อินเทลลิเจนท์เอเจนท์

Intelligent Agents

ปกรณ์พงศ์ โพธิพฤกษ์[1] ดร.ภัทรชัย ลลิตโรจน์วงศ์[2]

บทคัดย่อ

อินเทลลิเจนท์เอเจนท์ทำหน้าที่เป็นตัวแทนให้ผู้ใช้ได้ใช้ประโยชน์จากเครื่องมือเพื่อช่วยงานด้านใดด้านหนึ่งอย่างมีประสิทธิภาพ โดยสามารถทำงานดังกล่าวได้อย่างอัตโนมัติและตรงตามจุดประสงค์โดยที่ไม่รบกวนผู้ใช้มากเกินไป อินเทลลิเจนท์เอเจนท์มีรูปแบบการแสดงความรู้และสถาปัตยกรรมหลายแบบ แต่ที่นิยมใช้คือแบบจำลอง BDI และ Hybrid Architecture ตามลำดับ ในการพัฒนาเอเจนท์หนึ่งๆ ต้องคำนึงถึงการแสดงความรู้ สถาปัตยกรรม การทำงานร่วมกันระหว่างเอเจนท์และภาษาที่ใช้ด้วยเสมอ โดยภาษาที่นิยมใช้พัฒนาเอเจนท์ได้แก่ JAVA และ KQML ปัจจุบันเอเจนท์ได้รับความสนใจอย่างมากทั้งในเชิงวิชาการและปฏิบัติแต่อย่างไรก็ตาม เทคโนโลยีเอเจนท์ยังขาดการกำหนดมาตรฐานอยู่อีกมากซึ่งทำให้การพัฒนาเอเจนท์นั้นถูกจำกัดอยู่ที่ระดับหนึ่งเท่านั้น

Abstract

An *intelligent agent* is a representative of any users who want the powerful utilities for any specific tasks. It can automatically, goal-directed, do the jobs without interrupting the users too much. The intelligent agent can have various types of knowledge representations and architectures, but the most popular ones are an *BDI Model* and *hybrid agent architecture* respectively. To develop an agent, one must always think of agent's knowledge representation, architecture, collaborative work, and language. The most popular languages for developing an agent are JAVA and KQML. At present, agent technology is increasingly attractive in both technical and practical contexts. However, it is lacking in more standardization, and this causes agent development to be limited at a level.

1. บทนำ

เอเจนท์เป็นตัวแทนที่ทำหน้าที่ด้านหนึ่งโดยเฉพาะ เพื่อทำงานให้สำเร็จตามวัตถุประสงค์ โดยจะมีความสามารถในการรับรู้สภาพแวดล้อม สามารถนำข้อมูลดังกล่าวมาใช้ในการตัดสินใจเพื่อปฏิบัติงานได้โดยที่รบกวนผู้ใช้น้อยที่สุด อีกทั้งยังสามารถเพิ่มฐานความรู้ได้ด้วยตนเอง (ซึ่งต่างจากระบบผู้เชี่ยวชาญ) และมีความริเริ่ม (Initiative) ในการปฏิบัติงาน

การศึกษาเอเจนท์มี 2 แนวทาง ได้แก่ ระดับจุลภาค (Micro-issues) และระดับมหภาค (Macro-issues) โดยจะเป็นการศึกษาถึงสถาปัตยกรรมของเอเจนท์ และระบบปัญญาประดิษฐ์แบบกระจาย (Distributed Artificial Intelligence) ตามลำดับ ตัวอย่างของการศึกษาระดับมหภาค เช่น การศึกษาเกี่ยวกับ Mobile Agent และการศึกษาภาษาที่ใช้ในการติดต่อกันระหว่างเอเจนท์ เป็นต้น อย่างไรก็ตาม บทความนี้ได้เน้นศึกษาเอเจนท์ในระดับจุลภาคเป็นสำคัญ

เอเจนท์มักมีลักษณะเป็นแบบ Intentional System ซึ่งเป็นหลักการที่เกี่ยวกับพฤติกรรม (Behavior) ของมนุษย์ ที่เกิดจากการนำเอาทัศนคติ (Attitude) ประเภทต่างๆ มาใช้ในการตัด

---

[1] นักศึกษาปริญญาโทคณะเทคโนโลยีสารสนเทศ สจล.

[2] อาจารย์คณะเทคโนโลยีสารสนเทศ สจล.

สินใจที่จะแสดงพฤติกรรมใดพฤติกรรมหนึ่งออกมา โดยนิยมใช้การแสดงความรู้แบบ Possible World Semantics นอกจากนี้ในปัจจุบันได้มีการประยุกต์ใช้เอเจนท์หลายระบบในหลากหลายรูปแบบ แต่โดยสรุปแล้วเราสามารถแบ่งสถาปัตยกรรมของเอเจนท์ได้เป็น 3 ประเภท คือ Deliberate, Reactive และ Hybrid Architecture

ในการพัฒนาเอเจนท์ นอกจากต้องคำนึงถึงการแสดงความรู้และสถาปัตยกรรมแล้ว ยังต้องคำนึงถึงการร่วมมือกันระหว่างเอเจนท์หลายๆ ราย และภาษาที่ใช้ ทั้งภาษาที่ใช้เขียนตัวเอเจนท์เอง และภาษาที่ใช้ในการสื่อสารระหว่างเอเจนท์ด้วย โดยภาษาที่นิยมใช้เขียนเอเจนท์และใช้ในการสื่อสารได้แก่ Java และ KQML ตามลำดับ

2. นิยาม

โดยทั่วไปแล้ว เอเจนท์ (Agent) จะหมายถึงตัวแทนซึ่งทำหน้าที่เฉพาะในด้านใดด้านหนึ่ง นิยามของคำนี้ไม่ได้ถูกบัญญัติไว้โดยเฉพาะ อย่างไรก็ตาม นิยามของเอเจนท์ที่ใช้กันมากได้แก่ ระบบคอมพิวเตอร์ที่สามารถรับรู้สภาพแวดล้อมและสามารถปฏิบัติงานเพื่อบรรลุวัตถุประสงค์ที่กำหนดได้โดยอัตโนมัติ โดยที่รบกวนผู้ใช้น้อยที่สุด [5: pp. 3] ดังนั้นอาจกล่าวได้ว่าคุณสมบัติที่สำคัญของเอเจนท์คือ การปกครองตนเอง (Autonomy) นั่นเอง กล่าวคือเอเจนท์ต้องสามารถปฏิบัติงานได้โดยที่รบกวนผู้ใช้น้อยที่สุด และสามารถควบคุมการกระทำ (Actions) และสภาวะ (States) ของตนเองได้

อย่างไรก็ตาม เอเจนท์ที่มีความฉลาดนั้นจะมีนิยามที่แตกต่างออกไป โดยเอเจนท์ที่มีความยืดหยุ่น (Flexibility) เท่านั้นจึงจะจัดเป็นอินเทลลิเจนท์เอเจนท์ โดยความยืดหยุ่นนั้นประกอบไปด้วยสาระสำคัญดังนี้ [8]

- การโต้ตอบกับสภาพแวดล้อม (Reactivity หรือ Responsiveness) กล่าวคือ เอเจนท์ต้องสามารถรับรู้สภาพแวดล้อมของตนเองได้ (เช่น อินเตอร์เน็ต เป็นต้น) และตอบโต้ (response) ได้ในระยะเวลาที่รวดเร็วและเหมาะสม

- ความคิดริเริ่ม (Proactiveness) เอเจนท์ต้องสามารถปฏิบัติงานโดยเสนอความคิดริเริ่มในโอกาสที่เหมาะสมได้

- ความสามารถในการสังคม (Social Ability) เอเจนท์แต่ละตัวจะต้องสามารถสื่อสารกันได้ เพื่อแลกเปลี่ยนความรู้และ/หรือข้อมูลซึ่งกันและกัน

ดังนั้นอินเทลลิเจนท์เอเจนท์จึงหมายถึงระบบคอมพิวเตอร์ที่สามารถรับรู้และโต้ตอบกับสภาพแวดล้อมของตน สามารถปฏิบัติงานเพื่อบรรลุวัตถุประสงค์ที่กำหนดได้โดยอัตโนมัติ พร้อมนำเสนอแนวทางใหม่ๆ ในการทำงาน ณ เวลาที่เหมาะสม และสามารถติดต่อสื่อสารซึ่งกันและกันได้นั่นเอง

ตัวอย่างของเอเจนท์ที่ไม่ใช่อินเทลลิเจนท์เอเจนท์ ได้แก่ ระบบควบคุมกระบวนการทำงานต่างๆ (Process Control System) และซอฟต์แวร์ฝังตัวในหน่วยความจำ (Software Daemons) ซึ่งทั้งสองสามารถรับรู้สภาวะของตนเอง และโต้ตอบได้ แต่จะโต้ตอบในรูปแบบที่ตายตัวเสมอ ดังนั้น เอเจนท์ในลักษณะนี้จึงเหมือนกับผู้ช่วยในการปฏิบัติงานที่มีลักษณะเป็นงาน Routine นั่นเอง

3. การแสดงความรู้ในอินเทลลิเจนท์เอเจนท์

หลักการในการแสดงความรู้ และการทำ Reasoning ที่ใช้ในเอเจนท์นั้นมีหลายวิธีด้วยกัน เช่น Finite State Machine เป็นต้น แต่หลักการที่ได้รับความนิยมที่สุดในการนำมาประยุกต์ใช้กับอินเทลลิเจนท์เอเจนท์ก็คือ Intentional System [17]

เอเจนท์ที่มีลักษณะเป็นแบบ Intentional System จะจำลองพฤติกรรมของมนุษย์ โดยมีหลักการที่สำคัญคือ การตัดสินใจที่จะกระทำสิ่งหนึ่งสิ่งใดของมนุษย์ซึ่งเกิดจากการนำเอาทัศนคติประเภทต่างๆ มาประมวลผลร่วมกัน และแสดงออกมาในรูปพฤติกรรมแบบต่างๆ

Intentional System นั้นประกอบด้วยทัศนคติที่สำคัญสองประเภท [16] ได้แก่

- Information Attitude ได้แก่ ข้อมูลต่างๆ ที่เอเจนท์มีอยู่ เช่น ความเชื่อ (Belief) และความรู้ (Knowledge) เป็นต้น

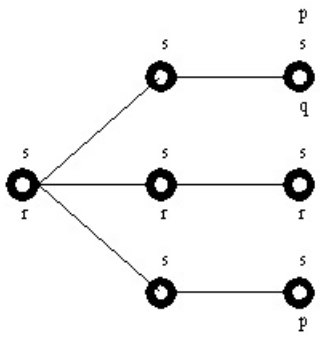
- Pro-attitude ได้แก่ ทัศนคติที่กำหนดแนวทางการกระทำของเอเจนท์ เช่น ความปรารถนา (Desire) และเจตนา (Intention) เป็นต้น

ตามหลักการแล้ว เอเจนท์ต้องมีทัศนคติทั้งสองประเภทอย่างน้อยประเภทละหนึ่งแบบ แต่โดยทั่วไปแล้วเอเจนท์มักจะใช้ระบบที่มีทัศนคติ 3 แบบ ได้แก่ ความเชื่อ ความปรารถนา และเจตนา ซึ่งระบบดังกล่าวจะเรียกว่า แบบจำลอง BDI (จัดเป็นสถาปัตยกรรมการทำ Reasoning ที่สามารถนำไปใช้จริงได้ และเป็นชนิดที่สำคัญที่สุด [7]) โดยความเชื่อแสดงข้อมูลที่เอเจนท์สามารถรวบรวมมาจากสภาพแวดล้อมของตนได้ ในขณะที่ความปรารถนาแสดงทางเลือกที่เอเจนท์สามารถเลือกปฏิบัติได้ และเจตนาแสดงทางเลือกที่เอเจนท์มีความตั้งใจและผูกมัดตัวเองที่จะปฏิบัติทางเลือกนั้นๆ

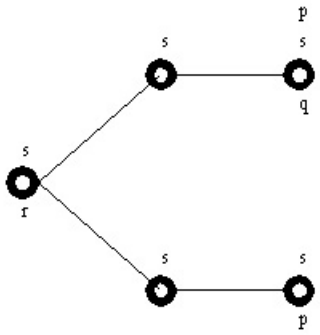
การใช้ First Order Logic แสดงความรู้ของระบบ Intentional System นั้น ไม่เหมาะสม เนื่องจากปัญหาทางความหมาย (Semantic) เป็นสำคัญ [16] กล่าวคือการที่ Predicate ประเภททัศนคติต่างๆ ไม่สามารถใช้กฎการอ้างอิงแบบ Proposition Calculus ได้ (Referentially opaque) เช่น ถ้า John เชื่อว่าพ่อของ Jupiter คือ Cronos [Bel(John, Father(Cronos, Jupiter))] และในความเป็นจริง Jupiter ก็คือ Zeus [Jupiter = Zeus] แล้ว มิอาจสรุปได้ว่า John เชื่อว่าพ่อของ Zeus ก็คือ Cronos [Bel(John, Father(Cronos, Zeus))]

ดังนั้น จึงต้องนำวิธีการแสดงความรู้แบบอื่นมาใช้กับ Predicate ประเภทนี้ ที่นิยมกันมากที่สุดก็คือ วิธี Possible Worlds Semantics โดยมีการสร้าง Modal Logic และ Axiom ต่างๆ ขึ้นมาใช้เอง

Rao and Georgeff [13] ได้ใช้ Temporal modalities เข้ามาใช้กับวิธี Possible Worlds Semantics ซึ่งจะทำให้โลก (World) มีลักษณะเป็นกิ่งไม้ ดังรูป



รูปที่ 1. โลกแห่งความเชื่อ



รูปที่ 2. โลกแห่งความปรารถนา

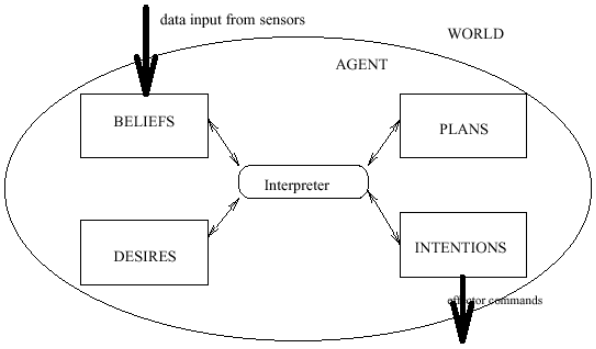
รูปที่ 1 และรูปที่ 2 แสดงโลกแห่งความเชื่อ (Belief-accessible world) และโลกแห่งความปรารถนา (Desire/Goal-accessible world) ตามลำดับ ซึ่งเอเจนท์หนึ่งๆ จะมีในเวลาต่างๆ กัน ด้านซ้ายมือจะเป็นความเชื่อ/ความปรารถนาที่เกิดก่อน โดยสรุปแล้ว วิธีนี้จะมีโลกอยู่ 3 ประเภทได้แก่ โลกแห่งความเชื่อ โลกแห่งความปรารถนา และโลกแห่งเจตนา (Intention-accessible world) โดยโลกแห่งเจตนาเป็นเซตย่อยของโลกแห่งความปรารถนา และโลกแห่งความปรารถนาเป็นเซตย่อยของโลกแห่งความเชื่ออีกทีหนึ่งนั่นเอง สำหรับตัวอย่างของการกำหนด Modal Logic และ Axiom ต่างๆ สามารถหาได้ใน [13 และ 17]

ตัวอย่างของสถาปัตยกรรมที่ใช้แบบจำลอง BDI ในการพัฒนาได้แก่ PRS (Procedural Reasoning System)

PRS นั้นเป็นสถาปัตยกรรมที่นำแบบจำลอง BDI เข้ามาใช้ โดยมีส่วนของประเภทข้อมูลอื่นที่เสริมเข้ามาคือ แผน (Plan) ดังรูปที่ 3 โดยแผนคือชุดของการกระทำต่างๆ ที่เอเจนท์สามารถปฏิบัติเพื่อให้บรรลุเจตนาแต่ละแบบได้ แผนที่เป็นไปได้ของเอเจนท์ซึ่งมีลักษณะเป็น Procedural Knowledge (Know-how) จะถูกจัดเก็บอยู่ใน Plan Library

แผนจะประกอบไปด้วยองค์ประกอบ 2 ส่วนเสมอ ได้แก่

- Body (หรือ Program) เป็นชุดของการกระทำหนึ่งๆ แผน 1 แผนจะประกอบไปด้วยชุดของการกระทำเพียง 1 ชุดเท่านั้น



รูปที่ 3. โครงสร้างแบบ PRS

- Descriptor ระบุสภาพแวดล้อมที่แผนดังกล่าวสามารถนำไปใช้ได้ (Pre-condition) และระบุเจตนาที่สามารถใช้แผนนี้เพื่อให้บรรลุเจตนาดังกล่าวได้ (Post-condition)

เอเจนท์แบบ PRS นี้จะใช้ Interpreter (ซึ่งเป็นเสมือนหน่วยประมวลผลกลางของระบบ) ในการควบคุมข้อมูลของเอเจนท์ทั้งสี่แบบ เช่น ทำหน้าที่ปรับปรุงแก้ไขความเชื่อของเอเจนท์เมื่อได้รับข้อมูลจากโลกของมันเข้ามาใหม่ สร้างความต้องการของเอเจนท์ขึ้นมาจากความเชื่อที่เพิ่งได้รับเข้ามา เลือกความต้องการที่เหมาะสมขึ้นมาเป็นความตั้งใจของเอเจนท์ และเลือกชุดของการกระทำที่เหมาะสมกับความตั้งใจดังกล่าวเพื่อนำไปปฏิบัติ เป็นต้น

4. สถาปัตยกรรมของเอเจนท์

แนวทางในการสร้างเอเจนท์ขึ้นมานั้นมีหลายวิธีด้วยกัน แต่แนวทางหลักๆ ที่นิยมใช้กันทั้งในอดีตและปัจจุบัน ได้แก่ Deliberate Architecture, Reactive Architecture และ Hybrid Architecture [17]

4.1 Deliberate Architecture

Deliberate Architecture เป็นวิธีการในการสร้างเอเจนท์แบบดั้งเดิมที่สุด โดยมองเอเจนท์เป็นระบบฐานความรู้ตัวหนึ่ง ดังนั้น จึงใช้หลักการแสดงความรู้และการ Reasoning แบบ Symbolic ซึ่งอาจกล่าวได้ว่า Deliberate Agent เป็น Symbolic Architecture ชนิดหนึ่งนั่นเอง

อย่างไรก็ตาม ปัญหาสำคัญที่จะต้องคำนึงในการนำ Deliberate Architecture มาใช้ในการสร้างเอเจนท์ก็คือ การแปลงความรู้ให้อยู่ในรูปของ Symbolic Structure และปัญหาของการแทนความรู้และการให้เหตุผล (Reasoning)

4.2 Reactive Architecture

Reactive Architecture เป็นวิธีการในการพัฒนาเอเจนท์โดยไม่ใช้ Symbolic Architecture เลย แต่หันไปใช้วิธีการอื่นแทน เพื่อเน้นการโต้ตอบกับสภาพแวดล้อมที่รวดเร็วกว่าแบบข้างต้น วิธีที่มีชื่อเสียงมากที่สุดได้แก่ Subsumption Architecture ของ Brooks [2] เขาได้แสดงให้ถึงข้อเสียของการแสดงความรู้ในงานทางปัญญาประดิษฐ์ว่า ต้องมีการจำกัดขอบเขตเฉพาะงานที่เกี่ยวข้องเท่านั้น และได้เสนอแนวทางในการสร้างความฉลาดให้แก่เอเจนท์โดยไม่จำเป็นต้องใช้การแสดงความรู้ โดยนำหลักการของ Finite State Machine (FSM) เข้ามาใช้แทนการใช้สัญลักษณ์ โดยการตัดสินใจที่จะกระทำ (Action) สิ่งใดๆ นั้น จะขึ้นอยู่กับการโต้ตอบกันของพฤติกรรมภายในตัวเอเจนท์เอง ซึ่งก็คือการโต้ตอบกันระหว่างแต่ละ FSM นั่นเอง

เอเจนท์ที่ใช้ Subsumption Architecture นั้นจะประกอบไปด้วยชั้นต่างๆ จำนวนหนึ่ง ในตัวอย่างของ Brooks ได้ทำการทดลองโดยสร้างหุ่นยนต์ขึ้นมา 4 แบบ เป็นหุ่นยนต์ที่จะต้องสามารถเคลื่อนที่ไปยังจุดหมายโดยหลบเลี่ยงสิ่งกีดขวางต่างๆ ได้ การทำงานของหุ่นยนต์เป็นลักษณะของเอเจนท์แบบ Subsumption Architecture ที่แบ่งการทำงานหลักๆ ออกเป็น 3 ชั้น ชั้นล่างสุดมีหน้าที่ในการหลบเลี่ยงสิ่งกีดขวาง ชั้นกลางมีหน้าที่ควบคุมการเคลื่อนที่ และชั้นบนสุดมีหน้าที่ในการควบคุมให้หุ่นยนต์เคลื่อนที่ไปถึงจุดหมายปลายทางที่ต้องการ แต่ละชั้นจะประกอบไปด้วย FSM จำนวนมาก และแต่ละ FSM จะทำหน้าที่ในย่อยอีกทีหนึ่ง เช่น Sonar FSM (ในชั้นล่างสุด) ทำหน้าที่ควบคุมเครื่องโซนาร์ในการตรวจจับวัตถุและแปลผลที่ได้ให้อยู่ในรูปพิกัดเชิงขั้ว และ Turn FSM (ในชั้นล่างสุด) มีหน้าที่เปลี่ยนทิศทางการเคลื่อนที่ของหุ่นไปจากทิศที่พบว่ามีสิ่งกีดขวางอยู่ Wander FSM (ในชั้นกลาง) มีหน้าที่ควบคุมให้เอเจนท์เคลื่อนที่ไป และ Pathplan FSM

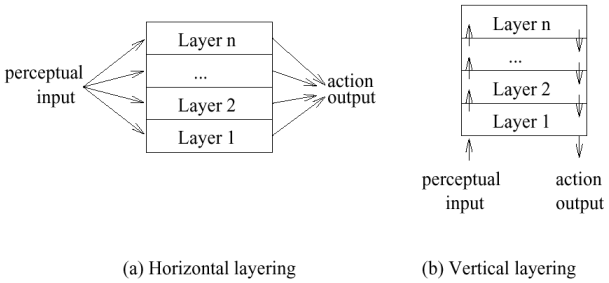
(ในชั้นบนสุด) ที่มีหน้าที่ในการคำนวณหาทิศทางที่ควรจะไปเพื่อให้หุ่นสามารถไปถึงจุดหมายได้ เป็นต้น FSM ในชั้นที่สูงกว่าสามารถที่จะควบคุม FSM ในชั้นที่ต่ำกว่าให้ทำงานตามที่ตนเองต้องการได้ โดยอาศัยการควบคุมข้อมูลที่เข้าออก FSM ในชั้นที่ต่ำกว่าให้เป็นไปตามที่ตนเองต้องการ โดยใช้กระบวนการ Suppression (ควบคุมข้อมูลที่เข้าสู่ FSM) และ Inhibition (ควบคุมข้อมูลที่ออกจาก FSM)

โดยทฤษฎีแล้ว FSM ในแต่ละชั้นนั้นจะสามารถติดต่อกับอุปกรณ์ที่รับรู้สภาพแวดล้อมและแสดงการกระทำได้อย่างเป็นอิสระจาก FSM ตัวอื่น แต่อย่างไรก็ตาม ไม่ใช่ทุก FSM ที่จะสามารถรับรู้สภาพและแสดงการกระทำได้ ขึ้นอยู่กับหน้าที่ที่ FSM นั้นๆ ได้รับการออกแบบมาให้ปฏิบัติ

อย่างไรก็ตาม สถาปัตยกรรมแบบ Reactive นี้มีข้อเสียหลายประการ เช่น การออกแบบให้เป็นแบบเรียนรู้ด้วยตนเองจะทำได้ยาก หลักการที่เอเจนท์ประเภทนี้ใช้ตัดสินใจทำงานใดๆ นั้นไม่สามารถทำความเข้าใจได้ และกรณีที่มีโครงสร้างประกอบไปด้วยชั้นจำนวนมาก จะทำให้ความสัมพันธ์ระหว่างพฤติกรรมภายในตัวเอเจนท์นั้น มีความซับซ้อนมากจนอาจจะไม่สามารถนำมาประยุกต์ใช้ได้จริง [7]

4.3 Hybrid Architecture

เป็นวิธีการที่รวมวิธีการทั้งแบบ Deliberate และ Reactive เข้าด้วยกัน สถาปัตยกรรมแบบนี้มักประกอบไปด้วยชั้นของซอฟต์แวร์หลายๆ ชั้นซ้อนกันอยู่ แต่ละชั้นทำหน้าที่ในระดับของ Abstraction ที่แตกต่างกัน (คล้าย Subsumption Architecture) โดยชั้นบนสุดจะมีความเป็น Abstraction มากที่สุด ขณะที่ชั้นล่างสุดจะมีความเป็น Abstraction น้อยที่สุด



รูปที่ 4. โครงสร้างของเอเจนท์ที่มีการแบ่งเป็นชั้นๆ

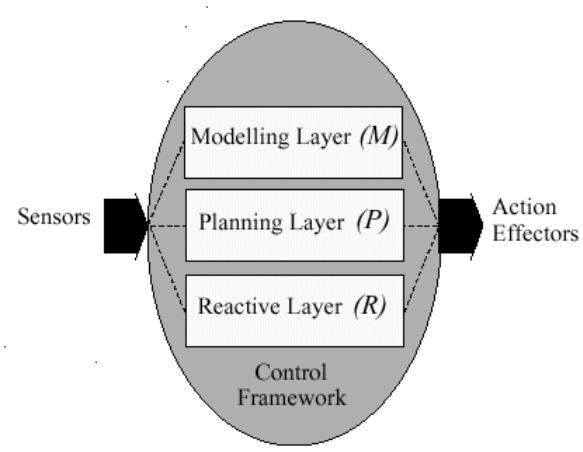
ในแต่ละชั้นจะมีวิธีการในการแสดงความรู้และการทำ Reasoning ต่างกัน โดยในส่วนบนของสถาปัตยกรรมประเภทนี้จะใช้การแสดงความรู้และการ Reasoning แบบ Symbolic หรือที่มักเรียกกันว่าการ Deliberate และในชั้นล่างจะใช้วิธีการแสดงความรู้แบบ Non-symbolic เช่น FSM เป็นต้น

การวางเป็นชั้นมี 2 แบบดังรูปที่ 4 คือ

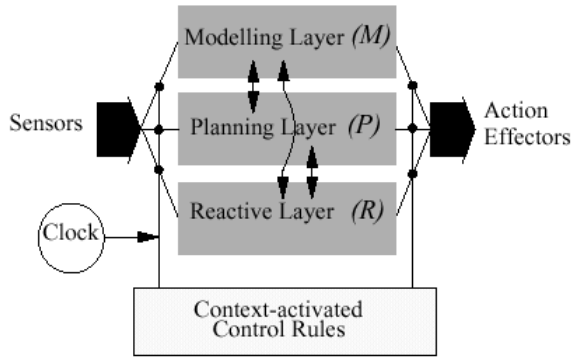
1) การวางเป็นชั้นในแนวนอน (Horizontal Layering) เป็นแบบที่ทุกชั้นสามารถรับรู้ข้อมูลจากสภาพแวดล้อมและแสดงการกระทำได้ สำหรับการรับรู้นั้นมักเป็นลักษณะของการ Synchronization กล่าวคือ ทุกชั้นจะได้รับทราบสภาพแวดล้อมเหมือนกันทั้งหมด แต่สำหรับการแสดงการกระทำนั้นแบ่งเป็น 2 แบบ ได้แก่ แบบที่แต่ละชั้นแสดงการกระทำเป็นอิสระต่อกัน และแบบที่ใช้ตัวกลาง เพื่อพิจารณาว่าควรจะแสดงการกระทำที่มาจากชั้นใดชั้นหนึ่ง ตัวอย่างเช่น TouringMachines โดย Ferguson [4] (รูปที่ 5)

2) การซ้อนกันในแนวตั้ง (Vertical Layering) เป็นแบบที่มีเพียงชั้นเดียวเท่านั้นที่สามารถรับรู้ข้อมูลและแสดงการกระทำ โดยมากมักเป็นชั้นที่อยู่ล่างสุดของสถาปัตยกรรม เนื่องจากเป็นชั้นที่มีความเป็น Abstraction น้อยที่สุด กล่าวคืออยู่ติดกับโลกของเอเจนท์นั้นๆ มากที่สุดนั่นเอง

อย่างไรก็ตาม โครงสร้างแบบ Hybrid Architecture นั้นนิยมมีเพียงแค่ 3 ชั้นเท่านั้น ได้แก่



รูปที่ 5. สถาปัตยกรรม TouringMachines



รูปที่ 6. การต่อรองระหว่างแต่ละชั้นของสถาปัตยกรรม TouringMachines

1) Reactive Layer (ชั้นต่ำสุด) ทำหน้าที่ตัดสินใจขั้นพื้นฐานเกี่ยวกับข้อมูลที่รับเข้ามา (Raw Input) เทคนิคการแสดงความรู้และการทำ Reasoning ที่ใช้กับองค์ประกอบที่อยู่ในชั้นนี้จะต้องทำงานได้อย่างรวดเร็วที่สุด เพราะว่างานที่ชั้นนี้รับผิดชอบเป็นงานเชิงปฏิบัติเป็นส่วนใหญ่ เช่น ชั้น Reactive Layer ของ TouringMachines จะใช้การแสดงความรู้และการทำ Reasoning แบบ Rule-based เพื่อหลบหลีกสิ่งกีดขวางต่างๆ และควบคุมการเดินทางให้อยู่ในช่องทางที่กำหนด ไม่เฉไปทางใดทางหนึ่ง

2) Knowledge Layer (ชั้นกลาง) ทำหน้าที่เกี่ยวกับความรู้ที่เอเจนท์มีอยู่ โดยจะทำงานในระดับของ Abstraction ที่สูงกว่า Reactive Layer ตัวอย่างเช่น ชั้น Planning Layer ของ TouringMachines ทำหน้าที่ในการวางแผนการเดินทางของตัวเอเจนท์เอง เป็นต้น องค์ประกอบในชั้นนี้จะใช้การแทนความรู้และการทำ Reasoning แบบ Symbolic

3) Social Knowledge Layer (ชั้นสูงสุด) ทำหน้าที่เกี่ยวกับการติดต่อหาความรู้ของเอเจนท์รายอื่น ได้แก่ ความเชื่อ ความปรารถนา และเจตนานั่นเอง เช่น ชั้น Modelling Layer ของ TouringMachines โดยนอกจากหน้าที่ดังกล่าวแล้ว มันยังทำหน้าที่ในการตรวจสอบหาความขัดแย้งในเป้าหมาย (Goal Conflict) อีกด้วย ทั้งความขัดแย้งระหว่างแต่ละองค์ประกอบภายในแต่ละชั้นของตัวเอเจนท์เอง และความขัดแย้งระหว่างเอเจนท์สองตัว

4.4 สรุป

สถาปัตยกรรมที่ใช้ในการพัฒนาตัวเอเจนท์นั้นมีการแบ่งประเภทไว้ชัดเจนเป็น 3 แบบ แต่อย่างไรก็ตามเป็นเพียงการมองเฉพาะเอเจนท์แต่ละตัวเท่านั้น มิได้มองถึงการทำงานร่วมกันของเอเจนท์แต่ละตัว ซึ่งจะได้กล่าวต่อไป

5. Multi-agent System (MAS)

MAS เป็นประเด็นที่เกี่ยวข้องกับการทำงานร่วมกันของเอเจนท์ตั้งแต่สองรายขึ้นไป งานในบางด้านนั้นเอเจนท์เพียงรายเดียวไม่สามารถที่จะทำงานได้ หรือถ้าทำได้ก็อาจมีประสิทธิภาพต่ำ จึงต้องใช้กลุ่มของเอเจนท์ในการแก้ปัญหาดังกล่าว หรือในกรณีที่เอเจนท์รายเดียวสามารถทำงานได้ แต่เนื่องจากสภาพแวดล้อมของเอเจนท์จะเปลี่ยนไปหากเอเจนท์อื่นๆ กระทำสิ่งใดสิ่งหนึ่ง ดังนั้น เอเจนท์ดังกล่าวอาจจะมีการแลกเปลี่ยนข้อมูลกันเพื่อวางแผนงานล่วงหน้าได้ ซึ่งก็จัดอยู่ในประเด็นของ MAS เช่นกัน

ในอดีต MAS เป็นงานทางระบบปัญญาประดิษฐ์แบบกระจายเช่นเดียวกับ DPS (Distributed Problem Solving) แต่ในปัจจุบันงานทั้งสองประเภทมักนิยมถูกเรียกรวมว่าเป็น MAS เหมือนกัน ขอเพียงสามารถทำงานได้สำเร็จด้วยตัวเองเท่านั้น โดย Wooldridge [16] ได้ให้นิยาม MAS ไว้ว่าเป็นกลุ่มเครือข่ายหลวมๆ ของผู้แก้ปัญหาที่สามารถทำงานเองได้ ซึ่งมาทำงานร่วมกัน เพื่อแก้ปัญหาต่างๆ ที่ไม่สามารถแก้ไขด้วยตัวเอง และมีคุณสมบัติดังนี้

- เอเจนท์ (ผู้แก้ปัญหา) แต่ละรายมีข้อมูลและความสามารถที่จำกัด
- ไม่มีส่วนควบคุมกลาง
- ข้อมูลกระจายอยู่ตามส่วนต่างๆ
- การทำงานของแต่ละเอเจนท์เป็นอิสระต่อกัน

ปัญหาในเอเจนท์แบบ MAS นั้นมีอยู่มากมาย โดยปัญหาที่สำคัญๆ ได้แก่ ปัญหาในการสื่อสารระหว่างเอเจนท์ และปัญหาของการทำงานร่วมกันของเอเจนท์ (เช่น ปัญหาความขัดแย้งในเจตนาของเอเจนท์แต่ละราย และการที่ระบบเกิดการแกว่งขึ้นเนื่องจากมีการส่งต่องานไปมาไม่รู้จบ เป็นต้น) ซึ่งจะทำให้ระบบนั้นมีประสิทธิภาพต่ำกว่าที่ควรจะเป็นมาก

ปัญหาในการสื่อสารสามารถแก้โดยการจัดให้มีภาษาและโพรโตคอลมาตรฐานในการสื่อสารระหว่างเอเจนท์ขึ้น ซึ่งจะได้กล่าวในหัวข้อภาษาสำหรับเอเจนท์ต่อไป

การเลือกกลไกในการแก้ปัญหาของการทำงานร่วมกันนั้นขึ้นอยู่กับประเภทของการทำงานร่วมกันของแต่ละระบบ โดย Jennings, Sycara and Wooldridge [7] แบ่ง MAS ตามประเภทการทำงานร่วมกันออกเป็น 2 ประเภท ได้แก่

1. Cooperative MAS เป็นระบบที่เอเจนท์จะทำงานร่วมกันเป็นกลุ่ม เช่น ระบบเอเจนท์สำหรับฟุตบอล RoboCup [15] เป็นต้น MAS ประเภทนี้แบ่งได้เป็น 2 ชนิด คือ
   - Common goals-based MAS เป็นชนิดที่วัตถุประสงค์ของกลุ่มจัดเป็นวัตถุประสงค์หลักของแต่ละเอเจนท์ ดังนั้นในระบบเอเจนท์ชนิดนี้จึงใช้กลไกในการร่วมมือที่เน้นประสิทธิภาพของกลุ่มเป็นหลัก MAS ชนิดนี้เป็นชนิดที่ได้รับความนิยมในอดีต เช่น PGP (Partial Global Planning) เป็นต้น
   - Explicit teamwork model-based MAS เป็นชนิดที่มีการกำหนดแบบจำลองในการทำงานเป็นกลุ่มของเอเจนท์อย่างชัดเจน เช่น Joint Intentions Framework เป็นต้น

2. Self-interested MAS เป็นระบบเอเจนท์ที่แต่ละรายจะทำงานของตนเป็นหลัก ซึ่งจะต้องมีกลไกการเจรจาต่อรอง (Negotiation) ระหว่างเอเจนท์เพื่อแก้ไขความขัดแย้งที่เกิดขึ้นในเจตนาของแต่ละเอเจนท์และเพื่อความร่วมมือของกลุ่มนั่นเอง

ในปัจจุบันมีการนำเอเจนท์ไปใช้ในอินเตอร์เน็ตจำนวนมาก ซึ่งทำให้เกิดปัญหาที่สำคัญยิ่งอีกประเด็นคือ การค้นหาเอเจนท์ที่ต้องการติดต่อด้วย เนื่องจากอินเตอร์เน็ตเป็นแหล่งความรู้ขนาดใหญ่จึงเป็นการยากที่เอเจนท์ใดๆ จะสามารถค้นหาเอเจนท์ที่ต้องการติดต่อด้วยพบ ดังนั้นจึงเกิดเอเจนท์ที่เรียกว่า Middle agents ขึ้นเพื่อแก้ไขปัญหาดังกล่าว โดยแบ่งเป็นหลายประเภทได้แก่ Matchmaker/Yellows Pages agent (ทำหน้าที่จับคู่เอเจนท์ที่ร้องขอบริการมา กับเอเจนท์ที่สามารถให้บริการดังกล่าวได้) Blackboard agent (ทำหน้าที่รวบรวมเอเจนท์ที่ร้องขอบริการเข้ามา) และ Broker agent (ทำหน้าที่ทั้งสองประเภทที่กล่าวมา)

ปัญหาที่สำคัญอีกประการหนึ่งของการพัฒนาระบบ MAS ก็คือการปรับปรุงแบบจำลอง BDI ให้สามารถนำมาใช้กับ MAS ได้ Rao and Georgeff [12] ได้เสนอไว้ว่าเราสามารถใช้ Possible Worlds Semantics ได้เหมือนเดิม โดยมีการปรับปรุงบางส่วน และเพิ่ม Axiom บางตัวเข้าไป ส่วนรายละเอียดนั้นเกินขอบเขตของรายงานฉบับนี้ดังนั้นจึงไม่ขอนำมากล่าวในที่นี้

นอกจากนี้ยังมีเอเจนท์ประเภทหนึ่งที่สามารถจัดอยู่ในกลุ่มของ MAS ได้ คือ Mobile Agent โดยจะมีลักษณะเป็นเอเจนท์ที่สามารถเคลื่อนที่ไปยังสถานที่ (Site) อื่นซึ่งเรียกว่าการ Migration เพื่อไปใช้บริการของเอเจนท์อื่น ณ สถานที่นั้นได้ เอเจนท์ประเภทนี้จัดเป็น Software Agent (ได้แก่เอเจนท์ที่อยู่ในรูปของซอฟต์แวร์คอมพิวเตอร์) ประเภทหนึ่ง ซึ่งต่างจากอินเทลลิเจนท์เอเจนท์ที่สามารถเป็นได้ทั้ง Software Agent และ Robotic Agent (ได้แก่เอเจนท์หุ่นยนต์ประเภทต่างๆ ซึ่งอยู่ในรูปของฮาร์ดแวร์ หรือซอฟต์แวร์คอมพิวเตอร์) แต่เราสามารถจัด Mobile Agent เป็นอินเทลลิเจนท์เอเจนท์ประเภทหนึ่งได้เช่นกัน

6. ภาษาสำหรับเอเจนท์

เอเจนท์หนึ่งๆ นั้นประกอบไปด้วยภาษา 2 ประเภท ได้แก่

1. Agent language
2. ACL (Agent communication language)

6.1 Agent Language

คือระบบที่อนุญาตให้สามารถโปรแกรมฮาร์ดแวร์หรือซอฟต์แวร์คอมพิวเตอร์ในรูปแบบของโครงสร้างและทฤษฎีทางเอเจนท์ได้ [17] เช่น สามารถโปรแกรมแบบจำลอง BDI ได้ เป็นต้น

Shoham [14] ได้เสนอระบบ AOP (Agent-oriented programming) ซึ่งเป็นการโปรแกรมเอเจนท์ในรูปแบบของ BDI แบบแรกๆ โดยแบ่งองค์ประกอบของ AOP ออกเป็น 3 ส่วนได้แก่

- ระบบทาง Logical เพื่อแสดงแบบจำลอง BDI
- ภาษาโปรแกรมมิ่ง
- ตัวแปลภาษา (Compiler)

และเขาได้พัฒนาภาษา AGENT0 ขึ้นมาตามแนวทางของระบบ AOP ในส่วนแรกเขาได้ใช้ Modality 3 ตัวได้แก่ ความเชื่อ ความผูกมัด (Commitment) และ ความสามารถ (Ability) แทนการใช้ Modality แบบปกติ (BDI Modality)

ในส่วนของภาษาโปรแกรมมิ่ง จะประกอบไปด้วย 3 ส่วนได้แก่ ชุดของการกระทำที่เอเจนท์สามารถทำได้ ชุดของความเชื่อและเจตนาเริ่มต้นของเอเจนท์นั้นๆ และชุดของกฎแห่งการผูกมัด (Commitment Rules) ซึ่งเป็นหัวใจหลักของโปรแกรม

กฎแห่งการผูกมัดประกอบด้วย Message Conditions , สภาวะของทัศนคติ (Mental States/Conditions) และสิ่งที่เอเจนท์สามารถกระทำได้ภายใต้กฎดังกล่าว หลักการคือเมื่อเอเจนท์ได้รับข้อความ (มี 3 แบบ ได้แก่ Request (เพื่อกระทำ) Unrequest (เพื่อหยุดกระทำ) และ Inform (เพื่อบอกการเปลี่ยนแปลงของความเชื่อที่มีในขณะนั้น)) เข้ามาก็จะนำมาเปรียบเทียบกับ Message Condition ของกฎแต่ละข้อ และนำความเชื่อมาเปรียบเทียบกับสภาวะของทัศนคติของกฎแต่ละข้อ ถ้าตรงกับกฎข้อใดกฎดังกล่าวก็จะถูกเลือกขึ้นมาใช้ และเอเจนท์จะมีเจตนา (และผูกมัดตัวเอง) เพื่อที่จะทำการกระทำที่ระบุไว้ในกฎดังกล่าวต่อไป

Agent language อีกประเภทที่ได้รับความนิยมจนถูกนำมาใช้ทางการค้าเป็นภาษาแรกได้แก่ ภาษา Telescript ของบริษัท General Magic ภาษานี้จะแบ่งระบบเอเจนท์ออกเป็น 2 ส่วนหลัก ได้แก่ สถานที่ (Places) และตัวเอเจนท์เอง โดยสถานที่จะหมายถึงสถานที่จำลอง (Virtual Location) ที่เอเจนท์หนึ่งๆ อาศัยอยู่ในระยะเวลาใดเวลาหนึ่ง และเอเจนท์จะหมายถึงกระบวนการทางซอฟต์แวร์ (Software Processes) ที่เป็นผู้ให้และรับบริการจากเอเจนท์อื่นๆ โดยตัวโปรแกรมเอเจนท์ และ/หรือสถานะของเอเจนท์สามารถเคลื่อนที่ไปยังสถานที่อื่นได้ ดังนั้นภาษานี้จึงใช้กับ MAS ประเภท Mobile Agent นั่นเอง

ภาษา Telescript แบ่งออกองค์ประกอบออกเป็น 4 ส่วนได้แก่ Telescript Language (ทำหน้าที่เกี่ยวกับการติดต่อสื่อสารต่างๆ) Telescript Engine (ทำหน้าที่เป็น Interpreter จัดตารางการทำงานของเอเจนท์ เป็นต้น) Telescript Protocols (ทำหน้าที่เกี่ยวกับการเคลื่อนย้ายเอเจนท์ระหว่างสถานที่) และ Development-supported Tools

โดยสรุปแล้ว Agent Language จะหมายถึงภาษาที่ใช้ในการโปรแกรมตัวเอเจนท์นั่นเอง (โดยในบางครั้งอาจรวมถึงภาษาที่ใช้ในการติดต่อสื่อสารระหว่างกันด้วย) ซึ่งโดยมากจะต้องสามารถโปรแกรมแบบจำลอง BDI (หรือแบบจำลองทางทัศนคติแบบอื่นๆ ที่เท่าเทียมกัน) ได้ และในปัจจุบันในการพัฒนาเอเจนท์นั้นนิยมใช้ภาษา JAVA เป็นส่วนใหญ่ [10] นอกจากนี้ Agent Language ยังจะหมายรวมถึงโปรแกรมคอมพิวเตอร์ในรูปแบบอื่นๆ ด้วย เช่น การโปรแกรม FSM ใน Subsumption Architecture เป็นต้น

6.2 ACL

เอเจนท์แต่ละรายจะต้องมีการติดต่อสื่อสารระหว่างกัน ไม่ว่าจะเป็นการติดต่อกันภายในกลุ่มของเอเจนท์ที่ทำงานร่วมกัน หรือการแลกเปลี่ยนความรู้ระหว่างเอเจนท์คนละกลุ่มกัน ประกอบกับในปัจจุบันมีสถาปัตยกรรมเอเจนท์ต่างๆ เกิดขึ้นมากมาย ดังนั้นการติดต่อดังกล่าวจึงต้องการภาษาที่เป็นมาตรฐานเพื่อให้เอเจนท์ต่างสถาปัตยกรรมกันสามารถติดต่อสื่อสารกันได้ ซึ่งก็คือ ACL (Agent Communication Language) นั่นเอง

ACL ได้แก่ภาษาและโพรโตคอลที่ใช้สำหรับสื่อสารกันระหว่างเอเจนท์ เพื่อแลกเปลี่ยนข้อมูลและความรู้ของเอเจนท์ ความรู้ที่เอเจนท์จะแลกเปลี่ยนกันได้แก่ สภาวะของทัศนคติต่างๆ (โดยมากมักเป็น BDI States) ซึ่งควรจะอยู่ในรูปของภาษาธรรมชาติเพื่อให้สามารถติดต่อระหว่างเอเจนท์ต่างประเภทกันได้ นอกจากนี้ACL ยังได้กำหนดประเภทของข้อความ (Messages) ที่เอเจนท์สามารถแลกเปลี่ยนกันได้ ซึ่งมักจะอยู่ในรูปของความเชื่อ ความปรารถนา และเจตนาเช่นเดียวกัน หรืออาจอยู่ในรูปของทัศนคติแบบอื่นๆ ที่มีความคล้ายคลึงกันก็ได้

ในปัจจุบัน ACL ที่มีชื่อนั้นแบ่งเป็น 2 ประเภท ได้แก่ KQML และ FIPA ACL [10]

1. KQML (Knowledge Query and Manipulation Language) คือภาษาและโพรโตคอลระดับสูงเพื่อการแลกเปลี่ยนข้อมูลและความรู้โดยใช้ข้อความ โดยไม่ขึ้นกับ Syntax ของเนื้อหาที่สื่อสารกัน ไม่ขึ้นกับ Ontology ซึ่งเป็นหลักการเกี่ยวกับวัตถุ ความรู้เกี่ยวกับวัตถุนั้น และความสัมพันธ์ระหว่างความรู้ ประกอบไปด้วย คำศัพท์ ความหมายของคำศัพท์ และ Axiom เกี่ยวกับคำศัพท์นั้น [6] ของแต่ละเอเจนท์ไม่ขึ้นกับโพรโตคอลที่ใช้ในการส่งข้อมูลที่อยู่ในชั้นต่ำกว่า (เช่น TCP/IP, SMTP เป็นต้น) และไม่ขึ้นกับภาษาที่ใช้เขียนเอเจนท์ (เช่น Prolog เป็นต้น) ข้อความแบบ KQML แบ่งเป็น 3 ส่วน ได้แก่ ส่วนเนื้อหา (Content) ส่วนสื่อสาร (Communication) และส่วนข้อความ (Message)

ส่วนเนื้อหา ได้แก่ข้อมูลที่เอเจนท์ต้องการจะส่ง ซึ่งอยู่ในรูปแบบของภาษาและการแสดงความรู้ที่เอเจนท์นั้นๆ เลือกใช้ (มักไม่รวมส่วนสารบัญของข้อมูล เว้นแต่กรณีที่ต้องการหาจุดสิ้นสุดของข้อมูลดังกล่าว)

ส่วนการสื่อสาร ประกอบด้วยพารามิเตอร์ต่างๆ ที่จำเป็นสำหรับการสื่อสาร เช่น Identifier ของผู้ส่ง ของผู้รับ และของการสื่อสารแต่ละครั้ง เป็นต้น

ส่วนข้อความ ระบุ Performative (ประเภทของข้อความ เช่น เป็นคำสั่ง คำถาม หรือคำยืนยัน เป็นต้น) ที่ผู้ส่งใช้ในการสื่อสารครั้งนั้นๆ

```
(ask-one
     :sender joe
     :content (PRICE IBM ?price)
     :receiver stock-server
     :reply-with ibm-stock
     :language LPROLOG
     :ontology NYSE-TICKS)
(a)

(tell
     :sender stock-server
     :content (PRICE IBM 14)
     :receiver joe
     :in-reply-to ibm-stock
     :language LPROLOG
     :ontology NYSE-TICKS)
(b)
```

รูปที่ 7. ตัวอย่างข้อความ KQML (a. ผู้ถาม และ b. ผู้ตอบ)

รูปที่ 7 เป็นตัวอย่างของข้อความแบบ KQML แบบหนึ่ง โดยแบ่งเป็นข้อความของผู้ส่ง และข้อความตอบกลับตามลำดับ สำหรับข้อความของผู้ส่งแบ่งเป็นส่วนๆ ได้ดังนี้

- ส่วนเนื้อหา ได้แก่ :content (PRICE IBM ?price)
- ส่วนการสื่อสาร ได้แก่ :sender joe :receiver stock-server และ :reply-with ibm-stock
- ส่วนข้อความ ได้แก่ :language LPROLOG และ :ontology NYSE_TICKS และรวมถึง Performative ที่ชื่อ ask-one (บอกว่าข้อความนี้เป็นข้อความประเภทสอบถาม) ด้วย

KQML จัดเป็น ACL ที่ได้รับความนิยมอย่างสูงในปัจจุบัน มีการนำ KQML มาประยุกต์ใช้ในระบบเอเจนท์ต่างๆ จำนวนมาก อย่างไรก็ตามการประยุกต์ดังกล่าวยังขาดความเป็นมาตรฐานอยู่อีกมาก เนื่องจากแต่ละระบบพยายามที่จะปรับปรุง KQML ให้เหมาะสมกับระบบของตนเองให้มากที่สุด ดังนั้นจึงมีรูปแบบย่อยๆ ของ KQML เกิดขึ้นมากมาย เช่น TKQML [3] และ KQML-CORBA [1] เป็นต้น

2. FIPA ACL (พัฒนาโดย Foundation for Intelligent Physical Agent ซึ่งเป็นองค์กรแรกๆ ที่พยายามตั้งมาตรฐานเกี่ยวกับเอเจนท์ทั้ง Software Agent และ Robotic Agent โดยก่อตั้งขึ้นในปี 2539) มีลักษณะคล้ายคลึงกับ KQML โดยเฉพาะอย่างยิ่งในด้านของ Syntax อย่างไรก็ตามเนื่องจาก FIPA ACL ได้รับความนิยมน้อยมากจึงไม่ขอกล่าวรายละเอียดในงานฉบับนี้ โดยรายละเอียดสามารถหาได้จากเว็บไซต์ http://www.fipa.org

กล่าวโดยสรุป ในการพัฒนาเอเจนท์โดยเฉพาะอย่างยิ่งประเภท MAS นั้นต้องการ ACL อย่างน้อยแบบใดแบบหนึ่งเพื่อการติดต่อสื่อสารระหว่างเอเจนท์ด้วยกัน โดยในการเลือกว่าจะใช้แบบใดนั้นควรพิจารณาจาก

- ชุดของ API ที่ ACL ประเภทนั้นมีให้ใช้ เช่น API สำหรับสร้าง ส่ง และรับข้อความ (ACL) เป็นต้น
- บริการขั้นพื้นฐานในการติดต่อสื่อสารระหว่างเอเจนท์ เช่น การลงทะเบียน การตั้งชื่อเอเจนท์ เป็นต้น ซึ่งบริการเหล่านี้ควรจะมาจากเอเจนท์ด้วยกันนั่นเอง

- รหัสแทนข้อความแต่ละประเภท สำหรับเอเจนท์ที่ทำงานในด้านใดด้านหนึ่ง ซึ่งเขียนขึ้นมาจากภาษาหนึ่งๆ และมีสถาปัตยกรรมเป็นแบบใดแบบหนึ่ง

Labrou, Finin และ Peng [10] ได้เสนอว่าในทางปฏิบัติแล้ว ACL ควรมีบริการข้างต้นให้กับผู้พัฒนาระบบเอเจนท์อยู่แล้ว แต่ในปัจจุบันนั้นยังไม่มีบริการใดๆ ที่กล่าวเลย ดังนั้น ACL นั้นจึงควรที่จะได้รับการพัฒนาในด้านนี้ให้มากขึ้นรวมถึงควรมีการสร้างมาตรฐานที่ชัดเจนและสามารถนำไปใช้ได้จริงของ ACL ไว้ด้วย

7. สรุป

ในการพัฒนาอินเทลลิเจนท์เอเจนท์นั้นต้องคำนึงถึงวิธีการแสดงความรู้ สถาปัตยกรรมที่จะเลือกใช้สำหรับการสร้างตัวเอเจนท์นั้นๆ การทำงานร่วมกันของกลุ่มเอเจนท์ที่จะสร้างขึ้นและการทำงานร่วมกันกับเอเจนท์อื่นๆ รวมถึงต้องเลือกใช้ภาษาในการโปรแกรมเอเจนท์ที่สามารถแสดงแบบจำลอง BDI หรือแบบจำลองทางด้านทัศนคติอื่นๆ ได้ และเลือกใช้ภาษาในการสื่อสารระหว่างเอเจนท์ที่มีความเหมาะสมกับระบบที่จะพัฒนาขึ้นให้มากที่สุด

ในปัจจุบันการพัฒนาและการวิจัยทางเอเจนท์ยังขาดมาตรฐานอยู่มาก อย่างไรก็ตามได้มีองค์กรที่ตั้งขึ้นเพื่อพยายามจัดตั้งมาตรฐานเกี่ยวกับเอเจนท์ขึ้น เช่น FIPA ซึ่งจะทำให้การพัฒนาเอเจนท์มีความก้าวหน้าไปได้รวดเร็วกว่าที่เป็นอยู่

เอกสารอ้างอิง

[1] Benech, D. and Desprats, T. **A KQML-CORBA based architecture for intelligent agents communication 13 in cooperative service and network management**, *Proceedings of 1997 IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, 1997.

[2] Brooks, R. A. **Intelligence without representation**, *Artificial Intelligence*, Vol. 47, pp. 139-159, 1991.

[3] Cost, R. S., et al. **TKQML: a scripting tool for building agents**, Wooldridge, M., Singh, M., and Rao, A. (eds.); *Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages*, pp. 336-340, Berlin, Springer-Verlag, 1997.

[4] Ferguson, I. A. **Touring-machines: autonomous agents with attitudes**, *IEEE Computer*, Vol. 25, No. 5, 1992.

[5] Frankin, S. and Graesser, A. **Is it an agent, or just a program?: a taxonomy for autonomous agents**, *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Berlin, Springer-Verlag, 1996.

[6] Gruber, T. R. **A translation approach to portable Ontology specification**, *Knowledge Acquisition*, Vol. 2, pp. 199-220, 1993.

[7] Jennings, N. R., Sycara K., and Wooldridge M. J. **A roadmap of agent research and development**, *Autonomous Agents and Multi-Agent Systems*, Vol.1, pp. 275-306, 1998.

[8] Jennings, N. R. and Wooldridge, M. J. **Applications of intelligent agents**, Jennings, N. R. and Wooldridge, M. J. (eds.); *Agent Technology: Foundations, Applications, and Markets*, pp. 3-28, Berlin, Springer, 1998.

[9] Kozma, J. **Intelligent agents**, *IEEE Potentials*, April/May, pp. 16-19, 1998.

[10] Labrou, Y., Finin, T., and Peng, Y. **Agent communication language**, *IEEE Intelligent Agents*, March/April, pp. 45-52, 1999.

[11] Luger, G. F. and Stubblefield, W. A. **Artificial Intelligence: Structures and Strategies for**

Complex Problem Solving, 3rd Ed., USA, Addison Wesley Longman, 1998.

[12] Rao, A. S. and Georgeff, M. P. **Formal models and decision procedures for multi-agent systems**, *Technical Note 61*, Australian Artificial Intelligence Institute, 1995.

[13] Rao, A. S. and Georgeff, M. P. **Modeling rational agents within a BDI-architecture**, *Technical Note 14*, Australian Artificial Intelligence Institute, 1991.

[14] Shoham, Y. **Agent-oriented programming**, *Artificial Intelligence*, Vol. 60, pp. 51-92, 1993.

[15] Veloso, M., et al. **A team of robotic soccer agents collaborating in an adversarial environment**, *Proceedings of the First International Workshop on RoboCup*, Nagoya, 1997.

[16] Wooldridge, M. J. **Agent-based computing**, *Interoperable Communication Networks*, Vol. 1, No. 1, pp. 71-97, 1998.

[17] Wooldridge, M. J. and Jennings, N. R. **Intelligent agents: theory and practice**, *Knowledge Engineering Review*, Vol. 10, No. 2 ,1995.