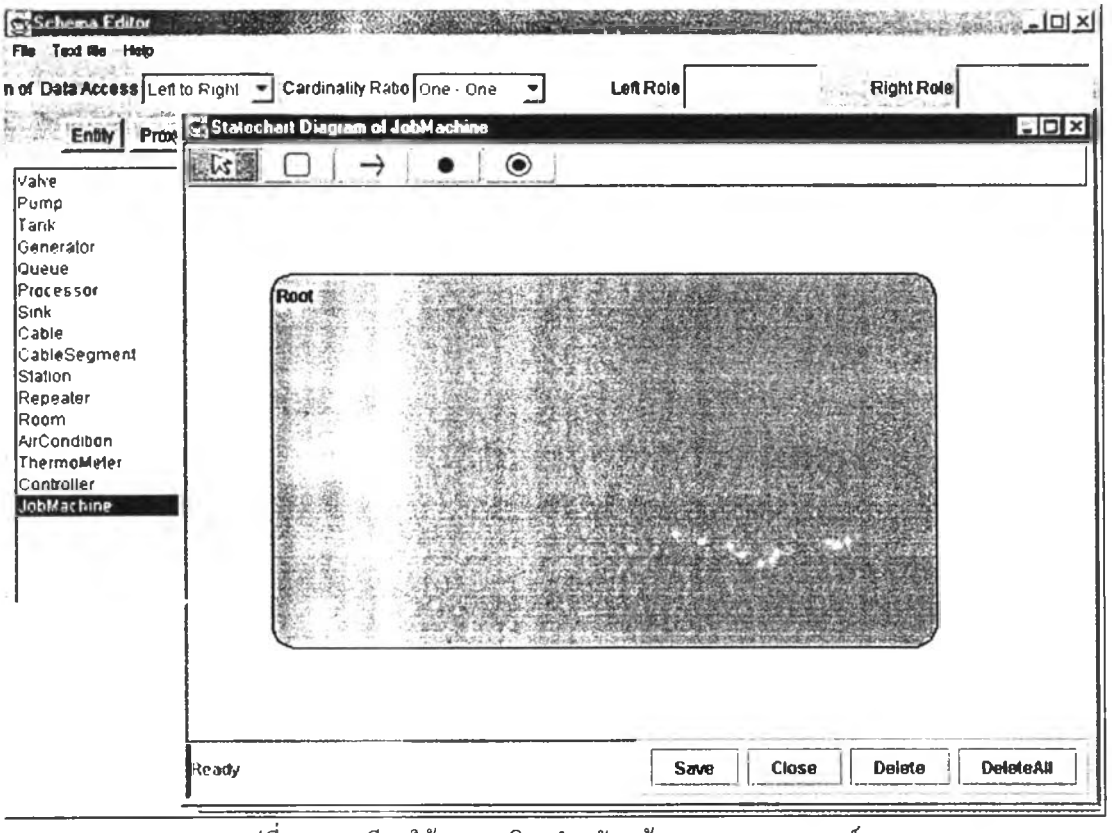
บทที่ 6

การใช้งานบรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ท

เนื้อหาในบทนี้เป็นการอธิบายวิธีการใช้งานบรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทและวิธีการสร้างชุดคำสั่งของวัตถุพร้อมทำงาน เนื้อหาตอนท้ายแสดงตัวอย่างการสร้างโปรแกรมประยุกต์โดยใช้บรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทในการกำหนดพฤติกรรมให้กับวัตถุพร้อมทำงาน

6.1 การเรียกใช้บรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ท

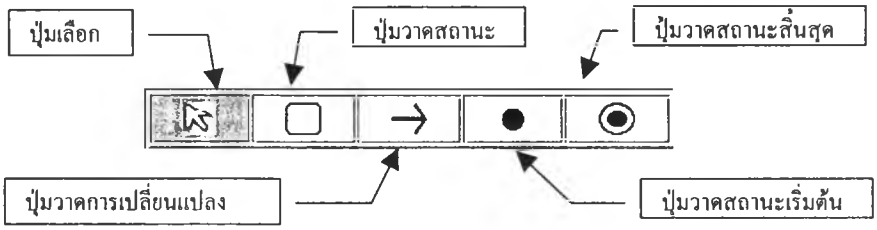
บรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทถูกกำหนดให้เป็นส่วนหนึ่งของบรรณาธิกรสำหรับสร้างเค้าร่างและมีหน้าที่สร้างแผนภาพสเตทชาร์ทให้กับชนิดของเอนทิตีแต่ละตัวในแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติม ดังนั้นผู้ใช้จะต้องสร้างแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติมก่อน หลังจากนั้นจึงทำการสร้างแผนภาพสเตทชาร์ทให้กับชนิดของเอนทิตีแต่ละตัวด้วยบรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทโดยการดับเบิลคลิกโดยตรงที่ชนิดของเอนทิตีที่ต้องการกำหนดแผนภาพสเตทชาร์ทเพื่อเรียกบรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ท หน้าจอของบรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทจะปรากฏขึ้นดังรูปที่ 6.1



รูปที่ 6.1 การเรียกใช้บรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ท

6.2 การใช้ปุ่มเครื่องมือวาดแผนภาพสเตทชาร์ท

บรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทมีปุ่มเครื่องมือที่เตรียมไว้สำหรับใช้วาดแผนภาพสเตทชาร์ทดังรูปที่ 6.2 ปุ่มเครื่องมือวาดแต่ละปุ่มมีหน้าที่และวิธีการใช้งานแตกต่างกันดังต่อไปนี้



รูปที่ 6.2 แถบเครื่องมือสำหรับใช้วาดแผนภาพสเตทชาร์ท

6.2.1 ปุ่มเลือก

ปุ่มเลือกคือปุ่มที่ไม่ได้ถูกใช้เป็นปุ่มสำหรับวาดแต่เป็นปุ่มที่ใช้สำหรับกำหนดสถานะการเลือก เมื่อปุ่มเลือกถูกกดผู้ใช้สามารถเลือกองค์ประกอบต่างๆ ในแผนภาพสเตทชาร์ทได้

6.2.2 ปุ่มวาดสถานะ

ปุ่มวาดสถานะคือปุ่มที่ใช้กำหนดการวาดสถานะซึ่งวิธีการวาดสถานะทำโดยการเลือกปุ่มวาดสถานะแล้วคลิกเมาส์ที่สถานะพ่อแม่ (Parent state) ที่ต้องการให้มีสถานะใหม่ปรากฏขึ้นเป็นสถานะย่อยของของสถานะพ่อแม่ที่เลือก

6.2.3 ปุ่มวาดการเปลี่ยนแปลง

ปุ่มวาดการเปลี่ยนแปลงคือปุ่มที่ใช้วาดการเปลี่ยนแปลงภายนอกซึ่งวิธีวาดการเปลี่ยนแปลงภายนอกทำโดยการเลือกปุ่มวาดการเปลี่ยนแปลงแล้วคลิกเมาส์ที่สถานะที่ต้องการให้เป็นสถานะตั้งต้นหลังจากนั้นคลิกเม้าส์อีกครั้งหนึ่งที่สถานะที่ต้องการให้เป็นสถานะเป้าหมาย แล้วเส้นหัวลูกศรจะปรากฏขึ้น

6.2.4 ปุ่มวาดสถานะเริ่มต้น

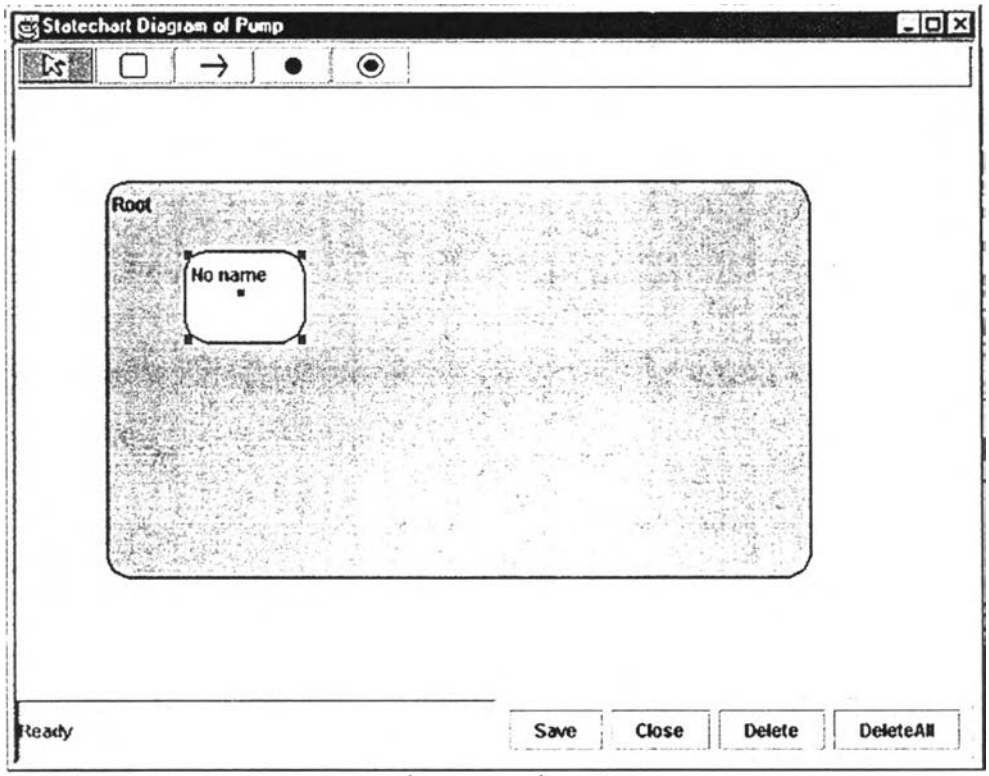
ปุ่มวาดสถานะเริ่มต้นคือปุ่มที่ใช้กำหนดการวาดสถานะเริ่มต้นซึ่งวิธีวาดสถานะเริ่มต้นทำโดยการเลือกปุ่มวาดสถานะเริ่มต้น แล้วคลิกเมาส์ที่สถานะพ่อแม่ที่ต้องการให้มีสถานะเริ่มต้น สถานะเริ่มต้นจะปรากฏภายในสถานะพ่อแม่ที่เลือก หลังจากนั้นเลือกปุ่มวาดการเปลี่ยนแปลงแล้วคลิกที่สถานะเริ่มต้นเพื่อกำหนดให้เป็นสถานะตั้งต้นหลังจากนั้นคลิกเมาส์ที่สถานะย่อยที่ต้องการให้เป็นสถานะเป้าหมาย

6.2.5 ปุ่มวาดสถานะสิ้นสุด

ปุ่มวาดสถานะสิ้นสุดคือปุ่มที่ใช้กำหนดการวาดสถานะสิ้นสุดซึ่งมีวิธีการใช้งานเหมือนกับปุ่มวาดสถานะเริ่มต้นแต่แตกต่างกันที่ไม่สามารถกำหนดให้เป็นสถานะตั้งต้นของการเปลี่ยนแปลงแต่กำหนดให้เป็นสถานะเป้าหมายของการเปลี่ยนแปลงได้เท่านั้น

6.3 การกำหนดคุณสมบัติให้กับสถานะ

หลังจากผู้ใช้ทำการสร้างสถานะโดยเลือกที่ปุ่มสร้างสถานะแล้วคลิกที่สถานะรากจะปรากฏสถานะพื้นฐานซึ่งเป็นสถานะย่อยของสถานะรากดังรูปที่ 6.3 สถานะพื้นฐานนี้สามารถทำการเปลี่ยนแปลงขนาดได้โดยการวางตำแหน่งของเมาส์ที่ขอบของสถานะแล้วลากเมาส์เพื่อเปลี่ยนแปลงขนาดของสถานะ ขนาดของสถานะจะเปลี่ยนแปลงไปตามตำแหน่งของเมาส์ที่ลาก



รูปที่ 6.3 สถานะที่ถูกสร้าง

ผู้ใช้สามารถกำหนดคุณสมบัติให้กับสถานะได้โดยการดับเบิลคลิกที่สถานะที่ต้องการ แล้วหน้าจอสำหรับกำหนดคุณสมบัติของสถานะจะปรากฏขึ้นดังรูปที่ 6.4 ซึ่งมีส่วนกำหนดคุณสมบัติให้กับสถานะดังต่อไปนี้

6.3.1 ส่วนกำหนดชื่อสถานะ

ส่วนกำหนดชื่อสถานะคือช่องเติมข้อความซึ่งผู้ใช้สามารถกำหนดชื่อของสถานะได้โดยการพิมพ์ชื่อสถานะในช่องเติมข้อความนี้

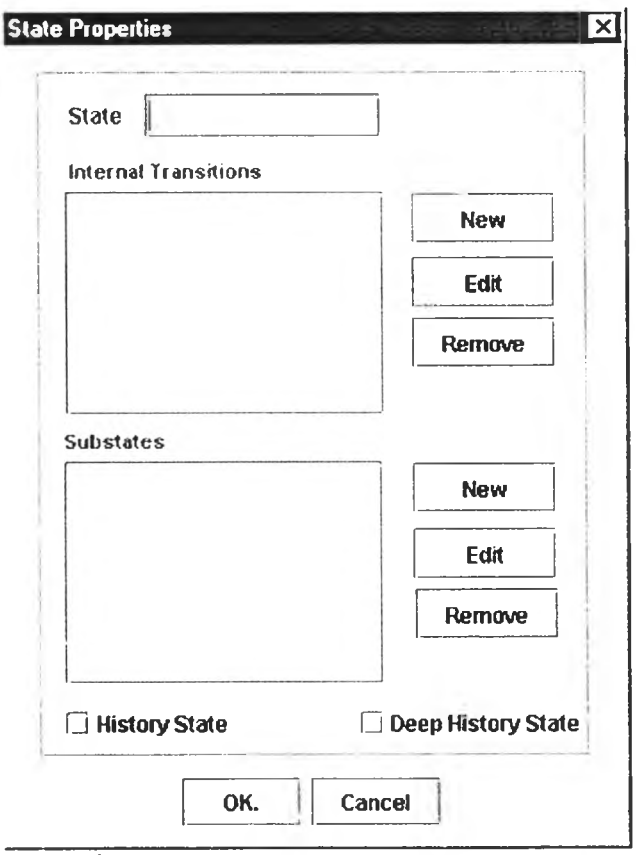
6.3.2 ส่วนกำหนดการเปลี่ยนแปลงภายใน

ผู้ใช้สามารถสร้างการเปลี่ยนแปลงภายในให้กับสถานะที่เลือกได้โดยการกดที่ปุ่ม New ของส่วนกำหนดการเปลี่ยนแปลงภายใน หลังจากนั้นจึงทำการกำหนดคุณสมบัติให้กับการเปลี่ยนแปลงภายในจากหน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลงซึ่งจะกล่าวถึงในหัวข้อถัดไป การเปลี่ยนแปลงภายในที่สร้างใหม่นี้จะปรากฏในรายการและปรากฏในสถานะที่เลือก ผู้ใช้สามารถทำการแก้ไขคุณ

สมบัติของการเปลี่ยนแปลงภายในได้โดยเลือกการเปลี่ยนแปลงภายในที่อยู่ในรายการแล้วกดปุ่ม Edit หรือลบการเปลี่ยนแปลงภายในโดยเลือกการเปลี่ยนแปลงภายในที่อยู่ในรายการแล้วกดปุ่ม Remove

6.3.3 ส่วนกำหนดสถานะย่อย

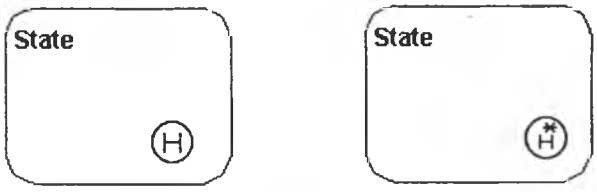
ผู้ใช้สามารถสร้างสถานะย่อยให้กับสถานะที่เลือกได้โดยวิธีการเดียวกับส่วนกำหนดการเปลี่ยนแปลงภายในคือ กดปุ่ม New เพื่อสร้างสถานะย่อยใหม่โดยจะแสดงหน้าจอสำหรับกำหนดคุณสมบัติของสถานะขึ้นอีกชั้นหนึ่งเพื่อใช้กำหนดคุณสมบัติให้กับสถานะย่อย กดปุ่ม Edit เพื่อแก้ไขสถานะย่อยที่เลือกในรายการ และกดปุ่ม Remove เพื่อลบสถานะย่อยที่เลือกในรายการ



รูปที่ 6.4 หน้าจอสำหรับกำหนดคุณสมบัติของสถานะ

6.3.4 ส่วนกำหนดสถานะครั้งก่อน

ส่วนกำหนดสถานะครั้งก่อนคือช่องสำหรับเลือกซึ่งมี 2 ช่องในหน้าจอได้แก่ ช่องที่ระบุด้วย History State ซึ่งกำหนดให้แสดงสถานะครั้งก่อนภายในสถานะที่เลือก และช่องที่ระบุด้วย Deep History State ซึ่งกำหนดให้แสดงสถานะลึกครั้งก่อนภายในสถานะที่เลือก รูปที่ 6.5 แสดงตัวอย่างของสถานะที่ถูกระบุให้มีสถานะครั้งก่อนและสถานะลึกครั้งก่อน

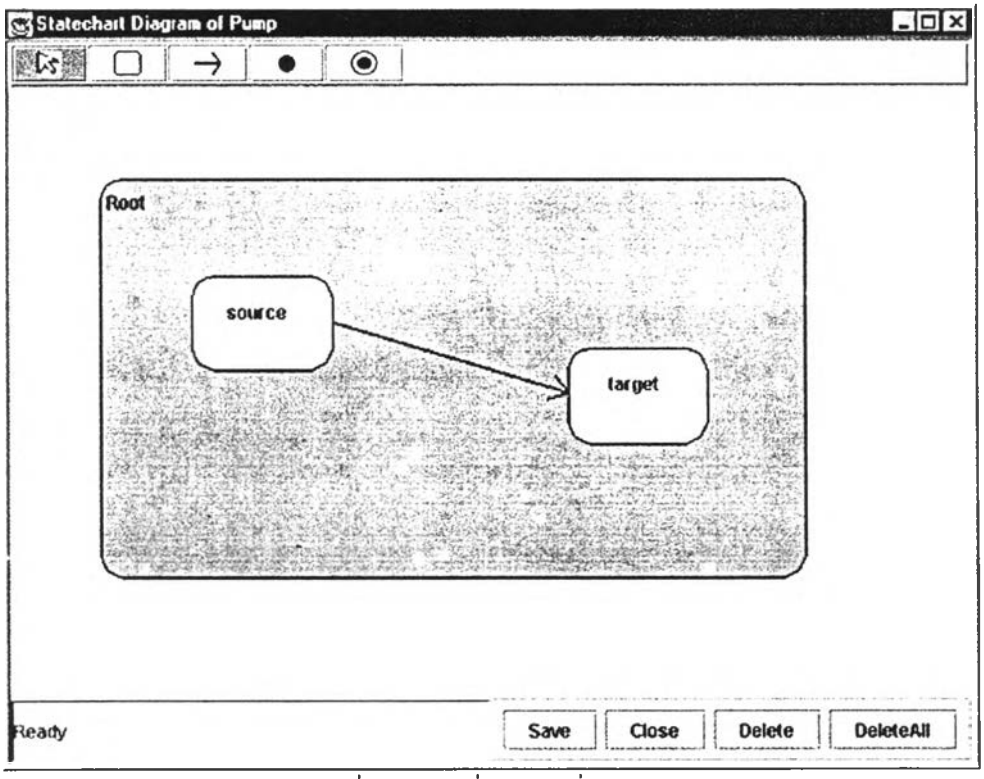


(ก) (ข)

รูปที่ 6.5 (ก) สถานะที่ระบุให้มีสถานะครั้งก่อนและ (ข) สถานะที่ระบุให้มีสถานะลึกครั้งก่อน

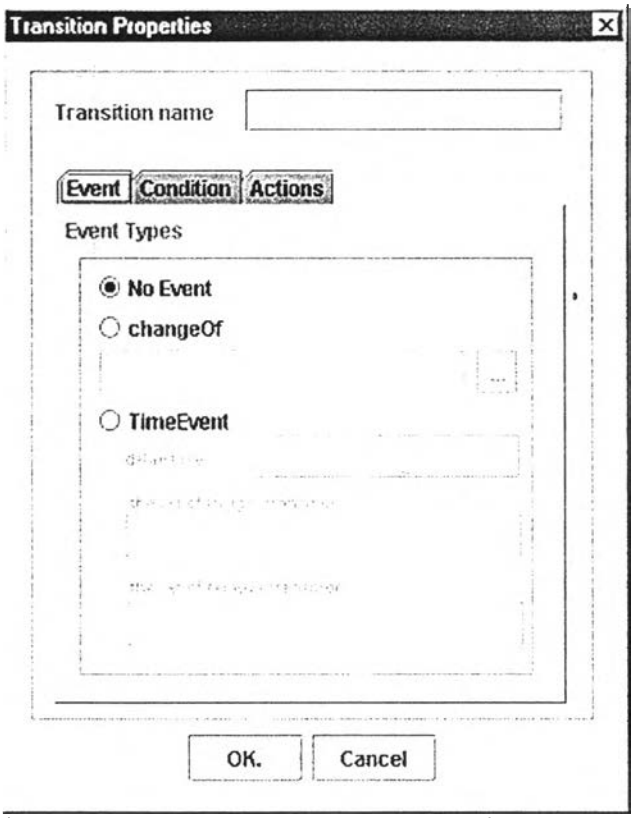
6.4 การกำหนดคุณสมบัติให้กับการเปลี่ยนแปลง

การสร้างการเปลี่ยนแปลงทำได้โดยการกดปุ่มสร้างการเปลี่ยนแปลงแล้วคลิกเมาส์ที่สถานะตั้งต้นและสถานะเป้าหมายตามลำดับจะปรากฏเส้นหัวลูกศรแสดงการเปลี่ยนแปลงจากสถานะตั้งต้นไปยังสถานะเป้าหมายดังรูปที่ 6.6



รูปที่ 6.6 การเปลี่ยนแปลงที่ถูกสร้าง

ผู้ใช้สามารถกำหนดคุณสมบัติให้กับการเปลี่ยนแปลงได้โดยการดับเบิลคลิกที่เส้นหัวลูกศรซึ่งจะปรากฏหน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลงดังรูปที่ 6.7



รูปที่ 6.7 หน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลงภายนอก

หน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลงมีส่วนที่ใช้กำหนดคุณสมบัติให้กับการเปลี่ยนแปลงดังต่อไปนี้

6.4.1 ส่วนกำหนดชื่อการเปลี่ยนแปลง

ส่วนกำหนดชื่อการเปลี่ยนแปลง คือช่องเติมข้อความที่ระบุด้วยข้อความ "Transition name" ผู้ใช้สามารถกำหนดชื่อการเปลี่ยนแปลงได้โดยการพิมพ์ชื่อการเปลี่ยนแปลงในช่องเติมข้อความนี้

6.4.2 ส่วนกำหนดชนิดของเหตุการณ์

ส่วนกำหนดชนิดของเหตุการณ์ คือส่วนที่ใช้กำหนดเหตุการณ์ที่เป็นตัวกระตุ้นให้เกิดการเปลี่ยนแปลง การกำหนดชนิดของเหตุการณ์ทำได้โดยเลือกที่ปุ่มของแต่ละชนิดเหตุการณ์ดังต่อไปนี้

1) ไม่กำหนดเหตุการณ์

ปุ่มที่ระบุด้วยข้อความ "No event" คือปุ่มสำหรับกำหนดให้การเปลี่ยนแปลงเป็นการเปลี่ยนแปลงแบบสมบูรณ์ซึ่งไม่มีเหตุการณ์เป็นตัวกระตุ้น

2) เหตุการณ์การเปลี่ยนค่าของตัวแปรพร้อมทำงาน

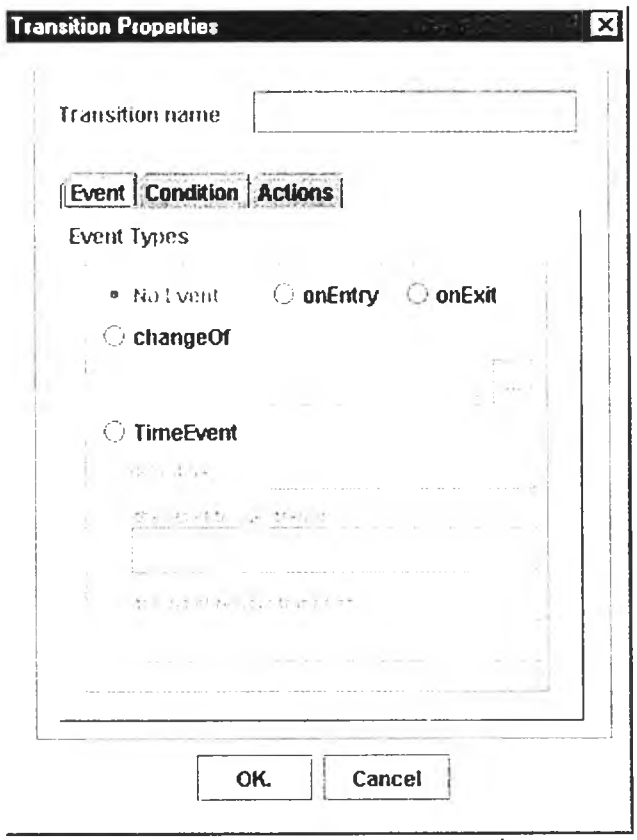
ปุ่มที่ระบุด้วยข้อความ "changeOf " คือปุ่มสำหรับกำหนดเหตุการณ์การเปลี่ยนค่าของตัวแปรพร้อมทำงาน การระบุตัวแปรพร้อมทำงานที่ใช้เป็นตัวเริ่มต้นการทำงานของการเปลี่ยนแปลงสามารถกำหนดภายในช่องเติมข้อความด้านล่างโดยการพิมพ์รายการของตัวแปรสมาชิกโดยตรงหรือการเรียกใช้บริการจากหน้าจอแสดงตัวแปรสมาชิก

3) เหตุการณ์ประเภทเวลา

ปุ่มที่ระบุด้วยข้อความ "TimeEvent" คือปุ่มสำหรับกำหนดเหตุการณ์ประเภทเวลาซึ่งจะต้องระบุเวลาหน่วงในช่องที่ระบุด้วยข้อความ "delay time" ถ้ามีการกำหนดการเปลี่ยนแปลงที่เป็นตัวเริ่มต้นเหตุการณ์ประเภทเวลาจะต้องระบุรายการของการเปลี่ยนแปลงในช่องที่ระบุด้วยข้อความ "the set of trigger transitions" หรือถ้ามีการกำหนดการเปลี่ยนแปลงที่เป็นตัวยกเลิกเหตุการณ์ประเภทเวลาจะต้องระบุรายการของการเปลี่ยนแปลงในช่องที่ระบุด้วยข้อความ "the set of remove transitions"

4) เหตุการณ์การเข้าสู่สถานะและเหตุการณ์การออกจากสถานะ

เหตุการณ์การเข้าสู่สถานะและเหตุการณ์การออกจากสถานะจะเป็นเหตุการณ์สำหรับการเปลี่ยนแปลงภายในเท่านั้นซึ่งหน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลงภายในจะมีปุ่มเลือกเหตุการณ์ปรากฏเพิ่มขึ้นอีก 2 ชนิด คือ ปุ่มที่ระบุด้วยข้อความ "onEntry" และปุ่มที่ระบุด้วยข้อความ "onExit" รูปที่ 6.8 แสดงปุ่มสำหรับกำหนดเหตุการณ์การเข้าสู่สถานะและเหตุการณ์การออกจากสถานะตามลำดับ

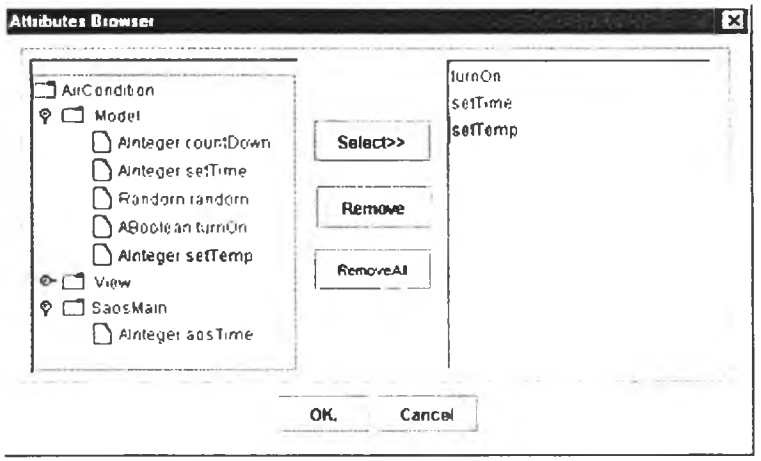


รูปที่ 6.8 หน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลงภายใน

6.4.3 การเรียกใช้หน้าจอสำหรับแสดงตัวแปรสมาชิก

การกำหนดเหตุการณ์ชนิดการเปลี่ยนค่าของตัวแปรพร้อมทำงานจำเป็นต้องทำการระบุรายการของตัวแปรพร้อมทำงาน หน้าจอแสดงตัวแปรสมาชิกของวัตถุพร้อมทำงานในรูปที่ 6.9 สามารถ

ช่วยให้ผู้ใช้ระบุรายการของตัวแปรพร้อมทำงานได้สะดวกขึ้น เพราะผู้ใช้สามารถเลือกตัวแปรพร้อมทำงานที่ต้องการได้จากโครงสร้างต้นไม้ซ้ายมือ เมื่อกดปุ่ม Select ชื่อของตัวแปรพร้อมทำงานที่เลือกจะปรากฏในรายการขวามือ เมื่อกดปุ่ม OK รายการของตัวแปรพร้อมทำงานที่เลือกจะปรากฏในส่วนระบุตัวแปรพร้อมทำงานของหน้าจอกำหนดคุณสมบัติของการเปลี่ยนแปลงทันที

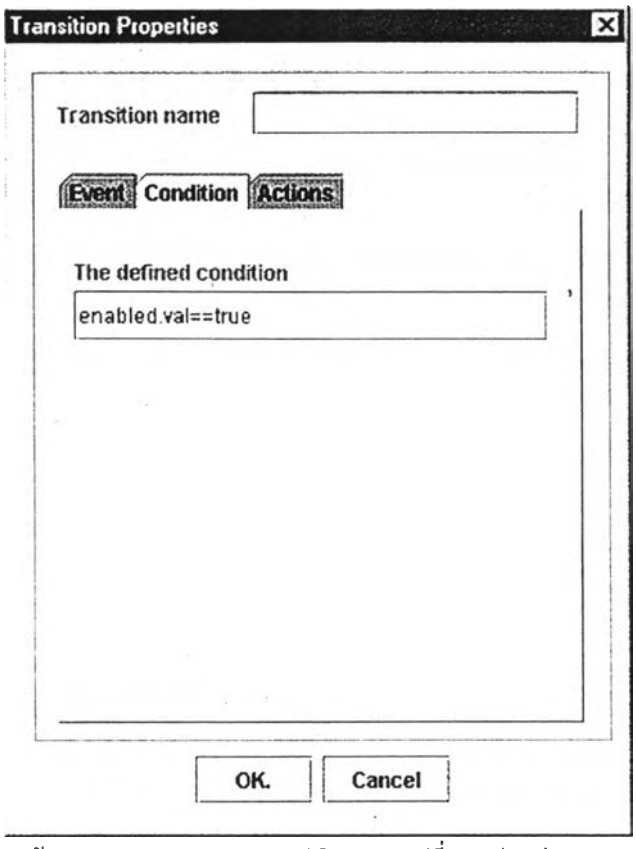


รูปที่ 6.9 หน้าจอสำหรับแสดงตัวแปรสมาชิกของวัตถุพร้อมทำงาน

โครงสร้างต้นไม้ของหน้าจอสำหรับแสดงตัวแปรสมาชิกของวัตถุพร้อมทำงานจะแบ่งเป็น 3 หมวดคือ หมวด Model ซึ่งเป็นหมวดของตัวแปรสมาชิกที่กำหนดให้อยู่ในคลาสโมเดล หมวด View ซึ่งเป็นหมวดของตัวแปรสมาชิกที่กำหนดให้อยู่ในคลาสวิว และหมวด SaosMain ซึ่งจะมีตัวแปรสมาชิกเพียงหนึ่งตัวคือ aosTime ซึ่งเป็นตัวแปรพร้อมทำงานของระบบซึ่งเปลี่ยนค่าตามเวลาที่ดำเนินไปในทุกๆ 1 หน่วยเวลา

6.4.4 ส่วนกำหนดเงื่อนไข

ส่วนกำหนดเงื่อนไข คือช่องเติมข้อความที่อยู่ในพื้นที่ของส่วน Condition ดังรูปที่ 6.10 ผู้ใช้สามารถกำหนดเงื่อนไขให้กับการเปลี่ยนแปลงได้โดยตรงด้วยวิธีพิมพ์เงื่อนไขที่ต้องการกำหนดในช่องสำหรับกำหนดเงื่อนไข



รูปที่ 6.10 หน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลงส่วนกำหนดเงื่อนไข

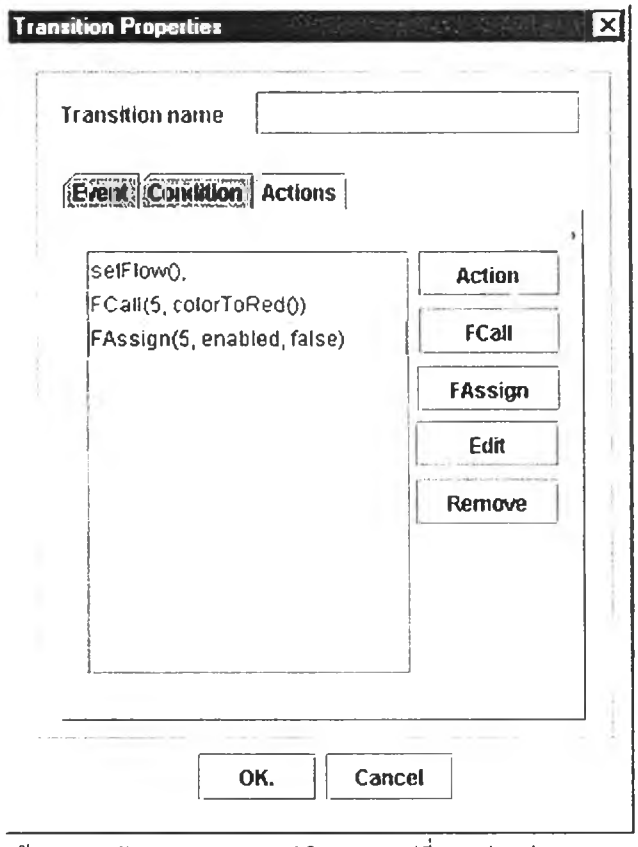
6.4.5 ส่วนกำหนดการกระทำ

ส่วนกำหนดการกระทำถูกกำหนดให้อยู่ในพื้นที่ของส่วน Actions ดังรูปที่ 6.11 ซึ่งพื้นที่ในส่วนนี้มีปุ่มสำหรับกำหนดการกระทำ 3 ปุ่มคือ ปุ่ม Action ใช้สำหรับกำหนดประโยคการกระทำ ปุ่ม FCall ใช้สำหรับกำหนดการเรียกฟังก์ชันล่วงหน้า และปุ่ม FAssign ใช้สำหรับกำหนดการกำหนดค่าล่วงหน้า สำหรับปุ่ม Edit และ ปุ่ม Remove ใช้สำหรับแก้ไขและลบการกระทำที่เลือกในรายการตามลำดับ

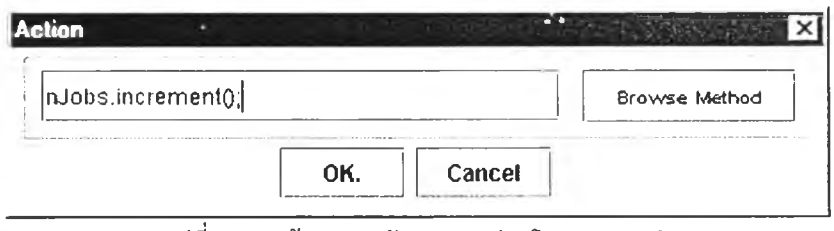
การกำหนดประโยคการกระทำทำได้โดยการกดปุ่ม Action ซึ่งจะปรากฏหน้าจอส่วนกำหนดประโยคการกระทำดังรูปที่ 6.12 ผู้ใช้สามารถพิมพ์ประโยคการกระทำที่ต้องการได้โดยตรงหรือกำหนดประโยคการเรียกใช้ฟังก์ชันโดยการเรียกใช้หน้าจอสำหรับแสดงฟังก์ชันสมาชิกซึ่งจะกล่าวถึงในหัวข้อถัดไป

การกำหนดการกระทำแบบการเรียกฟังก์ชันล่วงหน้าทำได้โดยการกดปุ่ม FCall ซึ่งจะปรากฏหน้าจอสำหรับกำหนดการเรียกฟังก์ชันล่วงหน้าดังรูปที่ 6.13 ผู้ใช้จะต้องกำหนดเวลาหน่วงในช่อง "delay time" และกำหนดชื่อของฟังก์ชันที่เรียกในช่อง "called function" โดยการพิมพ์โดยตรงหรือเรียกใช้หน้าจอสำหรับแสดงฟังก์ชันสมาชิก

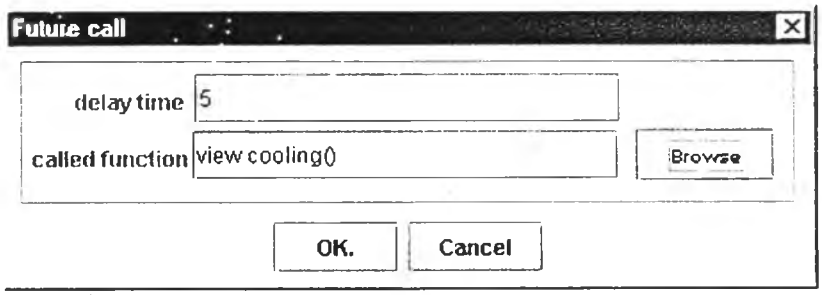
การกำหนดการกระทำแบบการกำหนดค่าล่วงหน้าทำได้โดยการกดปุ่ม FAssign ซึ่งจะปรากฏหน้าจอสำหรับกำหนดค่าล่วงหน้าดังรูปที่ 6.14 ผู้ใช้จะต้องกำหนดเวลาหน่วงในช่อง "delay time" กำหนดชื่อตัวแปรรับค่าในช่อง "the assigned variable" และกำหนดค่าในช่อง "the assigning value"



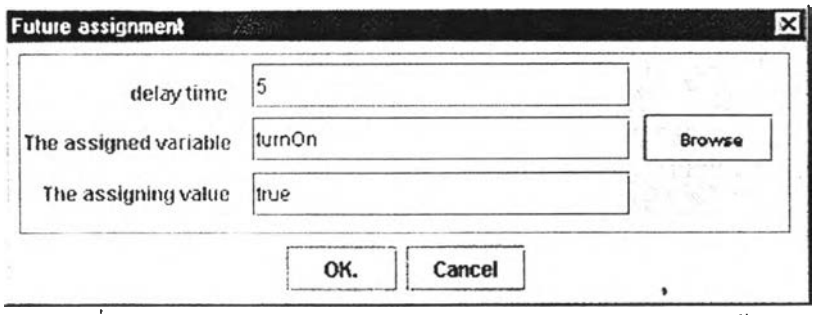
รูปที่ 6.11 หน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลงส่วนกำหนดการกระทำ



รูปที่ 6.12 หน้าจอสำหรับกำหนดประโยคการกระทำ



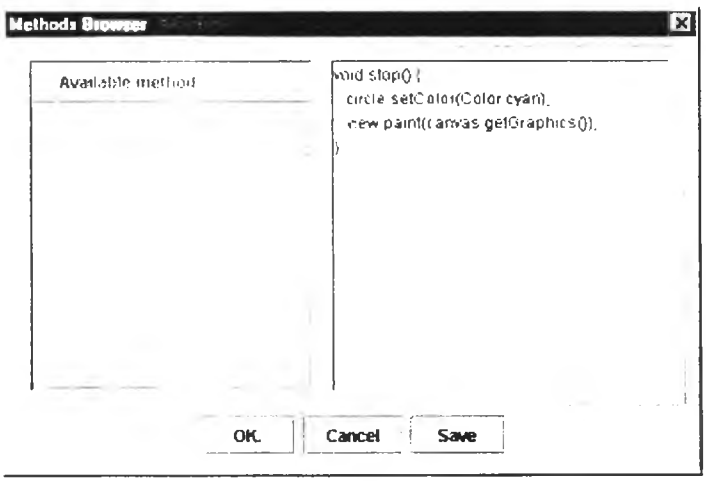
รูปที่ 6.13 หน้าจอสำหรับกำหนดการกระทำแบบการเรียกฟังก์ชันล่วงหน้า



รูปที่ 6.14 หน้าจอสำหรับกำหนดการกระทำแบบการกำหนดค่าล่วงหน้า

6.4.6 หน้าจอสำหรับแสดงรายการฟังก์ชันสมาชิก

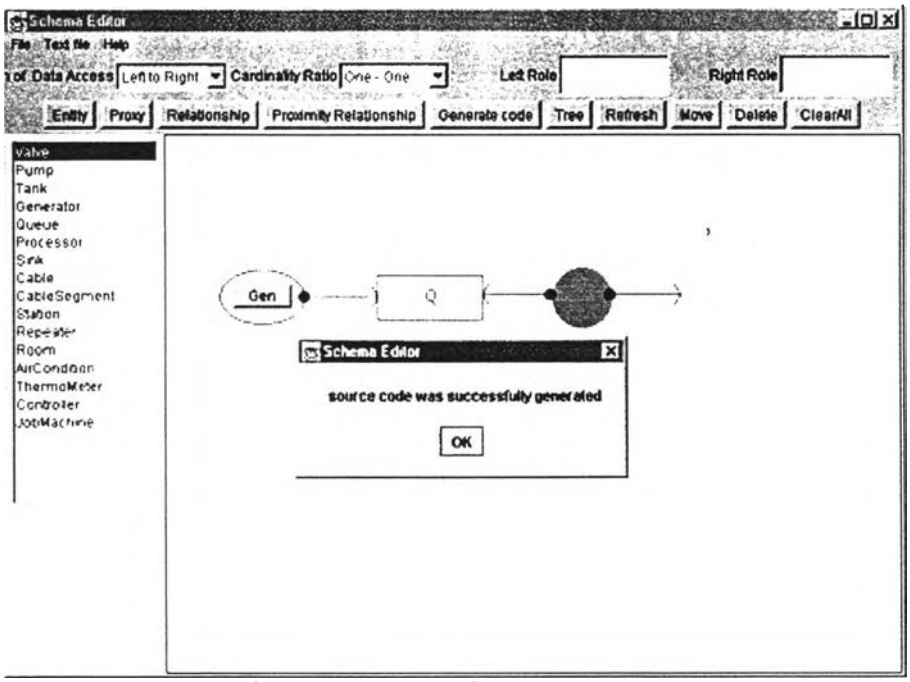
หน้าจอสำหรับแสดงรายการฟังก์ชันสมาชิกดังแสดงในรูปที่ 6.15 ใช้แสดงรายการของฟังก์ชันสมาชิกของวัตถุพร้อมทำงานแต่ละตัวโดยผู้ใช้สามารถเลือกฟังก์ชันในรายการด้านซ้ายมือเพื่อนำไปใช้กำหนดในประโยคการกระทำหรือการเรียกฟังก์ชันล่วงหน้า รายการของฟังก์ชันสมาชิกจะมีส่วนระบุว่าเป็นฟังก์ชันสมาชิกในคลาสโมเดลหรือในคลาสวิวอยู่ตอนหน้าของชื่อฟังก์ชัน นอกจากนั้นผู้ใช้ยังสามารถกำหนดรายละเอียดให้กับแต่ละฟังก์ชันสมาชิกได้ทันทีในส่วนพื้นที่ข้อความด้านขวามือ หลังจากกำหนดรายละเอียดให้กับฟังก์ชันแล้วจะต้องกดปุ่ม Save เพื่อทำการบันทึกรายละเอียดที่กำหนด รายละเอียดของฟังก์ชันสมาชิกจะอยู่ในชุดคำสั่งของวัตถุพร้อมทำงานหลังจากทำการสร้างชุดคำสั่ง



รูปที่ 6.15 หน้าจอสำหรับแสดงรายการฟังก์ชันสมาชิก

6.5 การสร้างชุดคำสั่ง

การสร้างชุดคำสั่งของวัตถุพร้อมทำงานจะทำหลังจากกำหนดแผนภาพสเตทชาร์ทให้กับชนิดของเอนทิตีแต่ละตัวในแผนภาพเอนทิตีและความสัมพันธ์เป็นที่เรียบร้อยแล้วโดยการกดที่ปุ่ม Generate Code ของบรรณาธิกรสำหรับสร้างเค้าร่าง หลังจากทำการสร้างชุดคำสั่งเสร็จเรียบร้อยแล้วจะปรากฏหน้าต่างแสดงข้อความ “source code was successfully generated” ดังรูปที่ 6.16



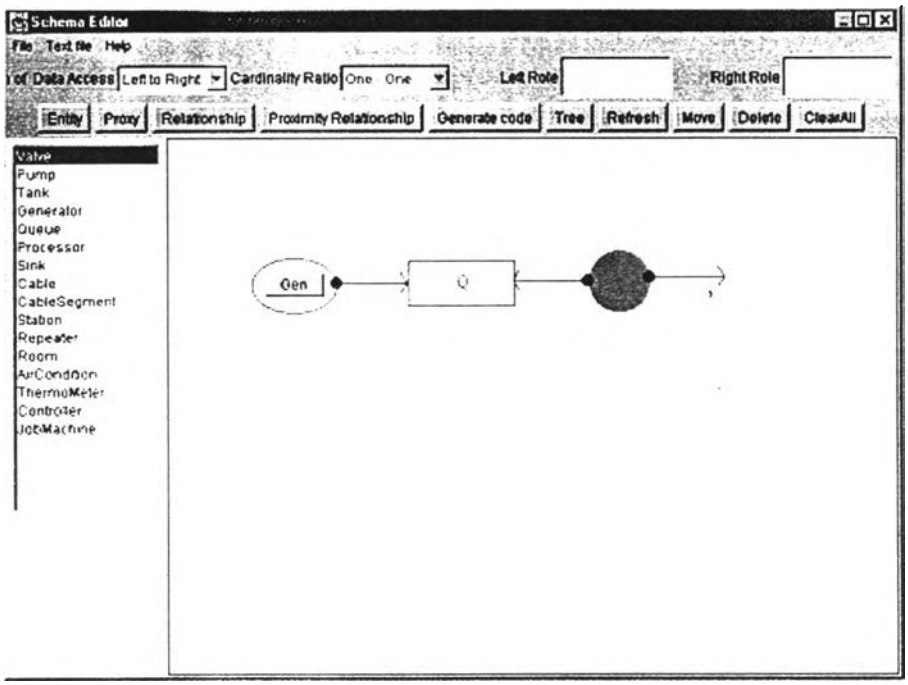
รูปที่ 6.16 การสร้างชุดคำสั่งของวัตถุพร้อมทำงาน

6.6 ตัวอย่างการสร้างโปรแกรมประยุกต์

หัวข้อนี้เป็นการแสดงตัวอย่างการสร้างโปรแกรมประยุกต์ระบบแถวคอย (Queue system)โดยใช้บรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทกำหนดพฤติกรรมให้กับแต่ละชนิดของเอนทิตีของระบบแถวคอยดังต่อไปนี้

6.6.1 สร้างแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติม

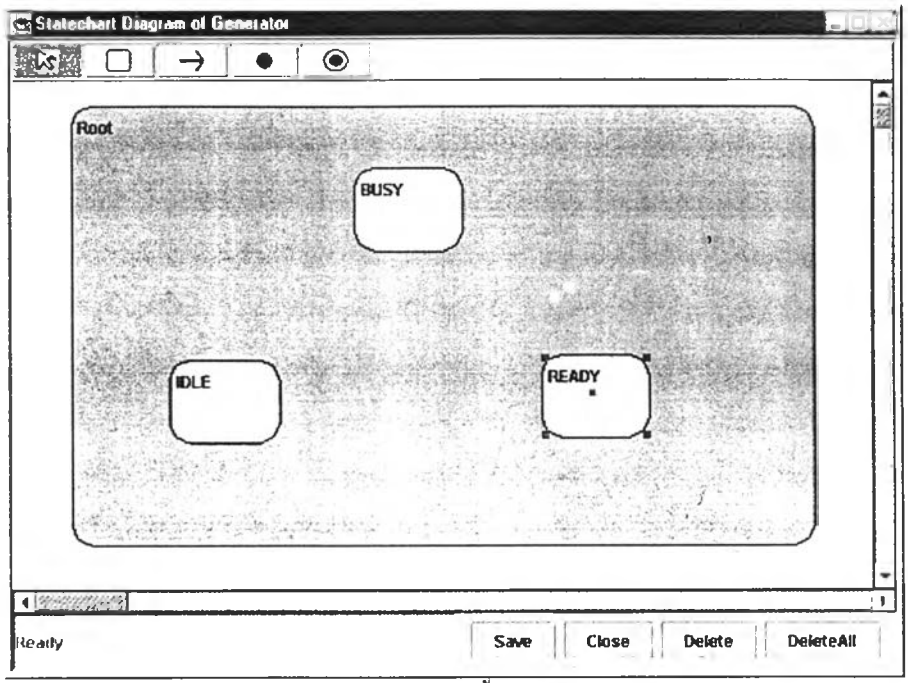
การสร้างโปรแกรมประยุกต์เริ่มต้นจากการสร้างแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติมโดยใช้บรรณาธิกรสำหรับสร้างเค้าร่างเป็นเครื่องมือในการสร้างแผนภาพ รูปที่ 6.17 เป็นแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติมของระบบแถวคอยที่ประกอบด้วย ตัวสร้างงาน (Generator) แถวคอย (Queue) และตัวประมวลผล (Processor) ซึ่งเป็นชนิดของเอนทิตีของระบบ แผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติมของระบบแถวคอยแสดงความสัมพันธ์ในการเชื่อมต่อดังนี้ ตัวสร้างงานหนึ่งตัวสามารถเชื่อมต่อกับแถวคอยได้เพียงตัวเดียว แถวคอยหนึ่งตัวสามารถเชื่อมต่อกับตัวสร้างงานได้หลายตัวและเชื่อมต่อกับตัวประมวลผลได้หลายตัว ตัวประมวลผลหนึ่งตัวสามารถเชื่อมต่อกับแถวคอยได้เพียงตัวเดียวที่ช่องนำเข้า (Input port) และตัวประมวลผลสามารถกำหนดผลลัพธ์ให้กับแถวคอยตัวถัดไปที่เชื่อมต่อกับช่องแสดงผลลัพธ์ (Output port) สำหรับทิศทางการเข้าถึงข้อมูลพิจารณาจากทิศของหัวลูกศรซึ่งแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติมของระบบแถวคอยแสดงไว้ดังนี้ ตัวสร้างงานสามารถเข้าถึงข้อมูลของแถวคอยได้ และตัวประมวลผลสามารถเข้าถึงข้อมูลของแถวคอยได้



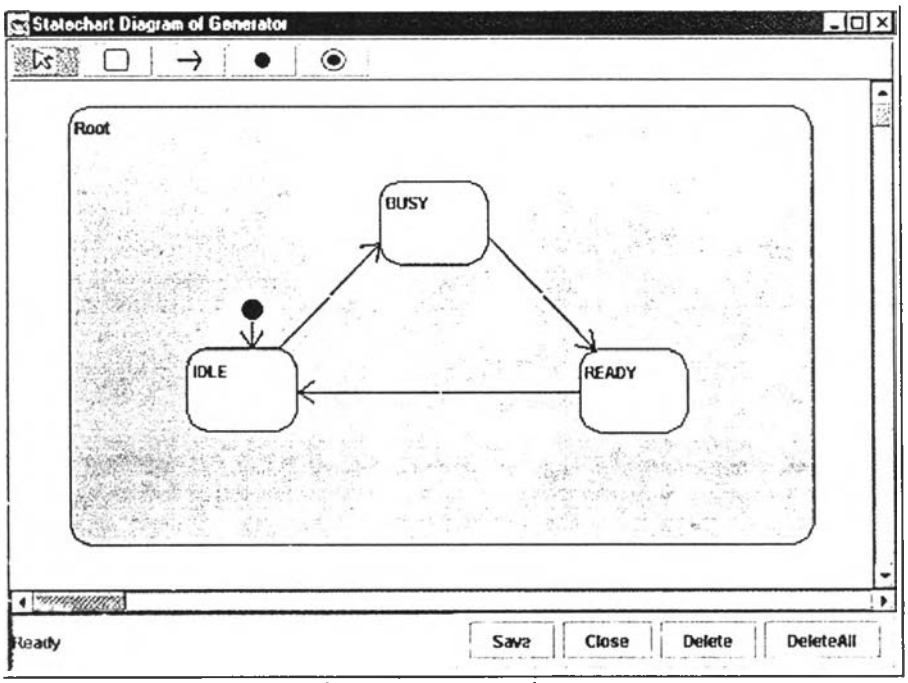
รูปที่ 6.17 การสร้างแผนภาพเอนทิตีและความสัมพันธ์ของระบบแถวคอย

6.6.2 สร้างแผนภาพสเตทชาร์ทของตัวสร้างงาน

ตัวสร้างงานคือวัตถุพร้อมทำงานที่ทำหน้าที่สร้างงานและส่งงานที่สร้างได้ให้กับแถวคอยที่กำลังเชื่อมต่ออยู่ตามเงื่อนไขและระยะเวลาที่กำหนด พฤติกรรมของตัวสร้างงานเป็นดังนี้ เมื่อผู้ใช้กดปุ่ม Gen ที่ตัวสร้างงาน ตัวสร้างงานจะใช้เวลาระยะหนึ่งในการสร้างงานใหม่ให้เกิดขึ้นในที่นี้กำหนดให้เป็น 2 หน่วยเวลา หลังจากผ่านระยะเวลาดังกล่าวไปแล้วตัวสร้างงานจะส่งงานที่สร้างได้ให้กับแถวคอย การส่งงานให้กับแถวคอยคือการเพิ่มค่าให้กับตัวแปรสมาชิก nJobs ของแถวคอย การกำหนดแผนภาพสเตทชาร์ทให้กับตัวสร้างงานเริ่มต้นจากการดับเบิลคลิกที่ชนิดของเอนทิตีของตัวสร้างงานเพื่อเรียกใช้งานบรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ท หลังจากหน้าจอของบรรณาธิกรสำหรับสร้างแผนภาพสเตทชาร์ทปรากฏขึ้นจึงทำการวาดสถานะพื้นฐานของตัวสร้างงานดังรูปที่ 6.18 สถานะของตัวสร้างงานคือ สถานะนิ่ง (IDLE) สถานะกำลังทำงาน (BUSY) และสถานะพร้อม (READY) หลังจากนั้นจึงทำการสร้างการเปลี่ยนแปลงเพื่อระบุลำดับของการเปลี่ยนสถานะของตัวสร้างงานดังรูปที่ 6.19 การระบุสถานะเริ่มต้นในแผนภาพสเตทชาร์ททำโดยวาดสถานะเริ่มต้นแล้ววาดการเปลี่ยนแปลงจากสถานะเริ่มต้นมายังสถานะนิ่งเพื่อระบุว่าสถานะนิ่งคือสถานะเริ่มต้นของตัวสร้างงาน การเปลี่ยนแปลงถัดไปคือ การเปลี่ยนแปลงจากสถานะนิ่งไปยังสถานะกำลังทำงาน การเปลี่ยนแปลงจากสถานะกำลังทำงานไปยังสถานะพร้อม และการเปลี่ยนแปลงจากสถานะพร้อมไปยังสถานะนิ่ง ตามลำดับ



รูปที่ 6.18 การสร้างสถานะพื้นฐานของตัวสร้างงาน



รูปที่ 6.19 การวาดการเปลี่ยนแปลง

ตัวอย่างนี้ระบุการเปลี่ยนแปลงภายในสองการเปลี่ยนแปลงที่สถานะรากเพื่อใช้ตอบสนองต่อเหตุการณ์ที่ผู้ใช้กดปุ่มของตัวสร้างงานซึ่งมีข้อความการเปลี่ยนแปลงดังนี้

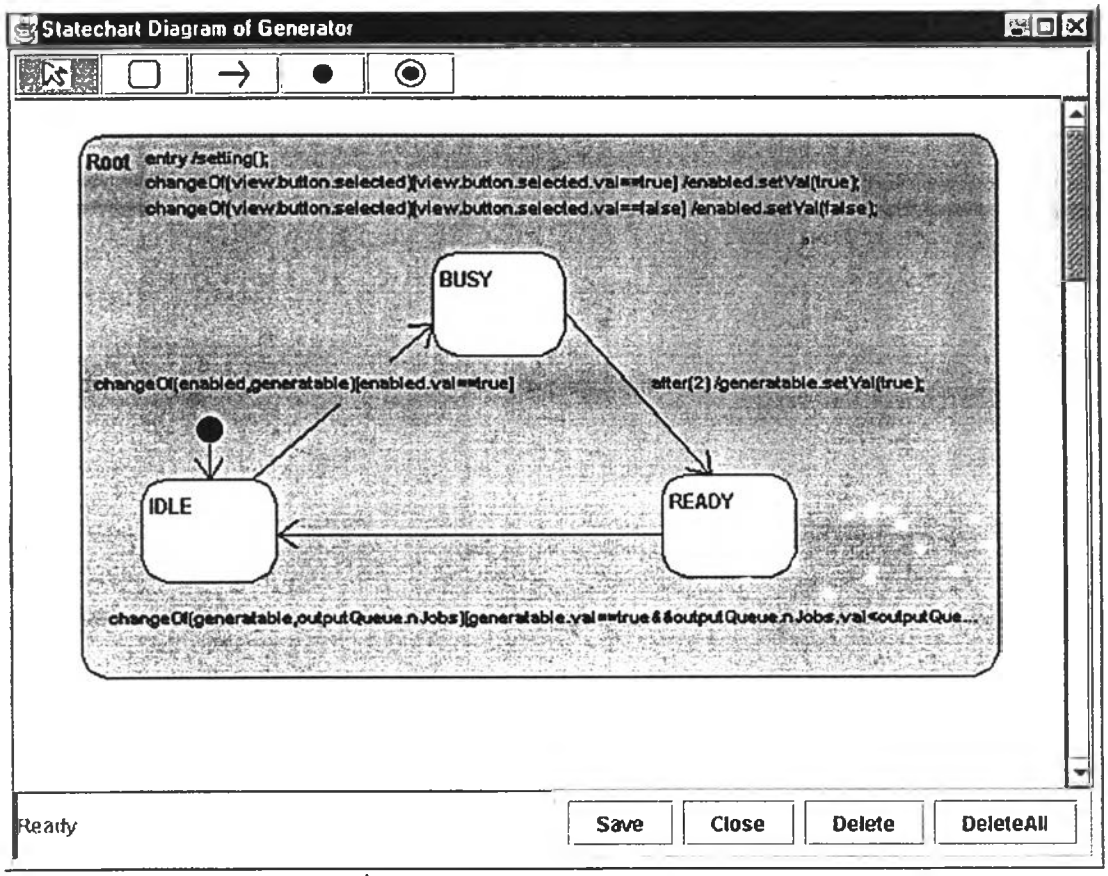
1) changeOf(view.button.selected) [view.button.selected.val==true] / enable.setVal(true)
2) changeOf(view.button.selected) [view.button.selected.val==false] / enable.setVal(false)

การเปลี่ยนแปลงทั้งสองระบุด้วยเหตุการณ์ประเภทการเปลี่ยนค่าของตัวแปรพร้อมทำงาน selected ซึ่งใช้สัญลักษณ์ changeOf ในการระบุชนิดของเหตุการณ์ การกดปุ่มแต่ละครั้งตัวแปรพร้อมทำงาน selected จะเปลี่ยนแปลงค่าไปมาระหว่าง true และ false ดังนั้นการเปลี่ยนค่าของตัวแปรพร้อมทำงาน selected เป็นการทำให้เกิดเหตุการณ์ขึ้นมากระตุ้นการเปลี่ยนแปลงทั้งสอง เมื่อการเปลี่ยนแปลงทั้งสองถูกกระตุ้นส่วนเงื่อนไขจะถูกตรวจสอบ ถ้าเงื่อนไขของการเปลี่ยนแปลงใดเป็นจริงเป็นจริงส่วนการกระทำของการเปลี่ยนแปลงนั้นจะถูกประมวลผลซึ่งมีรายละเอียดดังนี้ ถ้าค่าของตัวแปรพร้อมทำงาน selected เป็น true จะทำการกำหนดให้ตัวแปรพร้อมทำงาน enable เป็น true ถ้าค่าของตัวแปรพร้อมทำงาน selected เป็น false จะทำการกำหนดให้ตัวแปรพร้อมทำงาน enable เป็น false การกำหนดเงื่อนไขและการกระทำให้กับแต่ละการเปลี่ยนแปลงจะต้องระบุในส่วนเงื่อนไขและส่วนการกระทำของหน้าจอกำหนดคุณสมบัติของการเปลี่ยนแปลงดังที่ได้อธิบายไว้แล้วในหัวข้อ 6.44 และ 6.45

การกำหนดคุณสมบัติให้กับการเปลี่ยนแปลงภายนอกที่สร้างไว้แล้วทำโดยการดับเบิลคลิกที่เส้นหัวลูกศรเพื่อเรียกหน้าจอสำหรับกำหนดคุณสมบัติของการเปลี่ยนแปลง ทำการกำหนดคุณสมบัติให้กับแต่ละการเปลี่ยนแปลงดังที่ได้อธิบายไว้แล้ว ข้อความการเปลี่ยนแปลงที่กำหนดให้กับแต่ละการเปลี่ยนแปลงในแผนภาพสเตทชาร์ทเป็นดังตารางที่ 6.1 ตัวสร้างงานจะเปลี่ยนสถานะจากสถานะนิ่งไปยังสถานะกำลังทำงาน เมื่อตัวแปรพร้อมทำงาน enable หรือ generatable มีการเปลี่ยนแปลงค่าและค่าของตัวแปรพร้อมทำงาน enable เท่ากับ true เมื่อมีการเปลี่ยนสถานะมายังสถานะกำลังทำงานแล้วหลังจากนั้น 2 หน่วยเวลาจะทำการกำหนดค่าให้กับตัวแปรพร้อมทำงาน generatable เป็น true แล้วเปลี่ยนสถานะไปยังสถานะพร้อม เมื่อสถานะเป็นพร้อมถ้าค่าของตัวแปรพร้อมทำงาน generatable เป็น true และจำนวนงานในแถวคอยน้อยกว่าความจุของแถวคอยจะทำการจะทำการกำหนดค่าให้กับตัวแปรพร้อมทำงาน generatable เป็น false และเพิ่มจำนวนงานให้กับแถวคอยหลังจากนั้นจึงเปลี่ยนสถานะไปยังสถานะนิ่งอีกครั้งหนึ่ง แผนภาพสเตทชาร์ทของตัวสร้างงานที่ได้กำหนดคุณสมบัติให้กับทุกการเปลี่ยนแปลงเรียบร้อยแล้วแสดงในรูปที่ 6.20

ตารางที่ 6.1 ข้อความการเปลี่ยนแปลงของการเปลี่ยนแปลงภายนอกในแผนภาพสเตทชาร์ทของตัวสร้างงาน

การเปลี่ยนแปลงจาก		ข้อความการเปลี่ยนแปลง
สถานะตั้งต้น	สถานะเป้าหมาย	
IDLE	BUSY	**changeOf**(enable,generatable) [enable.val==true]
BUSY	READY	**after**(2)/generatable.setVal(true)
READY	IDLE	**changeOf**(generatable,outputQueue.nJobs) [generatable.val==true && outputQueue.nJobs.val< outputQueue.nSlots] / generatable.setVal(false) , outputQueue.nJobs.increment()



รูปที่ 6.20 แผนภาพสเตทชาร์ทของตัวสร้างงาน

6.6.3 แผนภาพสเตทชาร์ทของแถวคอย

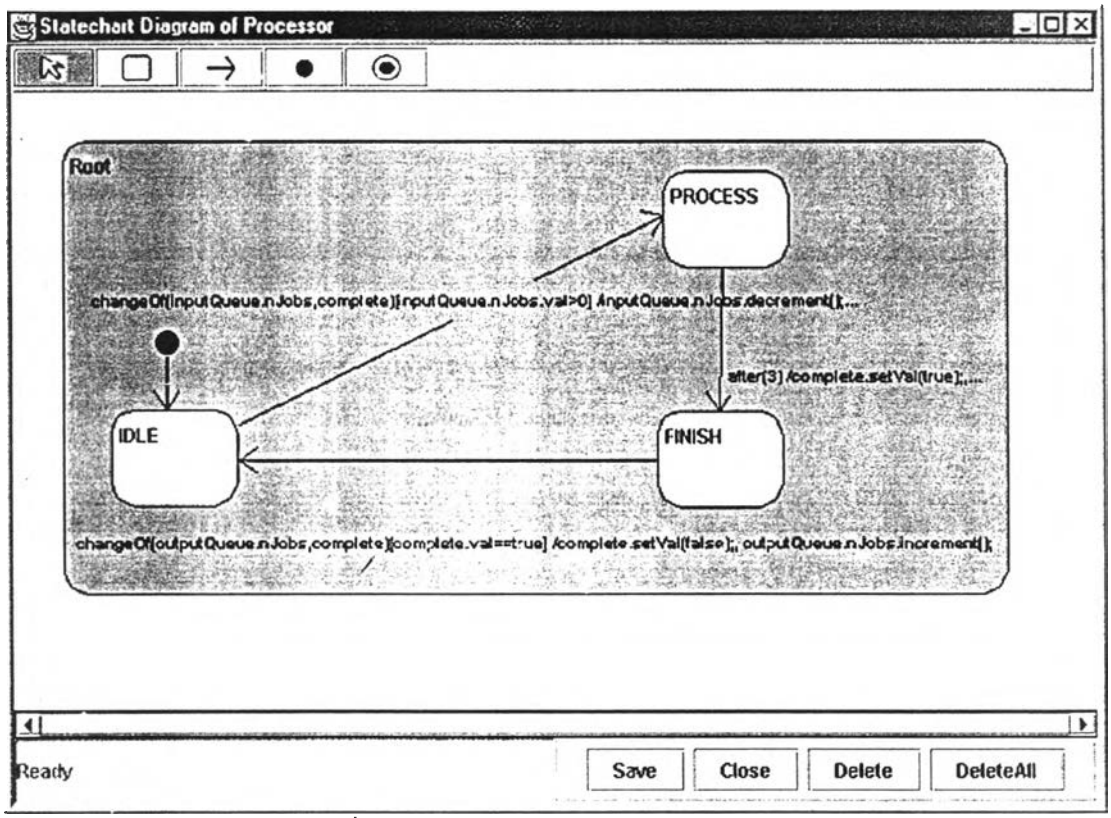
แถวคอยคือวัตถุพร้อมทำงานที่ทำหน้าที่เป็นตัวกลางระหว่างตัวสร้างงานและตัวประมวลผลโดยรับงานที่สร้างจากตัวสร้างงานและคอยให้ตัวประมวลผลนำงานในแถวคอยไปทำการประมวลผล ในโปรแกรมประยุกต์นี้ไม่มีการกำหนดแผนภาพสเตทชาร์ทให้กับแถวคอย

6.6.4 แผนภาพสเตทชาร์ทของตัวประมวลผล

ตัวประมวลผลคือวัตถุพร้อมทำงานที่ทำหน้าที่นำงานที่อยู่ในแถวคอยไปทำการประมวลผลแล้วกำหนดงานที่ประมวลผลเสร็จให้กับแถวคอยตัวถัดไป พฤติกรรมของตัวประมวลผลเป็นดังนี้ เมื่อมีงานเข้ามายังแถวคอยที่เชื่อมต่ออยู่กับตัวประมวลผล ตัวประมวลผลจะนำงานออกจากแถวคอยเพื่อทำการประมวลผล การประมวลผลแต่ละครั้งของตัวประมวลผลจะใช้ระยะเวลาช่วงหนึ่งซึ่งในตัวอย่างนี้กำหนดให้เป็น 3 หน่วยเวลา เมื่อผ่านระยะเวลาดังกล่าวไปแล้วการประมวลผลจะเสร็จสิ้นซึ่งตัวประมวลผลจะนำงานที่ประมวลผลเสร็จแล้วส่งให้กลับแถวคอยตัวถัดไปที่เชื่อมต่ออยู่ สถานะของตัวประมวลผลที่กำหนดไว้ได้แก่ สถานะนิ่ง (IDLE) สถานะประมวลผล (PROCESS) และสถานะเสร็จ (FINISH) ซึ่งวิธีการสร้างสถานะและการเปลี่ยนแปลงทำเช่นเดียวกับตัวสร้างงานในหัวข้อที่ผ่านมา ข้อความการเปลี่ยนแปลงที่กำหนดให้กับการเปลี่ยนแปลงแสดงในตารางที่ 6.2 และแผนภาพสเตทชาร์ทของตัวประมวลผลแสดงในรูปที่ 6.21

ตารางที่ 6.2 ข้อความการเปลี่ยนแปลงของการเปลี่ยนแปลงภายนอกในแผนภาพสเตทชาร์ทของตัวประมวลผล

การเปลี่ยนแปลงจาก		ข้อความการเปลี่ยนแปลง
สถานะตั้งต้น	สถานะเป้าหมาย	
IDLE	PROCESS	**changeOf**(inputQueue.nJobs,complete) [inputQueue.nJobs.val>0]/ inputQueue.nJobs.decrement()
PROCESS	FINISH	**after**(3)/complete.setVal(true)
FINISH	IDLE	**changeOf**(outputQueue.nJobs,complete)[complete.val==true && outputQueue.nJobs.val< outputQueue.nSlots] / complete.setVal(false) , outputQueue.nJobs.increment()



รูปที่ 6.21 แผนภาพสเตทชาร์ทของตัวประมวลผล

6.6.5 การสร้างชุดคำสั่งของระบบแถวคอย

การสร้างชุดคำสั่งของระบบแถวคอยทำหลังจากสร้างแผนภาพสเตทชาร์ทให้กับชนิดของเอนทิตีแต่ละตัวของระบบเรียบร้อยแล้วโดยการกดปุ่ม Generate Code ของบรรณาธิกรสำหรับสร้างเค้าร่าง ตัวสร้างชุดคำสั่งจะทำการแปลงแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติมและแผนภาพสเตทชาร์ทไปเป็นแฟ้มข้อมูลของชุดคำสั่ง ซึ่งชุดคำสั่งที่สร้างได้แบ่ง 2 ประเภทดังนี้

1) ชุดคำสั่งข้อกำหนดโปรแกรมประยุกต์

ชุดคำสั่งข้อกำหนดโปรแกรมประยุกต์คือชุดคำสั่งที่ทำหน้าที่สร้างเอนทิตีตามเค้าร่างที่กำหนดไว้และกำหนดเอนทิตีที่สร้างให้กับระบบของวัตถุพร้อมทำงาน ชุดคำสั่งข้อกำหนดโปรแกรมประยุกต์ของระบบแถวคอยแสดงในรูปที่ 6.22

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

class QueueSystem extends AObject
{
 public QueueSystem()
 {
  super("QueueSystem");
 }

 public EditApp createObject(int index, EditObject object, AppCanvas canvas, int x, int y)
 {
  try
  {
   switch (index)
   {
    case 0:
     EditGenerator tempGen =
                new EditGenerator( (EditCompound) (object), canvas, x, y);
     insert(tempGen, true);
     return tempGen;
    case 1:
     EditQueue tempQue =
                new EditQueue( (EditCompound) (object), canvas, x, y);
     insert(tempQue, true);
     return tempQue;
    case 2:
     EditProcessor tempPro =
                new EditProcessor( (EditCompound) (object), canvas, x, y);
     insert(tempPro, true);
     return tempPro;
    case 3:
     EditQueue tempQ =
                new EditQueue( (EditCompound) (object), canvas, x, y);
     insert(tempQ, true);
          return tempQ;
    default: return null;
   }
  } catch (SaosException ie)
  {
   System.out.println("ERROR: " + ie);
  }
  return null;
 }

}
```

รูปที่ 6.22 ชุดคำสั่งข้อกำหนดโปรแกรมประยุกต์ของระบบแถวคอย

2) ชุดคำสั่งของชนิดเอนทิตี

ชุดคำสั่งของชนิดเอนทิตีคือชุดคำสั่งที่ควบคุมการทำงานของเอนทิตีแต่ละตัวซึ่งมีทั้งคลาสวิวและคลาสโมเดล ดังตัวอย่างของชุดคำสั่งของตัวสร้างงานในรูปที่ 6.23

```
[ชุดคำสั่งของคลาสโมเดล]
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;
 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&&outputQueue.nJobs.val<outputQueue.nSlots&&( 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);
   }
 }
 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);
   .....
```

รูปที่ 6.23 ชุดคำสั่งของตัวสร้างงาน

```
    .....
    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();
  }

  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);
  }

}
```

[ชุดคำสั่งของคลาสวิว]

```
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");
  .....
```

รูปที่ 6.23 ชุดคำสั่งของตัวสร้างงาน (ต่อ)

```
  .....
  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();
 }

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

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

 public void dispatch(int funcIndex)
 {
 }

 public void delete()
 {
    view.delete();
 }
}
```

รูปที่ 6.23 ชุดคำสั่งของตัวสร้างงาน (ต่อ)