โครงงานผู้ช่วยผู้ดูแลระบบฐานข้อมูลออราเคิล

Oracle Administrator Assistance System

โดย

นายอภิวัฒน์ สังข์ทอง

นายเอกพงษ์ ภิรมย์โสภา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

หัวข้อปริญญานิพนธ์	โครงงานผู้ช่วยผู้ดูแลระบบฐานข้อมูลออราเคิล
Title	Oracle Administrator Assistance System
โดย	นายอภิวัฒน์ สังข์ทอง รหัสประจำตัว 41014520
	นายเอกพงษ์ ภิรมย์โสภา รหัสประจำตัว 41014547
อาจารย์ผู้ควบคุมปริญญานิพนธ์	อาจารย์สุธีรา พันธุ์ธีรานุรักษ์
	อาจารย์ภุชงค์ หงษ์สุวรรณ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2544

ปริญญานิพนธ์ฉบับนี้ ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(อาจารย์สุธีรา พันธุ์ธีรานุรักษ์)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	ระบบผู้ช่วยผู้ดูแลระบบฐานข้อมูลออราเคิล			
นักศึกษา	นายอภิวัฒน์	สังข์ทอง	รหัสประจำตัว	41014520
	นายเอกพงษ์	ภิรมย์โสภา	รหัสประจำตัว	41014547
อาจารย์ผู้ควบคุมปริญญานิพนธ์	อาจารย์สุธีรา	พันธุ์ธีรานุรักษ์		
	อาจารย์ภุชงค์	หงษ์สุวรรณ		
ระดับการศึกษา	ปริญญาวิศวกรรมศาสตรบัณฑิต			
ภาควิชา	วิศวกรรมสารสนเทศ			
ปีการศึกษา	2544			

บทคัดย่อ

ในปัจจุบันความสำคัญของข้อมูลข่าวสารเพิ่มมากขึ้น ทำให้ต้องมีระบบการจัดเก็บข้อมูลที่มีประสิทธิภาพ และออราเคิลเป็นระบบจัดการฐานข้อมูลที่มีผู้นิยมใช้เป็นจำนวนมาก และเนื่องจากออราเคิลเป็นโปรแกรมที่พัฒนามาจากต่างประเทศ ดังนั้นอาจจะมีปัญหาเกิดขึ้นในระหว่างการใช้งานได้ หากผู้ดูแลระบบไม่ทราบรายละเอียดต่างๆ เกี่ยวกับโปรแกรมอย่างแท้จริงแล้ว ในขณะที่มีปัญหาเกิดขึ้นอาจจะไม่สามารถแก้ไขปัญหาเหล่านั้นได้ ทำให้ผู้ดูแลระบบต้องติดต่อกับบริษัทผู้ผลิตโดยตรงส่งผลให้เสียเวลา และค่าใช้จ่ายเป็นจำนวนมาก ดังนั้นจึงได้มีแนวคิดที่จะรวบรวมปัญหาที่เกิดขึ้นบ่อยๆไว้ และนำเสนอวิธีการแก้ปัญหาในรูปของมัลติมีเดีย เพื่อให้ผู้ดูแลระบบสามารถแก้ปัญหาต่างๆได้ด้วยตนเอง

PROJECT TITLE	ORACLE ADMINISTRATOR ASSISTANCE SYSTEM
STUDENT	Mr. Apiwat Sangthong No.41014520
	Mr. Akegapong Piromsopa No.41014547
ADVISOR	Ms. Sutheera Puntheeranurak
	Mr. Puchong Hongsuwan
COURSE	Bachelor of Information Engineering
DEPARTMENT	Information Engineering
YEAR	2001

ABSTRACT

At the present, information is very important. Efficiency database system has to be created to manage the information. Oracle, Database manage system, is very popular and is used in many offices. But Oracle is a program that is developed from another country. Perhaps some problem might happen when using oracle. If database administrators are not good at using oracle, they have to contact company directly. It causes them to loose their money and have to spend more time to solve the problem then we made the Oracle Administrator Assistance System. It can collect many kind of problem that always happen when using oracle. So database administrator can find how to solve the problem easily and manage their work faster than ever.

กิตติกรรมประกาศ

ขอขอบพระคุณอาจารย์ที่ปรึกษา อาจารย์สุธีรา พันธุ์ธีรานุรักษ์ และ อาจารย์ภุชงค์ หงษ์สุวรรณ เป็นอย่างสูงที่คอยให้คำแนะนำ คำปรึกษาอย่างเต็มที่ ตลอดการทำปริญญานิพนธ์ระบบผู้ช่วยผู้ดูแลฐานข้อมูลออราเคิล

ขอขอบพระคุณคุณพ่อคุณแม่และอาจารย์ทุกท่านที่ให้ความสนับสนุนและกำลังใจยามเกิดปัญหาจนกระทั่งปริญญานิพนธ์นี้เสร็จสมบูรณ์เป็นรูปเล่มได้

อภิวัฒน์ สังข์ทอง
เอกพงษ์ ภิรมย์โสภา

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 แนวคิดและที่มาของปัญหา

ในปัจจุบันที่มีการนำเอาระบบฐานข้อมูลมาช่วยในการเก็บข้อมูลต่างๆในองค์กรทางธุรกิจ จึงจำเป็นที่จะต้องมีผู้ดูแลระบบฐานข้อมูลเพื่อจัดการด้านการจัดเก็บข้อมูล ในอดีตเมื่อระบบเกิดปัญหาซึ่งผู้ดูแลระบบไม่สามารถแก้ไขได้ขึ้น ผู้ดูแลระบบจำเป็นต้องติดต่อกับฝ่ายช่วยเหลือของบริษัทเจ้าของโปรแกรมฐานข้อมูลนั้น ๆ เพื่อขอคำแนะนำในการแก้ไขปัญหาซึ่งทำให้เกิดความล่าช้าในการติดต่อ และผู้ดูแลระบบอาจจะทำตามขั้นตอนที่ฝ่ายช่วยเหลือแนะนำมาไม่ได้ จึงได้มีแนวคิดในการสร้างโครงงานระบบผู้ช่วยผู้ดูแลระบบฐานข้อมูลขึ้น ซึ่งข้อดีของโครงงานระบบผู้ช่วยผู้ดูแลระบบฐานข้อมูล ก็คือสามารถช่วยให้ผู้ดูแลระบบประหยัดเวลา และค่าใช้จ่ายในการติดต่อกับฝ่ายช่วยเหลือของบริษัทผู้ผลิต โครงงานระบบผู้ช่วยผู้ดูแลระบบฐานข้อมูล เป็นการนำเสนอวิธีการแก้ไขปัญหาของระบบฐานข้อมูลโดยการจำลองหน้าจอของโปรแกรมฐานข้อมูล มีการแสดงผลเป็นภาษาไทยสำหรับผู้ที่ไม่ถนัดการใช้ภาษาอังกฤษ และแสดงวิธีการแก้ไขปัญหาเป็นขั้นตอนเพื่อช่วยให้เกิดความสะดวกกับผู้ดูแลระบบ

1.2 วัตถุประสงค์ของการทำโครงงาน

1. เพื่อศึกษาการพัฒนาแอปพลิเคชั่นโดยใช้ภาษาจาวา (Java Language)
2. เพื่อศึกษาการออกแบบระบบโดยใช้วิธีการของ UML (Unified Modeling Language)
3. เพื่อพัฒนาแอปพลิเคชั่นสำหรับช่วยแก้ไขปัญหาเกี่ยวกับการจัดการระบบฐานข้อมูล
4. เพื่อศึกษาการพัฒนาแอปพลิเคชันเพื่อช่วยในการเรียนรู้ (CAI)

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้ด้านการพัฒนาแอปพลิเคชันโดยใช้งานภาษาวิชวลเบสิค (Visual Basic Language)
2. สามารถออกแบบระบบโดยใช้วิธีการของ UML (Unified Modeling Language) ได้
3. สามารถพัฒนาแอปพลิเคชั่น สำหรับช่วยแก้ไขปัญหาเกี่ยวกับการจัดการระบบฐานข้อมูล ให้เหมาะสม เกิดประสิทธิภาพ และตรงตามความต้องการของผู้ดูแลระบบฐานข้อมูล มากที่สุด

4. ได้รับความรู้เกี่ยวกับการพัฒนาแอปพลิเคชันเพื่อช่วยในการเรียนรู้ (CAI)

1.4 ขอบเขตของปัญหา

1.4.1 เครื่องมือที่ใช้ในการพัฒนาระบบ

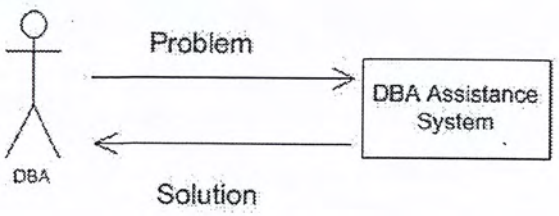
- Visual Basic Programming Language: เป็นเทคโนโลยีในการพัฒนาระบบที่ดี เพราะมีความสามารถที่โดดเด่นในเรื่องต่าง ๆ ดังนี้ Object-Oriented Programming
- Object-Oriented Technology: เป็นวิธีการที่ใช้ในการวิเคราะห์ปัญหา และ แก้ปัญหา เพราะ มองปัญหาและองค์ประกอบต่าง ๆ เป็นวัตถุ ซึ่งคุณสมบัติที่เด่นของวัตถุมีดังนี้ การแยกแยะเอกลักษณ์ (Abstraction), การสืบทอดคุณสมบัติ (Inheritance), การซ่อนรายละเอียด (Encapsulation), การส่งแมสแสจ (Message Sending), คอมโพซิชัน (Composition)

1.4.2 ความสามารถและคุณสมบัติของระบบ

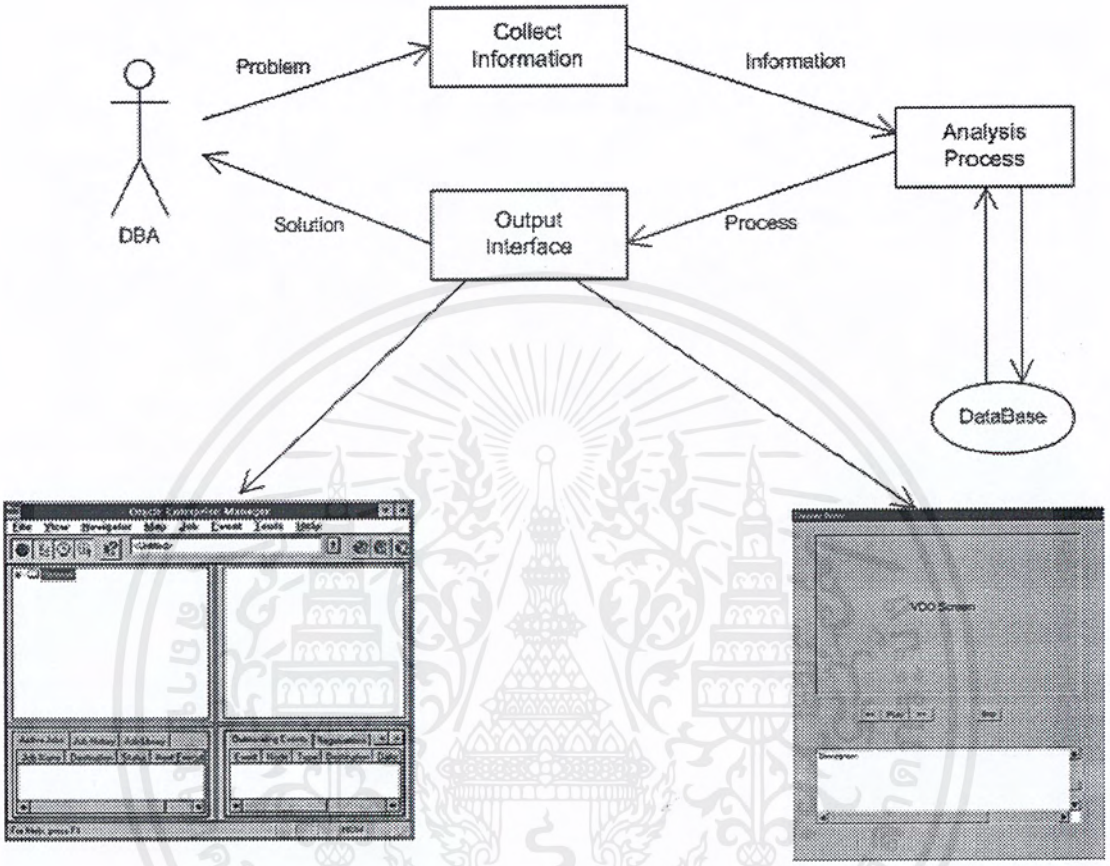
สามารถลดค่าใช้จ่าย, ประหยัดเวลาที่ใช้ในการแก้ไขปัญหาของระบบฐานข้อมูล และสามารถช่วยแก้ไขปัญหาของระบบฐานข้อมูลได้ดังต่อไปนี้

- SQL*Plus
- Database Users
- Tables
- Indexes
- Views
- Space Management
- SQL Statement Tuning
- Database Tuning

1.4.3 System Architecture



รูปที่ 1.1 แสดงภาพรวมของระบบ



รูปที่ 1.2 แสดงลักษณะการทำงานของระบบ

บทที่ 2

2.1 ทฤษฎีการออกแบบโปรแกรมเชิงวัตถุ(Object Oriented)

แนวความคิดของการออกแบบโปรแกรมเชิงวัตถุ (Object Oriented) เป็นการเปลี่ยนแปลงแนวคิดจากเดิมที่สนใจว่าจะเขียนตรรก ที่ใช้ในการควบคุมการทำงานอย่างไร โดยไม่สนใจว่าจะนิยามข้อมูลอย่างไร แต่ในการออกแบบโปรแกรมเชิงวัตถุ จะสนใจว่าวัตถุที่ถูกนิยามขึ้นมานั้นจะสามารถใช้งานได้อย่างไรมากกว่าที่จะสนใจในตรรกที่ใช้ควบคุม

ในชีวิตประจำวัน เมื่อมองดูรอบตัว เราจะพบเห็นวัตถุต่างๆมากมายไม่ว่าจะเป็นวัตถุที่เราสามารถเห็นได้และจับต้องได้ (Tangible Objects) เช่น โต๊ะ รถยนต์ คอมพิวเตอร์ หรือแม้แต่ตัวเราเองเป็นต้น และวัตถุที่มีอยู่จริงแต่ไม่สามารถจับต้องได้ (Intangible Objects) ตัวอย่างเช่น กฎหมาย เวลา หรือความรู้ต่างๆเป็นต้น

เราได้ทราบแล้วว่าในโลกของเรามีวัตถุต่างๆมากมายสิ่งที่เกิดขึ้นจากวัตถุต่างๆก็คือ กิจกรรม, ความเคลื่อนไหว, หรือการกระทำเช่น คนรับประทานอาหาร เป็นต้นหากพิจารณาโดยละเอียดแล้วจะพบว่ากิจกรรมต่างๆ ที่เกิดขึ้นในชีวิตประจำวันล้วนแล้วแต่เกิดจากการมีความสัมพันธ์และ การมีปฏิสัมพันธ์ (Interaction) ระหว่าง วัตถุ 2 ตัวขึ้นไป Object Oriented ก็มีลักษณะเช่นเดียวกัน ในขั้นตอนแรกของการออกแบบโปรแกรมเชิงวัตถุนั้น ขั้นตอนแรกก็คือการนิยามวัตถุว่าสามารถใช้งานอย่างไรได้บ้าง และ มีความสัมพันธ์อะไรบ้าง

ตัวอย่างที่1

นาย ก. เปิดตู้เย็นยี่ห้อ A (ซึ่งเป็นตู้เย็นของการนาย ก) แล้วหยิบน้ำ (ซึ่งอยู่ในตู้เย็น) มาดื่ม

จาก ตัวอย่าง เราสามารถแยกแยะเพื่อหา ความสัมพันธ์ของวัตถุ (Object Relationships) และการมีปฏิสัมพันธ์ได้ดังนี้

1. วัตถุที่เราสนใจ ในที่นี้ได้แก่ นาย ก, ตู้เย็น a, และน้ำ
2. ความสัมพันธ์ ระหว่างวัตถุที่เราสนใจ ได้แก่ ความสัมพันธ์ต่อไปนี้
   2.1. นาย ก. เป็นเจ้าของตู้เย็นยี่ห้อ A
   2.2. น้ำอยู่ในตู้เย็นยี่ห้อ A
3. การมีปฏิสัมพันธ์ระหว่างวัตถุที่เราสนใจ ได้แก่
   3.1. นาย ก. เปิดตู้เย็นห้อ A
   3.2. นาย ก. หยิบน้ำดื่ม

จากตัวอย่างข้างต้นนั้นถ้าสังเกตให้ดีจะพบว่า จะมีการพูดถึงแต่วัตถุที่เราสนใจ หรือความสัมพันธ์ที่เราสนใจเป็นต้น คำว่า "ที่เราสนใจ" นั้นเป็นการให้กรอบของสิ่งที่เราต้องการพิจารณาหรือสนใจ เพราะเราไม่สามารถสนใจทุกๆ วัตถุในโลกในเวลาเดียวกันได้ และในขณะเดียวกันเราเองก็ไม่สามารถสนใจในทุกๆ กิจกรรมที่เกิดขึ้นได้เช่นกัน จากตัวอย่างที่ 1 เราไม่สนใจว่าในตู้เย็นมีสิ่งอื่น เช่น ผลไม้อยู่ในตู้เย็นหรือไม่ และในขณะเดียวกันเราก็ไม่สนใจว่าน้ำที่นาย ก. ดื่มนั้นอยู่ในแก้ว หรือใส่อยู่ในภาชนะประเภทอื่นเป็นต้น กรอบของความสนใจนี้ เรียกว่า Domain ซึ่งใน Domain นั้นสามารถมี วัตถุได้ตั้งแต่ 2 ตัว จนนับไม่ถ้วน ในขณะเดียวกันวัตถุตัวเดียวกันก็สามารถอยู่ได้ในหลายๆ Domains ได้เช่นเดียวกัน ซึ่งนั่นขึ้นอยู่กับว่าเราจะกำหนด Domain ที่เราสนใจ (Domain of Interest) นั่นเอง

2.2 หลักการของการเขียนโปรแกรมเชิงวัตถุ

2.2.1 Abstraction และ Instantiation

Abstraction ก็คือการกระบวนการในการให้ แนวคิดที่จำเป็นให้กับ วัตถุจนเกิดเป็น คลาส ขึ้น การ Instantiation ก็คือการนำเอา คลาส ไปใช้งานจริง ตัวอย่างเช่น เมื่อเรานิยาม คลาส คนขึ้นมาให้มี แขน 2 แขน และ ขา 2 ขา Instance ของ คลาส ก็คือ นาย ก, นาย ข เป็นต้น

ชนิดของ Abstractions

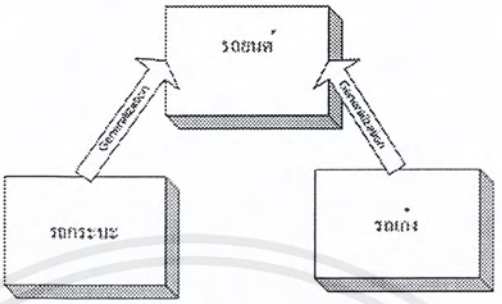
เราสามารถแบ่งชนิดของ Abstractions ได้เป็น 4 กระบวนการย่อย

1) Classification Abstraction
2) Aggregation Abstraction
3) Generalization Abstraction
4) Association Abstraction

กระบวนการ Classification Abstraction เป็นกระบวนการที่ใช้ ในการแยกประเภทวัตถุต่างๆที่อยู่ในความสนใจเพื่อให้ได้ คลาสพื้นฐานที่ต้องการ ยกตัวอย่างเช่น มี พนักงานฝ่ายจัดซื้อ, พนักงานฝ่ายขาย, พนักงานฝ่ายผลิต เราสามารถให้แนวคิดคือ 1. ชื่อ 2. ที่อยู่ และ 3. เงินเดือน เพื่อที่จะสร้างคลาสพื้นฐาน เป็นคลาส "คน"

กระบวนการ Aggregation Abstraction เป็นกระบวนการที่ใช้ในการสร้าง Class ใหม่ที่ใหญ่ขึ้น หรือซับซ้อนมากขึ้น โดยเกิดจากการรวมกันของ Class พื้นฐานหลายๆ Class ยกตัวอย่าง เช่น Class "รถยนต์" ประกอบด้วย Class "ตัวถังรถ", Class "ล้อ" และ Class "เครื่องยนต์" ส่วน Class "ตัวถังรถ"

กระบวนการ Generalization Abstraction เป็นกระบวนการที่ใช้ในการรวม Class ที่มีคุณสมบัติคล้ายคลึงกันหรือเหมือนกัน มารวมเป็น Class เดียวกัน ตัวอย่างเช่น เรามี Class "รถกระบะ", Class "รถเก๋ง" เราสามารถที่จะรวมกันเป็น Class ใหม่ คือ Class "รถยนต์" ได้



รูปที่ 2.1 รูปกระบวนการ Generalization

กระบวนการ Association Abstraction เป็นการใส่ความสัมพันธ์ให้กับ Class เพื่อบ่งบอกให้รู้ว่า Class แต่ละ Class มีความสัมพันธ์กันอย่างไร

2.2.2 Attributes และ Functions

Attributes คือ คุณสมบัติต่างๆของวัตถุซึ่งคุณสมบัติต่างๆเหล่านี้ต้องอยู่ในกรอบของ Domain ด้วย เช่น สีของประตูรถเป็นต้น

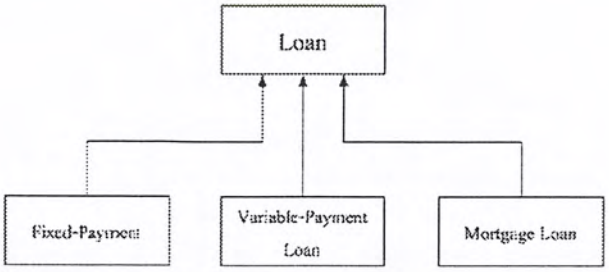
2.2.3 Function

Function คือ พฤติกรรมของ วัตถุที่ทำให้เกิดกิจกรรมต่างๆ

2.2.4 การสืบทอดคุณสมบัติ (Inheritance)

การสืบทอดคุณสมบัติเป็น แนวคิดในการนำ คลาส ของ วัตถุหรือคลาสย่อย ที่นิยามไว้แล้วมาสร้าง คลาส ใหม่ที่มีคุณลักษณะเหมือนกับ คลาส ต้นแบบทุกประการ

ตัวอย่างที่ 2 เรื่องการกู้ยืมเงิน



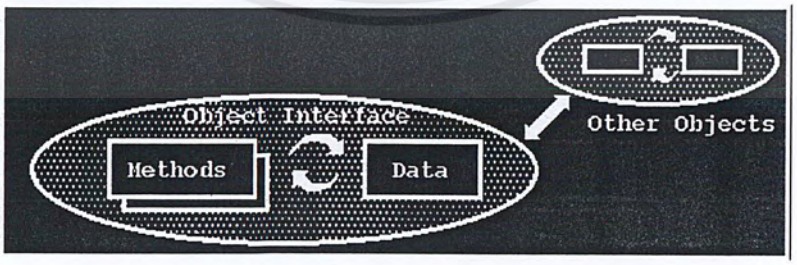
รูปที่ 2.2 รูปของตัวอย่างที่2

ภายใน Class แม่ ชื่อ "Loan" จะประกอบด้วย ตัวแปรชื่อ เงินต้น(Principle) และ อัตราดอกเบี้ย ดังนั้น คลาสลูกที่สืบทอดคุณสมบัติจากคลาสแม่ทั้ง 3 คลาส คือ Fixed-Payment , Variable-Payment Loan และ Mortgage Loan จะมี ตัวแปรทั้ง 2 ตัว ที่ได้รับการถ่ายทอดมาจาก Class "Loan" ด้วย (นอกจากนั้น Class ลูกทั้ง 3 Class นั้นยังสามารถมี ตัวแปรอื่นๆนอกเหนือจาก 2 ตัวที่กล่าวมาได้)

ประโยชน์ที่ได้จากการใช้ การสืบทอดคุณสมบัติ

- สามารถลดจำนวน ตัวแปรหรือ คุณสมบัติที่ซ้ำซ้อนระหว่าง คลาสที่สัมพันธ์กันได้
- สามารถทำความเข้าใจและปรับปรุงโครงสร้างของโปรแกรมง่ายขึ้น
- ง่ายในการสร้างคลาสที่สัมพันธ์กันขึ้นมาใหม่ เนื่องจากคลาสใหม่มี ตัวแปรบางส่วนสร้างและทดสอบไว้แล้ว

2.2.5 Encapsulation



รูปที่ 2.3 รูปการ Encapsulation

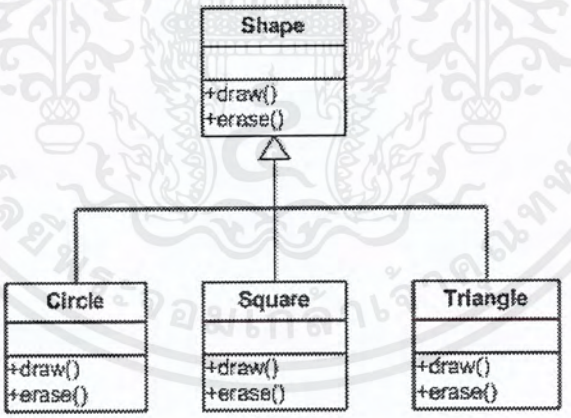
ในการเขียนโปรแกรมเชิงวัตถุ การติดต่อระหว่างกันของ วัตถุจะกระทำโดยผ่านข้อความ ทำให้วัตถุรู้จักแต่ Object's Interface ของ วัตถุที่ติดต่อด้วย ทำให้ข้อมูล และ ตรรกะถูกซ่อนไว้ไม่ให้ วัตถุอื่นเห็น เราเรียกกระบวนการนี้ว่า Encapsulate

การ Encapsulate มี 3 ชนิด

1. Private เป็นการกำหนดให้ ข้อมูล หรือ โอเปอร์เรชั่น(Operation) สามารถเข้าถึงได้เฉพาะคลาสเดียวกันเท่านั้น สามารถเข้าถึงได้ทั้งการอ่าน และเขียน
2. Protected เป็นการกำหนดให้สามารถเข้าถึงได้เฉพาะ คลาส นั้น และ คลาสที่มีการสืบทอดคุณสมบัติจาก คลาส นี้ไป (คลาสลูก) แต่มีข้อแตกต่างที่ ถ้าเป็นการเข้าถึงจาก Class นั้นจะสามารถอ่านและเขียนได้ แต่ถ้าเป็น คลาส ลูก(คลาสลูก) จะสามารถทำการอ่านได้เพียงอย่างเดียว
3. Public เป็นการกำหนดให้ ข้อมูลหรือ โอเปอร์เรชั่นสามารถเข้าถึงได้จากในคลาสเดียวกันและคลาสอื่น สามารถทำได้ทั้งการอ่านและการเขียน

2.2.6 Polymorphism

ในการเขียนโปรแกรมเชิงวัตถุ โพลิมอร์ฟิซึม(Polymorphism) เป็นวิธีการที่ทำให้สามารถใช้โอเปอร์เรชั่นที่มีชื่อเหมือนกันได้ แต่ว่าวิธีการทำงานแตกต่างกันตามคลาสที่เรียกใช้



รูปที่ 2.4 รูปกระบวนการ Polymorphism

จากรูปจะพบว่าคลาส"Circle", คลาส"Square", คลาส"Triangle" มีการสืบทอด โอเปอร์เรชั่น draw () และ erase () มาจาก คลาส" Shape" แต่การใช้งาน ฟังก์ชั่น จะมีความแตกต่างกัน โดยคลาส"Circle" จะเป็นการวาดภาพวงกลม คลาส"Square"จะเป็นการวาดภาพสี่เหลี่ยม เป็นต้น

การใช้โพลิมอร์ฟีซึมในการพัฒนาโปรแกรม จะช่วยให้สามารถสืบทอดคุณสมบัติของคลาสเดิมที่มีอยู่และทำการแก้ไขเฉพาะโอเปอร์เรชั่นที่จำเป็นทำให้ไม่ต้องแก้ไขคลาสเดิมซึ่งมีอยู่แล้ว ซึ่งทำให้ไม่เกิดผลกระทบกับตัวโปรแกรม

2.3 ความแตกต่างระหว่างและวัตถุ

ในการออกแบบโปรแกรมเชิงวัตถุ ปัญหาที่สำคัญที่สุดก็คือการจำลอง(Model) วัตถุในโลกของความเป็นจริง มาไว้ในเครื่องคอมพิวเตอร์ได้อย่างไร วัตถุในโลกของความเป็นจริงนั้นมีขอบเขตที่กว้างเกินกว่าที่จะนำมาใส่ในเครื่องคอมพิวเตอร์ได้หมด จึงจำเป็นที่จะต้องกำหนดความสัมพันธ์และ ความสัมพันธ์ระหว่าง วัตถุประเภทต่างๆในเครื่องคอมพิวเตอร์ด้วย และจากการที่เราไม่สามารถนำเอา วัตถุจากในโลกของความเป็นจริงมาใส่ไว้ในคอมพิวเตอร์ได้ แต่เราสามารถใส่แนวคิดสำหรับ วัตถุต่างๆในคอมพิวเตอร์

แนวคิดหมายถึง ความคิดรวบยอดที่มีให้กับ วัตถุนั้นๆ ภายใต้กรอบที่ได้กำหนดไว้ ตัวอย่างเช่น ถ้าต้องการให้ แนวคิดเกี่ยวกับรถยนต์ นั่นคือ รถทุกคันมีตัวถัง มีล้อ และเครื่องยนต์เหมือนกันทุกคัน เป็นต้น

การให้ แนวคิดกับวัตถุเป็นที่มาของการจัดกลุ่มของ วัตถุสิ่งที่ได้จากกระบวนการนี้เรียกว่า Abstract Objects หรือเรียกอีกอย่างว่าคลาสตัวอย่างเช่น รถยนต์ฮอนด้า รถยนต์โตโยต้า และรถยนต์มาสด้า ต่างก็มี 4 ล้อ มีเครื่องยนต์ และใช้น้ำมันเป็นเชื้อเพลิงเหมือนๆกัน ซึ่งเราสามารถจัดให้รถทั้งสามนี้เป็น วัตถุในคลาส “รถยนต์”

คลาสถือเป็นนามธรรม (Abstract) เราไม่สามารถทำให้ คลาส ดำเนินกิจกรรมใดๆ ได้เลย ตัวอย่างเช่น ถ้าเราพูดว่า รถวิ่งไปบนถนน ในโลกของ Object Orientation นั้นถือว่า รถ จะหมายถึงรถที่มีอยู่ทั่วไป และ ถนนก็จะหมายถึงถนนสายใดๆ ก็ได้ในโลก แต่ถ้าเราพูดว่า รถยนต์ของนาย ก วิ่งไปบนถนนมิตรภาพ นั่นหมายถึงรถยนต์ของนาย ก ซึ่งมีอยู่จริงในโลก และเป็น วัตถุ ของ คลาส “รถยนต์” วิ่งไปบนถนนมิตรภาพ ซึ่งก็เป็น วัตถุของ คลาส “ถนน” เมื่อเราได้ คลาส ต่างๆตาม Domain ที่เราสนใจแล้ว เมื่อเราต้องการให้เกิดกิจกรรมต่างๆในระบบคอมพิวเตอร์ เราต้องทำให้เกิดวัตถุของคลาสขึ้นเพื่อให้มันทำหน้าที่ของตัวมันเองได้

2.4 ประโยชน์ที่สำคัญที่ของการแบ่ง คลาสใน การเขียนโปรแกรมเชิงวัตถุ

- การแบ่งออกเป็น คลาส ทำให้โปรแกรมที่สร้างขึ้นมานั้นมีความง่ายในการแก้ไข เนื่องจากแต่ละ คลาส มีความอิสระต่อกัน ดังนั้นการแก้ไขสิ่งใดๆ ใน Class หนึ่งจะไม่มีผลต่อ คลาส อื่นๆที่เหลือ
- เมื่อแต่ละ คลาส ไม่มีผลอย่างใดอย่างหนึ่งกับ คลาส อื่น เมื่อมีการแก้ไข เราจึงสามารถนำ คลาส ต่างๆมาใช้กับโปรแกรมอื่นๆได้ ซึ่งแนวความคิดนี้ถูกเรียกว่า "Reusability" หรือว่าการนำกลับไปใช้ใหม่นั่นเอง

2.5 วงรอบการพัฒนาระบบงานคอมพิวเตอร์(System Development Life Cycle SDLC)

ในการพัฒนาระบบงานในคอมพิวเตอร์นั้น สามารถแบ่งออกเป็นขั้นตอนใหญ่ ๆ ได้ดังนี้

1. Analysis หมายถึง การวิเคราะห์ปัญหา หรือ Problem Domain (หรือบางทีอาจจะ ใช้คำว่า Requirement Analysis) ซึ่งข้อมูลเกี่ยวกับปัญหาเหล่านี้เราอาจได้มาจากการทำ Client Interview
2. Design หมายถึง การออกแบบเพื่อหาแนวทางหรือแบบแผน เพื่อการแก้ไขปัญหาโดยคอมพิวเตอร์ แต่อย่างไรก็ตามสิ่งที่เกิดขึ้นจากขั้นตอนนี้ก็ยัง ไม่ได้เกิดขึ้นจริงหรือมีตัวตนขึ้นจริงในคอมพิวเตอร์
3. Implementation หมายถึง การทำให้หนทางการแก้ปัญหาที่ได้จากขั้นตอนDesign เกิดขึ้นจริงหรือมีตัวตนขึ้นจริงในคอมพิวเตอร์ และที่สำคัญจะต้องสามารถใช้งานจริงได้ด้วย
4. Testing หมายถึง การทดสอบว่าหนทางการแก้ปัญหาที่สร้างขึ้น สามารถใช้ได้จริงและมีประสิทธิภาพ

2.6 UML (Unified Modeling Language)

เป็นภาษาเพื่อการวิเคราะห์และการออกแบบ ซึ่งจะเป็นภาษาการโมเดลในรูปแบบกราฟฟิก ซึ่งใช้ในการประกอบกันเป็น ไดอะแกรมโดยมีกฎในการประกอบกันเป็นอิลิเมนต์ต่าง ๆ ไดอะแกรมจะแสดงถึงมุมมองต่าง ๆ (Multi view) ของระบบ ซึ่งจะรวมเรียกว่า โมเดล โมเดลUML จะบ่งบอกบอกถึงรายละเอียดของระบบแต่ไม่บอกถึงวิธีในการพัฒนาระบบ

UML เกิดจากความร่วมมือของ Grady Booch, James Rumbaugh ซึ่งเป็นผู้เชี่ยวชาญระดับแนวหน้าของโลกด้าน Object Orientation โดยการเข้าร่วมในงาน Relational Software Corporation ในปี 1994 ปัจจุบัน UML เวอร์ชันล่าสุดคือ 1.3 ซึ่งโดยส่วนประกอบของภาษาจะประกอบด้วย

Diagram หลาย ๆ ชนิดด้วยกัน ซึ่งไม่จำเป็นต้องใช้ครบทุกDiagram ในการออกแบบระบบแต่ละระบบขึ้นอยู่กับความเหมาะสมซึ่งในเอกสารฉบับนี้จะกล่าวถึงโดยสรุปแต่จะเน้นเฉพาะ Diagram ที่ถูกใช้ในการออกแบบสำหรับการสร้างระบบนี้เท่านั้น โดย Diagramใน UML สามารถแบ่งเป็นประเภทได้ 2 ประเภทใหญ่ ๆ คือ 1.Static Diagram 2.Dynamic Diagram

- Static Diagram

  คือ Diagram ที่แสดงภาพในเชิงสถิตย์ (Static) ของProblem Domain นั่นคือการแสดงการมีอยู่ของClass ต่าง ๆ และ ความสัมพันธ์ ของ Class เหล่านั้น ในระบบโดยไม่แสดงถึงกิจกรรมที่เกิดขึ้นแต่อย่างใด ซึ่ง Static Diagram ที่ใช้ในการออกแบบระบบนี้ ได้แก่ Use Case Diagram และClass Diagram

- Dynamic Diagram

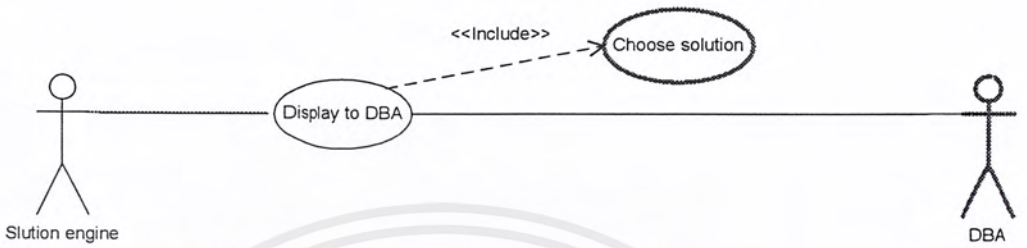
  คือ Diagram ที่แสดงภาพในเชิงกิจกรรม (Dynamic) ของProblem Domain นั่นคือการแสดงถึงสิ่งที่เกิดขึ้นจากกิจกรรมของ Class ต่าง ๆ ที่มีใน Problem Domain (ซึ่งแสดงโดย Class Diagram) จนทำให้เกิดเป็นกิจกรรมของ Problem domain ได้แก่ Activity Diagram

โดยใน Diagram 2 ประเภทใหญ่ ๆ นี้สามารถแบ่งเป็น Diagram ต่าง ๆ ได้ดังนี้

2.6.1 Use Case Diagram

โดยทั่วไปในการออกแบบระบบ เราต้องเริ่มต้นโดยการกำหนดความต้องการของระบบ ซึ่ง UML จะใช้ Use Case ในการกำหนดการใช้งานระบบ แนวความคิดของ Use Case คือ กลุ่มของสถานการณ์เกี่ยวกับการใช้งานระบบ (collection of scenarios about system use) แต่ละ Use Case จะแสดงโดย ลำดับของเหตุการณ์ แต่ละลำดับของเหตุการณ์จะเริ่มต้นด้วย บุคคลหรือระบบ โดยใน Use Case เรียกว่า Actor ผลที่ได้จากลำดับ จะเป็นสิ่งที่ใช้โดย Actor ที่เริ่มเหตุการณ์ หรือ Actor อื่น โดยUse Case จะสร้างจากมุมมองของผู้ใช้ระบบ โดยจะมีลักษณะเหมือนกับสิ่งที่เกิดขึ้นจริง จะต่าง

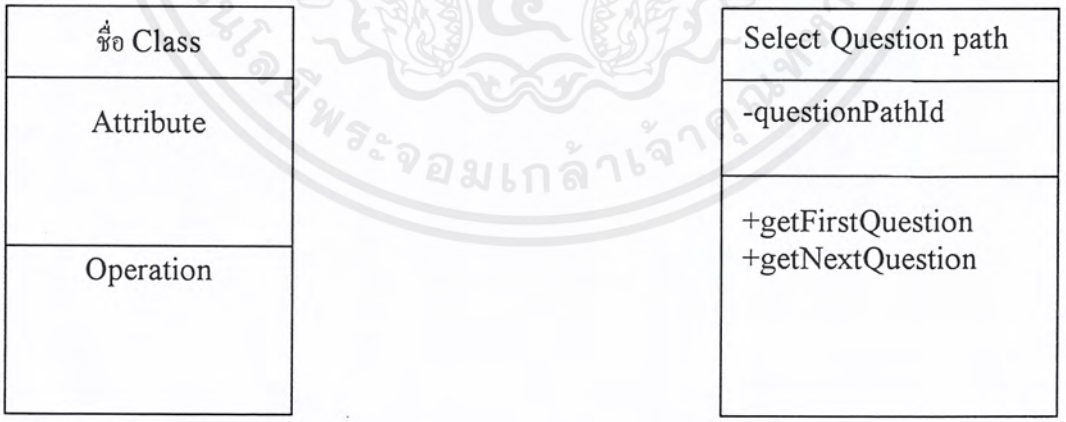
กับการออกแบบระบบแบบอื่น เช่น dataflow ที่ผู้ใช้ต้องหา Input ของระบบในเชิง Computer หรือข้อมูลที่ใช้ในระบบ ซึ่งบางครั้งผู้ใช้อาจจะไม่สามารถระบุได้ชัดเจนครบถ้วนจากรูป Actor คือ Solution engine กับDBA และ Display DBA คือเหตุการณ์ที่เกิดขึ้น



รูปที่ 2.5 Use Case Diagram

2.6.2 Class Diagram

เป็นอีก Diagram หนึ่งที่มีลักษณะเป็นแบบ Static โดย Class ใน UML ออบเจคต่าง ๆ ในระบบ จะประกอบด้วย แอททริบิวท์ และโอเปอร์เรชัน ออบเจคที่ประกอบด้วย แอททริบิวหรือโอเปอร์เรชันเดียวกันจะเรียกรวมกันว่า คลาส หรืออาจจะบอกได้ว่า ออบเจคก็คือ instance ของคลาส คลาสไดอะแกรมจะแสดงเป็นรูปสี่เหลี่ยมแบ่งเป็น3ส่วน คือ ชื่อ คลาส, แอททริบิว และ โอเปอร์เรชัน ดังรูป



รูปที่ 2.6 Class Diagram

ในทางปฏิบัติ แอททริบิวและโอเปอร์เรชันของคลาสจะกำหนดเฉพาะในความต้องการของระบบเท่านั้นเช่นในตัวอย่างนี้เป็น คลาสชื่อ Select Question path และมีแอททริบิวชื่อว่า

questionPathId ซึ่งเป็นแบบ private และมีโอเปอร์เรชันชื่อ getFirstQuestion และ getNextQuestion เป็นpublic แต่ถ้าหากเป็นคลาสชื่อ Select Question path ในระบบอื่นอาจจะมีแอททริบิวและโอเปอร์เรชันเป็นอย่างอื่นก็ได้ ขึ้นอยู่กับความต้องการของระบบนั้น ๆ

เครื่องหมาย -, # และ + เหล่านี้เรียกว่า Visibility ซึ่งจะถูกกำหนดให้กับ Attribute หรือ Operation เพื่อบอกถึงความสามารถในการเข้าถึงโดย Class อื่น โดยแบ่งออกได้ 3 ประเภท คือ Private, Protect, Public ซึ่งแทนด้วยสัญลักษ์ -, # และ + ตามลำดับ

- Private หมายความว่า สามารถเข้าถึงแต่เฉพาะ Class ของมันเองเท่านั้น
- Protect หมายความว่า สามารถเข้าถึงได้จาก Class ที่เป็น Subclass ของมันเองเท่านั้น
- Public หมายความว่า สามารถเข้าถึงจาก Class อื่นได้

2.6.3 Object Diagram

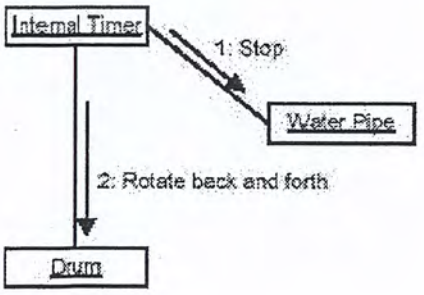
ออบเจค คือ Instance ของคลาสดังนั้นการกำหนดค่าให้กับออบเจคก็คือการกำหนดค่าให้กับ แอททริบิวของคลาส ออบเจคไดอะแกรมของUML จะแสดงเป็นรูปสี่เหลี่ยม โดยชื่อจะประกอบด้วย ชื่อออบเจคและชื่อคลาส คั่นระหว่างชื่อด้วยเครื่องหมาย : และขีดเส้นใต้ชื่อด้วย

2.6.4 Sequence Diagram

คลาสไดอะแกรม และออบเจคไดอะแกรม จะแสดงข้อมูลของ แต่ละคลาสหรือออบเจค ออบเจคต่าง ๆ ในระบบจะมีการติดต่อหรือผลกระทบระหว่างกัน(Interaction) ซึ่ง sequence diagram จะแสดง interaction ที่เปลี่ยนแปลงตามเวลา

2.6.5 Collaboration

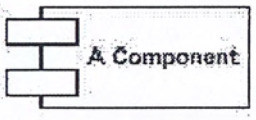
สิ่งต่าง ๆ ในระบบจะทำงานร่วมกันเพื่อให้เกิดผลตามจุดประสงค์ของระบบ รูปแบบการทำงานลักษณะนี้จะแสดงโดย Collaboration diagram



รูปที่ 2.7 Collaboration

2.6.6 Component Diagram

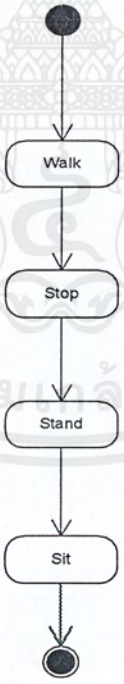
เป็นไดอะแกรมของการพัฒนา ซอฟแวร์ยุคใหม่ ที่ใช้หลักการของคอมโพเนนท์ เช่น คอนโทรลต่าง ๆ ใน Visual Basic หรือ คอมโพเนนท์ใน Delphi มีรูปแบบคือ



รูปที่ 2.8 Component Diagram

2.6.7 State Diagram

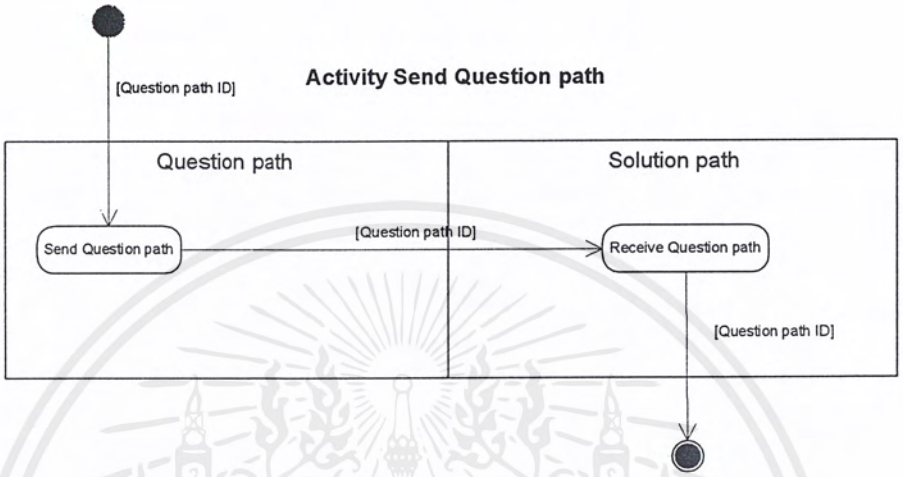
ในเวลาใด ๆ ออบเจคจะอยู่ในสถานะใดสถานะหนึ่ง เช่น คน ก็มีสถานะคือ เดิน หยุด ยืน นั่ง เป็นต้น UML state diagram ของคน ก็จะเป็นได้ดังรูป



รูปที่ 2.9 State Diagram

2.6.8 Activity Diagram

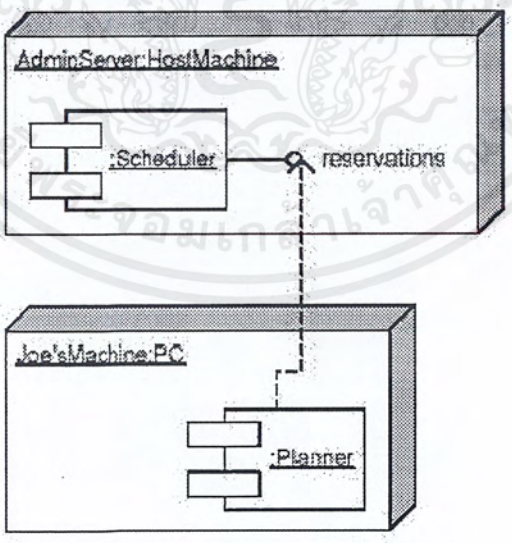
กิจกรรมที่เกิดขึ้นตาม Use Case หรือเกิดจากพฤติกรรมของออบเจคตามปกติเป็นลำดับของกิจกรรมดังรูปในตัวอย่าง



รูปที่ 2.10 Activity Diagram

2.6.9 Deployment Diagram

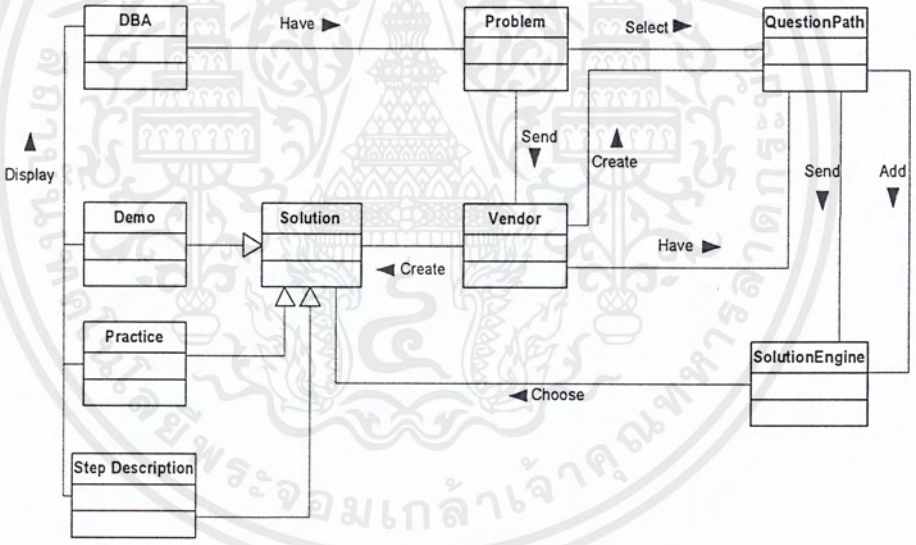
Deployment Diagram จะแสดงสถาปัตยกรรมของระบบคอมพิวเตอร์ และดีไวซ์ต่าง ๆ การเชื่อมต่อ และแสดงซอฟแวร์ที่ติดตั้งในแต่ละระบบ ซึ่งมีประโยชน์ในแง่ของการติดตั้ง



รูปที่ 2.11 Deployment Diagram

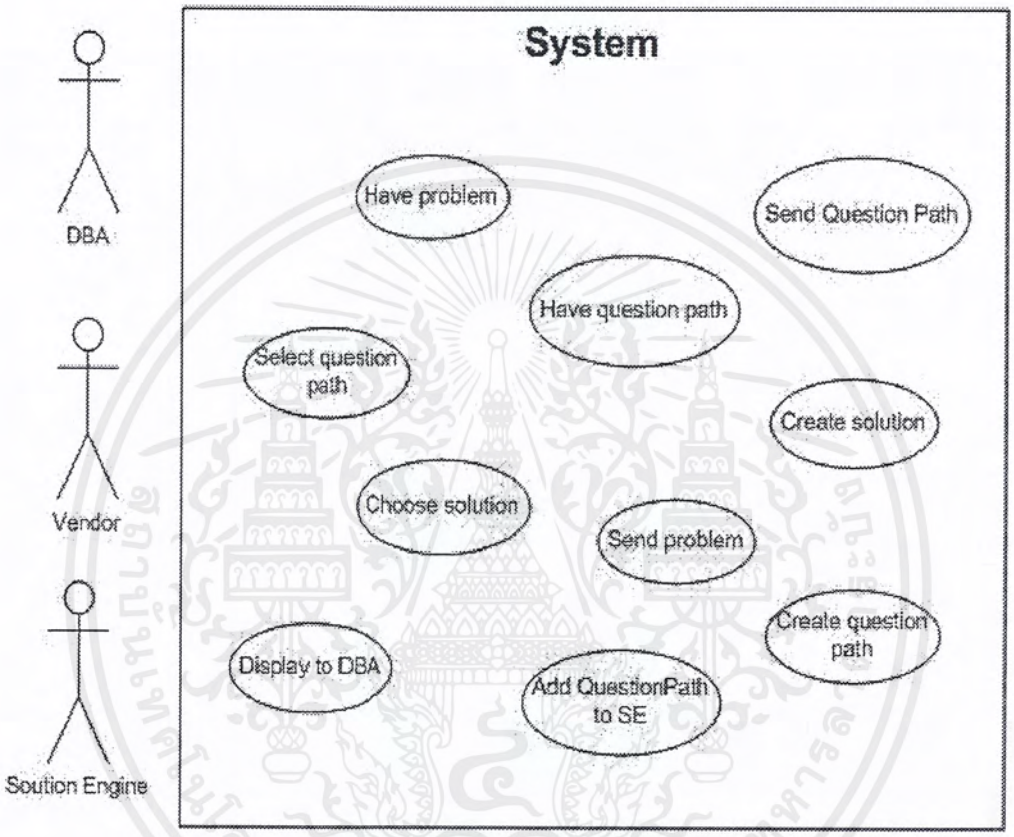
บทที่ 3
การออกแบบระบบผู้ช่วยผู้ดูแลระบบฐานข้อมูลออราเคิล

รูปข้างล่างเป็นภาพของ Client Interview ซึ่งได้มาจากการสัมภาษณ์ความต้องการของผู้ใช้งานระบบฐานข้อมูลออราเคิล ซึ่งสามารถนำมาออกแบบเป็นระบบเพื่อที่จะช่วยแก้ปัญหา โดยมีการทำงานเป็นขั้นตอนดังต่อไปนี้ เมื่อมีปัญหาเกิดขึ้นกับผู้ดูแลระบบ ผู้ดูแลระบบก็จะทำการเลือกหัวข้อของปัญหาที่เกี่ยวข้องกับปัญหาที่เกิดขึ้น หลังจากนั้นตัวระบบนี้ก็จะทำการถามคำถามให้ผู้ดูแลระบบตอบคำถาม แล้วระบบนี้ก็จะนำเอาคำตอบต่าง ๆ ที่ได้ไปทำการวิเคราะห์ว่าควรจะเป็นปัญหาที่มีสาเหตุจากอะไร หลังจากนั้นก็แสดงวิธีการแก้ไขปัญหาที่เกิดขึ้นให้กับผู้ดูแลระบบ



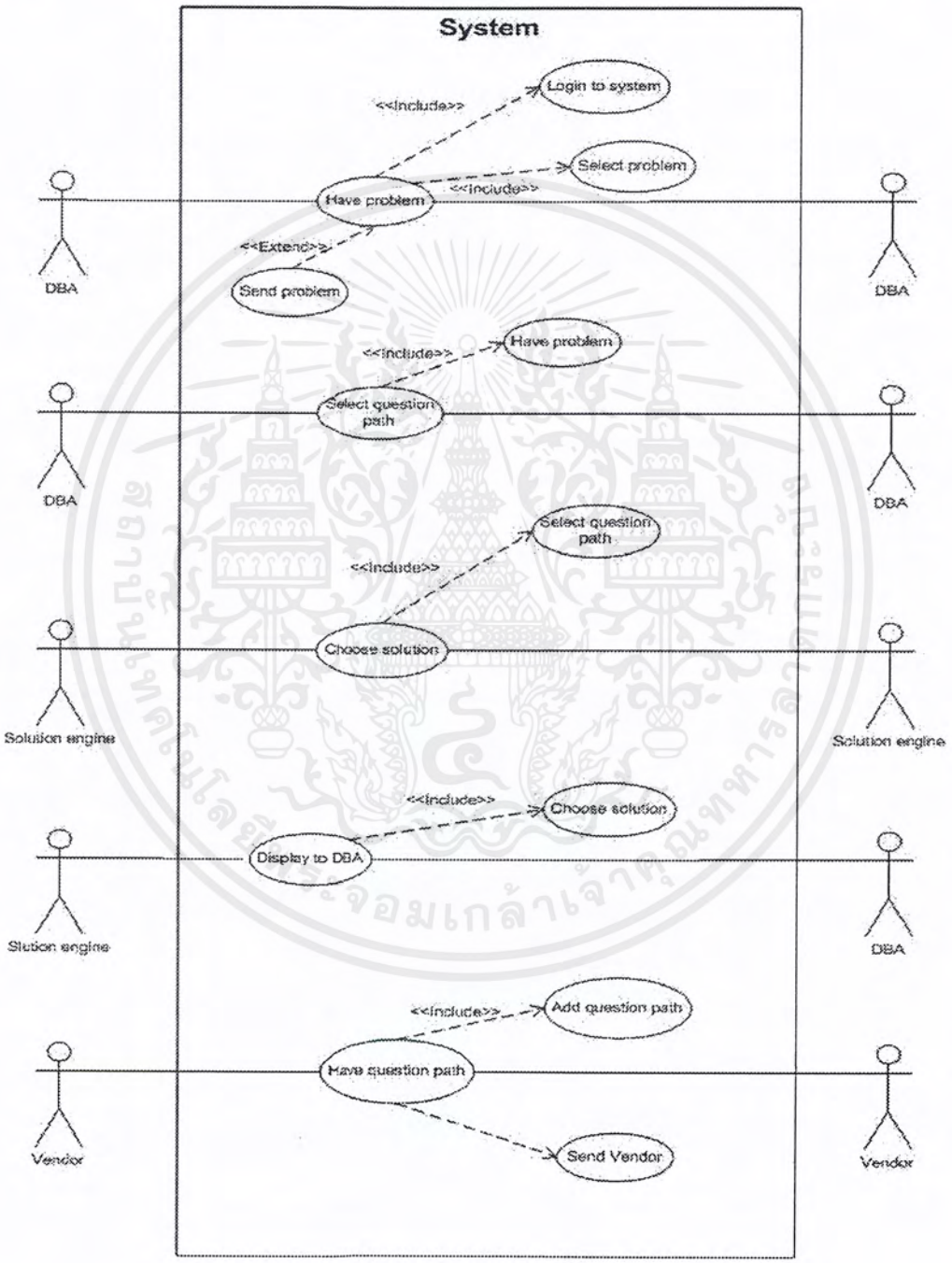
รูปที่ 3.1 Client Interview

ภาพข้างล่างเป็นภาพของ High Level Use Case ซึ่งใช้เพื่ออธิบายว่า ในระบบมีเหตุการณ์อะไรเกิดขึ้นบ้าง ซึ่งในระบบจะมีเหตุการณ์ที่สำคัญคือ Have Problem, Have Question Path Send Question Path, Select Question Path, Choose Solution, Send Problem, Create Solution, Display To DBA, Add Question Path to SE, Create Question Path

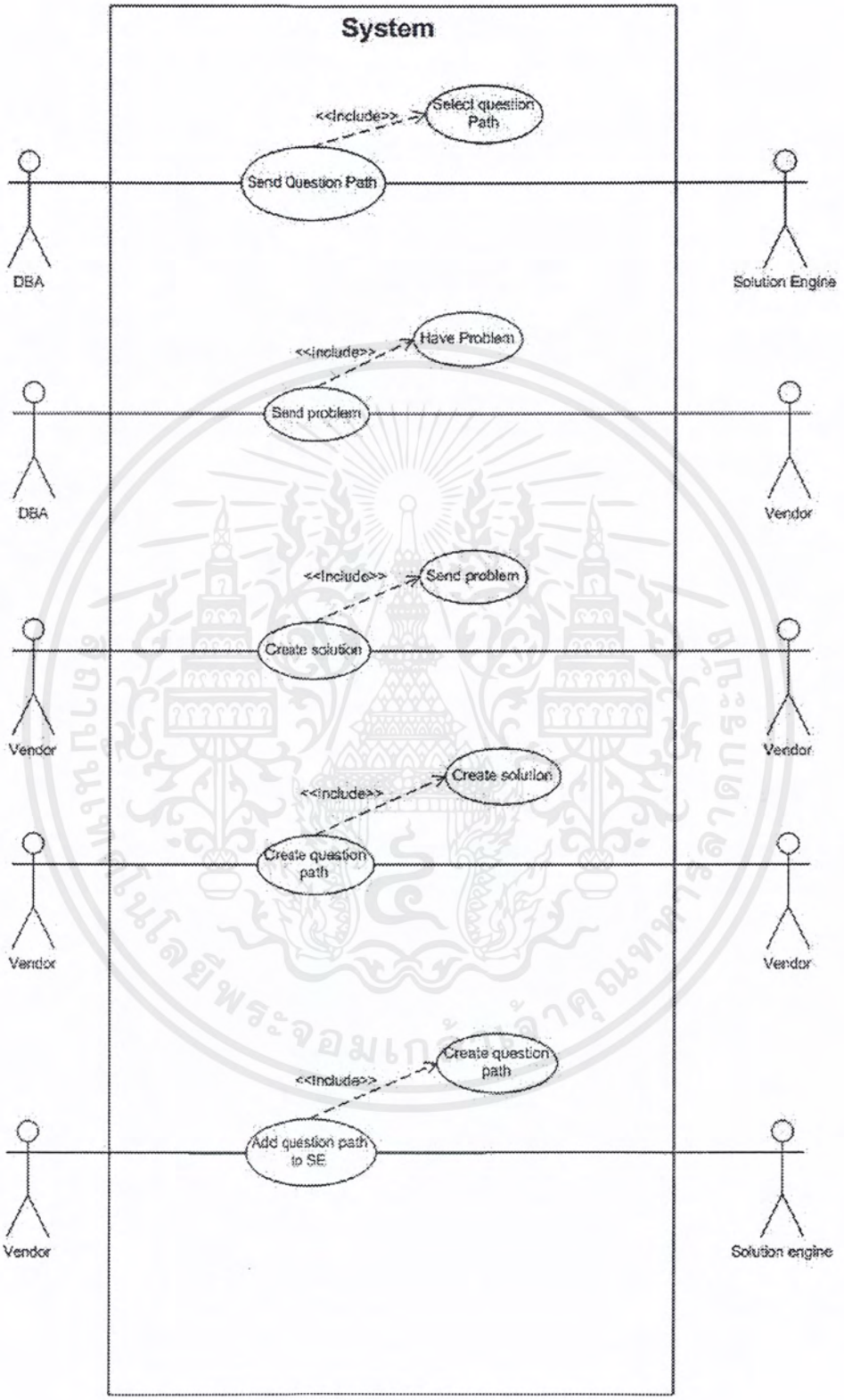


รูปที่3.2 High Level Use Case

ภาพข้างล่างเป็นภาพที่ใช้แสดงเหตุการณ์ต่างๆที่เกิดขึ้นในระบบ ใครเป็นผู้ทำให้เกิดเหตุการณ์(Event) และใครเป็นผู้ได้รับผล ยกตัวอย่างเช่น Use Case Have Problem มี DBA เป็นผู้เริ่มเหตุการณ์ หลังจากที่ DBA เกิดปัญหาขึ้น จะต้องทำการล็อกอินเข้าสู่ระบบก่อน จากนั้นจึงทำการเลือกปัญหาที่เกิดขึ้น ใกรณีที่ปัญหานั้นไม่มีในระบบจะต้องส่งปัญหาไปให้กับบริษัทที่ผลิต

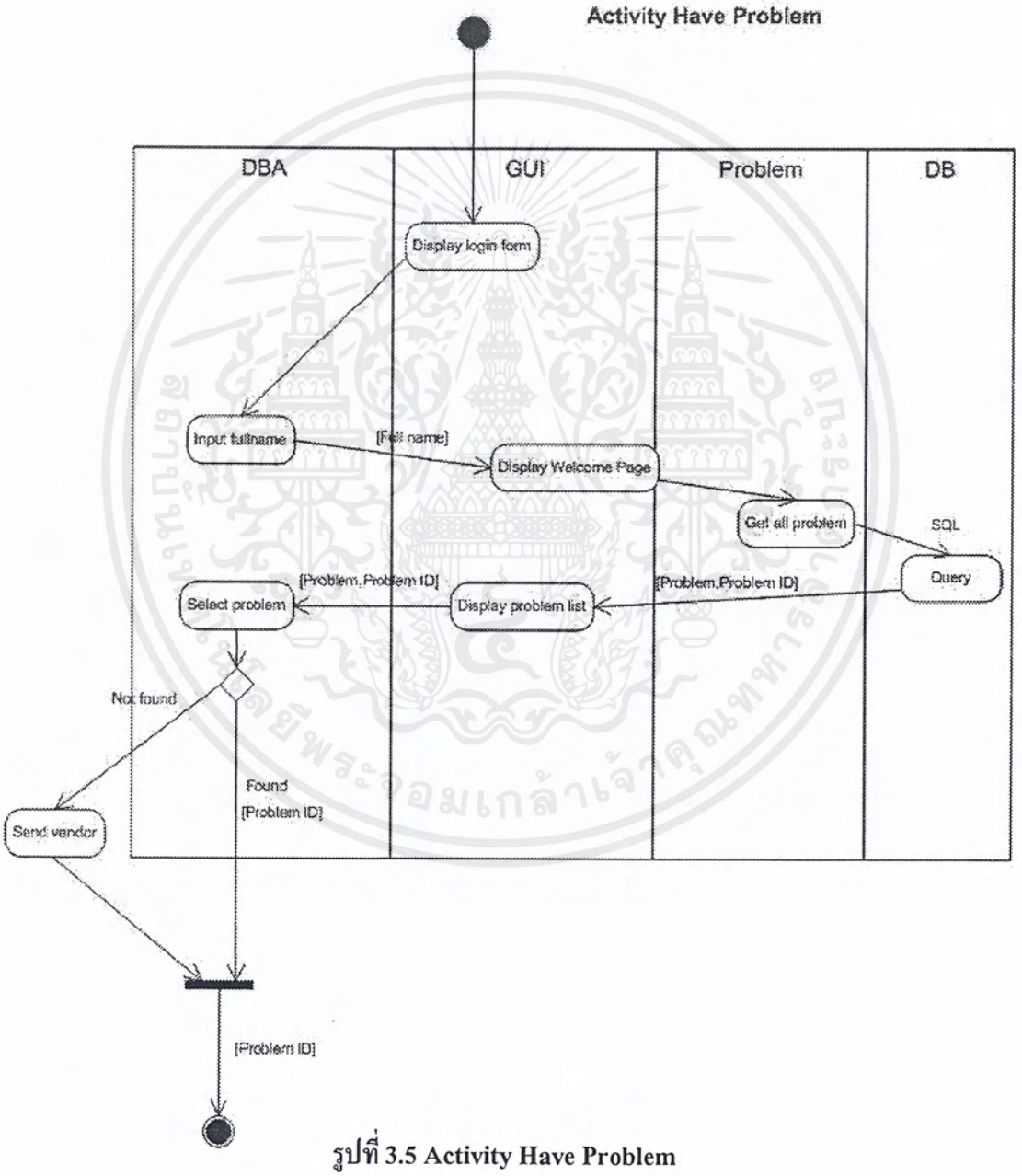


รูปที่ 3.3 Use Case Level 1

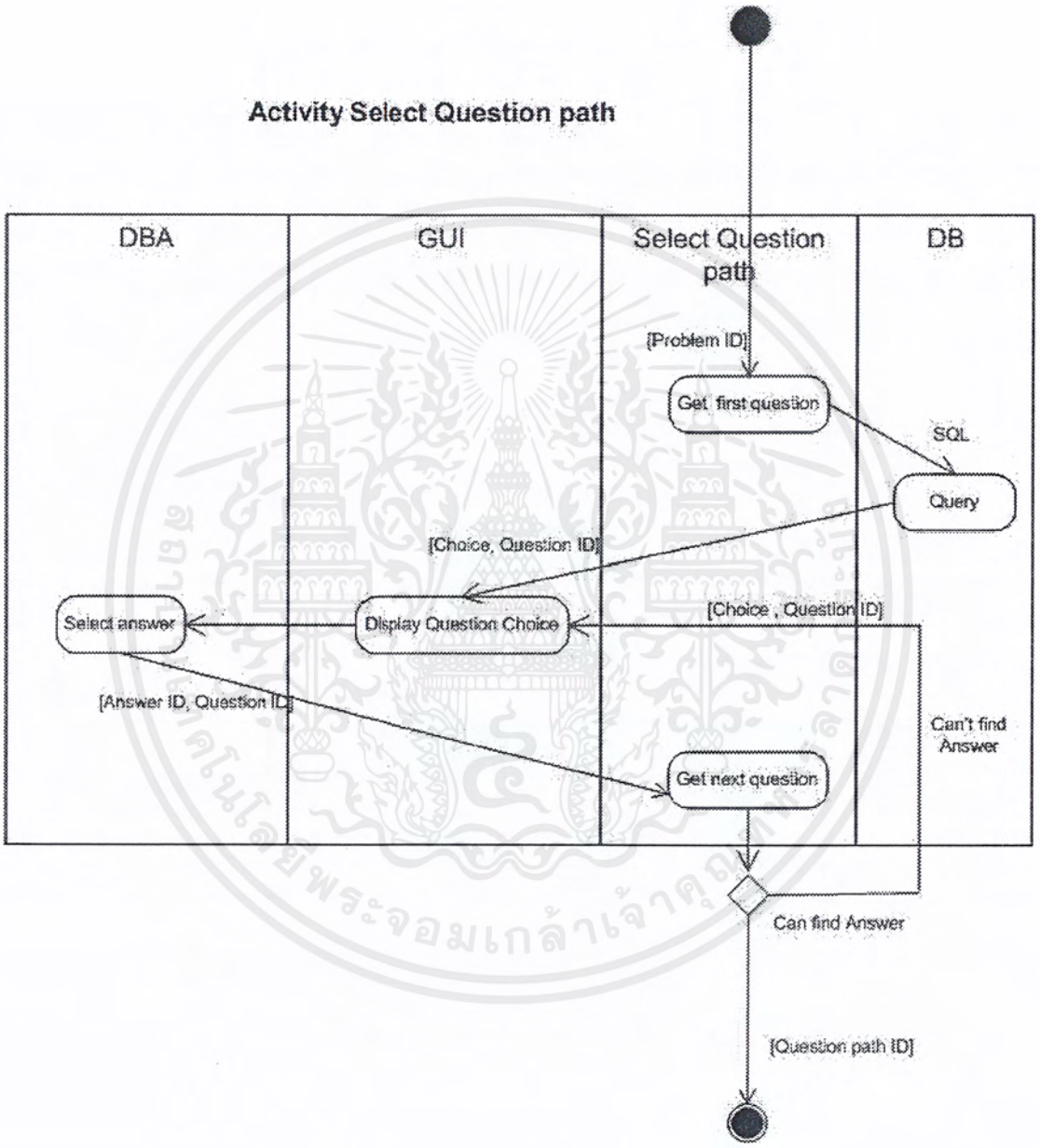


รูปที่ 3.4 Use Case Level 1 (ต่อ)

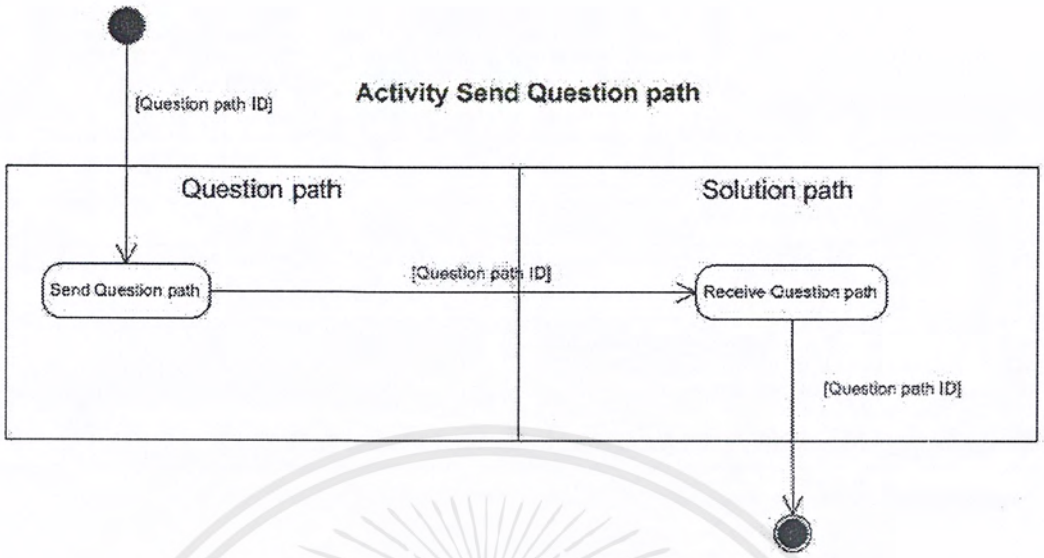
Activity Diagram เป็น Diagram ที่ใช้ในการแสดงขั้นตอนการทำงานของกิจกรรม (Activity)ต่างๆในโปรแกรมและ Classใดเป็นผู้กระทำ ยกตัวอย่างเช่น Activity Have Problem จะเริ่มจากการที่ Class GUI แสดงฟอร์มที่ใช้ในการลอกอิน หลังจากนั้นจะให้ Class DBA ใส่ชื่อ เมื่อเรียบร้อยก็จะแสดงหน้าจอ Welcome Page ขึ้นมา และจะมีการแสดงปัญหาทั้งหมดที่มีโดยเรียกจากฐานข้อมูล และส่งให้กับ Class GUI เพื่อแสดงให้กับผู้ใช้งานได้ทำการเลือกคำตอบเพื่อนำไปใช้ในการวิเคราะห์ต่อไป



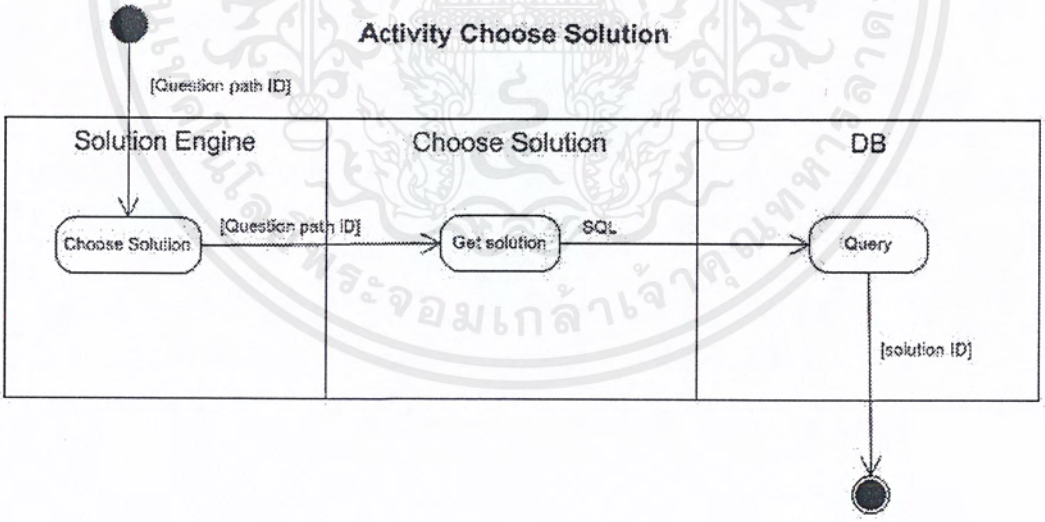
รูปที่ 3.5 Activity Have Problem



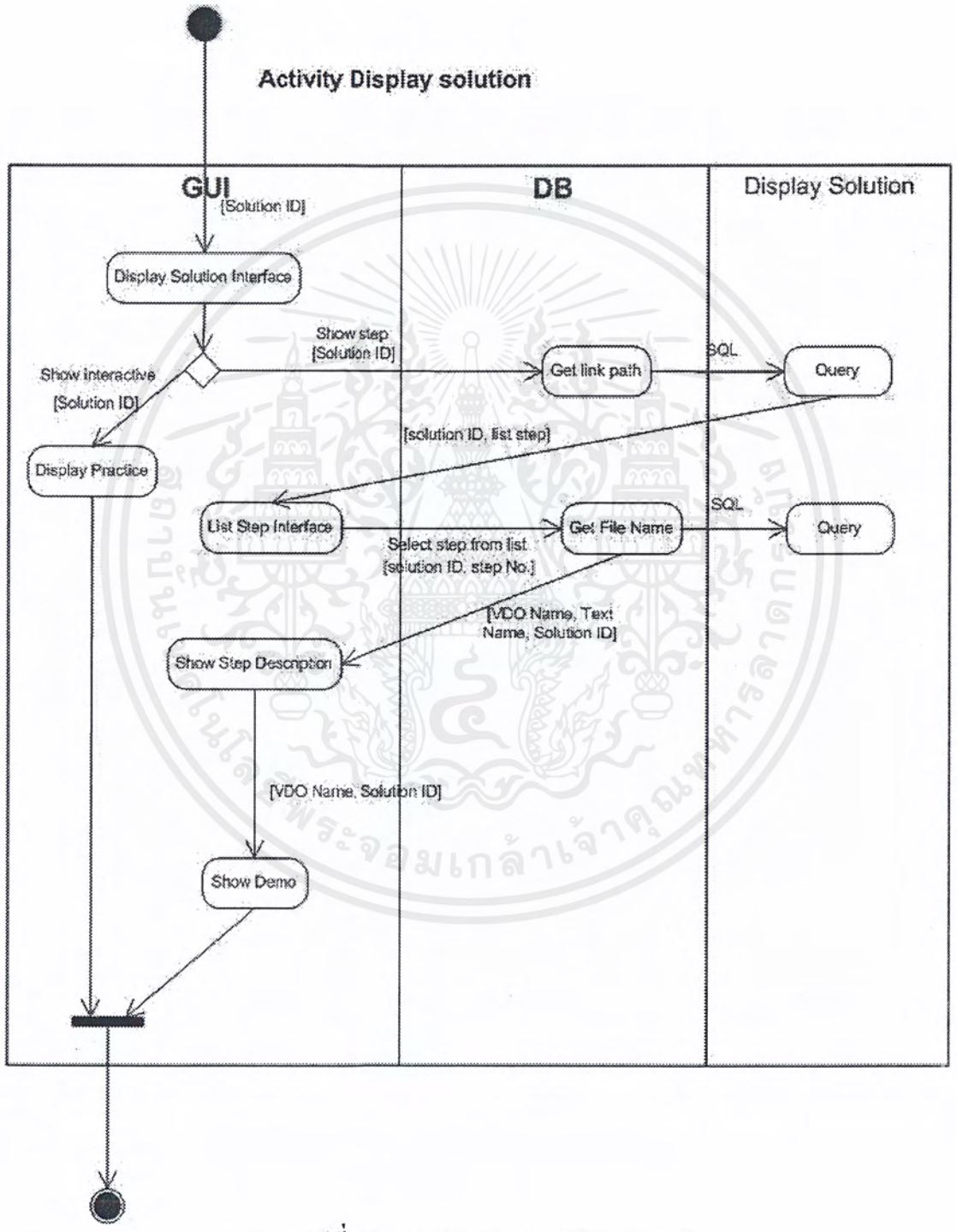
รูปที่ 3.6 Activity Select Question Path



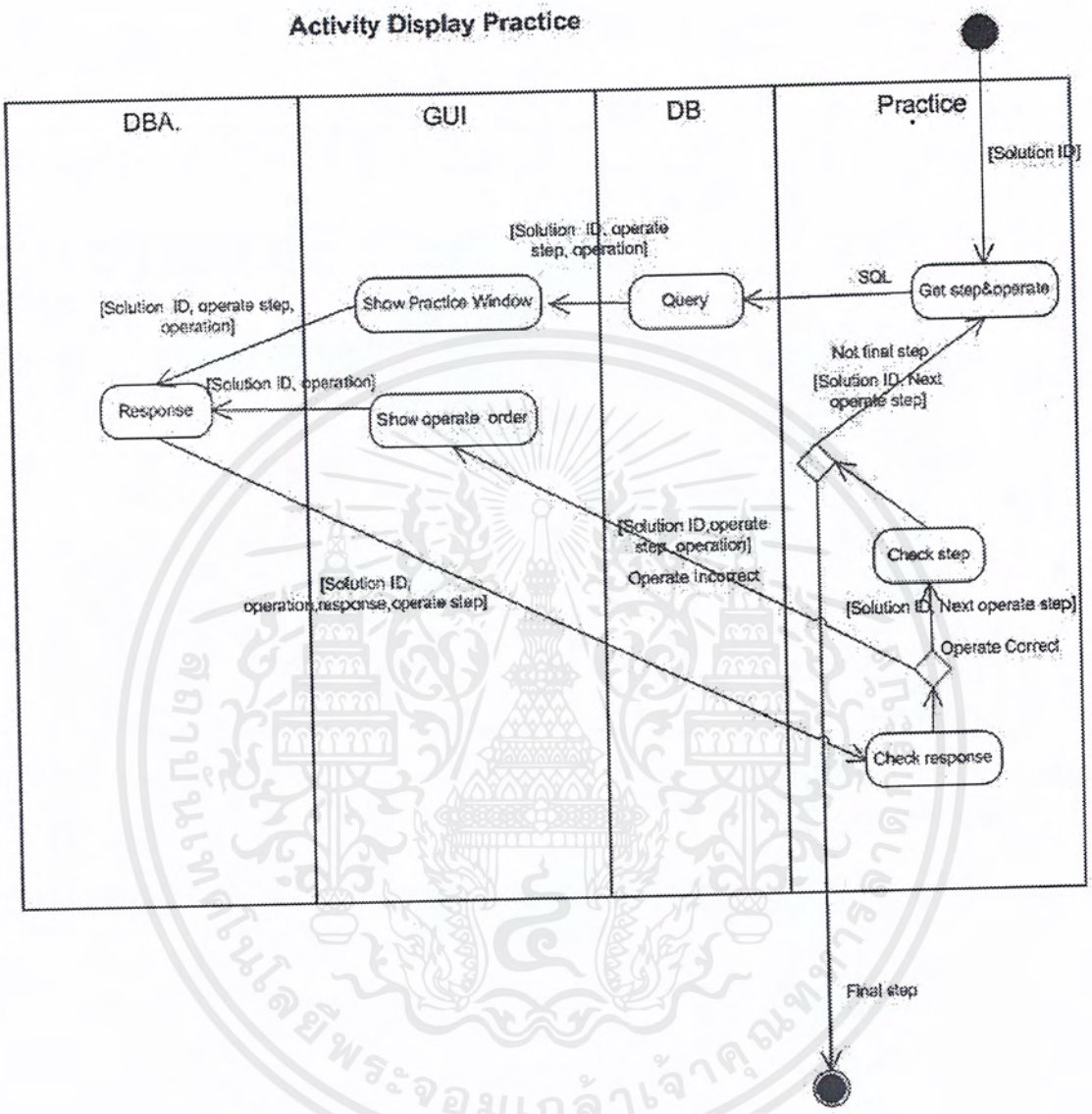
รูปที่ 3.7 Activity Send Question Path



รูปที่ 3.8 Activity Choose Solution

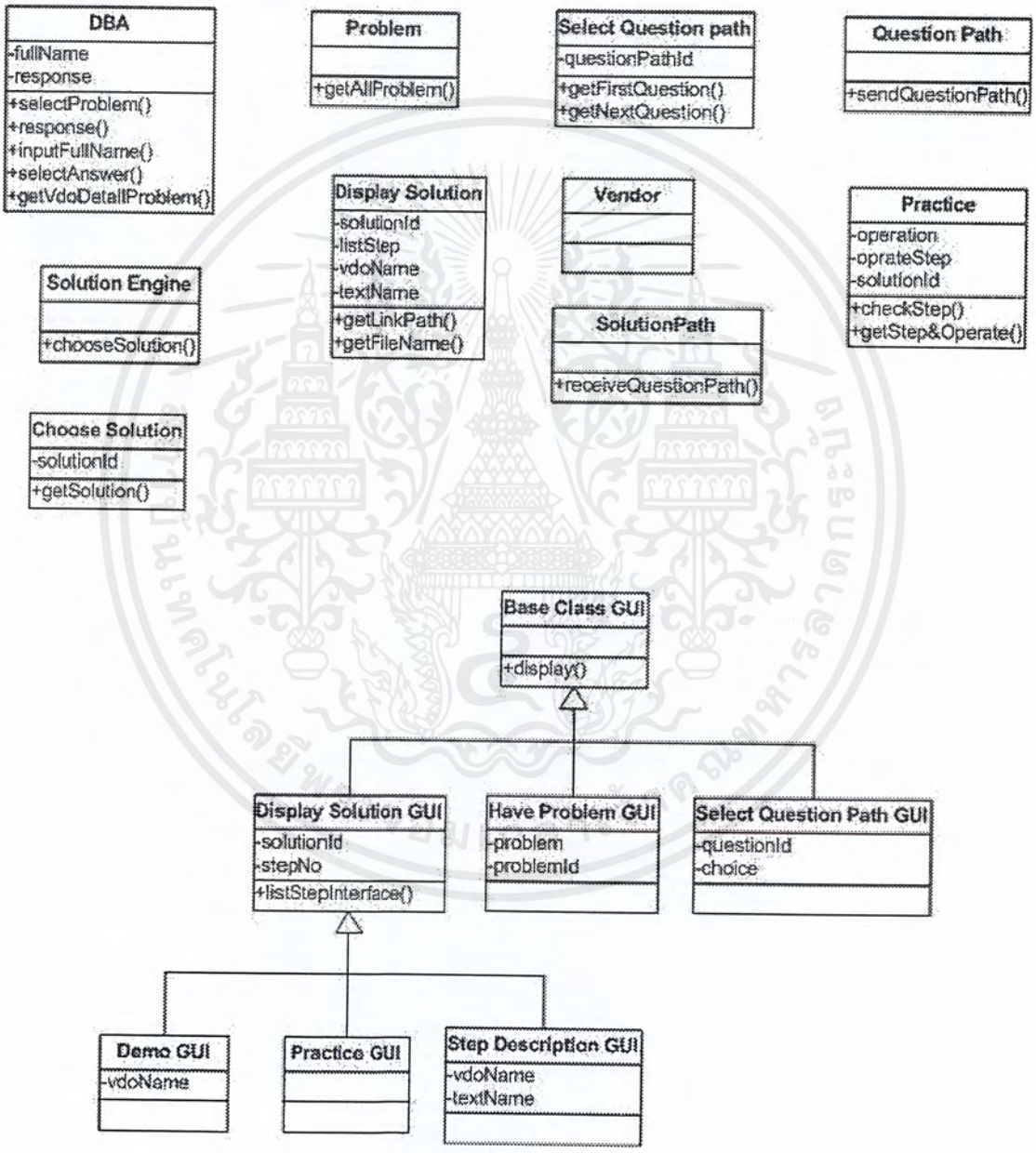


รูปที่ 3.9 Activity Display Solution



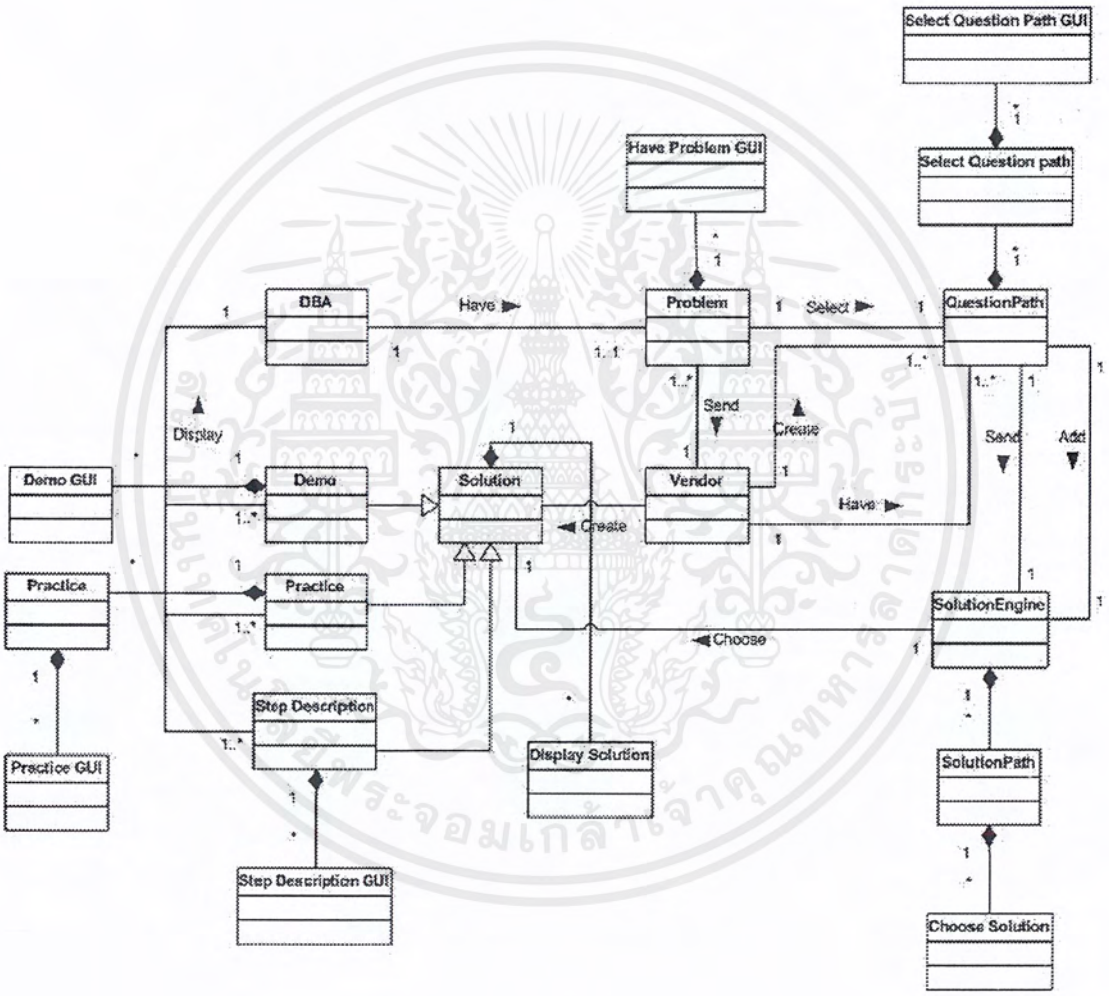
รูปที่ 3.10 Activity Display Practice

รูปด้านล่างนี้ เป็นรูปของ Class Diagram ซึ่งได้มาจากการออกแบบ Activity Diagram ซึ่ง Attribute ในแต่ละ Class ของ Class Diagram ได้มาจาก Function ต่าง ๆ ของภายในแต่ละ Activity



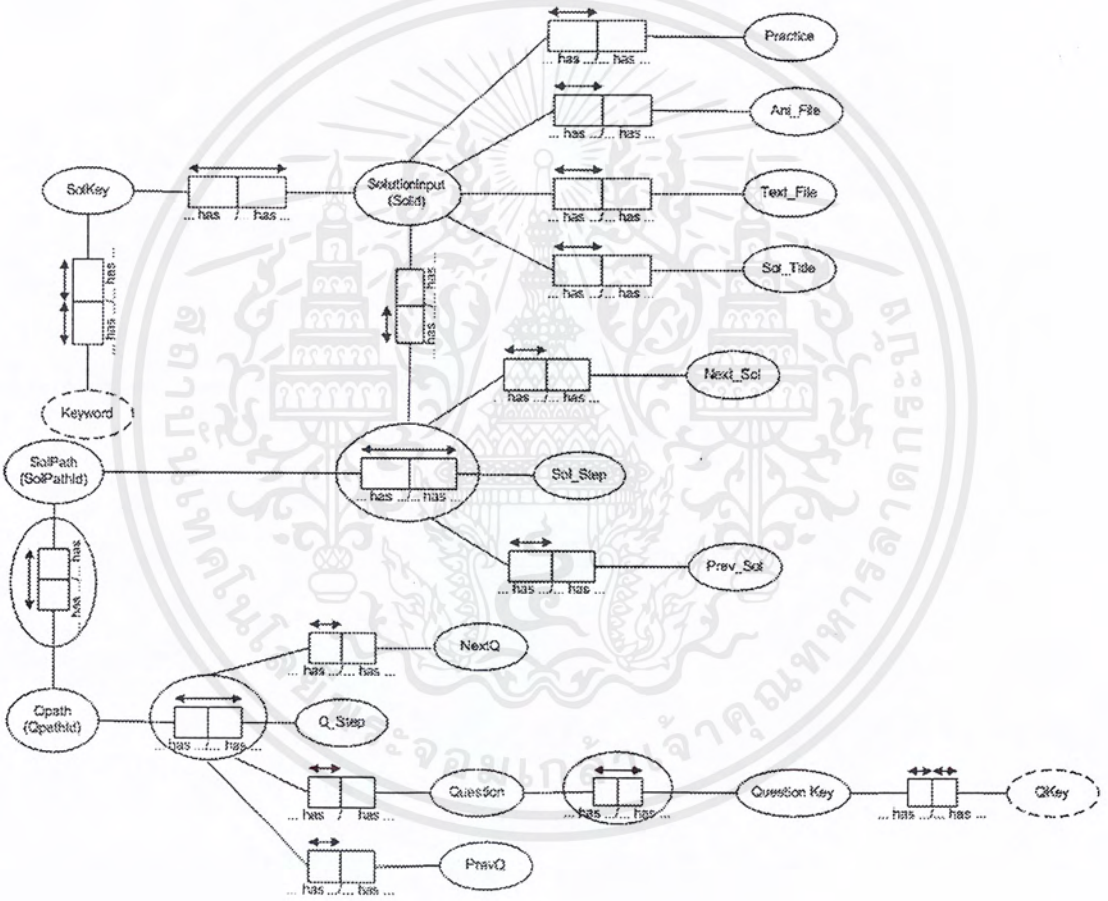
รูปที่ 3.11 Class Diagram

ด้านล่างนี้เป็นรูปของ Class Diagram ที่ใส่ความสัมพันธ์เข้าไป ทำให้เราทราบว่า Class แต่ละClass มีความสัมพันธ์กันอย่างไร



รูปที่ 3.12 Class Diagram (ต่อ)

ด้านล่างเป็นรูป ของ Niam Diagram ได้มาจากAttributeที่จำเป็นต้องเก็บลงบนฐานข้อมูล มาทำการนอมัลไลซ์ ให้มีความซ้ำซ้อนน้อยที่สุดเพื่อนำไปออกแบบตารางต่อไป



รูปที่ 3.13 Niam Diagram

รีเรชั่นนอลดาต้าเบส (Relational Database)

ตารางที่ 3.1 ตาราง SoltionInput ประกอบด้วย แอทริบิวต์

SolId	Sol_title	Ani_File	Text_File	Practice

ตารางที่ 3.2 ตาราง SolPath ประกอบด้วย แอทริบิวต์

SolPathId	Sol_step	SolId	Next_sol	Prev_sol

ตารางที่ 3.3 ตาราง QPath ประกอบด้วย แอทริบิวต์

QPathId	Q_step	Question	Next_Q	Prev_Q

ตารางที่ 3.4 ตาราง SolKey ประกอบด้วย แอทริบิวต์

SolId	Keyword

ตารางที่ 3.5 ตาราง Q_Sol ประกอบด้วย แอทริบิวต์

QPathId	SolPathId

ตารางที่ 3.6 ตาราง QuestioinKey ประกอบด้วย แอทริบิวต์

Question	Q_key

DataDictionary

ตาราง SoltionInput

Field Name	Key	Data Type	Data size	Description	Remark
SolId	Primary Key	Integer		ใช้แยกแต่ละ Solution	
Sol_title		String		ชื่อของSolution	
Ani_File		String		ชื่อของ Path ที่เก็บ ไฟล์ Animation	
Text_File		String		ชื่อของ Path ที่เก็บ ไฟล์ Text	
Practice		String		ชื่อของ Path ที่เก็บ ไฟล์ Practice	

Table SolPath

Field Name	Key	Data type	Data size	Description	Remark
SolPathId	Primary Key	Integer		เป็นเลขของแต่ละ Solution Path	
Sol_step	Primary Key	Integer		บอกว่าSolution นี้ทำงานเป็น Step ที่เท่าไรของ Solution Path	
SolId	Foreign Key	Integer		ใช้แยกแต่ละ Solution	
Next_sol		Integer		บอกว่า Step ต่อไป ทำที่ Solution ไหน	
Prev_sol		Integer		บอกว่า Step ก่อนหน้านั้น ทำที่ Solution ไหนมา	

ตาราง QPath

Field Name	Key	Data type	Data size	Description	Remark
QPathId	Primary Key	Integer		ใช้แยกแต่ละ Question Path	
Q_step	Primary Key	Integer		บอกว่า Question นี้ทำงานเป็น Step ที่เท่าไร	
Next_Q		String		บอกว่า Step ต่อไป ทำที่ Question ไหน	
Prev_Q		String		บอกว่า Step ก่อนหน้านั้นทำที่ Question ไหนมา	
Question		String		เป็นข้อความที่เป็นคำถามในแต่ละคำถาม	

ตาราง SolKey

Field Name	Key	Data type	Data size	Description	Remark
SolId	Primary Key	Integer		ใช้แยกแต่ละ Solution	
Keyword	Primary Key	String		เป็นคำที่ใช้ในการ ค้นหา Solution ที่เกี่ยวข้องกับที่ผู้ใช้ต้องการ	

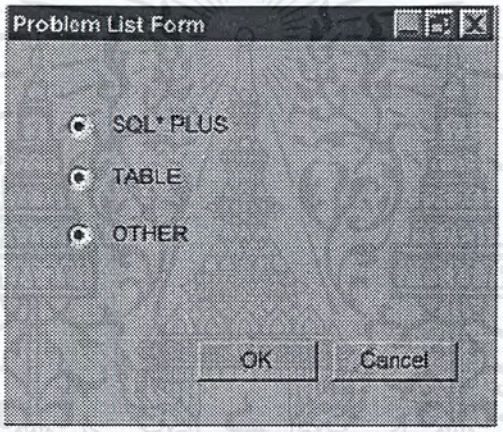
ตาราง Q_Sol

Field Name	Key	Data type	Data size	Description	Remark
QPathId	Primary Key	Integer		ใช้แยกแต่ละ Question Path	
SolPathId	Primary Key	Integer		เป็นเลขของแต่ละ Solution Path	

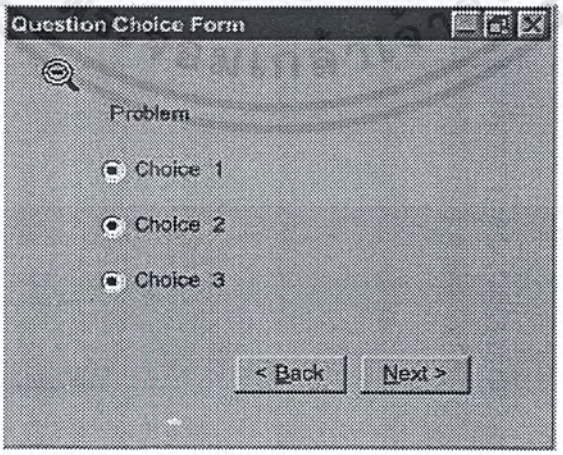
ตาราง QuestionKey

Field Name	Key	Data type	Data size	Description	Remark
Question	Primary Key	String		เป็นข้อความที่เป็นคำถามในแต่ละคำถาม	
Q_key	Primary Key	String		คำที่ใช้ในการ ค้นหาคำถามที่เกี่ยวข้องกับที่ผู้ใช้ต้องการ	

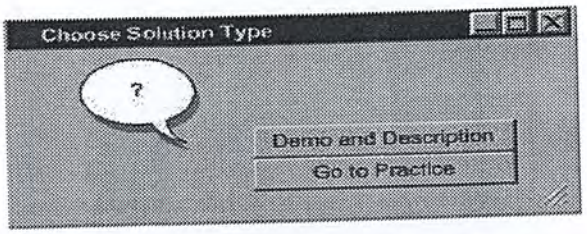
User Interface



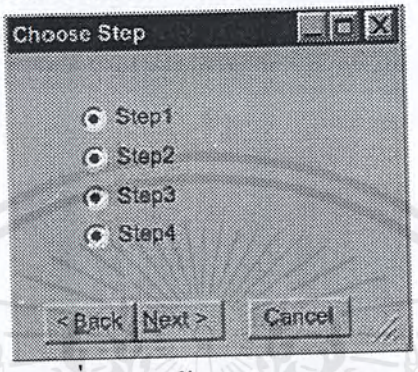
รูปที่ 3.14 หน้าจอ Problem List Form



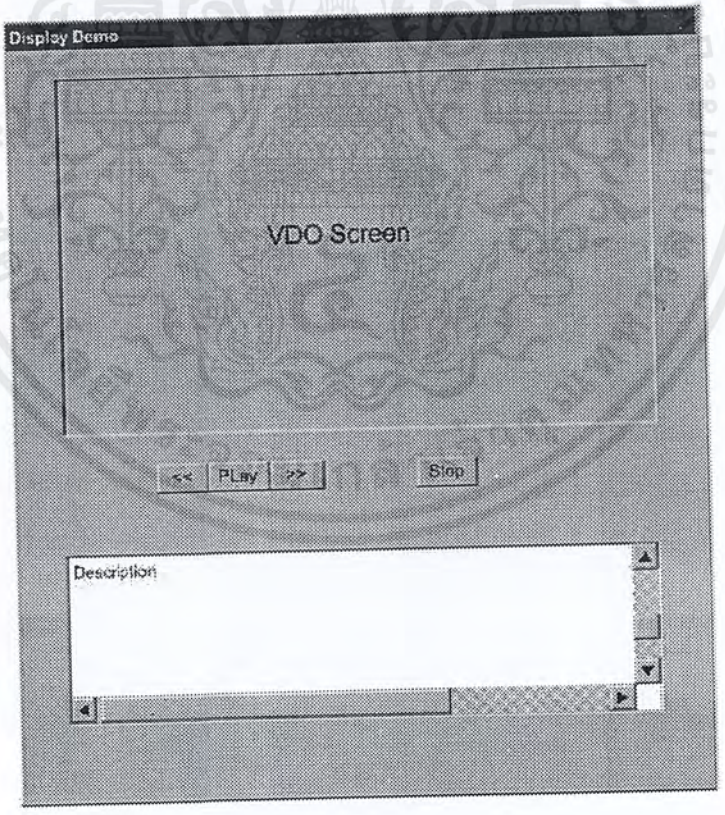
รูปที่ 3.15 หน้าจอ Question Choice Form



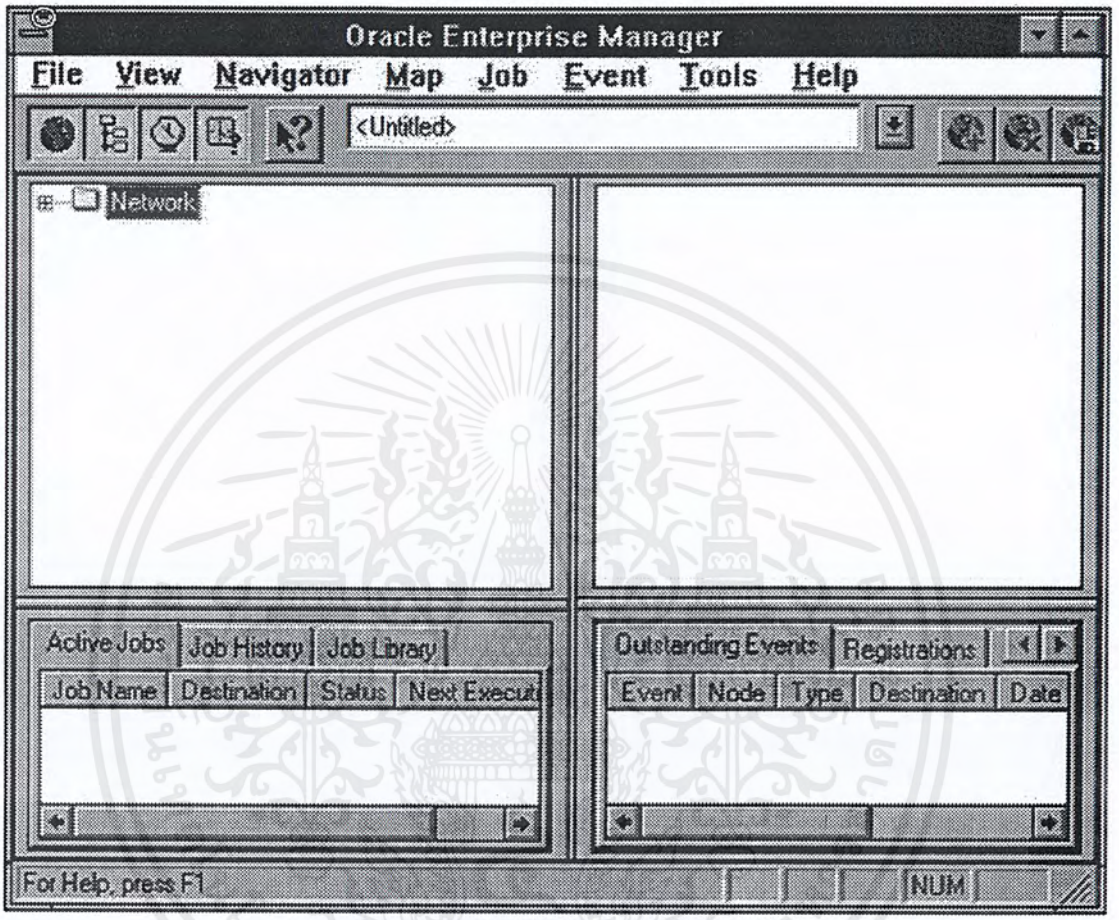
รูปที่ 3.16 หน้าจอ Choose Solution Type



รูปที่ 3.17 หน้าจอ Choose Step



รูปที่ 3.18 หน้าจอ Display Demo



รูปที่ 3.19 หน้าจอ Practice Interface

บทที่4
ระบบผู้ช่วยผู้ดูแลระบบฐานข้อมูลออราเคิล

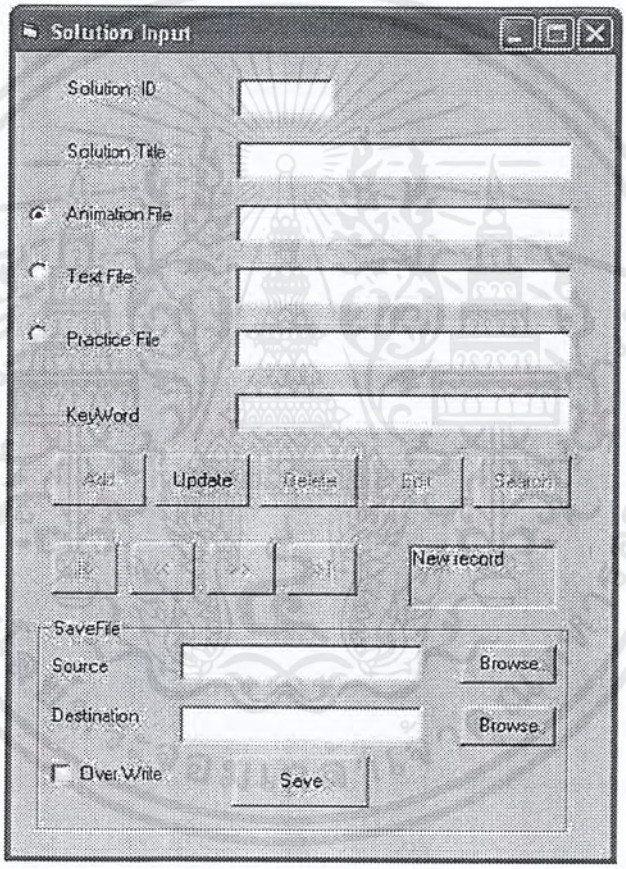
4.1 ส่วนประกอบของโปรแกรม

1. Practice พัฒนาจากโปรแกรม Authorware เป็น ไฟล์นามสกุล EXE มีหน้าที่แสดงวิธีการแก้ไขปัญหาต่างๆที่เกิดจากการใช้งานโปรแกรมออราเคิล
2. วีดีโอ เป็นไฟล์นามสกุล WMV ที่ใช้กับโปรแกรมวินโดว์มีเดียเพลย์เยอร์ วีดีโอ มีหน้าที่แสดงขั้นตอนการแก้ไขปัญหาอย่างคร่าวๆให้กับใช้งาน
3. ข้อความ เป็นไฟล์นามสกุล TXT มีหน้าที่ในการแสดงขั้นตอนต่างๆที่ใช้ในการแก้ไขปัญหา
4. เป็นส่วนที่ใช้ติดต่อกับผู้ใช้ที่พัฒนามาจากโปรแกรม Visual Basic ใช้ในการควบคุมให้ Practice, วีดีโอ, ข้อความทำงานร่วมกันได้ ซึ่งในส่วนติดต่อกับผู้ใช้นี้จะมีการแบ่งการใช้งานเป็น 2 ระดับ คือ
    1) ระดับผู้ใช้งานให้ผู้ใช้สามารถเลือกปัญหาที่เกิดขึ้นจากปัญหาที่มีทั้งหมดที่มีในระบบ จากนั้นก็ให้ผู้ใช้เลือกคำตอบเพื่อวิเคราะห์ว่าปัญหาที่เกิดเกิดจากสาเหตุใด หลังจากที่ทราบสาเหตุแล้วก็จะแสดงวิธีการแก้ไขปัญหาให้กับผู้ใช้
    2) ระดับผู้ดูแลมีฟังก์ชั่นการเพิ่มโซลูชั่นและส่วนของการเพิ่มส่วนของรายละเอียดของปัญหาว่าจะมีเส้นทางของคำถามอย่างไรโดยที่ส่วนนี้ถ้าหากเรามีการเพิ่มข้อมูลมากขึ้นเท่าไรก็จะทำให้สามารถหาคำตอบของคำถามได้ถูกต้องมากขึ้นด้วย โดยจะมีลักษณะเป็น tree ให้ใช้งานได้ง่ายขึ้น

4.2 รายละเอียดของแต่ละ Interface และลำดับการทำงาน

สำหรับรูปที่ 4.1 เป็นหน้าจอที่ใช้ในการ กรอกข้อมูลเกี่ยวกับชื่อของโซลูชั่น, ชื่อของไฟล์ที่เป็นอนิเมชั่น, ชื่อของไฟล์ที่เป็นข้อความ, ชื่อของไฟล์ที่เป็นแพลททิส และข้อความที่เป็นคีย์เวิลด์ เพื่อเก็บไว้ในฐานข้อมูลสำหรับเรียกใช้ต่อไป

- Solution Title คือชื่อโซลูชั่นที่เก็บข้อมูลเกี่ยวกับไฟล์ต่างๆที่ใช้ในการแก้ไขปัญหา
- Animation File คือชื่อของไฟล์ที่เป็นไฟล์วีดีโอ
- Text File คือชื่อของไฟล์ข้อความที่ใช้ในการอธิบายขั้นตอนการทำงานของโปรแกรม
- Practice File คือ ชื่อของไฟล์ที่ใช้ในการแสดงการแก้ไขปัญหาที่เกิดขึ้น
- KeyWord คือ คำที่ใช้สำหรับการค้นหาวิธีการแก้ไขปัญหา



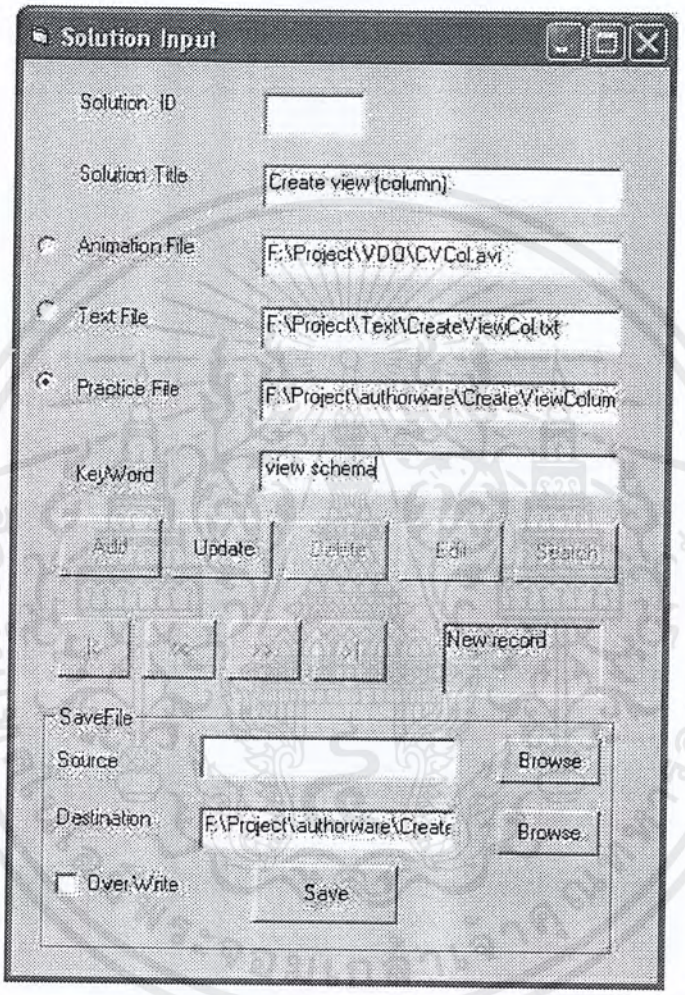
รูปที่ 4.1หน้าจอการกรอกข้อมูลของโซลูชั่น

สำหรับรูปที่ 4.2 เป็นการแสดงวิธีการใช้งานหน้าจอของ โซลูชั่นอินพุต

- ชื่อของโซลูชั่น คือ Create view(column)
- ชื่อของอนิเมชั่นไฟล์ คือ CVCol.avi
- ชื่อของไฟล์ข้อความ คือ CreateViewCol.txt

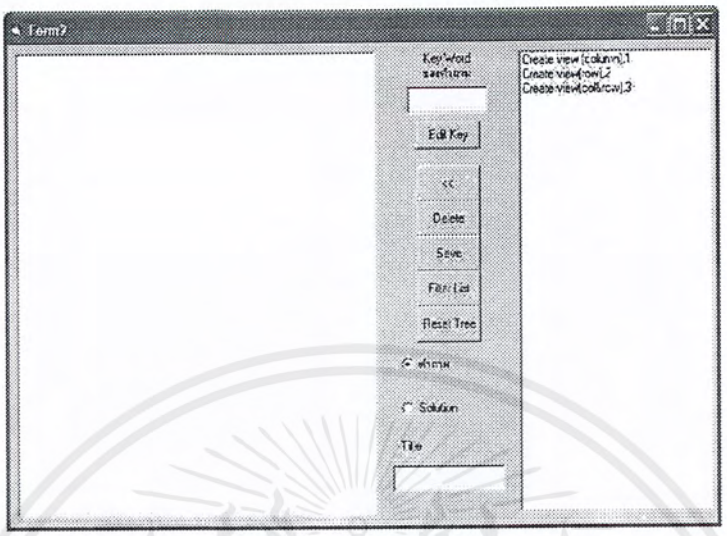
- ชื่อของไฟล์แพลททิส คือ CreateViewColumn.exe
- คีย์เวิลด์ที่ใช้ในการค้นหา คือ view และ column

หลังจากที่กรอกข้อมูลเรียบร้อยแล้วให้คลิกที่ปุ่ม Update เพื่อจบการทำงานของหน้าจอนี้



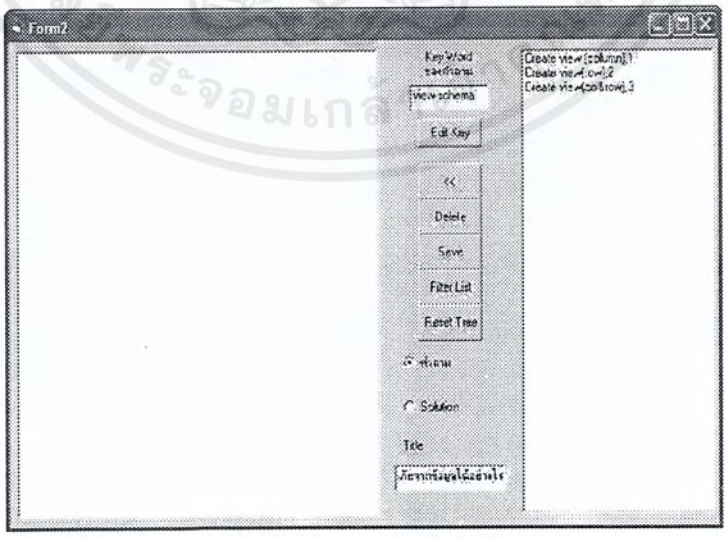
รูปที่ 4.2 ตัวอย่างการกรอกข้อมูลลงในฟอร์ม

สำหรับรูปที่ 4.3 เป็นหน้าจอที่ใช้สำหรับการกรอกคำถามและวิธีการที่ใช้ในการแก้ปัญหา หน้าจอทางซ้ายมือคือชื่อของโซลูชั่นที่จะใช้ในการแก้ไขปัญหา



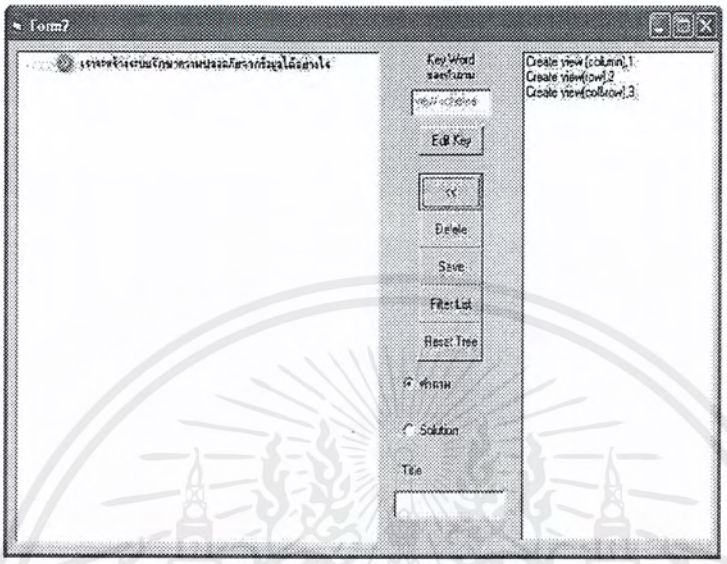
รูปที่ 4.3 หน้าจอที่ใช้สำหรับการเพิ่มคำถามและโซลูชั่น

สำหรับรูปที่ 4.4 เป็นตัวอย่างการเพิ่มคำถามให้กับโปรแกรม ให้ใส่คีย์เวิลด์ลงในช่องของคีย์เวิลด์เพื่อใช้ในการค้นหาและใส่คำถามลงในช่องของคำถาม จากนั้นให้คลิกที่ปุ่ม <<



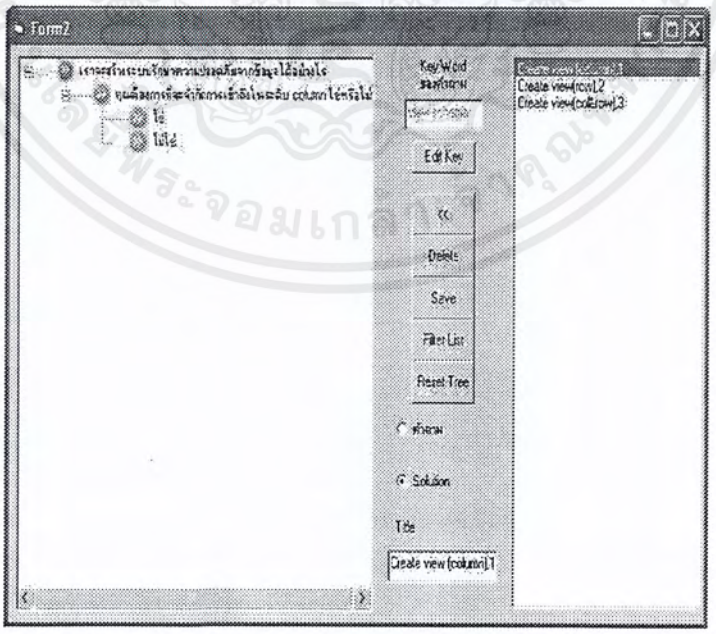
รูปที่ 4.4 ตัวอย่างการเพิ่มคำถาม

สำหรับรูปที่ 4.5 เป็นตัวอย่างที่ได้เพิ่มคำถามแล้ว



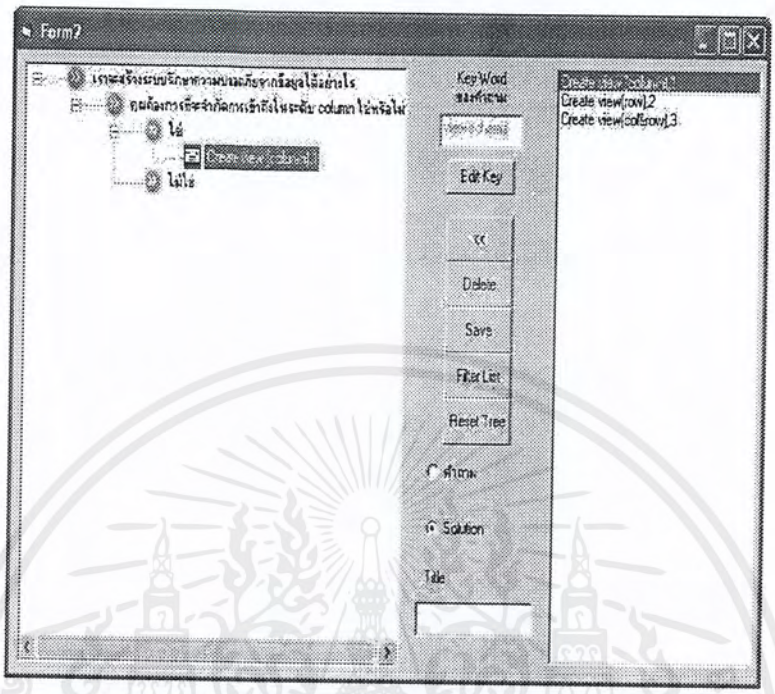
รูปที่ 4.5 ตัวอย่างการเพิ่มคำถาม

สำหรับรูปที่ 4.6 เป็นตัวอย่างการเพิ่มโซลูชั่น ขั้นตอนการทำงาน ให้คลิกที่ โซลูชั่นที่ต้องการจากทางขวามือ ในตัวอย่างนี้เลือกที่ Create view(column) จากนั้นให้คลิกที่ <<

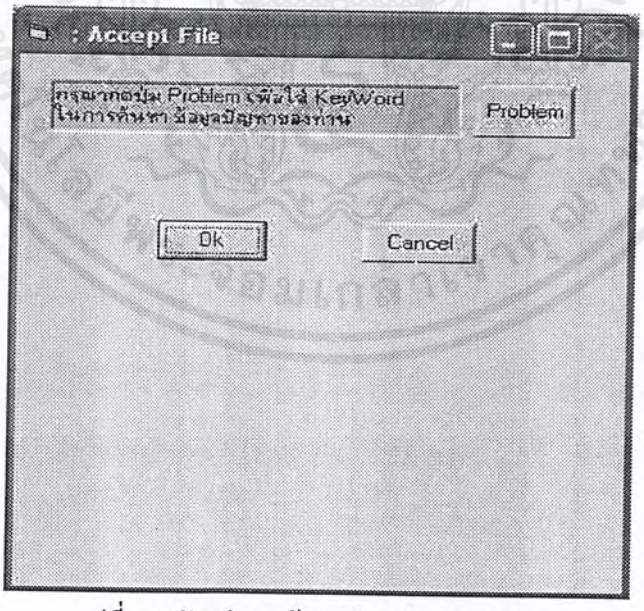


รูปที่ 4.6 ตัวอย่างการเพิ่มโซลูชั่น

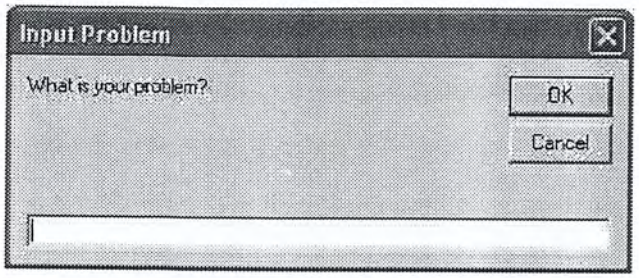
สำหรับรูปที่ 4.7 เป็นรูปแบบของการเพิ่มโซลูชั่นพาทที่เสร็จเรียบร้อยแล้ว



รูปที่ 4.7 ตัวอย่างการเพิ่ม โซลูชั่นพาทที่เสร็จเรียบร้อยแล้ว

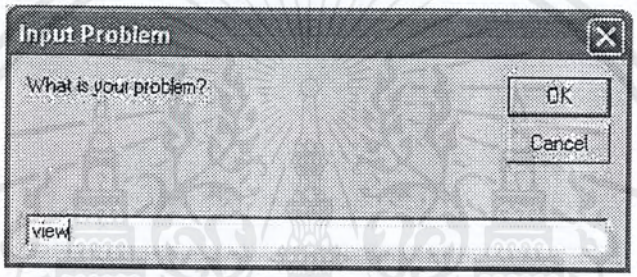


รูปที่ 4.8 ตัวอย่างหน้าจอสำหรับการหาSolution

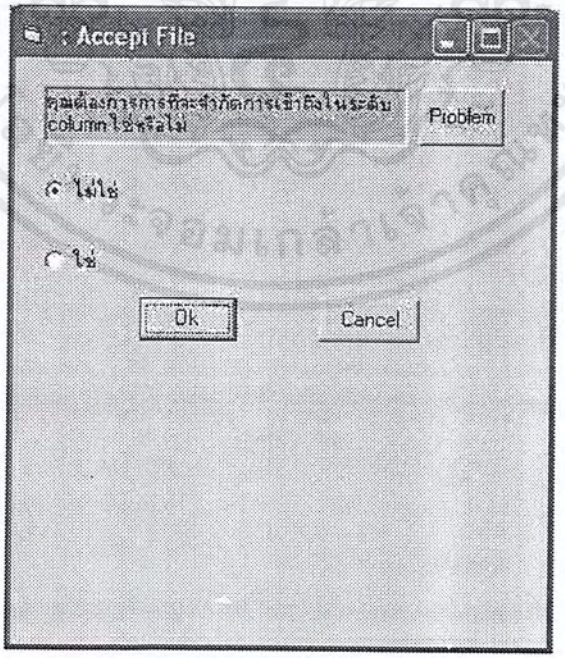


รูปที่ 4.9 หน้าจอที่ใช้ในการกรอกคีย์เวิลด์

จากนั้นให้ใส่คีย์เวิร์ดที่ต้องการ

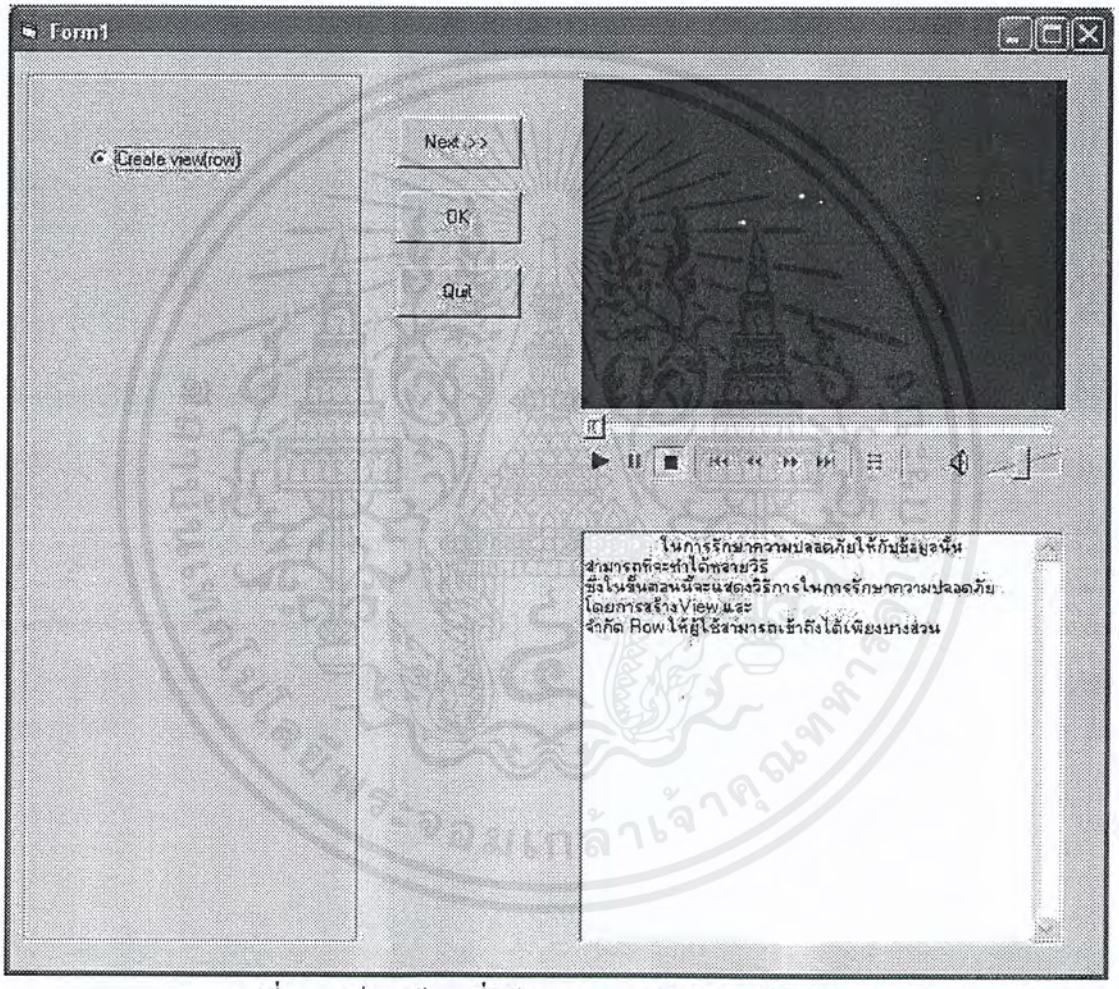


รูปที่ 4.10 แสดงการใส่ตัวอย่าง คีเวิร์ด



รูปที่ 4.11 หน้าจอที่ใช้ในการแสดงคำถาม

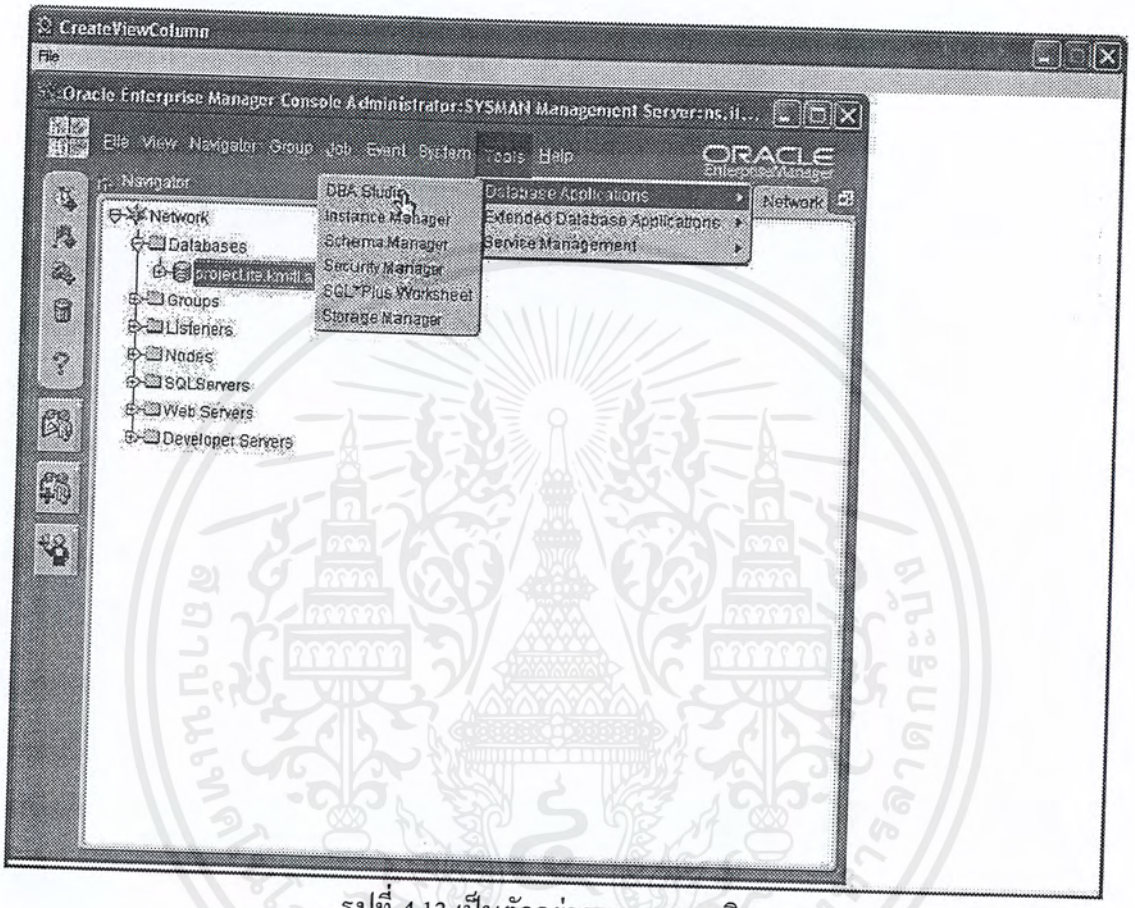
จะปรากฏหน้าจอที่แสดงคำถามให้ผู้ใช้เลือกคำตอบดังรูปที่ 4.11 เพื่อนำไปสู่การแก้ไขปัญหาในรูปที่ 4.12 จะแบ่งหน้าจอออกเป็น 3 ส่วน ส่วนแรกจะเป็นขั้นตอนในการแก้ไขปัญหา ส่วนที่สองจะเป็นวีดีโอที่บอกขั้นตอนในการแก้ไขปัญหา และส่วนที่สามจะเป็นส่วนข้อความที่ใช้อธิบายการแก้ไขปัญหา



รูปที่ 4.12 เป็นหน้าจอที่ใช้ในการแสดงวิธีการแก้ไขปัญหา

สำหรับรูปที่ 4.13 เป็นตัวอย่างของแพลททิสที่ใช้ในการแก้ไขปัญหา ซึ่งจะแนะนำวิธีการแก้ไขปัญหาให้ปฏิบัติตามทีละขั้นตอน

สำหรับรูปที่ 4.13 เป็นตัวอย่างของแพลททิสที่ใช้ในการแก้ไขปัญหา ซึ่งจะแนะนำวิธีการแก้ไขปัญหาให้ปฏิบัติตามทีละขั้นตอน



รูปที่ 4.13 เป็นตัวอย่างของ แพลททิส

บทที่ 5
ปัญหาและแนวทางการแก้ไขปัญหาในอนาคต

ปัญหาที่พบในการทำงาน

1. พบปัญหาในการติดตั้งโปรแกรมออราเคิล
2. เอกสารภาษาไทยที่เกี่ยวข้องกับการใช้งานออราเคิล มีน้อยไม่พอต่อการใช้งาน
3. โปรแกรม Visio ที่ใช้สำหรับการสร้างไดอะแกรม ไม่เสถียรทำให้การสร้างรายงานล่าช้า
4. หลังจากการติดตั้งออราเคิลแล้ว การใช้งานไม่เสถียรสามารถใช้งานได้เป็นบางครั้ง

การพัฒนาในอนาคต

1. เพิ่มเนื้อหาของส่วนโซลูชั่นให้มากขึ้น เพื่อให้สามารถครอบคลุมเนื้อหาได้มากขึ้น
2. พัฒนาให้ส่วนเพิ่มข้อมูลของโซลูชั่น ให้สามารถเพิ่มได้โดยการออนไลน์

บรรณานุกรม

1. Joseph Schmuller, ***UML in 24 Hours, First Edition, Sams Publishing***
2. อาจารย์ อภิเนตร อูนากูล, ***Object-Oriented Analysis and Design***, *พิมพ์ครั้งที่ 1*, แผนกตำรา คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
3. กิตติ ภักดีวัฒนะกุล-จำลอง ครูอุตสาหะ, ***Visual Basic ฉบับโปรแกรมเมอร์***, *พิมพ์ครั้งที่ 5*, สำนักพิมพ์ KTP COMP&CONSULT
4. กิตติ ภักดีวัฒนะกุล-กิตติพงษ์ กลมกล่อม, ***UML วิเคราะห์และออกแบบระบบเชิงวัตถุ***, สำนักพิมพ์ KTP COMP&CONSULT