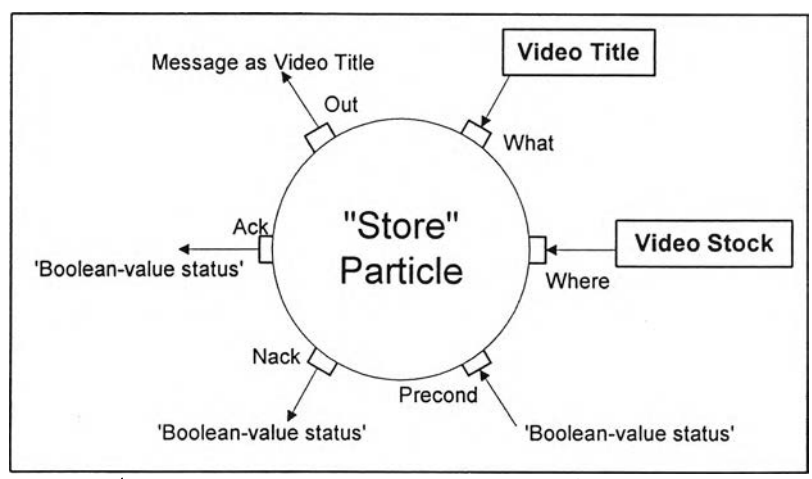
# บทที่ 2
# งานวิจัยและทฤษฎีที่เกี่ยวข้อง

## 2.1 งานวิจัยที่เกี่ยวข้อง

2.1.1 เครือข่ายอนุภาคความต้องการ: แนวคิดในการสร้างแบบจำลองความต้องการการทำงานของซอฟต์แวร์เชิงรูปนัย [5] (Requirements Particle Netwoks: An Approach to Formal Software Functional Requirements Modeling)

งานวิจัยนี้ได้นำเสนอแบบจำลองความต้องการการทำงานของซอฟต์แวร์ (Software Functional Requirements Modelling) โดยใช้เครือข่ายอนุภาคความต้องการ ความต้องการย่อยๆ แต่ละส่วนถูกนำมาเชื่อมต่อกันเพื่อใช้เป็นแบบจำลองประกอบการวิเคราะห์การออกแบบระบบซอฟต์แวร์ในขั้นตอนของการวิเคราะห์ระบบ และช่วยให้ทำการสร้างข้อกำหนดรูปนัยได้ง่ายสำหรับผู้ที่ไม่มีประสบการณ์ในการเขียนข้อกำหนดรูปนัยมาก่อน

งานวิจัยนี้ได้ใช้สัญกรณ์เซด (Z Notation) ในการอธิบายข้อกำหนดรูปนัย เนื่องจากสัญกรณ์เซดได้เขียนขึ้นมาจากทฤษฎีของเซต (Set) และตรรกศาสตร์ (Logic) ทางคณิตศาสตร์ ทำให้สามารถทำการพิสูจน์ความถูกต้องและสอดคล้องกันของข้อกำหนดได้ ส่งผลให้ซอฟต์แวร์มีคุณภาพมากขึ้น




รูปที่ 2.1 ตัวอย่างของอนุภาคความต้องการชื่อ “Store”

จากตัวอย่างของอนุภาคชื่อ “Store” ในรูปที่ 2.1 เมื่อใช้กฎและขั้นตอนวิธีที่กำหนดไว้มาเขียน เป็นข้อกำหนด รูปนัยภาษาเซด จะได้เป็น

Store $\widehat{=}$ [Video Title? : Video Titile Type; Video Stock?, Video Stock! : Video Stock Type; Ack! : Boolean; Nack! : Boolean; Out! : Out_type; Precond? : Boolean | Precond? $\wedge$ Video Stock! = Video Stock? $\cup$ {Video Title?} $\wedge$ Out! = Video Title? $\wedge$ Ack! = Video Title? $\notin$ Video Stock!]

งานวิจัยดังกล่าวใช้ภาษาเซดเป็นกรณีศึกษาสำหรับการเขียนข้อกำหนดรูปนัยของเครือข่ายอนุภาคความต้องการ แต่ยังไม่มีการศึกษาวิจัยการเขียนข้อกำหนดรูปนัยของเครือข่ายอนุภาคความต้องการด้วยภาษาคาเฟโอบีเจ

2.1.2 การเขียนข้อกำหนดรูปนัยจากแผนภาพสถานะ [8] (A Formal Specification Method from State Diagram)

งานวิจัยนี้ เป็นการนำเสนอกฎที่ใช้แปลงแผนภาพสถานะ (State Diagram) มาเป็นข้อกำหนดรูปนัยภาษาคาเฟโอบีเจโดยใช้แผนภาพสถานะของยูเอ็มแอล (UML) ในงานวิจัยนี้ได้กำหนดให้แต่ละสถานะของแผนภาพมีคุณสมบัติเป็นเสมือนวัตถุ (Object) ทั้งที่เป็นระบบพลวัต (Dynamic System) และระบบเสถียร (Static System) โดยได้ตั้งกฎขึ้นมา 6 ข้อ สำหรับการแปลงจากแผนภาพสถานะไปเป็นข้อกำหนดรูปนัยคาเฟโอบีเจ แล้วนำผลลัพธ์ที่ได้ไปพิสูจน์ความถูกต้องกับโปรแกรมแปลภาษาคาเฟโอบี กฎที่ตั้งขึ้นมาจะช่วยเป็นแนวทางให้ผู้ที่ไม่มีประสบการณ์ในการเขียนข้อกำหนดรูปนัย ภาษาคาเฟโอบีเจสามารถทำการเขียนได้รวดเร็วขึ้นจากแผนภาพสถานะที่ได้กำหนดไว้

2.1.3 การประกอบและแบ่งละเอียดของวัตถุโดยใช้การดำเนินการแบบไม่สังเกตค่าโปรเจ็คชัน: กรณีศึกษาระบบเครื่องรับจ่ายเงินอัตโนมัติ [9] (Object Composition and Refinement by using Non-Observable Projection Operators: A Case Study of the Automated Teller Machine system)

งานวิจัยนี้ ได้นำเสนอวิธีการเขียนข้อกำหนดเชิงพฤติกรรมของระบบเครื่องรับจ่ายเงินอัตโนมัติจากแผนภาพการประกอบของวัตถุ (Object Composition) โดยใช้วิธีการแทนที่ของวัตถุ (Object Replacement Methodology) ซึ่งใช้การประกอบกันของวัตถุ จากนั้นทำการแบ่งวัตถุให้มีความละเอียดมากขึ้น (Object Refinement) จนกระทั่งได้ข้อกำหนดที่มีความชัดเจนมากขึ้น การแบ่งละเอียดนั้นจะทำการแบ่งในลักษณะแบ่งทีละขั้น (Stepwise Refinement) ตัวดำเนินการที่ใช้จะใช้แบบนอนอ๊อบเซอเวเบิลโปรเจ็คชั่น (non-observable projection operator) ซึ่งตัวดำเนินการชนิดนี้จะช่วยให้ไม่จำเป็นต้องทราบรายละเอียดการสร้างของวัตถุหรือ การซ่อนของ

ข้อมูล (Information Hiding) ยังมีผลให้ช่วยการตรวจสอบความถูกต้อง (Verification) โดยจะกระทำการตรวจสอบเฉพาะคุณสมบัติเชิงพฤติกรรมในระดับที่สูงที่สุด เท่านั้น

จากงานวิจัยนี้ ทำให้ได้แนวคิดและตัวอย่างการเขียนข้อกำหนดรูปนัยคาเฟโอบีเจจากวิธีการเชิงวัตถุ ( Object Oriented) ได้เป็นอย่างดี

## 2.2 ทฤษฎีที่เกี่ยวข้อง

ในงานวิจัยนี้ มีทฤษฎีที่เกี่ยวข้องดังนี้

2.2.1 ข้อกำหนดรูปนัย (Formal Specification) [1,10]

วิธีรูปนัย เป็นวิธีการที่พัฒนาขึ้นโดยการนำเอาสัญกรณ์และทฤษฎีทางคณิตศาสตร์ที่ได้มีการกำหนดความหมายไว้อย่างดีแล้ว มาเป็นเครื่องมือในการอธิบายถึงพฤติกรรมของระบบเพื่อไม่ให้เกิดความกำกวมเหมือนกับที่เกิดขึ้นจากการใช้ภาษาธรรมชาติ และยังสามารถที่จะตรวจสอบกระบวนการพัฒนาซอฟต์แวร์ได้ตั้งแต่ขั้นตอนของการออกแบบโดยอาศัยการพิสูจน์ทางคณิตศาสตร์ ในขณะที่วิธีอรูปนัยและวิธีกึ่งรูปนัย จะสามารถทำได้ก็ต่อเมื่อได้ทำการพัฒนาซอฟต์แวร์ออกมาเป็นที่เรียบร้อยแล้ว

ภาษาข้อกำหนดรูปนัย ประกอบด้วยส่วนประกอบหลักๆ 3 ส่วนคือ

- วากยสัมพันธ์ สำหรับการอธิบายสัญกรณ์ข้อกำหนด ซึ่งใช้ในการอธิบายข้อ กำหนด
- ความหมาย (Semantic) สำหรับการอธิบาย "เอกภพสัมพัทธ์ของวัตถุ (Universal Objects)" ที่ใช้ในการอธิบายระบบ
- เซตของความสัมพันธ์ สำหรับอธิบายกฎในการแสดงคุณสมบัติของข้อกำหนด

ข้อกำหนดรูปนัยสามารถแบ่งออกเป็น 2 กลุ่มใหญ่ ๆ ด้วยกันคือ แบบโมเดลเบส (Model Based) และ แบบพร็อบเพอร์ตี้เบส (Property Based) ซึ่งถึงแม้ว่าทั้งสองแบบจะมีพื้นฐานมาจากภินทนศาสตร์ (Discrete Mathematics) เหมือนกันแต่จุดที่แตกต่างของทั้งสองแบบคือรูปแบบและวิธีการในการเขียนข้อกำหนดและแต่ละแบบ ก็เหมาะสมในการที่จะนำไปใช้ในสภาพแวดล้อมหรือระบบคอมพิวเตอร์ที่แตกต่างกัน

2.2.1.1 ข้อกำหนดแบบโมเดลเบส (Model based Specification)

เป็นการสร้างรูปแบบทางทฤษฎีเชิงนามธรรมของระบบที่จะทำการพัฒนาโดยการอธิบายสิ่งที่ ระบบต้องกระทำ ซึ่งจะเหมาะกับการประมวลผลแบบ 1 โปรเซส

2.2.1.1.1 ข้อกำหนดโดยปริยาย (Implicit Specification)

ข้อกำหนดโดยปริยายนั้นจะบอกรายละเอียดของการทำงานเพียงแค่ว่าต้องการอะไรเพื่อให้การทำงานสำเร็จ แต่จะไม่บอกว่าจะทำได้อย่างไรโดยแสดงในรูปของเงื่อนไขก่อนและหลังการทำงาน (Pre-, Post- condition)

2.2.1.1.2 ข้อกำหนดชัดแจ้ง (Explicit Specification)

ข้อกำหนดชัดแจ้ง เป็นข้อกำหนดที่บอกว่าจะทำอย่างไรให้การทำบรรลุถึงเป้าหมาย ซึ่งในส่วนของข้อกำหนดโดยปริยายต้องมีการทำให้ละเอียดขึ้น (Refinement) โดยการแปลงให้เป็นข้อกำหนดที่ชัดแจ้งมากขึ้น ซึ่งข้อกำหนดแบบนี้ยังถูกแบ่งออกเป็นกลุ่มย่อยอีกคือ

ก) ข้อกำหนดเชิงคำสั่ง (Imperative or State-based Specification)

ข้อกำหนดเชิงคำสั่งจะเป็นข้อกำหนดที่แสดงในรูปของการเปลี่ยนแปลงค่าของตัวแปรเป็นหลัก ซึ่งลำดับภายในข้อกำหนดเชิงคำสั่งนั้นมีความสำคัญเนื่องจากการเปลี่ยนลำดับของข้อกำหนดจะก่อให้เกิดผลลัพธ์ที่แตกต่างกัน

ข) ข้อกำหนดเชิงประกาศ (Declarative Specification)

ข้อกำหนดแบบเชิงประกาศจะเป็นข้อกำหนดที่จะแสดงในรูปของกลุ่มของสมการหรือความสัมพันธ์ของข้อมูลเป็นหลักโดยถูกแบ่งออกเป็น

- ข้อกำหนดเชิงฟังก์ชัน (Functional Specification) ข้อกำหนดที่ได้จะแสดงในรูปของฟังก์ชันไม่มีการใช้ตัวแปรในลักษณะของตัวแปรส่วนกลาง (Global Variable) หรือ ตัวแปรภายนอก (External Variable) การทำงานจะใช้เพียงการส่งผ่านค่าไปทางพารามิเตอร์ (Parameter) ของฟังก์ชันเท่านั้น
- ข้อกำหนดเชิงตรรกะ (Logical Specification) ข้อกำหนดที่ได้จะแสดงอยู่ในรูปของนิพจน์ที่สามารถหาค่าความเป็นจริงได้ โดยทั่วไปจะใช้ทฤษฎีตรรกศาสตร์ภาคแสดงในการเขียน

2.2.1.2 ข้อกำหนดแบบพร็อบเพอร์ตี้เบส (Property based or Axiomatic Specification)

ข้อกำหนดแบบพร็อบเพอร์ตี้เบสจะทำการอธิบายพฤติกรรมของระบบ โดยมีชุดของสัจพจน์ที่เป็นตัวกำหนด คุณสมบัติ (Property) ของระบบ ซึ่งคุณสมบัติของระบบ ณ ช่วงเวลาใดๆต้องสอดคล้องกับสัจพจน์ที่มีอยู่เท่านั้น โดยใช้ทฤษฎีตรรกศาสตร์ภาคแสดงอันดับแรกบนพื้นฐานของ ตรรกศาสตร์ของฮอร์ (Hoare Logic) ในการบอกถึงเงื่อนไขก่อนและหลังการทำงานในรูปของสัจพจน์ ซึ่งจะ เหมาะกับระบบกระจายและโพรโตคอล

2.2.1.2.1 ข้อกำหนดเชิงพีชคณิต (Algebraic Specification)

เป็นข้อกำหนดที่ใช้อธิบายชนิดข้อมูลเชิงนามธรรม (ADTs : Abstract Data Type) ที่อธิบายในรูปของการดำเนินการ (Operations) และความสัมพันธ์ระหว่างการดำเนินการของชนิดข้อมูลเชิงนามธรรม โดยมีส่วนประกอบ 4 ส่วนดังนี้

- ส่วนการแนะนำ เป็นส่วนที่ประกาศชนิดของข้อมูลหรือวัตถุ
- ส่วนการบรรยาย เป็นส่วนที่อธิบาย การดำเนินการของชนิดข้อมูลเชิงนามธรรม
- ส่วนการประกาศ (Signature) เป็นส่วนที่กำหนดวากยสัมพันธ์ของส่วนต่อประสาน (Interface) ของชนิดข้อมูลเชิงนามธรรมหรือวัตถุ
- ส่วนสัจพจน์ เป็นส่วนที่อธิบายถึงคุณสมบัติของการดำเนินการโดยเขียนในรูปของสมการเชิงพีชคณิต

2.2.2 คาเฟโอบีเจ [6,11]

คาเฟโอบีเจ เป็นภาษาที่ใช้อธิบายข้อกำหนดรูปนัยของระบบ ที่พัฒนามาจากภาษาโอบีเจ (OBJ) ซึ่งเป็นภาษาเชิงพีชคณิตที่ใช้ตรรกะในการอธิบายเพื่อให้สามารถอธิบายระบบได้อย่างชัดเจน โดยตรรกะที่เพิ่มเข้าไปคือ ตรรกะการเขียนใหม่ (Rewriting Logic) และพีชคณิตเชิงพฤติกรรม (Behavioral Algebra หรือ Hidden-sorted Algebra) โดยที่ตรรกะการเขียนใหม่ใช้ในการเขียนข้อกำหนดการทำงานที่พร้อมกัน (Concurrent Specification) และพีชคณิตเชิงพฤติกรรม ใช้ในการเขียนข้อกำหนดเชิงพฤติกรรม (Behavioral Specification) ซึ่งตรรกะที่เพิ่มเข้าไปนี้เป็นส่วนที่ใช้ในการเขียนข้อกำหนดเชิงวัตถุรูปนัย (Formal Object-Oriented Specification)

โครงสร้างของภาษาคาเฟโอบีเจจะประกอบไปด้วย 3 ส่วนคือ

2.2.2.1 ส่วนการประกาศมอดูล (Module Declaration)

ใช้สำหรับประกาศชื่อมอดูล การประกาศพารามิเตอร์ (ถ้ามี) และการประกาศการนำเข้า (Import) มอดูลอื่นมาใช้สำหรับการอธิบายคุณสมบัติของมอดูลปัจจุบัน โดยมอดูลในภาษาคาเฟโอบีเจ มีได้ 2 ลักษณะคือ

- มอดูลแบบยึดติด (Tight Modules หรือ Tight Denotation)
  คือมอดูลที่เป็นเอกเทศ (Unique) เพราะเมื่อ ทำให้เกิดผล (Implementation) แล้วจะสามารถทำได้แค่ 1 โมเดล (Model)

- มอดูลแบบอิสระ (Loose Modules หรือ Loose Denotation)
  คือมอดูลที่มีลักษณะเป็นคลาส ของโมเดลหรือมอดูลที่มีมากกว่า 1 โมเดล

การประกาศการนำเข้า มีอยู่ 3 แบบคือ

- การนำเข้าแบบป้องกัน (Protecting Mode) การนำเข้าแบบนี้จะไม่ยอมให้มีการเพิ่มหรือยุบส่วนย่อย (Element) ของมอดูลที่นำเข้า มาโดยเด็ดขาด
- การนำเข้าแบบขยาย (Extending Mode) การนำเข้าแบบนี้จะยอมให้มีการเพิ่มส่วนย่อยของมอดูลที่นำเข้ามาได้ แต่จะไม่ยอม ให้มีการยุบส่วนย่อย
- การนำเข้าแบบการใช้ (Using Mode) การนำเข้าแบบนี้สามารถที่จะเพิ่มหรือยุบส่วนย่อยของมอดูลที่นำเข้ามาได้ตามต้องการ

2.2.2.2 ส่วนการกำหนดสัญลักษณ์ทางฟังก์ชัน (Signatures Declaration)
ในส่วนนี้ใช้ในการกำหนดชนิดของข้อมูล (Sort) และสัญลักษณ์ทางฟังก์ชันที่กระทำกับ ชนิดข้อมูลที่ได้กำหนดไว้แล้ว โดยจะมีรูปแบบของแต่ละส่วนดังนี้

2.2.2.2.1 การกำหนดชนิดของข้อมูล (Sort Declaration) โดยจะแบ่งได้เป็น 2 ประเภทคือ

- ชนิดข้อมูลสังเกตค่าได้ (Visible Sort) เช่น จำนวนเต็ม (Integer) บูล (Boolean) เป็นต้น มีรูปแบบในการกำหนดดังนี้

  [sort_name1 sort_name2 sort_name3 ...]

  เมื่อ [ ] ใช้สำหรับชนิดข้อมูลที่สังเกตค่าได้
  sort_name1 sort_name2 sort_name3 ... คือ ชนิดข้อมูลที่สนใจ
- ชนิดข้อมูลแฝง (Hidden Sort) เช่น แถวลำดับของจำนวนเต็ม (Array of Integer) สวิตช์ (Switch) เป็นต้น ซึ่งค่าของชนิดข้อมูลจะทราบได้โดยผ่าน ชนิดข้อมูลที่สังเกตค่าได้คือ จำนวนเต็ม โดยมีรูปแบบในการกำหนดดังนี้

  *[ sort_name1 sort_name2 sort_name3 ...]*

เมื่อ *[ ]* ใช้สำหรับชนิดข้อมูลแฝง

sort_name1 sort_name2 sort_name3 ... คือชนิดข้อมูลที่สนใจ

ซึ่งการที่จะกำหนดว่าเป็นชนิดข้อมูลที่สังเกตค่าได้หรือชนิดข้อมูลแฝงนั้นขึ้นอยู่กับการพิจารณาของผู้เขียนข้อกำหนด

ในกรณีที่มีการกำหนดในลักษณะของชนิดข้อมูลย่อย (Subsort) หรือชนิดข้อมูลหนึ่งเป็นเซตย่อยของอีกชนิดข้อมูลหนึ่ง จะมีรูปแบบดังนี้

[sort_name1 < sort_name2]

*[sort_name1 < sort_name2]*

เมื่อ sort_name1 < sort_name2 คือ sort_name1 เป็นชนิดข้อมูลย่อยของ sort_name2

2.2.2.2.2 การกำหนดตัวดำเนินการ (Operator Declaration) สามารถแบ่งได้เป็น 3 ชนิดคือ

- ตัวดำเนินการเชิงพฤติกรรม (Behavioral Operator) เป็นตัวดำเนินการที่มีผลทำให้สถานะของชนิดข้อมูล หรือวัตถุ (Object) เปลี่ยนแปลง โดยตัวดำเนินการประเภทนี้จะมีอาร์กิวเมนต์ (Argument or Arity) อย่างน้อย 1 ตัวเป็นชนิดข้อมูลแฝงของมอดูลปัจจุบัน ซึ่งมีรูปแบบดังนี้

  bop op_name : list_of_sort → sort

  เมื่อ bop คือ ตัวดำเนินการเชิงพฤติกรรม
  op_name คือ ชื่อตัวดำเนินการ
  list_of_sort คือ รายการของอาร์กิวเมนต์หรืออาริตี้ (Arity) ที่มีอย่างน้อย 1 อาร์กิวเมนต์เป็นชนิดข้อมูลแฝงของมอดูลปัจจุบัน
  sort คือ ชนิดของผลลัพธ์หรือโคอาริตี้ (Coarity)

- ตัวดำเนินการสังเกตค่า (Observation Operator) เป็นตัวดำเนินการที่ไม่มีผลทำให้สถานะของชนิดข้อมูล หรือวัตถุเปลี่ยนแปลง แต่ใช้สำหรับการดูค่าของชนิดข้อมูล หรือวัตถุ ณ สถานะใด

สถานะหนึ่ง โดยชนิดของผลลัพธ์ของตัวดำเนินการประเภทนี้จะเป็นแบบสังเกตค่าได้ ซึ่งจะมีรูปแบบดังนี้

op op_name : list_of_sort → vsort

หรือ

bop op_name : list_of_sort → vsort

เมื่อ op_name คือ ชื่อตัวดำเนินการ

list_of_sort คือ รายการของอาร์กิวเมนต์

vsort คือ โคอาริตี้ หรือชนิดข้อมูลที่สังเกตค่าได้ (Visible Sort)

- ตัวดำเนินการค่าคงที่ (Constant Operator) เป็นตัวดำเนินการที่ใช้สำหรับการกำหนดค่าคงที่ของชนิดข้อมูลหรือวัตถุ โดยที่ตัวดำเนินการประเภทนี้จะไม่มีรายการของอาร์กิวเมนต์เลย ซึ่งมีรูปแบบดังนี้

op op_name → sort

เมื่อ op_name คือ ชื่อตัวดำเนินการ

sort คือ ชนิดข้อมูลใดๆ

2.2.2.3 ส่วนการกำหนดสัจพจน์ (Axioms Declaration)

เป็นส่วนที่ใช้ในการกำหนดพฤติกรรมของชนิดข้อมูลหรือวัตถุ ด้วยการใช้สมการทางคณิตศาสตร์มาเป็นตัวกำหนด โดยส่วนการกำหนดสัจพจน์จะมีรายละเอียดดังนี้

2.2.2.3.1 การกำหนดตัวแปร (Variable Declaration) ใช้สำหรับการระบุตัวแปรทั้งหมดที่ต้องใช้ในการเขียนสมการเพื่ออธิบายพฤติกรรมของชนิดข้อมูลหรือวัตถุ โดยมีรูปแบบดังนี้

var var_name : sort_name

หรือ

var var_name1, var_name2,... : sort_name

เมื่อ var_name คือ ชื่อของตัวแปร

var_name1, var_name2,... คือ รายการชื่อของตัวแปรที่มีชนิดข้อมูลเดียวกัน

sort_name คือ ชนิดข้อมูล

2.2.2.3.2 การกำหนดสมการ (Equation Declaration) สามารถแบ่งได้เป็น 2 ประเภทคือ

- สมการแบบไม่มีเงื่อนไข (Unconditional Equation Declaration) มีรูปแบบดังนี้

  eq term = term .

  beq term = term .

  เมื่อ eq เป็นการกำหนดสมการแบบไม่มีเงื่อนไข

  beq เป็นการกำหนดสมการเชิงพฤติกรรมแบบไม่มีเงื่อนไข

  term คือ การรวมกันของส่วนย่อย (Element)

- สมการแบบมีเงื่อนไข (Conditional Equation Declaration) มีรูปแบบดังนี้

  ceq term = term if boolean_term .

  bceq term = term if boolean_term .

  เมื่อ ceq เป็นการกำหนดสมการแบบมีเงื่อนไข

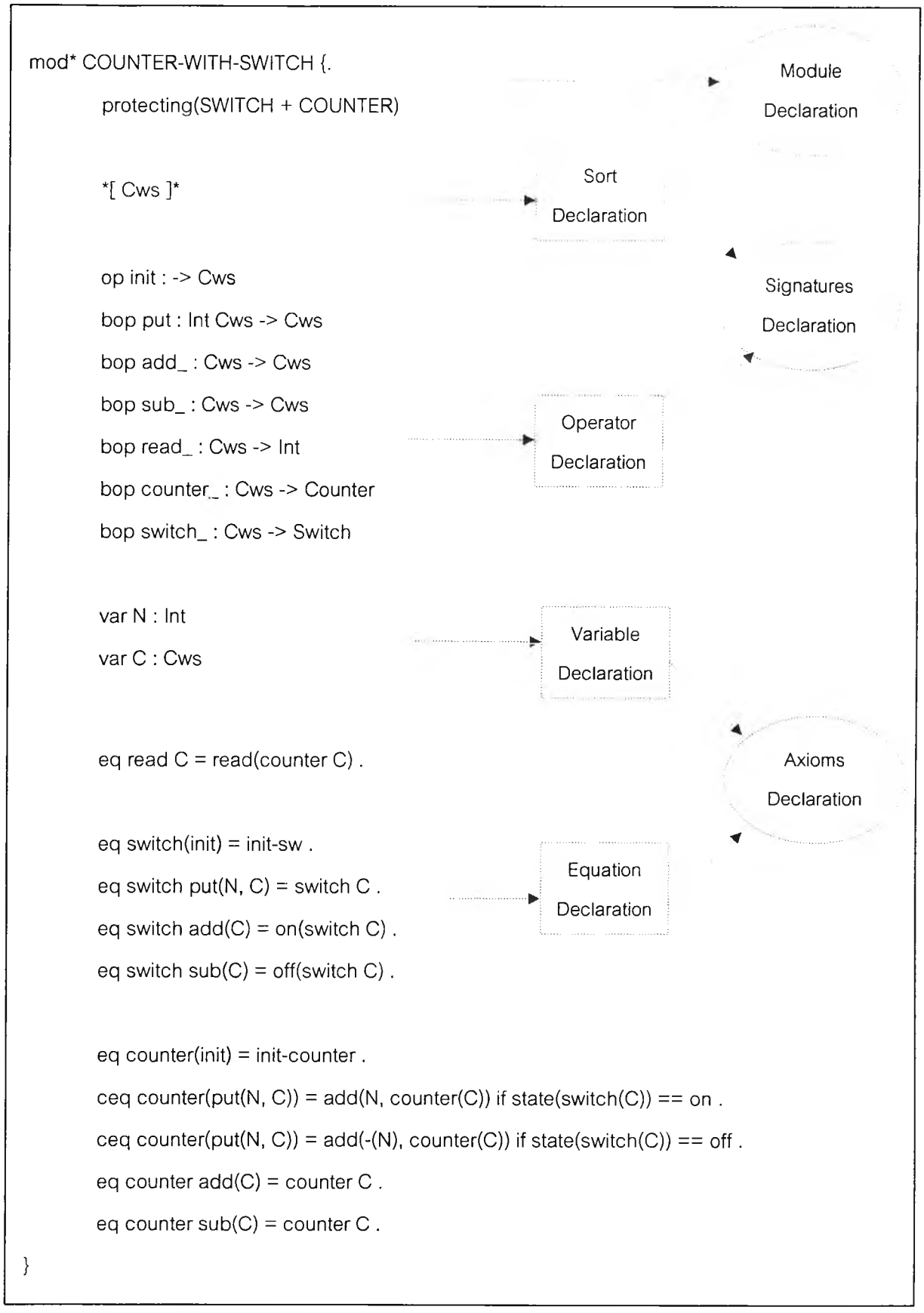
  bceq เป็นการกำหนดสมการเชิงพฤติกรรมแบบมีเงื่อนไข

  term คือ การรวมกันของส่วนย่อย (Element)

  boolean_term คือ term ที่ให้ผลลัพธ์เป็น จริง หรือ เท็จ

จากรูปที่ 2.2 แสดงมอดูลของข้อกำหนดนัยคาเฟโอบีเจของมอดูลการนับที่มีสวิตซ์เป็นตัวควบคุม โดยจะมีการนำเข้ามอดูลย่อย 2 มอดูลคือ

- มอดูลการนับ ที่ใช้สำหรับการบวก หรือลบเลขจำนวนเต็ม
- มอดูลสวิตซ์ ที่ใช้สำหรับความคุมว่าทำการบวกหรือการลบ โดยถ้าสวิตซ์เป็น on จะเป็นการ บวกเลข ถ้าสวิตซ์เป็น off จะเป็นการลบเลข




รูปที่ 2.2 ข้อกำหนดคาเฟโอบีเจของมอดูล COUNTER-WITH-SWITCH [12]

ส่วนประกอบของข้อกำหนดคาเฟโอบีเจจากรูปที่ 2.2 ได้มีการประกาศชื่อมอดูลชื่อ COUNTER-WITH-SWITCH และการประกาศการนำเข้าแบบป้องกันสำหรับมอดูลชื่อ SWITCH+COUNTER

ส่วนการกำหนดสัญลักษณ์ทางฟังก์ชัน ได้มีการกำหนดชนิดของข้อมูลแฝงชื่อ CWS และตัวดำเนินการต่างๆทั้งส่วนที่เป็นตัวดำเนินการเชิงพฤติกรรม และตัวดำเนินการสังเกตุค่าได้

ส่วนของการกำหนดสัจพจน์ ได้กำหนดตัวแปรชื่อ N เป็นข้อมูลชนิดตัวเลข และ C เป็นข้อมูลชนิด CWS เพื่อนำมาใช้ในสมการที่จะกำหนดในส่วนของการกำหนดสมการ ซึ่งการกำหนดสมการ มีทั้งแบบที่มีเงื่อนไขและไม่มีเงื่อนไข

2.2.3 เครือข่ายอนุภาคความต้องการ (Requirements Particle Networks) [4]

เครือข่ายอนุภาคความต้องการ หรือ อาร์พีเอ็น (RPN) เป็นการแสดงแผนภาพข้อกำหนดความต้องการอย่างหนึ่งที่ใช้ในการอธิบายการไหลของข้อมูล โดยจะมีลักษณะเป็นอนุภาคย่อยหลายๆ หน่วยต่อเชื่อมกันเป็นเครือข่าย

เครือข่ายอนุภาคความต้องการถูกเขียนแสดงในลักษณะของไฮเปอร์กราฟ ซึ่งในโครงสร้างจะประกอบไปด้วยอนุภาคความต้องการหรือโหนดและเส้นเชื่อมต่อระหว่างโหนด โดยที่โหนดมีสองแบบด้วยกันคือ โหนดการดำเนินการ (Operation node) และโหนดข้อมูล (Data entity node)

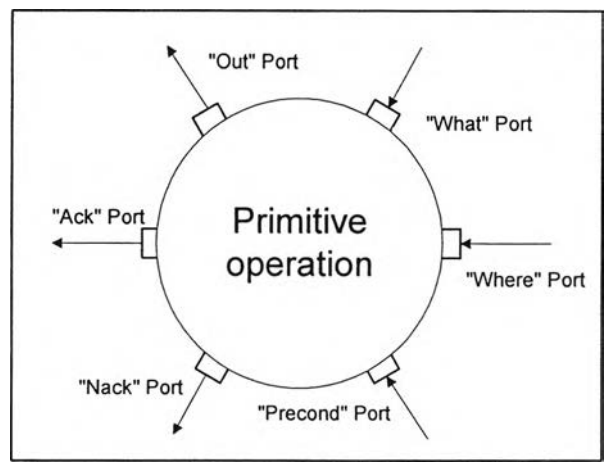
2.2.3.1 โหนดการดำเนินการสำหรับเครือข่ายอนุภาคความต้องการ

ในแต่ละโหนดของเครือข่ายอนุภาคความต้องการ จะประกอบไปด้วย ช่อง (port) สื่อสาร 6 ช่อง โดยเป็นช่องสำหรับเป็นข้อมูลเข้า (input) ข้อมูลออก (output) และเงื่อนไขสำหรับข้อมูลเข้า เงื่อนไขสำหรับข้อมูลออก ช่องการสื่อสารแต่ละช่องจะแยกกันโดยชนิดของข้อมูลและกลุ่มของข้อมูลในแต่ละช่อง ดังสัญกรณ์รูปภาพของโหนดการดำเนินการสำหรับเครือข่ายอนุภาคความต้องการ ในรูปที่ 2.3

สำหรับช่องการสื่อสารแต่ละช่อง จะมีชื่อสำหรับแต่ละช่องโดยช่องการสื่อสารสำหรับข้อมูลเข้า ประกอบไปด้วย "*What*", "*Where*" และ "*Precond*" ส่วนช่องการสื่อสารสำหรับข้อมูลออกประกอบไปด้วย "*Out*", "*Ack*" และ "*Nack*" โดยแต่ละช่องการสื่อสารมีหน้าที่ดังต่อไปนี้

- "*What*" ใช้สำหรับระบุค่าของข้อมูลเข้า ที่จะถูกกระทำโดยการดำเนินการพื้นฐาน
- "*Where*" ใช้สำหรับระบุเป้าหมายของเซตข้อมูลหรือที่จัดเก็บข้อมูล ที่เกี่ยวข้องในการดำเนินการพื้นฐาน

- "*Precond*" ใช้รับค่าตัวแปรที่เข้ามายังการดำเนินการพื้นฐาน โดยตัวแปรที่เข้ามา จะบอกสถานะของเงื่อนไขก่อนหน้า (Precondition) และผลลัพธ์ของค่าที่เข้ามาจะมีค่าเป็นจริง
- "*Out*" ใช้ในการส่งต่อข้อความไปยังการดำเนินการถัดไป โดยข้อความที่ถูกส่งต่อไปอาจเป็นได้ทั้งค่าของข้อมูลหรือเซตของข้อมูล
- "*Ack*" ผลของการกระทำของการดำเนินการที่สามารถกระทำการสำเร็จ ค่าที่เป็นจริงจะถูกส่งต่อออกไป
- "*Nack*" จะให้ค่าตรงกันข้ามกับช่องสื่อสาร Ack โดยจะส่งค่าที่เป็นเท็จออกไป หากการกระทำของตัวดำเนินการไม่ประสบความสำเร็จ



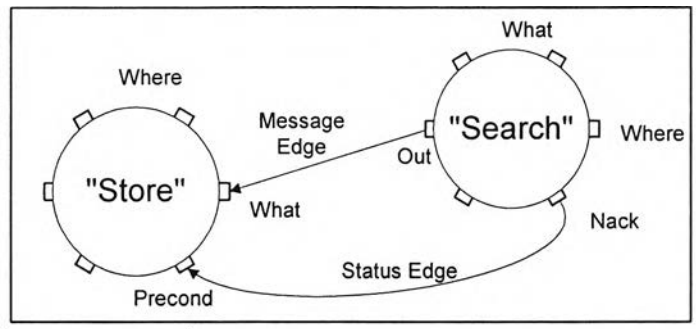

รูปที่ 2.3 สัญกรณ์รูปภาพโหนดของเครือข่ายอนุภาคความต้องการที่แสดงการดำเนินการพื้นฐาน

2.2.3.2 เส้นเชื่อมในเครือข่ายอนุภาคความต้องการ

การติดต่อกันระหว่างเครือข่ายอนุภาคความต้องการแต่ละโหนด จะกระทำผ่านช่องสื่อสารซึ่งจะมีเส้นเชื่อมระหว่างโหนดเป็นตัวถ่ายทอดทั้งข้อมูลเข้าและข้อมูลออก โดยเส้นเชื่อมจะมี 2 ประเภทคือ เส้นเชื่อมแบบ "ข้อความ" (Message) และเส้นเชื่อมแบบ "สถานะ" (status)

- เส้นเชื่อมแบบ "ข้อความ" ใช้สำหรับส่งผ่านข้อมูลจากโหนดหนึ่งไปยังโหนดอื่นตามที่ได้ระบุไว้ โดยแต่ละเส้นเชื่อมแบบข้อความ จะต่อเข้ากับช่องสื่อสารเพียงช่องเดียวเท่านั้นทั้งที่โหนดต้นทางและโหนดปลายทาง
- เส้นเชื่อมแบบ "สถานะ" ใช้สำหรับระบุทิศทางของค่าทางตรรกศาสตร์ (จริงหรือเท็จ) ระหว่างช่องสื่อสารต้นทางและปลายทาง

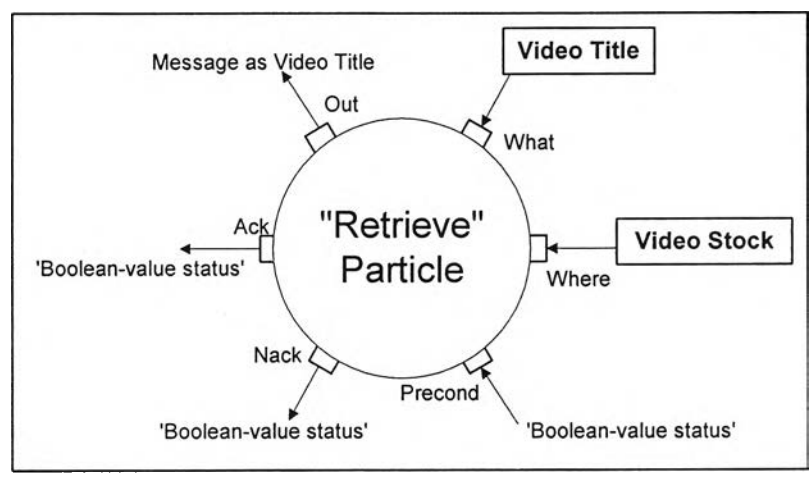
สำหรับเส้นเชื่อมแบบข้อความ จะเชื่อมต่อระหว่างโหนดข้อมูลกับช่องการสื่อสาร "*What*" หรือ "*Where*" หรือ "*Out*" หรือ ระหว่างช่องการสื่อสาร "*What*" และ "*Out*" ส่วนเส้นเชื่อมแบบสถานะ จะเชื่อมต่อระหว่างช่องการสื่อสาร "*Ack*" หรือ "*Nack*" กับ "*Precond*" ดังรูปที่ 2.4




รูปที่ 2.4 แสดงเส้นเชื่อมแบบ "ข้อความ" และ "สถานะ" ระหว่างสองโหนด

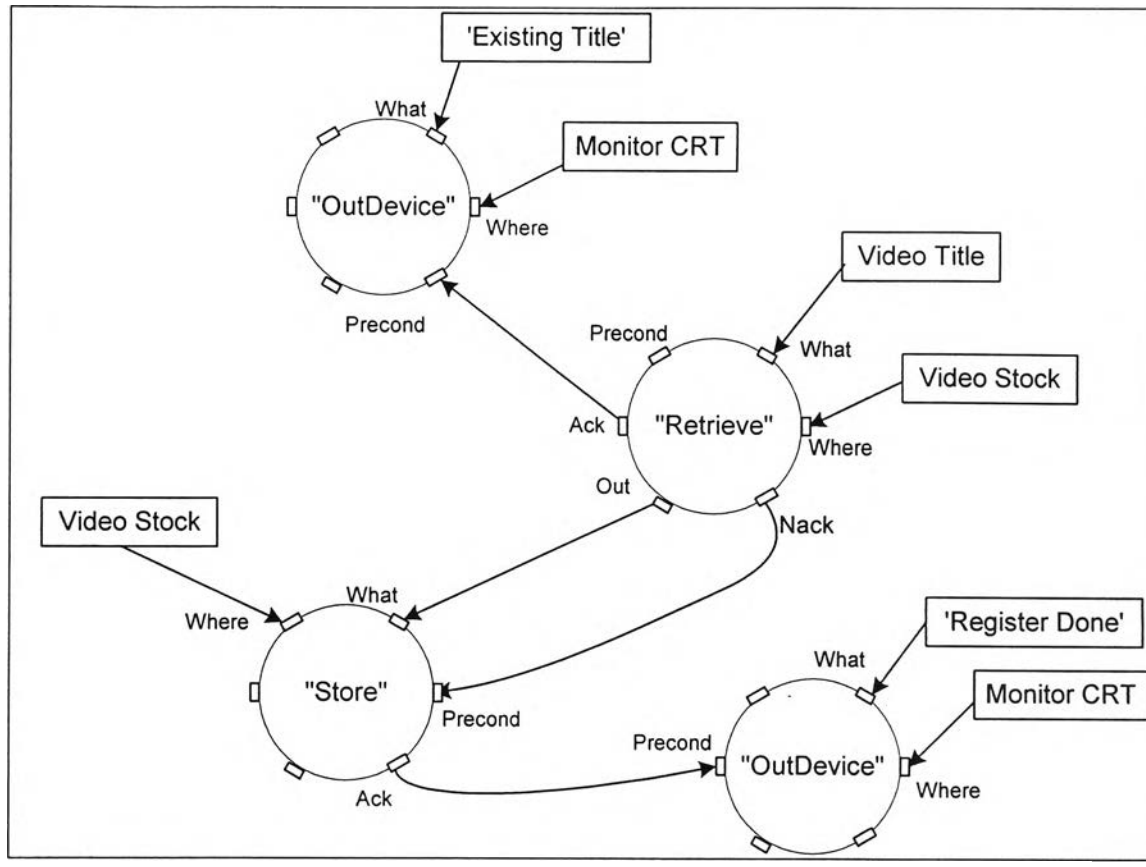
2.2.3.3 โหนดข้อมูลภายในเครือข่ายอนุภาคความต้องการ

ในทางปฏิบัติ โหนดข้อมูลสามารถเขียนอยู่ในรูปของโหนดสิ้นสุด คือโหนดที่ไม่มีทั้งข้อมูลเข้าและข้อมูลออก เพื่อเป็นแหล่งข้อมูลสำหรับป้อนค่าของข้อมูลเข้าไปยังโหนดต่อไป ดังตัวอย่างในรูป 2.5 โหนดข้อมูลที่ชื่อว่า "Video Title" เป็นแหล่งข้อมูลสำหรับโหนดการดำเนินการที่ชื่อว่า "Retrieve" โดยโหนด "Retrieve" คาดหวังว่าค่าจาก "Video Title" ที่เข้ามาจะสามารถทำการค้นหาหรือช่วยเหลือเพื่อให้ทราบว่าชื่อวิดีโอที่ต้องการทราบมีอยู่ในฐานข้อมูลแล้วหรือไม่




รูปที่ 2.5 ตัวอย่างของโหนดการดำเนินการและโหนดข้อมูลสำหรับเครือข่ายอนุภาคความต้องการ

สำหรับตัวอย่างแผนภาพเครือข่ายอนุภาคความต้องการดังรูปที่ 2.6 ได้ยกตัวอย่างการทำงานของระบบลงทะเบียนวิดีโอเทปใหม่สำหรับร้านเช่าวีดีโอ




รูปที่ 2.6 ตัวอย่างเครือข่ายอนุภาคข้อกำหนดความต้องการสำหรับระบบการลงทะเบียนวีดีโอเทปใหม่