บทที่ 2

งานวิจัยและทฤษฎีที่เกี่ยวข้อง

2.1 ระบบวัตถุพร้อมทำงานแบบโครงสร้าง (Structural Active Object System : SAOS)

Toshimi Minoura และ Sungwoon Choi ได้เสนอแนวคิดของ "ระบบวัตถุพร้อมทำงานแบบโครงสร้าง" [1,2] และได้พัฒนาเป็นเฟรมเวิร์คเพื่อนำไปใช้ในการพัฒนาซอฟต์แวร์ที่ประกอบด้วยวัตถุพร้อมทำงาน ความหมายของวัตถุพร้อมทำงานคือวัตถุที่มีส่วนควบคุมเป็นของตนเอง ดังนั้นวัตถุพร้อมทำงานจึงสามารถทำงานได้ด้วยตัวเองโดยที่ไม่ต้องรอข้อความร้องขอจากวัตถุตัวอื่น ซึ่งเป็นข้อดีคือทำให้การทำงานของระบบที่ประกอบด้วยวัตถุพร้อมทำงานเป็นการทำงานที่เกิดขึ้นแบบพร้อมๆ กัน สามารถแสดงตัวอย่างของวัตถุพร้อมทำงานในความเป็นจริงด้วยการทำงานของสัญญาณไฟจราจร ซึ่งจะต้องมีตัวจับเวลาในตัวเองเพื่อคอยจับเวลาในการเปลี่ยนสีของสัญญาณไฟและส่วนควบคุมคอยตรวจสอบว่าจะต้องเปลี่ยนเป็นสีใด เมื่อเขียนโปรแกรมเพื่อจำลองการทำงานของสัญญาณไฟนี้ในรูปแบบวัตถุที่คอยตอบสนองต่อคำร้องขอ นอกจากจะต้องมีตัวสัญญาณไฟจราจรแล้วจะต้องสร้างวัตถุอีกตัวหนึ่งขึ้นมาเพื่อใช้เป็นตัวควบคุมโดยคอยส่งข้อความให้สัญญาณไฟเปลี่ยนสีตามช่วงเวลาที่กำหนด แต่ถ้านำมาเขียนโปรแกรมแบบวัตถุพร้อมทำงานจะมีรูปแบบที่แตกต่างกันคือ การควบคุมการเปลี่ยนสัญญาณไฟจะอยู่ภายในตัววัตถุที่เป็นสัญญาณไฟเองและสามารถทำงานได้เองโดยไม่ต้องรอข้อความร้องขอจากวัตถุที่เป็นตัวควบคุม พฤติกรรมของวัตถุพร้อมทำงานสามารถกำหนดได้โดยใช้ประโยคแสดงการเปลี่ยนแปลง (Transition statement) ซึ่งแบ่งออกได้เป็น 3 แบบได้แก่

2.1.1 กฎการเปลี่ยนแปลง (Transition rules)

การกำหนดพฤติกรรมในแบบของกฎการเปลี่ยนแปลงจะต้องมีองค์ประกอบ 2 ส่วนคือ ส่วนเงื่อนไข (Condition) และส่วนของการกระทำ (Action) โดยมีตัวแปรพร้อมทำงาน (Active variable) เป็นตัวเริ่มต้นการทำงานให้กับกฎการเปลี่ยนแปลงดังนี้ เมื่อตัวแปรพร้อมทำงานเกิดการเปลี่ยนแปลงค่าเงื่อนไขที่กำหนดไว้จะถูกตรวจสอบและทำการประมวลผลส่วนของการกระทำถ้าเงื่อนไขนั้นเป็นจริง

ตัวอย่างของกฎการเปลี่ยนแปลงแสดงได้จากพฤติกรรมของตัวสร้างงาน (Generator) ซึ่งเป็นวัตถุพร้อมทำงานที่เป็นองค์ประกอบในระบบแถวคอย (Queue system) ตัวสร้างงานมีตัวแปรพร้อมทำงานชื่อ state เป็นตัวระบุสถานะในปัจจุบัน ตัวสร้างงานมีหน้าที่ป้อนงานให้เข้าสู่แถวคอยตามช่วงเวลาและเงื่อนไขที่กำหนด ดังนั้นจึงสามารถกำหนดเงื่อนไขและการกระทำของตัวสร้างงานได้ดังนี้

เงื่อนไข : เมื่อสถานะของตนเองเป็น "พร้อม" (Ready)

การกระทำ : ป้อนงานสู่แถวคอย

เมื่อสถานะของตัวสร้างงานเปลี่ยนซึ่งหมายถึงตัวแปรพร้อมทำงาน state มีการเปลี่ยนแปลงค่า ระบบจะตรวจสอบเงื่อนไขที่กำหนดไว้ ถ้าเงื่อนไขเป็นจริงการกระทำ "ป้อนงานสู่แถวคอย" จะถูกประมวลผล

2.1.2 สมการกำหนดค่า (Equational assignment)

การกำหนดพฤติกรรมในแบบสมการกำหนดค่าจะคล้ายกับกฎการเปลี่ยนแปลงแต่จะต่างกันที่จะต้องกำหนดให้อยู่ในรูปสมการกำหนดค่าซึ่งประกอบด้วยตัวแปรรับค่าและประโยคการคำนวณ ซึ่งจะต้องประกอบด้วยตัวแปรพร้อมทำงานอย่างน้อยหนึ่งตัว เมื่อไรก็ตามที่ตัวแปรพร้อมทำงานในประโยคการคำนวณมีการเปลี่ยนแปลงค่า ระบบจะทำการประมวลผลประโยคการคำนวณเพื่อคำนวณผลลัพธ์ให้กับตัวแปรรับค่า

ตัวอย่างของสมการกำหนดค่าคือพฤติกรรมของแอนเกต (AND gate) ที่ประกอบด้วยตัวแปรที่เป็นข้อมูลเข้า 2 ตัวแปร ได้แก่ "input1" และ "input2" ในที่นี้ตัวแปรทั้งสองถูกกำหนดให้เป็นตัวแปรพร้อมทำงาน และตัวแปรที่เป็นผลลัพธ์ของแอนเกตคือ "output" ซึ่งอาจมีค่าเป็นจริงหรือเท็จก็ได้ ขึ้นอยู่กับการคำนวณของประโยค "input1 && input2" เมื่อกำหนดเป็นสมการกำหนดค่าสามารถเขียนได้ดังนี้

"output = input1 && input2"

เมื่อไรก็ตามที่ "input1" หรือ "input2" เปลี่ยนแปลงค่าจะต้องมีการคำนวณผลลัพธ์ใหม่ทุกครั้งเพื่อกำหนดค่าที่ถูกต้องให้กับ "output"

2.1.3 ส่วนกำหนดเหตุการณ์ (Event routines)

พฤติกรรมแบบส่วนกำหนดเหตุการณ์อยู่ในรูปของ การเรียกฟังก์ชันล่วงหน้า และการกำหนดค่าล่วงหน้า การเรียกฟังก์ชันล่วงหน้าเป็นการเรียกฟังก์ชันโดยมีการกำหนดเวลาหน่วง โดยที่ฟังก์ชันดังกล่าวจะถูกประมวลผลก็ต่อเมื่อเวลาผ่านไปเท่ากับที่กำหนดไว้ตามเวลาหน่วง ส่วนการกำหนดค่าล่วงหน้าก็มีลักษณะการทำงานที่คล้ายกันแต่เป็นการกำหนดค่าให้กับตัวแปรแทนที่จะเป็นการเรียกฟังก์ชัน

ตัวอย่างของการกำหนดค่าล่วงหน้าสามารถแสดงได้จากพฤติกรรมของสัญญาณไฟจราจรที่เปลี่ยนสีไปตามเวลาที่กำหนดเช่น ในขณะที่สีของไฟจราจรเป็นสีแดงอาจกำหนดให้ค่าของสีเปลี่ยนเป็นสีเหลืองภายในอีก 2 วินาทีข้างหน้า เมื่อไฟเป็นสีเหลืองกำหนดค่าของสีเป็นสีเขียวในอีก 1 วินาทีข้างหน้า และเมื่อไฟเป็นสีเขียวให้เปลี่ยนเป็นสีแดงในอีก 3 วินาทีข้างหน้าเป็นต้น

ตัวอย่างของการเรียกฟังก์ชันล่วงหน้าสามารถแสดงได้จากพฤติกรรมของเครื่องทวนสัญญาณ (Repeater) ในระบบเครือข่ายซึ่งจะทำการประมวลผลฟังก์ชัน "fromPort1ToPort2()" ในอีก 5 หน่วยเวลาข้างหน้าเมื่อมีสัญญาณเข้ามายังที่เครื่องทวนสัญญาณ

กลไกสำคัญในส่วนควบคุมของระบบวัตถุพร้อมทำงานคือ การใช้ตัวแปรพร้อมทำงานเป็นตัวเริ่มการทำงาน (Trigger element) ของฟังก์ชันสมาชิกของวัตถุพร้อมทำงาน ตัวแปรพร้อมทำงานคือตัวแปรที่มีชนิดเป็นคลาส "AInteger" "ABoolean" หรือ "ADouble" ซึ่งใช้เก็บข้อมูลตามชนิดของข้อมูลที่ต้องการ สำหรับกลไกเริ่มการทำงานคือวิธีการในการเชื่อมโยงความสัมพันธ์ระหว่างตัวแปรพร้อมทำงานกับฟังก์ชันสมาชิกของวัตถุพร้อมทำงาน ทุกครั้งที่ตัวแปรพร้อมทำงานมีการเปลี่ยนแปลงค่าจะมีการเรียกฟังก์ชันสมาชิกที่ได้สร้างความสัมพันธ์ไว้ให้ทำงาน การเปลี่ยนค่าให้กับตัวแปรพร้อมทำงานสามารถทำ

ได้โดยกำหนดค่าผ่านฟังก์ชัน "setVal()" ตัวอย่างเช่น runnable.setVal(true) คือการกำหนดให้ตัวแปรพร้อมทำงาน runnable มีค่าเป็นจริง หรือเรียกการทำงานของฟังก์ชัน "decrement()" หรือ "increment()" ซึ่งเป็นการลดค่าหรือเพิ่มค่าของตัวแปรพร้อมทำงานอีกหนึ่งหน่วยตามลำดับ ในระบบของวัตถุพร้อมทำงานแบบโครงสร้างตัวแปรพร้อมทำงานจะทำหน้าที่เป็นส่วนติดต่อกับวัตถุอื่นจึงถูกเรียกว่า "ตัวแปรสำหรับติดต่อ" (interface variable) ซึ่งแตกต่างจากวัตถุที่คอยตอบสนองต่อคำร้องขอที่มีฟังก์ชันสมาชิกเป็นส่วนติดต่อกับวัตถุอื่น ดังนั้นการติดต่อกับวัตถุที่คอยตอบสนองต่อคำร้องขอจึงเป็นการเรียกฟังก์ชันสมาชิกมาทำการประมวลผลแต่การติดต่อกับวัตถุพร้อมทำงานเป็นการกำหนดค่าให้กับตัวแปรพร้อมทำงานซึ่งมีข้อดีคือ ทำให้สะดวกต่อการนำองค์ประกอบมาใช้ประกอบกันแบบโครงสร้างเพื่อสร้างซอฟต์แวร์ เพราะรูปแบบของการประกอบโดยให้แต่ละองค์ประกอบติดต่อกันผ่านตัวแปรสำหรับติดต่อนั้นมีความใกล้เคียงกับวิธีการประกอบองค์ประกอบของฮาร์ดแวร์ ซึ่งเป็นวิธีการประกอบที่คุ้นเคยต่อผู้พัฒนา

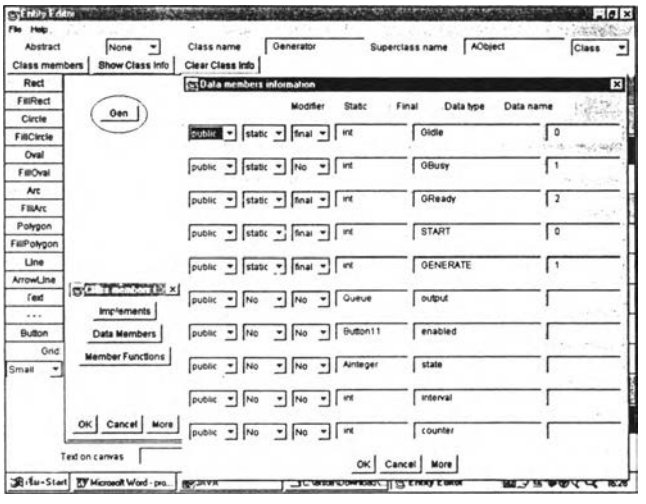
ส่วนควบคุมของระบบวัตถุพร้อมทำงาน (SAOS runtime kernel) ถูกพัฒนาขึ้นในครั้งแรกด้วยภาษาซีพลัสพลัส (C++) โปรแกรมประยุกต์ที่ประกอบด้วยระบบวัตถุพร้อมทำงานจะต้องทำงานอยู่ภายใต้ส่วนควบคุมนี้ซึ่งมีหน้าที่ในการกำกับดูแลวัตถุทุกตัวที่อยู่ในระบบให้ทำงานได้ตามพฤติกรรมที่ได้กำหนดไว้ ต่อมา Yanbing Lu ได้ทำการแปลงส่วนควบคุมของระบบวัตถุพร้อมทำงานและโปรแกรมประยุกต์ของระบบวัตถุพร้อมทำงานจากภาษาซีพลัสพลัส เป็นภาษาจาวาเพื่อให้โปรแกรมสามารถทำงานได้บนเวบเบาว์เซอร์ที่รองรับโปรแกรมของภาษาจาวา

2.2 สิ่งแวดล้อมสำหรับพัฒนาโปรแกรมด้วยแผนภาพเอนทิตีและความสัมพันธ์

ในงานวิจัยของ พรศิริ หมื่นไชยศรี ได้ทำการรวบรวมแนวคิดของการเชื่อมต่อองค์ประกอบของซอฟต์แวร์แบบมองเห็นภาพ (A visual software component composition) แผนภาพเอนทิตีและความสัมพันธ์ (Entity-Relationship Diagram) และวัตถุพร้อมทำงาน (Active object) เข้าด้วยกันเพื่อทำการพัฒนาสิ่งแวดล้อมสำหรับพัฒนาโปรแกรมประยุกต์ที่เรียกว่า "สิ่งแวดล้อมสำหรับพัฒนาโปรแกรมด้วยแผนภาพเอนทิตีและความสัมพันธ์" (Entity-Relationship Software Development Environment: ERSDE) [3] โดยมุ่งเน้นให้ผู้พัฒนามีแนวทางในการเชื่อมต่อองค์ประกอบซอฟต์แวร์จาก แผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติม (Extended Entity-Relationship Diagram: EERD) ซึ่งในที่นี้ถูกใช้เป็นตัวแบบและรายการเลือกสำหรับสร้างโปรแกรมประยุกต์ ขั้นตอนการพัฒนาโปรแกรมประยุกต์ในสิ่งแวดล้อมสำหรับพัฒนาโปรแกรมด้วยแผนภาพเอนทิตีและความสัมพันธ์มี 3 ขั้นตอนคือ สร้างชนิดของเอนทิตี สร้างเค้าร่าง และสร้างโปรแกรมประยุกต์ ในแต่ละขั้นตอนใช้บรรณาธิกรที่แตกต่างกันได้แก่ บรรณาธิกรสำหรับสร้างชนิดของเอนทิตี (Entity-type editor) บรรณาธิกรสำหรับสร้างเค้าร่าง (Schema editor) และบรรณาธิกรสำหรับสร้างโปรแกรมประยุกต์ (Application editor) บรรณาธิกรแต่ละชนิดนั้นจะถูกใช้ในงานที่แตกต่างกันโดยมีรายละเอียดดังต่อไปนี้

2.2.1 บรรณาธิกรสำหรับสร้างชนิดของเอนทิตี

บรรณาธิกรนี้ใช้สำหรับสร้างชนิดของเอนทิตี โดยที่ผู้พัฒนาสามารถทำการสร้างรูปภาพสัญลักษณ์เพื่อใช้แทนชนิดของเอนทิตีแต่ละตัวพร้อมทั้งกำหนดข้อมูลสมาชิกและฟังก์ชันสมาชิกให้กับชนิดของเอนทิตีนั้นได้ หน้าจอของบรรณาธิกรสำหรับสร้างชนิดของเอนทิตีถูกแสดงไว้ในรูปที่ 2.1



รูปที่ 2.1 บรรณาธิกรสำหรับสร้างชนิดของเอนทิตี

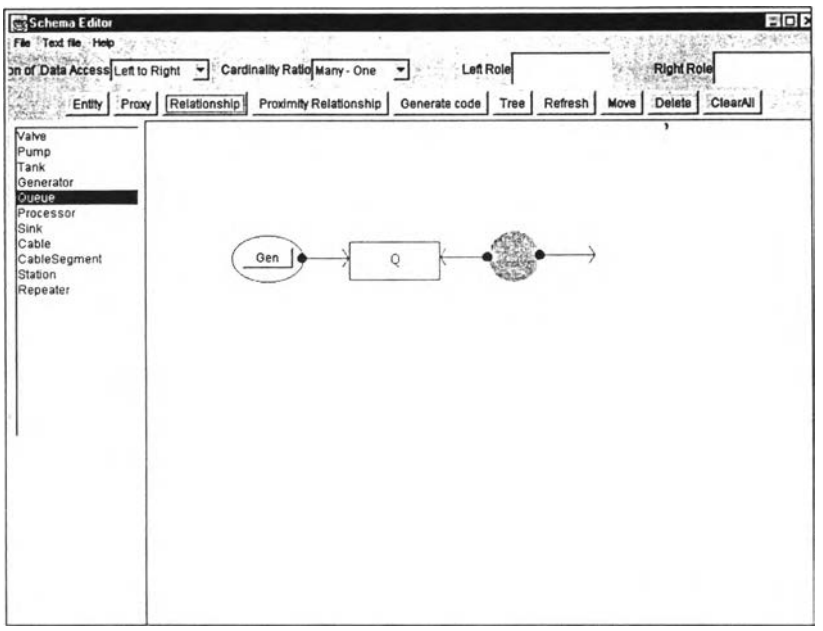
2.2.2 บรรณาธิกรสำหรับสร้างเค้าร่าง

บรรณาธิกรสำหรับสร้างเค้าร่างใช้สำหรับสร้างแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติมของแต่ละขอบเขตการสร้างโปรแกรมประยุกต์ โดยใช้ชนิดของเอนทิตีต่างๆ ที่ถูกสร้างมาจากบรรณาธิกรสำหรับสร้างชนิดของเอนทิตีมาทำการเชื่อมโยงชนิดของความสัมพันธ์ (Relationship type) ระหว่างชนิดของเอนทิตีเหล่านั้นเข้าด้วยกันโดยกำหนดคาร์ดินัลลิตีเรโช (Cardinality ratios) ทิศทางการเข้าถึงข้อมูล (Direction of data access) และบทบาทของความสัมพันธ์ (Role) ในรูปที่ 2.2 เป็นการแสดงหน้าจอของบรรณาธิกรสำหรับสร้างเค้าร่างที่ได้ทำการสร้างแผนภาพเอนทิตีและความสัมพันธ์ที่ถูกต่อเติมของระบบแถวคอย ซึ่งมีชนิดของเอนทิตีที่ปรากฏในแผนภาพเอนทิตีและความสัมพันธ์ คือ

- ตัวสร้างงาน (Generator) ซึ่งใช้สัญลักษณ์รูปวงรีที่มีปุ่ม “Gen” ตรงกลาง
- แถวคอย (Queue) ใช้สัญลักษณ์รูปสี่เหลี่ยม
- ตัวประมวลผล (Processor) ใช้สัญลักษณ์รูปวงกลม

ชนิดของความสัมพันธ์ที่ใช้ทำการเชื่อมโยงชนิดของเอนทิตีภายในระบบแถวคอยมีรายละเอียดดังต่อไปนี้ ตัวสร้างงานมีความสัมพันธ์กับแถวคอย แถวคอยมีความสัมพันธ์กับตัวประมวลผลและอีกด้านหนึ่งของตัวประมวลผลจะเชื่อมต่อเข้ากับแถวคอยแถวตัวต่อไปซึ่งถูกแทนด้วยตัวแทนของชนิดเอนทิตี (Proxy entity-type) หัวลูกศรแทนทิศทางการเข้าถึงข้อมูลของเอนทิตีเช่น ตัวสร้างงานและตัวประมวลผลสามารถเปลี่ยนแปลงค่าของแถวคอยได้ สำหรับปลายของลูกศรแสดงคาร์ดินัลลิตีเรโช ถ้าปลายลูกศรด้านใดมีวงกลมสีดำแสดงว่าเอนทิตีนั้นสามารถเชื่อมต่อกับเอนทิตีอีกชนิดหนึ่งได้มากกว่า

หนึ่งตัว จากแผนภาพที่ปรากฏในรูปที่ 2.2 แสดงให้เห็นว่าตัวสร้างงานหลายตัวสามารถเชื่อมต่อกับแถวคอยเดียวกันได้ และแถวคอยแต่ละตัวสามารถเชื่อมต่อกับตัวประมวลผลหลายตัวได้



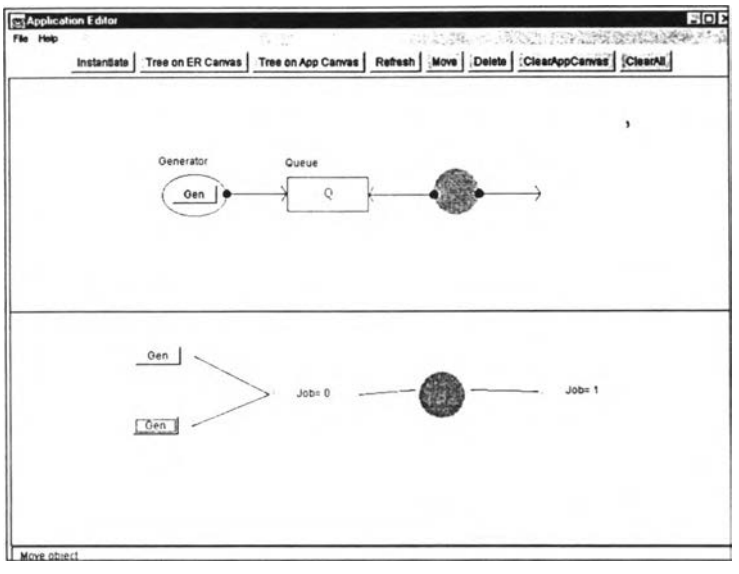
รูปที่ 2.2 บรรณาธิกรสำหรับสร้างเค้าร่าง

เมื่อผู้พัฒนากำหนดแผนภาพความสัมพันธ์เรียบร้อยแล้ว สามารถสั่งให้บรรณาธิกรสำหรับสร้างเค้าร่างทำการสร้างชุดคำสั่งภาษาจาวาได้โดยอัตโนมัติ ผลของการสร้างชุดคำสั่งคือได้ชุดคำสั่งเป็น คลาสโมเดล (Model) และคลาสวิว (View) ของแต่ละชนิดของเอนทิตี โดยที่คลาสโมเดลมีชื่อคลาสเป็นชื่อเดียวกับชนิดของเอนทิตีและทำหน้าที่เกี่ยวกับการควบคุมการทำงานโดยไม่เกี่ยวกับการแสดงผลบนหน้าจอ คลาสวิวจะขึ้นต้นด้วย “Edit” แล้วตามด้วยชื่อของชนิดของเอนทิตีและทำหน้าที่เกี่ยวกับการแสดงผลการทำงานเป็นภาพบนหน้าจอ แต่ชุดคำสั่งที่สร้างได้นี้เป็นเพียงโครงของชุดคำสั่งเท่านั้น ผู้พัฒนาโปรแกรมจะต้องทำการเขียนชุดคำสั่งเกี่ยวกับพฤติกรรมด้วยตนเองแล้วจึงทำการแปลโปรแกรมเพื่อนำไปใช้ในบรรณาธิกรสำหรับสร้างโปรแกรมประยุกต์ต่อไป

2.2.3 บรรณาธิกรสำหรับสร้างโปรแกรมประยุกต์

บรรณาธิกรสำหรับสร้างโปรแกรมประยุกต์มีหน้าที่สำหรับใช้สร้างโปรแกรมประยุกต์ตามขอบเขตที่กำหนดโดยแผนภาพเอนทิตีและความสัมพันธ์ หน้าจอหลักของบรรณาธิกรสำหรับสร้างโปรแกรมประยุกต์ถูกแสดงไว้ในรูปที่ 2.3 ซึ่งประกอบด้วยพื้นที่ส่วนบนเป็นการแสดงแผนภาพเอนทิตีและความสัมพันธ์ของระบบแถวคอยที่ได้ทำการสร้างไว้แล้วจากบรรณาธิกรสำหรับสร้างเค้าร่างซึ่งถูกใช้เป็นรายการเลือกสำหรับสร้างโปรแกรมประยุกต์ พื้นที่ส่วนล่างถูกใช้สำหรับให้ผู้พัฒนาทำการสร้างโปรแกรมประยุกต์ของระบบแถวคอย เมื่อผู้ใช้งานเลือกชนิดของเอนทิตีในแผนภาพด้านบนและคลิกที่บริเวณพื้นที่ด้านล่างระบบจะสร้างเอนทิตีชนิดนั้นขึ้นทันที หลังจากเชื่อมต่อเอนทิตีเหล่านี้เข้าด้วยกันตามเค้าร่างที่กำหนดเป็นที่เรียบร้อยแล้วระบบจะสามารถจำลองการทำงานของระบบแถวคอยตามที่ผู้พัฒนา

ทำการเชื่อมต่อไว้ นอกจากนั้นผู้พัฒนาสามารถเปลี่ยนแปลงรูปแบบการเชื่อมต่อได้ในหลายลักษณะบนจอภาพโดยไม่ต้องแก้ไขและคอมไพล์โปรแกรมใหม่

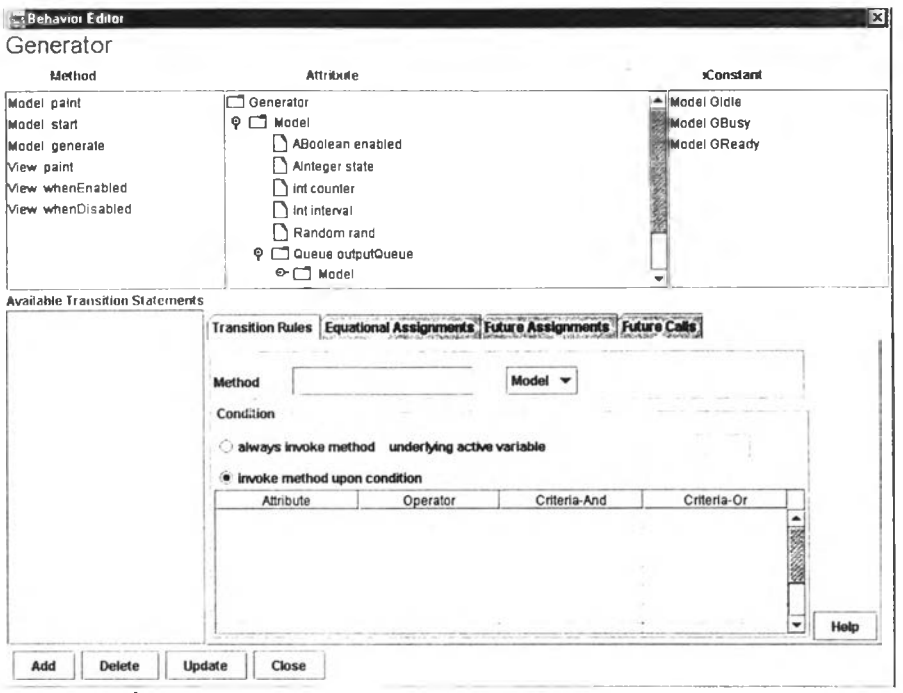


รูปที่ 2.3 บรรณาธิกรสำหรับสร้างโปรแกรมประยุกต์

2.3 การพัฒนาบรรณาธิกรสำหรับกำหนดพฤติกรรมของวัตถุพร้อมทำงานแบบวิชวล

ผู้ทำการวิจัยคือ จันทร์พร ลัภยพร ได้ทำการพัฒนา "บรรณาธิกรสำหรับกำหนดพฤติกรรมของวัตถุพร้อมทำงาน" [5] เพิ่มเติมให้กับ "สิ่งแวดล้อมสำหรับพัฒนาโปรแกรมด้วยแผนภาพเอนทิตีและความสัมพันธ์" ซึ่งใช้เป็นส่วนหนึ่งของบรรณาธิกรสำหรับสร้างเค้าร่าง บรรณาธิกรสำหรับกำหนดพฤติกรรมของวัตถุพร้อมทำงานนี้สามารถใช้กำหนดประโยคการเปลี่ยนแปลงให้กับชนิดของเอนทิตีแต่ละตัวในเค้าร่าง ซึ่งประโยคการเปลี่ยนแปลงนี้เป็นสิ่งที่ใช้กำหนดพฤติกรรมให้กับระบบวัตถุพร้อมทำงานแบบโครงสร้าง บรรณาธิกรนี้สามารถอำนวยความสะดวกให้กับผู้พัฒนาโปรแกรมได้เป็นอย่างมาก เนื่องจากมีส่วนติดต่อกับผู้ใช้แบบกราฟิก (GUI) จึงทำให้ง่ายต่อการใช้งาน นอกจากนั้นสามารถนำข้อมูลพฤติกรรมที่ผู้พัฒนากำหนดมาทำการสร้างเป็นชุดคำสั่งในส่วนพฤติกรรมของวัตถุพร้อมทำงานได้โดยอัตโนมัติ หน้าจอของบรรณาธิกรสำหรับกำหนดพฤติกรรมของวัตถุพร้อมทำงานแสดงไว้ในรูปที่ 2.4 ผู้วิจัยได้ทำการทดลองกำหนดพฤติกรรมให้กับชนิดของเอนทิตีต่างๆ ในโปรแกรมประยุกต์ 3 โปรแกรมได้แก่ ระบบแถวคอย ระบบแท็งก์ และระบบเครือข่าย แล้วทำการสร้างชุดคำสั่งอัตโนมัติ ผลปรากฏว่าระบบสามารถสร้างคำสั่งที่รองรับกลไกของพฤติกรรมแบบต่างๆ ตามที่กำหนดได้อย่างถูกต้องโดยมีร้อยละของจำนวนบรรทัดคำสั่งที่สร้างได้โดยอัตโนมัติเทียบกับจำนวนบรรทัดคำสั่งในโปรแกรมที่ใช้งานได้จริงของระบบแถวคอยเท่ากับ 76.0 ระบบแท็งก์เท่ากับ 52.7 และระบบเครือข่ายเท่ากับ 48.7 ดังนั้นบรรณาธิกรสำหรับกำหนดพฤติกรรมจึงมีประโยชน์ในการช่วยลดระยะเวลาในการเขียนชุดคำสั่งด้วยตัวเองเพราะโปรแกรมสามารถสร้างชุดคำสั่งให้โดยอัตโนมัติ ถึงแม้ว่าจะไม่สามารถสร้างชุดคำสั่งได้ทั้งหมดแต่จำนวนชุดคำสั่งที่ต้องเพิ่มเติมด้วยตนเองก็จะมีน้อยลง แต่อย่างไรก็ตามการกำหนดพฤติกรรมของวัตถุพร้อมทำงานด้วย

บรรณาธิกรสำหรับกำหนดพฤติกรรมไม่สามารถแสดงแบบจำลองเชิงพฤติกรรมของวัตถุพร้อมทำงานได้ ซึ่งทำให้ยากต่อการแก้ไขเปลี่ยนแปลงพฤติกรรมในภายหลังเพราะผู้พัฒนาไม่สามารถทำความเข้าใจในภาพรวมของพฤติกรรมในปัจจุบันได้



รูปที่ 2.4 บรรณาธิกรสำหรับกำหนดพฤติกรรมของวัตถุพร้อมทำงาน

2.4 แผนภาพสเตทชาร์ท

แผนภาพสเตทชาร์ทถูกเสนอในครั้งแรกในปี ค.ศ. 1983 โดย David Harel จากงานวิจัยเรื่อง "Statecharts: A Visual Formalism for Complex Systems" โดยมีจุดประสงค์เพื่อใช้แผนภาพสเตทชาร์ทในการแสดงพฤติกรรมของระบบเชิงโต้ตอบ (Reactive system) [8] แผนภาพสเตทชาร์ทถูกรวบรวมให้เป็นแผนภาพหนึ่งของยูเอ็มแอล เพื่อใช้ในการแสดงพฤติกรรมของวัตถุใดวัตถุหนึ่งโดยแสดงให้เห็นถึงการตอบสนองต่อเหตุการณ์ของวัตถุเมื่ออยู่ในสถานะต่างๆ [6,7,8,9] แผนภาพสเตทชาร์ทมี 4 องค์ประกอบที่สำคัญคือ สถานะ (State) การเปลี่ยนแปลง (Transition) เหตุการณ์ (Event) และการกระทำ (Action) ซึ่งแต่ละองค์ประกอบมีรายละเอียดดังต่อไปนี้

2.4.1 สถานะ

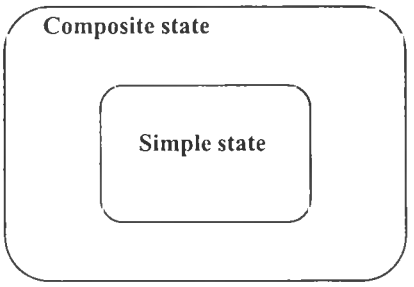
สถานะคือสภาวะที่วัตถุคอยการเกิดของเหตุการณ์บางอย่างเพื่อที่จะตอบสนองต่อเหตุการณ์นั้น สถานะถูกจำแนกไว้ 3 ประเภทดังต่อไปนี้

2.4.1.1 สถานะพื้นฐาน (Simple state)

สถานะพื้นฐานคือสถานะที่ไม่มีสถานะย่อย (Substate) อยู่ภายใน สัญลักษณ์ที่ใช้แทนสถานะพื้นฐานคือสี่เหลี่ยมมุมโค้งมนและระบุด้วยชื่อของสถานะดังรูปที่ 2.5

2.4.1.2 สถานะประกอบ (Composite state)

สถานะประกอบคือสถานะที่มีการกำหนดสถานะย่อยไว้ภายในซึ่งสถานะย่อยอาจจะเป็นสถานะพื้นฐานหรือสถานะประกอบก็ได้ สัญลักษณ์ของสถานะประกอบเป็นเช่นเดียวกับสถานะพื้นฐานแต่มีสถานะย่อยกำหนดอยู่ภายใน ดังที่แสดงในรูปที่ 2.5



รูปที่ 2.5 สถานะพื้นฐานและสถานะประกอบ

2.4.1.3 สถานะเทียม (Pseudo state)

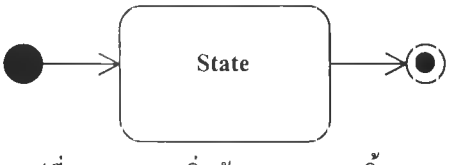
สถานะเทียมคือสัญลักษณ์ที่ไม่ได้ใช้สำหรับแสดงถึงสถานะของวัตถุแต่เป็นสัญลักษณ์ที่ช่วยแสดงความหมายให้แผนภาพสเตทชาร์ทมีความสมบูรณ์ยิ่งขึ้น สถานะเทียมถูกกำหนดไว้หลายชนิดซึ่งมีสัญลักษณ์และความหมายที่แตกต่างกันแต่สถานะเทียมที่จะอ้างถึงต่อไปนี้เป็นสถานะเทียมเฉพาะที่เกี่ยวข้องกับวิทยานิพนธ์นี้เท่านั้น

1) สถานะเริ่มต้น (Initial state)

สถานะเริ่มต้นคือสัญลักษณ์ที่ใช้บอกถึงจุดเริ่มต้นของสถานะของวัตถุในแผนภาพสเตทชาร์ท โดยใช้เป็นตัวชี้ว่าสถานะใดเป็นสถานะในครั้งแรกของวัตถุหลังจากถูกสร้างขึ้น ถ้ากำหนดสถานะเริ่มต้นไว้ภายในสถานะประกอบสถานะเริ่มต้นจะบอกถึงสถานะย่อยแรกหลังจากเกิดการเปลี่ยนแปลงเข้าสู่สถานะประกอบนั้น สัญลักษณ์ของสถานะเริ่มต้นแสดงด้วยวงกลมเล็กสีดำดังรูปที่ 2.6

2) สถานะสิ้นสุด (Final state)

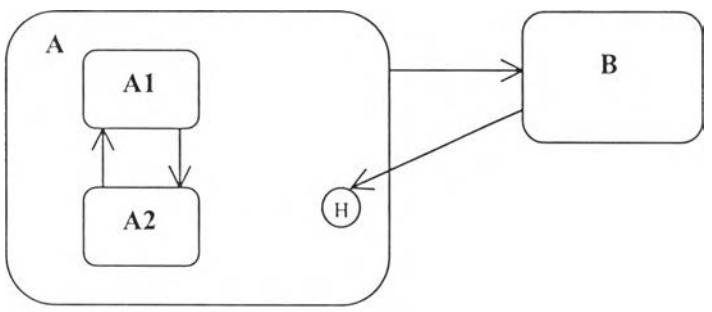
สถานะสิ้นสุดคือสัญลักษณ์ที่ใช้บอกถึงจุดสิ้นสุดของการทำงานของวัตถุ ซึ่งหมายถึงวัตถุจะหยุดการทำงานเมื่อเกิดการเปลี่ยนแปลงมายังสถานะสิ้นสุด ถ้ากำหนดสถานะสิ้นสุดไว้ภายในสถานะประกอบและเกิดการเปลี่ยนแปลงมายังสถานะสิ้นสุดนั้นจะหมายถึงการกระตุ้นให้เกิดการเปลี่ยนแปลงแบบสมบูรณ์ (Complete transition) ของสถานะประกอบดังกล่าวขึ้น สัญลักษณ์ของสถานะสิ้นสุดแสดงด้วยวงกลมเล็กสีดำและมีวงกลมล้อมรอบอีกชั้นหนึ่งดังรูปที่ 2.6



รูปที่ 2.6 สถานะเริ่มต้นและสถานะสิ้นสุด

3) สถานะครั้งก่อน (History state)

สถานะครั้งก่อนคือสัญลักษณ์ที่ใช้กำหนดภายในสถานะประกอบโดยเป็นตัวแทนของสถานะย่อยล่าสุดก่อนเกิดการเปลี่ยนแปลงออกจากสถานะประกอบนั้น เมื่อกำหนดให้มีการเปลี่ยนแปลงกลับมายังสถานะครั้งก่อนเป็นการแสดงถึงการเปลี่ยนสถานะกลับมายังสถานะย่อยล่าสุดก่อนเกิดการเปลี่ยนแปลงออกจากสถานะประกอบนั้น ดังตัวอย่างในรูปที่ 2.7 สถานะ A มีการระบุสถานะครั้งก่อนไว้ภายในซึ่งสัญลักษณ์ของสถานะครั้งก่อนคือวงกลมที่มีตัวอักษร H อยู่ภายใน สถานะ A เป็นสถานะประกอบโดยมีสถานะย่อย A1 และ A2 ถ้าในขณะที่สถานะของวัตถุอยู่ที่สถานะ A1 แล้วเกิดการเปลี่ยนแปลงไปยังสถานะ B เมื่อเกิดการเปลี่ยนแปลงกลับมาที่สถานะครั้งก่อนสถานะของวัตถุจะเป็น A1 เช่นเดียวกันถ้าในขณะที่สถานะของวัตถุอยู่ที่สถานะ A2 แล้วเกิดการเปลี่ยนแปลงไปยังสถานะ B เมื่อเกิดการเปลี่ยนแปลงกลับมาที่สถานะครั้งก่อนสถานะของวัตถุจะเป็น A2



รูปที่ 2.7 สถานะครั้งก่อน

4) สถานะลึกครั้งก่อน (Deep history state)

สถานะลึกครั้งก่อนคือสัญลักษณ์ที่ใช้เป็นตัวแทนของสถานะย่อยล่าสุดเช่นเดียวกับสถานะครั้งก่อน แต่สถานะลึกครั้งก่อนจะหมายถึงสถานะย่อยที่อยู่ระดับลึกสุดของโครงสร้างระดับชั้นของสถานะซึ่งแตกต่างจากสถานะครั้งก่อนที่สถานะครั้งก่อนจะเป็นตัวแทนของสถานะย่อยที่อยู่ในระดับแรกของสถานะย่อยเท่านั้น สัญลักษณ์ของสถานะลึกครั้งก่อนคือวงกลมที่มีตัวอักษร H^* อยู่ภายใน

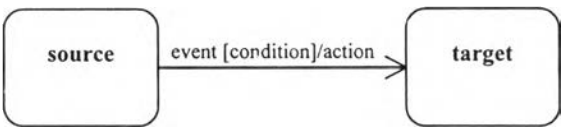
2.4.2 การเปลี่ยนแปลง

การเปลี่ยนแปลงคือการตอบสนองต่อเหตุการณ์ที่เกิดขึ้นของวัตถุโดยที่ผลของการตอบสนองนั้นอาจทำให้เกิดการกระทำหรือการเปลี่ยนสถานะของวัตถุ การแสดงการเปลี่ยนแปลงในแผนภาพสเตทชาร์ทจะต้องระบุด้วยข้อความการเปลี่ยนแปลง (Transition string) เพื่อให้ทราบถึงคุณสมบัติของการเปลี่ยนแปลงนั้นๆ รูปแบบของข้อความการเปลี่ยนแปลงคือ "event [condition] / action" โดยที่ event คือเหตุการณ์ที่เป็นตัวกระตุ้นให้เกิดการเปลี่ยนแปลง condition คือเงื่อนไขที่จะถูกตรวจสอบเมื่อเกิด

เหตุการณ์ที่ระบุขึ้น และ action คือการกระทำที่ถูกประมวลผลถ้าเงื่อนไขที่กำหนดเป็นจริง การเปลี่ยนแปลงสามารถจำแนกได้ 2 ประเภทดังนี้

4.2.2.1 การเปลี่ยนแปลงภายนอก (External transition)

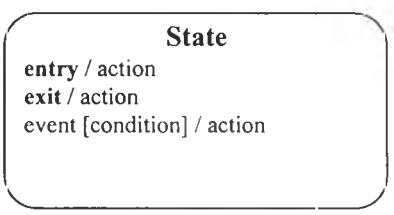
การเปลี่ยนแปลงภายนอกคือการตอบสนองต่อเหตุการณ์ที่ทำให้มีการเปลี่ยนสถานะของวัตถุ สัญลักษณ์ของการเปลี่ยนแปลงภายนอกคือ เส้นหัวลูกศรและอธิบายควบคู่ด้วยข้อความการเปลี่ยนแปลงดังรูปที่ 2.8 สัญลักษณ์ของการเปลี่ยนแปลงภายนอกแสดงให้เห็นถึงการเปลี่ยนสถานะจากสถานะตั้งต้น (Source state) ไปยังสถานะเป้าหมาย (Target state)



รูปที่ 2.8 การเปลี่ยนแปลงภายนอก

4.2.2.2 การเปลี่ยนแปลงภายใน (Internal transition)

การเปลี่ยนแปลงภายในคือการตอบสนองต่อเหตุการณ์ที่เกิดขึ้นแต่ไม่เกิดการเปลี่ยนสถานะของวัตถุ ดังนั้นสัญลักษณ์ที่ใช้ระบุถึงการเปลี่ยนแปลงภายในจึงมีเฉพาะข้อความการเปลี่ยนแปลงที่ต้องกำหนดไว้ภายในสถานะดังรูปที่ 2.9 สำหรับ entry และ exit เป็นคำสงวนที่ใช้สำหรับการเปลี่ยนแปลงภายในซึ่งระบุถึงเหตุการณ์เมื่อเข้ามายังสถานะและออกจากสถานะตามลำดับ



รูปที่ 2.9 การเปลี่ยนแปลงภายใน

2.4.3 เหตุการณ์

เหตุการณ์คือสิ่งที่เกิดขึ้น ณ เวลาหนึ่งเพื่อกระตุ้นให้เกิดการตอบสนองของวัตถุ การแสดงเหตุการณ์ในแผนภาพสเตทชาร์ทจะต้องระบุในข้อความการเปลี่ยนแปลงเพื่อให้ทราบว่าการเปลี่ยนแปลงนั้นจะถูกกระตุ้นให้เกิดขึ้นด้วยเหตุการณ์อะไร เหตุการณ์ถูกกำหนดไว้ 4 ประเภทดังนี้

2.4.3.1 เหตุการณ์ประเภทเรียกการทำงาน (Call event)

เหตุการณ์ประเภทเรียกการทำงานคือเหตุการณ์ที่เกิดขึ้นเมื่อมีการร้องขอบริการจากวัตถุอื่นซึ่งเป็นผู้ส่งข้อความร้องขอมาเพื่อให้ทำการประมวลผลการทำงานของฟังก์ชันของวัตถุโดยที่วัตถุที่เป็นผู้ร้องขอจะคอยจนกระทั่งการทำงานเสร็จสิ้นจึงจะสามารถทำงานอื่นต่อ

ไปได้ สัญลักษณ์ที่ใช้แสดงเหตุการณ์ประเภทนี้คือ ชื่อของการทำงานซึ่งเป็นชื่อของฟังก์ชันสมาชิกที่ต้องการให้ถูกประมวลผล

2.4.3.2 เหตุการณ์ประเภทสัญญาณ (Signal event)

เหตุการณ์ประเภทสัญญาณคือเหตุการณ์ที่เกิดขึ้นเมื่อมีการส่งสัญญาณมาจากวัตถุอื่น หลังจากส่งสัญญาณเสร็จสิ้นแล้ววัตถุผู้ส่งสัญญาณสามารถทำงานอื่นต่อไปได้ทันทีโดยไม่ต้องคอยให้วัตถุผู้รับสัญญาณทำงานเสร็จก่อน สัญลักษณ์ที่ใช้แสดงเหตุการณ์ประเภทสัญญาณคือชื่อของสัญญาณที่ได้รับเช่น MouseButtonDown เป็นต้น

2.4.3.3 เหตุการณ์ประเภทการเปลี่ยนแปลง (Change event)

เหตุการณ์ประเภทการเปลี่ยนแปลงคือเหตุการณ์ที่เกิดขึ้นเมื่อมีการเปลี่ยนแปลงคุณสมบัติภายในวัตถุ ซึ่งการตรวจสอบการเปลี่ยนแปลงของคุณสมบัติภายในวัตถุทำโดยการตรวจสอบค่าจากประโยคบูลีนถ้าค่าของประโยคบูลีนเป็นจริงหมายถึงเหตุการณ์ประเภทการเปลี่ยนแปลงได้เกิดขึ้น แต่ถ้าค่าของประโยคบูลีนเป็นเท็จหมายถึงไม่มีเหตุการณ์ประเภทการเปลี่ยนแปลงเกิดขึ้น ดังนั้นการตรวจสอบค่าของประโยคบูลีนจึงเป็นการตรวจสอบแบบต่อเนื่องตลอดเวลาซึ่งต่างจากการตรวจสอบเงื่อนไขในประโยคการเปลี่ยนแปลงซึ่งจะถูกตรวจสอบเพียงครั้งเดียวเมื่อมีเหตุการณ์เกิดขึ้น สัญลักษณ์ที่กำหนดให้กับเหตุการณ์ประเภทการเปลี่ยนแปลงคือ “when (Boolean expression)”

2.4.3.4 เหตุการณ์ประเภทเวลา (Time event)

เหตุการณ์ประเภทเวลาคือเหตุการณ์ที่เกิดขึ้นเมื่อเวลาได้ดำเนินไปครบระยะเวลาที่กำหนด สัญลักษณ์ที่ใช้แสดงเหตุการณ์ประเภทเวลาคือ “after(time)” เวลาที่กำหนดในสัญลักษณ์คือช่วงเวลาที่ทำการหน่วงการเกิดของเหตุการณ์โดยที่หน่วยของเวลาขึ้นอยู่กับผู้ใช้จะกำหนด

2.4.4 การกระทำ

การกระทำคือสิ่งที่จะต้องถูกประมวลผลเมื่อเกิดการเปลี่ยนแปลงขึ้น การกระทำไม่มีสัญลักษณ์ที่เป็นรูปแบบแต่จะกำหนดในรูปของประโยคที่ต้องการทำการประมวลผลเช่น “count++” หรือ “A=B+C” เป็นต้น ประโยคการกระทำจะต้องกำหนดภายในข้อความการเปลี่ยนแปลงโดยที่หนึ่งข้อความการเปลี่ยนแปลงจะสามารถมีประโยคการกระทำได้มากกว่าหนึ่งประโยคซึ่งแต่ละประโยคจะต้องคั่นด้วยเครื่องหมายลูกน้ำ ดังเช่นรูปแบบของข้อความการเปลี่ยนแปลงต่อไปนี้

$$\text{event [condition] / action}_1, \text{action}_2, \ldots, \text{action}_n$$