อุปกรณ์วิเคราะห์การจัดคิวในเน็ตเวิร์ก

QUEUEING NETWORK ANALYZER

โดย

นายนพฤทธิ์ ทรัพย์ทิพยรัตนา

นายอานนท์ ชัยสุริยเทพกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

อุปกรณ์วิเคราะห์การจัดคิวในเน็ตเวิร์ก

QUEUEING NETWORK ANALYZER

โดย

นายนพฤทธิ์ ทรัพย์ทิพยรัตนา รหัสประจำตัว 36014206

นายอานนท์ ชัยสุริยเทพกุล รหัสประจำตัว 36014565

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

อุปกรณ์วิเคราะห์การจัดคิวในเน็ตเวิร์ก

QUEUEING NETWORK ANALYZER

โดย

นายนพฤทธิ์ ทรัพย์ทิพยรัตนา รหัสประจำตัว 36014206

นายอานนท์ ชัยสุริยเทพกุล รหัสประจำตัว 36014565

อาจารย์ที่ปรึกษา

อาจารย์อภิเนตร อูนากูล

วิทยานิพนธ์ปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง อุปกรณ์วิเคราะห์การจัดคิวในเน็ตเวิร์ก

(Queueing Network Analyzer)

ผู้จัดทำ

1. นายนพฤทธิ์ ทรัพย์ทิพยรัตนา รหัสประจำตัว 36014206
2. นายอานนท์ ชัยสุริยเทพกุล รหัสประจำตัว 36014565

อาจารย์ที่ปรึกษา

(อาจารย์อภิเนตร อูนากูล)

อุปกรณ์วิเคราะห์การจัดคิวในเน็ตเวิร์ก

นายนพฤทธิ์ ทรัพย์ทิพยรัตนา
นายอานนท์ ชัยสุริยเทพกุล
อาจารย์อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2539

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอระบบการจัดคิวบนเน็ตเวิร์ก ซึ่งทำขึ้นเพื่อช่วยอำนวยความสะดวกให้ผู้ใช้ในการวิเคราะห์ระบบเน็ตเวิร์ก ทำให้เกิดความแม่นยำในการคำนวณยิ่งขึ้น อุปกรณ์การจัดคิวบนเน็ตเวิร์กสามารถแบ่งการทำงานออกเป็นส่วนหลักๆ 3 ส่วน ได้แก่ ส่วนติดต่อกับผู้ใช้ ส่วนเก็บข้อมูล และ ส่วนวิเคราะห์เน็ตเวิร์ก

อุปกรณ์การจัดคิวบนเน็ตเวิร์กสร้างโดยใช้ภาษาเดลไฟทั้งหมด ส่วนการออกแบบได้นำออปเจ็กต์โดเม่น และ วิชวลซิมเน็ต มาช่วยในการออกแบบคลาส และการเชื่อมต่อต่างๆ สำหรับจากการวิเคราะห์ของโปรแกรม มีผลให้ทราบถึงการไหลของข้อมูลที่เข้ามาในระบบทำให้ผู้ออกแบบระบบสามารถจัดวางทรัพยากรได้อย่างเหมาะสม

Queueing Network Analyzer

Nopparit Suptippayarattana

Anont Chaisuriyathepkul

Apinetr Unakul Advisor

1996

Abstract

This thesis is about a preseutation of queueing system on the network which is developed and implemented to convenient the users in analysis the network , which helps on the calculating to be more accurate . The network queueing device can be provided into 3 main parts , which are the interface part , the database part and the network analysis part

All the network queueing devices is written in Delphi , software is designed by using Object-Domain and Visual Simnet for designing classes and the connections. The result of the analysis is program show the flow of data that gets in the system which helps the system designer has better plans for the resource managment.

สารบัญ

	หน้า
บทคัดย่อ	
บทที่ 1 บทนำ	8
บทที่ 2 ทฤษฎีและหลักการ	9
2.1 ทฤษฎีของคิว	9
2.2 ทฤษฎีของออปเจ็กต์	16
2.3 พื้นฐานของโปรแกรม	36
2.4 การใช้งานโปรแกรมอื่นๆ	48
บทที่ 3 การคำนวณและการออกแบบ	50
3.1 การคำนวณเกี่ยวกับการจัดคิว	50
3.2 ออปเจ็กต์และพารามิเตอร์ต่างๆทั้งหมดในโปรแกรม	53
บทที่ 4 การทดลองและผล	61
บทที่ 5 บทวิจารณ์และสรุป	68
5.1 ประโยชน์ของโครงงาน	68
5.2 บทวิจารณ์สรุป	68
5.3 แนวทางการพัฒนาต่อไป	69
กิตติกรรมประกาศ	70
หนังสืออ้างอิง	71

สารบัญภาพ

	หน้า
รูปที่ 1.1 ไดอะแกรมของระบบการจัดคิวบนเน็ตเวิร์ก	8
รูปที่ 2.1 รูปแบบของทฤษฎีคิว	12
รูปที่ 2.2 รูปแบบของทฤษฎีคิวเมื่อไม่เข้าคิว	12
รูปที่ 2.3 ลักษณะและรูปแบบของคอมโปเนนต์หลักที่ใช้จริง	14
รูปที่ 2.4 ลักษณะของคอมโปเนนต์ขั้นสูง	15
รูปที่ 2.5 ลักษณะของโปรแกรเดลไฟที่ใช้	47
รูปที่ 2.6 การใช้ทูลออปเจ็กต์โดเมนต์ออกแบบคลาส	48
รูปที่ 2.7 ลักษณะของโปรแกรมวิชวลซิมเน็ต	49
รูปที่ 3.1 คลาสไดอะแกรม	53
รูปที่ 3.2 ลักษณะของการติดต่อกับผู้ใช้โปรแกรม	59
รูปที่ 3.3 การใช้งานโปรแกรมวิเคราะห์การจัดคิว	60
รูปที่ 4.1 การใช้งานคอมโปเนนต์ทูลต่างๆ	61
รูปที่ 4.2 การใช้คลิ๊กคอมมานต์ในโปรแกรม	62
รูปที่ 4.3 เมนูหลักที่ใช้ในโปรแกรม	63
รูปที่ 4.4 การใส่ข้อมูลสำหรับออปเจ็กต์ต่างๆ	64
รูปที่ 4.5 โมเดลจำลองของแอพลิเคชั่น	66
รูปที่ 4.6 การวิเคราะห์ของโปรแกรม	67

บทที่ 1
บทนำ

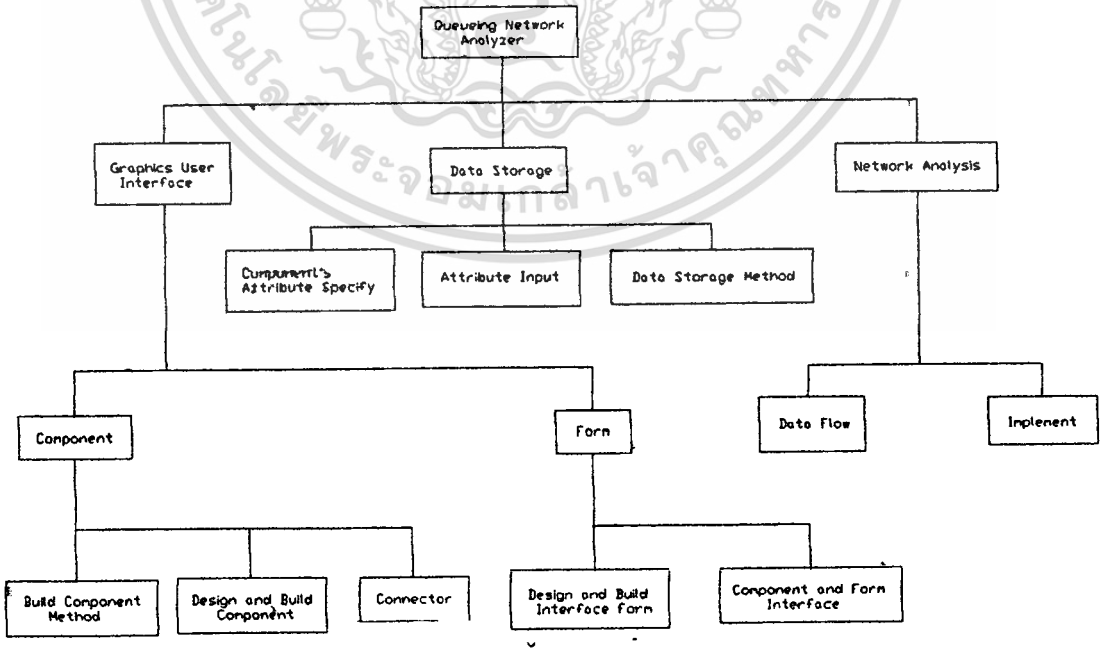
ในปัจจุบันองค์กรทั่วไป และ สถานศึกษาต่างๆ นิยมใช้ และให้ความสนใจในการศึกษาเกี่ยวกับ เน็ตเวิร์ก (Network) อย่างกว้างขวาง เมื่อมีการใช้งานมากขึ้น จึงเกิดการรอใช้งานขึ้น เนื่องด้วยทรัพยากรมีจำกัด

ดังนั้นจึงมีการคิดค้น ทำเครื่องมือ ขึ้นมาเพื่อใช้ในการ ออกแบบระบบ เน็ตเวิร์ก เพื่อให้องค์กรต่างๆใช้ทรัพยากรที่มีอยู่อย่างจำกัดนั้นให้มีประสิทธิภาพมากที่สุด เรียก เครื่องมือ นี้ ว่า อุปกรณ์การจัดคิวบนเน็ตเวิร์ก (Queueing Network Analyzer)

อุปกรณ์การจัดคิวบนเน็ตเวิร์ก สามารถแบ่งการทำงาน ออกเป็น เฟส(Phase)การทำงานได้ 3 ส่วนหลัก คือ

1. ส่วน ติดต่อกับผู้ใช้ (Graphics User Interface)
2. ส่วนเก็บข้อมูล (Data Storage)
3. ส่วนวิเคราะห์เน็ตเวิร์ก (Network Analysis)

ซึ่งแต่ละส่วนได้มีการออกแบบย่อยลงมา เป็น Top Down Design ดังรูปที่ 1.1



รูปที่ 1.1 ไดอะแกรมของระบบการจัดคิวบนเน็ตเวิร์ก

บทที่ 2
ทฤษฎีและหลักการ

2.1 ทฤษฎีของคิว (Queue)

ระบบของคิว

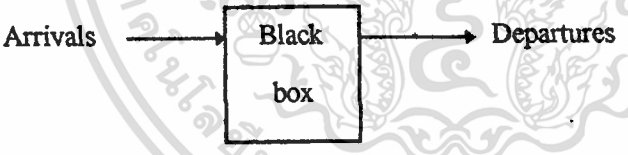
คิว มีหลายชนิด ซึ่งประกอบด้วยส่วนหลัก 3 ส่วนคือ

1. การมาขอรับบริการของโพรเซส (Process)
2. การ กระจายการให้บริการ (Distribution)
3. จำนวนของผู้ให้บริการ (Server)

ทฤษฎีของลิตเติ้ล (Little)

ทฤษฎี ลิตเติ้ล เป็นทฤษฎีที่ใช้เพื่อแสดงความสัมพันธ์ของจำนวนงานในระบบใดๆ กับเวลาที่ได้ใช้ในระบบนั้นๆ โดยเขียนเป็นสมการได้ดังนี้

จำนวนงานเฉลี่ยในระบบ = อัตราการเข้ารับบริการ x เวลาเฉลี่ย



$$N = \lambda T \ , \ Nq = \lambda W$$

N = จำนวนของผู้ขอใช้บริการในระบบโดยเฉลี่ย

T = เวลาที่ผู้ใช้บริการใช้ในระบบโดยเฉลี่ย

Nq = จำนวนของผู้ขอใช้บริการที่รอในคิวโดยเฉลี่ย

W = เวลาที่ผู้ขอใช้บริการรอในคิวโดยเฉลี่ย

λ = อัตราการขอใช้บริการ

ตัวอย่างของคิว

M/M/1

- พารามิเตอร์ (Parameters) :
  - λ = อัตราการขอรับบริการ (Arrival rate in jobs per unit time)
  - μ = อัตราการให้บริการ (Service rate in jobs per unit time)
- ความหนาแน่นของการจราจร (Traffic intensity) : $\rho = \lambda / \mu$

M/M/m

- พารามิเตอร์ :
  - λ = อัตราการขอรับบริการ
  - μ = อัตราการให้บริการ
  - m = จำนวนของผู้ให้บริการ (Number of Servers)
- ความหนาแน่นของการจราจร : $\rho = \lambda / (m\mu)$

การเข้าของข้อมูลอาจแบ่งเป็น 3 แบบ คือ

1. Uniform คือ การเข้าของข้อมูลแบบ คงที่
2. Poisson คือ การเข้าของข้อมูลในลักษณะ Exponential
3. Normal Distribution คือการเข้าของข้อมูลในลักษณะเชิงสถิติ

สิ่งที่ต้องทราบเกี่ยวกับข้อมูลที่เข้ามาใน คิว

1. การสุ่มของโพรเซส (Random Process)
2. อนุกรมคือ การแปลงจาก Exponential เป็นอนุกรมได้ แล้วใช้ อนุกรมของเทย์เลอร์ (Taylor Series) กระจาย Exponential

สมการของพัวซอง (Poisson)

n = จำนวนผู้เข้ารับบริการในระบบ

λ = อัตราการให้บริการ (รวมการรอในคิว)

อาจเปลี่ยน λ เป็น μ ได้ ถ้าหากว่าใช้กับการบริการแบบไม่รวมการรอในคิว

ความหมายของตัวอักษรต่างๆ

ระบบของคิว สามารถกำหนดด้วย อักษร 3 ตัว คั่นด้วย "/" เช่น M/M/1

อักษรตัวที่ 1 เกี่ยวกับการเข้ามาทั้งหมด รวมถึงระยะห่างจาก ข้อมูล1 กับ ข้อมูล2 ที่เข้ามาในคิว

อักษรตัวที่ 2 เกี่ยวกับการกระจายการให้บริการ โดยอาจแบ่งได้เป็น 3 แบบ คือ

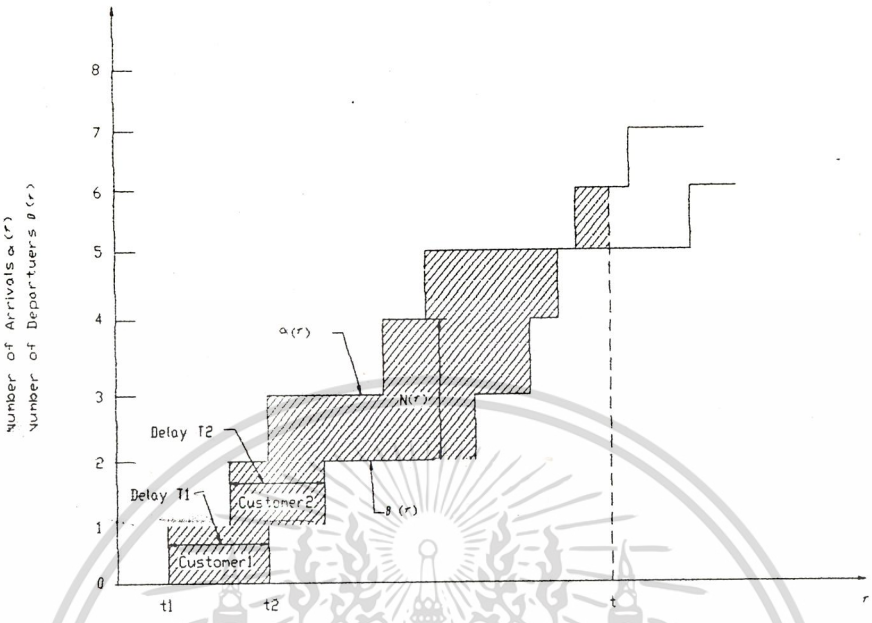
1. M : หมายถึง ไม่มีหน่วยความจำ (Memory)คือเป็นแบบ พัวซอง
2. G : หมายถึง General คือ เป็นการกระจายปกติ
3. D : หมายถึง Deterministic เป็นค่าเฉพาะ ไม่ต้องแปลงอีก เป็นแบบคงที่

อักษรตัวที่ 3 เกี่ยวกับจำนวนการให้บริการ

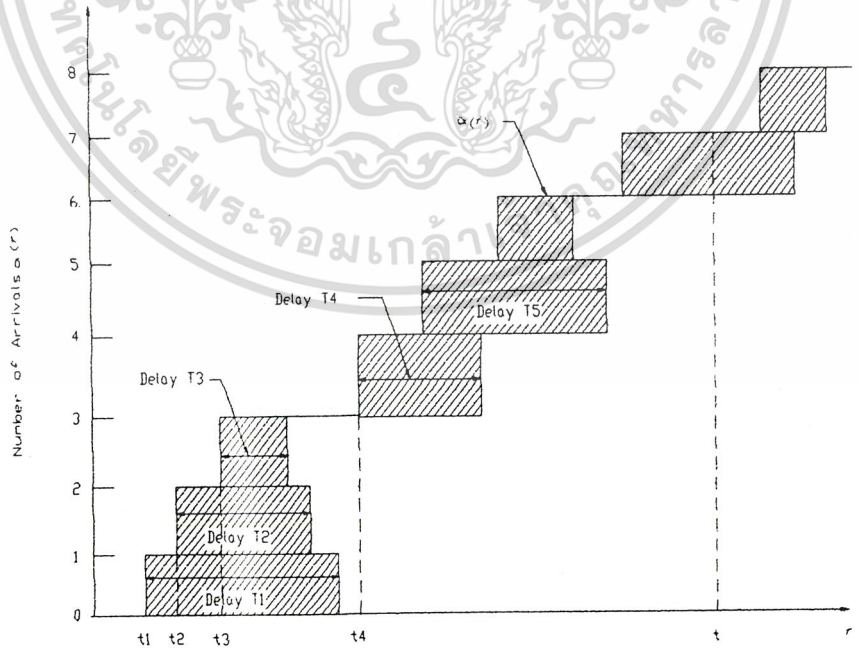
รูปแบบของความเป็นไปได้ของทฤษฎีลิตเติ้ล แสดงในรูปที่ 2.1

รูปแบบเมื่อไม่คำนึงถึงการบริการแบบไม่เข้าคิว จะได้ลักษณะของ ทฤษฎีลิตเติ้ลแสดงในรูปที่ 2.2

รูปที่ 2.2 รูปแบบของทฤษฎีคิวเมื่อไม่เข้าคิว



รูปที่ 2.1 รูปแบบของทฤษฎีคิว



รูปที่ 2.2 รูปแบบของทฤษฎีคิวเมื่อไม่เข้าคิว

ความสามารถของ ซอฟต์แวร์ (Software) ที่จะได้รับ

ซอฟต์แวร์ ที่สร้างขึ้นจะเป็น อุปกรณ์ ที่มีความสามารถหลักๆอยู่ 3 ประการดังนี้คือ

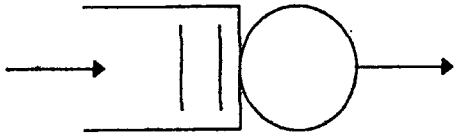
1. ช่วยออกแบบระบบการจัดคิวบนเน็ตเวิร์ก
2. ทำการวิเคราะห์ ระบบการจัดคิวบนเน็ตเวิร์ก
3. ทำการจำลอง ระบบการจัดคิวบนเน็ตเวิร์ก

การออกแบบระบบ

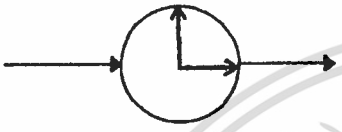
ซอฟต์แวร์ ที่สร้างขึ้นมีความสามารถในการ Drag และ Drop Icon ต่างๆเพื่อนำมาสร้างเป็น คอมโปเนนต์ (Component) บนฟอร์มที่ใช้ในการออกแบบ สามารถใช้การ Drag และ Drop เพื่อทำการเชื่อมต่อ อุกศร (Arc) เข้ากับ ตัวเชื่อมต่อ (Connector) แต่ละตัว มีความสามารถในการทำการย่อขยาย (Scaling) ซึ่งจะช่วยให้สามารถทำการซูม (Zoom) ในส่วนสำคัญต่างๆได้ ทำแผนผัง (Hierachy Tree) ของ ฟอร์มที่ใช้ในการออกแบบ ได้ รวมทั้งสามารถใส่ค่า แอตทริบิวต์ (Attribute) ต่างๆขณะทำการออกแบบ เพื่อ เก็บใช้ในการทำ การวิเคราะห์ และ การจำลอง ได้ โดยเพียงแต่คลิ๊ก (Click) ที่ตัว คอมโปเนนต์ที่ต้องการก็จะเกิด กล่องรับอินพุท (Input box) เพื่อให้ใส่ค่า แอตทริบิวต์ ต่างๆที่ต้องการ

การทำการ Analysis

ซอฟต์แวร์ ที่สร้างขึ้นจะมีความสามารถในการช่วยชี้หาจุดที่เป็นจุดบกพร่อง หรือจุดที่ทำให้ การทำงาน (Performance)ของระบบตกลงหรือในส่วนที่มี ข้อบกพร่อง (Error) ต่างๆที่อาจเกิดขึ้นในขั้นตอนการออกแบบ เพื่อทำให้ระบบที่เราทำการออกแบบ ถูกต้องสมบูรณ์ที่สุดและสามารถนำไปใช้งานได้ทันที



Service

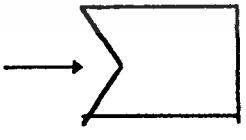


Delay

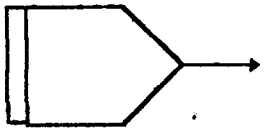
Submodel

Branch

Source

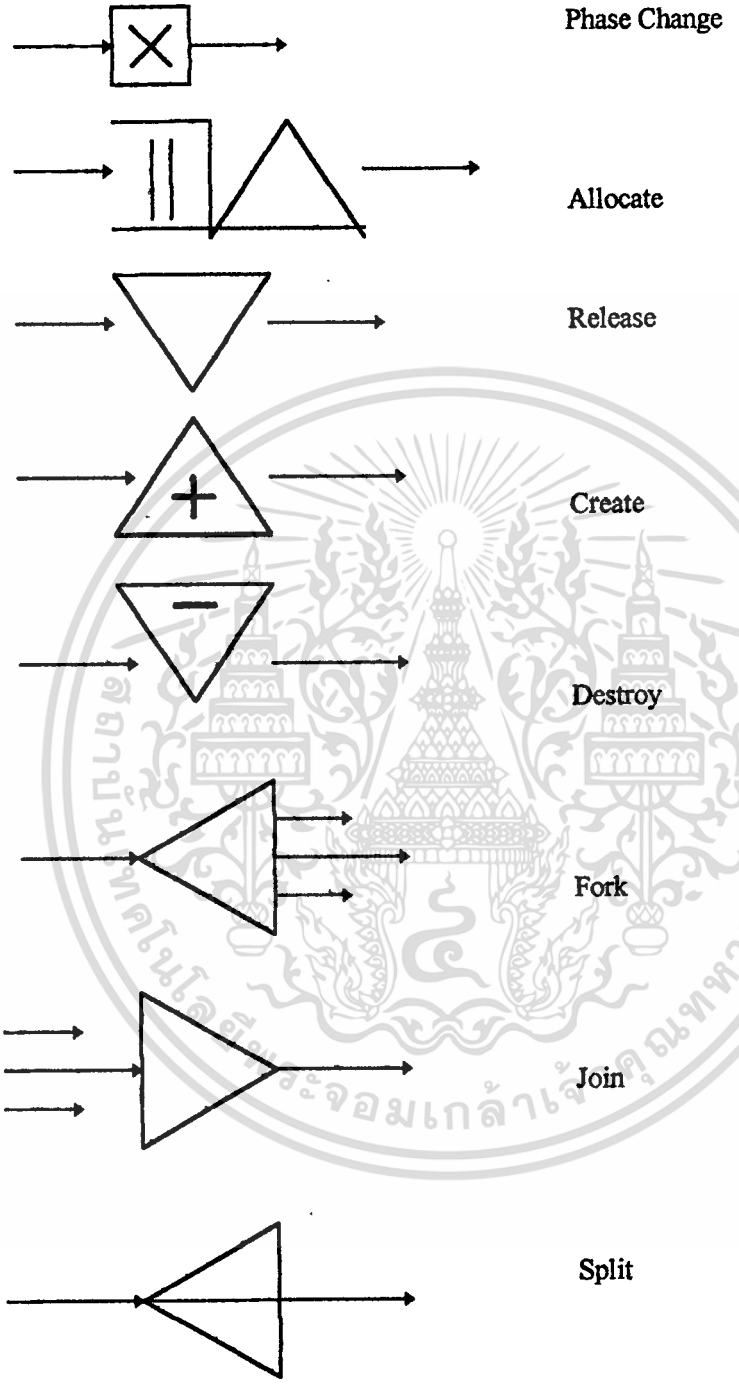


Sink



Submodel Entry

รูปที่ 2.3 ลักษณะและรูปแบบของคอมโปเนนต์ หลักที่ใช้



รูปที่ 2.4 ลักษณะของ คอมโปเนนต์ ขั้นสูง

2.2 ทฤษฎีของออปเจ็กต์ (Object)

โอโอพี (OOP)

การอธิบายความหมายของโอโอพี(OOP : Object - Oriented Programming) ซึ่งจะต้องเขียนโปรแกรมด้วยภาษาที่เป็นโอโอแอล (OOL : Object - Oriented Language) อธิบายได้หลายแนวทาง แนวทางหนึ่งคือ กล่าวถึงเรื่องสกุล ตามที่จะนำมากล่าวในหัวข้อต่อไปนี้

สกุลและการสร้าง (Class and Create)

สกุล (Class) คือแบบจำลอง (Model) ของสิ่งใดก็ได้ที่สามารถสร้างได้ด้วยโปรแกรมคอมพิวเตอร์เช่น เมื่อต้องการเขียนโปรแกรมแสดงภาพบิตแม็พ (Bitmap)บนจอภาพควรต้องเตรียมโปรแกรมย่อยอย่างน้อย 2 โปรแกรม คือ

1. Load เพื่อโหลดภาพจากไฟล์บิตแม็พ มาเก็บไว้ในหน่วยความจำ
2. Show เพื่อให้แสดงภาพ ณ ตำแหน่งที่ต้องการ

และจะต้องกำหนดตัวแปรอย่างน้อย 1 ตัว เพื่อเก็บภาพบิตแม็พ ซึ่งถ้าต้องการเขียนโปรแกรมในแบบโอโอพี จะต้องนำโปรแกรมส่วนนี้ไปสร้างเป็นสกุลดังแสดงใน ลิสติงที่ 2.1 โดยสมมติว่าเป็นโปรแกรมบนดอส (รันไม่ได้) กล่าวคือ

1. ได้กำหนดสกุล MyImageT ด้วยคำสงวน (Reserved Word) ว่า ออปเจ็กต์ ในเดลไฟใช้เป็นคลาส มีลักษณะเช่นเดียวกับ เรคคอร์ด (Record) คือประกอบด้วยฟิลด์ (Field) ต่างๆแตกต่างกันที่ สกุลจะประกอบด้วยฟิลด์ข้อมูล หรือ แอตทริบิวต์ เช่น x,y1 และ bmpP เป็นต้น กับฟิลด์กิจกรรม (Method) คือ โหลด และ แสดง สำหรับ ค่า Init และ Destroy มีฐานะเป็นคอนสตรัคเตอร์และดีสตรัคเตอร์ตามลำดับ
2. เมื่อกำหนดสกุลแล้วให้ป้อนโปรแกรมของกิจกรรมตามปกติแต่ให้ระบุชื่อสกุลกำกับ เช่น แสดงเป็น MyImageT.Show
3. การสร้างสกุลตามที่กล่าวมานี้ประกอบด้วย การกำหนดสกุลซึ่งให้กำหนดในส่วนการประกาศ (Declaration Part) และป้อนโปรแกรมของกิจกรรมซึ่งให้ป้อนโปรแกรมของกิจกรรมในส่วนการใช้งาน (Implement Part) เป็นเพียงแต่การแยกส่วนเท่านั้น โดยให้ส่วนการประกาศ อยู่ก่อน ส่วนการใช้งาน แต่ถ้านำสกุลนี้ไปทำเป็นยูนิต (ในเดลไฟต้องทำเป็นยูนิต) จะต้องกำหนดสกุลในส่วนการเชื่อมต่อ (Interface) และป้อนโปรแกรมในส่วนการประกาศ ดังที่จะกล่าวต่อไป

และนี่คือการสร้างแบบจำลอง เพราะอาจสร้างเป็นแบบจำลองของนามบัตร ปฏิทิน ตารางสอน ใบสั่งซื้อ ฯลฯ

ลิสติงที่ 2.1

```
program OOPExample;
uses WinCrt;

type
 MyImageTP    = ^ MyImageT;
 MyImageT     = class
   x1,y1        : Integer;
   bufSi        : LongInt;
   bmpP         : Pointer;
 constructor  Init(xx1,yy1  : Integer );
 destructor   Destroy;
 procedure  Load(fiNme  : String );
 procedure  Show;
          end;
(* ------------ Implement Part ---------------- *)
constructor  MyImageTInit(xx1,yy1 : Integer );
begin
 x1 := xx1 ;  y1 := yy1 ;  bmpP : nil;
end;

procedure  MyImageT.Load(fiNme : String);
begin
 bufSi := BitmapSize;  GetMem(bmpP,bufSi);
 { Load Bitmap Code }
end;
procedure  MyImageT.Show;
begin
```

```
  { Display Bitmap Code }
end;

(* ---------- Main Program ---------- *)

var myPct : MyImageT ;  pctP : MyImageTP ;

begin
  myPct.Init(,);  myPct.Load('c:\aaa.bmp');  myPct.Show ;
  pctP := New(MyImageTP , Init(300,0));
  pctP^.Load('c:\bbb.bmp');
  Dispose(pctP , Destroy);
  myPct.Destroy;
end.
```

การใช้งานสกุล

เมื่อสร้างสกุลแล้ว สามารถนำไปใช้งานโดยตรง หรือนำไปสร้างผู้สืบสกุลต่อ ในหัวข้อนี้จะแสดงถึงการใช้งานโดยตรง ตรงส่วนที่แสดงข้อความว่า MAIN PROGRAM กล่าวคือ

1. ได้กำหนดตัวแปร myPct เป็นอินสแตนซ์ (Instance) ของสกุล MyImageT อินสแตนซ์ต่างกับตัวแปรตรงที่ยังให้ค่าไม่ได้
2. ให้กำเนิด myPct เป็นออปเจ็กต์ด้วยคอนสตรัคเตอร์คือ Init ซึ่งด้วยวิธีการของไอโอพีจะมีการเตรียมค่าต่างๆ
3. เมื่อให้กำเนิดแล้ว จึงจะอ่านหรือให้ค่า รวมทั้งส่งข้อความ (Message) ไปให้ได้ ข้อความคือชื่อกิจกรรม ซึ่งเมื่อนำมาใช้ในลักษณะนี้ จะเรียกว่าข้อความเช่น myPct.Load ("C:/aaa.bmp") หมายถึงให้ myPct โหลดภาพบิตแม็พที่อยู่ในไฟล์ C:/aaa.bmp
4. เนื่องด้วยจะใช้คำว่าอินสแตนซ์ในช่วงสั้น คือตั้งแต่ var ถึง Init เท่านั้นส่วนมากจึงไม่กล่าวถึงคำนี้ จะใช้คำว่าออปเจ็กต์แทน ในหนังสือนี้เช่นกัน จะใช้คำว่าอินสแตนซ์เมื่อต้องการแยกเท่านั้น

5. เมื่อไม่ใช้ออปเจ็กต์แล้ว จะต้องยุบด้วยดีสตรัคเตอร์คือ Destroy เพื่อยุบค่าต่างๆในส่วนของ OOP ซึ่งในโปรแกรมได้ให้ยกเลิกหน่วยความจำที่จองไว้ด้วย

จากการกำหนดตัวแปร จะได้ myPct เป็นออปเจ็กต์ที่เป็นสแตติกส์ (Static) ซึ่งไม่ค่อยใช้ และโดยเฉพาะเมื่อเขียนโปรแกรมกับวินโดวส์ จะใช้ออปเจ็กต์ที่เป็นไดนามิกส์ (Dynamics) เช่น pctP จะอ้างถึงออปเจ็กต์ซึ่งเป็นพอยน์เตอร์ จะต้องอ้างถึงในฐานะพอยน์เตอร์ คือใช้เครื่องหมาย " ^ " คือ Caret กำกับ ซึ่งในเดลไฟจะให้กำเนิดออปเจ็กต์เป็นไดนามิกส์เท่านั้น จึงกำหนดให้ไม่ต้องใช้ New และ Dispose คือเมื่อเรียกใช้คอนสตรัคเตอร์ จะให้กำเนิดใน Heap และเมื่อเรียกใช้ดีสตรัคเตอร์ จะยกเลิกหน่วยความจำให้ด้วย กับไม่ต้องใช้ Caret เช่น เขียนโปรแกรมได้เป็น pctP.Load() นั่นคือชื่อออปเจ็กต์ทุกชื่อในโปรแกรมต่างๆที่จริงแล้วเป็นพอยน์เตอร์

อนึ่ง ในการสร้างสกุล ผู้เขียนโปรแกรมอาจสร้างสกุล A แล้วนำไปสร้างเป็นสกุล AB และ AC ซึ่งส่วนมากแล้วไม่สามารถนำสกุล A มาใช้ได้โดยตรง ต้องดูในคำอธิบาย เช่น สกุล Tstrings ในเดลไฟ จะแนะนำว่าหากต้องการใช้โดยตรง ให้ใช้ สกุล TStringList แทน และการใช้ สกุล โดยตรงตามที่กล่าวมานี้ เป็นวิธี อีกวิธีหนึ่งคือนำไปใช้ในสกุลอื่น จึงจะขอเรียกเป็นการใช้โดยตรง กับการใช้ในสกุลอื่น ในการอ้างถึงต่อไป

สร้างผู้สืบสกุล

ในเดลไฟได้นำสกุลจำนวนหนึ่งมาทำเป็นคอมโปเนนต์ เพื่อให้เราสามารถใช้โดยตรงได้โดยง่าย ซึ่งสามารถนำมาใช้สร้างผู้สืบสกุลได้เช่นกัน รวมทั้งสกุลที่ไม่ได้ทำเป็นคอมโปเนนต์ด้วย คือทุกสกุล แต่ที่ใช้บ่อยคือสกุล Tform ดังเช่นเมื่อเปิดโปรเจ็กต์ใหม่ เดลไฟจะเปิด Form1 และ Unit1 ให้ใน Unit1 จะมี สกุล Tform1 ซึ่งสร้างเสร็จแล้ว

1. ได้กำหนดให้ Tform1 สืบสกุลจากTform คือที่ระบุไว้ในวงเล็บหลังคำว่า Class
2. โดยที่ Tform เป็นสกุลหน้าต่าง จึงได้ Tform1 เป็นสกุลหน้าต่างด้วยจากการสืบสกุล
3. Tform1 เป็นสกุลที่ถูกต้องตามหลักการเขียนโปรแกรมในแบบโอโอพีแล้วเพียงแต่เป็นสกุลเปล่า
4. เพราะเป็นสกุลที่ถูกต้องจึงนำไปใช้ได้ โดยกำหนดอินสแตนซ์ของสกุลนี้เป็น Form1 และให้กำเนิดเป็นออปเจ็กต์ (เมื่อรัน) ในโปรแกรมหลัก ดังนั้นจึงทำให้สามารถรันโปรแกรมเมื่อเปิดโปรเจ็กต์ใหม่ได้ ได้เป็นโปรแกรมเปล่า

ต่อมาเมื่อเรากำหนดคอมโปเนนต์ในฟอร์ม เราจะกำหนดอินสแตนซ์ของสกุลของคอมโปเนนต์นั้นให้ในสกุล Tform1 ซึ่งเป็นของโปรแกรม MyFirst และเมื่อป้อนโปรแกรม จะได้เป็นสกุล

ที่สมบูรณ์สำหรับงานนี้ จะเห็นว่าการโปรแกรมด้วยเดลไฟ แท้ที่จริงแล้วเป็นการโปรแกรมในแบบโอโอพี (อย่างผู้ชำนาญการเขียนโปรแกรมในแบบโอโอพี) สำหรับการให้กำเนิดอินสแตนซ์ที่อยู่ในสกุล Tfrom1 เป็นออปเจ็กต์ เข้าใจว่าอยู่ในสกุล Tform (ซึ่งหมายรวมทั้งอาจอยู่ในบรรพบุรุษของ Tform) และเช่นเดียวกันกับการยุบ

ออปเจ็กต์ (Object)

เมื่อสร้างสกุลแล้วสามารถนำไปกำหนดอินสแตนซ์ให้กำเนิดเป็นออปเจ็กต์ได้หลายตัว ซึ่งเป็นข้อดีของการโปรแกรมไปแบบโอโอพี ดังเช่นเราสามารถกำหนดคอมโปเนนต์ได้ชนิดละหลายตัว แต่ในเรื่องของฟอร์ม เนื่องจากเป็นโปรแกรมของหน้าต่างหนึ่งๆเท่านั้นในเดลไฟจึงกำหนดให้ใช้สำหรับหน้าต่างเดียว คือนำสกุลไปให้กำเนิดเป็นออปเจ็กต์เพียงตัวเดียวเช่น Form1 หากต้องการใช้กับหน้าต่างหลายบาน ให้ใช้หน้าต่าง MDI

สกุลกับการสืบสกุล

สกุลตามที่กล่าวมานี้ทางด้านโอโอพีเรียกว่าเป็นการเอนแคปซูล (Encapsulation) หมายถึงให้รวมข้อมูลและกิจกรรมไว้ในที่แห่งเดียวกัน และให้อยู่ในที่ปกป้อง ไม่ให้ผู้อื่นเข้าถึงได้โดยง่าย ผู้อื่นก็คือผู้ใช้ (ใช้ออปเจ็กต์ของสกุล) กับผู้สืบสกุล (นำ สกุล นี้ไปสร้างผู้สืบสกุลต่อ) ซึ่งผู้สร้าง ผู้ใช้ ผู้สืบสกุล อาจเป็นคนเดียวกันต่างวาระกัน และผู้ใช้กับผู้สืบสกุลอาจเป็นคนเดียวกันในวาระเดียวกัน จะเห็นว่าเราเป็นทั้งผู้ใช้ ใช้งาน เช่น Label1 และเป็นผู้สืบสกุล สร้างสกุล Tform1 สืบสกุลจาก Tform ของเดลไฟ คือถือว่าการสร้างสกุลเป็นการปกป้องข้อมูลและกิจกรรม โดยให้ผู้ใช้และผู้สืบสกุลเข้าถึงได้ตามเงื่อนไข และมีเรื่องสืบเนื่องดังนี้

การเข้าถึง

ให้ผู้ใช้ เข้าถึงได้ผ่านทางออปเจ็กต์ของสกุลนั้น ดังเช่นในโปรแกรม MyFirst เราอาจใช้คำสั่งว่า Label1.Hide เป็นการส่งสื่อความ (message) ไปให้ Label1 ให้หายตัว ซึ่งสื่อความก็คือชื่อกิจกรรมในสกุล Tlabel สำหรับผู้สืบสกุลให้เข้าถึงได้โดยตรง เช่นเราอาจใช้คำสั่งว่า Hide ซึ่งหมายถึงให้ Form1 หายตัว เป็นการเรียกใช้กิจกรรมโดยตรง

การปกป้อง (Protection)

นอกจากความเป็นสกุลแล้ว ทางด้านโอโอพียังต้องการมาตรการของการปกป้องเพิ่มอีก ซึ่งในภาษาที่เป็นโอโอแอลจะต้องจัดให้มีคำว่า Private กับ Public เพื่อให้เรากำหนดฟิลด์ข้อมูลและฟิลด์กิจกรรม ภายใต้หัวข้อเหล่านี้ ทำให้แบ่งส่วนการกำหนดสกุลของสกุล Tform1 ออกได้เป็น 3 ส่วน คือส่วนบนที่มีฟิลด์ข้อมูลเป็น Label1 , Edit1 และ Button1 และ ฟิลด์กิจกรรมอีก 1 กิจกรรม ซึ่งจะมีความเป็นส่วนรวมกับส่วนของ Private และ Public ซึ่งมีความหมายดังนี้

1. Private หมายถึงความเป็นส่วนตัว คือถ้ากำหนดสิ่งใด (ฟิลด์ข้อมูลหรือฟิลด์กิจกรรม) ไว้ในส่วนนี้ จะรู้จักหรือเห็นในสกุลนี้เท่านั้น เป็นการปกป้องทั้งผู้ใช้และผู้สืบสกุลไม่ให้เข้าถึงสิ่งเหล่านี้
2. Public หมายถึงความเป็นส่วนรวม ผู้ใช้และผู้สืบสกุลสามารถเข้าถึงสิ่งที่กำหนดไว้ในส่วนนี้ได้ (ด้วยวิธีที่กล่าวในหัวข้อ " การเข้าถึง ")ซึ่งแต่เดิมการใช้คำสงวนสองคำนี้เป็นเพียงการใช้สลับกัน เพราะนิยมกำหนดฟิลด์ข้อมูลทั้งหมดก่อน แต่ในเดลไฟมีข้อกำหนดเพิ่มคือ ถ้าเป็นยูนิตที่เดลไฟเตรียมให้ ห้ามเรากำหนดสิ่งต่างๆในส่วนบน คือ ต้อง กำหนดในส่วนตัว หรือ ส่วนรวม เท่านั้น แต่ทั้งนี้หมายถึงเราสามารถกำหนดคำว่า ส่วนตัว กับ ส่วนรวม สลับกันต่อไปได้

ฟิลด์ข้อมูลกับพรอพเพอร์ตี้ส์

เป็นข้อกำหนดทางด้านโอโอพีที่ไม่ต้องการให้ทั้งผู้ใช้และผู้เขียนโปรแกรมเข้าถึงฟิลด์ข้อมูลได้โดยตรง ให้ผ่านทางกิจกรรม (ซึ่งเรื่องนี้ในผลิตภัณฑ์ของบอร์แลนด์รุ่นก่อน นำมาใช้บ้างไม่ใช้บ้าง เช่น ให้อ่านค่าได้ แต่ให้ค่าไม่ได้) แต่ที่เราอ่านและให้ค่าพรอพเพอร์ตี้ส์ได้ เพราะพรอพเพอร์ตี้ส์ไม่ใช้ฟิลด์ข้อมูล มีฟิลด์ข้อมูลแยกต่างหากเช่น Left จะมีฟิลด์ข้อมูลเป็น fLeft ตัวพรอพเพอร์ตี้ส์เองซึ่งใช้คำว่า Property
(แทน Procedure หรือ Function) เป็นฟิลด์ที่มีลักษณะพิเศษของตนเอง มีลักษณะเป็นฟังก์ชันที่ให้ค่าได้จึงทำให้อ่านและให้ค่าได้เช่นเดียวกับฟิลด์ข้อมูลและมีการปฏิบัติเช่นเดียวกับฟิลด์กิจกรรม เช่น เมื่อให้ค่าแคปชัน (Caption) จะต้องนำออกแสดง ด้วย หรือเมื่อให้ค่า Left ก็จะต้องมีการเลื่อนหน้าต่างๆหรือคอนโทรล (Control) แล้วแต่เป็น Left ของใคร

อีเวนต์ (Even) เป็นพรอพเพอร์ตี้ส์

ถึงแม้การให้ค่าอีเวนต์จะแตกต่างกับการให้ค่าพรอพเพอร์ตี้ส์ แต่อีเวนต์คือพรอพเพอร์ตี้ส์ จะเห็นว่า เมื่อเราเลือกคอมโปเนนต์และเลือกอีเวนต์ จะเป็นการกำหนดชื่อกิจกรรมอีเวนต์ และเมื่อดับเบิลคลิ๊ก เดลไฟจะเตรียมโครงโปรแกรมให้ เรา ป้อน งานส่วนนี้จะจบเพียงนี้ แต่ที่เดลไฟดำเนินการต่อคือ นำชื่อกิจกรรมอีเวนต์ไปให้แก่อีเวนต์เช่นให้ค่า Button1Click แก่ OnClick เป็นงานอีกส่วนหนึ่ง ซึ่งเราอาจใช้กิจกรรมอีเวนต์นี้ หรือไม่ใช้ (ลบออก) หรือใช้ กิจกรรมอีเวนต์อื่น รวมทั้งอาจให้ค่าใหม่ในโปรแกรมด้วย เช่น

```
Button1.OnClick := Button2Click ;
Button3.OnClick := Button1.OnClick;
```

จะทำให้ทั้งปุ่ม Button1 และ Button3 เปลี่ยนไปใช้กิจกรรมของปุ่ม Button2 และ รวมทั้งสามารถให้ชื่อกิจกรรมที่เราเขียนขึ้นใหม่

ค่าออปเจ็กต์

ตามที่กล่าวมานี้จะได้ว่า ออปเจ็กต์คือออปเจ็กต์ของสิ่งหนึ่ง เช่นเป็นปุ่มปุ่มหนึ่ง และเป็นตัวแทนของสิ่งนั้น ซึ่งในขณะที่เขียนโปรแกรมจะมองเป็นสิ่งนั้น เช่น เป็นปุ่ม Hi ปุ่ม OK เป็นต้น ซึ่งมีเพียงหนึ่งเดียว แต่ออปเจ็กต์คือตัวแปร จึงสามารถให้ค่า ถอกค่าไปไว้ที่อื่นอีกได้ และโดยเฉพาะเมื่อเป็นค่าพอยน์เตอร์ ซึ่งใช้หน่วยความจำเพียง 4 ไบต์ จึงนิยมลอกไปไว้ที่อื่นอีกเพื่อความสะดวกในการเขียนโปรแกรม ดังเช่นเมื่อเรากำหนดคอมโปเนนต์ ใน Form1 เดลไฟจะกำหนดอินสแตนซ์ สกุลของคอมโปเนนต์นั้นให้ในสกุล Tform1 ให้กำเนิดเป็นออปเจ็กต์ แล้วนำค่าออปเจ็กต์นี้ไปเก็บไว้ใน อะเรย์คอมโปเนนต์ แทนที่จะยุบออปเจ็กต์ในสกุล Tform1 เป็นตัวๆ

อีเวนต์ของฟอร์ม

ตามที่กล่าวมานี้ อาจทำให้เข้าใจว่า ในเดลไฟแปลงเมสเสจเป็นอีเวนต์หนึ่งต่อหนึ่ง แต่ที่จริงแล้วเป็นหลายเมสเสจต่อหนึ่งอีเวนต์ หรือหนึ่งเมสเสจต่อหลายอีเวนต์

ดังจะนำอีเวนต์ของฟอร์มตามที่เห็นสมควรมากล่าวถึง อีเวนต์เหล่านี้บางอีเวนต์มีในคอนโทรลด้วย สำหรับที่เป็นของคอนโทรลโดยเฉพาะจะกล่าวเมื่อกล่าวถึงคอนโทรลนั้น และเนื่องจากอีเวนต์อาจเกิดจากเหตุต่างๆเหตุที่นำมาแสดงคือเหตุที่เกิดบ่อย หากต้องการทราบรายละเอียดขอให้ดูใน Help

มีอีเวนต์ของฟอร์มที่เกิดจากเหตุต่างๆดังนี้

1. OnActive เมื่อหน้าต่างนั้นแอคทีฟ
2. OnClick เมื่อผู้ใช้เลือก (ด้วยเม้าส์หรือด้วยคีย์บอร์ด สำหรับเม้าส์เมื่อปล่อยมือ)
3. OnClose เมื่อหน้าต่างจะปิดด้วย Close (ผู้ใช้เลือกรายการ Close หรือในโปรแกรมใช้คำสั่ง Close) ให้เลือกว่าจะปิดหรือไม่ หรือให้หายตัว หรือมินิไมส์
4. OnCloseQuery เมื่อหน้าต่างจะปิดด้วย Close ให้เลือกว่าจะปิดหรือไม่
5. OnDestroy เมื่อหน้าต่างได้ปิดแล้วด้วย Close สำหรับหน้าต่างหลัก หรือด้วยคำสั่ง Release สำหรับหน้าต่างรอง
6. OnDragOver เมื่อเคอร์เซอร์อยู่บนคอนโทรลใด หลังจาก Drag คือคลิ๊กเลื่อนมาจากคอนโทรลต้นทาง
7. OnDragDrop เมื่อผู้ใช้ปล่อยมือจากเม้าส์ในการ Drag และ Drop
8. OnHide เมื่อจะหายตัว
9. OnKeyDown และ OnKeyUp เมื่อผู้ใช้กดคีย์ขาลงและขาขึ้น ให้ค่าคีย์ที่เป็น วิชวลคีย์โค๊ต (Virtual Key Code)
10. OnKeyPress เมื่อผู้ใช้กดคีย์ให้ค่าคีย์เป็นรหัส ASCII คือ Char
11. OnMouseDown และ OnMouseUp เมื่อผู้ใช้คลิ๊กเม้าส์ขาลงและขาขึ้น
12. OnMouseMove เมื่อผู้ใช้เลื่อนเม้าส์ ไม่ว่าจะกดปุ่มหรือไม่
13. OnPaint เมื่อต้องเพนต์ภาพในหน้าต่างใหม่
14. OnResize เมื่อขนาดของหน้าต่างเปลี่ยนไป
15. OnShow เมื่อจะแสดงตัว

ทำไมเราถึงต้องการโอโอพี

สาเหตุที่ต้องพัฒนาการโปรแกรมแบบกำหนดวัตถุเป้าหมายหรือโอโอพีขึ้นมา ก็เนื่องมาจากการค้นพบขีดจำกัดในวิธีการโปรแกรมที่มีอยู่ก่อนหน้านี้ การที่จะทำให้เกิดแนวความคิดที่ว่าโอโอพีทำอะไรได้นั้น เราจำเป็นจะต้องเข้าใจเสียก่อนว่าขีดจำกัดเหล่านี้คืออะไร และเกิดขึ้นมาจากภาษาการโปรแกรมแบบดั้งเดิมได้อย่างไร

ภาษาเชิงกระบวนคำสั่ง

ภาษาปาสคาล ภาษา C ภาษาเบสิก ภาษาฟอร์มแทรนและภาษาอื่นๆเป็นภาษาประเภทภาษาเชิงกระบวนคำสั่งหรือภาษากระบวนงาน (Procedural Languages) นั่นคือ ข้อความสั่ง

(Statement) แต่ละข้อความสั่งที่ปรากฏอยู่ในภาษา จะบอกคอมพิวเตอร์ให้ทำบางสิ่งบางอย่าง เช่น การรับเข้าบางอย่าง บวกตัวเลข หรือหารด้วย 6 แล้ว แสดงผลลัพธ์นั้นออกมา โปรแกรมที่อยู่ในภาษากระบวนงานจะเป็น รายการของคำสั่ง (Instructions) ในกรณีที่โปรแกรมมีขนาดเล็กมากๆ ไม่จำเป็นต้องอาศัยหลักการจัดระบบอื่นๆ (Paradigm) นักโปรแกรมหรือโปรแกรมเมอร์ (Programmer) จะสร้างรายการของคำสั่งขึ้นและคอมพิวเตอร์ก็จะทำตามรายการของคำสั่งนั้น

การแบ่งออกไปเป็นฟังก์ชัน

เมื่อโปรแกรมเริ่มมีขนาดใหญ่ขึ้น รายการของคำสั่งแต่เพียงอย่างเดียวจะเริ่มเทอะทะ นักโปรแกรมที่จะสามารถเข้าใจโปรแกรมที่มีข้อความสั่งนับร้อยๆนั้นมีจำนวนไม่มากนัก ยกเว้นแต่จะได้แบ่งย่อยโปรแกรมออกไปเป็นหน่วยเล็กๆ ด้วยเหตุนี้จึงได้มี การยอมรับหลักการของฟังก์ชัน (Function) เป็นแนวทางในการทำให้โปรแกรมเป็นที่เข้าใจได้ง่ายขึ้นแก่ผู้เขียนโปรแกรม (คำว่าฟังก์ชันเป็นคำที่ใช้อยู่ใน C++ และ C ส่วนในภาษาอื่นๆนั้นอาจเรียกว่าเป็นรูทีนย่อย (Subroutine) หรือ โปรแกรมย่อย (Subprogram) หรือกระบวนงาน (Procedure) เราจะแบ่งย่อยโปรแกรมออกไปเป็นฟังก์ชัน ซึ่งแต่ละฟังก์ชันมีจุดประสงค์ที่ถูกนิยามไว้อย่างชัดเจน (อย่างน้อยที่สุดในทางอุดมคติ) และมีอินเตอร์เฟซ (Interface) กับฟังก์ชันอื่นๆในโปรแกรมที่ได้ถูกนิยามไว้อย่างชัดเจนแล้วเช่นกัน)

ความคิดเกี่ยวกับการแบ่งแยกโปรแกรมออกไปเป็นฟังก์ชัน สามารถที่จะขยายออกไปให้กว้างอีก โดยการจัดกลุ่มฟังก์ชันจำนวนหนึ่งเข้าด้วยกันไปเป็นเอนทิตี (Entity) ขนาดใหญ่ เรียกว่า มอดูล (Module) แต่หลักการนั้นยังคงเหมือนเดิมคือ เป็นการจัดกลุ่มส่วนประกอบที่ปฏิบัติภารกิจเฉพาะอย่าง

การแบ่งโปรแกรมออกเป็นฟังก์ชันและมอดูล เป็นพื้นฐานประการหนึ่งของการโปรแกรมเชิงโครงสร้าง (Structured Programming) ซึ่งเป็นสาขาของการโปรแกรมที่ถูกนิยามไว้อย่างกว้างๆ และมีอิทธิพลต่อการจัดระบบการโปรแกรมมามากกว่าหนึ่งทศวรรษแล้ว

ปัญหาที่เกิดขึ้นกับการโปรแกรมเชิงโครงสร้าง

เมื่อโปรแกรมมีขนาดใหญ่ขึ้นและซับซ้อนเพิ่มมากขึ้น แม้แต่วิธีการโปรแกรมเชิงโครงสร้างเองก็เริ่มที่จะส่อเค้าของ ความยุ่งยาก คุณคงเคยได้ยิน หรือ ได้เข้าไปสัมผัสกับเรื่องเล่าที่น่ากลัวของการพัฒนาโปรแกรม มาบ้างแล้วเช่น โครงการซับซ้อนจนเกินไป ต้องเลื่อนตารางเวลาออกไป มีการเพิ่มเติมนักโปรแกรมมากขึ้นค่าใช้จ่ายเพิ่มสูงขึ้น และตามมาด้วยความล้มเหลว เป็นต้น

เมื่อวิเคราะห์หาสาเหตุที่ทำให้เกิดความล้มเหลวนี้ ได้พบว่ามีจุดบกพร่องปรากฏอยู่ในตัวของแบบแผนของกระบวนงานเอง ดังนั้นไม่ว่าจะมีการใช้วิธีการโปรแกรมเชิงโครงสร้างที่ดีเพียงใดก็ตามแต่โปรแกรมขนาดใหญ่ก็เริ่มที่จะมีความซับซ้อนมากจนเกินไป

ถ้าเช่นนั้นสาเหตุที่ทำให้ภาษากระบวนงานนี้เกิดความล้มเหลวคืออะไร ? เหตุผลที่สำคัญมากที่สุดประการหนึ่งก็ คือบทบาทที่ข้อมูลแสดงออกมา

โอโอพี: กลวิธีหนึ่งในการจัดระบบ

ควรระลึกไว้ในใจเสมอว่า โดยส่วนใหญ่แล้วนั้นการโปรแกรมแบบกำหนดวัตถุเป้าหมายไม่ได้เกี่ยวข้องกับรายละเอียดของการดำเนินการของโปรแกรม แต่จะเกี่ยวข้องกับการจัดระบบโดยรวมของโปรแกรม แต่ละข้อความสั่งของโปรแกรมที่ปรากฏอยู่ใน C++ ส่วนใหญ่แล้วจะคล้ายคลึงกับข้อความสั่งในภาษากระบวนงาน และมีอยู่หลายข้อความสั่งที่เหมือนกันกับข้อความสั่งในภาษา C ตามความเป็นจริงแล้วนั้น ฟังก์ชันสมาชิกทั้งหมดในโปรแกรม C++ อาจคล้ายคลึงกันกับฟังก์ชันกระบวนงานใน C เป็นอย่างมากคุณจะสามารถกำหนดได้ว่า ข้อความสั่งหรือฟังก์ชันหนึ่งเป็นส่วนหนึ่งของโปรแกรมกระบวนงาน C หรือของโปรแกรมแบบกำหนดวัตถุเป้าหมาย C++ ก็ต่อเมื่อคุณพิจารณาสิ่งแวดล้อมโดยรวมเท่านั้น

ลักษณะของภาษาแบบกำหนดวัตถุเป้าหมาย

ในตอนนี้ลองมาดูส่วนประกอบที่สำคัญของภาษาแบบกำหนดวัตถุเป้าหมายในลักษณะทั่วๆไปและใน C++ เป็นการเฉพาะ

วัตถุเป้าหมาย

เมื่อคุณเข้าไปเกี่ยวข้องกับปัญหาการโปรแกรมในภาษาแบบกำหนดวัตถุเป้าหมาย คุณจะไม่ถามอีกต่อไปว่า " จะแบ่งปัญหาออกไปเป็นฟังก์ชันได้อย่างไร? " การพิจารณาในเรื่องของวัตถุเป้าหมายแทนที่จะเป็นในเรื่องของฟังก์ชัน จะส่งผลทำให้ออกแบบโปรแกรมได้ง่ายขึ้นอย่างน่าประหลาด ทั้งนี้ก็เนื่องมาจากการจับคู่กันอย่างใกล้ชิดระหว่างวัตถุเป้าหมายในนัยของการโปรแกรมกับวัตถุในโลกที่เป็นจริง

สิ่งของชนิดใดที่ควรจะมาเป็นวัตถุเป้าหมายในโปรแกรมแบบกำหนดวัตถุเป้าหมาย ?สิ่งที่จะจำกัดคำตอบของปัญหานี้ก็มีเพียงจินตนาการของคุณเท่านั้น ต่อไปนี้เป็นตัวอย่างจำนวนหนึ่งที่จะนำมาเพื่อเริ่มต้นความนึกคิดของคุณ

- วัตถุเป้าหมายเชิงกายภาพ (Physical Objects)
  - รถยนต์ในการจำลองการเคลื่อนตัวของการจราจร
  - ส่วนประกอบไฟฟ้าในโปรแกรมการออกแบบวงจร
  - ประเทศในแบบจำลองทางเศรษฐกิจ
  - เครื่องบินในระบบการควบคุมการจราจรทางอากาศ
- ส่วนประกอบของสภาพแวดล้อมที่เกี่ยวข้องกับผู้ใช้คอมพิวเตอร์
  - วินโดว์
  - เมนู
  - วัตถุกราฟิก (เส้น สี่เหลี่ยมมุมฉาก วงกลม)
  - เมาส์และคีย์บอร์ด
- โครงประกอบการโปรแกรม
  - อะร์เรย์แบบสั่งทำ (Customized Arrays)
  - กองซ้อน (Stacks)
  - รายการโยง (Linked List)
  - รูปต้นไม้แบบทวิภาค (Binary Trees)
- การรวมกลุ่มข้อมูล
  - บัญชีรายการสินค้า
  - แฟ้มบุคลากร
  - พจนานุกรม
  - ตารางของละติจูดและลองจิจูดของเมืองต่างๆในโลก
- ชนิดข้อมูลที่ผู้ใช้นิยาม
  - เวลา
  - มุม
  - จำนวนเชิงซ้อน
  - จุดบนระนาบ
- ส่วนประกอบในเกมคอมพิวเตอร์
  - ปีศาจในเกมเขาวงกต
  - ตำแหน่งในเกมกระดาน
  - สัตว์ในการจำลองทางนิเวศวิทยา
  - ศัตรูและมิตรในเกมผจญภัย

การจับคู่กันระหว่างวัตถุเป้าหมายของการโปรแกรมและวัตถุของโลกที่เป็นจริง ทำให้เกิดการรวมข้อมูลและฟังก์ชันเข้าด้วยกันอย่างพอเหมาะ วัตถุเป้าหมายที่เกิดขึ้นได้ทำให้เกิดการปฏิวัติการออกแบบโปรแกรม การจับคู่กันระหว่าง องค์ประกอบของการโปรแกรมกับรายการที่ถูกจำลองอย่างใกล้ชิดเช่นนั้น จะไม่มีปรากฏอยู่ในภาษากระบวนงาน

คลาส (Class)

ในโอโอพีนั้นเรากล่าวว่า วัตถุเป้าหมายเป็นสมาชิกของคลาส (Members of Classes) ประโยคนี้หมายความว่าอย่างไร ? เพื่อให้เข้าใจมากขึ้นลองมาดูข้อเปรียบเทียบดังนี้ ภาษาคอมพิวเตอร์เกือบทั้งหมดจะมีชนิดข้อมูลแบบในตัว เช่น ใน C++ จะนิยามชนิดข้อมูล Int ซึ่งหมายถึง จำนวนเต็ม (Integer) ไว้เป็นการล่วงหน้า (ดังที่เราจะได้ศึกษาในบทที่ 3) คุณจะสามารถประกาศตัวแปรให้เป็นชนิด จำนวนเต็ม ไว้เป็นการล่วงหน้า (ดังที่เราจะได้ศึกษาในบทที่ 3) คุณจะสามารถประกาศตัวแปรให้เป็นชนิด จำนวนเต็ม ได้มากตามจำนวนที่คุณต้องการให้มีอยู่ในโปรแกรม เช่น

```
int day;
int count;
int divisor;
int answer;
```

ในทำนองเดียวกันคุณสามารถนิยามวัตถุเป้าหมายจำนวนมากให้อยู่ในคลาสเดียวกัน การนิยามคลาส ไม่ได้ทำให้เกิดการสร้างวัตถุเป้าหมายใดๆ ในทำนองเดียวกันกับการที่มีแต่เพียงชนิดจำนวนเต็ม ก็ไม่ได้ทำให้เกิดการสร้างตัวแปรใดๆขึ้น

ดังนั้นคลาสจึงเป็นกลุ่มของวัตถุเป้าหมายที่คล้ายคลึงกัน ความหมายเช่นนี้ตรงกันกับความเข้าใจเกี่ยวกับคำว่า " คลาส " ในทางที่ไม่ใช้วิชาการ ตัวอย่างเช่น นักร้องชื่อดังเช่น ปรินซ์ สติง และมาดอนนา เป็นสมาชิกของคลาสของนักร้องเพลงร็อค เราจะไม่เรียกคนทั่วๆไปว่าเป็น " นักร้องเพลงร็อค " แต่คนที่มีลักษณะเฉพาะและมีชื่อเฉพาะเท่านั้นที่จะเป็นสมาชิกของคลาสนี้ถ้าเขาเหล่านั้นสามารถครอบครองลักษณะเฉพาะบางอย่างไว้

การสืบทอด (Inheritance)

ความคิดเกี่ยวกับคลาสได้ก่อให้เกิดความคิดเกี่ยวกับการสืบทอดขึ้น ในชีวิตประจำวันของเรานั้น เราใช้แนวคิดเกี่ยวกันการแบ่งคลาสออกไปเป็นคลาสย่อย (Subclass) เช่นเราทรจบดีว่าจะแบ่งคลาสของสัตว์ออกไปเป็นสัตว์เลี้ยงลูกด้วยนม สัตว์เลื้อยคลาน แมลง นก ฯลฯ ส่วนคลาสของพาหนะจะแบ่งออกไปเป็นรถยนต์ รถบรรทุก รถประจำทาง และรถจักรยานยนต์

หลักการที่จะนำมาใช้ในการแบ่งเช่นนี้อยู่ตรงที่ คลาสย่อยแต่ละคลาสย่อยจะมีลักษณะร่วมกันบางประการซึ่งเหมือนกันกับคลาสที่เป็นต้นกำเนิด เช่น รถยนต์ รถบรรทุก รถประจำทาง และรถจักรยานยนต์ ทั้งหมดมีล้อและเครื่องยนต์ ซึ่งลักษณะเหล่านี้เป็นลักษณะของพาหนะที่นิยามไว้ นอกเหนือจากลักษณะที่มีร่วมกันกับสมาชิกอื่นๆของคลาสแล้ว แต่ละคลาสย่อยยังมีลักษณะเฉพาะบางอย่างที่เป็นของตัวเองอีกด้วย เช่น รถประจำทางจะมีที่นั่งสำหรับผู้คนจำนวนมาก ในขณะที่รถบรรทุกจะมีเนื้อที่สำหรับบรรทุกสัมภาระที่หนักๆ

ในทำนองเดียวกัน เราสามารถแบ่งคลาสของโอโอพีออกไปเป็นคลาสย่อย ใน C++ เรียกคลาสต้นกำเนิดว่า คลาสฐาน (Base Class) จากคลาสฐานสามารถนิยามคลาสอื่นๆที่มีลักษณะเฉพาะของคลาสฐานร่วมอยู่ พร้อมกับมีการเพิ่มเติมลักษณะของตัวเองร่วมไปด้วย เรียกคลาสที่นิยามขึ้นมาใหม่ว่า คลาสอนุพันธ์ (Derived Classes)

อย่าสันสนระหว่างความสัมพันธ์ของวัตถุเป้าหมายกับคลาสไปด้านหนึ่ง กับความสัมพันธ์ของคลาสฐานกับคลาสอนุพันธ์ในอีกด้านหนึ่ง วัตถุเป้าหมายซึ่งปรากฏอยู่ในหน่วยความจำของคอมพิวเตอร์จะรวบรวมเอกลักษณะเฉพาะที่ชัดเจนของคลาสเอาไว้ ซึ่งคลาสจะเปรียบเสมือนกับเป็นเทมเพลต ส่วนคลาสอนุพันธ์จะสืบทอดลักษณะเฉพาะบางอย่างมาจากคลาสฐาน แต่มีการเพิ่มเติมลักษณะใหม่ของตัวมันเองเข้าไปด้วย

การสืบทอดนี้มีลักษณะที่ค่อนข้างจะคล้ายคลึงกับการใช้ฟังก์ชันเพื่อทำโปรแกรมกระบวนงานแบบดั้งเดิมให้ง่ายขึ้น ถ้าเราพบว่าส่วน (Section) ของโปรแกรมกระบวนงานที่แตกต่างกันสามส่วน ทำในสิ่งเดียวกันเกือบจะชัดแจ้ง เราจะรู้ถึงโอกาสที่จะสกัดส่วนประกอบร่วมของทั้งสามส่วนนี้และนำเอาส่วนประกอบนี้เข้าไปไว้ในฟังก์ชันเดี่ยวๆทั้งสามส่วนของโปรแกรมสามารถเรียกฟังก์ชันนั้นเพื่อทำการดำเนินงานที่เป็นลักษณะร่วม และส่วนเหล่านั้นยังสามารถทำการประมวลที่เป็นของตัวเองได้อีกด้วย ในทำนองเดียวกัน คลาสฐานจะประกอบไปด้วยส่วนประกอบที่พบร่วมกันอยู่ในกลุ่มของคลาสอนุพันธ์ การสืบทอดจะช่วยทำให้โปรแกรมแบบกำหนดวัตถุเป้าหมายมีขนาดสั้นลง และทำให้สัมพันธภาพในหมู่ส่วนประกอบของโปรแกรมกระจ่างชัดขึ้นในลักษณะเดียวกันกับที่ฟังก์ชันกระทำในโปรแกรมกระบวนงาน

คุณลักษณะของการออกแบบออปเจ็กต์ (OOD) คือ

1. ออปเจ็กต์ จะทำหน้าที่จัดการกับสถานะต่างๆของตัวมันและให้บริการกับ ออปเจ็กต์อื่นๆ
2. ออปเจ็กต์ แต่ละตัวจะเป็นอิสระ ทำให้ง่ายในการเปลี่ยนแปลง เพราะสถานะต่างๆ
3. ข้อมูลต่างๆจะถูกบรรจุอยู่ใน ออปเจ็กต์ ทำให้การเปลี่ยนแปลงสิ่งต่างๆภายในออปเจ็กต์ไม่ต้องอ้างถึงออปเจ็กต์ ตัวอื่นๆซึ่งเป็นลักษณะของการแยกอิสระ (Encapsulate)
4. หน้าที่ของแต่ละ ออปเจ็กต์ คือจะแสดงอยู่ในรูปของการรวม การทำงาน (Operation) ที่ออปเจ็กต์ นี้สามารถทำได้หรือสามารถให้บริการได้
5. ไม่มีการแบ่งใช้ข้อมูล (Share Data) ออปเจ็กต์จะติดต่อกันด้วยการทำงาน ต่างๆของแต่ละออปเจ็กต์ซึ่งเป็นการลดการเชื่อมโยงกันของระบบ ซึ่งทำให้ไม่มีการแก้ไขข้อมูลของส่วนที่ใช้ร่วมกัน ซึ่งวิธีดังกล่าวนั้นก็คือการทำ Message Passing
6. ออปเจ็กต์อาจจะกระจายกันอยู่และอาจมีการ Execute ของแต่ละตัวเป็นแบบเรียงตามลำดับ (Sequential) หรือ แบบขนาน (Parallel)

- ภายในออปเจ็กต์ ของตัวมันเอง สถานะ (State) ต่างๆของมันจะมองว่าเป็นการแบ่ง (Share) ชุดของการทำงาน (Function) ต่างๆเป็นแบบครอบคลุม (Global) ภายในตัว ออปเจ็กต์ ของมันเอง

ข้อดีของโอโอดี

1. ระบบออปเจ็กต์ง่ายต่อการบำรุงรักษาเพราะว่า ออปเจ็กต์แต่ละตัวเป็นอิสระ ดังนั้นการทำความเข้าใจและการแก้ไขเปลี่ยนแปลงทำได้ง่ายเพราะว่าเป็น เอนติตีเดี่ยว (Stand-Alone Entity) การเปลี่ยนแปลง ออปเจ็กต์ หรือการเพิ่มการทำงานเข้าไปก็จะไม่มีผลกับออปเจ็กต์อื่นๆของระบบ
2. ออปเจ็กต์ยังมีการนำส่วนต่างๆกลับมาใช้ใหม่ได้อีก เพราะว่ามันเป็นอิสระต่อกัน (Encapsulation) ซึ่งการเป็นอิสระต่อกัน นี้จะเป็นการรวมสถานะต่างๆและการทำงานต่างๆไว้ด้วยกัน
3. ในการออกแบบและพัฒนาโดยใช้ออปเจ็กต์ที่มีการออกแบบไว้ก่อนแล้วจะช่วยลดต้นทุนของการออกแบบ , การเขียนโปรแกรม และ การตรวจสอบความถูกต้อง ซึ่งนำไปใช้เป็นออปเจ็กต์มาตรฐาน (Standard Object)ได้ และช่วยลดการเกิดความเสี่ยงในการพัฒนาซอฟต์แวร์ บางระบบจะเห็นได้ชัดเจนว่าการแม็พ (Map)จาก Real World Entity ไปเป็นระบบออปเจ็กต์ นั้นไม่ต่างกันนัก ทำให้เข้าใจถึงการออกแบบและการดูแลรักษาง่าย

- Object - Oriented Analysis , Design และ Programming ทั้งหมดนี้เป็น Object - Oriented Development
- Object - Oriented Strategy คือการใช้ Object - Oriented Development ทั้งหมดในการพัฒนา Process

การพัฒนาออปเจ็กต์ (Object - Oriented Development)

1. การวิเคราะห์ , การออกแบบ และ การโปรแกรม ซึ่งจะมีความสัมพันธ์กันโดยแต่ละตัวจะแยกกันเป็นลักษณะเฉพาะของตัวมันเอง
2. โอโอดี จะเกี่ยวข้องกับการพัฒนา โมเดลของออปเจ็กต์ ตามขอบเขตที่กำหนด
3. โอโอดีจะเกี่ยวข้องกันการพัฒนาระบบ โมเดลของออปเจ็กต์ เพื่อสร้างตามความต้องการที่ระบุไว้
4. โอโอดีจะเกี่ยวข้องกับการนำ ออปเจ็กต์มาใช้ในการเขียนโปรแกรม แบบออปเจ็กต์ใช้งานได้จริง โดยการใช้ภาษาที่เป็นโอโอพี ซึ่งมันสนับสนุนการทำ ออปเจ็กต์และ จัดคลาสของออปเจ็กต์ และ การสืบสกุล

วิธีการออกแบบออปเจ็กต์ ที่นิยมใช้กัน

1. ระบุออปเจ็กต์ ในระบบถึง แอตทริบิวต์ ต่างๆและการทำงาน ต่างๆ
2. จัด ออปเจ็กต์ให้รวมกันเป็นแบบแผนผัง Hierarchy ซึ่งจะแสดงให้เห็นว่า ออปเจ็กต์ที่มีแต่ละส่วนเป็นส่วนหนึ่งของออปเจ็กต์ อื่นๆเป็นอย่างไร
3. นำออปเจ็กต์ มาสร้างเป็น ไดอะแกรม (Diagrams) ซึ่งจะแสดงให้เห็นถึงว่าออปเจ็กต์ให้บริการแก่ออปเจ็กต์ อื่นๆอย่างไร
4. กำหนดการเชื่อมต่อของออปเจ็กต์

ออปเจ็กต์ , คลาสของออปเจ็กต์ และการสืบสกุล (Object , Object Classes and Inheritance)

1. ออปเจ็กต์คือ เอนติตี ที่มีสถานะ และมีการระบุถึง การทำงาน ต่างๆที่จะทำงานบน สถานะ
2. สถานะ คือ แอตทริบิวต์ ต่างๆของออปเจ็กต์
3. การทำงานต่างๆจะถูกรวมไว้ใน ออปเจ็กต์เพื่อจัดให้บริการแก่ออปเจ็กต์อื่นๆ
4. ออปเจ็กต์ต่างๆจะมีการสร้างตาม คลาสของออปเจ็กต์ที่ได้กำหนดไว้
5. การกำหนดคลาสของออปเจ็กต์เป็นการกำหนดรูปแบบสำหรับออปเจ็กต์ต่างๆซึ่งรวมถึงการแสดงซึ่ง แอตทริบิวต์ ทั้งหมดและ การทำงาน ต่างๆซึ่งรวมกันใน ออปเจ็กต์ ของ คลาส นั้นๆ

โพสเซสของออปเจ็กต์จะเกี่ยวข้องกับการออกแบบ คลาสของออปเจ็กต์เมื่อออกแบบเสร็จแล้วจะนำไปสร้างใช้งาน คลาสของออปเจ็กต์เมื่อออกแบบเสร็จแล้วจะนำไปสร้างใช้งานออปเจ็กต์ต่างๆจะต้องการใช้ คลาส ตามที่ได้กำหนดไว้

การ ติดต่อของ ออปเจ็กต์ (Object Communication)

1. ทำการให้บริการกับออปเจ็กต์อื่นๆที่ขอ (Request) มาและถ้าจำเป็นต้องมีการแลกเปลี่ยนข้อมูล ก็จะคอยให้บริการตามความต้องการในระบบกระจาย (Distributed)การสื่อสารของออปเจ็กต์ สามารถใช้ ข้อความแบบ Text มาสร้างได้โดยตรง ซึ่งจะใช้เพื่อการแลกเปลี่ยนของออปเจ็กต์ ต่างๆเป็นแบบ Message Passing การติดต่อของออปเจ็กต์ มักจะสร้างเป็น โปรแกรมย่อย หรือ การเรียกทำงาน (Function Call) โดยชื่อของบริการนี้ต้องเหมือนกับชื่อของการทำงานในออปเจ็กต์ ที่ได้จัดให้บริการนี้ (ซึ่งชื่อก็คือ Message) โดยจะมีการลอกข้อมูลที่จำเป็นในการ Execute ของการทำงานนี้ และผลลัพธ์ที่ Execute แล้วจะถูกส่งพารามิเตอร์ กลับไป
2. สถานะทั้งหมดที่มีการเปลี่ยนแปลงนั้นสร้างมาจากการนำการทำงาน ต่างๆที่กำหนดไว้ในออปเจ็กต์ มาทำการอินเตอร์เฟสกัน
3. Active Object คือ ออปเจ็กต์ที่สามารถเปลี่ยนแปลงสถานะของมันเองโดยปราศจากการอินเตอร์เฟสกับ การทำงาน อื่นๆ
4. การออกแบบที่ดีควรมีการปิดบังข้อมูลของออปเจ็กต์ต่างๆคือ ไม่ควรที่จะมีการเข้าถึง (Access) ได้จากออปเจ็กต์ภายนอกแอตทริบิวต์ ควรจะมีการเข้าถึงและเปลี่ยนแปลง (Modify) ผ่าน Update Function ที่เหมาะสมของตัวออปเจ็กต์ของมัน เมื่อมีการออกแบบหรือประยุกต์ใช้โพรเซส ต้องไม่มีผลกระทบกับ Object อื่นๆ

การสืบสกุล

1. ออปเจ็กต์ต่างๆที่สร้างขึ้นมาจะได้รับการถ่ายทอดคุณสมบัติตามแอตทริบิวต์ และการทำงานต่างๆของคลาส ของมัน
2. คลาสของออปเจ็กต์ อาจมีการรับรอง แอตทริบิวต์ มาจาก คลาส อื่นๆ (ที่เป็น Super Class ของมัน) แผนผังการสืบสกุล (Inheritance Trees)จะแสดงถึงออปเจ็กต์ต่างๆที่รับเอาคุณสมบัติตามแอตทริบิวต์ และ การทำงานจาก คลาสหลัก (Super Class) ของมัน และอาจมีการเพิ่ม การทำงาน หรือ แอตทริบิวต์ ใหม่ๆแก้ไปเป็นของตัวเอง

3. แผนผังการสืบสกุลที่สร้างขึ้นเป็น คลาสที่ได้รับการถ่ายทอดคุณสมบัติแอตทริบิวต์ การทำงานมาจากคลาสหลักแบบเดี่ยว
4. การสืบสกุลแบบเน็ตเวิร์ก (Inheritance Network หรือ Multiple Inheritance) คือได้รับการถ่ายทอดแอตทริบิวส์มาจาก คลาสโดยตรงมากกว่า 1 ตัว ดังนั้น คลาสที่มี บรรพบุรุษ (Parent) หลายตัวจะมี แอตทริบิวต์และการทำงาน ที่สร้างขึ้นโดยรวมเอาแอตทริบิวต์ และการทำงาน จาก คลาสหลัก ทั้งหมดของมัน
5. ปัญหาอย่างหนึ่งของ การสืบสกุลแบบเน็ตเวิร์ก คือ จะมีชื่อที่ซ้ำกันอยู่ แต่เป็นคนละคลาสหลัก ทางแก้ก็คือต้อง เปลี่ยนชื่อ แอตทริบิวต์หรือการทำงาน นั้นใหม่

การสืบสกุลมี 2 บทบาทที่ใช้ในการพัฒนาออปเจ็กต์

1. เป็นกลไกที่ใช้ในการแยกประเภทของ เอนติตี ใน โมเดลของระบบ
2. เป็นกลไกการนำโค๊ตของโปรแกรม กลับมาใช้ใหม่และยอมให้ทำการเปลี่ยนแปลงออปเจ็กต์ โดยปราศจากผลกระทบที่จะเกิดขึ้นกับส่วนอื่นๆของระบบ

ข้อดีของการสืบสกุล

1. เป็นหลักในการแยกประเภทของเอนติตี
2. เป็นกลไกที่นำ ออปเจ็กต์กลับมาใช้ใหม่ ทั้งในการออกแบบ และการโปรแกรม
3. กราฟของการสืบสกุล (Inheritance Graph) จะทำให้รู้ถึงจากจัดการเกี่ยวกับขอบเขตและเกี่ยวกับระบบ

ปัญหาของการสืบสกุล

1. คลาสของออปเจ็กต์ไม่มีความสมบูรณ์ในตัวเอง คือไม่สามารถทำความเข้าใจด้วยตัวมันเองโดยปราศจากการอ้างถึง คลาสหลัก อื่น
2. นักออกแบบมักมีแนวโน้มที่จะนำ กราฟของการสืบสกุล ที่เคยสร้างไว้แล้วกลับมาใช้ทำการวิเคราะห์ ซึ่งอาจทำให้การออกแบบไม่มีประสิทธิภาพ เนื่องจาก กราฟ นั้นจะบ่งบอกถึงขอบเขตของการใช้งานนั้นๆมากกว่า การที่จะพัฒนาระบบ
3. การวิเคราะห์และออกแบบมีการนำไปสร้างที่แตกต่างกัน ทั้งหมดขณะที่กราฟของการสืบสกุลนั้นอาจจะคล้ายๆกัน จึงทำให้สับสนในการบำรุงรักษา ระบบในอนาคต

การระบุออปเจ็กต์ (Object Identification)

1. การกำหนด ออปเจ็กต์ ต่างๆในระบบเป็นส่วนที่ยุ่งยากของ โอโอดี
2. ไม่มีหลักว่าในการกำหนดออปเจ็กต์นักออกแบบจะต้องใช้ทักษะและประสบการณ์และขอบเขตของระบบในการออกแบบ
3. การกำหนดออปเจ็กต์ เป็นแบบ โพรเซส วนรอบ (Iterative Process) คือต้องฝึกฝนบ่อยๆ

ข้อเสนอแนะในการกำหนดออปเจ็กต์

1. วิเคราะห์ตามหลักไวยกรณ์ของลักษณะภาษาธรรมชาติ คือ ออปเจ็กต์ต่างๆและ แอตทริบิวต์ เป็นนาม การทำงานต่างๆเป็นกริยา
2. การกำหนดให้คิดว่ามันสามารถสัมผัสได้ในขอบเขตของการใช้งาน
3. ใช้ลักษณะของพฤติกรรมในการออกแบบ โดยขั้นแรกต้องเข้าใจถึงพฤติกรรม ทั้งหมดของระบบก่อน หลายๆพฤติกรรมมีการกำหนด อยู่ในส่วนต่างๆของระบบ ซึ่งต้องเข้าใจถึงว่าพฤติกรรมเข้าร่วมกับส่วนใด
4. ใช้การวิเคราะห์แบบ Scenario ขณะที่ในระบบอาจมีหลายๆ Scenario ในการใช้กำหนดและวิเคราะห์แต่ละ Scenario ของนักวิเคราะห์จะเป็นทีมที่ตอบสนองสำหรับการวิเคราะห์ที่จะกำหนดถึงความต้องการของออปเจ็กต์ แอตทริบิวต์ และ การทำงาน เรียกวิธีการวิเคราะห์แบบนี้ว่า CRC Card

การรวมออปเจ็กต์ (Object Aggregation)

การจัดโครงสร้างการรวมกัน (Aggregation Structure)จะแสดงให้เห็นว่าออปเจ็กต์ หนึ่งๆ ประกอบด้วยออปเจ็กต์อื่นๆอย่างไรบ้าง การรวมกันจะอยู่ในความสัมพันธ์ระหว่างออปเจ็กต์ ที่เป็นความสัมพันธ์แบบสแตติก คือในการสร้างออปเจ็กต์นั้น จะมีส่วนของออปเจ็กต์ที่เป็นออปเจ็กต์ย่อย

การออกแบบอินเตอร์เฟสของออปเจ็กต์ (Object Interface Design)

1. เกี่ยวข้องกับการระบุรายละเอียดของการอินเตอร์เฟส ออปเจ็กต์ คือกำหนดชนิด(Type) ของออปเจ็กต์ แอตทริบิวส์ สัญลักษณ์ และ รูปแบบ ของ การทำงาน
2. นักออกแบบควรหลีกเลี่ยงการออกแบบอินเตอร์เฟส คือ ซ่อนข้อมูลเอาไว้และจะใช้ ออปเจ็กต์ การทำงาน จัดการเข้าถึง และ เปลี่ยนแปลง ข้อมูลส่วนนี้
3. การทำงานที่ควรจัดให้มีเสมอคือ การจัดเก็บและนำข้อมูลของแอตทริบิวต์ ออกมา

การปรับปรุงการออกแบบ (Design Evolution)

1. ข้อมูลที่ซ่อนไว้ในออปเจ็กต์เมื่อเปลี่ยนแปลงออปเจ็กต์ด้วยไม่มีผลกระทบกับออปเจ็กต์ อื่นๆ
2. ออปเจ็กต์เป็นแบบ loosely Coupled คือ ออปเจ็กต์ ใหม่ๆจะไม่มีผลกระทบกับระบบ
3. ออปเจ็กต์จะมีลักษณะที่ Robustness คือแข็งแกร่ง ทนทาน เช่นการทำการตรวจสอบภาวะมลพิษ เพิ่มแก้ไขในแต่ละสถานีอากาศ เป็นการวัดคุณภาพของอากาศ และ คำนวณหา มลพิษที่เปลี่ยนไปในชั้นบรรยากาศ โดยแสดงออกแบบ เพิ่มเติมดังนี้

    1. ออปเจ็กต์ของ คุณภาพอากาศ (Air Quality) จะเป็นส่วนหนึ่งของ สถานีอากาศที่ระดับเดียวกับอากาศในวันนั้นๆ
    2. มีการส่งข้อมูลเกี่ยวกับมลภาวะทางอากาศ (Operation Transmit Pollution Data) เพิ่มเข้าไปในสถานีอากาศ เพื่อทำการส่งข้อมูลมลพิษ ไปยังศูนย์คอมพิวเตอร์ส่วนกลางของสถานีอากาศ ซึ่งจะมีซอฟต์แวร์คอยควบคุมในการทำการแก้ไขการอ่านค่ามลพิษเป็นลักษณะอัตโนมัติ ซึ่งจะทำงานเมื่อระบบเปิดสวิตซ์
    3. ออปเจ็กต์ต่างๆจะแทนด้วยชนิดของมลพิษที่เพิ่มเข้าไปก็คือระดับของ Nitrous Oxide และ ของเหลวจากถ่านหินที่ใช้ผลิตพลาสติก ที่จะต้องทำการอัด
    4. ออปเจ็กต์ของฮาร์ดแวร์ควบคุม (Hardware Control Object) คือ มาตรวัดคุณภาพอากาศ (Air Quality Meter)จะต้องเพิ่มเข้าไปใน ออปเจ็กต์ย่อยของคุณภาพอากาศ มีแอตทริบิวต์ แทนแต่ละประเภทของการวัด

ส่วนกลางของระบบสถานีอากาศ จะไม่ต้องเปลี่ยนแปลง ซอฟต์แวร์ ดังนั้นการเพิ่มเติมส่วนนี้เข้าไปไม่ได้มีผลกระทบกับข้อมูลอากาศ ที่ใช้ร่วมกัน ดังนั้นข้อมูลก็จะอยู่ในลักษณะของการแยกอิสระในออปเจ็กต์ ต่างๆจะไม่มีผลกับการออกแบบ เพิ่มเติม

ชุดของออปเจ็กต์ (Concurrent Object)

การสร้างเป็น โปรแกรมย่อย หรือ การเรียกใช้งาน นั้น จะใช้วิธีการแบบ การส่งผ่านข้อความ ส่วนประกอบของ ชุดของออปเจ็กต์ มี 2 ส่วน คือ

1. Passive Object มีการขนานกันของโพรเซส กับชุด ที่เหมือนกันในการกำหนด ออปเจ็กต์ การทำงาน ถ้ามีการเรียก(Call) ไปยังโพรเซสที่ใช้สำหรับประมวลผลออปเจ็กต์ โพรเซสนี้ก็จะหยุดการประมวลผลตัวมันเอง

2. Active Object คือ สถานะของออปเจ็กต์ อาจมีการเปลี่ยนแปลงโดยการทำงานภายในของออปเจ็กต์ มันเอง โพรเซส นี้ ออปเจ็กต์จะทำงานอย่างต่อเนื่องโดยไม่มีการหยุดด้วยตัวของมันเอง

2.3 พื้นฐานการโปรแกรมเดลไฟ (Delphi)

หน้าต่าง

โดยปกติแอพพลิเคชันบนวินโดวส์จะต้องจัดให้มีหน้าต่างอย่างน้อย 1 บาน เป็นหน้าต่างหลัก และ อาจมีหน้าต่างรองได้อีกตามต้องการ ดังที่ในเดลไฟมีหน้าต่างที่เปิดใช้เป็นประจำ 4 บาน เมื่อผู้ใช้ใช้งานหน้าต่างใด จะทำให้หน้าต่างนั้นแอคทีฟ ซึ่งจะแสดงไฮไลต์ที่ไตเติลบาร์

โปรแกรมหลัก

เมื่อเขียนโปรแกรมกับวินโดวส์โดยตรง จะต้องจัดให้ไฟล์ .pas ไฟล์หนึ่งเป็นไฟล์โปรแกรมหลัก มีโปรแกรมของหน้าต่างหลัก ซึ่งถ้ามีหน้าต่างหลายบาน อาจรวมโปรแกรมไว้ในไฟล์โปรแกรมหลักได้ คือไม่ต้องแยกไปไว้ในยูนิต แต่ถ้าโปรแกรมยาวมาก อาจนำบางส่วนไปทำเป็นยูนิต แต่สำหรับเดลไฟจะเตรียมโปรแกรมหลักให้เรียกได้ว่าเป็นการเขียนโปรแกรมแทนเรา โดยจัดให้อยู่ในไฟล์ .dpr และมีไฟล์ .pas เป็นไฟล์ของยูนิตเพื่อเก็บโปรแกรมของหน้าต่าง ตามที่ผู้เขียนโปรแกรมจะป้อนดังตัวอย่างต่างๆ สำหรับไฟล์ .dpr เป็นไบนารีไฟล์ แต่ขอดูในฐานะเท็กซ์ไฟล์ได้โดยเลือกรายการ View/Project Source

และขอกล่าวเสริมว่า ถ้าเป็นการเขียนโปรแกรมโดยตรง ก่อนที่จะเข้าทำงานในวงรอบการรันจะต้อง รีจีสเตอร์ (Register) หน้าต่างเท่ากับเป็นการให้แอดเดรสของโปรแกรมย่อยของหน้าต่าง ซึ่งนิยมตั้งชื่อว่า WndProc แก่วินโดวส์

เมสเสจของวินโดวส์

เมื่อเกิดเหตุ ซึ่งเหตุส่วนมากจะเกิดจากผู้ใช้ เช่น คลิ๊กเม้าส์ กดคีย์บอร์ด หรือเลือกสิ่งต่างๆ วินโดวส์จะส่งเมสเสจ (Message) ให้แก่แอพพลิเคชันเจ้าของหน้าต่างโดยใส่ไว้ในคิว (Application Message Queue) ให้แอพพลิเคชันนั้นถอนจากคิว ส่งให้วินโดวส์ตรวจ แล้วส่งให้วินโดวส์ส่งให้แก่หน้าต่าง ซึ่งวินโดวส์จะส่งให้แก่ WndProc ในเมสเสจ จะแจ้งเหตุที่เกิดขึ้น เรียกเป็นชื่อเมสเสจ เช่น เมื่อผู้ใช้กดคีย์จะได้เมสเสจเป็น wm_KeyDown ซึ่งเป็นค่าคงตัวเลขจำนวนเต็ม ผู้เขียนโปรแกรมใน WndProc จึงต้องนำตัวแปรที่เก็บเมสเสจนี้มาแยกด้วย Case และเขียนโปรแกรมตอบสนองเหตุที่ต้องการ เหตุใดที่ไม่เขียนโปรแกรมตอบสนองก็จะถูกทิ้งไปเรียกเป็นถอนทิ้ง คือคิดจากที่ถอนจากคิว นอกจากนี้ในเมสเสจยังมีรายละเอียดอื่นอีก เช่น ถ้ากดคีย์ก็จะมีค่าคีย์ ถ้าคลิ๊กเม้าส์ก็จะมีตำแหน่งที่คลิ๊กเม้าส์

ด้วยวิธีนี้ทำให้วินโดวส์สามารถรองรับการรันโปรแกรมในแบบการทำงานหลายอย่าง

(Multi Tasking) ให้ผู้ใช้สามารถเปิดใช้แอพพลิเคชันได้พร้อมกันหลายๆแอพพลิเคชัน (ถึงแม้จะไม่ใช้ Multi Tasking ที่แท้จริง) และทำให้การเขียนโปรแกรมกับวินโดวส์มีลักษณะพิเศษ 2 ประการ คือ

1. เป็นการเขียนโปรแกรมให้แก่หน้าต่าง โดยไม่เกี่ยวกับโปรแกรมหลัก ซึ่งมีหน้าที่แต่เพียงให้กำเนิดหน้าต่าง และถอนเมสเสจจากคิว
2. เป็นการเขียนโปรแกรมขับเหตุ (Event Driven Programming) และขับเหตุเท่านั้น

เมสเสจของใคร

ในเมสเสจ นอกจากระบุเหตุที่เกิดพร้อมรายละเอียดแล้ว ถ้าเป็นเหตุที่เกิดแก่คอนโทรล จะแจ้งด้วยว่าเป็นเหตุที่เกิดแก่คอนโทรลตัวใด เพื่อให้ผู้เขียนโปรแกรมแยกได้ และตามที่กล่าวแล้วว่าเมสเสจจะเป็นเมสเสจ ของหน้าต่าง คือส่งให้แก่หน้าต่าง การเขียนโปรแกรมจึงเป็นการเขียนโปรแกรมของหน้าต่าง ซึ่งโดยเฉพาะเมื่อเขียนโปรแกรมในแบบโอโอพี ผู้เขียนโปรแกรมมักทำตัวเป็นหน้าต่าง ซึ่งจะมีรูปแบบของการเขียนโปรแกรมตอบรับเมสเสจ เมื่อมีเหตุเกิดขึ้นที่คอนโทรล มีลักษณะเป็นการรับเมสเสจ จากคอนโทรล จึงมักจะมองคอนโทรลนั้นเป็นคอนโทรลต้นเหตุเป็นผู้ส่งเมสเสจ

อีเวนต์ของเดลไฟ

เพื่อให้สามารถโปรแกรมด้วยการกำหนดคอมโปเนนต์ได้ เดลไฟจึงเป็นผู้แยกเมสเสจแล้วทำเป็นอีเวนต์ (Event) ในความหมายที่หมายตรงถึงเหตุ เช่น OnKeyDown หมายถึงเมื่อผู้ใช้กดคีย์ แต่ที่จริงคือ เป็นอีเวนต์อันเกิดจากการที่ผู้ใช้กดคีย์ (ทำให้วินโดวส์ส่งเมสเสจ WmKeyDown ซึ่งในเดลไฟได้แปลงเป็นอีเวนต์ OnKeyDown) ให้เรานำมากำหนดเป็นกิจกรรมอีเวนต์คือเป็นกิจกรรมสนองตอบอีเวนต์โดยจะต้องระบุว่าต้องการให้เป็นกิจกรรมอีเวนต์ของใคร ดังที่ให้เรากำหนดกิจกรรมอีเวนต์โดยเลือกฟอร์มหรือคอนโทรลที่ต้องการแล้วเลือกอีเวนต์(หรือกลับกัน) จะได้เป็นกิจกรรมอีเวนต์ของสิ่งนั้นของอีเวนต์นั้น

Sender หมายถึง ผู้ส่งซึ่งก็คือคอนโทรลต้นเหตุดังกล่าวมา คือ Form1 และ Button1

เมื่อมองตามนี้จะไม่เห็นประโยชน์ของ Sender และในโปรแกรมตัวอย่างที่ผ่านมาไม่ได้นำมาใช้ จะใช้ต่อเมื่อกำหนดกิจกรรมอีเวนต์เป็นกิจกรรมร่วม เพื่อใช้กับคอนโทรลหลายตัว จึงต้องนำค่าใน Sender มาตรวจหาคอนโทรลต้นเหตุ

หลักการเขียนโปรแกรมกับวินโดวส์

ไม่ทราบว่ามีข้อกำหนดไว้ที่ใดหรือไม่ แต่จะเห็นจากแอพพลิเคชันต่างๆและหนังสือเกี่ยวกับการเขียนโปรแกรมกับวินโดวส์ ดูเหมือนจะถือปฏิบัติเป็นแนวทางเดียวกัน คือ

1. สิ่งใดที่ให้ผู้ใช้ทำด้วยเม้าส์ได้ จะต้องให้ทำด้วยคีย์บอร์ดได้ด้วย เช่น
   - ใช้คีย์ด่วน
   - ใช้คีย์ Tab กับ Enter
   - ให้เลือกจากเมนู

   โปรแกรมในหนังสือนี้ไม่ถือหลักตามนี้ เพราะเห็นว่าเป็นเพียงโปรแกรมตัวอย่าง
2. เมื่อผู้ใด (โปรแกรมส่วนใด) จองหน่วยความจำหรือให้กำเนิดสิ่งใด จะต้องเป็นผู้ยกเลิก
3. เมื่อจะขอสิ่งใดจากวินโดวส์ จะต้องส่งตัวแปรที่มีขนาดเพียงพอไปรับ และเมื่อให้ค่าสิ่งใดแก่วินโดวส์ วินโดวส์จะถอกไว้

สองเรื่องหลังนี้เป็นเพียงเรื่องที่จะต้องปฏิบัติ แต่ในเดลไฟได้เตรียมการไว้ ให้เราเขียนโปรแกรม เหมือนการเขียนโปรแกรมด้วยภาษาปาสคาล ตามปกติส่วนใหญ่แล้วไม่ต้องคำนึงถึงสองเรื่องนี้ หากมีกรณีที่เข้าข่ายจะแจ้งให้ทราบ

ชุดของเดลไฟ

บอร์แลนด์ออกแบบมา 2 ชุด คือ Delphi กับ Delphi Client/Server แตกต่างกันในเรื่องของฐานข้อมูล ส่วนในเรื่องของการเขียนโปรแกรมโดยทั่วไปเช่นที่กล่าวในหนังสือนี้จะเหมือนกัน

ความต้องการทางด้านฮาร์ดแวร์

ในคู่มือของเดลไฟไม่ได้แสดงความต้องการทางด้านฮาร์ดแวร์ มีในเอกสารแจกจ่ายซึ่งกำหนดความต้องการไว้ดังนี้

1. ซีพียู 386 ขึ้นไป (ควรเป็น 486 DX2-66 ขึ้นไป)
2. หน่วยความจำหลักอย่างน้อย 6 เมกะไบต์ (ควรเป็น 8 เมกะไบต์ขึ้นไป)
3. เนื้อที่ในฮาร์ดดิสก์ ตามการติดตั้ง (ควรมีเนื้อที่ว่างอย่างน้อย 80 เมกะไบต์ขึ้นไป) หากใช้ซีพียูที่มีความเร็วต่ำกว่า 486DX2-66 เมื่อดำเนินการใดแล้ว ในหลายกรณีจะต้องคอย ซึ่งอาจสงสัยได้เพราะไม่ได้แสดงเคอร์เซอร์เป็นรูปนาฬิกาทราย

การติดตั้ง

การติดตั้งต้องกระทำบนวินโดวส์ โดยรันโปรแกรม INSTALL.EXE และควรติดตั้งตามคำแนะนำในการติดตั้ง ซึ่งจะเป็นการติดตั้งโดยสมบูรณ์ โดยกด Enter ต่อทุกคำถาม และให้ปฏิบัติตามคำแนะนำที่ให้เปลี่ยนค่า

แอพพลิเคชันในชุดของเดลไฟ

เมื่อติดตั้งเดลไฟ โปรแกรมที่ติดตั้งจะสร้างหน้าต่างกลุ่ม (Group Window) ของเดลไฟภายในประกอบด้วยไอคอนของแอพพลิเคชันต่างๆในชุดของเดลไฟจำนวนหนึ่ง กล่าวคือ

1. เดลไฟ คือตัวเดลไฟ ซึ่งเมื่อต้องการรันเดลไฟให้เปิดใช้ในกรตรวจหาที่ผิด
2. WinSight และ WinSpector เพื่อใช้ช่วยในการตรวจหาที่ผิด

ไอคอนโปรแกรมนอกนั้นเป็นของแอพพลิเคชันทางด้านฐานข้อมูล

หน้าต่างของเดลไฟ

เมื่อเปิดใช้เดลไฟ จะแสดงหน้าต่าง ประกอบด้วยหน้าต่าง 4 บาน คือ

1. หน้าต่างหลัก อยู่ด้านบนสุด แบ่งออกเป็น 4 ส่วน คือ
   - ไตเติลบาร์ ขณะนี้แสดงข้อความว่า Delphi-Project1 ข้อความนี้จะเปลี่ยนไปตามสถานะ
   - เมนูบาร์ แสดงรายการ File ถึง Help
   - สปีดบาร์ อยู่ได้เมนูบาร์ระหว่าง File ถึง View
   - กล่องคอมโปเนนต์ (component palette) อยู่ด้านขวาของสปีดบาร์ แบ่งเป็น 8 หน้า ซ้อนทับกัน ดังที่ทำเป็นรูปที่คั่นหน้า (tab) ให้ชื่อเป็น Standard ถึง Samples เมื่อเลือกหน้าใด จะแสดงแถวของปุ่มคอมโปเนนต์ของหน้านั้น ในการอ้างถึงเพื่อช่วยให้หาปุ่มเหล่านี้ได้ง่าย จะอ้างชื่อหน้ากำกับ เช่น

Additional :: BitBttn

หมายถึงให้เลือกคอมโปเนนต์เป็น BitBttn ในหน้า Additional

กล่องคอมโปเนนต์มีไว้เพื่อให้เลือกคอมโปเนนต์ คือเลือกจากปุ่มคอมโปเนนต์ เช่นเป็น BitBttn ดังกล่าวมา แล้วไปกำหนดในฟอร์ม (ตามที่จะกล่าวต่อไป) เช่น ด้วยการคลิกเม้าส์ได้เป็นคอมโปเนนต์ในฟอร์ม

2. หน้าต่างฟอร์ม (Form1) อยู่ด้านขวาใต้กล่องคอมโปเนนต์ เพื่อการวางรูปแบบของฟอร์ม
3. หน้าต่างยูนิต (UNIT1.PAS) ซ้อนทับกับหน้าต่างฟอร์ม เพื่อการป้อนโปรแกรม
4. หน้าต่าง Object Inspector อยู่ทางด้านซ้ายใต้สปีดบาร์ แบ่งเป็น 3 ส่วนคือ
    4.1 ไตเติลบาร์ แสดงข้อความว่า Object Inspector
    4.2 กรอบรายการออปเจ็กต์ เป็นกรอบคอมโบอยู่ใต้ไตเติลบาร์
    4.3 หน้า Properties และหน้า Events พื้นที่ที่เหลือเป็นของสองหน้านี้ ตามที่ทำที่คั่นหน้าไว้ด้านล่างเมื่อเลือกหน้าใดจะแสดงค่าของหน้านั้น คือ
        - หน้า Properties เพื่อการเลือกให้ค่าพรอพเพอร์ตี้ส์ตามต้องการ
        - หน้า Events เพื่อการกำหนดกิจกรรมอีเวนต์คือกิจกรรมสนองตอบอีเวนต์

คำที่จะใช้

เพื่อให้มีคำที่จะใช้ในการอธิบายต่อไป จะขอนำคำที่จะใช้มากล่าวถึงก่อนคือ

1. เคอร์เซอร์และแคเรต เคอร์เซอร์ (Cursor) คือพอยน์เตอร์ของเม้าส์ ส่วนที่เรียกว่าเคอร์เซอร์เดิมในการเขียนโปรแกรมเรียกว่า แคเรต (Caret) ซึ่งในคู่มือการใช้วินโดวส์เรียกเป็น Insertion Point
2. เม้าส์ (Mouse) ขอให้รวมถึงเคอร์เซอร์ด้วย เช่น เลื่อนเม้าส์ไปที่ หมายถึงให้เลื่อนเม้าส์เพื่อให้เคอร์เซอร์ไปอยู่ที่
3. คลิ๊กเม้าส์ หมายถึงกดปุ่มด้านซ้ายของเม้าส์ 1 ครั้ง ถ้าเป็นด้านขวาจะบ่งว่าเป็นเม้าส์ขวา
4. ดับเบิลคลิ๊กเม้าส์ให้คลิ๊กเม้าส์ 2 ครั้งติดต่อกัน
5. คลิ๊กเลื่อน หมายถึงให้คลิ๊กเม้าส์แต่ยังไม่ปล่อยมือ แล้วเลื่อนเม้าส์ไปปล่อยมือ ณ ตำแหน่งที่กำหนด
6. กรอบ หมายถึงกรอบสี่เหลี่ยม ซึ่งอ้างถึงโดยจุดมุมตรงข้าม 2 จุด
7. ตีกรอบ คลิ๊กเลื่อนจากมุมหนึ่งเป็นการกำหนดตำแหน่ง ไปปล่อยมือที่มุมตรงข้าม เป็นการกำหนดขนาด
8. แดร็กแอนด์ดร็อบ (Drag and Drop) หมายถึงการแดร็ก คือคลิ๊กเลื่อนจากออปเจ็กต์หนึ่งไปดร็อบคือปล่อยมือ ณ อีกออปเจ็กต์หนึ่ง

9. เลือก กระทำโดยคลิ๊กเม้าส์ ณ สิ่งที่ให้เลือก หรือใช้คีย์แท็บ (Tab) หรือคีย์ลูกศร เลื่อนกรอบแท็บ (กรอบเส้นประ) หรือไฮไลต์ไปยังสิ่งที่ต้องการ แล้วกดคีย์ Enter อีกประเด็นหนึ่งคือ เมื่อกล่าวว่าให้เลือก หากอยู่ในสถานะที่เลือกอยู่แล้ว ไม่จำเป็นต้องเลือกซ้ำ
10. การเชคและอันเชค จะใช้กับปุ่มวิทยุ (Radio Button) ซึ่งเมื่อเชคจะแสดงจุดดำตรงกลางและกับปุ่มเชค Check Box) ซึ่งเมื่อเชคจะแสดงเครื่องหมายกากบาท ส่วนการอันเชคหมายถึงยกเลิกการเชคซึ่งจะกระทำได้โดยเลือกปุ่มนั้นจะกลับสถานะ ระหว่าง เชค กับอันเชคทุกครั้งที่เลือก เว้นปุ่มวิทยุซึ่งจะเกี่ยวพันกับเรื่องอื่นอีก
11. การเชคจะใช้กับไดอะล็อกบ็อกซ์หรือหน้าต่างที่ทำหน้าที่เดียวกัน ประกอบด้วยวิธีให้ค่าต่างๆเช่น ป้อนข้อความเลือกจากรายการ และเชคปุ่มต่างๆเป็นต้น ซึ่งเมื่อให้ค่าแล้วเลือกปุ่ม OK (หรือปุ่มอื่นตามที่จะแจ้ง) จะเป็นการเซตค่าไว้ใช้ในโปรแกรม จนกว่าจะเซตค่าใหม่
12. ปุ่มที่หน้าต่าง
    - ซิสเต็มเมนู (System Menu หรือ Control Menu Box) อยู่ด้านซ้าย
    - ปุ่มมินิไมส์ (Minimize) และ แม็กซิไมส์ (Maximize) อยู่ทางด้านขวา
13. คำที่เกี่ยวกับไฟล์
    - ชื่อไดเรคทอรี ประกอบด้วยไดรฟ์และไดเรคทอรีต่างๆจนถึงไดเรคทอรีที่
    - ต้องการ
    - ชื่อไฟล์ ประกอบด้วยชื่อและนามสกุล เช่น myfirst.pas
    - ไวล์การ์ด (Wildcard) เช่น *.pas และ *.* เป็นต้น
    - มาสค์ (Mask) ประกอบด้วยไวล์การ์ดจำนวนหนึ่ง แยกจากกันด้วย ;
14. แอพพลิเคชันและโปรแกรม แอพพลิเคชัน (Application) หมายถึงโปรแกรมที่สมบูรณ์เป็นโปรแกรมใช้งานแล้ว ส่วนโปรแกรมอาจเป็นโปรแกรมที่สมบูรณ์หรือเป็นส่วนหนึ่งของโปรแกรม
15. เปิดใช้แอพพลิเคชัน (Starting the Application) คือการรันโปรแกรม เช่น เปิดใช้จากไอคอนคือ ดับเบิลคลิ๊กที่ไอคอน หรือเลือกไอคอนนั้นแล้วกดคีย์ Enter
16. ฟอร์มกับหน้าต่าง ฟอร์ม คือฟอร์มที่ใช้ในการกำหนดคอมโปเนนต์ ซึ่งเมื่อรันจะเป็นหน้าต่างตามความหมายของวินโดวส์
17. ผู้ใช้ หมายถึงผู้ที่ใช้โปรแกรมในขณะรัน รวมทั้งผู้เขียนโปรแกรมด้วยเมื่ออยู่ในฐานะผู้ใช้ และเมื่ออ้างถึงผู้ใช้ จึงหมายถึงในขณะรันไปในตัว

จากสกุลถึงคอนโทรล

เพื่อความเข้าใจในการเขียนโปรแกรมต่อไป จะขอแจงความสัมพันธ์ของสกุล คอมโปเนนต์ออปเจ็กต์ และคอนโทรล ซึ่งแยกได้เป็น 3 เรื่อง คือ

การตั้งชื่อ

ในเดลไฟจะตั้งชื่อสกุลให้ขึ้นต้นด้วยตัวอักษร T เช่น Etdit เมื่อนำไปทำเป็นคอมโปเนนต์ จะนำชื่อ สกุลมาตัดอักษร T ออกใช้เป็นชื่อคอมโปเนนต์ เช่น Edit และเมื่อนำไปกำหนดในฟอร์ม ได้เป็นออปเจ็กต์ จะให้ชื่อออปเจ็กต์ตามชื่อคอมโปเนนต์ต่อท้ายด้วยหมายเลข เช่น Edit1 หากต่อไป เรากำหนดคอมโปเนนต์ชนิดเดียวกันในฟอร์มชนิดเดียวกันในฟอร์มเดียวกัน จะได้ชื่อออปเจ็กต์ เป็น Edit2 และต่อๆไป

สถานะ

ทางด้านสถานะ จะแจงความสัมพันธ์ได้ดังนี้คือ

1. ในเดลไฟได้สร้างสกุลไว้เป็นจำนวนมาก แล้วนำจำนวนหนึ่งไปทำเป็นคอมโปเนนต์ให้ผู้เขียนโปรแกรมกำหนดในฟอร์ม
2. คอมโปเนนต์ส่วนหนึ่งจะเป็นคอนโทรล (control) ในสายตาของผู้ใช้ คือจะปรากฏตัวในขณะรัน
3. คอนโทรลส่วนหนึ่งมีฐานะเป็นวินโดว์คอนโทรล (windowed control) คือ
    - รับโฟกัสได้
    - มีคอนโทรลอื่นอยู่ภายใน ในฐานะม่กับลูกได้
    - มีค่าแฮนเดิล

ชื่อออปเจ็กต์กับชื่อคอนโทรล

เมื่อกำหนดคอมโปเนนต์ในฟอร์มได้เป็นออปเจ็กต์ มีชื่อออปเจ็กต์ใน Name ซึ่งชื่อนี้จะปรากฏในที่อื่นอีก และมีความสัมพันธ์กับชื่อคอนโทรลดังนี้

1. ออปเจ็กต์ มีชื่ออยู่ในที่ต่างๆดังนี้
    - เป็นค่า Name ให้ป้อนแก้ได้
    - อยู่ในรายการออปเจ็กต์ ในกรอบคอมโบได้ ไตเติลบาร์ของหน้าต่าง ObjectsInspector เป็นอีกแห่งหนึ่งที่จะใช้เลือกคอมโปเนนต์ (เช่นเมื่อบังกัน) และจำแสดงชื่อออปเจ็กต์ร่วมกับชื่อ สกุลที่กรอบอิดิตของกรอบคอมโบ

- เป็นชื่ออินสแตนซ์อยู่ในสกุลของฟอร์ม (เช่น Tform1) ซึ่งเป็นออปเจ็กต์ตัวจริง
- ที่ชื่อกิจกรรมอีเวนต์ (2 แห่ง) ร่วมกับชื่ออีเวนต์เช่น Button1Click
- เมื่อให้ชื่อแก่ Name ชื่อในที่ต่างๆเหล่านี้จะเปลี่ยนตาม (หลังจากกดคีย์ Enter แล้ว) และจะทำให้ชื่อคอนโทรลเปลี่ยนตามด้วย

2. ชื่อคอนโทรล คอนโทรลที่มีชื่อคอนโทรล คือคอนโทรลที่มีค่า Caption หรือ Text และนำออกแสดงที่ตัวคอนโทรลเป็นภาพข้อความ ซึ่งผู้ใช้จะเห็นเป็นชื่อคอนโทรลหรือข้อความ เมื่อให้ชื่อคอนโทรลใหม่ คือให้แก่ Caption หรือ Text จะทำให้ภาพข้อความเปลี่ยนไปด้วย และจะไม่เปลี่ยนตามค่า Name อีกต่อไป

โฟกัส

เมื่อคอนโทรลใดได้รับโฟกัส คอนโทรลนั้นจะเป็นผู้รับเหตุจากคีย์บอร์ด คือถ้าจัดกิจกรรมอีเวนต์เช่น OnKeyPress ให้แก่คอนโทรลทุกตัว (ที่จัดได้) จะมาทำงานที่กิจกรรมอีเวนต์ของคอนโทรลที่ได้รับโฟกัส เช่น ถ้า Edit1 ได้รับโฟกัส ก็จะมาทำงานที่ Edit1KeyPress ไม่ใช่ Edit2KeyPress เมื่อคอนโทรลใดได้รับโฟกัส จะแสดงเป็นกรอบเส้นประเล็กๆ แต่ถ้าเป็นกรอบอิดิตจะแสดงแคเรตกะพริบ ซึ่งจะขอรวมเรียกเป็นกรอบแท็บ เพราะสามารถใช้คีย์แท็บ เลื่อนกรอบนี้ คือ โฟกัสไปยังคอนโทรลอื่นได้

การเลื่อนกรอบแท็บจะเป็นไปตามลำดับที่ของแท็บสต็อบ (Tab Stop) ซึ่งค่านี้จะปรากฏที่ค่าพรอพเพอร์ตี้ส์ TabOrder ตามลำดับที่กำหนดคอนโทรล หรือให้ค่าใหม่ทดลองได้โดยเตรียมโปรแกรม คือเพียงแต่กำหนด คอมโปเนนต์ รันทดสอบแล้วกดคีย์แท็บ จะเห็นว่ากรอบแท็บเลื่อนไปอยู่ที่คอนโทรลถัดไป (และได้รับโฟกัส) ทุกครั้งที่กดคีย์แท็บ ซึ่งถ้าเป็นกรอบอิดิตคือ Edit1 หรือ Memo1 สามารถป้อนข้อความได้ แต่จะไม่เลื่อนเข้าไปใน GroupBox1 และ RadioGroup1 ทั้งที่เป็นวินโดว์คอนโทรล เว้นเมื่อมีวินโดย์คอนโทรลอื่นอยู่ภายในตามที่จะกล่าวในหัวข้อกรอบกลุ่ม ส่วน Label1 และ Panel1 ไม่ใช่วินโดว์คอนโทรล ไม่รับโฟกัส

สำหรับหน้าต่าง เมื่อหน้าต่างใดแอคทีฟหน้าต่างนั้นจะได้รับโฟกัส แต่ถ้ามีคอนโทรลที่รับโฟกัสอยู่ในหน้าต่าง หน้าต่างจะส่งต่อโฟกัสไปให้คอนโทรล เทียบได้รับว่าหน้าต่างนั้นไม่รับโฟกัส

กรอบกลุ่ม

กรอบกลุ่มตามมาตรฐานของวินโดว์คือ Group Box ใช้สำหรับกำหนดคอนโทรลไว้ภายในซึ่งจะทำให้เกิดสถานะแม่กับลูก (ตามที่จะกล่าวในหัวข้อ " สถานะแม่กับลูก ") แต่เดิมการใช้กรอบกลุ่มส่วนใหญ่จะเป็นเพียงการจับกลุ่มปุ่มวิทยุและปุ่มเชคในสายตาของผู้ใช้ แต่ถ้ามีปุ่มวิทยุอยู่ภายใน จะมีความหมายเป็นพิเศษดังจะกล่าวในหัวข้อ " ปุ่มเลือก " ในปัจจุบันนิยมใช้คอนโทรลในหน้าต่างเพิ่มมากขึ้น บทบาทของกรอบกลุ่มก็เพิ่มตาม เช่นใช้ในการแบ่งพื้นที่ของหน้าต่างออกเป็นกรอบเป็นส่วนย่อย หรือเพิ่มพื้นที่ทั้งในทางกว้าง (ScrollBox) และ ในทางลึก (NoteBook) ทำให้กรอบกลุ่มเปรียบเสมือนหน้าต่างย่อยในหน้าต่างใหญ่ (ขอให้มองการใช้งานในแนวทางนี้ด้วย) ลักษณะที่เหมือนกับหน้าต่างอีกเรื่องหนึ่งคือ ถึงตัวมันเองจะอยู่ในสถานะรับโฟกัส แต่จะส่งโฟกัสให้แก่คอนโทรลภายใน ทำให้ดูเหมือนตัวมันไม่รับโฟกัส และที่เป็นพิเศษคือ ผู้ใช้ต้องใช้คีย์ลูกศรในการย้ายกรอบแท็บภายในกรอบกลุ่ม

นอกจากกรอบกลุ่มมาตรฐาน ในเดลไฟได้จัดทำคอนโทรลเสริมมีลักษณะของกรอบกลุ่มอีกดังนี้ โดยจะกล่าวถึงการนำมาใช้เทียบกับ GroupBox (และจะแสดงชื่อโปรแกรมที่แสดงตัวอย่างการใช้ในวงเล็บ) กล่าวคือ

1. Panel จุดประสงค์เพื่อใช้ทำสปีดบาร์ร่วมกับ SpeedButton แต่อาจนำมาใช้แทนกรอบกลุ่ม เมื่อต้องการพื้นที่เต็มกรอบ (โปรแกรม StepBar) ทั้งนี้เพราะ GroupBox ต้องการพื้นที่สำหรับแสดงชื่อกรอบ ถึงแม้จะให้งดแสดง
2. NoteBook มีลักษณะเป็นกรอบ Panelที่ซ้อนทับกันเป็นหน้า (เช่น กลุ่มคอมโปเนนต์) ปกติใช้ร่วมกับ TabSet เป็นที่คั่นหน้า
3. TabbedNoteBook คือ NoteBook ที่มีคั่นหน้าโดยตรงในตัว มีวิธีใช้เช่นเดียวกับ NoteBook
4. ScrollBox มีลักษณะเป็นกรอบ Panel ที่มีสกรอลบาร์

ความสัมพันธ์แม่กับลูก

ความสัมพันธ์แม่กับลูก (Parent and Child) จะเกิดระหว่างหน้าต่างหรือกรอบกลุ่ม ซึ่งจะขอรวมเรียกว่ากรอบแม่กับคอนโทรลซึ่งรวมกรอบกลุ่มด้วย เมื่อกำหนดคอนโทรลในกรอบแม่ จะทำให้เกิดความสัมพันธ์ฉันท์แม่กับลูก คือกรอบแม่กับคอรโทรลลูก ให้ผลดังนี้

1. คอนโทรลลูก จะอ้างถึงตำแหน่งของตนสัมพันธ์กับกรอบแม่ นั่นคือเมื่อย้ายกรอบแม่ คอนโทรลลูกจะตามไปด้วย

2. เมื่อให้กรอบแม่หายตัว คอนโทรลลูกจะหายตัวตาม และเมื่อให้กรอบแม่แสดงตัว จะแสดงตัวตาม
3. เมื่อยุบกรอบแม่ คอนโทรลลูกจะถูกยุบด้วย (และนำมาใช้ไม่ได้อีกทั้งกรอบแม่และคอนโทรลลูก)

แอพพลิเคชัน

เมื่อเราเปิดโปรเจ็กต์ ในเดลไฟจะบ่งใช้ยูนิต Forms ให้ ในยูนิตนี้จะมีออปเจ็กต์ต่างๆซึ่งเราสามารถนำพรอพเพอร์ตี้ส์ กิจกรรม และอีเวนต์มาใช้ได้ และมีในยูนิตอื่นอีก ตามที่จะขอนำบางเรื่องมากล่าวในหัวข้อนี้ ดังนี้

ตัวแทนของแอพพลิเคชันนั้น มีค่าต่างๆดังนี้

1. ค่าพรอพเพอร์ตี้ส์ ได้แก่
   - Icon เป็นไอคอนของแอพพลิเคชัน
   - Name เป็นชื่อที่ให้แสดงเมื่อมินิไมส์ ให้ไว้เป็นชื่อโปรแกรม
2. กิจกรรม ได้แก่
   - CreateForm เพื่อให้กำเนิดหน้าต่าง
   - MessageBox เพื่อแสดงไดอะล็อกบ็อกซ์แสดงข้อความ (ในหนังสือนี้ใช้ของวินโดวส์ จึงไม่มีคำว่า Application กำกับ)
   - Minimize ย่อหน้าต่างเป็นไอคอน เช่นเดียวกับเมื่อเลือกปุ่ม Minimize
   - Restore เปิดหน้าต่างตามเดิม เช่นเดียวกับรายการ Restore
   - Run เป็นวงรอบการรับหลัก ทำหน้าที่ถอนเมสเสจจากคิว ตีความ และแจกจ่าย (ทำให้เกิดเป็นอีเวนต์ต่าง) จนกว่าผู้ใช้จะยุติการรัน ในแอพพลิเคชันหนึ่งๆจะมีวงรอบการรันหลักได้เพียงแห่งเดียว (จึงไม่ใช่กิจกรรมสำหรับให้ใช้)
   - ProcessMessage เป็นวงรอบการรันรอง จะถอนเมสเสจ (รวมทั้งตีความและแจกจ่าย) จนหมดเมสเสจในคิวในขณะนั้น จึงเหมาะสำหรับทำเป็นวงรอบรอเวลา หรือให้ถอนเมสเสจเมื่อต้องการ
3. อีเวนต์ ได้แก่
   - OnIdle จะเกิดอีเวนต์นี้เมื่อมีเมสเสจในคิว ซึ่งจะมีค่าพารามิเตอร์เป็น Done หากให้ค่าเป็น False จำได้รับอีเวนต์นี้ต่อเนื่อง ถึงแม้ไม่มี เมสเสจในคิว จึงเหมาะสำหรับใช้ทำวงรอบการทำงาน หรือการตรวจจับ

- OnMessage จะเกิดอีเวนต์นี้เมื่อมี เมสเสจในคิวและก่อนการแจกจ่าย โดยให้ค่า เมสเสจมาด้วย

จอภาพ (Screen)

ทำหน้าที่เป็นตัวแทนของจอภาพ แต่เป็นเรื่องของแอพพลิเคชันนั้นเท่านั้น ที่จะนำมาใช้มีแต่ค่าพรอพเพอร์ตี้ส์ เพียงอย่างเดียว ได้แก่

- Cursors เป็นอะเรย์ของเคอร์เซอร์ ซึ่งเป็นค่าแฮนเดิล เพื่อนำออกแสดงได้ทันที โดยเก็บเคอร์เซอร์ที่สร้างไว้แล้วในช่อง -1 ถึง -16 ให้เราเก็บในช่อง 1 ขึ้นไป ส่วนช่อง 0 สำหรับเก็บ
- เคอร์เซอร์ในขณะนั้น และจะขอรวมเรียกเคอร์เซอร์ในช่องลบทั้งหลายว่าเคอร์เซอร์มาตรฐาน
- Cursor เป็นเคอร์เซอร์ในขณะนั้น ซึ่งปกติจะใช้ค่าอินเด็กซ์ในอะเรย์ Cursors คือ crHourGlass ซึ่งมีค่าเป็น -11
- Forms เป็นอะเรย์ของฟอร์ม เก็บออปเจ็กต์ของฟอร์ม คือหน้าต่างไปแอพพลิเคชัน
- FormCount เป็นจำนวนหน้าต่าง
- Fonts เก็บรายชื่อฟอนต์ของระบบ

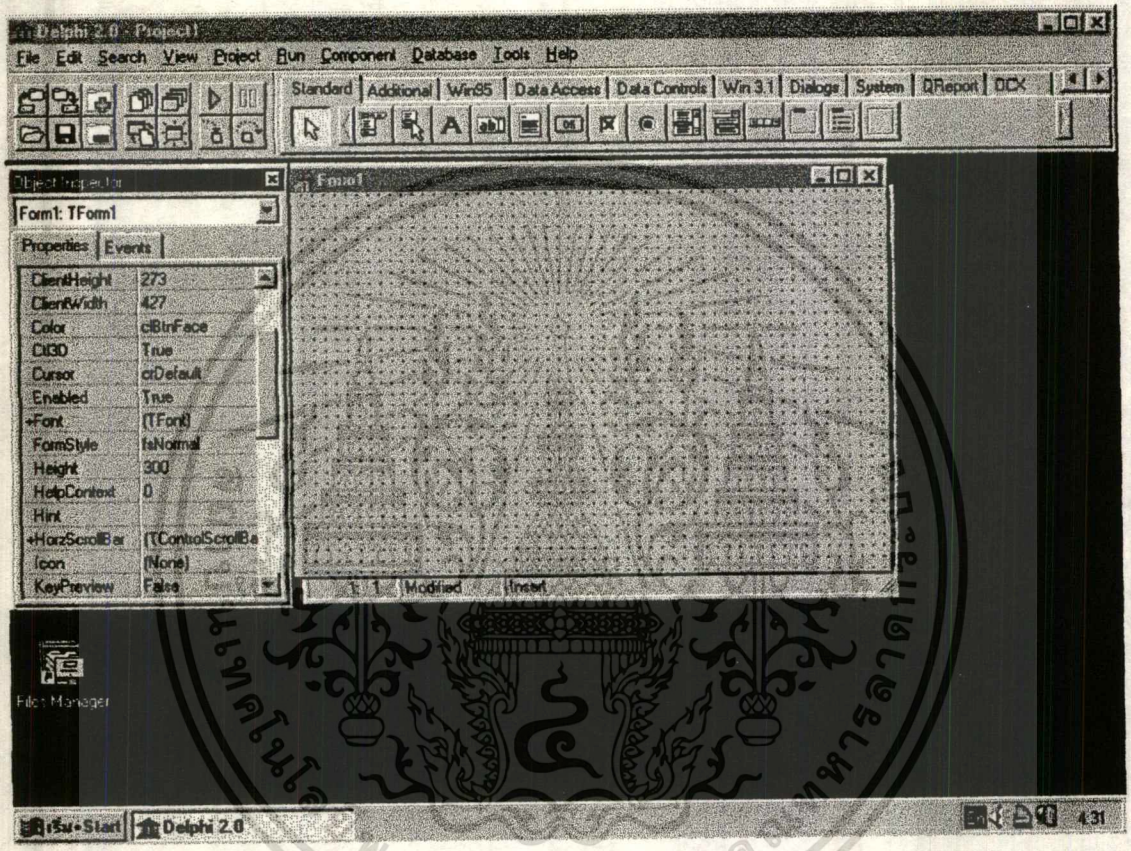
ตัวแปร

นอกจากนี้ยังมีตัวแปรที่กำหนดไว้เป็นโกลบอลในยูนิต System อีกจำนวนหนึ่งจะนำมาใช้ได้แก่

- CmdLine เป็นค่าที่ผู้ใช้ป้อนต่อท้ายชื่อโปรแกรม เช่น เมื่อรันโปรแกรมด้วย File/Run ดังที่ได้นำมาใช้ในโปรแกรม TextEdit
- hinstance เป็นค่า instance ของวินโดวส์ (ไม่ใช่ทางด้านโอโอพี) ซึ่งวินโดวส์จะให้ค่านี้แก่แอพพลิเคชันเมื่อผู้ใช้เปิดใช้ เพื่อใช้กับฟังก์ชันของวินโดวส์ที่ไม่เกี่ยวกับหน้าต่าง เช่น
- การโหลดรีซอร์ส (ของหน้าต่างใช้ค่า Handle ของหน้าต่างนั้น)
- hPrevinst เป็นค่า instance ของก็อบปี้ก่อน เมื่อผู้ใช้เปิดใช้ แอพพลิเคชันนั้นหลายก็อบปี้

รูปของเดลไฟแสดงในรูปที่ 2.5

ลักษณะของโปรแกรมเดลไฟที่ใช้

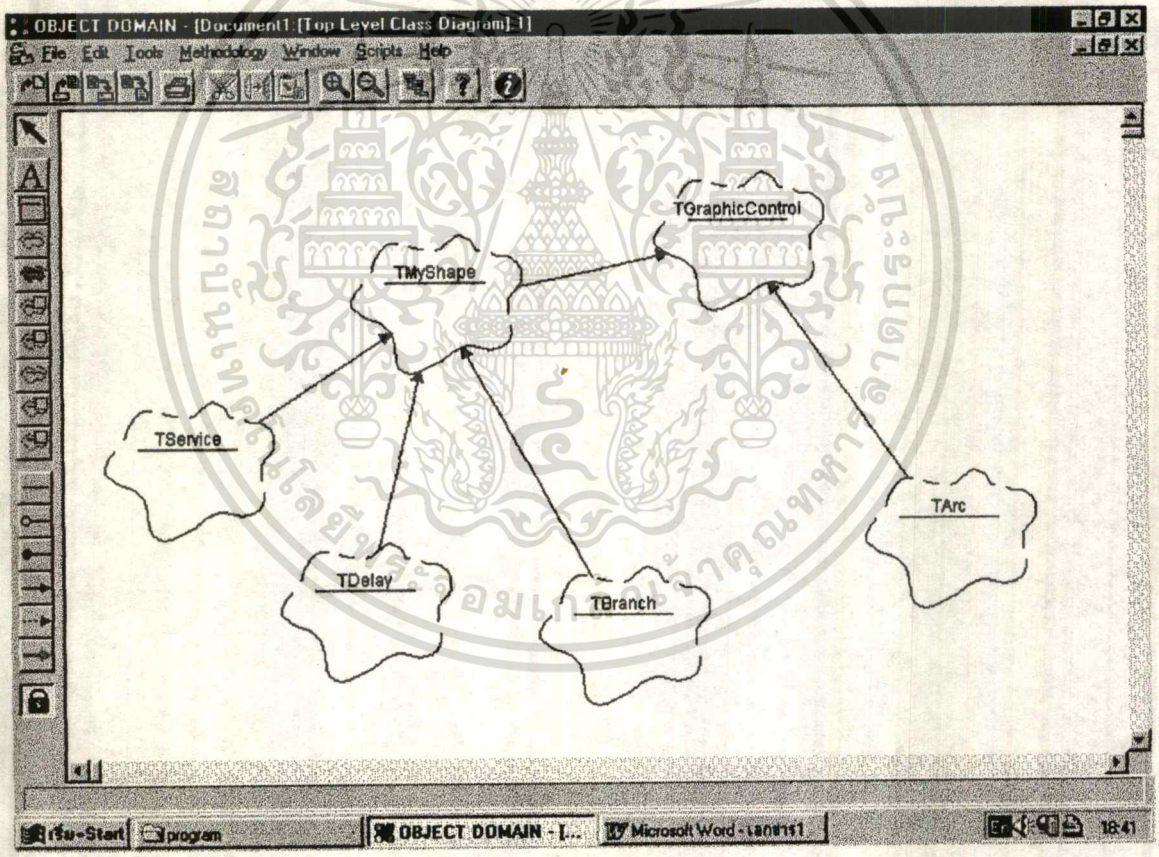


รูปที่ 2.5 ภาพแสดงลักษณะของโปรแกรมเดลไฟที่ใช้

2.4 การใช้งานโปรแกรมอื่นๆ

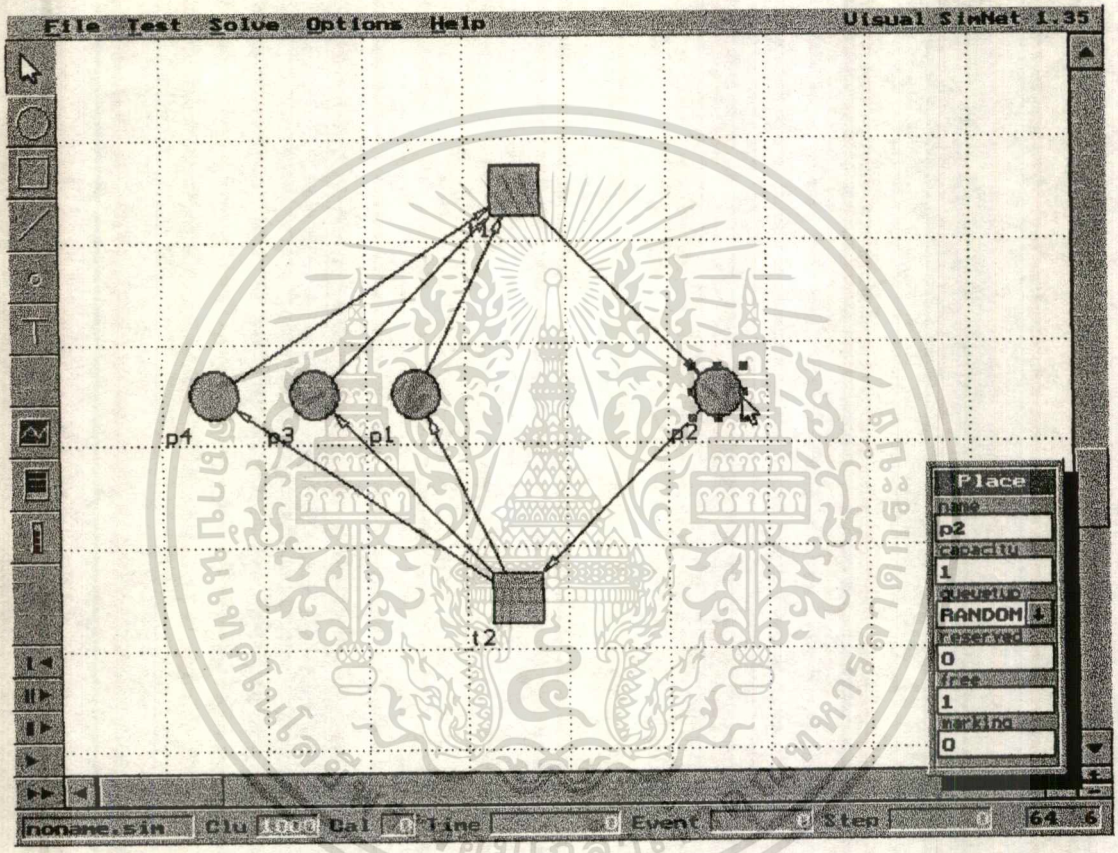
โปรแกรมอื่นๆที่นำมาประยุกต์ใช้ได้แก่ ออปเจ็กต์โดเมน(Object Domain) ดังแสดงในรูปที่ 2.6 โปรแกรมซิมเน็ต (Simnet) ดังแสดงในรูปที่ 2.7

การใช้ออปเจ็กต์โดเมน



รูปที่ 2.6 ภาพแสดงการใช้ทูลออปเจ็กต์โดเมนในการออกแบบคลาสต่างๆ

การทดลองใช้วิชวลซิมเน็ต



รูปที่ 2.7 ภาพแสดงลักษณะของโปรแกรมวิชวลซิมเน็ต

บทที่ 3

การคำนวณและการออกแบบ

3.1 การคำนวณเกี่ยวกับการจัดคิว

พารามิเตอร์ที่ใช้ในการอินพุทของหน่วยบริการเดี่ยว

λ = อัตราการขอเข้ารับบริการ

μ = อัตราการบริการ

m = จำนวนของหน่วยบริการ

การคำนวณเอาท์พุทของหน่วยบริการเดี่ยว

1. ความหนาแน่นของการจราจร

$$\rho = \lambda/(m\mu)$$

2. ความน่าจะเป็นของงานเริ่มต้นในระบบ

$$p_0 = \left[1 + \frac{(m\rho)^m}{m!(1-\rho)} + \sum_{n=1}^{m-1}\frac{(m\rho)^n}{n!}\right]^{-1}$$

3. ความน่าจะเป็นของงานที่ n ในคิว

$$p_n = \begin{cases} p_0 \dfrac{(m\rho)^n}{n!}, & n < m \\ p_0 \dfrac{\rho^n m^m}{m!}, & n \geq m \end{cases}$$

4. ความน่าจะเป็นของคิว

$$\varsigma = P(\geq m\text{jobs}) = \frac{(m\rho)^m}{m!(1-\rho)^{p_0}}$$

5. จำนวนของงานเฉลี่ยในระบบ

$$E[n] = m\rho + \rho\varsigma/(1-\rho)$$

6. จำนวนของงานเฉลี่ยในคิว

$$E[n_q] = \rho\varsigma / (1-\rho)$$

7. เวลาเฉลี่ยที่ใช้ในหน่วยบริการ

$$E[r] = \frac{1}{\mu}\left(1+\frac{\varsigma}{m(1-\rho)}\right)$$

8. เวลาเฉลี่ยที่ใช้รอในคิว

$$E[w] = E[n_q] / \lambda = \varsigma / [m\mu(1-\rho)]$$

พารามิเตอร์ที่ใช้ในการอินพุทของหน่วยบริการเป็นระบบ

v = อัตราการใช้บริการ
s = เวลาที่ใช้ในการให้บริการ
z = เวลาที่ใช้ในการพักข้อมูล
m = จำนวนหน่วยบริการ

การคำนวณเอาท์พุทของหน่วยบริการเป็นระบบ

1. เวลาตอบสนองของหน่วยบริการ

$r_i = s_i(1+q_i)$

2. เวลาตอบสนองของระบบ

$R = \sum r_i v_i$

3. Throughput ของระบบ

$$x = m/(z+R)$$

4. ความยาวของคิว

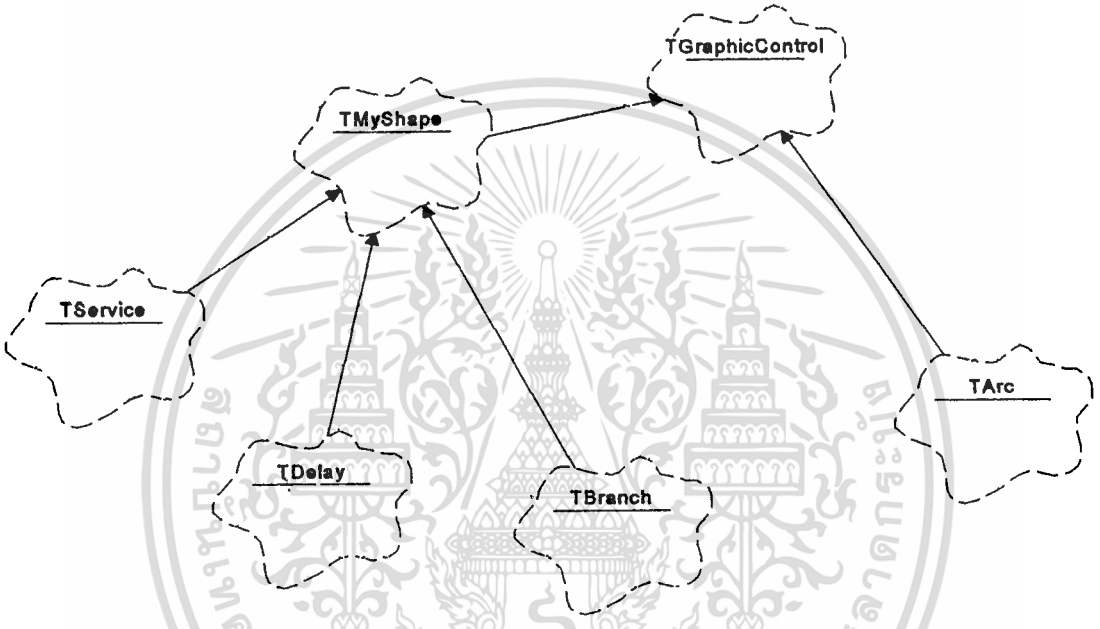
$$q_i = xv_i r_i$$

5. ประสิทธิภาพของระบบ

$$u_i = xs_i v_i$$

3.2 ออปเจ็กต์และพารามิเตอร์ต่างๆทั้งหมดในโปรแกรม

ตัวอย่าง คลาสไดอะแกรม แสดงในรูปที่ 3.1



รูปที่ 3.1 คลาสไดอะแกรม

TMyShape

- TMyShape คือ ออปเจ็กต์ที่ใช้เก็บข้อมูลทางด้านกราฟิกทั่วไป รวมทั้งการเชื่อมต่อระหว่างออปเจ็กต์
- ออปเจ็กต์ TMyShape มีพารามิเตอร์ดังต่อไปนี้

Num	เก็บข้อมูลเป็นแบบ Integer แสดงเลขที่ของแต่ละออปเจ็กต์
X1,Y1	เก็บข้อมูลที่เป็นจุดศูนย์กลางของออปเจ็กต์
Inpx,Inpy	จุดที่เป็นทางเข้าของออปเจ็กต์
Outx,Outy	จุดที่เป็นทางออกของออปเจ็กต์
Direction	เก็บข้อมูลเป็นแบบ Character ซึ่งมีเพียง 4 ค่าเท่านั้นคือ r (right),l(left),u(up),d(down) ซึ่งแสดงทศทางของออปเจ็กต์
ComponentType	เก็บชนิดของออปเจ็กต์แต่ละออปเจ็กต์ เป็น String
InputObj,OutputObj	เก็บออปเจ็กต์ที่เป็นอินพุทและเอาท์พุทของแต่ละออปเจ็กต์

- TMyShape เป็นออปเจ็กต์ที่อนุพันธ์มาจาก TGraphicControl
- การใช้งานมีขั้นตอนดังต่อไปนี้
  1. ต้องมีการประกาศค่าตัวแปร เช่น MyShape1 เป็น TMyShape
  2. ทำการสร้าง (Create) เช่น MyShape1 := TMyShape.Create
  3. การอ้างอิงถึงพารามิเตอร์ภายในคล้ายการอ้างอิงของเรคคอร์ด
     เช่น MyShape1.Name := 'MS1'
  4. การใช้ Method เริ่มต้นด้วยชื่อ ออปเจ็กต์ ตามด้วย '.' และตามด้วยชื่อ Method
     เช่น MyShape1.Refresh
  5. การทำลาย เช่น MyShape1.Free

TService

- TService คือ วิชวลออปเจ็กต์ใช้แสดงภาพ Service และเป็นที่เก็บข้อมูลสำหรับการคำนวณ รวมทั้งข้อมูลเฉพาะของ Service ในการทำงานด้านกราฟิก
- ออปเจ็กต์ TService มีพารามิเตอร์ดังต่อไปนี้

Servicetime	เก็บข้อมูลที่เป็นเวลาที่ใช้ในการบริการ
NoServer	จำนวนของของหน่วยบริการ
NoServeProcess	จำนวนของโพรเซสที่ใช้บริการ
NoWaitProcess	จำนวนของโพรเซสที่คอยในคิว

MaxProcess จำนวนของโพรเซสสูงสุดที่รอในคิว

- TService เป็นออปเจ็กต์ที่อนุพันธ์มาจาก TMyShape
- การใช้งานมีขั้นตอนดังต่อไปนี้
  1. ต้องมีการประกาศค่าตัวแปร เช่น Service1 เป็น TService
  2. ทำการสร้าง (Create) เช่น Service1 := TService.Create
  3. การอ้างอิงถึงพารามิเตอร์ภายในคล้ายการอ้างอิงของเรคคอร์ด
     เช่น Service1.Name := 'SV1'
  4. การใช้ Method เริ่มต้นด้วยชื่อ ออปเจ็กต์ ตามด้วย '.' และตามด้วยชื่อ Method
     เช่น Service1.Refresh
  5. การทำลาย เช่น Service1.Free

TDelay

- TDelay คือ วิชวลออปเจ็กต์ใช้แสดงภาพ Delay และเป็นที่เก็บข้อมูลสำหรับการคำนวณ รวมทั้งข้อมูลเฉพาะของ Delay ในการทำงานด้านกราฟิก
- ออปเจ็กต์ TDelay มีพารามิเตอร์ดังต่อไปนี้

  Delaytime เวลาที่ใช้ในการดีเลย์

  Count จำนวนโพรเซสที่ผ่าน
- TDelay เป็นออปเจ็กต์ที่อนุพันธ์มาจาก TMyShape
- การใช้งานมีขั้นตอนดังต่อไปนี้
  1. ต้องมีการประกาศค่าตัวแปร เช่น Delay1 เป็น TDelay
  2. ทำการสร้าง (Create) เช่น Delay1 := TDelay.Create
  3. การอ้างอิงถึงพารามิเตอร์ภายในคล้ายการอ้างอิงของเรคคอร์ด
     เช่น Delay1.Name := 'D1'
  4. การใช้ Method เริ่มต้นด้วยชื่อ ออปเจ็กต์ ตามด้วย '.' และตามด้วยชื่อ Method
     เช่น Delay1.Refresh
  5. การทำลาย เช่น Delay1.Free

TBranch

- TBranch คือ วิชวลออปเจ็กต์ใช้แสดงภาพ Branch และเป็นที่เก็บข้อมูลสำหรับการคำนวณ รวมทั้งข้อมูลเฉพาะของ Branch ในการทำงานด้านกราฟิก TBranch ไม่เก็บการเชื่อมต่อ (เป็น Nil)
- ออปเจ็กต์ TBranch มีพารามิเตอร์ดังต่อไปนี้

Count	จำนวนโพรเซสที่ผ่าน
Radius	รัศมีของบรานช์

- TBranch เป็นออปเจ็กต์ที่อนุพันธ์มาจาก TMyShape
- การใช้งานมีขั้นตอนดังต่อไปนี้
  1. ต้องมีการประกาศค่าตัวแปร เช่น Branch1 เป็น TBranch
  2. ทำการสร้าง (Create) เช่น Branch 1 := TBranch.Create
  3. การอ้างอิงถึงพารามิเตอร์ภายในคล้ายการอ้างอิงของเรคคอร์ด
     เช่น Branch 1.Name := 'BR1'
  4. การใช้ Method เริ่มต้นด้วยชื่อ ออปเจ็กต์ ตามด้วย '.' และตามด้วยชื่อ Method
     เช่น Branch1.Refresh
  5. การทำลาย เช่น Branch 1.Free

TArc

- TArc เป็นออปเจ็กต์ที่เก็บตำแหน่งที่ใช้ในการแสดงทางด้านกราฟิกของ Arc และ เก็บข้อมูลสำหรับการคำนวณรวมทั้งการเชื่อมต่อระหว่างแต่ละออปเจ็กต์
- ออปเจ็กต์ TArc มีพารามิเตอร์ดังต่อไปนี้

InputObj,OutputObj	เก็บออปเจ็กต์ที่เป็นอินพุทและออปเจ็กต์ที่เป็นเอาท์พุท
Num	เก็บเลขที่ของออปเจ็กต์
ComponentType	ชนิดของคอมโปเนนต์
Prop	เก็บความน่าจะเป็นของออปเจ็กต์ที่ควรจะไหลผ่าน
Inpx,Inpy,Outx,Outy	เก็บจุดอินพุทและเอาท์พุท

- TArc เป็นออปเจ็กต์ที่อนุพันธ์มาจาก TGraphicControl
- การใช้งานมีขั้นตอนดังต่อไปนี้
  1. ต้องมีการประกาศค่าตัวแปร เช่น Arc1 เป็น TArc
  2. ทำการสร้าง (Create) เช่น Arc1 := TArc.Create
  3. การอ้างอิงถึงพารามิเตอร์ภายในคล้ายการอ้างอิงของเรคคอร์ด

เช่น Arc1.Name := 'AR1'

4. การใช้ Method เริ่มต้นด้วยชื่อ ออปเจ็กต์ ตามด้วย '.' และตามด้วยชื่อ Method เช่น Arc1.Minint(A,B)
5. การทำลาย เช่น Arc1.Free

การเชื่อมต่อคลาสต่างๆ

- Input และ Output ของ Service ต้องมีเพียงตัวเดียวเท่านั้น
- Input และ Output ของ Delay ต้องมีเพียงตัวเดียวเท่านั้น
- Input และ Output ของ Branch อาจมีหลายๆตัวก็ได้
- การเชื่อมต่อกันระหว่าง Service , Delay และ Branch กระทำได้โดยใช้ Arc ดังนั้น Input และ Output ของ Service , Delay และ Branch ต้องเป็น Arc เท่านั้น และ Input และ Output ของ Arc เป็น Service , Delay หรือ Branch

การกำหนดอีเวนท์ของแต่ละออปเจ็กต์

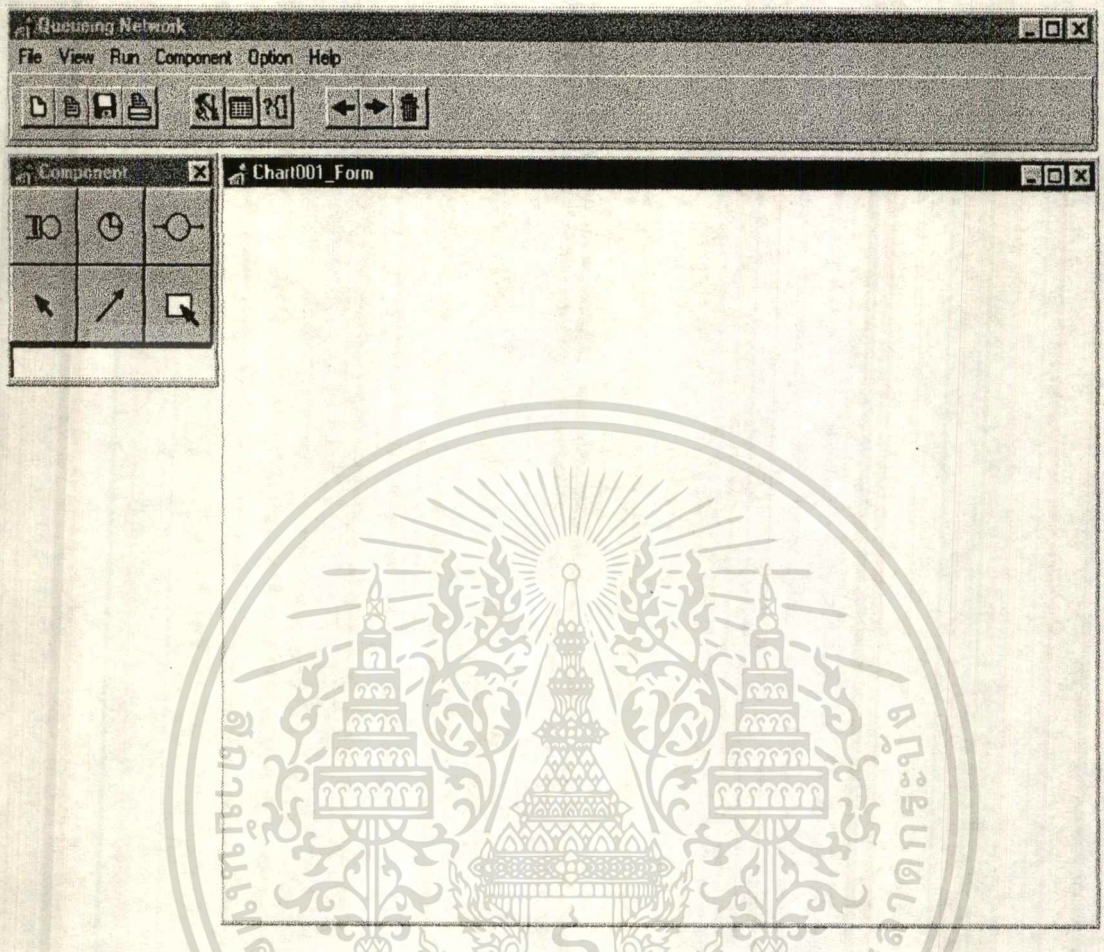
1. เซอร์วิสออปเจ็กต์ เนื่องจากเซอร์วิสออปเจ็กต์เป็นออปเจ็กต์ที่ต้องสามารถเปลี่ยนแปลงค่าได้ดังนั้นจึงต้องมีอีเวนท์ดับเบิ้ลคลิ๊กรวมทั้งต้องสามารถโฟกัสเพื่อเลือกคอมโปเนนต์ให้ได้จึงต้องมีอีเวนท์คลิ๊กและเพื่อใช้ในการทำการเคลื่อนย้ายออปเจ็กต์จึงต้องมีอีเวนท์ออนเมาส์ดาวน์, ออนเมาส์มูฟ และ ออนเมาส์อัพด้วย
2. ดีเลย์ออปเจ็กต์ เนื่องจากดีเลย์ออปเจ็กต์เป็นออปเจ็กต์ที่ต้องสามารถเปลี่ยนแปลงค่าได้ดังนั้นจึงต้องมีอีเวนท์ดับเบิ้ลคลิ๊กรวมทั้งต้องสามารถโฟกัสเพื่อเลือกคอมโปเนนต์ให้ได้จึงต้องมีอีเวนท์คลิ๊กและเพื่อใช้ในการทำการเคลื่อนย้ายออปเจ็กจึงต้องมีอีเวนท์ออนเมาส์ดาวน์, ออนเมาส์มูฟ และ ออนเมาส์อัพด้วย
3. บรานช์ออปเจ็กต์ เนื่องจากบรานช์ออปเจ็กต์เป็นออปเจ็กต์ที่ต้องสามารถเปลี่ยนแปลงค่าได้ดังนั้นจึงต้องมีอีเวนท์ดับเบิ้ลคลิ๊กรวมทั้งต้องสามารถโฟกัสเพื่อเลือกคอมโปเนนต์ให้ได้จึงต้องมีอีเวนท์คลิ๊กและเพื่อใช้ในการทำการเคลื่อนย้ายออปเจ็กจึงต้องมีอีเวนท์ออนเมาส์ดาวน์, ออนเมาส์มูฟ และ ออนเมาส์อัพด้วย
4. อาร์ค ออปเจ็กต์ เนื่องจากอาร์คต้องสามารถโฟกัสได้จึงต้องมีอีเวนท์ออนคลิ๊กและต้องแก้ไขข้อมูลได้จึงต้องมีอีเวนท์ออนดับเบิ้ลคลิ๊กด้วย

- หมายเหตุ เนื่องจากออปเจ็กต์เซอร์วิส,ดีเลย์ และ บรานช์ ใช้อีเวนท์ที่เหมือนกันและทั้งสามออปเจ็กต์เป็นอนุพันธ์มาจากคลาสทีมายออปเจ็กต์เหมือนกันเพราะฉะนั้นจึงกำหนดให้เป็นอีเวนท์ที่ใช้ร่วมกัน

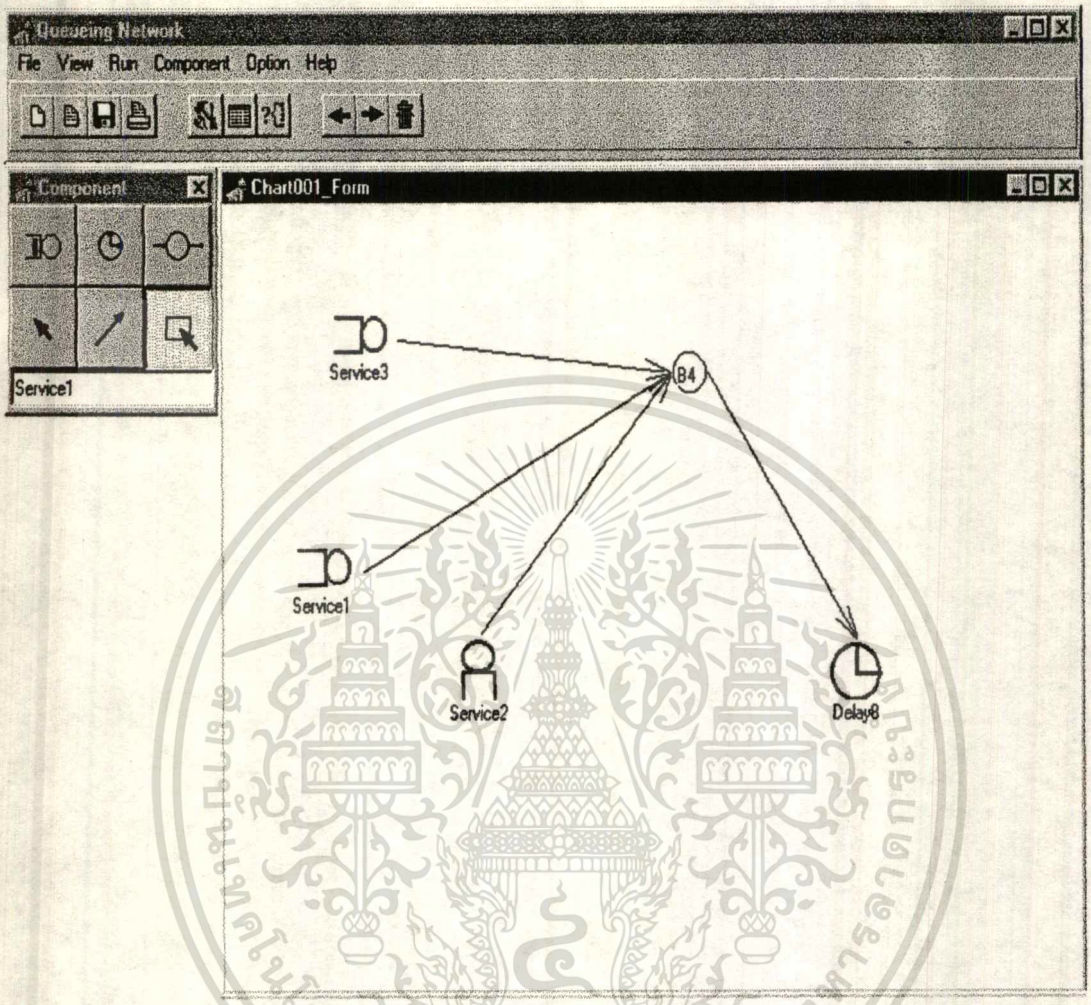
การสร้างกราฟฟิกยูสเซอร์อินเทอร์เฟส

เนื่องจากโปรแกรมที่สร้างใช้เคลไฟเป็นตัวสร้างซึ่งมีการเอื้ออำนวยต่อการทำกราฟฟิกยูสเซอร์อินเตอร์เฟสอยู่แล้วจึงใช้คอมโปเนนต์ที่เคลไฟมีในการสร้าง

ตัวอย่างการใช้งานโปรแกรมอุปกรณ์การจัดคิวบนเน็ตเวิร์ก ดังแสดงในรูปที่ 3.2 และรูปที่ 3.3



รูปที่ 3.2 ภาพแสดงลักษณะของการติดต่อกับผู้ใช้ของโปรแกรมวิเคราะห์ การจัดคิวในเน็ตเวิร์ก

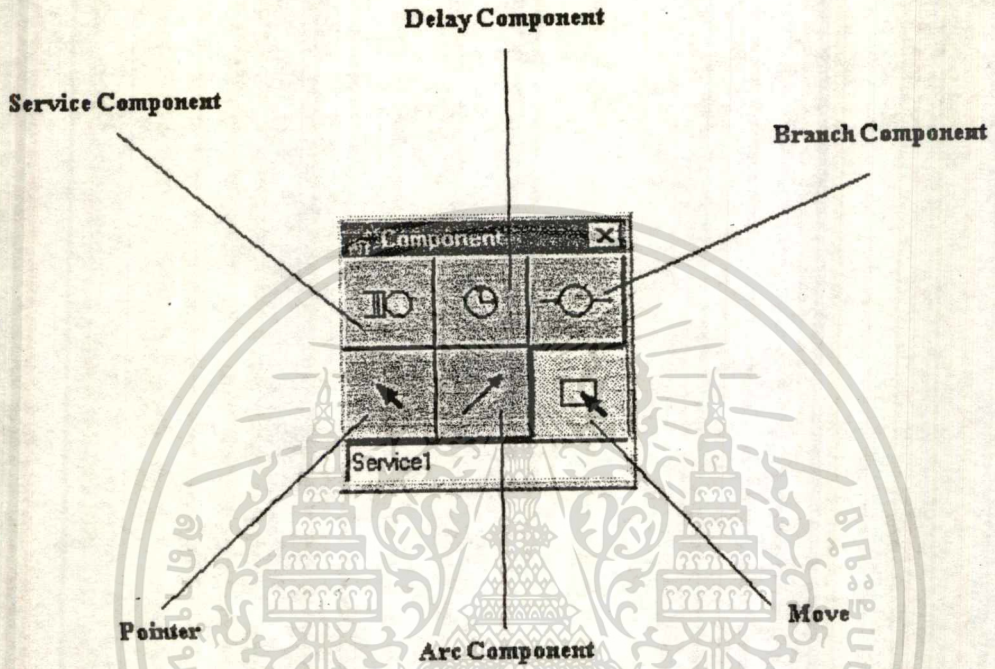


รูปที่ 3.3 ภาพแสดงการใช้งานโปรแกรมวิเคราะห์การจัดคิวบนระบบเน็ตเวิร์ก

บทที่ 4

การทดลองและผล

การใช้ปุ่มต่างๆบนคอมโปเนนต์ฟอร์ม



รูปที่ 4.1 ภาพแสดงการใช้งานคอมโปเนนต์ทูลต่างๆ

คอมโปเนนต์ทูลมีลักษณะเป็นปุ่มทั้งหมดหกปุ่มดังนี้

1.ปุ่มเซอร์วิสใช้ในการให้กำเนิดออปเจ็กต์เซอร์วิสโดยทำการคลิ๊กที่ปุ่มแล้วหลังจากนั้นจึงไปคลิ๊กในฟอร์มในตำแหน่งที่ต้องการ

2.ปุ่มดีเลย์ใช้ในการให้กำเนิดออปเจ็กต์ดีเลย์ โดยทำการคลิ๊กที่ปุ่มแล้วหลังจากนั้นจึงไปคลิ๊กในฟอร์มในตำแหน่งที่ต้องการ

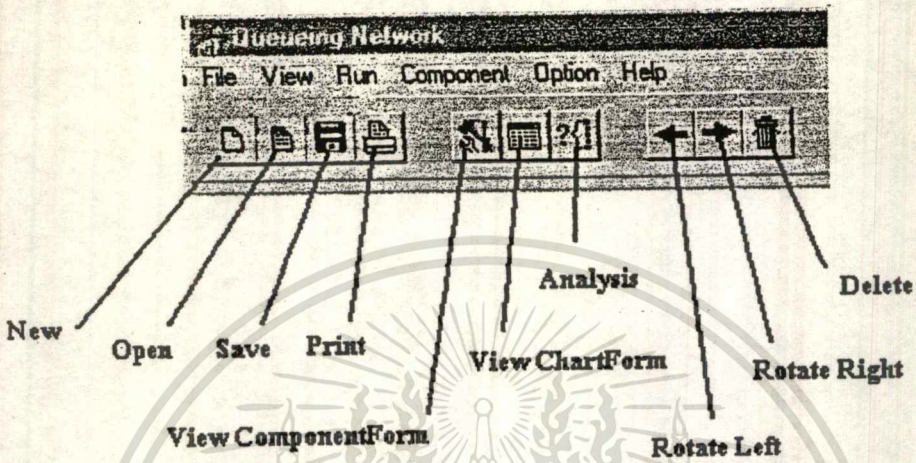
3.ปุ่มบรานช์ใช้ในการให้กำเนิดออปเจ็กต์บรานช์โดยทำการคลิ๊กที่ปุ่มแล้วหลังจากนั้นจึงไปคลิ๊กในฟอร์มในตำแหน่งที่ต้องการ

4.ปุ่มพอยเตอร์ใช้ในการดับเบิลคลิ๊กออปเจ็กต์เพื่อเปลี่ยนแปลงค่าโดยดับเบิ้ลคลิ๊กที่ออปเจกต์ที่ต้องการ

5.ปุ่มอาร์คใช้ในการให้กำเนิดออปเจ็กต์อาร์คโดยคลิกที่ออปเจ็กต์ต้นทางและปลายทางตามลำดับ

6.ปุ่มมูฟว์ใช้ในการเคลื่อนย้ายออปเจ็กต์จากที่หนึ่งไปยังอีกที่หนึ่งโดยใช้วิธีการแดร็กดร็อป

วิธีการใช้ควิ๊กเมนูของแอพพลิเคชั่น

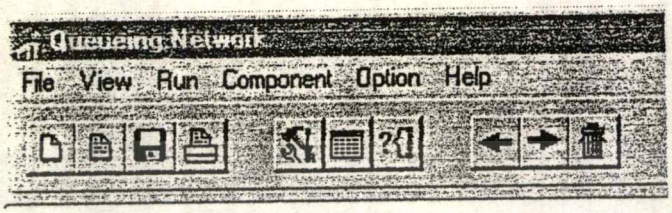


รูปที่ 4.2 ภาพแสดงการใช้ควิ๊กคอมมานด์ต่างๆในโปรแกรม

ควิกคอมมานด์ต่างๆมีรายละเอียดดังนี้

1. New เป็นการเริ่มทำแอพพลิเคชั่นใหม่โดยจะทำการลบแอพพลิเคชั่นเดิมทิ้ง
2. Open เป็นการนำแอพพลิเคชั่นเดิมที่ทำไว้มาทำต่อ
3. Save เป็นการเก็บข้อมูลของแอพพลิเคชั่นปัจจุบันที่ทำอยู่ลงในไฟล์
4. Print เป็นการพิมพ์แอพพลิเคชั่นออกมาทางเครื่องพิมพ์
5. View ComponentForm เป็นการเรียกคอมโปเนนต์ฟอร์มขึ้นมาแสดง
6. View ChartForm เป็นการเรียกชาร์จฟอร์มขึ้นมาแสดง
7. Analysis เป็นการทำการวิเคราะห์ระบบที่เราสร้างขึ้น
8. Rotate Left เป็นการหมุนออปเจ็กต์ที่ถูกเลือกไปทางซ้ายเก้าสิบองศา
9. Rotate Right เป็นการหมุนออปเจ็กต์ที่ถูกเลือกไปทางขวาเก้าสิบองศา
10. Delete เป็นการลบออปเจ็กต์ที่กำลังถูกเลือกอยู่

วิธีการใช้เมนูหลักและเมนูรองต่างๆในแอพพลิเคชั่น



รูปที่ 4.3 ภาพแสดงเมนูหลักที่ใช้ในโปรแกรม

เมนูหลักและเมนูย่อยทั้งหมดมีดังนี้

1.File มีเมนูย่อยดังนี้

New	ใช้เปิดแอพพลิเคชั่นใหม่
Open	ใช้โหลดข้อมูลเดิมมาทำงานต่อ
Save	ใช้เก็บข้อมูลลงไฟล์
Save As	ใช้เก็บข้อมูลลงไฟล์ตามชื่อที่ระบุ
Print	ใช้พิมพ์แอพพลิเคชั่นออกมาทางเครื่องพิมพ์
Exit	ออกจากโปรแกรม

2.View มีเมนูย่อยดังนี้

Component Tool	ใช้แสดงคอมโปเนนต์ฟอร์ม
Chart Form	ใช้แสดงชาร์จฟอร์ม

3.Run มีเมนูย่อยดังนี้

Analysis	ใช้วิเคราะห์แอพพลิเคชั่นที่เราสร้างขึ้น

4.Component มีเมนูย่อยดังนี้

Add - Service	ใช้เพิ่มออปเจ็กต์เซอร์วิสลงในแอพพลิเคชั่น
- Delay	ใช้เพิ่มออปเจ็กต์ดีเลย์ลงในแอพพลิเคชั่น
- Branch	ใช้เพิ่มออปเจ็กต์บรานช์ลงในแอพพลิเคชั่น
- Arc	ใช้เพิ่มออปเจ็กต์อาร์คลงในแอพพลิเคชั่น
Delete	ใช้ลบออปเจ็กต์ที่กำลังถูกเลือกออกจากแอพพลิเคชั่น
Rotate Left	ใช้หมุนออปเจ็กต์ที่กำลังถูกเลือกไปทางซ้ายเก้าสิบองศา
Rotate Right	ใช้หมุนออปเจ็กต์ที่กำลังถูกเลือกไปทางขวาเก้าสิบองศา

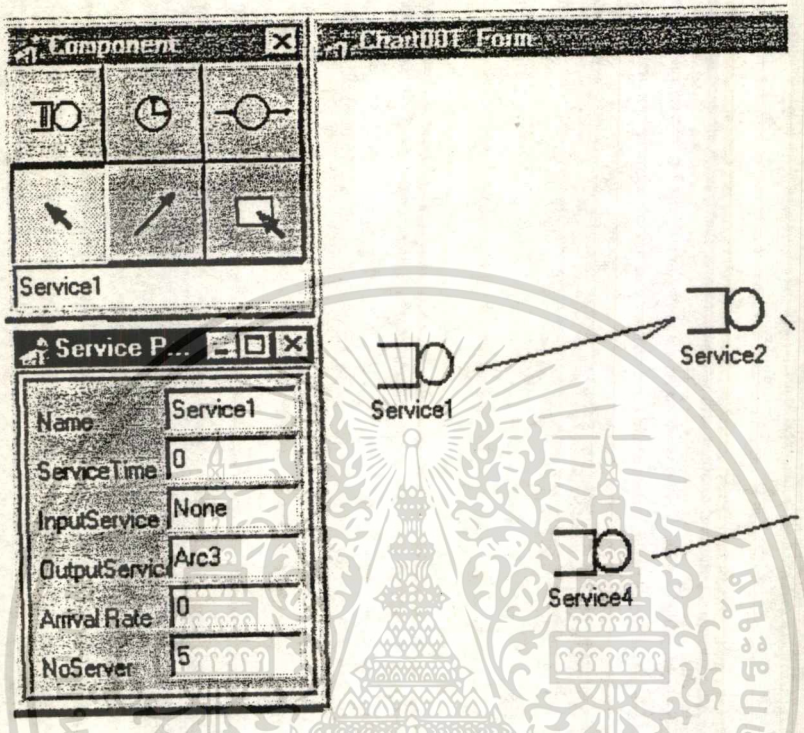
5.Option มีเมนูย่อยดังนี้

Directory	ใช้เก็บค่าดีฟอลท์ไดเรกทอรี

6.Help มีเมนูย่อยดังนี้

About	ใช้บอกเวอร์ชั่นของโปรแกรม

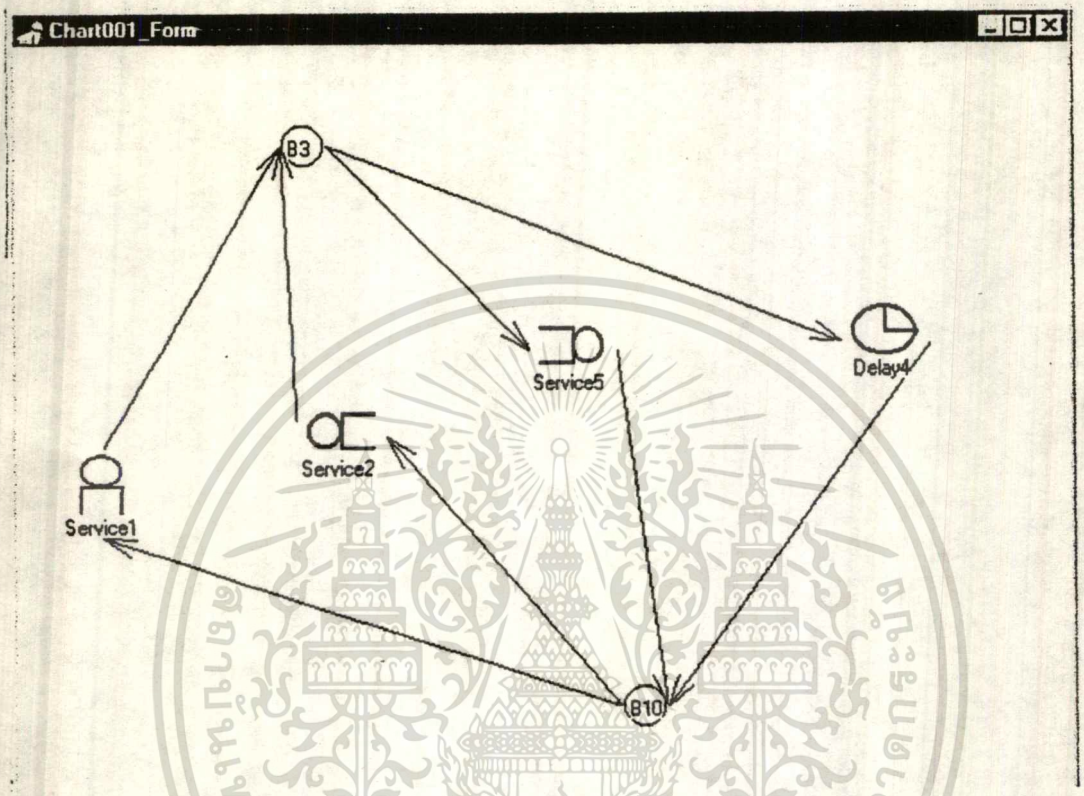
วิธีการแก้ไขข้อมูลของออปเจ็กต์แต่ละออปเจ็กต์



รูปที่ 4.4 ภาพแสดงการใส่ข้อมูลสำหรับออปเจ็กต์ต่างๆ

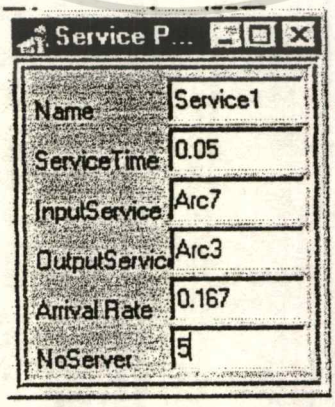
เมื่อเราคลิ๊กที่ปุ่มพอยเตอร์แล้วมาดับเบิ้ลคลิ๊กที่ออปเจ็กต์เซอร์วิส1 จะทำให้เกิดหน้าต่างขึ้นมาเพื่อใช้ในการรับข้อมูลที่ต้องการแก้ไขสำหรับแต่ละออปเจ็กต์ซึ่งข้อมูลที่แก้ไขจะถูกเก็บโดยอัตโนมัติเมื่อทำการปิดหน้าต่างที่เกิดขึ้นมาเท่านั้น ซึ่งข้อมูลเหล่านี้จะถูกนำไปใช้ในการคำนวณและหาค่าผลลัพธ์ออกมาเมื่อเราทำการวิเคราะห์

โมเดลที่ใช้ทดลองคำนวณ

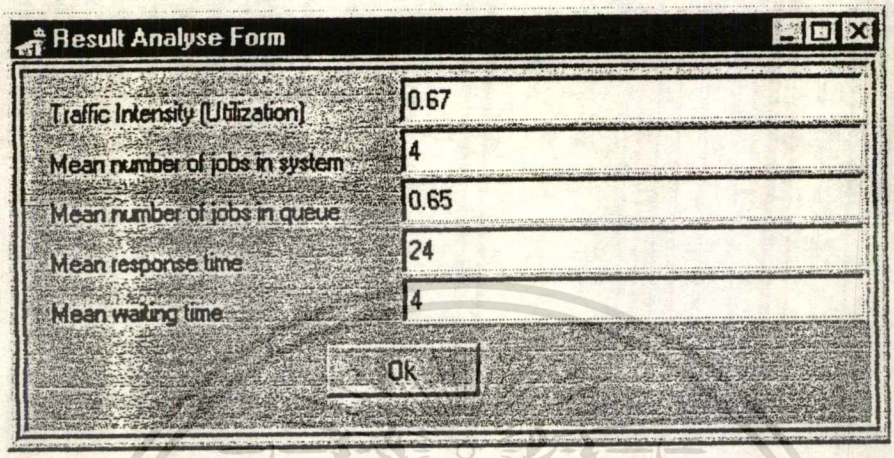


รูปที่ 4.5 ภาพแสดงโมเดลจำลองของแอพพลิเคชั่น

โดยกำหนดให้เซอร์วิส1 มีค่าพารามิเตอร์ดังนี้



ผลลัพธ์จากการวิเคราะห์



รูปที่ 4.6 ภาพแสดงการวิเคราะห์ของโปรแกรม

หลังจากได้มีการสั่งวิเคราะห์ในเมนูรัน-วิเคราะห์แล้วจะได้ผลลัพธ์ดังหน้าต่างข้างบน

บทที่ 5
บทวิจารณ์และสรุป

5.1 ประโยชน์ของโครงงาน

โครงงานอุปกรณ์วิเคราะห์การจัดคิวในเน็ตเวิร์ก ได้มีการพัฒนาซึ่งสามารถนำไปใช้งานได้ มีขีดความสามารถและประโยชน์ต่างๆดังนี้

1. ในงานขนาดใหญ่ที่มีความยุ่งยากซับซ้อน หากใช้คอมพิวเตอร์จะก่อให้เกิดความรวดเร็ว ในการคำนวณและการตรวจสอบ
2. ในระบบเน็ตเวิร์ก หากไม่ได้ออกแบบให้รองรับการใช้งานตั้งแต่ต้นแล้ว จะทำให้ยากในการพัฒนาระบบในภายหลัง
3. ทำให้สามารถเรียนรู้การออกแบบ ออปเจ็กต์โอเรี้ยนเต็ด ได้ซึ่ง ออปเจ็กต์โอเรี้ยนเต็ด เป็นรูปแบบที่ได้รับความนิยมอย่างมากและมีประโยชน์ในการจัดระบบโปรแกรม
4. เพิ่มทักษะในการใช้งานโปรแกรมอื่นๆ เช่น Simnet , Object Domain

5.2 บทวิจารณ์สรุป

โครงงานอุปกรณ์วิเคราะห์การจัดคิวในเน็ตเวิร์ก เป็นการประยุกต์นำเอาคอมพิวเตอร์มาใช้คำนวณโดยนำแนวคิดและหลักการของ การจัดคิวมาใช้ ซึ่งพบว่ามีประสิทธิภาพเพียงพอที่จะใช้งานได้ในระดับหนึ่งคือ

1. ผู้ออกแบบระบบเน็ตเวิร์กจะต้องทราบความสามารถของอุปกรณ์บริการที่จะนำมาใช้ ซึ่งสิ่งนี้มีผลต่อการคำนวณอย่างมาก
2. ทางด้านความเร็วในการคำนวณ จะสูงเมื่อเทียบกับการคำนวณโดยไม่ใช้คอมพิวเตอร์
3. ด้านความสะดวกในการใช้งาน โปรแกรมมีรูปแบบการทำงานบนวินโดวส์ มีไอคอนต่างๆทำให้ผู้ใช้สามารถเข้าใจการทำงานได้ง่ายขึ้น
4. เนื่องจากใช้คอมพิวเตอร์ในการคำนวณ ดังนั้นจึงมีความถูกต้องแม่นยำสูง

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี โดยได้รับความช่วยเหลือและได้รับคำแนะนำจากบุคคลหลายท่าน ได้แก่

- อาจารย์อภิเนตร อูนากูล เป็นที่ปรึกษา , คอยให้คำแนะนำ , ช่วยเสนอแนวทางในการพัฒนาโปรแกรม และ ให้ยืมหนังสือดีๆหลายเล่ม
- อาจารย์อรพินท์ อัสรางชัย ที่ช่วยสอนและแนะนำเกี่ยวกับทฤษฎีของคิว
- พี่ๆ เพื่อนๆ และน้องๆ ที่ให้กำลังใจ ช่วยแปลภาษา ช่วยพิมพ์ และให้ยืมอุปกรณ์ต่างๆ

คณะผู้จัดทำขอแสดงความนับถือและขอขอบพระคุณ ทุกท่านที่ได้ให้ความช่วยเหลือ มา ไว้ ณ โอกาสนี้ด้วย

คณะผู้จัดทำ

หนังสืออ้างอิง

ก. ภาษาไทย

- Delphi (เริ่มเรียนเขียนโปรแกรมบนวินโดวส์ด้วย Delphi 1.0 และ 2.0 ทุกชุด บนวินโดวส์ 3.11 , 95 และ NT , ของบุญเลิศ เอี่ยมทัศนา , สำนักพิมพ์ บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)
- การเขียนโปรแกรมแบบโอโอพีด้วย เทอร์โบและบอร์แลนด์ C++ , โดยโรเบิร์ต ลาฟอเร่, สำนักพิมพ์ บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)

ข. ภาษาอังกฤษ

- Borland Delphi for Windows 95 & Windows NT , Borland International ,Inc.
- The Art of Computer System Performance Analysis , John Wiley & Sons , Inc.
- Data Network , Prentice-Hall , Inc.
- Performance Engineering of Software Systems , Addison - Wesley Publishing Company

หนังสืออ้างอิง

ก. ภาษาไทย

- Delphi (เริ่มเรียนเขียนโปรแกรมบนวินโดวส์ด้วย Delphi 1.0 และ 2.0 ทุกชุด บนวินโดวส์ 3.11 , 95 และ NT , ของบุญเลิศ เอี่ยมทัศนา , สำนักพิมพ์ บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)
- การเขียนโปรแกรมแบบโอโอพีด้วย เทอร์โบและบอร์แลนด์ C++ , โดยโรเบิร์ต ลาฟอเร่, สำนักพิมพ์ บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)

ข. ภาษาอังกฤษ

- Borland Delphi for Windows 95 & Windows NT , Borland International ,Inc.
- The Art of Computer System Performance Analysis , John Wiley & Sons , Inc.
- Data Network , Prentice-Hall , Inc.
- Performance Engineering of Software Systems , Addison - Wesley Publishing Company