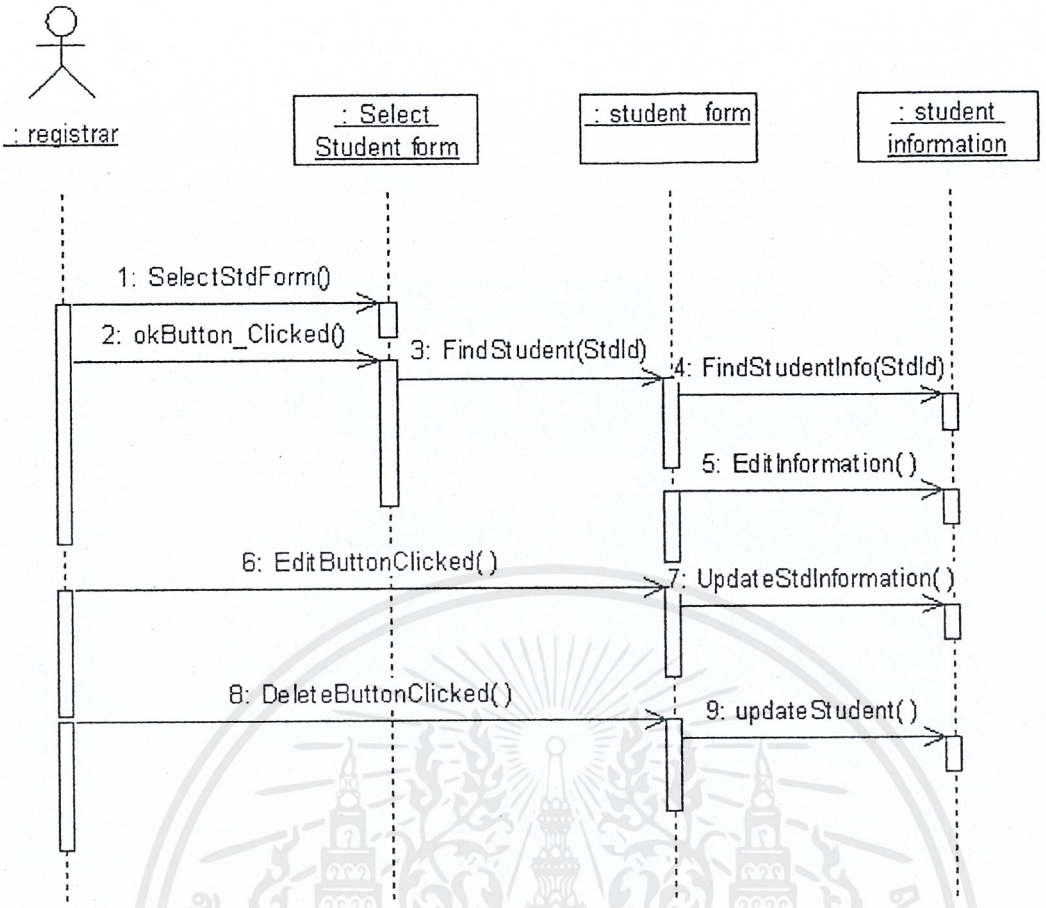
รูปที่ 4.17 แสดง Collaboration diagram ของ register for course function



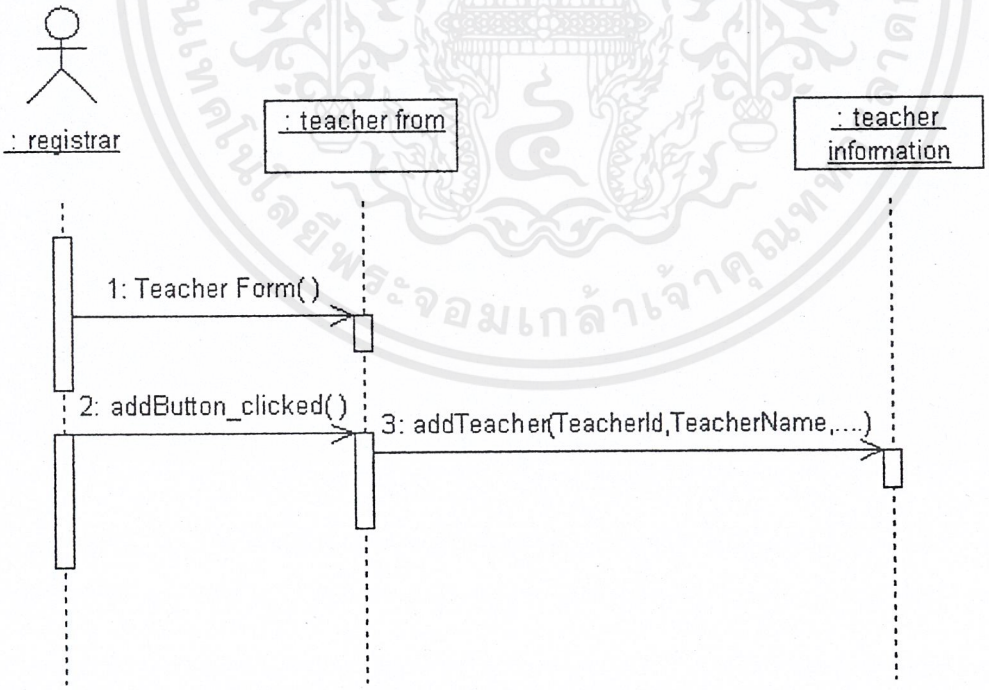
รูปที่ 4.18 แสดง Sequence diagram ของ change register function



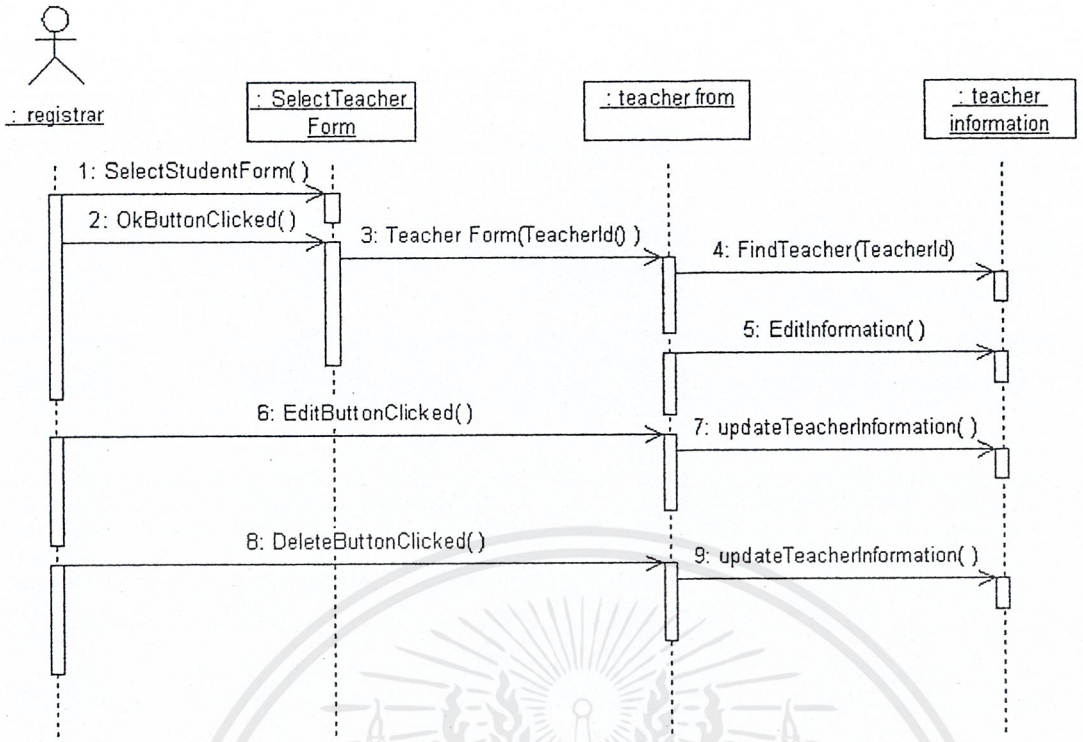
รูปที่ 4.19 แสดง sequence diagram ของ add student function



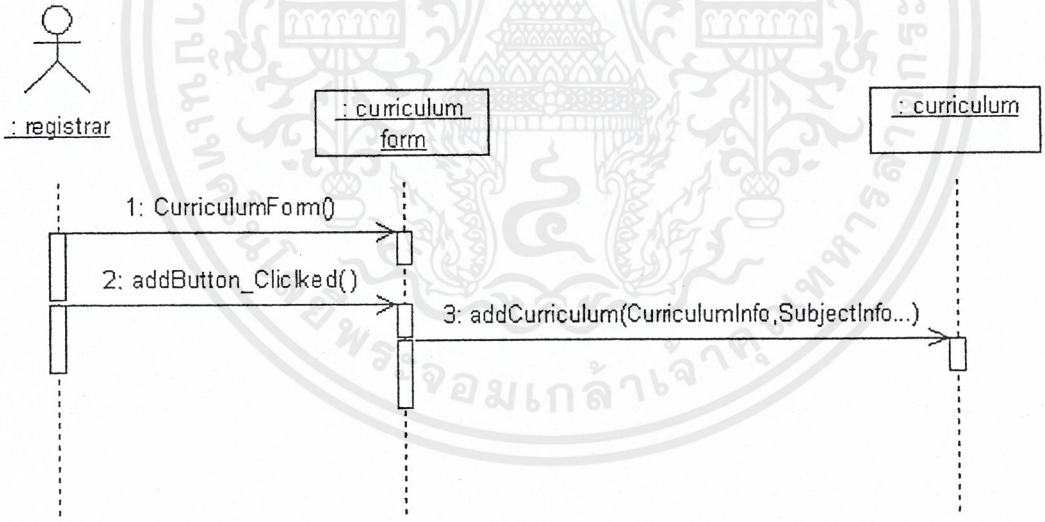
รูปที่ 4.20 แสดง Sequence diagram ของ update student function



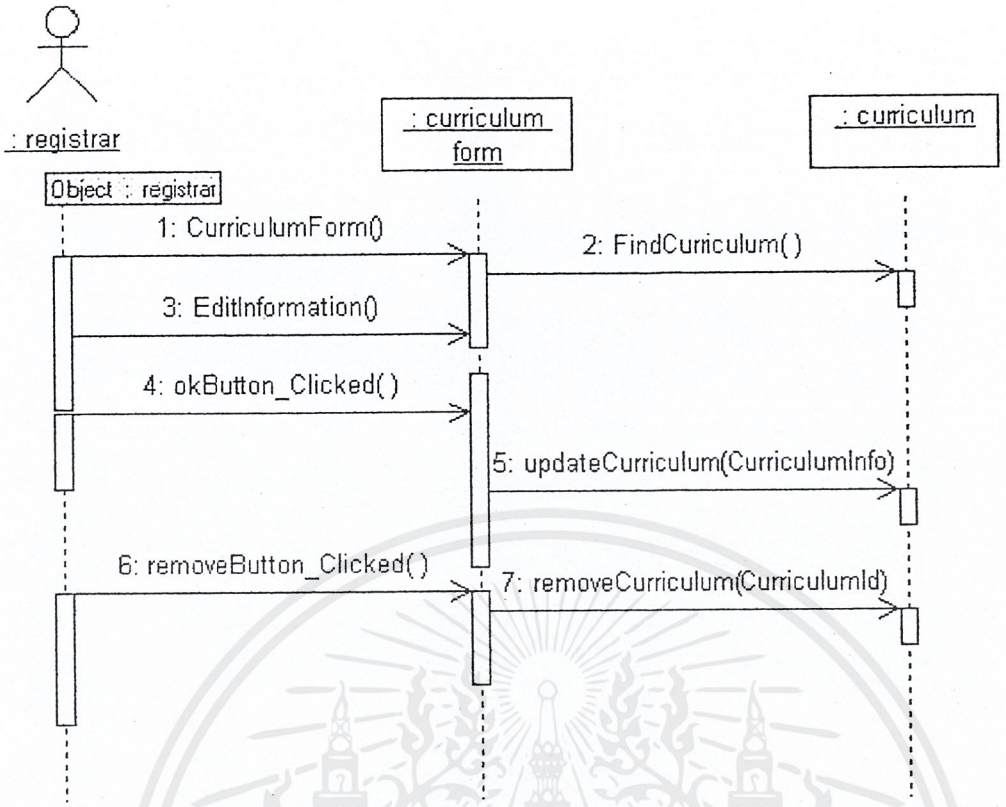
รูปที่ 4.21 แสดง sequence diagram ของ add teacher function



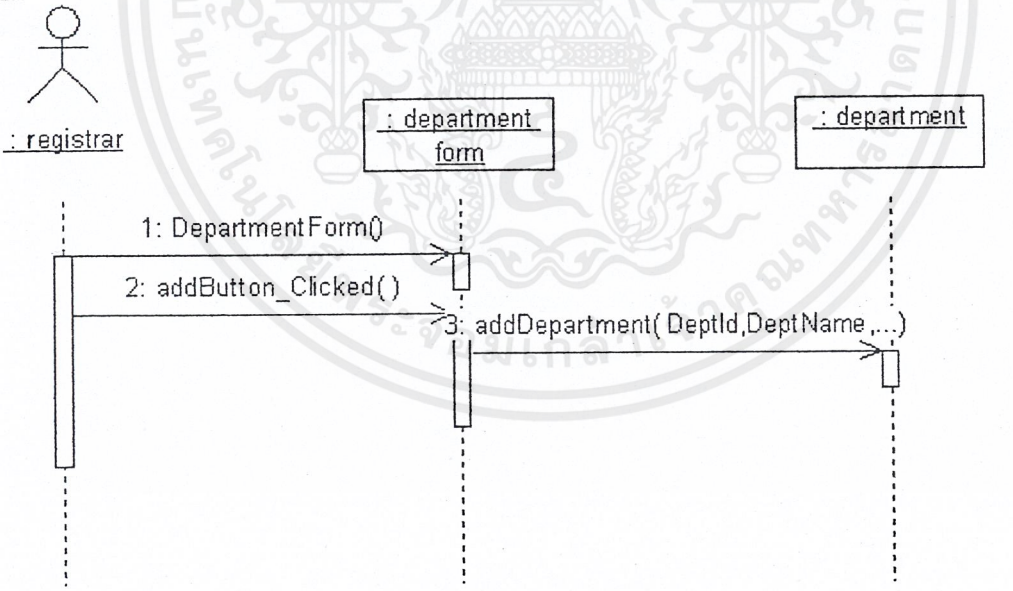
รูปที่ 4.22 แสดง Sequence diagram ของ update teacher function



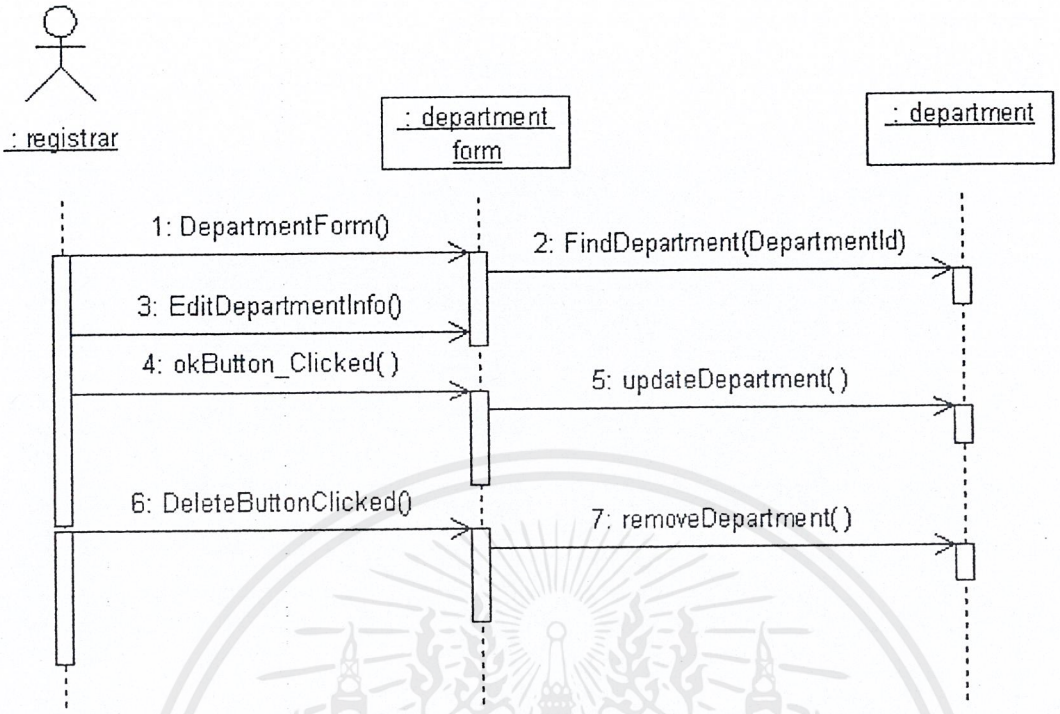
รูปที่ 4.23 แสดง Sequence diagram ของ add cirriculum function



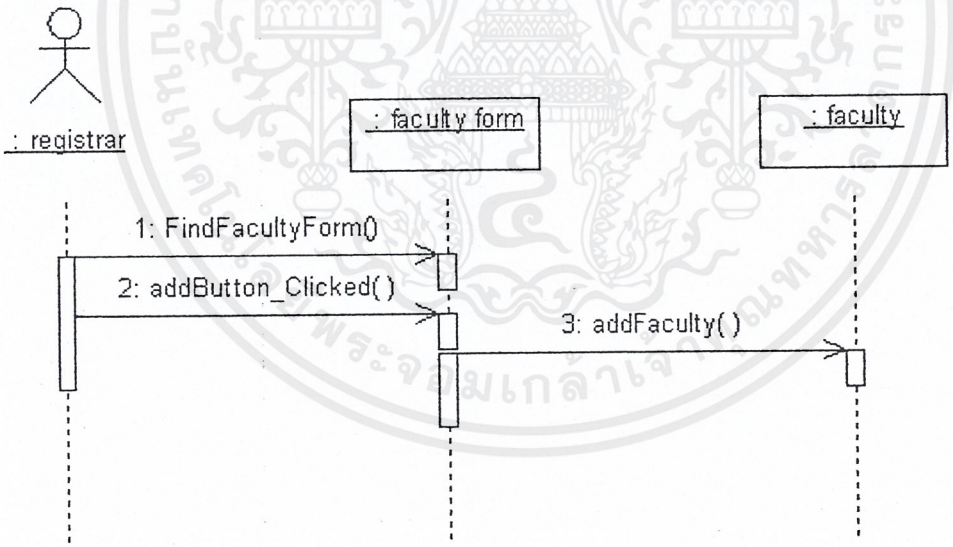
รูปที่ 4.24 แสดง Sequence diagram ของ update cirriculum function



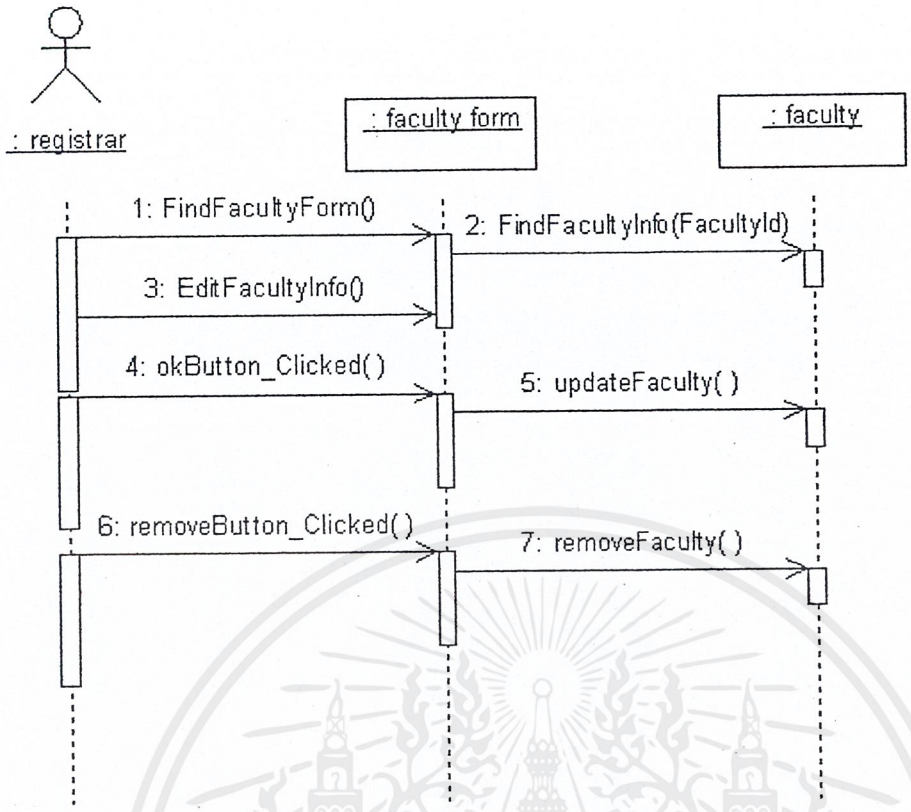
รูปที่ 4.25 แสดง Sequence diagram ของ add department function



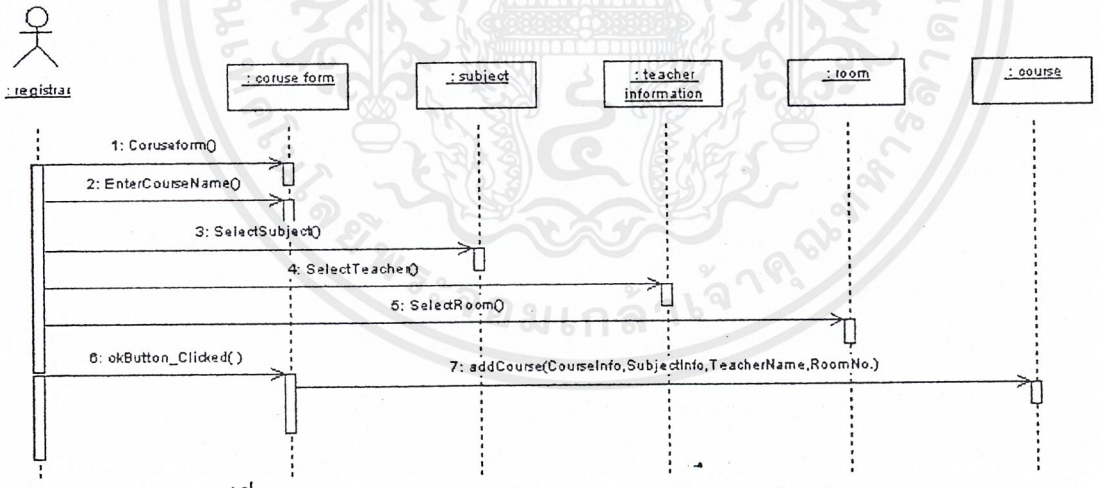
รูปที่ 4.26 แสดง Sequence diagram ของ update department function



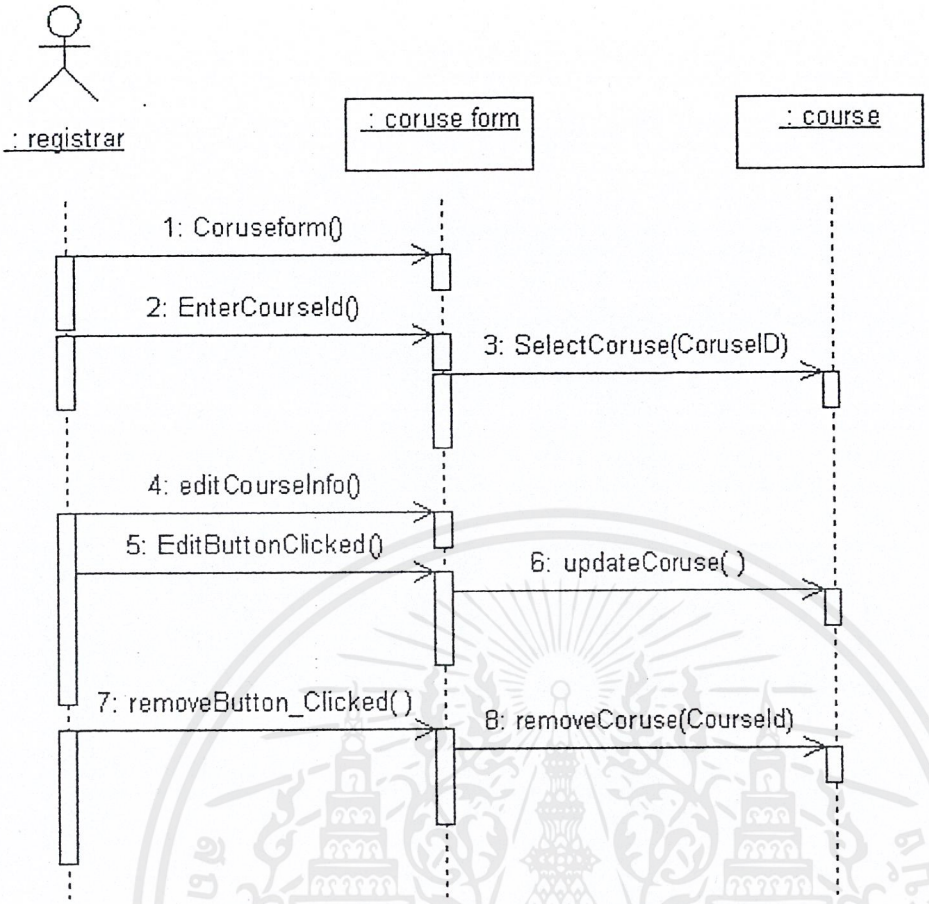
รูปที่ 4.27 แสดง Sequence diagram ของ add faculty function



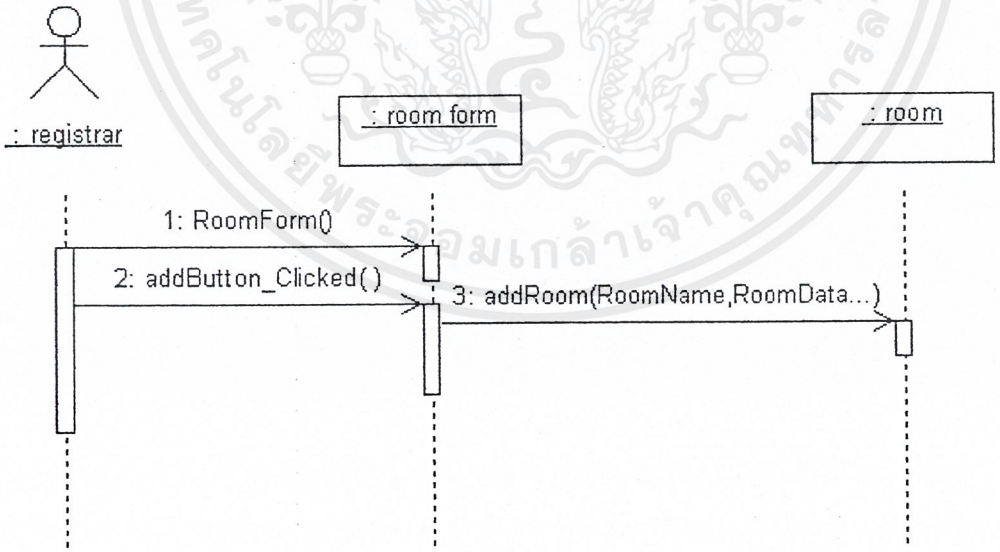
รูปที่ 4.28 แสดง Sequence diagram ของ update faculty function



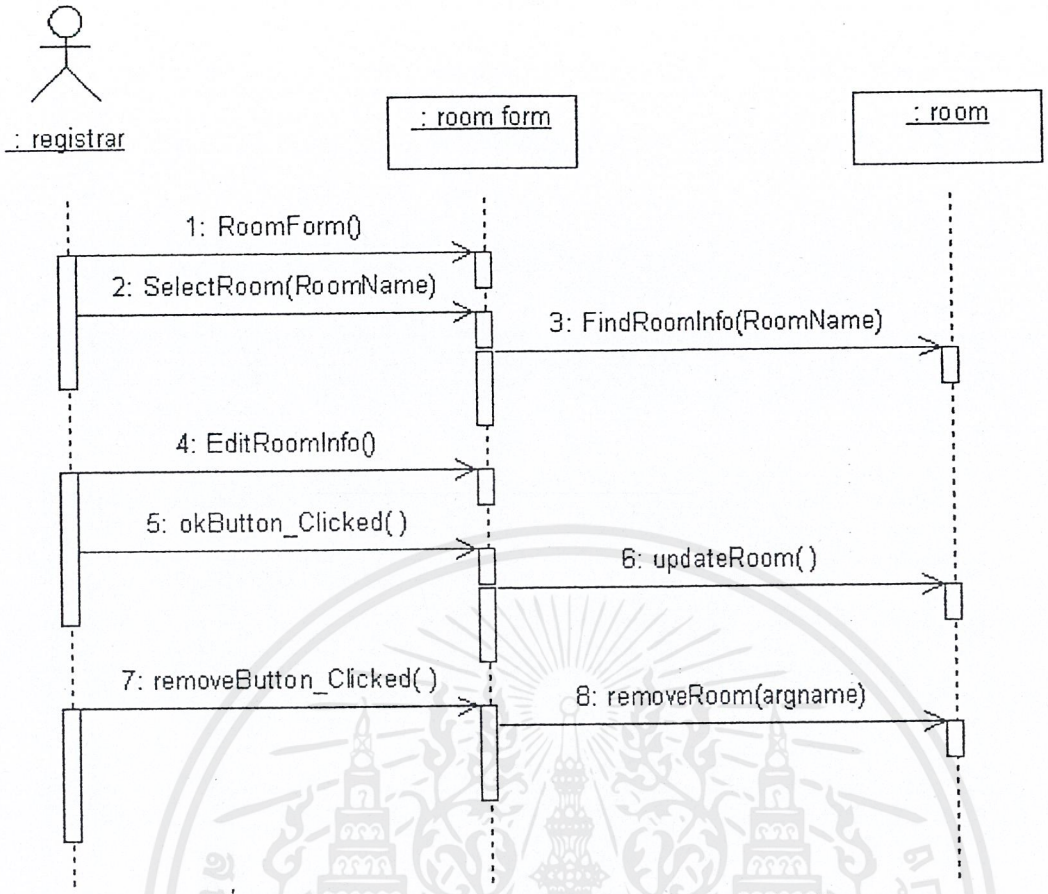
รูปที่ 4.29 แสดง Sequence diagram ของ add course function



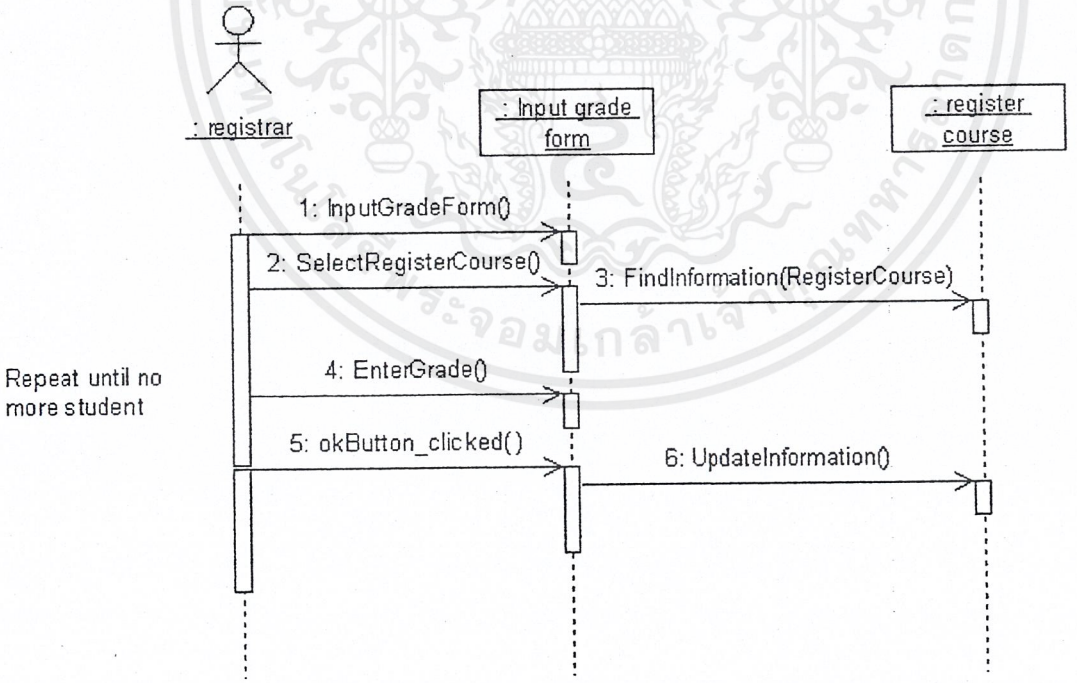
รูปที่ 4.30 แสดง Sequence diagram ของ update course function



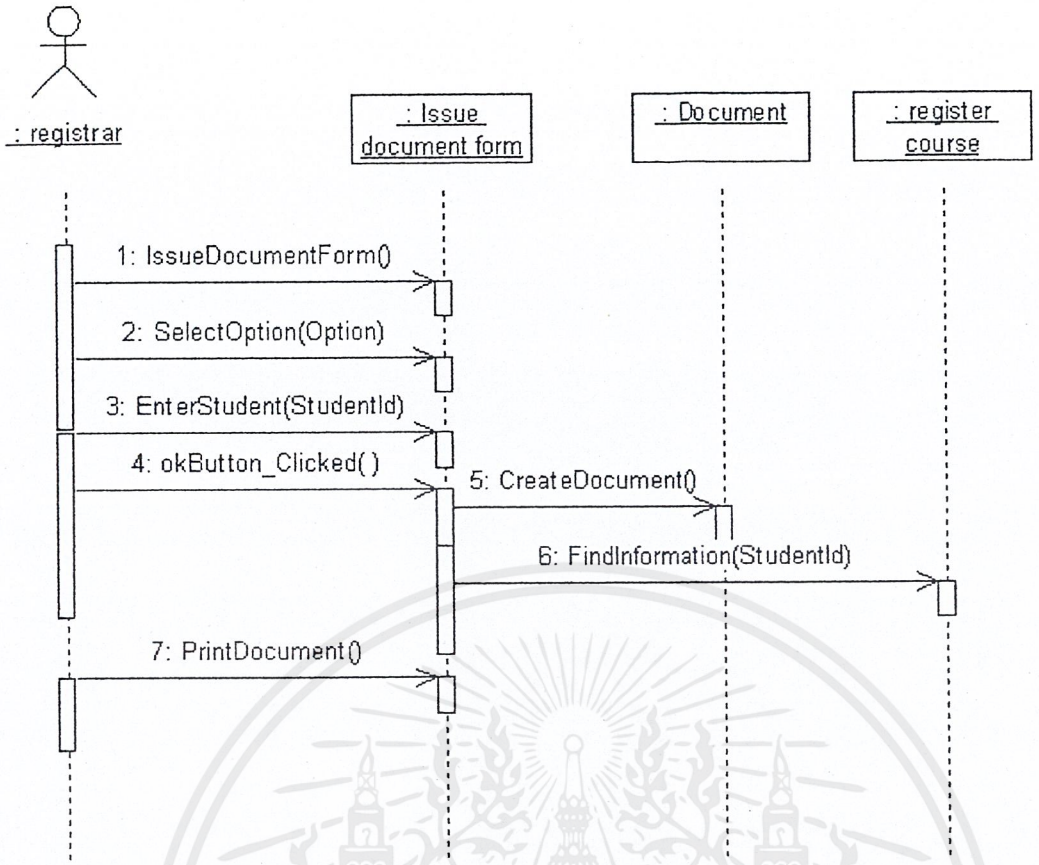
รูปที่ 4.31 แสดง Sequence diagram ของ add room function



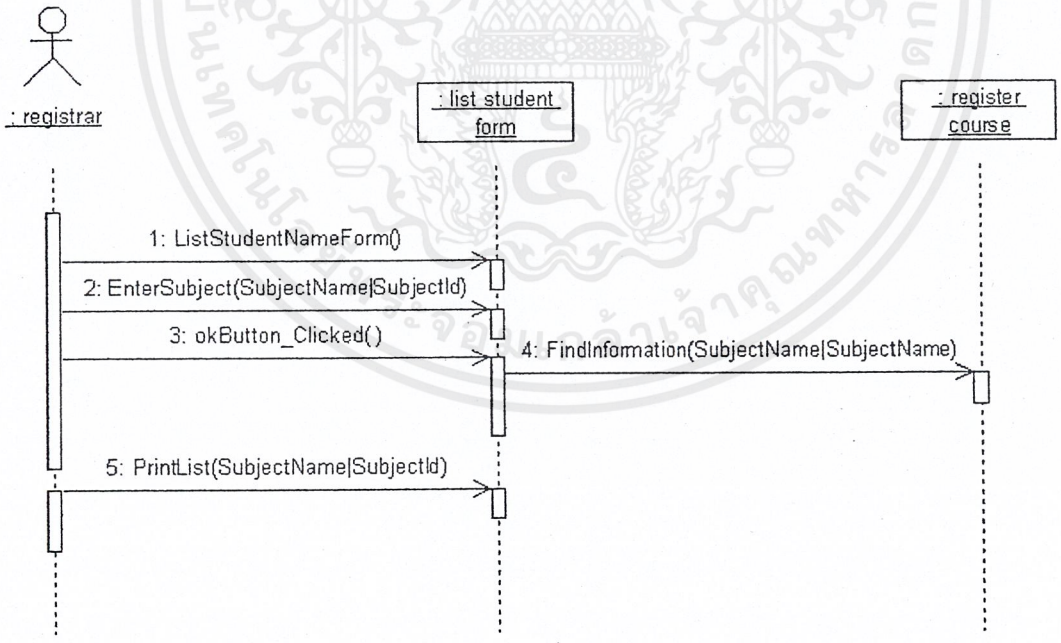
รูปที่ 4.32 แสดง Sequence diagram ของ update room function



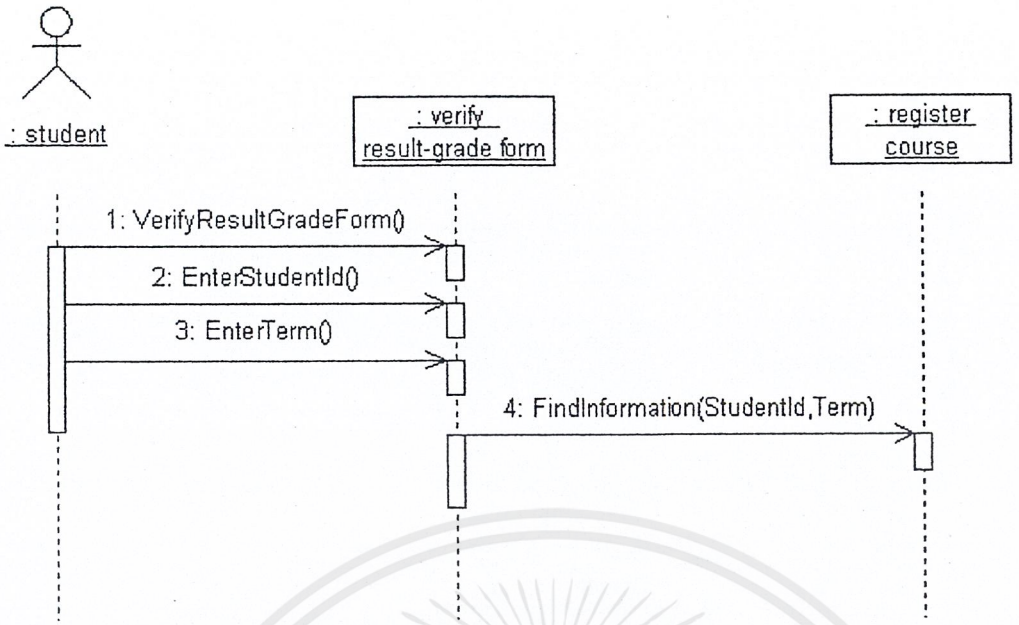
รูปที่ 4.33 แสดง Sequence diagram ของ input result grade function



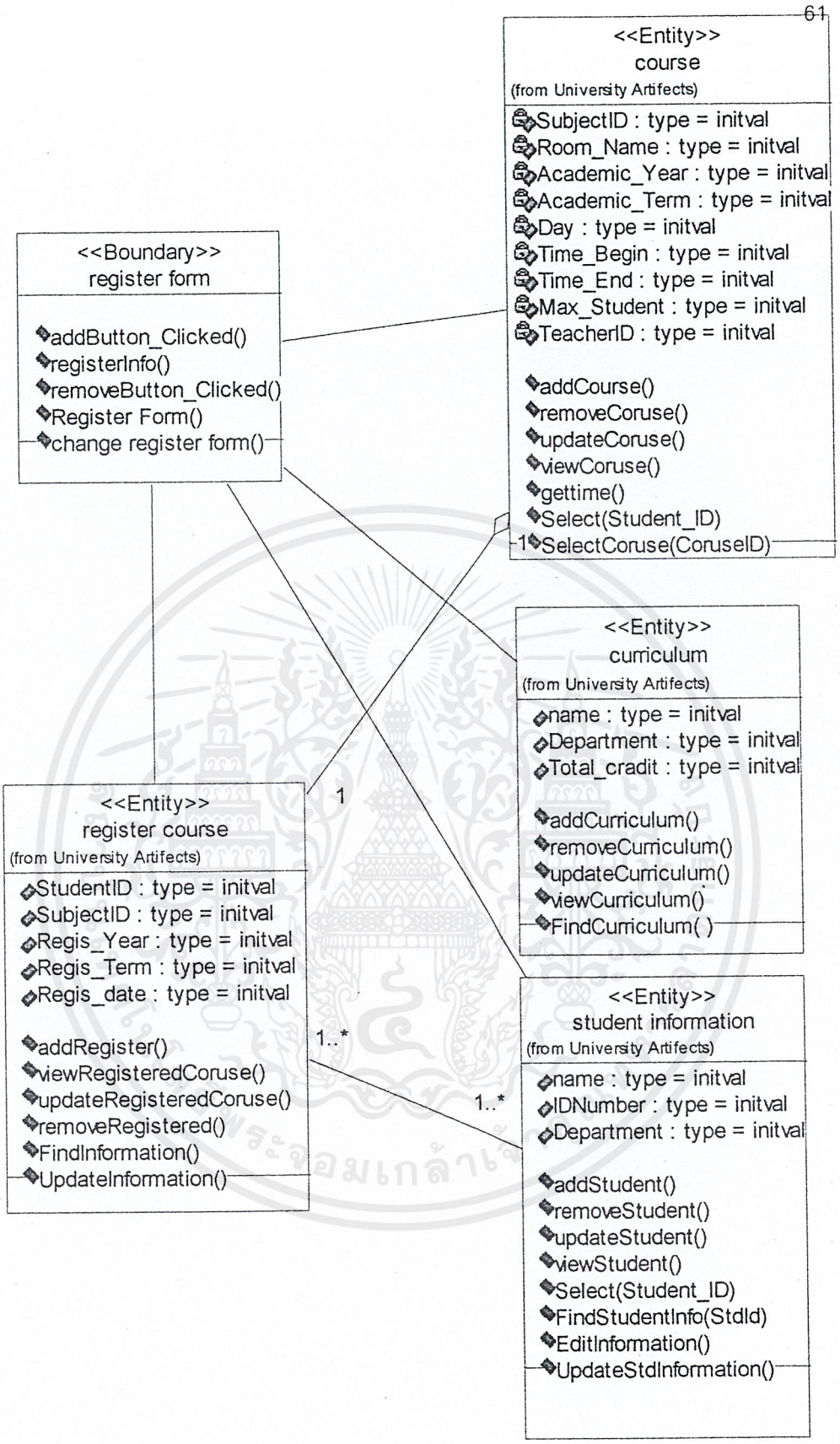
รูปที่ 4.34 แสดง Sequence diagram ของ issue document function



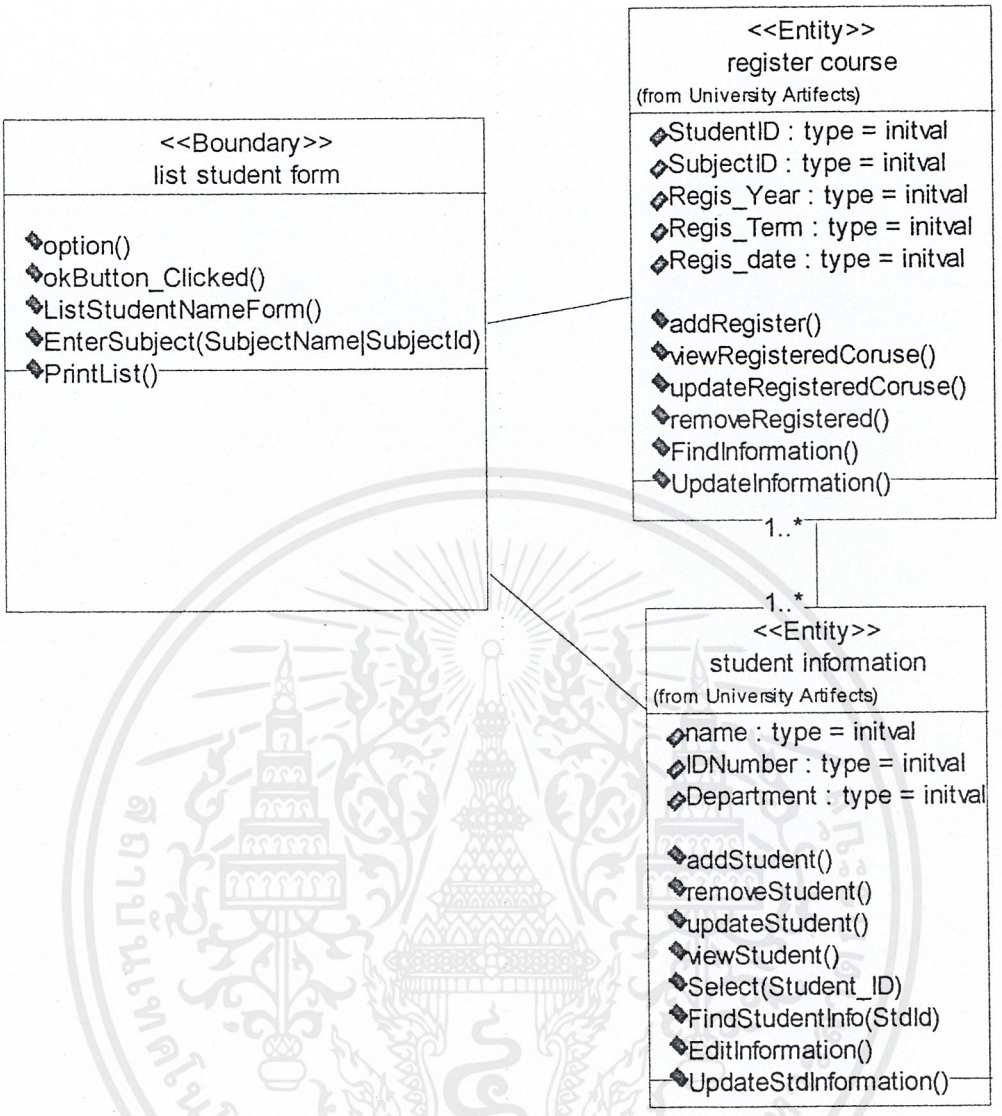
รูปที่ 4.35 แสดง Sequence diagram ของ list student name function



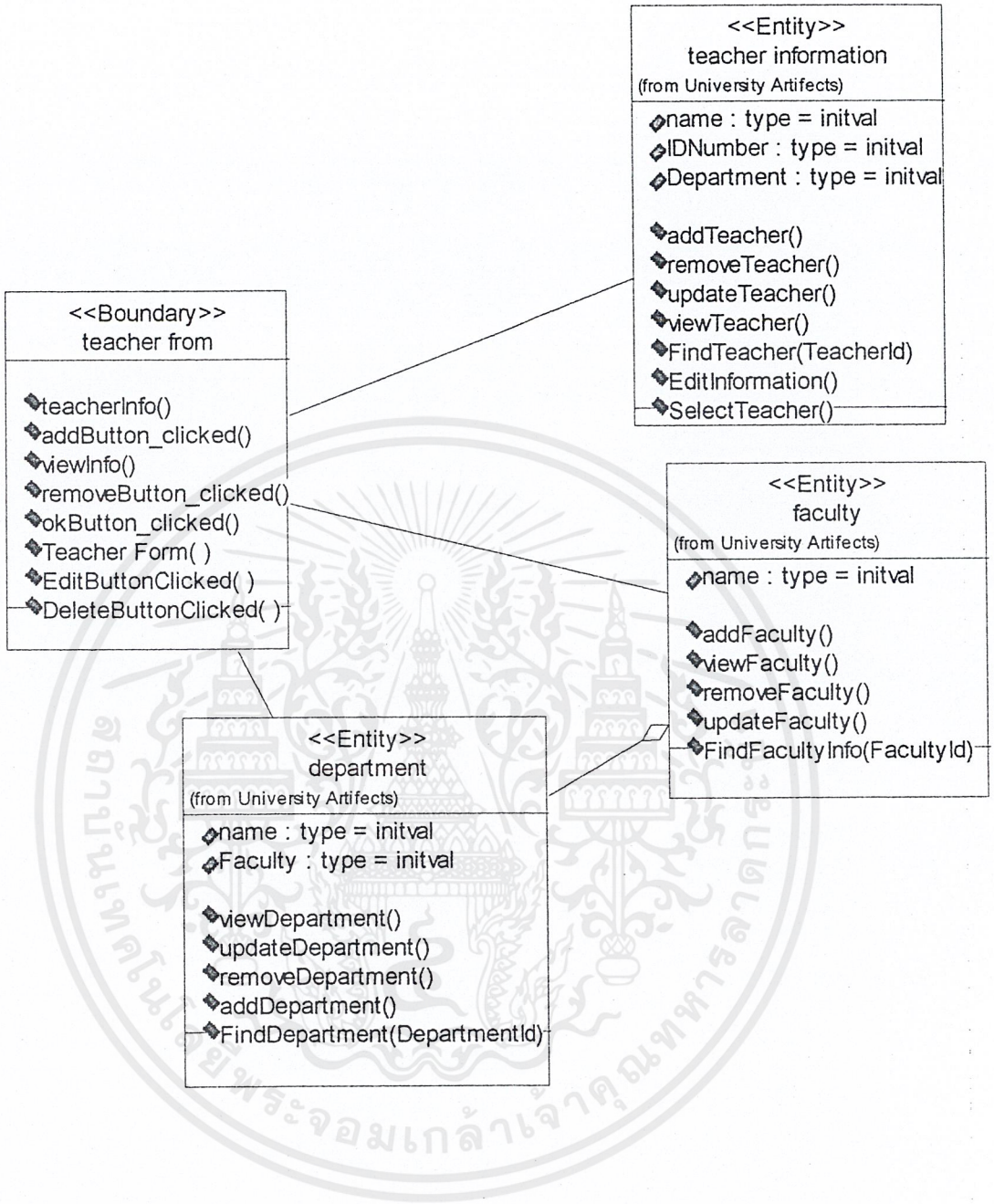
รูปที่ 4.36 แสดง Sequence diagram ของ verify result grad function



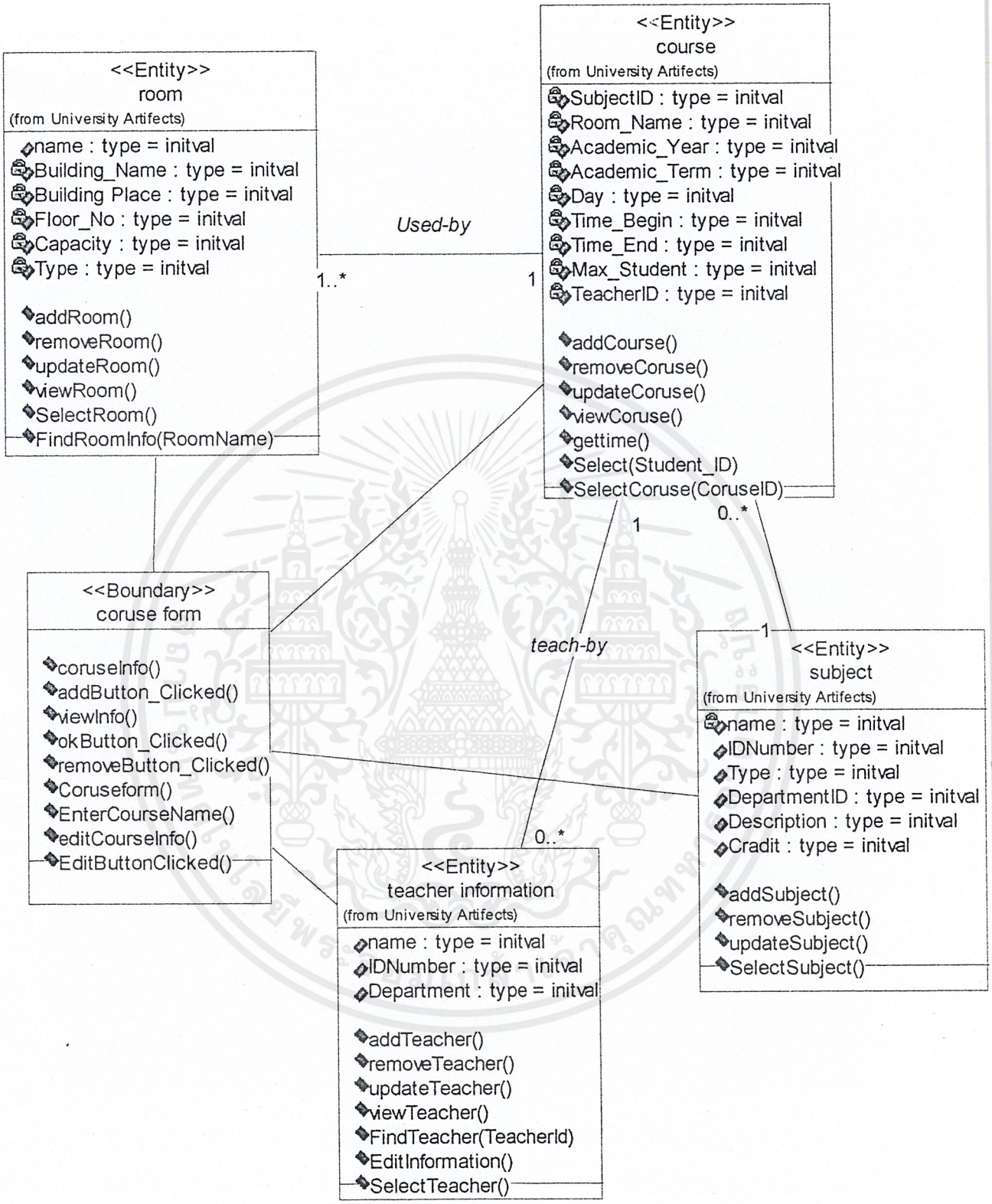
รูปที่ 4.37 class diagram : Interface/register form class แสดงความสัมพันธ์ของคลาส register form กับคลาสอื่นๆ



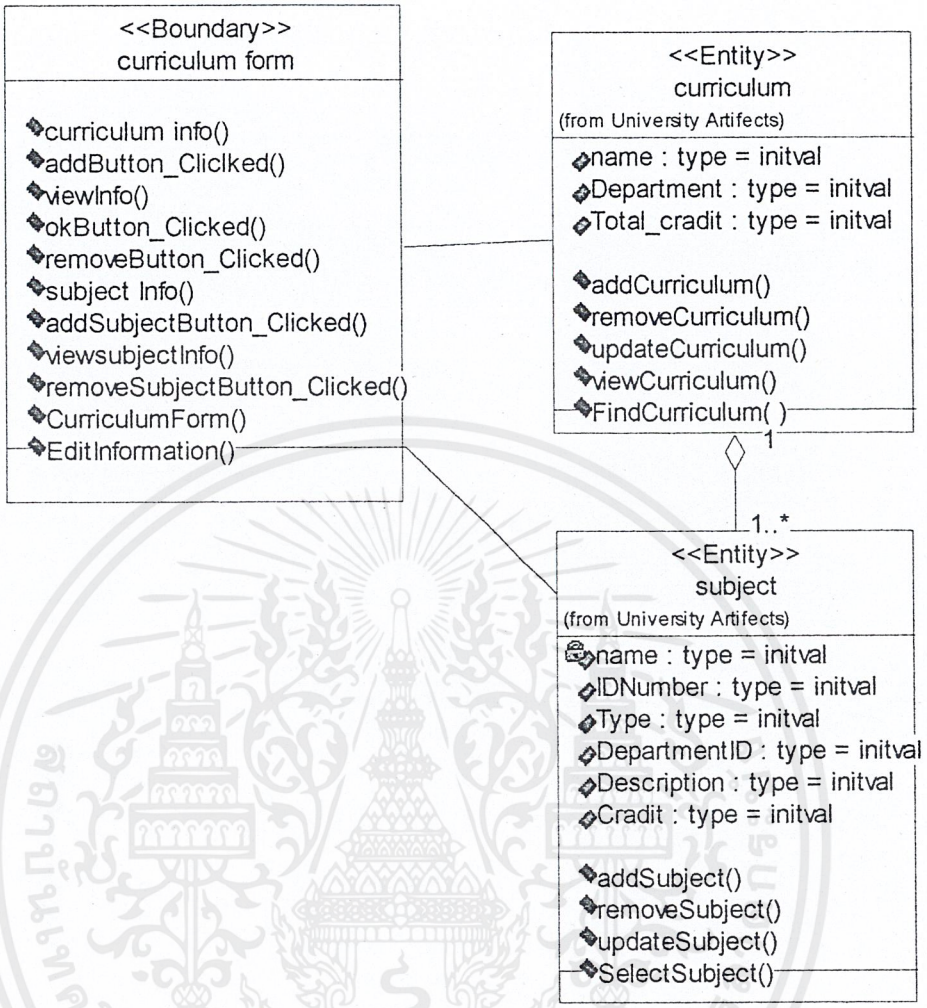
รูปที่ 4.38 class diagram : Interface/list student form class แสดงความสัมพันธ์ของคลาส list student form กับคลาสอื่นๆ



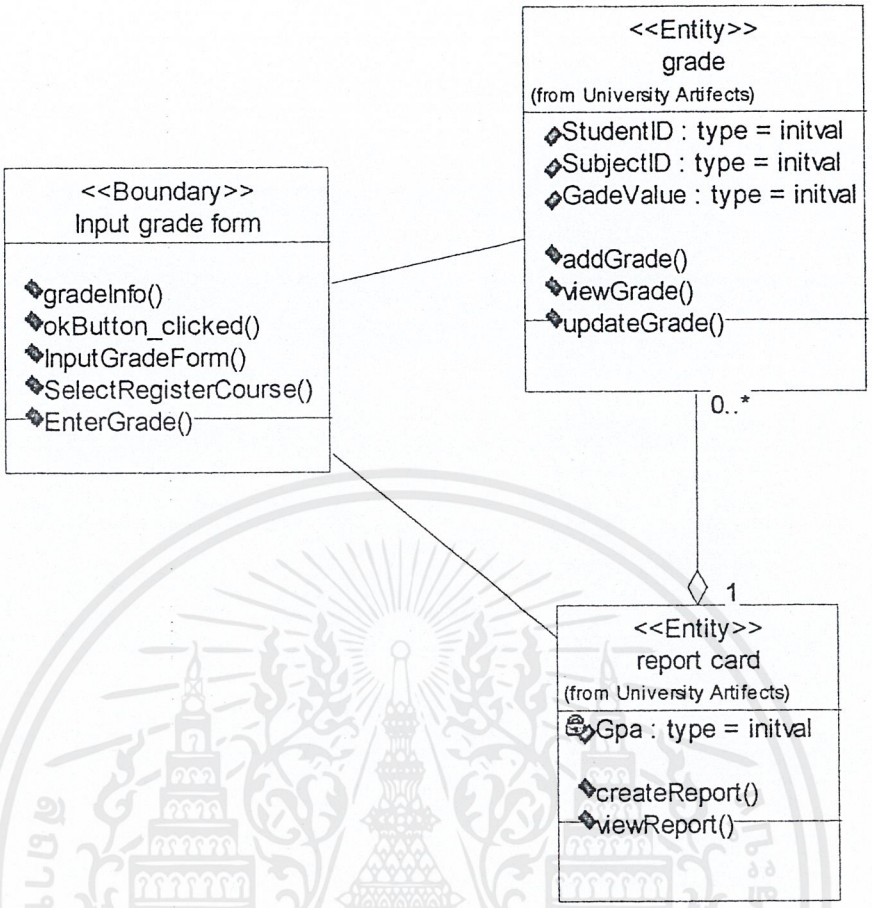
รูปที่ 4.39 class diagram : Interface/teacher form class แสดงความสัมพันธ์ของคลาส teachert form กับคลาสอื่นๆ



รูปที่ 4.40 class diagram : Interface/course form class แสดงความสัมพันธ์ของคลาส courset form กับคลาสอื่นๆ



รูปที่ 4.41 class diagram : Interface/cirriculum form class แสดงความสัมพันธ์ของคลาส cirriculum form กับคลาสอื่นๆ



รูปที่ 4.42 class diagram : Interface/input grade form class แสดงความสัมพันธ์ของคลาส input grade form กับคลาสอื่นๆ

บทที่ 5
การทดลอง ผลการทดลอง

5.1 ฮาร์ดแวร์และซอฟท์แวร์ที่ใช้ในการทดลอง

เครื่องคอมพิวเตอร์ที่ใช้

- หน่วยประมวลผลกลาง เพนเทียม 133 เมกะเฮิร์ต (CPU Pentium 133 MHz)
- หน่วยความจำหลัก 80 เมกะไบท์ (Main Memory 80 MB)
- ฮาร์ดดิสค์ 6 จิกะไบท์ (Harddisk 6 GB)

ซอฟท์แวร์

- ระบบปฏิบัติการ (Operating System) Windows NT
- MS SQL Server version 6.5
- MS Visual C++ version 6.0

การใช้งานโปรแกรมจะติดต่อกับ DBMS โดยผ่าน ODBC โดยการทดลองทำบน Local Server

5.2 การทดลอง

การทดลองสร้าง คลาสที่ติดต่อกับ database เฉพาะ ขึ้นมา เพื่ออำนวยความสะดวกในการทำงานด้านทะเบียน เมื่อใช้งานในระบบเครือข่ายจะช่วยลดความแออัดในเครือข่าย โดยสร้างเป็นคลาสที่ทำการติดต่อกับ ODBC โดยจะดึงข้อมูลในตารางขึ้นมาสร้างในรูป Collection ของออบเจ็กต์

การ insert , delete , update จะทำบนออบเจ็กต์ ใน collection จนกระทั่งเมื่อเลิกใช้ collection ก็จะทำการ save ข้อมูลลง database

ผลการทดลอง

- ระหว่างการนำเข้าข้อมูลลงบนออบเจ็กต์ ใน collection เป็นไปได้ด้วยดี ทั้งการนำเข้าข้อมูล แก้ไข หรือ ลบข้อมูล
- หลังจากออกจากหน้าจอ ในกรณีที่ข้อมูลที่ใส่ลงไปไม่มีข้อผิดพลาด จะสามารถทำงานต่อไปได้อย่างถูกต้อง
- กรณีที่มีข้อผิดพลาด (error) ของข้อมูลที่ใส่หรือแก้ไขจะมีการแจ้ง error มาจาก ODBC และนอกจาก record ที่มีข้อผิดพลาด จะไม่สามารถ update ได้ถูกต้องและยังส่งผลให้การทำงานอื่นๆ ไม่ว่าจะเป็นการ insert , delete , update record อื่นๆไม่สามารถทำได้
- ในกรณีที่มีข้อมูลจำนวนมากขึ้นจะเสียเวลาในการ initialize และการนำข้อมูลลง database แต่ไม่มากนัก ซึ่งอาจเป็นเพราะใช้งานอยู่บนเครื่องเดียวกัน

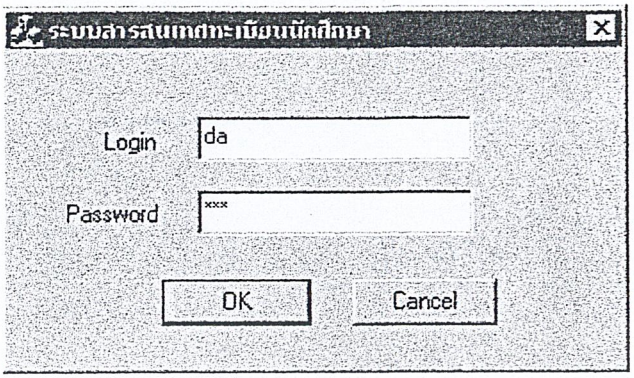
ผลจากการทดลองทำให้เห็นได้ว่ายังไม่สามารถใช้งานคลาสที่ติดต่อกับ database ที่ทดลองสร้างขึ้น ไปใช้งานทะเบียนจริงได้ เนื่องจากยังขาดความถูกต้อง

การทดลองใช้ Microsoft Foundation Class Library ทำการติดต่อกับ database

- ทดลอง insert ข้อมูลปรากฏว่าสามารถ insert ข้อมูลได้อย่างถูกต้อง
- delete ข้อมูลได้อย่างถูกต้อง
- update เกิดปัญหา error โดยไม่ทราบสาเหตุซึ่งจะเป็นเฉพาะการ update ในบางครั้งเท่านั้น ส่งผลให้โปรแกรมมีข้อผิดพลาดอยู่และยังไม่มีเสถียรภาพมากพอ

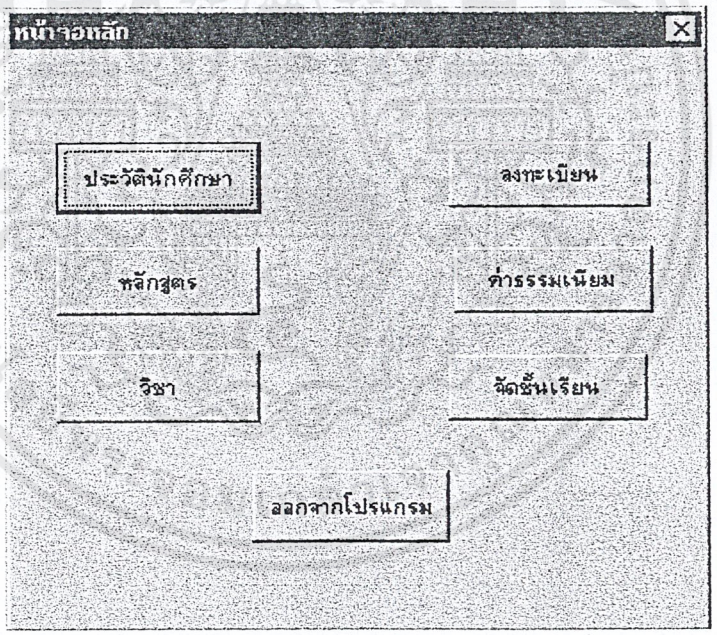
หน้าจอแสดงการทำงาน

1.หน้าจอตรวจสอบสิทธิ์ในการใช้โปรแกรม



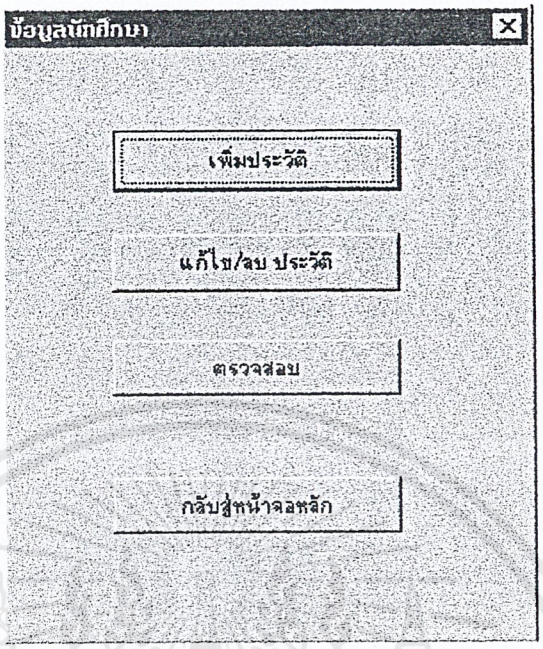
ป้อน login และ password กด OK ระบบจะตรวจสอบ login และ password กับ SQL Server ถ้าถูกต้องก็จะผ่านเข้าระบบได้

2.หน้าจอหลัก



หน้าจอหลักให้เลือกการทำงานต่างๆ หรือออกจากโปรแกรม

3.หน้าจอข้อมูลนักศึกษา

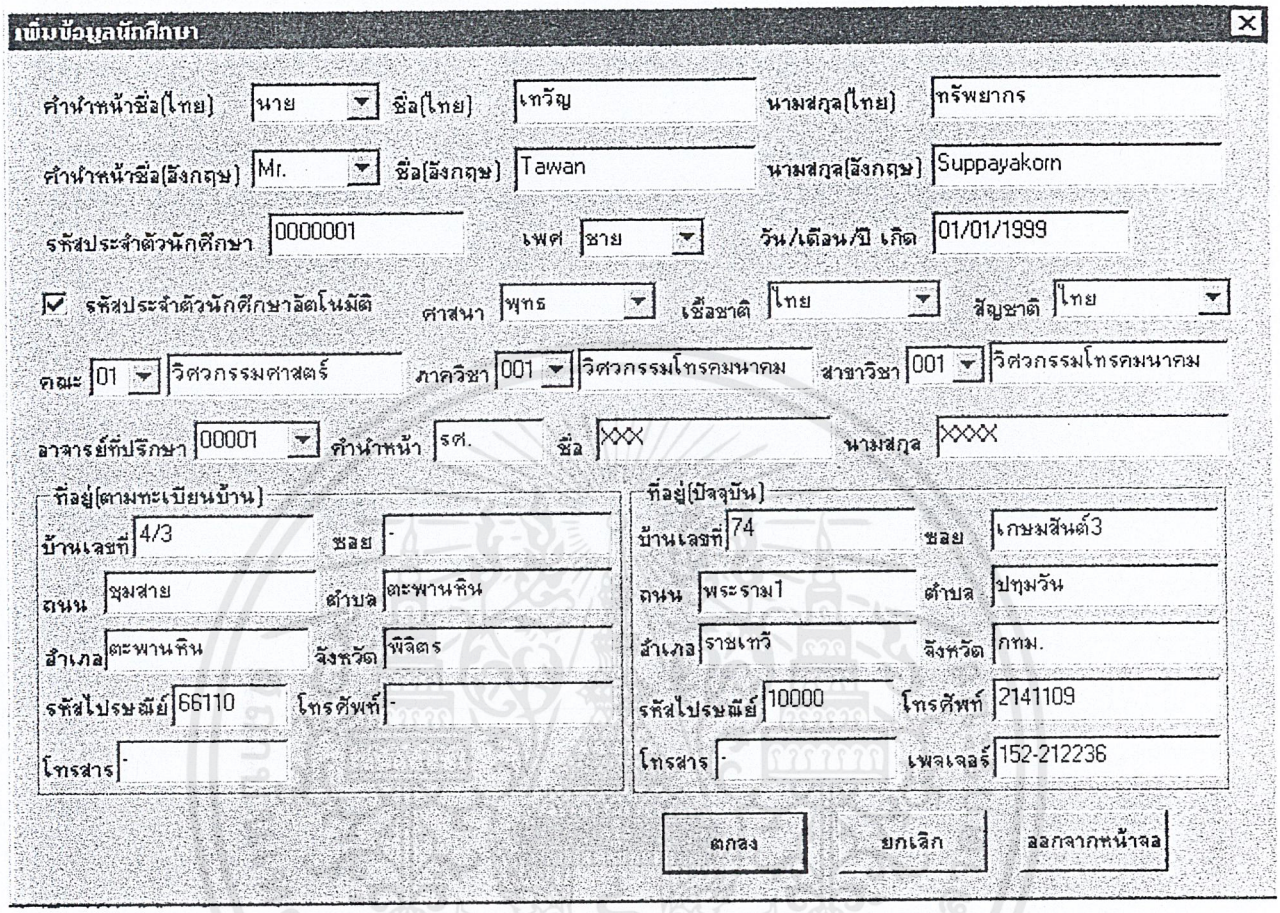


เมื่อเลือกการทำงาน “ประวัตินักศึกษา” ระบบจะแสดงหน้าจอข้อมูลนักศึกษา ให้เลือกการทำงาน หรือกลับสู่หน้าจอหลัก

เลือกการทำงาน ดังนี้

- เพิ่มประวัตินักศึกษา
- แก้ไข / ลบ ประวัตินักศึกษา
- ตรวจสอบ
- กลับสู่หน้าจอหลัก

4.หน้าจอเพิ่มข้อมูลนักศึกษา



เมื่อใส่ข้อมูลเสร็จ เลือกการทำงาน

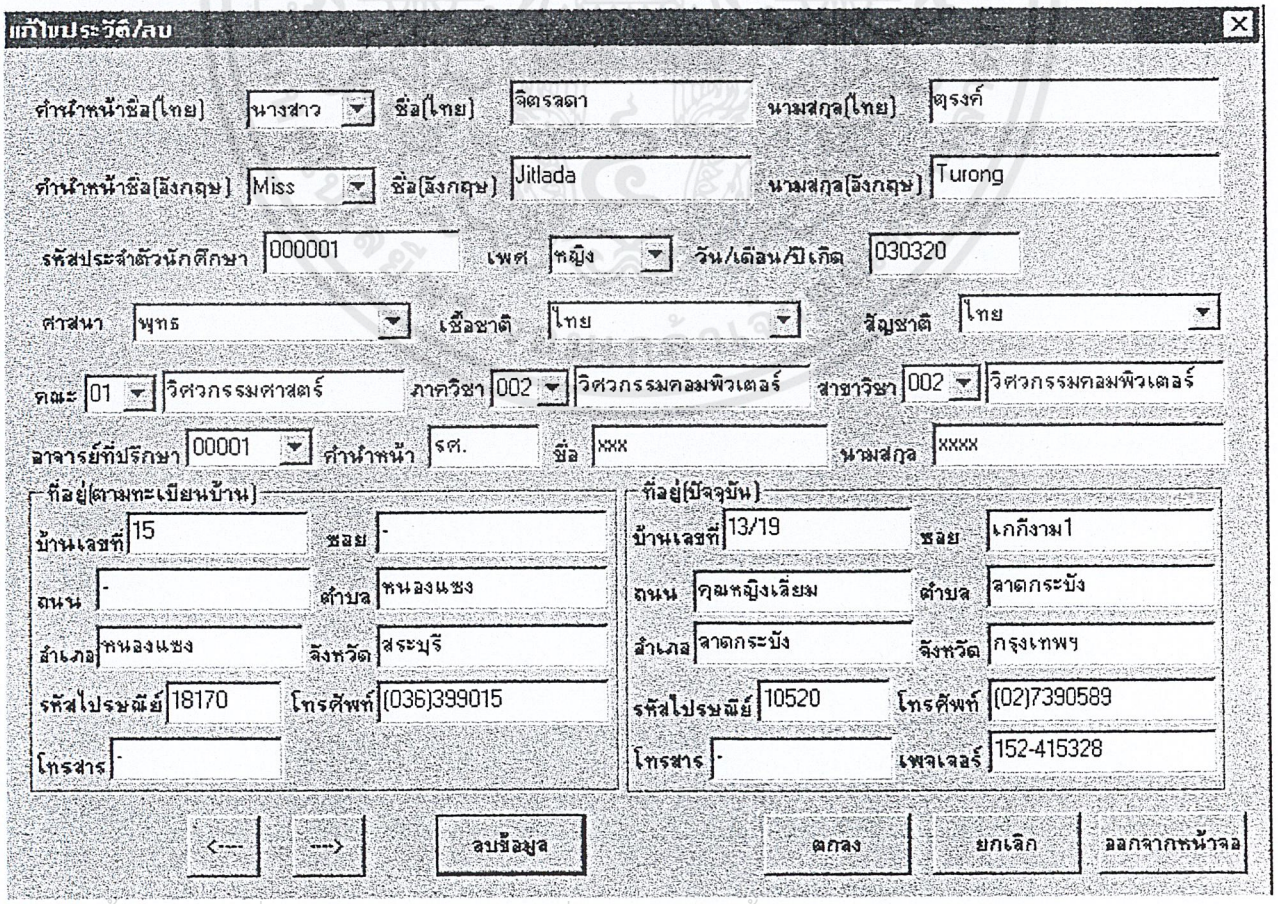
- ตกลง เพื่อเก็บข้อมูลลงในฐานข้อมูล
- ยกเลิก เมื่อไม่ต้องการเก็บข้อมูลลงในฐานข้อมูล
- ออกจากหน้าจอ เพื่อกลับสู่หน้าจอ"ข้อมูลนักศึกษา"

5.หน้าจอเลือกนักศึกษา

เมื่อเลือกทำการลบ / แก้ไข ข้อมูลนักศึกษา ระบบจะแสดงหน้าจอให้เลือกนักศึกษา

- เลือกตามรหัสนักศึกษา กรอกรหัสนักศึกษา ระบบจะแสดงข้อมูลนักศึกษารหัสดังที่เลือก
- เลือกตามชื่อนักศึกษา กรอกชื่อนักศึกษา ระบบจะแสดงข้อมูลนักศึกษาตามชื่อที่เลือก
- เลือกทั้งหมด ระบบจะแสดงข้อมูลนักศึกษาได้ทุกคนโดยใช้ปุ่มเลื่อน ดังหน้าจอถัดไป

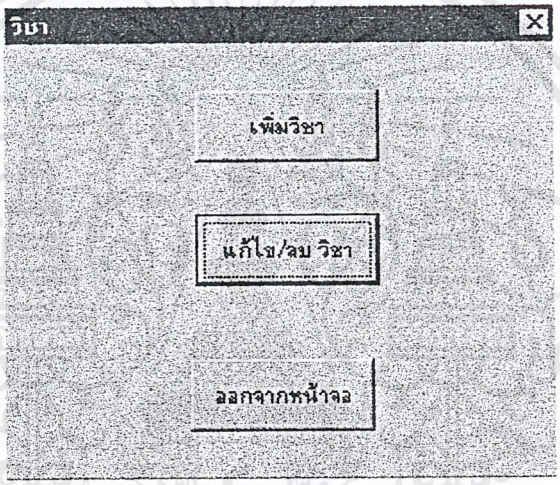
6.หน้าจอแก้ไข/ลบ ข้อมูลนักศึกษา



เลือกการทำงาน

- "-->" เลื่อนไปข้อมูลใน Record ถัดไป
- "<--" เลื่อนไปข้อมูลใน Record ก่อนหน้า
- "ลบข้อมูล" เพื่อลบข้อมูลที่ Record ปัจจุบัน
- "ตกลง" เพื่อทำการแก้ไขข้อมูล
- "ยกเลิก" เพื่อยกเลิกการแก้ไขข้อมูล
- "ออกจากหน้าจอ" เพื่อกลับไปยังหน้าจอข้อมูลนักศึกษา

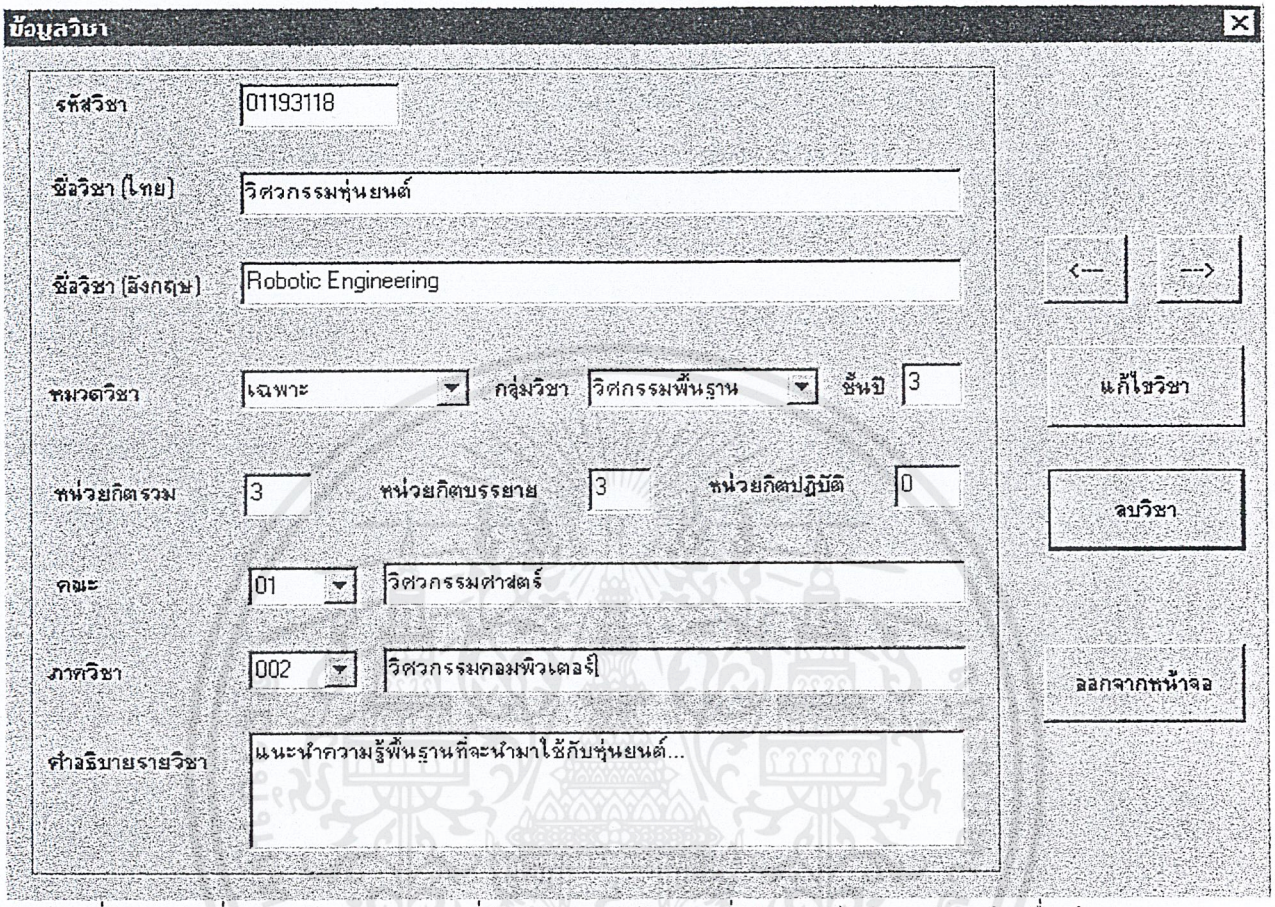
7.หน้าจอข้อมูลวิชา



เลือกการทำงาน ดังนี้

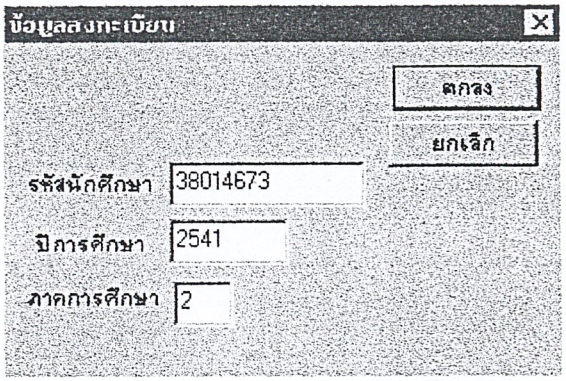
- เพิ่มวิชา เมื่อต้องการเพิ่มข้อมูลวิชาลงในฐานข้อมูล
- แก้ไข/ลบวิชา เมื่อต้องการเปลี่ยนแปลงข้อมูลวิชาในฐานข้อมูล
- ออกจากหน้าจอ เพื่อกลับสู่หน้าจอหลัก

8. หน้าจอแก้ไข/ลบข้อมูลวิชา



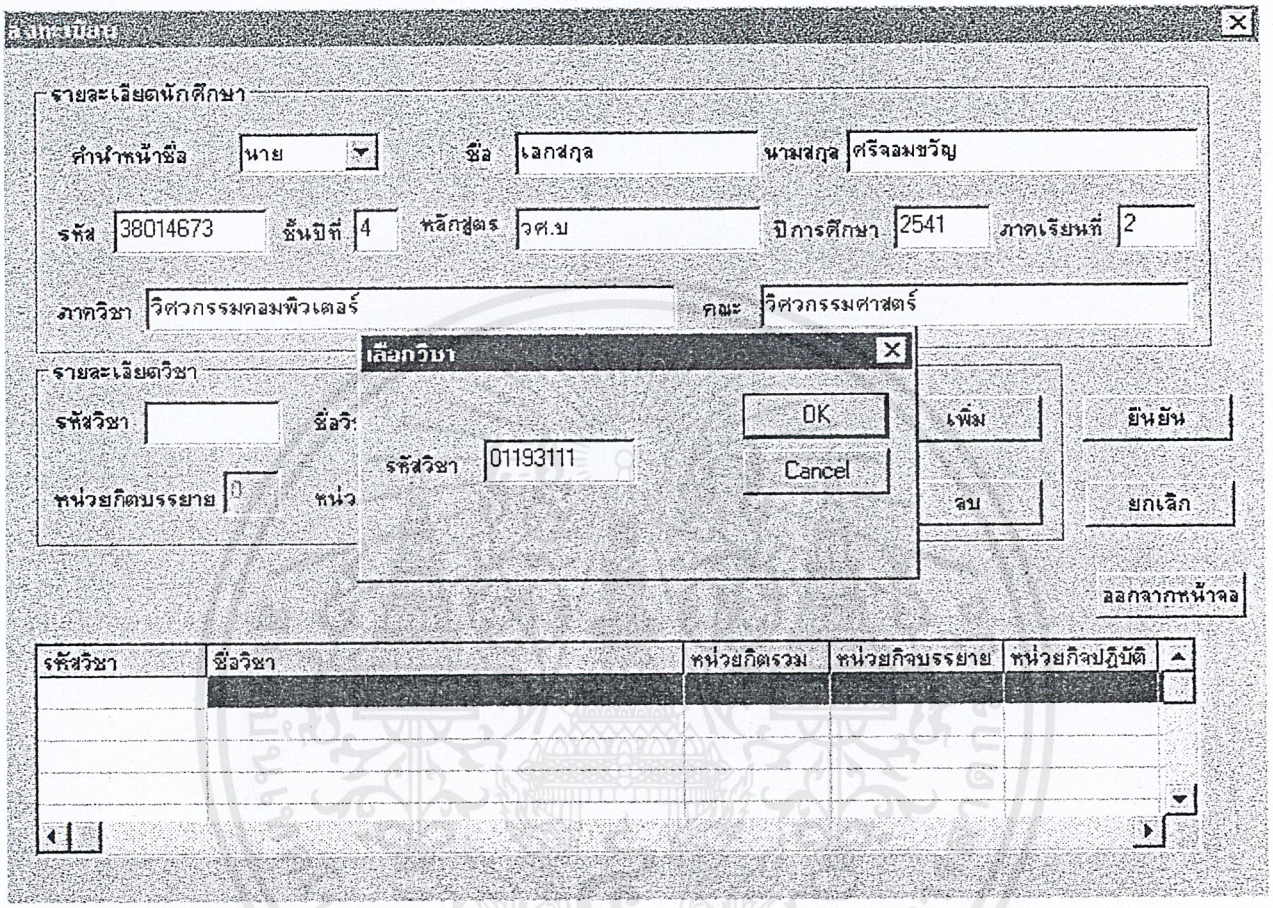
เลือกวิชาที่ต้องการเปลี่ยนแปลงโดยกดปุ่ม → เพื่อไปยังข้อมูลถัดไป ← เพื่อย้อนกลับไปข้อมูลก่อนหน้า เมื่อแก้ไขข้อมูลที่ต้องการเสร็จเรียบร้อย กดปุ่มแก้ไข หากต้องการลบข้อมูลวิชากดปุ่ม ลบวิชา หรือกดปุ่มออกจากหน้าจอเพื่อกลับสู่หน้าจอข้อมูลวิชา

10.หน้าจอข้อมูลลงทะเบียน



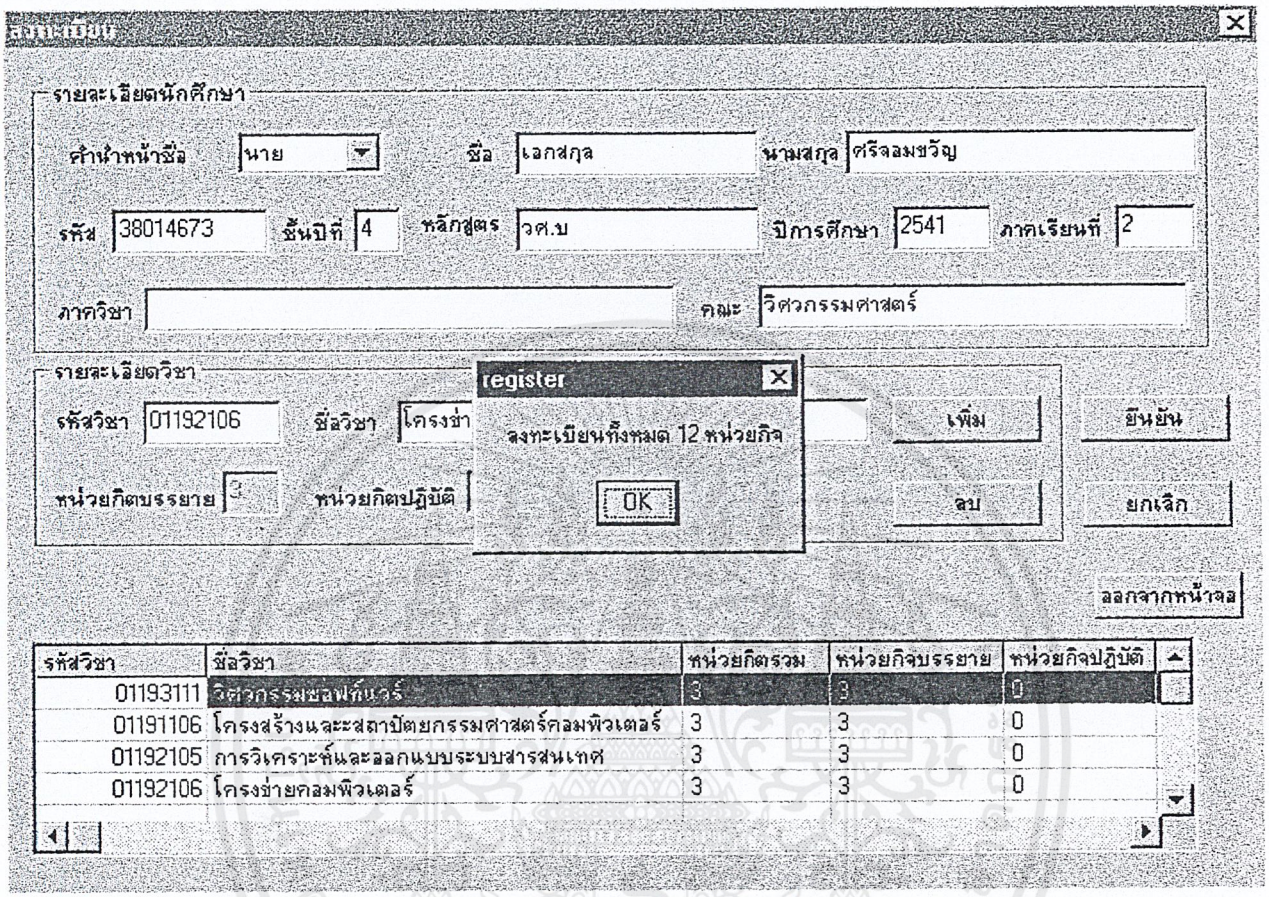
เมื่อเลือกทำการลงทะเบียน จะปรากฏหน้าจอให้เลือกนักศึกษาที่ต้องการลงทะเบียนและใส่ปีและภาคการศึกษา กดปุ่ม ตกลงจะปรากฏหน้าจอลงทะเบียน

11.หน้าจอลงทะเบียน



- เพิ่มรายวิชาที่ต้องการลงทะเบียน โดย กดปุ่มเพิ่ม ใส่รหัสวิชาที่ต้องการแล้วกดปุ่มตกลง รายละเอียดของวิชาที่เลือกจะปรากฏใน list
- ลบรายวิชาที่ทำการลงทะเบียนออกจาก list โดยเลือกรายวิชาที่ต้องการลบ แล้วกดปุ่มลบ
- ยืนยันการลงทะเบียน กดปุ่ม ยืนยันเพื่อยืนยันการลงทะเบียน ระบบจะแสดงจำนวนหน่วยกิตที่ลงทะเบียนทั้งหมด ตอบตกลง ข้อมูลจะถูกบันทึกลงฐานข้อมูล
- ยกเลิกการลงทะเบียน เมื่อไม่ต้องการลงทะเบียนตามรายวิชาที่เลือก
- ออกจากหน้าจอ เมื่อต้องการออกจากหน้าจอการลงทะเบียน

12. หน้าจอลงทะเบียนเมื่อกดปุ่มยืนยัน



บทที่ 6
สรุปและวิจารณ์ผลการทดลอง

6.1 สรุป

โครงงานนี้สามารถจำลองระบบงานทะเบียนสารสนเทศเพื่อใช้เป็นแนวทางในการพัฒนาระบบที่สมบูรณ์และใช้งานจริง หรือระบบงานอื่นๆแบบ Object Oriented โดยใช้ UML โดยที่ซอฟท์แวร์ที่พัฒนาขึ้นนั้นไม่สามารถใช้งานได้จริง

ในตอนเริ่มต้นโครงงานนี้ ผู้จัดทำมีความคิดที่จะพัฒนาระบบทะเบียนสารสนเทศเพื่อให้ใช้งานได้จริง แต่เนื่องจากหลังจากการศึกษาแล้วพบว่าระบบจะมีความซับซ้อนมาก ประกอบกับเวลาที่จำกัด และที่สำคัญคือผู้จัดทำขาดประสบการณ์ทางด้านการพัฒนาซอฟท์แวร์ในกระบวนการต่างๆ ทำให้เสียเวลาในการศึกษามากเกินไป ทำให้ไม่สามารถสร้างซอฟท์แวร์ให้ทำงานได้ทั้งหมด

6.2 แนวทางในการพัฒนาต่อไป

1. ซอฟท์แวร์ที่ได้จากโครงงานนี้สามารถใช้เป็น Prototype ในการพัฒนา เพื่อช่วยในการเก็บ requirement จาก user ได้ เพื่อให้ผู้พัฒนาเข้าใจความต้องการของระบบในรายละเอียดได้มากขึ้นกว่าการสัมภาษณ์เพียงอย่างเดียว
2. หลังจากได้เก็บรายละเอียดเพิ่มเติมจาก Prototype สามารถนำ diagram ต่างๆที่มีอยู่มาดัดแปลงแก้ไขให้เป็นไปตามที่ผู้ใช้ต้องการได้
3. เนื่องจากซอฟท์แวร์ได้มีการใช้การพัฒนาแบบ object oriented บางส่วนของโปรแกรม เช่น User Interface สามารถนำไปใช้ในโครงงานที่จะพัฒนาต่อไปได้
4. ควรเปลี่ยน Database ไปเป็น Object Oriented Database เพื่อให้สามารถใช้ความสัมพันธ์ที่มีความซับซ้อนขึ้น เช่น Inheritance ซึ่งไม่สามารถแทนความสัมพันธ์แบบนี้ไว้ใน relational database ได้

บรรณานุกรม

[1] Martin Fowler with Kendall Scott : "UML Distilled : Applying the Standard Object Modeling Language", Addison Wesley Longman,Inc., December 1997.

[2] Hans-Erik Eriksson and Magnus Penker :"UML Toolkit", John Wiley & Son, Inc., Wiley Computer Publishing, 1998.

[3] Craig Larman : "Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design", Prentice Hall PTR, 1998.

[4] Daniel Tkach, Richard Puttick : "Object Technology In Application Development", Addison-Wesley Publishing Company, 1996.

[5] Terry Quatrani : "Visual Modeling with Ratonal Rose And UML", Addison-Wesley Longman,Inc., 1998.

[6] Setrag Khoshafian : "Object-Oriented Database", John Wiley & Sons,Inc., 1993

[7] Davis Chapman : "SAMS Teach Yourself Visual C++ 6 ", SAMS Publishing, 1998.

[8] Robert Lafore : "Object-Oriented Programming in C++ Third Edition", SAMS Publishing, 1999.

[9] Lyn Robinson : "SAMS Teach yourself Database Programming with Visual C++ 6", SAMS Publishing, 1999.

[10] นิรุธ อำนวยศิลป์ : "คู่มือการเขียนโปรแกรม Microsoft Visual C++ Version 6.0 ฉบับเพื่อการใช้งานจริง", บริษัท ซัคเซส มีเดีย จำกัด, 1999.

[11] John Hubbard : "Theory and Ploblems of Programming with C++", Mc Graw Hill,Inc., 1996.

[12] Beck Zaratian : "The Essential Guide to Microsoft Visual C++ 6.0 : Microsoft Visual C++ 6.0 Programmer's Guide", Microsoft Press, 1998.

[13] Microsoft Corporation (1993) : "Microsoft Visual C++ User's Guide", Microsoft Press, 1994.

[14] สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง : คู่มือนักศึกษา 2541 คณะวิศวกรรมศาสตร์