รายการอ้างอิง

1. Toshimi Minoura and Sungwoon Choi. "Structural active-object systems fundamentals", Technical Report 93-40-04, Dept. of Computer Science, Oregon State University, 1993.
2. Toshimi Minoura and Sungwoon Choi and Anil Srivastava. "The SAOS Approach to Software Lifecycle Support", Proc. Fifth International Conference on Software Engineering and Knowledge Engineering, pp. 54-61, 1993.
3. Pornsiri Muenchaisri and Toshimi Minoura, "Entity-Relationship Software Development Environment", the proceedings of Technology of Object-Oriented Language and Systems (TOOLS USA'99), Santa Barbara, CA, August 1-5, 1999.
4. Pornsiri Muenchaisri. "Visualization of A Structural Active-Object System with UML", 3rd National Computer Science and Engineering Conference, December 15-17, 1999. Bangkok, Thailand.
5. Chanporn Lappayaporn and Pornsiri Muenchaisri, "A Development of an Editor for Visually Specifying the Behavior of Active Objects", The 4th Annual National Symposium on Computational Science and Engineering (ANSCSE2000), Bangkok, Thailand, March 27-29, 2000.
6. J. Rumbaugh , I. Jacobson and G. Booch . "The Unified Modeling Language Reference Manual", Addison-Wesley,1999.
7. David Harel and Amnon Naamad . "The STATEMATE Semantics of Statecharts", ACM Trans. on Software Engineering Method, 1996.
8. D. Harel and E. Gery, "Executable Object Modeling with Statecharts", Revised February, 1997. To appear in IEEE Computer Early version in Proc. 18th Int. Conf. Soft. Eng., IEEE Press, March 1996.
9. Bruce Powel Douglass, "UML Statecharts", http://www.embedded.com/1999/9901/9901feat1.htm.

ภาคผนวก

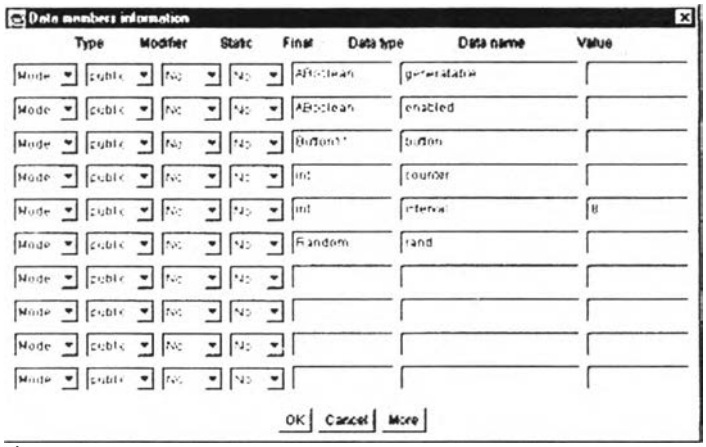
ภาคผนวก ก.

องค์ประกอบของชุดคำสั่งที่สร้างได้โดยอัตโนมัติ

ภาคผนวกนี้แสดงถึงที่มาขององค์ประกอบในชุดคำสั่งของวัตถุพร้อมทำงานที่สร้างจากแผนภาพเอนทิตีและความสัมพันธ์และแผนภาพสเตทชาร์ท โดยอธิบายตามลำดับขององค์ประกอบที่สำคัญต่อไปนี้

ก-1 ส่วนประกาศตัวแปรสมาชิกและค่าคงที่

ตัวแปรสมาชิกและค่าคงที่ซึ่งปรากฏในชุดคำสั่งของวัตถุพร้อมทำงานส่วนหนึ่งมาจากการกำหนดตัวแปรในส่วนกำหนดตัวแปรสมาชิกของบรรณาธิกรสำหรับสร้างชนิดเอนทิตีดังรูปที่ ก-1 เมื่อทำการสร้างชุดคำสั่งรายละเอียดที่กำหนดนี้จะถูกสร้างเป็นส่วนประกาศตัวแปรสมาชิกในชุดคำสั่งของวัตถุพร้อมทำงาน ดังรูปที่ ก-2

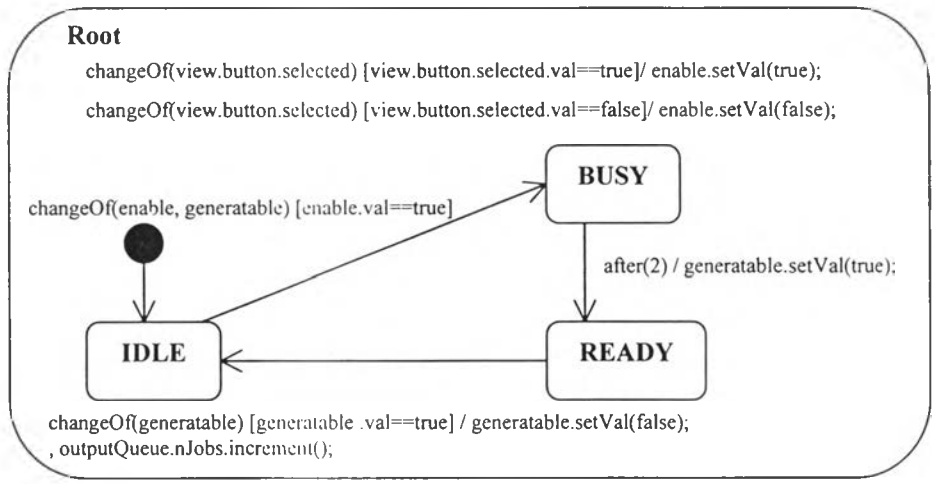


รูปที่ ก-1 ส่วนกำหนดตัวแปรสมาชิกของบรรณาธิกรสำหรับสร้างชนิดเอนทิตี

```
public class Generator  extends  AObject
{
    public ABoolean generatable;
    public ABoolean enabled;
    public int counter;
    public int interval = 8;
    public Random rand;
.....
```

รูปที่ ก-2 ส่วนประกาศตัวแปรสมาชิกที่กำหนดจากบรรณาธิกรสำหรับสร้างชนิดเอนทิตี

นอกจากตัวแปรสมาชิกที่กำหนดจากส่วนกำหนดตัวแปรสมาชิกของบรรณาธิกรสำหรับสร้างชนิดเอนทิตีนี้แล้ว ยังมีตัวแปรสมาชิกที่สร้างโดยตัวสร้างชุดคำสั่งคือ ตัวแปร state ซึ่งเป็นตัวแปรพร้อมทำงานชนิด AInteger และถูกใช้เป็นตัวแปรที่เก็บข้อมูลสถานะปัจจุบันของวัตถุ เมื่อวัตถุมีการเปลี่ยนสถานะจะทำการกำหนดค่าของสถานะใหม่ให้กับตัวแปรนี้ สำหรับสถานะที่แสดงในแผนภาพสเตทชาร์ทจะถูกแปลงเป็นค่าคงที่โดยกำหนดชื่อสถานะเป็นชื่อของค่าคงที่ ดังตัวอย่างของแผนภาพสเตทชาร์ทของตัวสร้างงานในรูปที่ ก-3 ซึ่งสถานะถูกแปลงเป็นค่าคงที่ในชุดคำสั่งในรูปที่ ก-4



รูปที่ ก-3 แผนภาพสเตทชาร์ทของตัวสร้างงาน

```
public class Generator  extends  AObject
{
 …..
public AInteger state = new AInteger();

 static final int IDLE= 0;
 static final int BUSY= 1;          } แปลงจากสถานะในแผนภาพสเตทชาร์ท
 static final int READY= 2;
…..
```

รูปที่ ก-4 ตัวแปรสมาชิกและค่าคงที่ที่สร้างโดยตัวสร้างชุดคำสั่ง

ก-2 ส่วนฟังก์ชันสมาชิก

ฟังก์ชันสมาชิกในชุดคำสั่งส่วนหนึ่งมาจากการแปลงรายละเอียดของข้อความการเปลี่ยนแปลงในแผนภาพสเตทชาร์ทมาเป็นฟังก์ชันสมาชิกดังนี้

1) แปลงชื่อการเปลี่ยนแปลงไปเป็นชื่อฟังก์ชัน
2) แปลงส่วนเงื่อนไขไปเป็นประโยคเงื่อนไขในฟังก์ชัน
3) แปลงส่วนการกระทำเป็นประโยคคำสั่งในฟังก์ชัน

ถ้าการเปลี่ยนแปลงนั้นไม่มีการระบุชื่อไว้ตัวสร้างชุดคำสั่งจะทำการกำหนดให้โดยอัตโนมัติ เช่น ตัวอย่างชุดคำสั่งของตัวสร้างงานในรูปที่ ก-5 ซึ่งแปลงมาจากการเปลี่ยนแปลงในแผนภาพสเตทชาร์ทของตัวสร้างงานรูปที่ ก-3

```
public class Generator  extends  AObject
{
 ……
 public void transition1() throws SaosException{
   if(view.button.selected.val==false){
   enabled.setVal(false);
   }
 }
 public void transition2() throws SaosException{
   if(view.button.selected.val==true){
   enabled.setVal(true);
   }
 }
public void transition3() throws SaosException{
   if(generatable.val==true &&( state.val==READY)){
   generatable.setVal(false);
   outputQueue.nJobs.increment();
   state.setVal(IDLE);
   }
 }
 public void transition4() throws SaosException{
   if( state.val==BUSY){
   generatable.setVal(true);
   state.setVal(READY);
   }
 }
 public void transition5() throws SaosException{
   if(enabled.val==true&&( state.val==IDLE)){
   state.setVal(BUSY);
   fcallOftransition4 = FCall.fCall((AObject)this,TRANSITION4, 2, "transition4()",SaosMain.AosUser);
   }
 }
…..
```

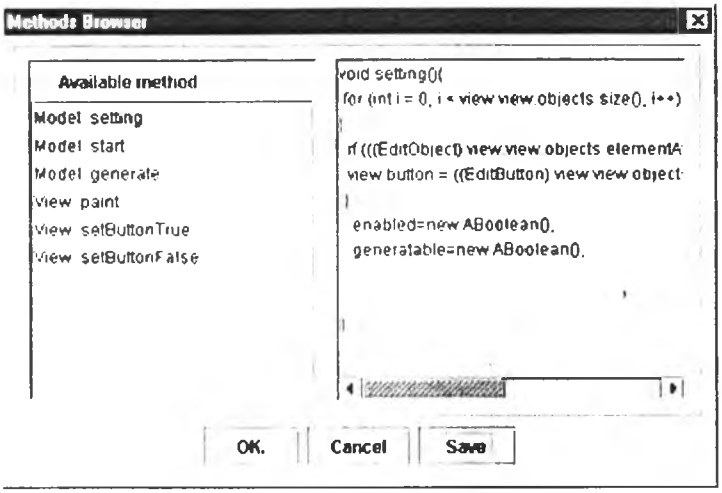
แปลงจากชื่อการเปลี่ยนแปลง

แปลงจากส่วนเงื่อนไข

แปลงจากส่วนการกระทำ

รูปที่ ก-5 ฟังก์ชันสมาชิกที่แปลงจากการเปลี่ยนแปลง

นอกจากฟังก์ชันสมาชิกที่แปลงมาจากการเปลี่ยนแปลงของแผนภาพสเตทชาร์ทแล้วยังมีฟังก์ชันสมาชิกที่สร้างมาจากการกำหนดรายละเอียดโดยตรงของบรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทในส่วนแสดงฟังก์ชันสมาชิก ในรูปที่ ก-6 ผู้พัฒนาโปรแกรมประยุกต์สามารถกำหนดรายละเอียดให้กับฟังก์ชัน setting ของตัวสร้างงานได้โดยตรงซึ่งยังไม่มีรายละเอียดของประโยคคำสั่งภายในฟังก์ชัน ดังนั้นผู้พัฒนาโปรแกรมประยุกต์ต้องทำการกำหนดประโยคคำสั่งเพื่อทำให้ฟังก์ชันนี้สมบูรณ์



รูปที่ ก-6 การกำหนดรายละเอียดฟังก์ชันสมาชิก

เมื่อทำการสร้างชุดคำสั่งตัวสร้างชุดคำสั่งจะนำรายละเอียดของฟังก์ชันในส่วนนี้มากำหนดเป็นรายละเอียดของฟังก์ชันในชุดคำสั่งดังรูปที่ ก-7

```
public class Generator  extends  AObject
{
 ……
   public void setting () throws SaosException{
    for (int i = 0; i < view.view.objects.size(); i++)
              {
               if (((EditObject) view.view.objects.elementAt(i)).getClass().getName().equals("EditButton"))
                 view.button = ((EditButton) view.view.objects.elementAt(i)).button;
    }
    enabled=new ABoolean();
    generatable=new ABoolean();
  }
…..
```

รูปที่ ก-7 ฟังก์ชันสมาชิกที่ได้จากการกำหนดชุดคำสั่งโดยตรง

ก-3 ส่วนสร้างตัวเริ่มต้นการทำงาน

ชุดคำสั่งในส่วนสร้างตัวเริ่มต้นการทำงานคือชุดคำสั่งที่สร้างความสัมพันธ์ระหว่างตัวแปรพร้อมทำงานและฟังก์ชันสมาชิกเพื่อให้ระบบสามารถทราบได้ว่าจะต้องประมวลผลฟังก์ชันสมาชิกใดเมื่อตัวแปรพร้อมทำงานเปลี่ยนค่า การกำหนดประโยคคำสั่งของการสร้างตัวเริ่มต้นการทำงานจะต้องกำหนดในภายในฟังก์ชันดังต่อไปนี้

1) ฟังก์ชัน initialize

กำหนดประโยคคำสั่งของการสร้างตัวเริ่มต้นการทำงานภายในฟังก์ชัน initialize ในกรณีที่ ตัวแปรพร้อมทำงานที่อ้างถึงเป็นตัวแปรสมาชิกของวัตถุนี้

2) ฟังก์ชัน connectModelViewInput

กำหนดประโยคคำสั่งของการสร้างตัวเริ่มต้นการทำงานภายในฟังก์ชัน connectModel-ViewInput ในกรณีที่ ตัวแปรพร้อมทำงานที่อ้างถึงเป็นตัวแปรสมาชิกของวัตถุที่เชื่อมต่อกับส่วนนำเข้า (Input port)

3) ฟังก์ชัน connectModelViewOutput

กำหนดประโยคคำสั่งของการสร้างตัวเริ่มต้นการทำงานภายในฟังก์ชัน connectModel-ViewOutput ในกรณีที่ตัวแปรพร้อมทำงานที่อ้างถึงเป็นตัวแปรสมาชิกของวัตถุที่เชื่อมต่อกับส่วนส่งออก (Output port)

ความสัมพันธ์ระหว่างตัวแปรพร้อมทำงานและฟังก์ชันสมาชิกพิจารณาจากการเปลี่ยนแปลงที่กำหนดเหตุการณ์ด้วยสัญลักษณ์ "changeOf(av_1,av_2,…,av_n)" โดยทำการสร้างความสัมพันธ์ระหว่างตัวแปรพร้อมทำงานในสัญลักษณ์กับชื่อของการเปลี่ยนแปลงซึ่งเป็นชื่อของฟังก์ชันสมาชิก ดังชุดคำสั่งในรูปที่ ก-8 เป็นการกำหนดประโยคคำสั่งของการสร้างตัวเริ่มต้นการทำงานของตัวสร้างงานที่แปลงจากแผนภาพสเตทชาร์ทในรูปที่ ก-3 ซึ่งจะเห็นว่าการเปลี่ยนแปลงที่กำหนดด้วยสัญลักษณ์ "changeOf" ถูกทำการเชื่อมโยงกับตัวแปรพร้อมทำงานที่กำหนดในสัญลักษณ์

```
public class Generator  extends  AObject
{
 ……
public void initialize(){
  try{

    initIDLE();

    view.button.selected.addTE((AObject)this, TRANSITION1,"transition1()", SaosMain.AosUser);
    view.button.selected.addTE((AObject)this, TRANSITION2,"transition2()", SaosMain.AosUser);
    generatable.addTE((AObject)this, TRANSITION3,"transition3()", SaosMain.AosUser);
    enabled.addTE((AObject)this, TRANSITION5,"transition5()", SaosMain.AosUser);
    generatable.addTE((AObject)this, TRANSITION5,"transition5()", SaosMain.AosUser);

  }catch(SaosException e){
   System.out.println("Error :"+e);
  }
}
public void connectModelViewInput(PortView port) throws SaosException
{
}
public void connectModelViewOutput(PortView port) throws SaosException
  {
   outputQueue = (Queue) port.address.objectPtr;
    outputQueue.nJobs.addTE((AObject)this, TRANSITION3,"transition3()", SaosMain.AosUser);
  }
…..
```

รูปที่ ก-8 ประโยคคำสั่งของการสร้างตัวเริ่มต้นการทำงาน

ก-4 ส่วนประโยคคำสั่งของเหตุการณ์ในอนาคต

จากแผนภาพสเตทชาร์ทของตัวสร้างงานในรูปที่ ก-3 มีการเปลี่ยนแปลงหนึ่งที่กำหนดสัญลักษณ์ของเหตุการณ์ด้วยสัญลักษณ์ของเหตุการณ์แบบเวลา จากความหมายในแผนภาพสเตทชาร์ทหมายถึง เมื่อเปลี่ยนสถานะมายังสถานะ BUSY แล้วหลังจากนั้นอีก 2 หน่วยเวลาให้เปลี่ยนสถานะไปยัง READY ชุดคำสั่งของวัตถุพร้อมทำงานสำหรับกำหนดเหตุการณ์ในอนาคตทำโดย กำหนดประโยคการเรียกฟังก์ชันล่วงหน้าไว้ในฟังก์ชันที่แปลงมาจากการเปลี่ยนแปลงที่เป็นตัวเริ่มของเหตุการณ์ประเภทเวลา ในที่นี้คือการเปลี่ยนแปลงจากสถานะ IDLE มายังสถานะ BUSY (การเปลี่ยนแปลงนี้คือtransition5) ในประโยคการเรียกฟังก์ชันล่วงหน้าได้ทำการระบุให้เรียกฟังก์ชันที่แปลงมาจากการเปลี่ยนแปลงที่ระบุด้วยเหตุการณ์ประเภทเวลา (การเปลี่ยนแปลงนี้คือ transition4) ดังที่แสดงในรูปที่ ก-9

```
public class Generator  extends  AObject
{
 ……
 public void transition4() throws SaosException{
    if( state.val==BUSY){
    generatable.setVal(true);
    state.setVal(READY);
    }
 }
 public void transition5() throws SaosException{
    if(enabled.val==true&&( state.val==IDLE)){
    state.setVal(BUSY);
    fcallOftransition4 = FCall.fCall((AObject)this,TRANSITION4, 2, "transition4()",SaosMain.AosUser);
    }
 }                                          ประโยคการเรียกฟังก์ชันล่วงหน้า
…..
```

รูปที่ ก-9 ชุดคำสั่งของเหตุการณ์ในอนาคต

สำหรับการกระทำในอนาคตที่กำหนดด้วยสัญลักษณ์ FCall และ FAssign จะถูกทำการแปลงเป็นประโยคการเรียกฟังก์ชันล่วงหน้าและประโยคการกำหนดค่าล่วงหน้าโดยตรง แล้วกำหนดเป็นประโยคคำสั่งไว้ภายในฟังก์ชัน

ก-5 ส่วนฟังก์ชันดิสแพทช์ชิง

ฟังก์ชันดิสแพทช์ชิงคือฟังก์ชันที่ทำหน้าที่แปลงดัชนีฟังก์ชันให้เป็นการเรียกฟังก์ชันที่แท้จริง การสร้างฟังก์ชันดิสแพทช์ชิงทำโดยการสร้างค่าคงที่เพื่อใช้เป็นดัชนีฟังก์ชันจากชื่อของฟังก์ชันสมาชิกที่แปลงมาจากการเปลี่ยนแปลงของแผนภาพสเตทชาร์ทและชื่อของฟังก์ชันสมาชิกที่ถูกเรียกโดยการเรียกฟังก์ชันล่วงหน้า เพราะฟังก์ชันสมาชิกเหล่านี้จะถูกเรียกโดยฟังก์ชันดิสแพทช์ชิง สำหรับรายละเอียดของฟังก์ชันดิสแพทช์ชิงคือการกำหนดความสัมพันธ์ระหว่างดัชนีฟังก์ชันกับประโยคการเรียกฟังก์ชัน ดังรูปที่ ก-10

```
public class Generator  extends  AObject
{
 ……
 static final int TRANSITION4 = 0;
 static final int TRANSITION1 = 1;
 static final int TRANSITION2 = 2;
 static final int TRANSITION3 = 3;
 static final int TRANSITION5 = 4;

 public void dispatch(int funcIndex) throws SaosException
 {
  switch(funcIndex)
  {
    case TRANSITION4: transition4(); break;
    case TRANSITION1: transition1(); break;
    case TRANSITION2: transition2(); break;
    case TRANSITION3: transition3(); break;
    case TRANSITION5: transition5(); break;
  default: System.out.println("ERROR:dispatch():"+this+
    ":functionIndex="+funcIndex);
  }
 }
…..
```

รูปที่ ก-10 ชุดคำสั่งส่วนฟังก์ชันดิสแพทช์ชิง

ก-6 ตัวอย่างชุดคำสั่งที่สมบูรณ์ของตัวสร้างงาน

ในส่วนนี้แสดงตัวอย่างของชุดคำสั่งที่สมบูรณ์ของตัวสร้างงาน โดยแสดงรายละเอียดของคลาสโมเดลในรูปที่ ก-11 และคลาสวิวในรูปที่ ก-12

```
import saos.gui.*;
import saos.kernel.*;
import java.awt.*;
import java.io.*;
import java.lang.Math;
import java.util.*;
public class Generator  extends  AObject
{
 public ABoolean generatable;
 public ABoolean enabled;
 public int counter;
 public int interval = 8;
 public Random rand;
 EditGenerator view;
 public AInteger state = new AInteger();
 static final int IDLE= 0;
 static final int BUSY= 1;
 static final int READY= 2;
 public Queue outputQueue ;
 FCall fcallOftransition4=null;
 public Generator(String name)
 {
   super(name);
 }
 public void transition1() throws SaosException{
   if(view.button.selected.val==false){
   enabled.setVal(false);
   }
 }
 public void transition2() throws SaosException{
   if(view.button.selected.val==true){
   enabled.setVal(true);
   }
 }
public void transition3() throws SaosException{
   if(generatable.val==true &&( state.val==READY)){
   generatable.setVal(false);
   outputQueue.nJobs.increment();
   state.setVal(IDLE);
   }
 }
 public void transition4() throws SaosException{
   if( state.val==BUSY){
   generatable.setVal(true);
   state.setVal(READY);
   }
 }
public void transition5() throws SaosException{
   if(enabled.val==true&&( state.val==IDLE)){
   state.setVal(BUSY);
   fcallOftransition4 = FCall.fCall((AObject)this,TRANSITION4, 2, "transition4()",SaosMain.AosUser);
   }
 }
.....
```

รูปที่ ก-11 ชุดคำสั่งในคลาสโมเดลของตัวสร้างงาน

```
public void initIDLE() throws SaosException{

   entryRoot();
   state.setVal(IDLE);
 }
 public void entryRoot() throws SaosException{

   setting();
 }
 public void initialize(){
  try{

   initIDLE();

   view.button.selected.addTE((AObject)this, TRANSITION1,"transition1()", SaosMain.AosUser);
   view.button.selected.addTE((AObject)this, TRANSITION2,"transition2()", SaosMain.AosUser);
   generatable.addTE((AObject)this, TRANSITION3,"transition3()", SaosMain.AosUser);
   enabled.addTE((AObject)this, TRANSITION5,"transition5()", SaosMain.AosUser);
   generatable.addTE((AObject)this, TRANSITION5,"transition5()", SaosMain.AosUser);

  }catch(SaosException e){
  System.out.println("Error :"+e);
  }
 }
 static final int TRANSITION4 = 0;
 static final int TRANSITION1 = 1;
 static final int TRANSITION2 = 2;
 static final int TRANSITION3 = 3;
 static final int TRANSITION5 = 4;

 public void dispatch(int funcIndex) throws SaosException
 {
  switch(funcIndex)
  {
   case TRANSITION4: transition4(); break;
   case TRANSITION1: transition1(); break;
   case TRANSITION2: transition2(); break;
   case TRANSITION3: transition3(); break;
   case TRANSITION5: transition5(); break;
  default: System.out.println("ERROR:dispatch():"+this+
   ":functionIndex="+funcIndex);
  }
 }
public void setting () throws SaosException{
    for (int i = 0; i < view.view.objects.size(); i++)
            {
             if (((EditObject) view.view.objects.elementAt(i)).getClass().getName().equals("EditButton"))
              view.button = ((EditButton) view.view.objects.elementAt(i)).button;
    }
   enabled=new ABoolean();
   generatable=new ABoolean();
 }
.....
```

รูปที่ ก-11 ชุดคำสั่งในคลาสโมเดลของตัวสร้างงาน (ต่อ)

```
.....
public void connectModelViewInput(PortView port) throws SaosException
{
}
public void connectModelViewOutput(PortView port) throws SaosException
 {
  outputQueue = (Queue) port.address.objectPtr;
  outputQueue.nJobs.addTE((AObject)this, TRANSITION3,"transition3()", SaosMain.AosUser);
 }
}
```

รูปที่ ก-11 ชุดคำสั่งในคลาสโมเดลของตัวสร้างงาน (ต่อ)

```
import saos.gui.*;
import saos.kernel.*;
import java.awt.*;
import java.io.*;
import java.lang.Math;
import java.util.*;

class  EditGenerator  extends  EditApp implements Constants
{
 Generator  model;
 public Button11 button;

 public EditGenerator(EditCompound object, AppCanvas canvas, int x, int y)
 {
  super(x, y);
  model = new Generator("Generator");
  model.view = this;
  view  = (EditCompound) object.clone();
  this.canvas = canvas;
  name = new String("Generator");
  outPortView = new PortView(OUT, canvas, this, SINGLE);
 }
public void initialize() throws SaosException
 {
  insert(model, true);
  view.updateCoordinates(x,y, x - view.x, y - view.y);
  view.setCanvas(canvas);
  outPortStatus = false;
  portNumber = 2;
  view.initialize(canvas.getGraphics());
  outPortView.initialize();
 }
 public void paint (Graphics g) {
   outPortView.initialize();
 }

.....
```

รูปที่ ก-12 ชุดคำสั่งในคลาสวิวของตัวสร้างงาน

```
.....
public void connectModelViewInput(PortView port) throws SaosException
  {
    model.connectModelViewInput(port);
  }
  public void connectModelViewOutput(PortView port) throws SaosException
  {
    model.connectModelViewOutput(port);
  }
}
```

รูปที่ ก-12 ชุดคำสั่งในคลาสวิวของตัวสร้างงาน (ต่อ)

หอสมุดกลาง สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย

ประวัติผู้เขียนวิทยานิพนธ์

นายวุฒิพงษ์ เรือนทอง เกิดวันที่ 9 เมษายน พ.ศ. 2517 สำเร็จการศึกษาปริญญาตรีวิทยาศาสตร์บัณฑิต สาขาวิทยาการคอมพิวเตอร์ จากมหาวิทยาลัยนเรศวรในปีการศึกษา 2540 จากนั้นเข้ารับราชการในตำแหน่งอาจารย์ระดับ 3 ที่มหาวิทยาลัยนเรศวรจนถึงปีการศึกษา 2541 แล้วลาศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต ที่คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย