State Diagram และ Activity Diagram

Dynamic หรือ Behavior View

ในบทก่อนหน้าได้กล่าวถึง UML Diagram ในกลุ่มของ Dynamic หรือ Behavior มาแล้ว 2 diagrams คือ Sequence และ Collaboration Diagram

ในบทนี้จะกล่าวถึง Diagrams ที่เหลือในกลุ่ม Dynamic คือ State และ Activity Diagram

มุมมองเชิงโครงสร้าง (Static หรือ Structure)	**มุมมองเชิงพฤติกรรม (Dynamic หรือ Behavior)**
Class Diagram	Use case Diagram
Object Diagram	Sequence Diagram
Component Diagram	Collaboration Diagram
Deployment Diagram	State Diagram
	Activity Diagram

Interaction Diagrams

State Diagram

State Diagram อาจเรียกว่า

State transition diagram หรือ

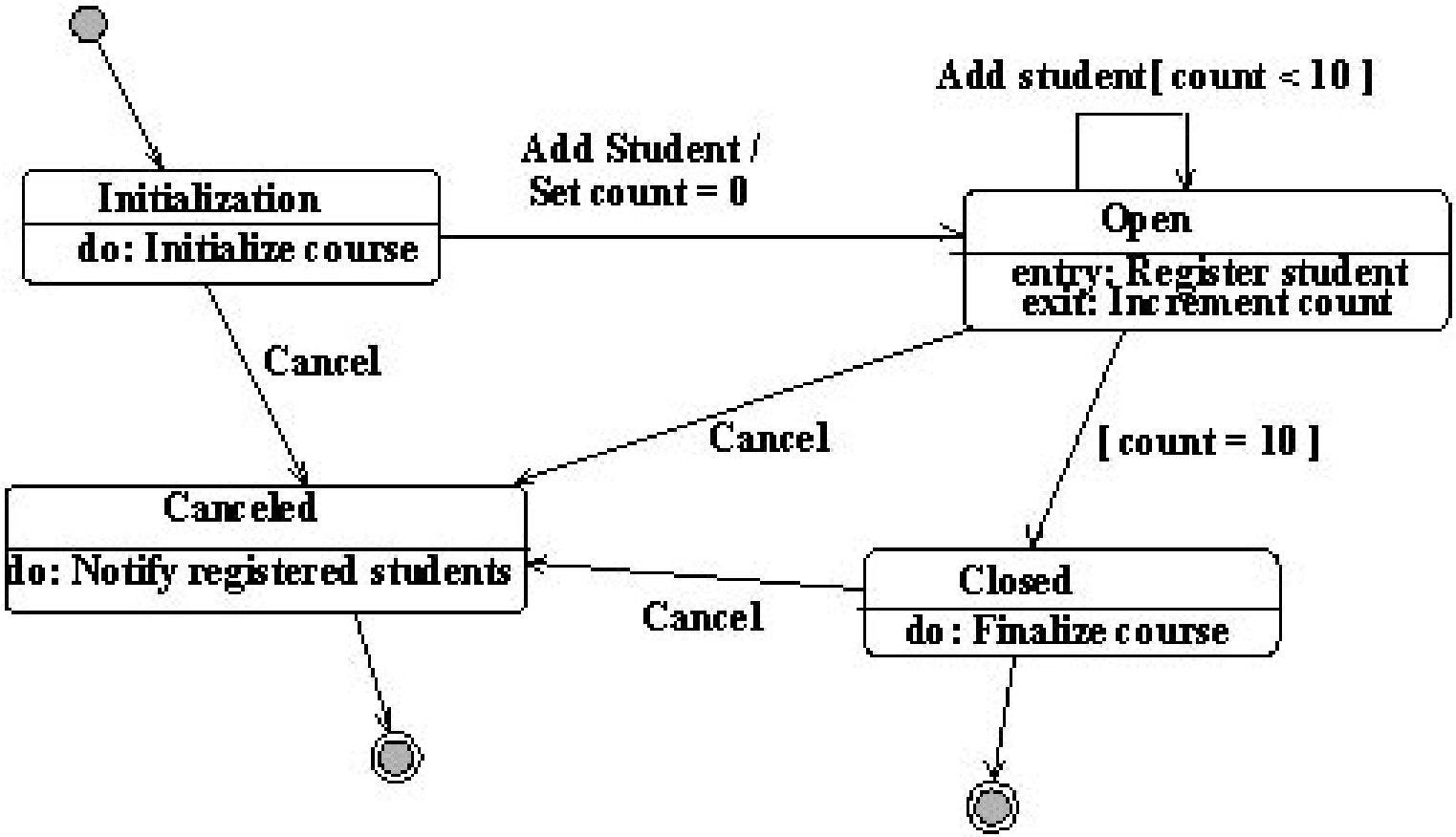
Harel diagram (statecharts)

State Diagram เป็นแผนภาพใช้แสดงสถานะ(state) ต่างๆ ของ Object ที่เป็นได้ในระหว่างช่วงชีวิต ในการตอบสนองต่อเหตุการณ์ (Event) ที่เกิดขึ้น

โดยทั่วไปแล้ว State Diagram จะไม่ถูกใช้กับ Class ทั้งหมด แต่จะใช้อธิบายเฉพาะ Class ที่มีความซับซ้อนสูงเท่านั้น เพื่อที่จะช่วยให้การออกแบบ Algorithm ง่ายขึ้น

State Diagram

ตัวอย่าง State Diagram



State Diagram

State และ Transition

ในระบบใดๆก็ตาม สิ่งที่เคลื่อนไหวหรือการเปลี่ยนแปลงใด ๆ ที่เกิดขึ้นในระบบนั้น เรียกว่า กิจกรรม (Activity) ซึ่งกิจกรรมนั้นเกิดขึ้นจากการที่ Objects ในระบบมีปฏิสัมพันธ์กัน

สิ่งที่ใช้เพื่อบรรยายกิจกรรมโดยรวมที่เกิดขึ้นในระบบก็คือ Sequence Diagram แต่เมื่อพิจารณาเข้าไปในรายละเอียดของกิจกรรมที่เกิดขึ้นจะพบว่ากิจกรรมโดยรวมของระบบเกิดจากกิจกรรมย่อยของ Object แต่ละตัวรวมกันนั่นเอง

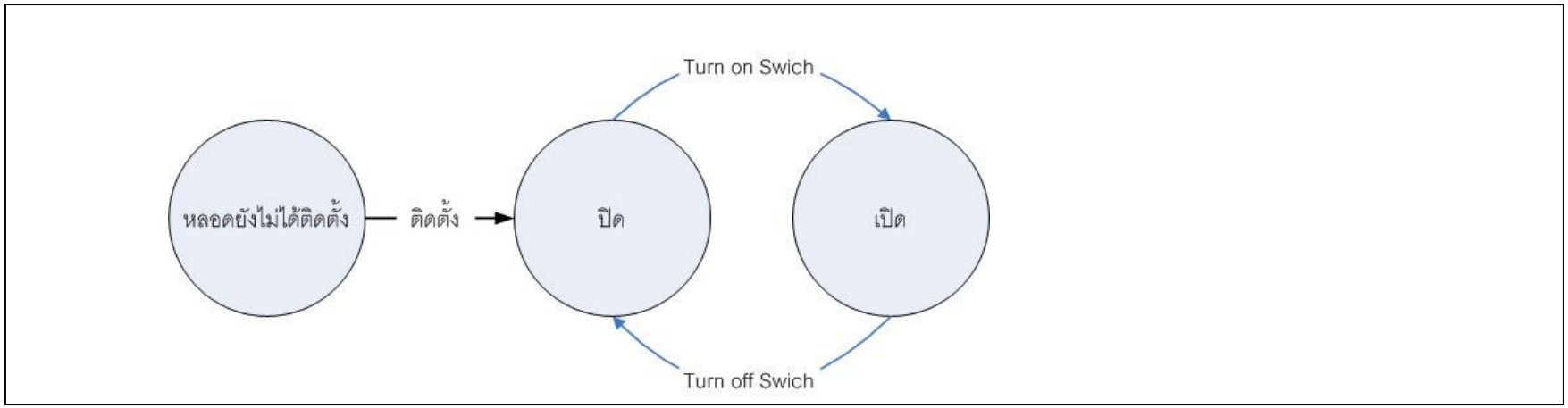
กลไกที่ทำให้ระบบมีกิจกรรมก็คือการรับ/ส่ง Message

State Diagram

State และ Transition

กิจกรรมที่เกิดขึ้นใน Object นั้น เกิดจาก 2 สิ่งประกอบกัน นั่นคือ สถานะ(State) และ การเปลี่ยนสถานะ (Transition) การที่ Object ใด ๆ เปลี่ยนจาก State ที่ 1 ไปยัง State ที่ 2 จะทำให้เกิดกิจกรรม หรือส่วนของกิจกรรมขึ้นในตัว Object นั้น ดังรูปตัวอย่างของกิจกรรมของหลอดไฟ

State Diagram



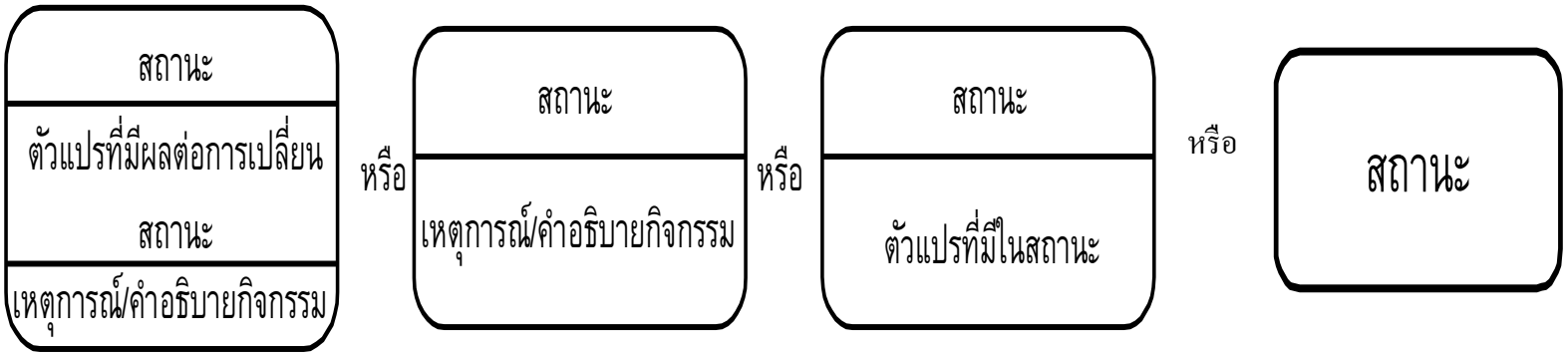
รูปแสดงกิจกรรมที่เกิดขึ้นในหลอดไฟโดยใช้ State และ Transition เป็นสื่อในการอธิบาย ดังนี้

1. หลอดไฟจะเริ่มต้นกิจกรรมทั้งหมดที่ State หลอดยังไม่ได้ติดตั้ง
2. เมื่อได้รับการติดตั้งแล้วหลอดไฟจะอยู่ใน State ปิด
3. จากสถานะปิด เมื่อเกิด Transition Turn on Switch ขึ้น หลอดไฟจะเปลี่ยน State ไปยัง State เปิด
4. แต่จาก State เปิด เมื่อได้รับ Transition Turn off Switch หลอดไฟจะกลับมายัง State เปิด อีกครั้ง
5. หลังจากนั้น State ของหลอดไฟจะเปลี่ยน State ไปมาระหว่าง State ปิด – เปิด เช่นนี้ต่อไป

State Diagram

สัญลักษณ์ที่ใช้ใน State Diagram

State สถานะของ Object
แทนด้วย สี่เหลี่ยมมุมมน



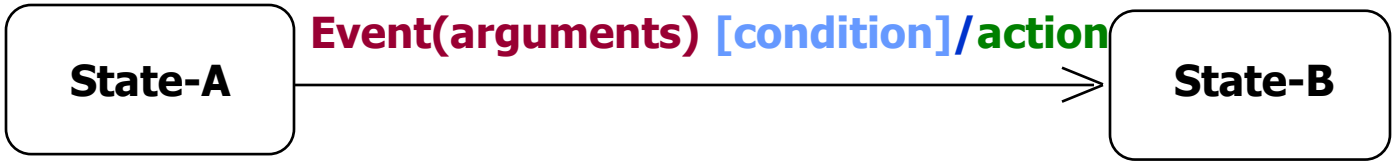
State Diagram

สัญลักษณ์ที่ใช้ใน State Diagram

Transition แทนด้วย ลูกศร
ลากจาก state เริ่มต้นไปยัง state ที่ต้องการ
เขียนชื่อ Event บนลูกศร มีรูปแบบคือ

[Condition]/[Action]

Condition คือ เงื่อนไขในการเข้าหรือออกจาก state
Action คือ กิจกรรมที่ทำระหว่างการเปลี่ยน state



บนเส้นลูกศรอาจ
- กำกับด้วย Event การกดลิฟต์
- กำกับด้วยฟังก์ชัน ^ borrower.overdue
- กำกับด้วยเงื่อนไข [dueDate < Now] , [วันที่ยืม > 30 วัน]

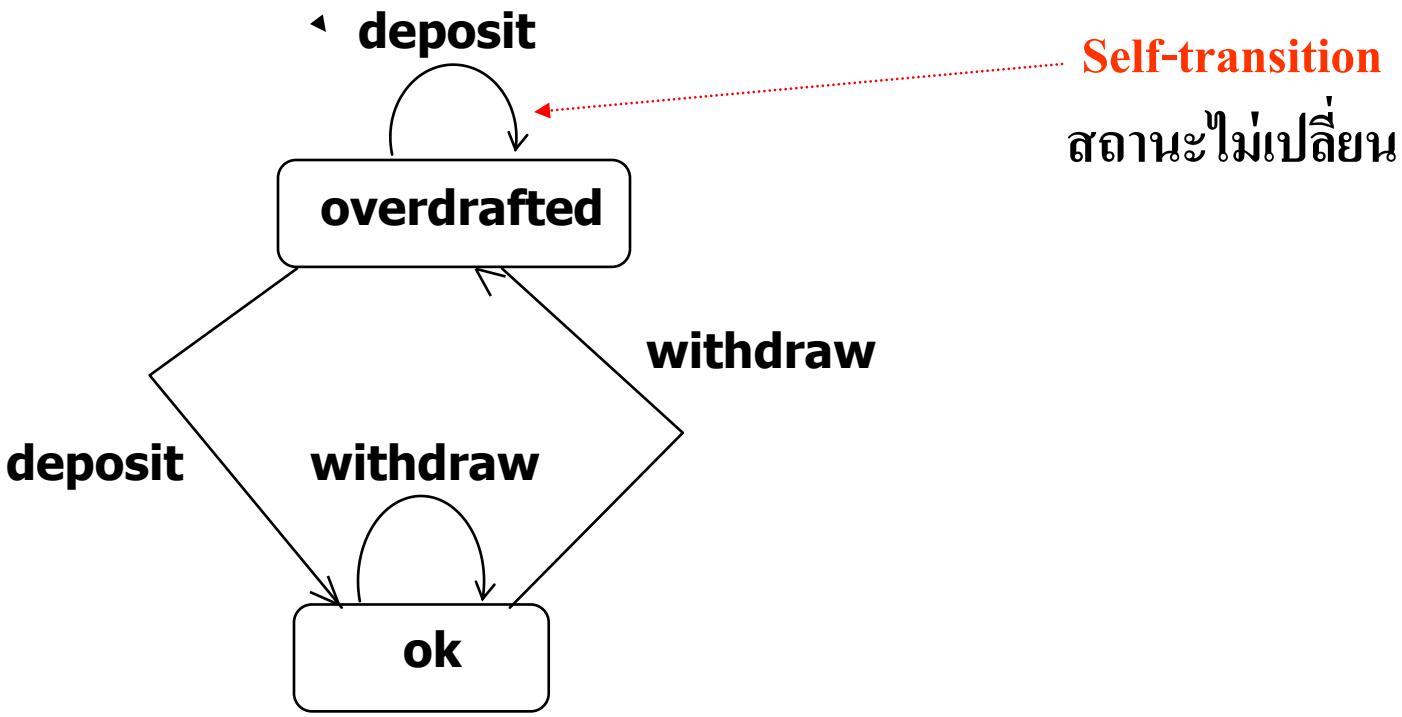
- มีเหตุการณ์ และเงื่อนไขที่เป็นผลต่อการเปลี่ยนสถานะ
[dueDate < Now]/borrower.overdue

State Diagram

สัญลักษณ์ที่ใช้ใน State Diagram

Event (เหตุการณ์) หมายถึง Message หรือ Signal ที่วัตถุได้รับ

Events อาจจะทำให้สถานะของวัตถุเปลี่ยนแปลง หรือไม่ก็ได้



State Diagram

สัญลักษณ์ที่ใช้ใน State Diagram

จุดเริ่มต้น จุดเริ่มต้นของกิจกรรมต่าง ๆ ใน state diagram เรียกว่า initial state แทนด้วยวงกลมทึบ 

จุดสิ้นสุด จุดสิ้นสุดของกิจกรรมทั้งหมดเรียกว่า End state แทนด้วยวงกลมใส ล้อมรอบวงกลมทึบ 

State Diagram

การจำลองกิจกรรมภายใน state (Internal Acitivity)

ใช้เพื่อระบุรายละเอียดในการทำงานของ state ต่าง ๆ ให้ชัดเจนขึ้น แบ่งได้เป็น

- กิจกรรมที่ทำเมื่อเข้ามาใน state (**entry/action**)
- กิจกรรมที่ทำระหว่างอยู่ใน state (**do/action**)
- กิจกรรมที่ทำก่อนที่จะออกจาก state (**exit/action**)
- กิจกรรมที่ทำเมื่อเกิดเงื่อนไขต่างๆ ขึ้น (**condition/action**)

State Diagram

กิจกรรมที่ทำเมื่อเข้ามาใน state

entry/action

หมายถึง เมื่อเข้ามายัง state นี้ให้ทำกิจกรรม action

เช่น entry/count=0

หมายถึง เมื่อเข้ามายัง state ให้ค่า count เป็น 0

State Diagram

กิจกรรมที่ทำระหว่างอยู่ใน state

do/action

หมายถึง หลังจากเข้ามายัง state นี้แล้ว หากไม่มีเงื่อนไขอื่นใด ให้ทำกิจกรรม action

เช่น do/count:=count+1

หมายถึง เมื่อเข้ามายัง state นี้ให้เพิ่มค่า count ทีละ 1

State Diagram

กิจกรรมที่ทำก่อนออกจาก state

exit/action

หมายถึง ขณะที่จะออกจาก state นี้ให้ทำกิจกรรม action

เช่น exit/show “Good Bye” message

หมายถึง หากออก state นี้ให้แสดงข้อความ “Good Bye”

State Diagram

กิจกรรมที่ทำเมื่อเกิดเงื่อนไขต่างๆขึ้น

condition/action

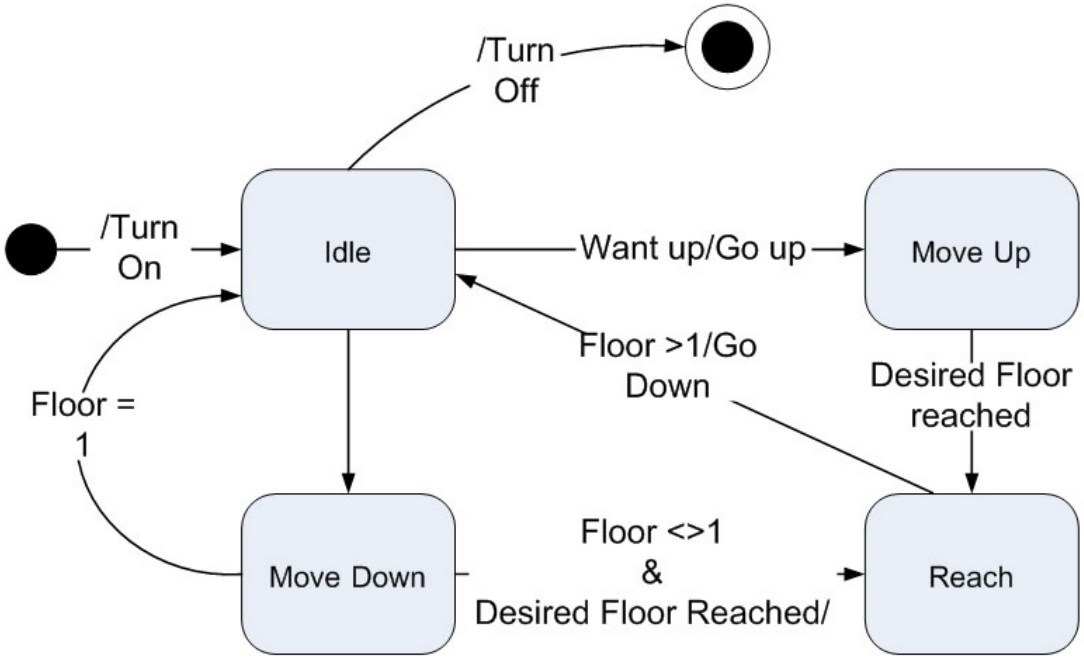
หมายถึง ขณะที่ยังอยู่ใน state นี้เมื่อเกิดเงื่อนไขใด ๆ ที่กำหนดโดย condition ให้ทำกิจกรรม action

เช่น every 2 seconds/phone ring

หมายถึง ทุก ๆ 2 วินาทีให้เสียงโทรศัพท์ดัง 1 ครั้ง

State Diagram

ตัวอย่าง State Diagram การทำงานของลิฟต์



รูปแสดง State Diagram เพื่อจำลองการทำงานของลิฟต์ โดยมีเงื่อนไขว่า ไม่ว่าลิฟต์จะเคลื่อนที่ไปยังจุดหมายที่ชั้นใดก็ตาม ลิฟต์จะต้องเคลื่อนที่กลับมาอยู่ที่ชั้น 1 ตามเดิม (Idle)

State Diagram

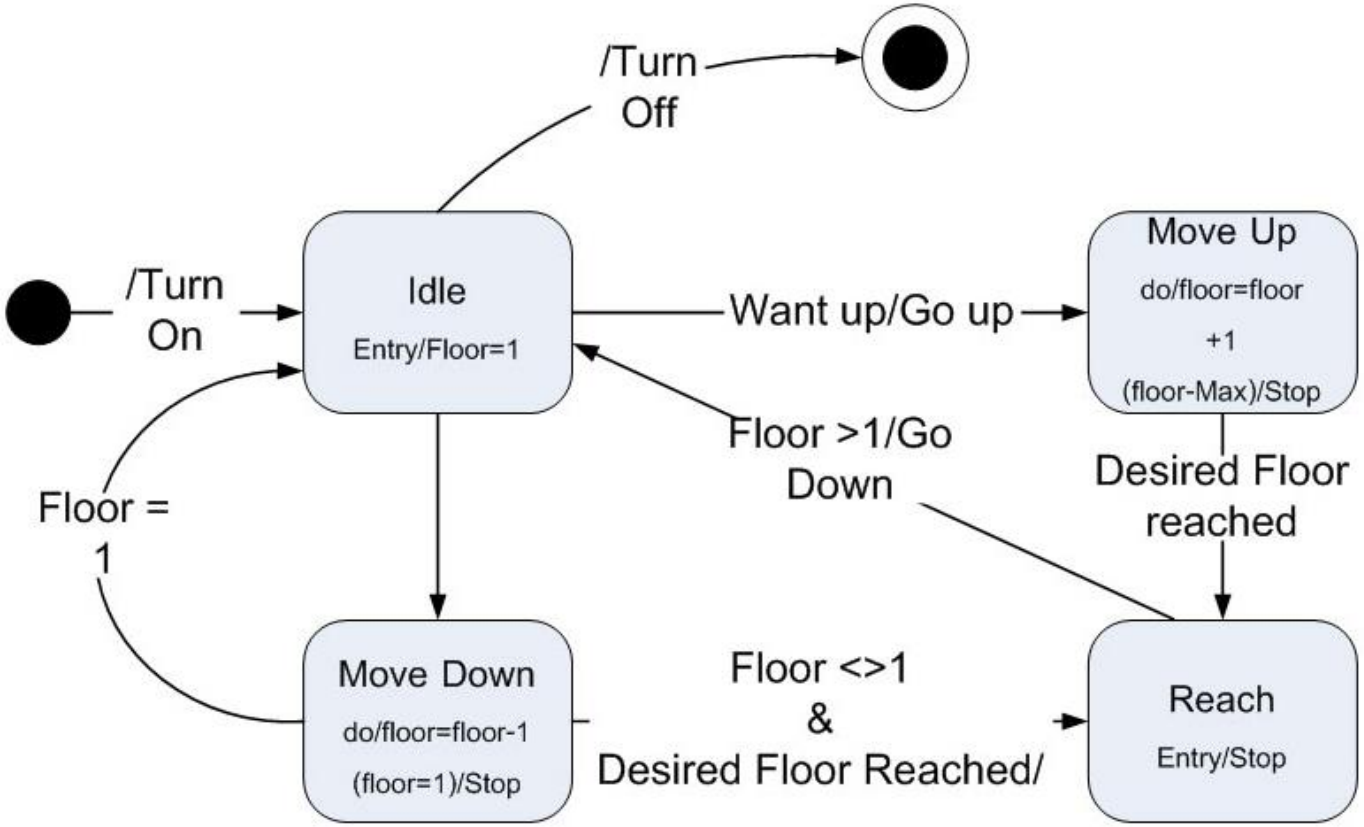
State Diagram นี้ อธิบายการทำงานของลิฟต์ได้ดังนี้

1. เริ่มต้นที่ Initial State เมื่อมีการเปิดสวิตช์ของลิฟต์ ลิฟต์จึงเข้าสู่สภาวะหยุดนิ่ง (Idle)
2. เมื่อมีคนต้องการที่จะขึ้น (Want Up) ลิฟต์จึงมีการเคลื่อนที่ขึ้น (Go up) ทำให้ลิฟต์อยู่ในสภาวะเคลื่อนขึ้น (Move Up)
3. แต่ถ้ามีคนต้องการที่จะลง (Want Down) ลิฟต์จึงมีการเคลื่อนที่ลง (Go Down) ทำให้ลิฟต์อยู่ในสภาวะเคลื่อนลง (Move Down)
4. สืบเนื่องจากข้อ 2 และ 3 เมื่อลิฟต์มาถึงชั้นที่ต้องการ (แต่ไม่ใช่ชั้นที่ 1 ซึ่งเป็นชั้นล่างสุด - floor <> 1) ลิฟต์จะอยู่ในสถานะ Reach จนกว่าคนจะลงจากลิฟต์จนหมด จึงเลื่อนลงมา (Go Down) ที่ชั้นที่ 1 แล้วกลับสู่สภาวะ Idle อีกครั้ง
5. แต่จากข้อ 3 ถ้าลิฟต์มาถึงชั้นที่ 1 แล้วจะ Idle ทันที
6. เมื่อลิฟต์อยู่ในสภาวะ Idle เมื่อใดก็ตามที่มีการปิดสวิตช์ (Turn Off) กิจกรรมทั้งหมดของลิฟต์จะหยุดทันที (การทำงานมาถึง Final State แล้ว) นั่นหมายความว่าลิฟต์จะถูกปิดได้ก็ต่อเมื่อลิฟต์อยู่ในสภาวะ Idle เท่านั้น

State Diagram

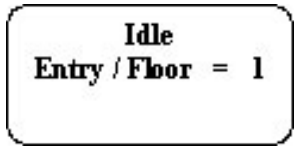
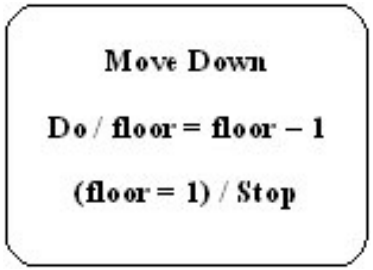
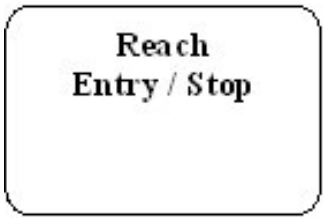
จากการใช้ Internal Activity เพื่อบรรยายกิจกรรมที่เกิดขึ้นกับ Objects ต่างๆ ในตัวอย่างที่ผ่านมา

เราสามารถสร้าง State Diagram ที่มี State ซึ่งมี Internal Activity ได้ดังรูปต่อไปนี้



State Diagram

State Diagram ที่มี Internal Activity ในแต่ละ State สามารถอธิบายได้ดังนี้

	**State Idle หมายถึง** State ของลิฟต์ เมื่อ Lift ยังอยู่ที่ชั้นที่ 1 เข้ามายัง State นี้ ค่า Floor จะถูก Set ไว้ที่ 1 เสมอ
	**State Move Down** หมายถึง State ของลิฟต์ ที่มีการเคลื่อนที่ลงทีละชั้น โดยเมื่อใดก็ตามที่เข้ามายัง State นี้ ค่า floor จะลดลง ทีละ 1 และเมื่อใดก็ตามที่ Floor มีค่าเป็น 1 ให้หยุดลิฟต์ทันที ซึ่งการหยุดลิฟต์ที่ floor เท่ากับ 1 นั้นเท่ากับเป็นการบังคับให้ลิฟต์เข้ามาอยู่ในสถานะ Idle โดยปริยาย
	**State Move Up** หมายถึง State ของลิฟต์ ที่มีการเคลื่อนที่ใดก็ตามที่เข้ามายัง State นี้ ค่า floor จะเพิ่มขึ้นทีละ 1 และเมื่อใดก็ตามที่ Floor มีค่าเป็น Max ซึ่งหมายถึงลิฟต์อยู่ในชั้นสูงสุด ให้หยุดลิฟต์ทันที
	**Reach** หมายถึง State ที่ลิฟต์มาถึงยังชั้นที่กำหนด (มีคนต้องขึ้นหรือลงจากลิฟต์) ซึ่งเมื่อใดก็ตามที่เข้ามาถึง State นี้แล้วต้องหยุดลิฟต์ทันที

State Diagram

วัตถุดิบที่นำมาใช้ในการสร้าง State Diagram คือ Class Diagram และ Sequence Diagram

Class Diagram จะทำให้เห็นภาพของ Class แต่ละ Class และแต่ละ Method ของ Class จะหมายถึง State Diagram หนึ่งชุด

Sequence Diagram จะทำให้เห็นภาพกิจกรรมของ Class ซึ่งจะใช้เพื่อโต้ตอบกับ Class อื่น ๆ ใน Problem Domain ซึ่งมีส่วนช่วยในการพิจารณาแนวการดำเนินไปของการเปลี่ยน State ของ Class หนึ่ง ๆ นั่นเอง

State Diagram

หลักในการเขียน State Diagram ให้มีประสิทธิภาพมีดังนี้

1. จาก Class Diagram ให้ดูว่ามี State Diagram กี่ตัวที่ต้องเขียน ซึ่งปกติแล้วจะเท่ากับจำนวน Method ของแต่ละ Class รวมกัน

แต่อย่างไรก็ตาม ไม่จำเป็นที่จะต้องเขียน State Diagram ของทุก ๆ Method ของทุก ๆ Class ในบาง Method ที่ไม่ได้มีกิจกรรมที่ซับซ้อน ก็ไม่จำเป็นต้องมี State Diagram

2. ในแต่ละ Class ให้พิจารณาว่า จะมี State อะไรบ้าง (โดยยึดจากหลักการของความเป็นจริง) โดยยังไม่ต้องคำนึงว่ามี Method อะไรอยู่บ้าง

3. จาก State ที่มีอยู่ให้เขียน State Diagram ของแต่ละ Method

4. หากพบว่ามี State ใดที่จะต้องเพิ่ม เพื่อทำให้ State Diagram สมบูรณ์ขึ้น ให้เพิ่มเข้าไป

5. ทำข้อ 3 และ 4 จนกว่าจะได้ State Diagram ของ 1 Class ที่สมบูรณ์

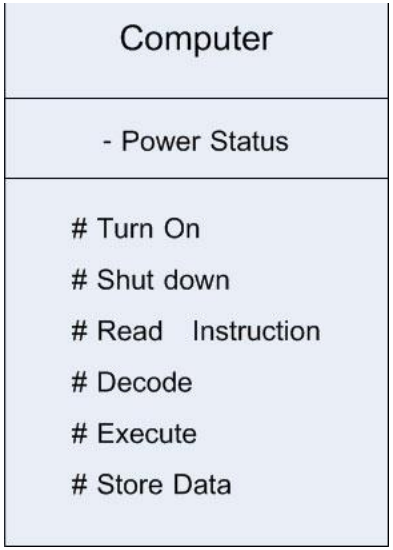
6. ทำข้อ 1 – 5 จนครบทุก ๆ Class ใน Class Diagram

State Diagram

พิจารณาหลักการเขียน State Diagram

ตัวอย่างที่ การเขียน State Diagram ของ Method ต่างๆ ของ Class Computer ได้ดังต่อไปนี้

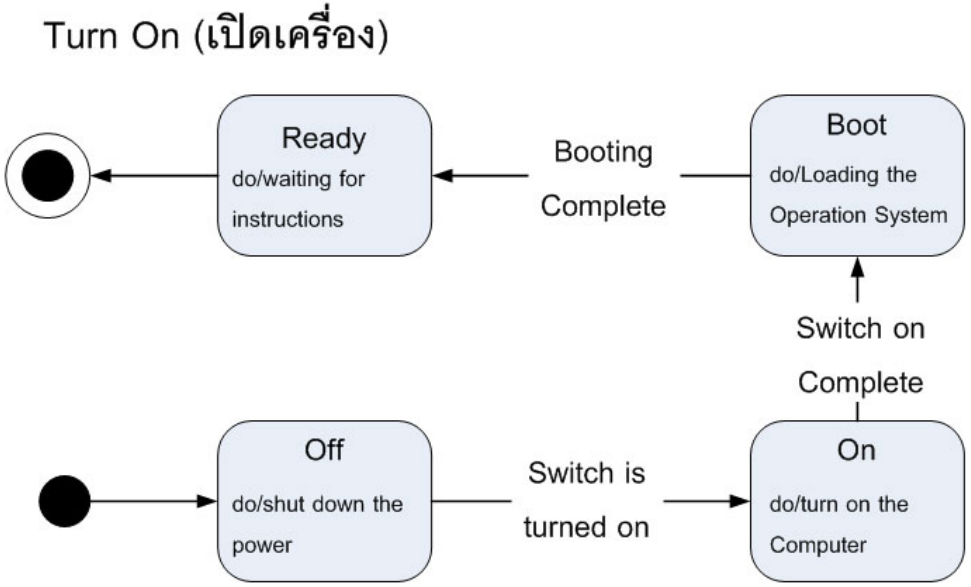
State ที่ควรจะมีของ Class Computer คือ



- Off (เครื่องปิด)
- On (เครื่องเปิด)
- Boot (เครื่องกำลังเริ่มทำงาน)
- Ready (เครื่องพร้อมทำงาน)
- Reading (อ่านคำสั่งจากหน่วยความจำ (Memory))
- Sending (ส่งคำสั่งที่อ่านได้ไปยัง CPU)
- Decoding (ถอดรหัสคำสั่งโดย CPU)
- Executing (ประมวลผลคำสั่ง โดย CPU)
- Buffering (เก็บผลลัพธ์จากการประมวลผลไว้ใน Memory ชั่วคราว เพื่อรอการประมวลผลเสร็จสิ้น)
- Output (การแสดง Output ออกทางอุปกรณ์แสดงผลต่างๆ)
- Storing Data (การเก็บผลลัพธ์จากการประมวลผลไว้ใน Memory

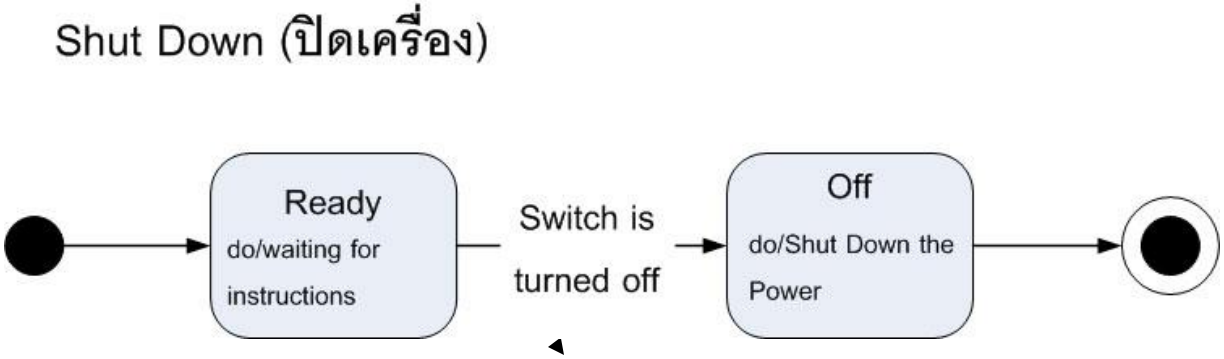
State Diagram

State Diagram ของแต่ละ Method เป็นดังนี้



การเปิดเครื่องคอมพิวเตอร์ เริ่มต้นที่เครื่องคอมพิวเตอร์ที่อยู่ใน State Off ได้รับการเปิด (Switch is turned on) จึงทำให้คอมพิวเตอร์เปลี่ยนไปอยู่ที่ State On และเมื่อเปิดเครื่องเสร็จเรียบร้อย การ Boot จึงเกิดขึ้น จนทำให้คอมพิวเตอร์มาอยู่ที่ State Boot และเมื่อ Boot เสร็จเรียบร้อยแล้ว เครื่องคอมพิวเตอร์จึงมาอยู่ใน State Ready ซึ่งรอรับคำสั่งและพร้อมจะทำงานต่อไป

State Diagram



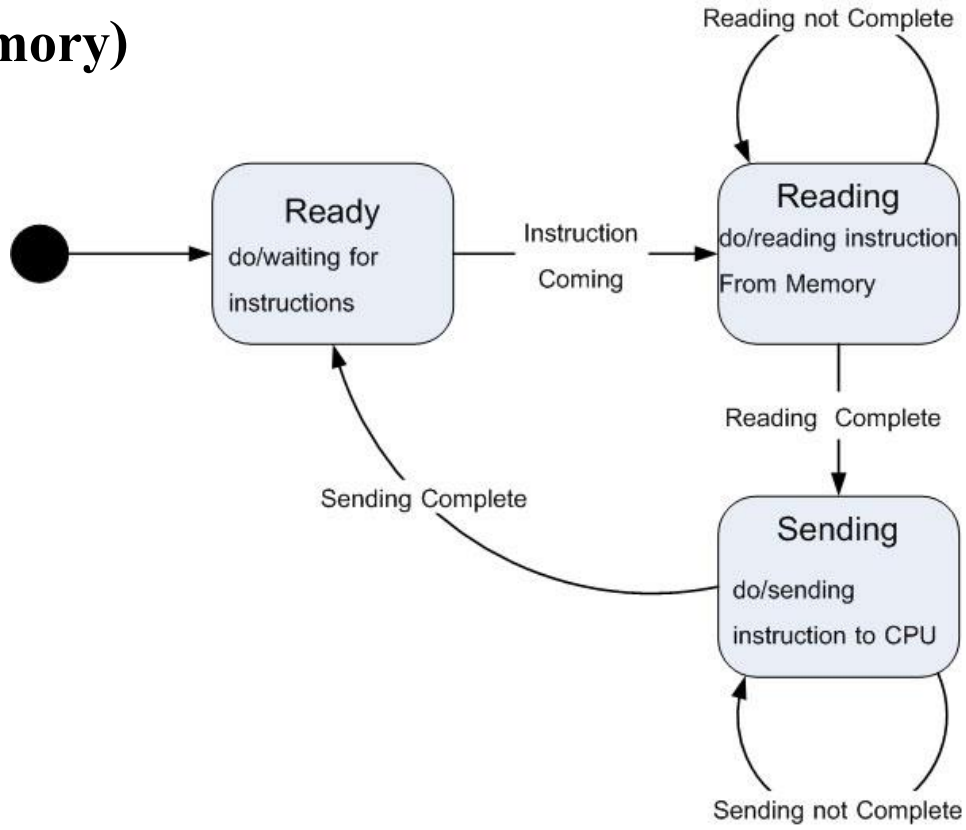
ในการปิดเครื่องคอมพิวเตอร์คือการเปลี่ยน State จาก Ready ซึ่งไม่มีการทำงานใด ๆ ไปยัง State Off ซึ่งการจะเปลี่ยน State เช่นนี้ได้ต้องใช้ Transition การปิดเครื่อง (Switch is turned off)

State Diagram

Read Instruction (อ่านคำสั่งจาก Memory)

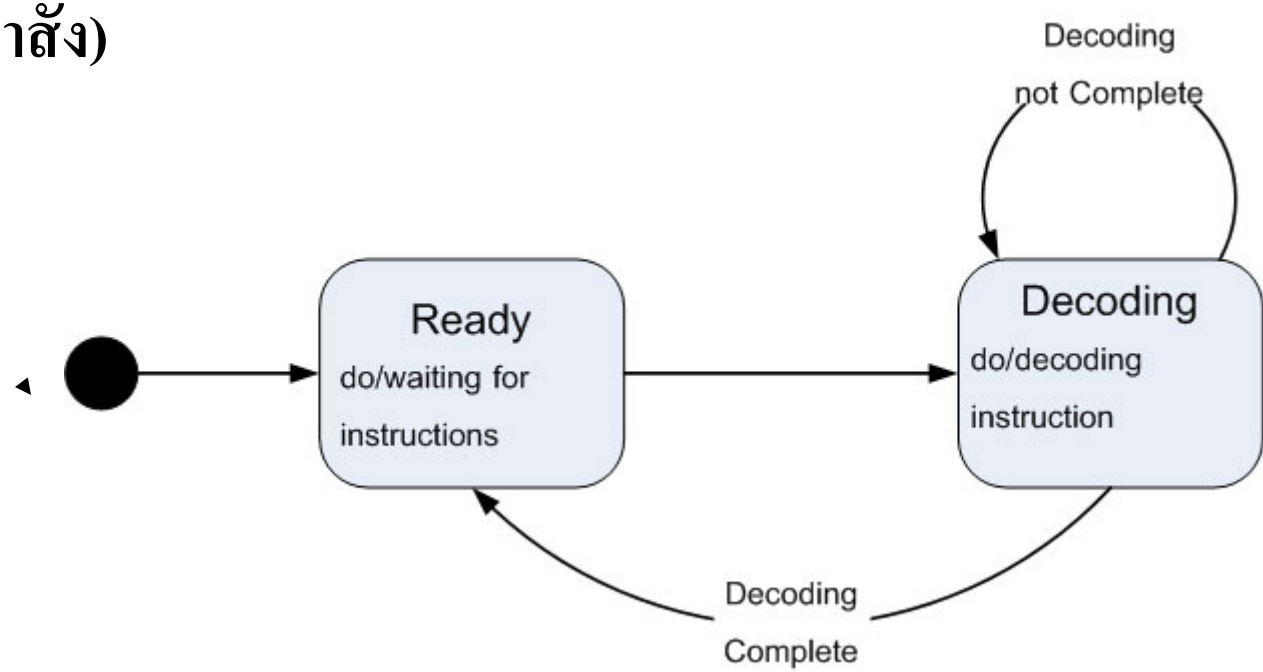
ในการอ่านคำสั่งใดๆ จาก Memory ของคอมพิวเตอร์นั้นจะเริ่มต้นใน State Ready ก่อน เมื่อมีคำสั่งเข้ามาใน Memory แล้ว ตาม Transition Instruction Coming คอมพิวเตอร์จะเริ่มเข้าไปยัง State Reading ซึ่งจะอ่านคำสั่งจาก Memory ทีละคำสั่งไปจนกว่าจะเสร็จสิ้น (Reading Complete)

ซึ่งเมื่ออ่านเสร็จแล้ว คอมพิวเตอร์จะส่งคำสั่งที่อ่านได้ไปยัง CPU ดังระบุไว้ใน State Sending และคอมพิวเตอร์จะวนอยู่ใน State นี้ (ดังจะเห็นจาก Transition Sending not Complete) จนกว่าจะเสร็จสิ้น จึงกลับเข้าไปยัง State Ready



State Diagram

Decode (การถอดรหัสคำสั่ง)

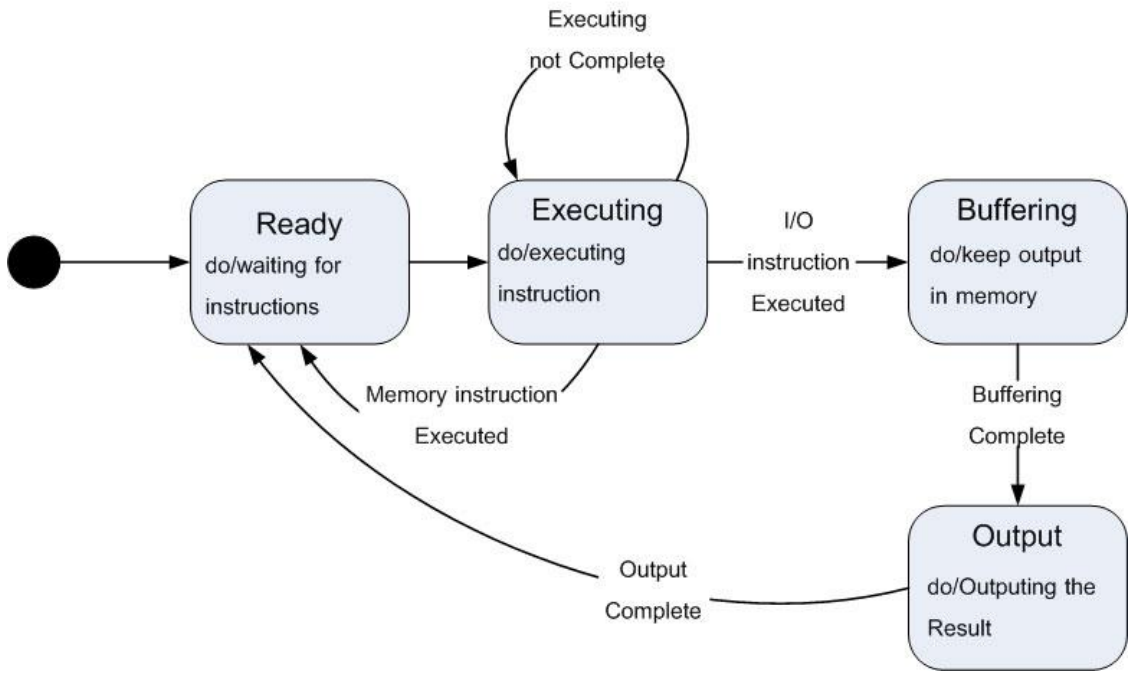


ในการถอดรหัสคำสั่ง โดยไม่ต้องมีเงื่อนไขใด ๆ คอมพิวเตอร์จะเปลี่ยนจากสถานะ Ready มายังสถานะ Decoding ซึ่งใน State นี้ คอมพิวเตอร์จะถอดรหัสคำสั่งทีละคำสั่งจนกว่าจะหมด และเมื่อการถอดรหัสเสร็จสิ้นแล้ว (Decoding Complete) จึงกลับไปอยู่ในสถานะ Ready เพื่อรอคำสั่งใหม่ต่อไป

State Diagram

Execute (การประมวลผล)

การประมวลผลในคอมพิวเตอร์จะเริ่มต้นที่ State Ready แล้วเข้าไปยัง State Executing ซึ่งจะวนอยู่ใน State นี้ จนกระทั่งคำสั่งถูกประมวลผลเสร็จสิ้น



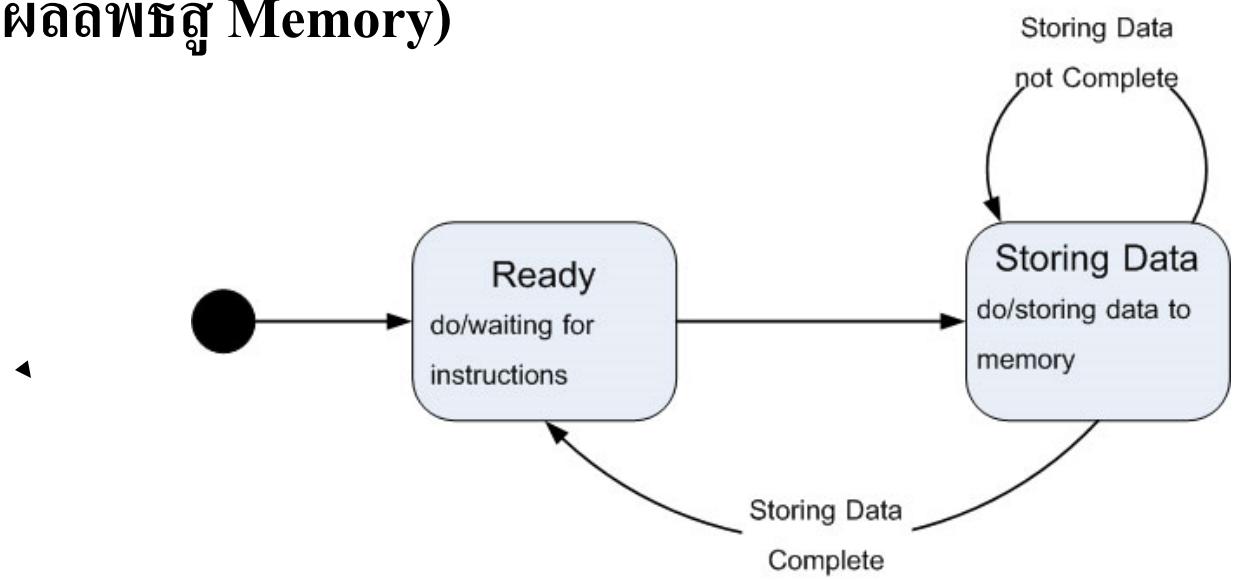
ซึ่งการประมวลผลเสร็จสิ้นนั้นแบ่งออกเป็น2 แบบ คือ การประมวลผลคำสั่งเกี่ยวกับ Memory และการประมวลผลทาง Input / Output

ซึ่งการเสร็จสิ้นการประมวลผลทาง Memory นั้นคอมพิวเตอร์จะย้ายกลับไปยัง State Ready

ในขณะที่ เมื่อการประมวลผลทาง Input / Output เสร็จสิ้น คอมพิวเตอร์จะย้ายไป State Buffering ซึ่งเป็นการบันทึกผลการประมวลผลไว้ใน Memory เพื่อรอการนำออกไปยังอุปกรณ์ Output หลังจากที่การทำ Buffering เสร็จเรียบร้อย จะเข้าไปยัง State Output ซึ่งเมื่อเข้าไปยัง State นี้ จะนำผลที่ได้ออกทางอุปกรณ์ และเมื่อการนำข้อมูลออกแสดงทางอุปกรณ์ Output แล้วจึงกลับมาสู่ State Ready ตามเดิม

State Diagram

Store Data (การบันทึกผลลัพธ์สู่ Memory)



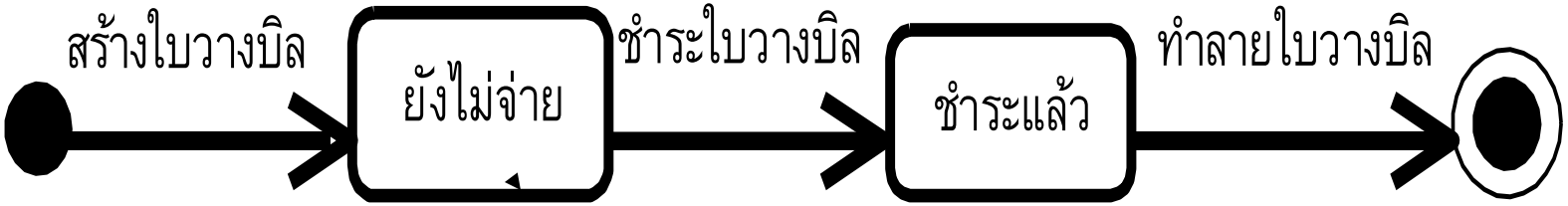
การ Store Data เริ่มต้นที่ State Ready แล้วเข้าสู่ State Storing Data ซึ่งจะบันทึกข้อมูลใน Memory จนกว่าจะครบถ้วนในทุก ๆ หน่วยข้อมูล หลังจากนั้นจึงกลับเข้าสู่ State Ready ตามเดิม

State Diagram

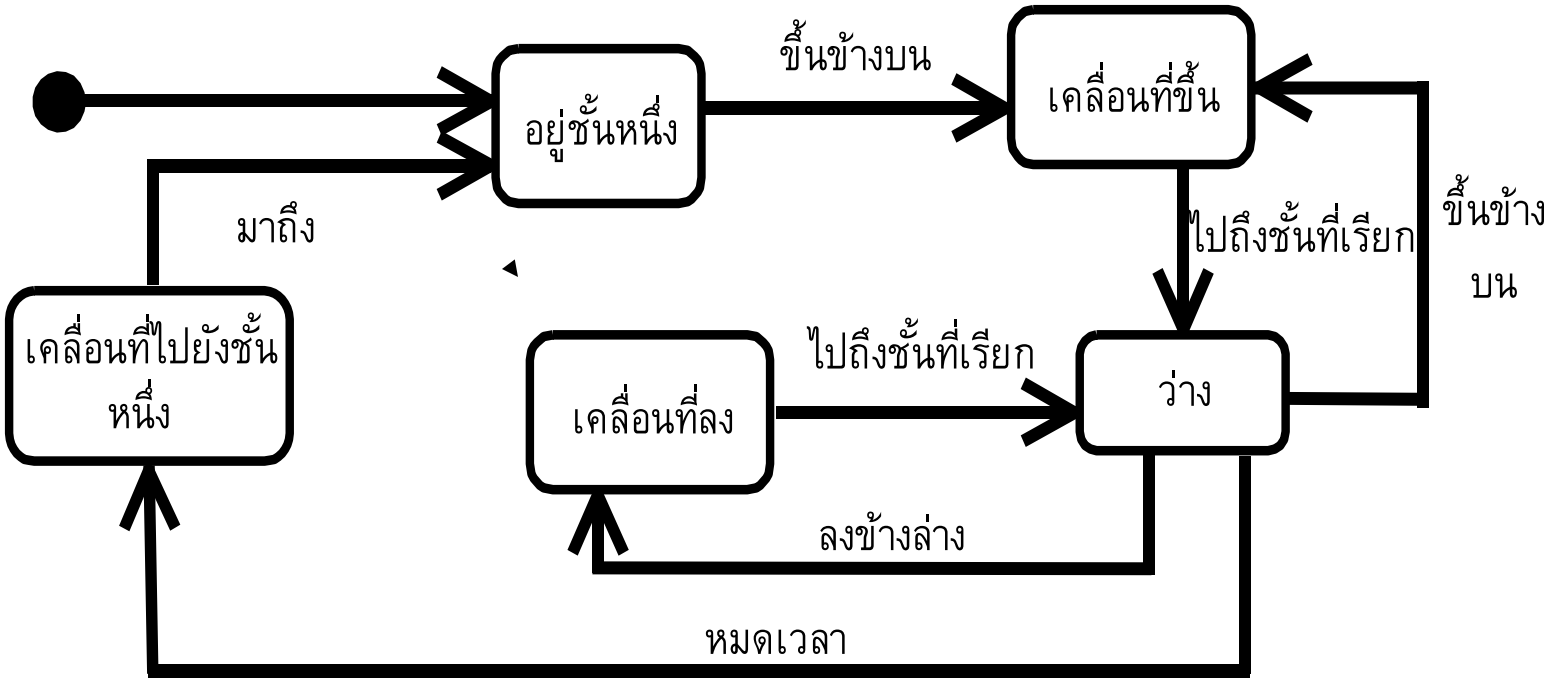
ข้อควรคำนึงในการเขียน State Diagram ใน Analysis Phase คือ ต้องเขียน State ให้ครบในภาพรวมทั้งหมดก่อน โดยยังไม่ต้องคำนึงถึงรายละเอียดของแต่ละ State และ Transition มากนัก

แต่ที่สำคัญคือต้องไม่มี State และ Transition ใดตกหล่น หรือหายไป แล้วขั้นตอนของ Design Phase นั้น เราจะทำให้ State Diagram มีความละเอียดมากขึ้นจนสามารถนำไปสร้างเป็นโปรแกรมได้ต่อไป

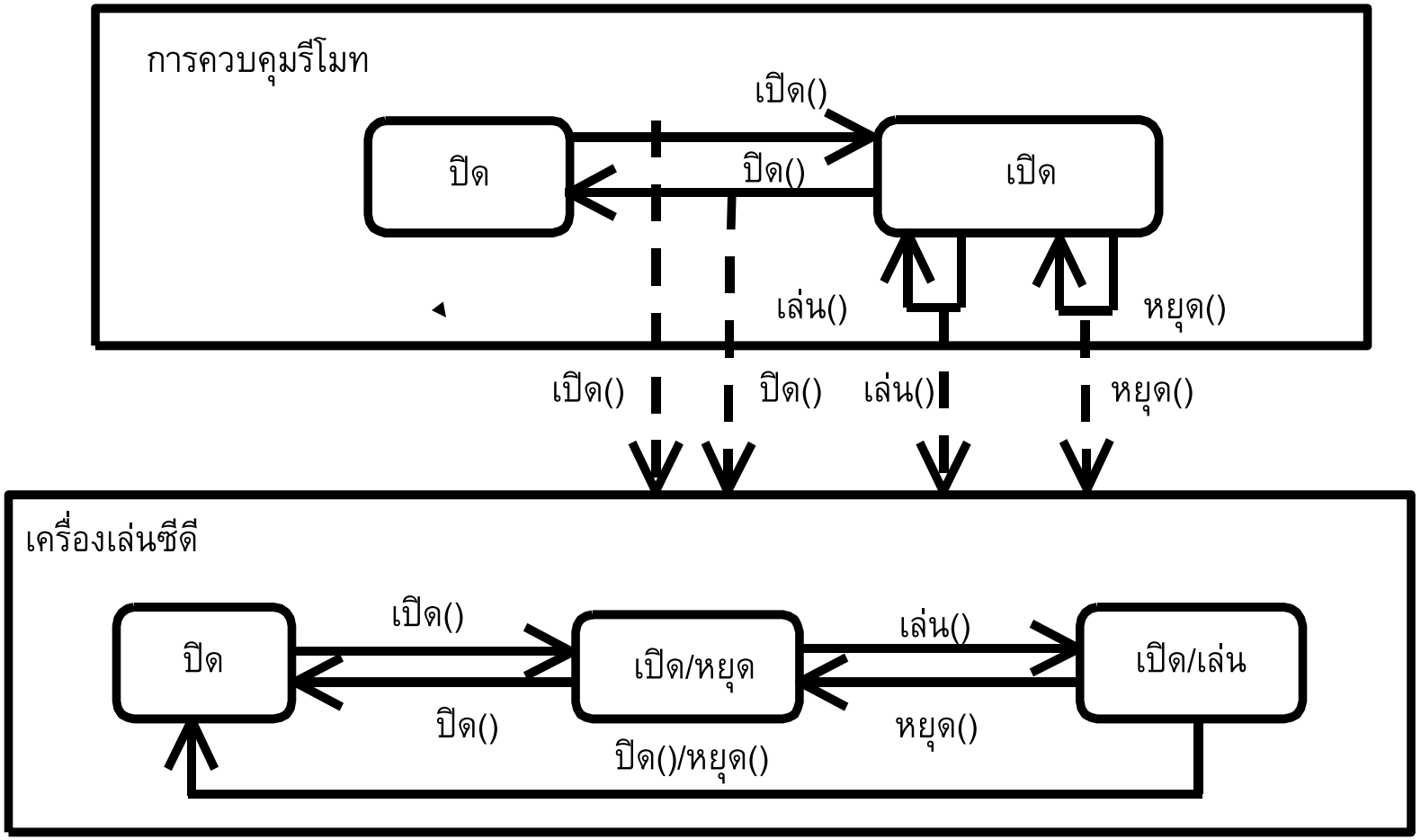
ตัวอย่าง State Diagram



ตัวอย่าง State Diagram



ตัวอย่าง State Diagram



แบบฝึกหัด

รถเด็กเล่นชนิดหนึ่ง เมื่อเปิดสวิตซ์จะวิ่งไปมา แต่เมื่อมาเจอขอบของวัตถุเช่นขอบโต๊ะซึ่งอาจจะทำให้มันหล่นลงพื้นได้ มันจะหยุด และจะเลี้ยวซ้าย แต่ถ้าเลี้ยวซ้ายแล้วเจอสิ่งกีดขวางจะเลี้ยวขวาแทน แต่ถ้าไปไม่ได้จะถอยหลัง และในที่สุดถ้าถอยหลังไม่ได้มันจะปิดสวิตซ์ตนเองโดยอัตโนมัติ

Software Engineering

State Diagram และ Activity Diagram

(ต่อ)

Activity Diagram

State Diagram ใช้อธิบายการเปลี่ยนแปลงจาก State หนึ่งไปยังอีก State หนึ่ง

ส่วน Activity Diagram หรือแผนภาพแสดงกิจกรรม ใช้อธิบายกิจกรรมที่เกิดขึ้นในลักษณะกระแสการไหลของการทำงาน (workflow)

Activity Diagram จะมีลักษณะเดียวกับ Flowchart (แสดงขั้นตอนการทำงานของระบบ) โดยขั้นตอนในการทำงานแต่ละขั้นตอนซึ่งเรียกว่า Activity

ใช้ Activity Diagram

- อธิบาย กระแสการไหลของการทำงาน (workflow)
- แสดงขั้นตอนการทำงานของระบบ

Activity Diagram

Activity อาจเป็นการทำงานต่างๆ ได้แก่

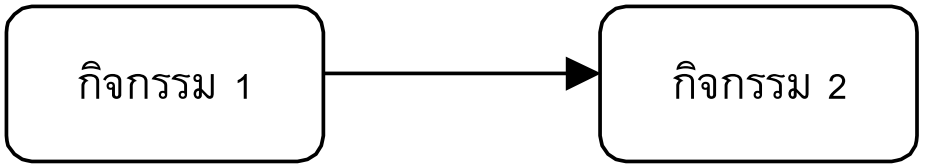
- การคำนวณผลลัพธ์บางอย่าง
- การเปลี่ยนแปลงสถานะ (State) ของระบบ
- การส่งค่ากลับคืน
- การส่งสัญญาณ
- การเรียกให้ Operation (Method) อื่นๆเพื่อทำงาน
- การสร้าง หรือ ทำลายวัตถุ

Activity Diagram

สัญลักษณ์ใน Activity Diagram

1.กิจกรรม (Activity)

2.เส้นทางการไหลของกิจกรรม

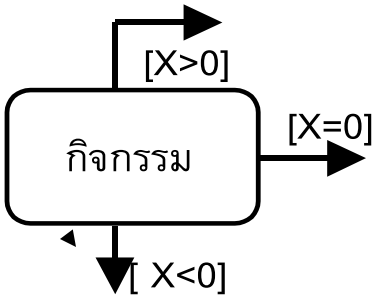


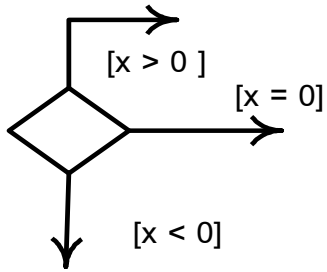
กรณี Synchronization และ Join

Activity Diagram

สัญลักษณ์ใน Activity Diagram

กรณีมีเงื่อนไข

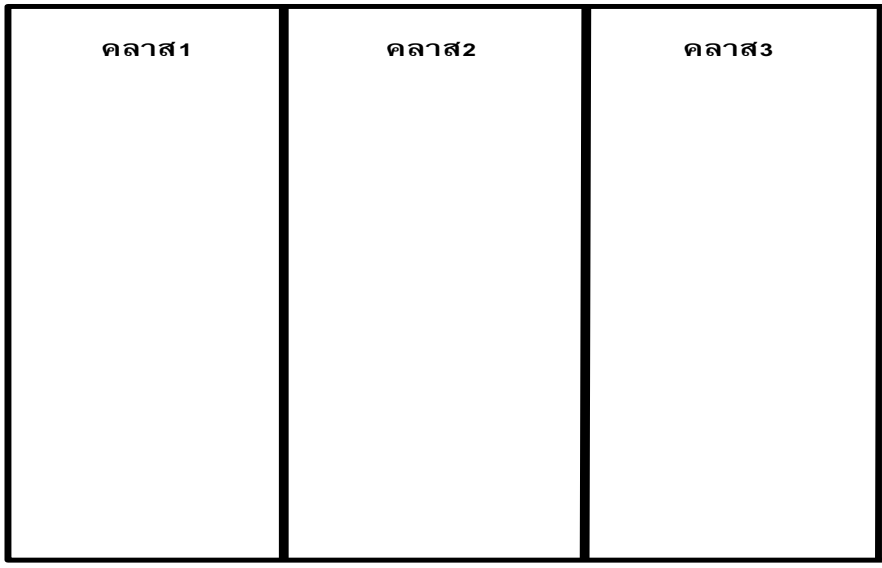




3. จุดเริ่มต้น ●

4. จุดสิ้นสุด ◉

5. สวิมเลนส์ (SWIMLANES)

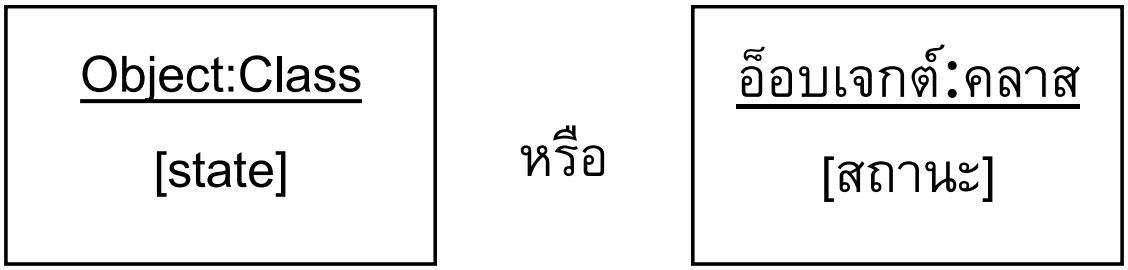


Activity Diagram

สัญลักษณ์ใน Activity Diagram

6. แสดงการไหลของอ็อบเจกต์ (Object Flow) (------▸)

7.



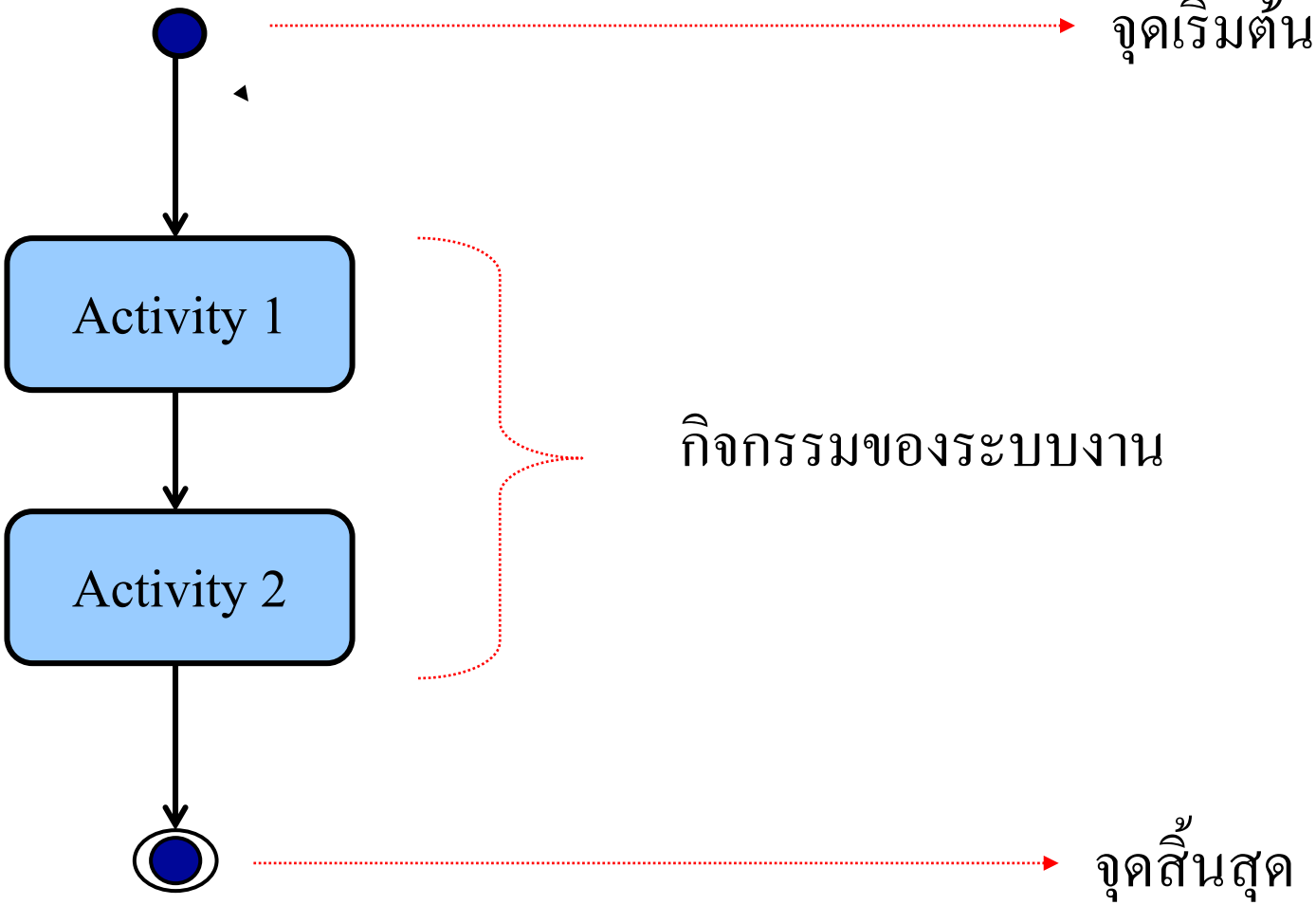
Activity Diagram

ขั้นตอนในการเขียน Activity Diagram

1. พิจารณากิจกรรมต่าง ๆ ที่ได้จากผลการวิเคราะห์ที่ควรอธิบาย
2. พิจารณากิจกรรมย่อยที่เกิดขึ้น เงื่อนไขหรือกรณีต่าง ๆ ที่เกิดขึ้น เมื่อเป็นไปตามเงื่อนไข
3. เรียงลำดับกิจกรรมที่เกิดก่อนหลัง
4. เขียนกิจกรรมย่อย ด้วยสัญลักษณ์แสดงกิจกรรม
5. เขียนจุดเริ่มต้น ●
6. เขียนจุดสิ้นสุด ◉

Activity Diagram

Activity Diagram จะต้องมีจุดเริ่มต้นและจุดสิ้นสุด และระหว่างจุดเริ่มต้นกับจุดสิ้นสุด ก็จะมีขั้นตอนหรือ activity ต่าง ๆ ของระบบ ดังรูป



Activity Diagram

รูปแบบการใช้ activity diagram มีหลายแบบ ได้แก่

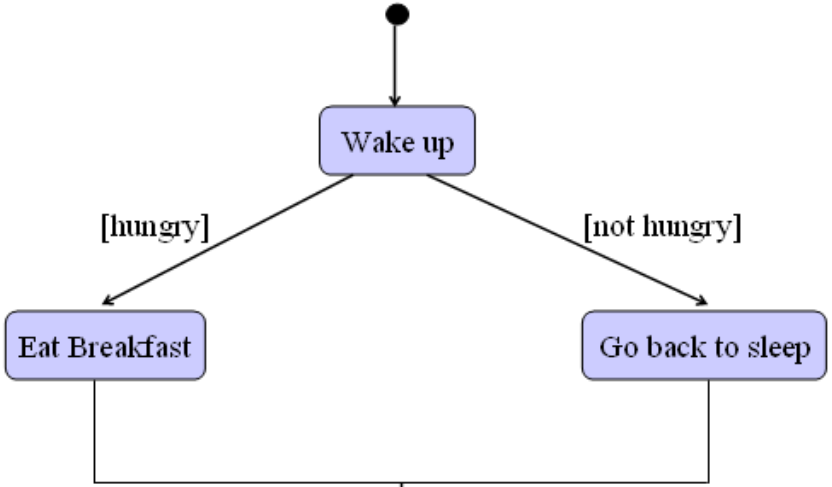
1. แบบทั่วไป
2. แบบมีทางเลือกให้ตัดสินใจ
3. แบบที่มีการทำงานพร้อม ๆ กันหลายงาน
4. แบบการส่งสัญญาณ

Activity Diagram

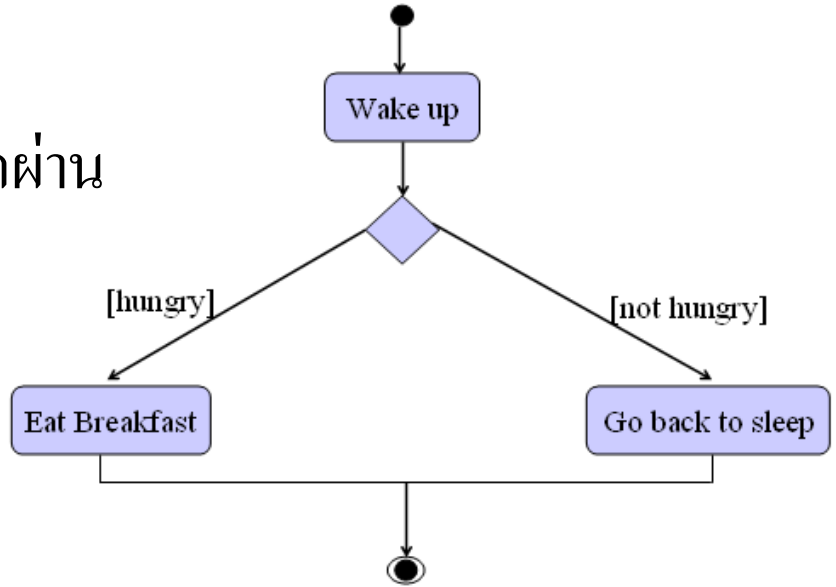
การสร้างทางเลือกใน Activity Diagram

สามารถทำได้ 2 วิธีคือ

1. ใช้ลูกศรของแต่ละทางเลือกไปยัง activity ผลลัพธ์ของทางเลือกโดยตรง

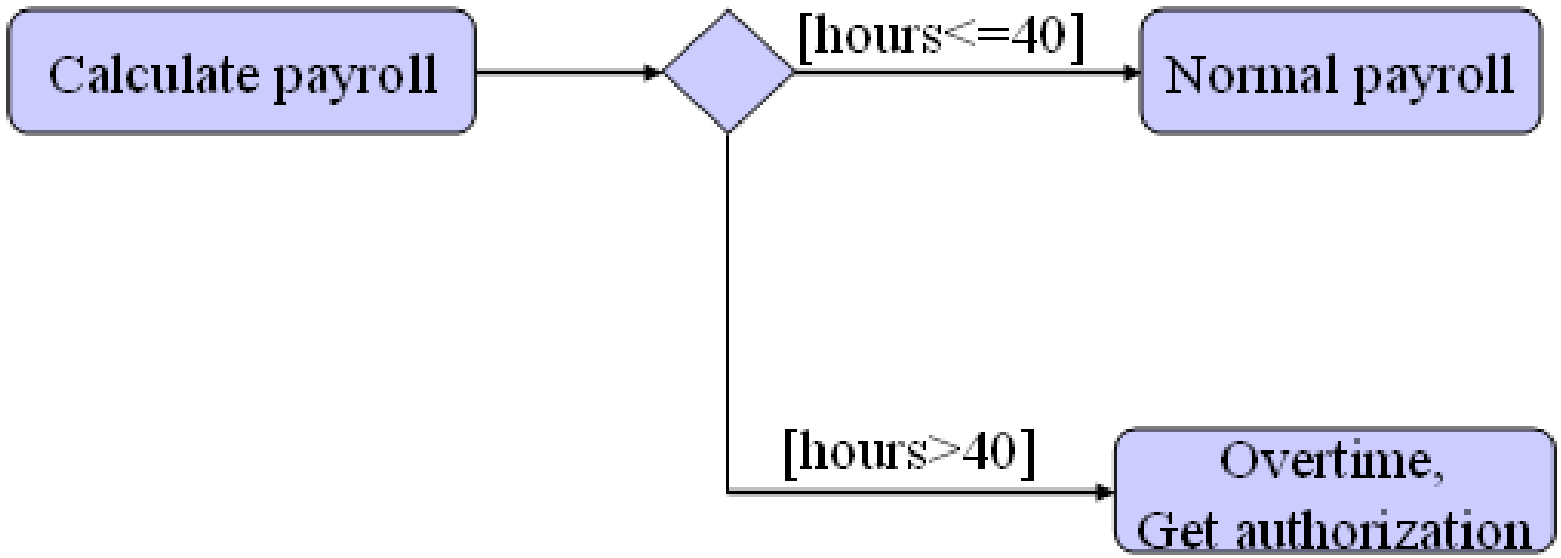


2. ใช้ลูกศรของแต่ละทางเลือกผ่านรูปสี่เหลี่ยมขนมเปียกปูนก่อน



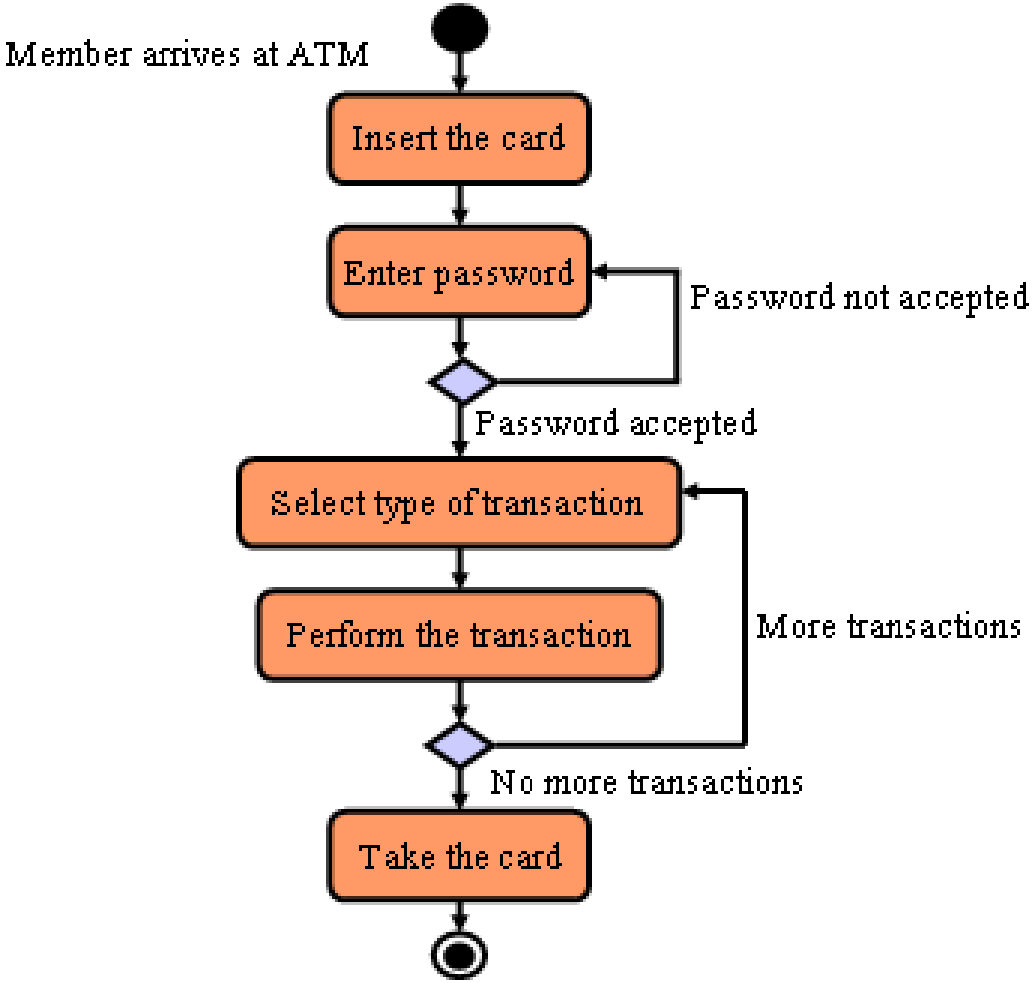
Activity Diagram

ตัวอย่าง Activity Diagram ที่มีทางเลือก



Activity Diagram

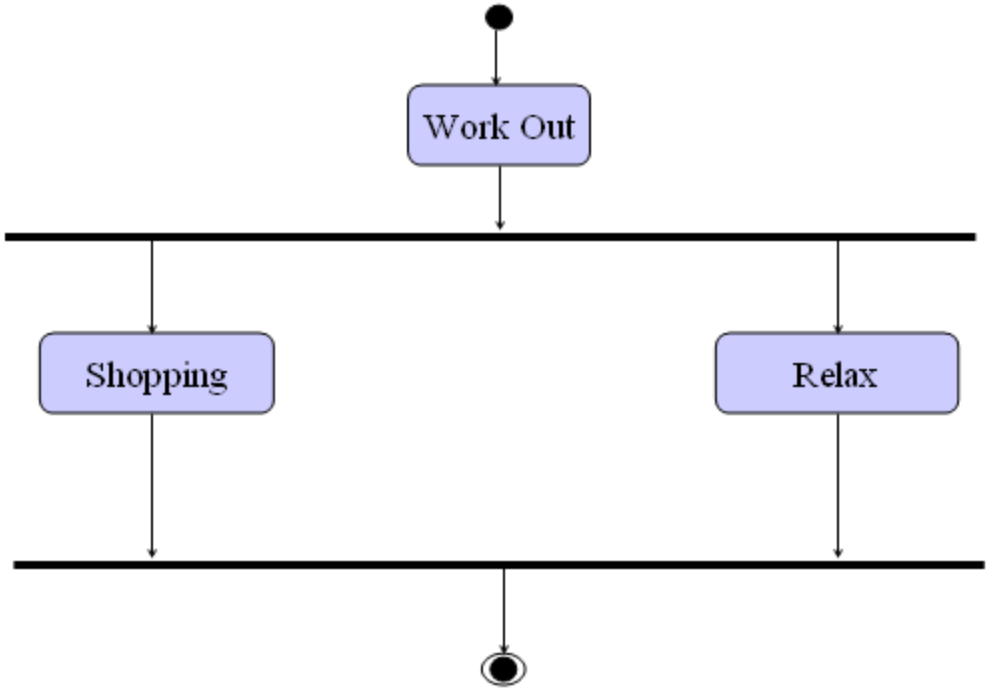
ตัวอย่าง Activity Diagram ของ ATM



Activity Diagram

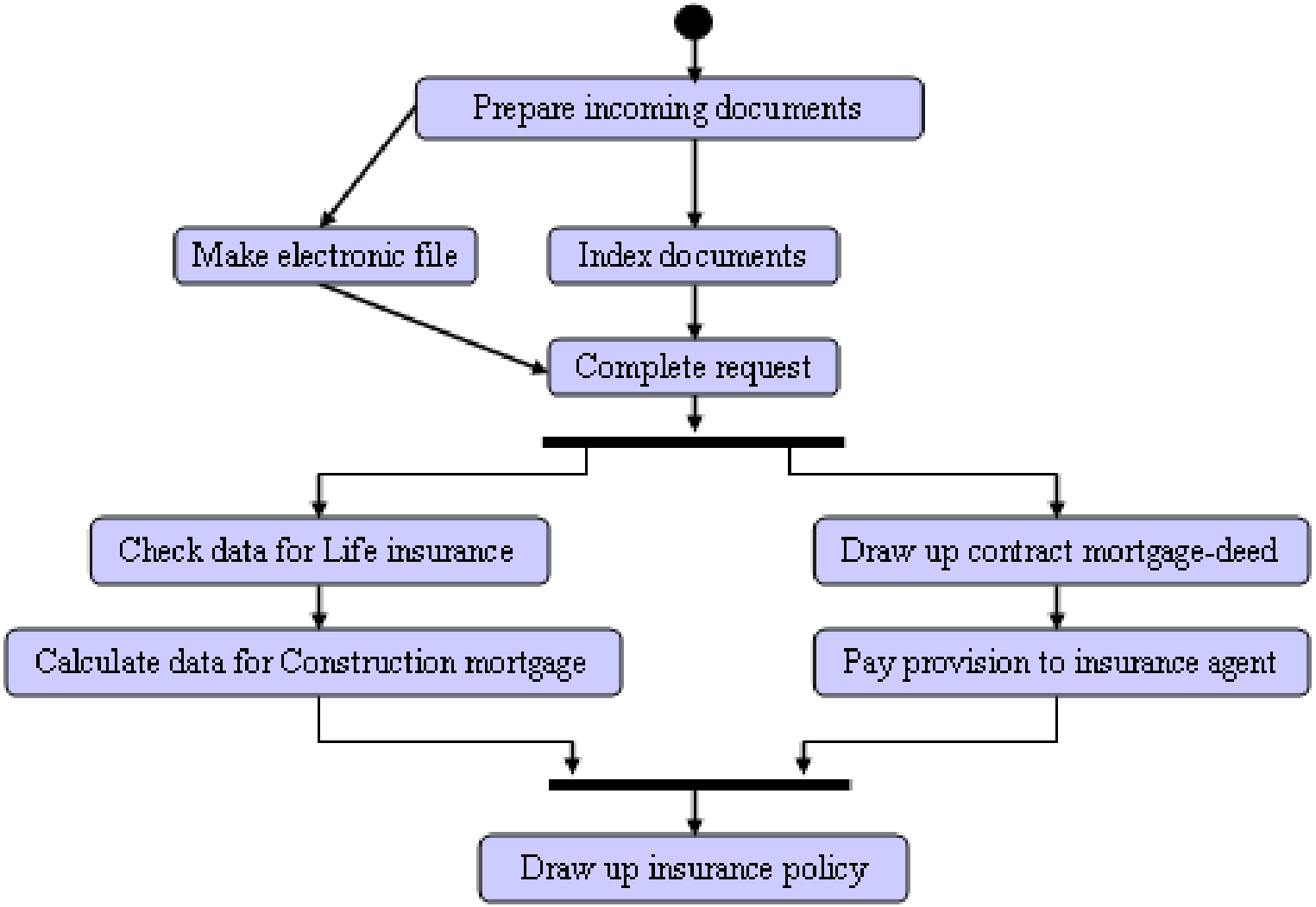
การทำงานหลายๆงานพร้อมกันใน Activity Diagram

ในกรณีที่เรามีงานหลายงานที่มีการทำงานไปพร้อมกัน จะใช้เส้นตรงแนวนอนเส้นหนา มาเป็นสัญลักษณ์ที่ใช้จัดกลุ่มงานที่มีการทำพร้อม ๆ กัน โดยมีลักษณะดังรูป



Activity Diagram

ตัวอย่าง Activity Diagram



Activity Diagram

Activity Diagram แสดงการส่งสัญญาณ

ในกระบวนการทำงานอาจเป็นไปได้ว่าจะมีการส่งสัญญาณบางอย่างในระหว่างการทำงาน เมื่อเกิดการส่ง-รับสัญญาณ เราก็จะเรียกว่าเกิด Activity ขึ้นเช่นเดียวกัน

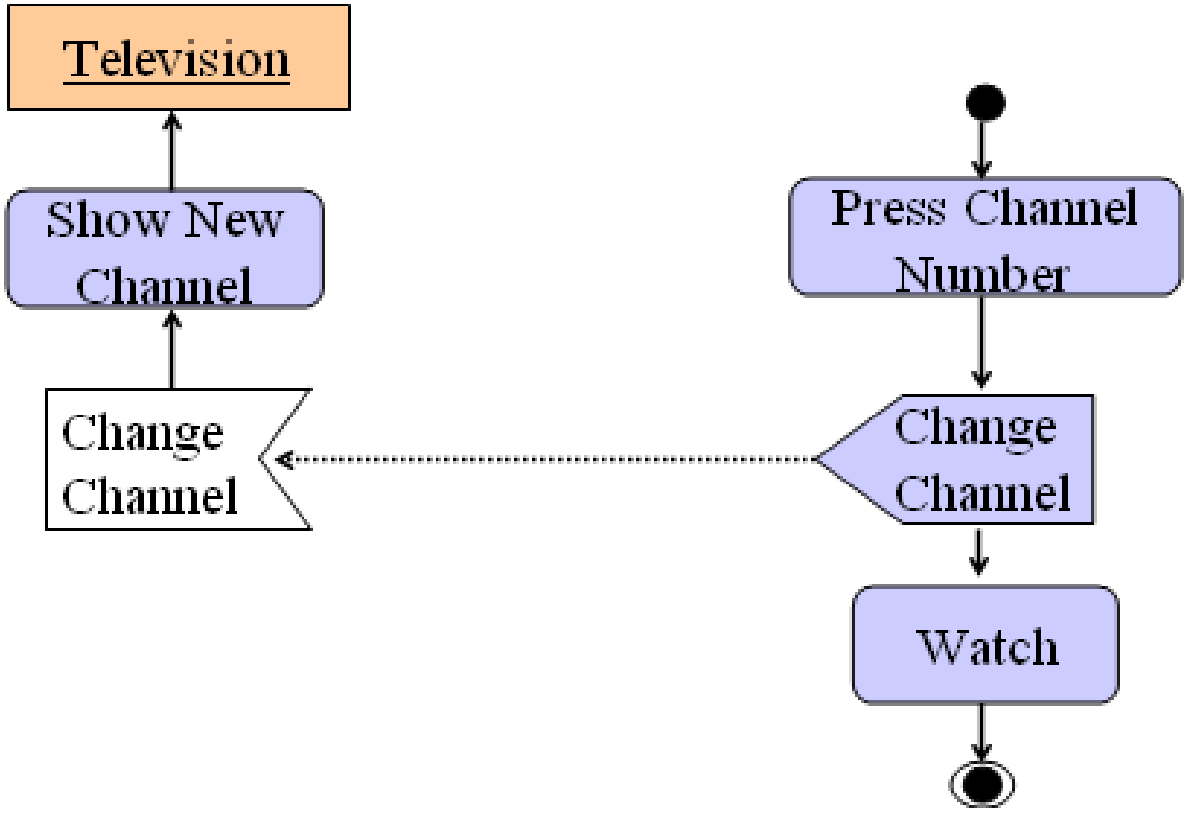
ในการเขียน Activity Diagram สำหรับการส่งสัญญาณ จะใช้รูปหลายเหลี่ยมแทน Activity ที่มีการส่งสัญญาณโดยที่

แทนเหตุการณ์ที่เป็น input

แทนเหตุการณ์ที่เป็น output

Activity Diagram

ตัวอย่าง แสดงการส่งสัญญาณ โดยระบบที่สนใจคือการกดปุ่มรีโมทคอนโทรล เพื่อเปลี่ยนช่องโทรทัศน์



Activity Diagram

แบ่งการทำงานให้เป็นสัดส่วนด้วย Swimlanes

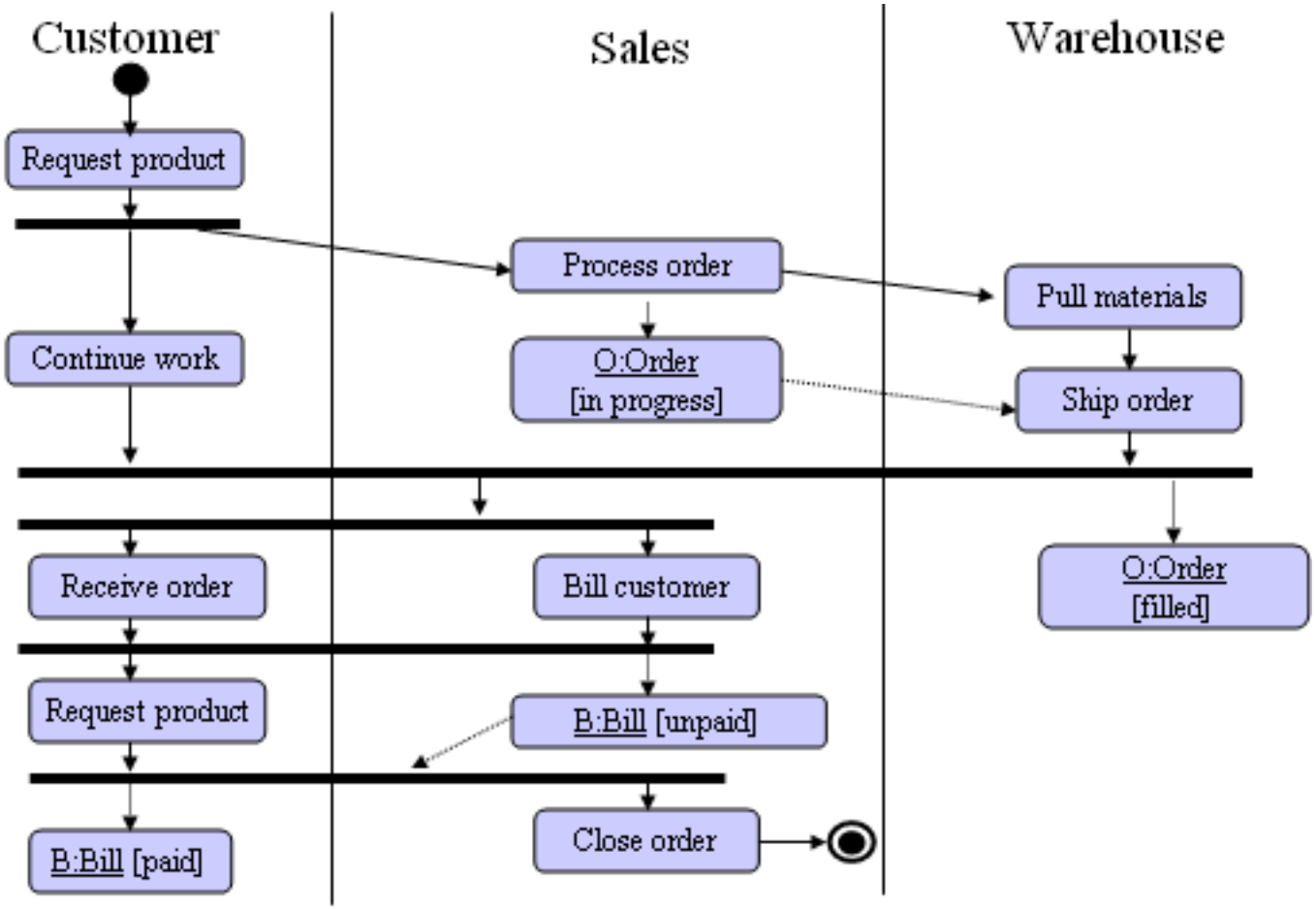
คุณลักษณะอีกอย่างหนึ่งคือสามารถแสดงให้เห็นได้ว่าใครเป็นผู้มีหน้าที่รับผิดชอบในแต่ละ activity ในกระบวนการทำงานหนึ่ง ๆ

หลักการของการแสดงหน้าที่ จะทำโดยการแบ่งกลุ่มของการรับผิดชอบเป็นกลุ่มๆ ซึ่งเปรียบเหมือนการแข่งว่ายน้ำ เรียกกลไกนี้ว่า **Swimlanes**

ในแต่ละ swimlane จะมีการกำหนดชื่อกำกับเอาไว้ เช่น กระบวนการของการสั่งซื้อสินค้า เราอาจแบ่งกลุ่มของคนที่มีส่วนเกี่ยวข้องเป็น 3 ส่วน ได้แก่ ลูกค้า , ฝ่ายขาย และคลังสินค้า

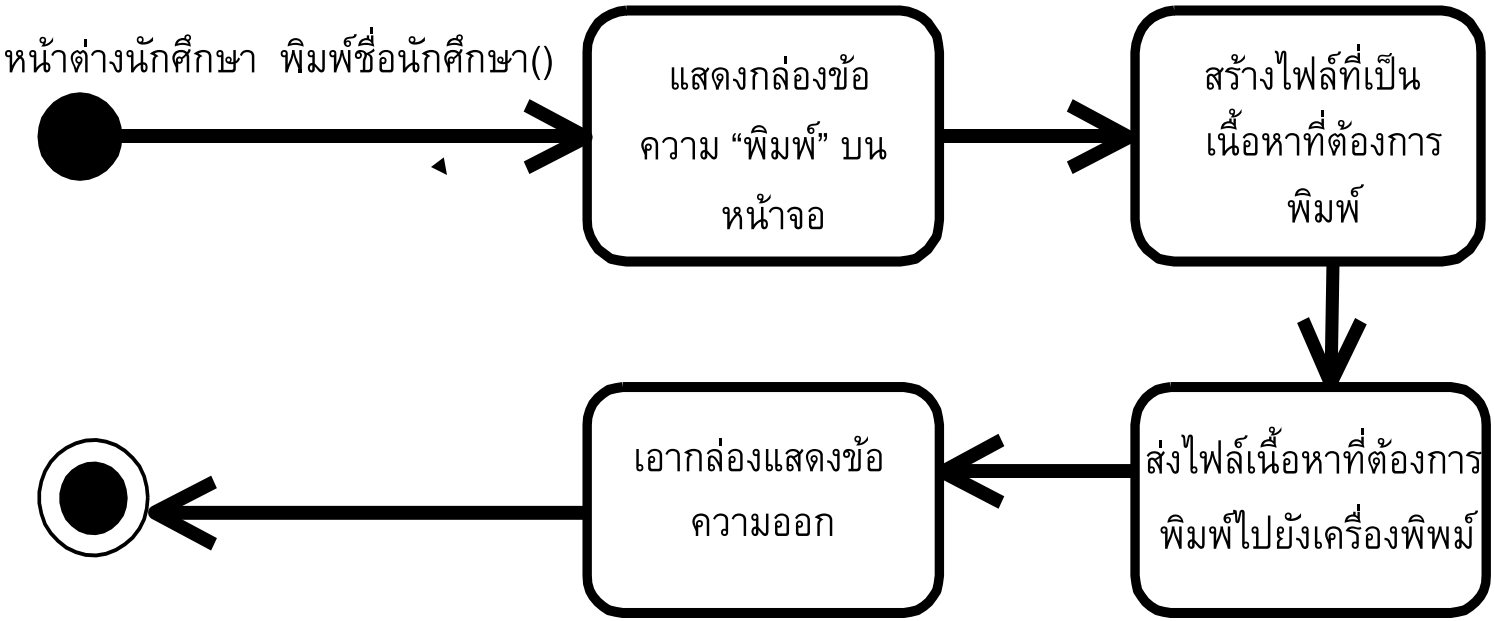
Activity Diagram

Activity หนึ่ง ๆ จะอยู่ภายใน 1 swimlane เท่านั้น แต่การติดต่อหรือส่งผ่านระหว่าง activity สามารถเกิดขึ้นข้ามจาก swimlane หนึ่งไปยังอีก swimlane หนึ่งได้ ดังรูปต่อไป



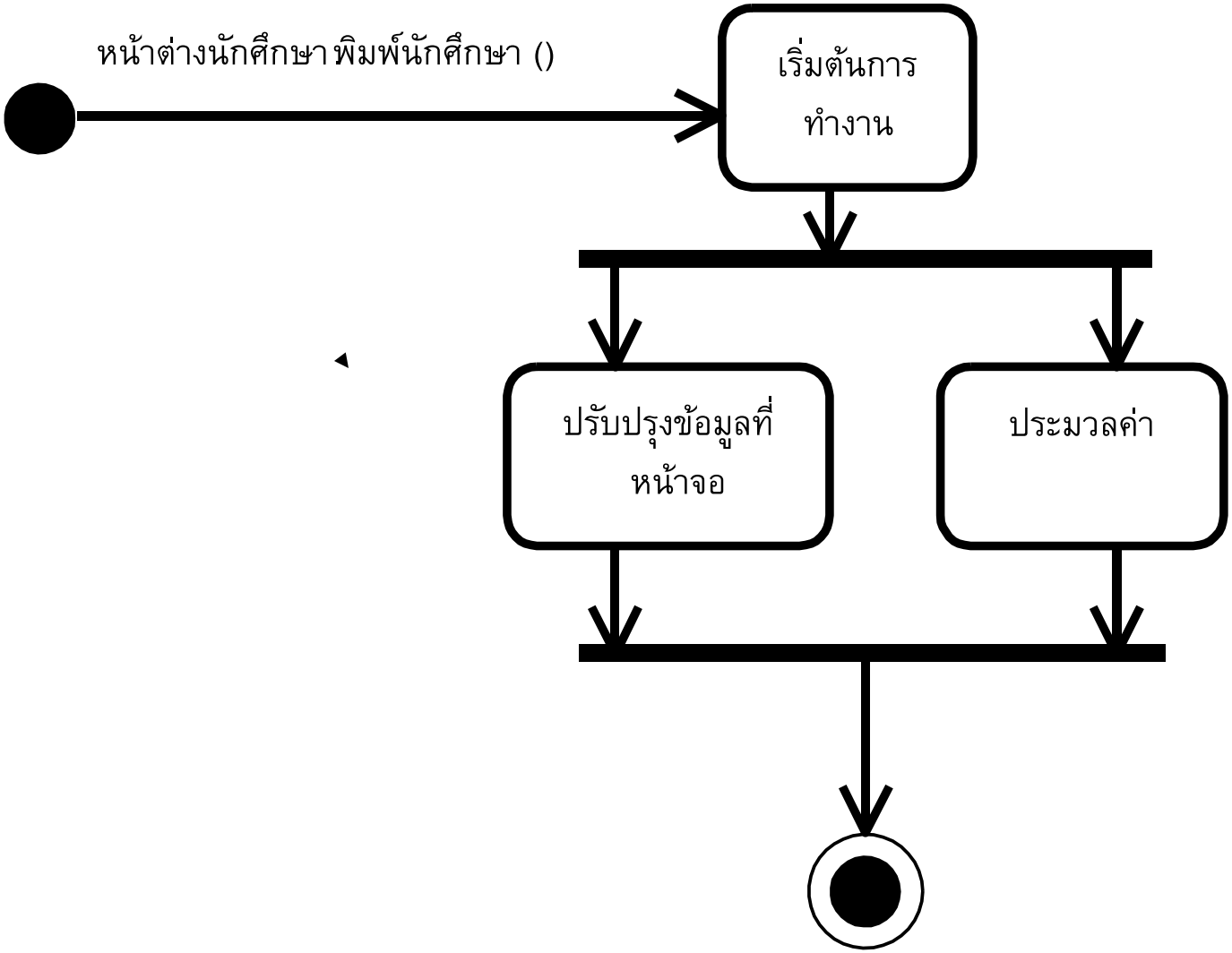
Activity Diagram

ตัวอย่าง



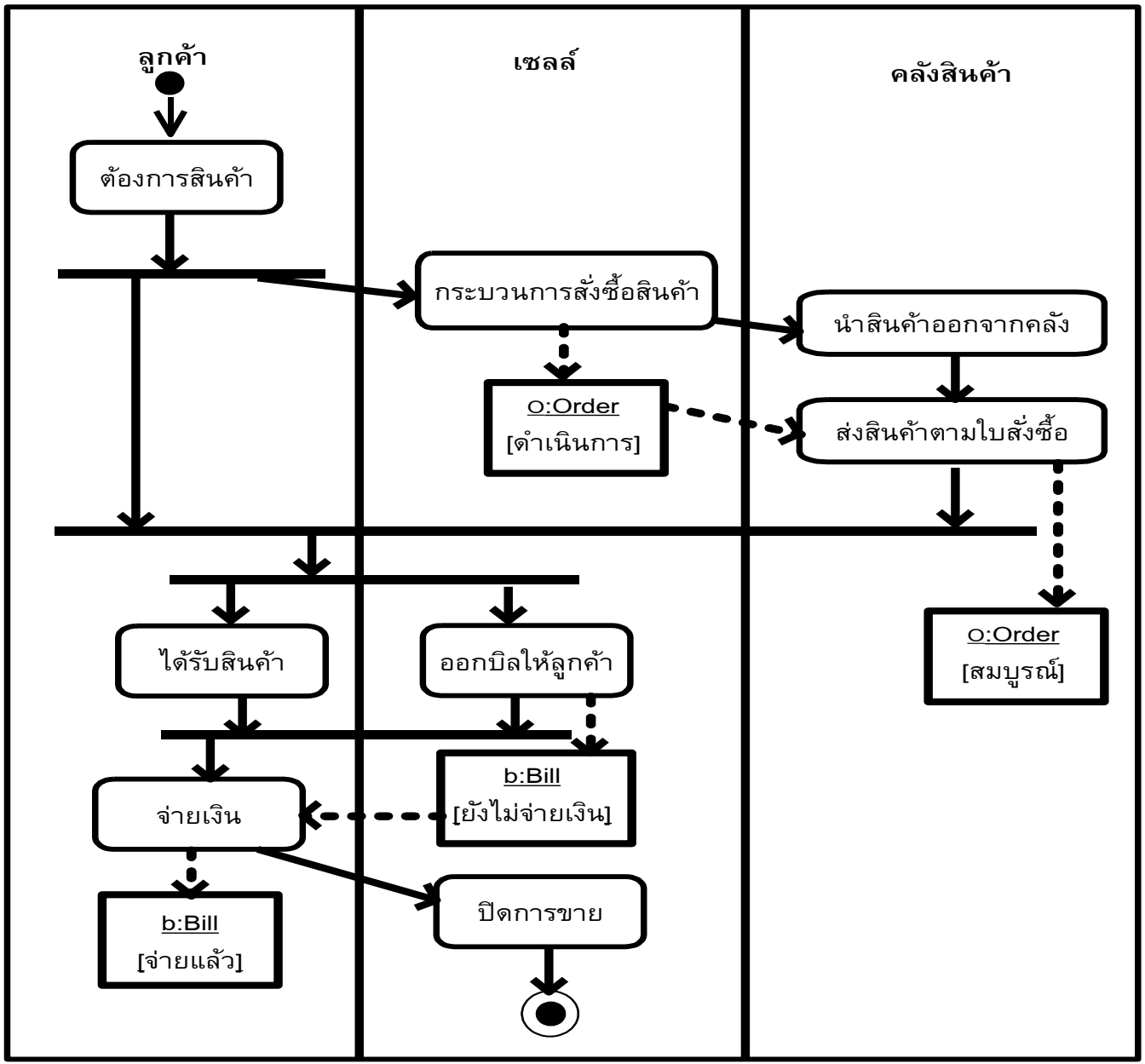
Activity Diagram

ตัวอย่าง



Activity Diagram

ตัวอย่าง



Activity Diagram

แผนภาพแสดงสถานะ (State Diagram) เป็นแผนภาพที่ใช้แสดงสถานะของ Object ต่างๆ ในระบบว่ามีสถานะอะไรบ้าง และจะเปลี่ยนแปลงสถานะไปตามเหตุการณ์ต่างๆ ที่เกิดขึ้น

แผนภาพแสดงกิจกรรม (Activity Diagram เป็นแผนภาพที่แสดงกิจกรรมที่เป็นงานย่อยของ Object ในแต่ละ Use Case สัญลักษณ์ที่ใช้ในการแสดงกิจกรรมจะเป็นสี่เหลี่ยมแคปซูล และมีเส้นลูกศร เพื่อแสดงลำดับของกิจกรรม โดยมีจุดเริ่มต้นและจุดสิ้นสุดเช่นเดียวกับแผนภาพแสดงสถานะ