Chương 2. CƠ SỞ LÝ THUYẾT

2.1. AGENT VÀ HỆ THỐNG ĐA AGENT

2.1.1. Định nghĩa Agent

Vì là công nghệ mới nên vẫn chưa thống nhất định nghĩa về Agent. Có rất nhiều định nghĩa khác nhau, thậm chí mâu thuẫn nhau về agent. Đây từng là đề tài tranh cãi giữa các tác giả, các nhóm tác giả, thậm chí có nhiều bài báo mà nội dung chính là so sánh và phân tích các định nghĩa agent. Nguyên nhân chính dẫn đến nhiều định nghĩa agent là các tác giả khác nhau thường có yêu cầu khác nhau về đặc điểm của các agent tuỳ theo ứng dụng cụ thể của mình.

2.1.1.1. Định nghĩa tổng quát

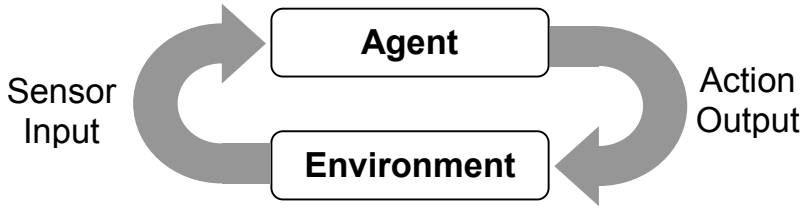
- **American Heritage Dictionary**: "Một thực thể hành động hoặc tự chủ hành động hay đại diện cho một thực thể khác".
- **Russel and Norvig**: "Một Agent là một thực thể có thể cảm nhận môi trường thông qua các cảm biến và phản ứng lại môi trường thông qua các cơ quan phản ứng kích thích".
- **Maes, Pattie:** "Các Agents tự động là các hệ thống tính toán trong môi trường thay đổi phức tạp, chúng cảm nhận và phản ứng tự động trong môi trường này, và thực thi để hoàn thành nhiệm vụ hay công việc của nó".

2.1.1.2. Định nghĩa chi tiết hơn

- **Smith, Cypher and Spohrer:** "Một Agent là một phần mềm bền bỉ thiết kế cho một mục đích chuyên biệt". "Bền bỉ" nhằm phân biệt Agent và thủ tục, Agent có những ý đồ riêng về việc làm sao để thực hiện nhiệm vụ "Chuyên biệt": một Agent không phải là một ứng dụng đa chức năng, nó nhỏ hơn.
- **Hayes-Roth:** "Các Agent thông minh liên tục thực hiện ba chức năng: nhận thức các điều kiện thay đổi của môi trường; hành động ảnh hưởng đến môi trường, lập luận để hiểu sự nhận thức, giải quyết vấn đề, nỗ lực tìm kết luận và xác định hành động".

2.1.1.3. Định nghĩa trong công nghiệp

- **IBM:** "Agent thông minh là phần mềm tiến hành các thao tác với tư cách đại diện cho một người hay một chương trình".
- Nhưng định nghĩa được nhiều nhà nghiên cứu chấp nhận là định nghĩa của Wooldridge và Jennings (1995) và được sử dụng trong [17]. "*Agent là một hệ thống máy tính được đặt trong môi trường (enviroment) nào đó và có khả năng hoạt động tự trị (autonomous) trong môi trường này để đạt được mục tiêu của nó*".



Hình 2.1: Agent cảm nhận và tác động lại môi trường

2.1.2. Các tính chất của Agent

- **Tự động (Autonomous):** Tự động được chỉ đạo từ trước chứ không chỉ đạo trực tiếp. Agent hoạt động một cách độc lập và có khả năng tự mình đưa ra quyết định. Đây là một trong những thuộc tính quan trọng để phân biệt Agent với Object.
- **Đặt trong một tình huống (Situated):** Không phải là một khái niệm ràng buộc, vì tất cả các phần mềm đều phải hoạt động trong một môi trường nào đó. Nhưng điểm khác biệt của Agent chính là *loại môi trường*. Agent được dùng trong các môi trường đầy thách thức: động, không đoán trước được và không tin cậy, ... Chúng ta sẽ tìm hiểu kỹ hơn về môi trường ở phần 2.1.7.

Ví dụ về Agent đơn giản

Máy điều nhiệt tự động có một bộ cảm biến để xác định nhiệt độ trong phòng. Bộ cảm biến này được đặt trong môi trường (ví dụ như trong phòng) và nó sản sinh ra 1 trong 2 tín hiệu: một tín hiệu xác định nhiệt độ quá thấp, một tín hiệu xác định nhiệt độ OK. Các hành động mà máy có thể thực hiện gồm "bật" và "tắt". Hành động "bật" mục đích làm tăng nhiệt độ phòng nhưng chưa chắc có hiệu quả (chẳng hạn nếu cửa phòng mở - nhiệt độ phòng không tăng lên). Bộ phận ra quyết định của máy điều nhiệt tự động này cực kỳ đơn giản, gồm 2 luật:

Quá lạnh ⟶ Bật

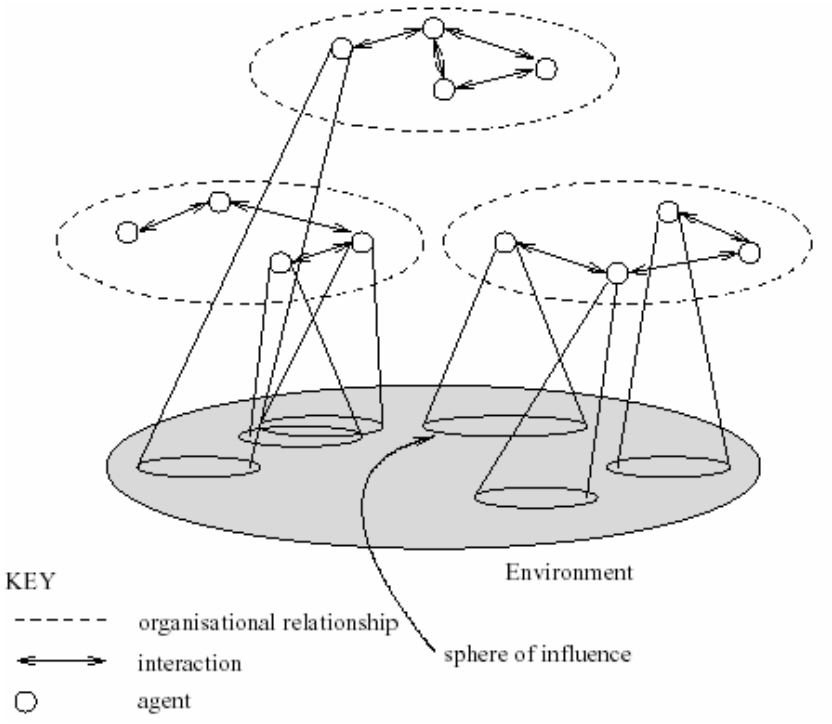
Nhiệt độ OK ⟶ Tắt

Agent trong ví dụ trên là những agent đơn giản, tầm thường, không được coi là thông minh. Những ví dụ phức tạp hơn như: hệ thống điều khiển máy bay, hệ thống điều khiển lò phản ứng hạt nhân, … là các agent thông minh.

2.1.3. Agent thông minh (Intelligent Agent)

Agent thông minh là một agent có thêm những khả sau [02]:

- *Phản ứng lại* (Reactive): Agent thường được đặt trong môi trường động, luôn thay đổi. Vì vậy nó luôn luôn quan sát môi trường và phản ứng lại kịp thời những thay đổi của môi trường.
- *Hành động chuyên nghiệp hay tính hướng đích* (Proactive): Là một thuộc tính quan trọng của agent, giúp phân biệt agent với object. Agent luôn kiên trì theo đuổi mục tiêu, mặc dù các nỗ lực có thể thất bại. Agent có khả năng tạo ra mục tiêu tại những thời điểm khác nhau, có khả năng nắm bắt các cơ hội, ... Vấn đề quan trọng trong các kiến trúc agent đó là cân bằng giữa phản ứng và hoạt động. Nếu agent thiên về phản ứng lại thì nó sẽ điều chỉnh kế hoạch của nó và dẫn đến không đạt được mục tiêu, nếu agent không phản ứng lại thích đáng thì nó sẽ phí thời gian cố gắng theo đuổi các kế hoạch không còn thích đáng hoặc thích hợp nữa.
- *Chắc chắn* (robust): Do hành động có khả năng thất bại trong môi trường luôn thay đổi, agent phải có thể khôi phục lại từ những thất bại này, nghĩa là nó phải chắc chắn.
- *Mềm dẻo* (flexible): Là một tiếp cận tự nhiên để đạt đến sự chắc chắn. Bằng cách có một loạt cách thức để đạt được mục tiêu đề ra, agent luôn có sự thay thế khi kế hoạch thất bại.
- *Xã hội* (social): Agent hầu như luôn cần phải tương tác với các agent khác (hay con người) để đạt được mục đích. Sự tương tác này thường ở mức độ cao hơn việc trao đổi thông điệp như “thông báo”, “yêu cầu”, “đồng ý”… Tương tác agent được coi giống như là các loại tương tác của con người: thương lượng, phối hợp, hợp tác và làm việc theo nhóm.



Hình 2.2: Mô hình hệ thống Multi-Agent

- *Cơ động* (Mobility): Agent có khả năng di chuyển trên mạng.
- *Trung thực* (Veracity): Agent không có ý truyền đạt thông tin sai, không làm gì ngược lại với mục tiêu đề ra.
- *Rộng lượng* (Benevolence): Các agent không có những mục tiêu xung đột, vì vậy mỗi agent sẽ cố gắng thực hiện công việc của nó.
- *Hợp lý* (Rationality): Agent sẽ hành động để đạt mục tiêu của nó, và sẽ không hành động để ngăn cản mục tiêu đạt được (thuộc tính này là cơ sở của mô hình Belief-Desire-Intention – Rao and Georgeff 1992).
- *Học hỏi/thích nghi* (Learning/Adaption): Agent luôn cải thiện sự thực thi. Là khả năng rút ra được một kết luận gì đó sau một thời gian dài quan sát từ những hành động lặp đi lặp lại nhiều lần của người sử dụng.

Cách thức hoạt động của Intelligent Agent như sau: từ tập giả thiết thu nhận từ môi trường và tập các quy tắc của Agent, Agent sẽ chọn lựa hành động phù hợp, sau khi thực hiện hành động đã chọn, môi trường sẽ thay đổi và quá trình được lặp lại.

```
Tập giả thiết F từ môi trường  E
Tập qui tắc R
While (true) do {
   F =   Thunhận (E);
   A = Lựachọnhànhđộng(F, R);
   E.hànhđộng(A);
}
```

Biểu diễn tri thức của Agent

2.1.4. Phân loại agent

- **Collaborative agents:** Nhấn mạnh khả năng tự trị và hợp tác với các agent khác.
- **Interface agents**: Nhấn mạnh khả năng tự trị và khả năng học của Agent.
- **Mobile agents**: Nhấn mạnh khả năng di chuyển trong mạng WAN, tương tác với các host và thi hành nhiệm vụ được giao.
- **Information/Internet agents**: Nhấn mạnh khả năng quản lý, vận dụng, sắp xếp các thông tin từ nhiều nguồn phân tán.
- **Reactive agents**: Agent không chứa mô hình symbolic về môi trường, chỉ hoạt động dựa trên các luật "điều kiện → hành động". Các agent này trả lời ngay lập tức những biến đổi của môi trường.
- **Hybrid agents**: Agent mà kết cấu là kết hợp của hai hay nhiều khái-niệm-triết-học trong một agent đơn lẻ.
- **Smart agents**: Nhấn mạnh tính tự trị, khả năng học và hợp tác.

2.1.5. Phân biệt Agent với các đối tượng khác

2.1.5.1. Agents và Objects

Theo quan điểm truyền thống thì giữa Object và Agent có ít nhất 3 điểm khác biệt sau:

- Agent thể hiện tính tự động cao hơn Object, cụ thể là tự Agent quyết định có thực hiện hay không một hành động khi được một Agent khác yêu cầu.

- Agent có các khả năng hành động linh hoạt (reactive, proactive, social), còn Object thì không.
- Một MAS vốn dĩ là đa luồng (Multi-thread) trong đó mỗi Agent có riêng ít nhất một luồng điều khiển.

2.1.5.2. Agents và Expert Systems

Sự khác biệt chính giữa Agent và Expert System (hệ chuyên gia) như sau:

- Agent thì được đặt trong một môi trường còn hệ chuyên gia thì tách biệt – chúng không gắn liền với bất kỳ môi trường mà nó hoạt động thông qua người sử dụng như một người trung gian.
- Những hệ chuyên gia không có khả năng: tác động lại, hướng đích.
- Hệ chuyên gia không được trang bị khả năng giao tiếp, như sự hợp tác, phối hợp và sự dàn xếp.

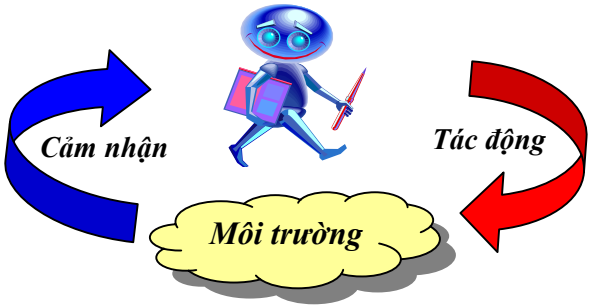
Mặc dù sự khác biệt này, nhiều hệ chuyên gia được coi như những Agent. Ví dụ hệ ARCHON.

2.1.6. Các khái niệm dùng để xây dựng agent

Để xây dựng agent, người ta dựa trên các tính chất của agent đã mô tả trong 2.1.3 như tính tự hoạt động trong môi trường thay đổi liên tục, tính chuyên nghiệp, chắc chắn, mềm dẻo, tính xã hội, …

- **Percepts và Actions**

Hai khái niệm cơ bản mô tả giao tiếp giữa agent và môi trường đó là *cảm nhận* (percepts) từ môi trường qua bộ cảm ứng (sensors) và *hành động* (actions) để gây ảnh hưởng lên môi trường qua bộ tác động (actuators) [33].



Hình 2.3: Agent cảm nhận và tác động lại môi trường

- **Goals và Events**

Mục tiêu (goal) được định nghĩa là các trạng thái mà agent cần phải hướng đến. Mục tiêu giúp cho agent tự hoạt động và hoạt động chuyên nghiệp. Khía cạnh quan trọng của hoạt động chuyên nghiệp đó là agent đeo đuổi mục tiêu bền bỉ: nếu kế hoạch đạt mục tiêu bị thất bại thì nó xem xét các kế hoạch thay thế cho đến khi nó tin rằng không thể đạt được mục tiêu hoặc mục tiêu không còn thích hợp nữa.

Biến cố (event) là sự kiện xảy ra mà agent phải phản ứng lại. Biến cố có thể dẫn đến mục tiêu mới, gây ra những thay đổi thông tin về môi trường và dẫn đến hành động thực hiện ngay lập tức.

- **Plans và Beliefs**

Lòng tin (beliefs) của agent là tri thức và thông tin của agent về môi trường. Do môi trường luôn thay đổi liên tục nên agent cần lưu giữ thông tin mà nó nhận được. Ví dụ agent tin là có đám cháy ở vị trí X vì nó vừa thấy ở đó, mặc dù hiện giờ agent không thấy.

Kế hoạch (plans) là một "thư viện các phương pháp". Ví dụ kế hoạch để đạt mục tiêu chữa cháy gồm 3 bước: xác định đường dẫn đến đám cháy, đi theo đường dẫn đến đám cháy và phun nước cho đến khi dập tắt đám cháy.

Mỗi mục tiêu có thể đạt được bằng một số kế hoạch khác nhau. Khi thực thi, agent sẽ chọn kế hoạch để đạt mục tiêu đề ra. Nếu kế hoạch được chọn thất bại, thì nó sẽ thử kế hoạch khác. Bằng cách này, agent rất mềm dẻo, vì nó có nhiều kế hoạch để đạt mục tiêu và nó rất chắc chắn vì khi kế hoạch thất bại thì không có nghĩa là mục tiêu không đạt được.

- **Messages và Protocols**

Agent tương tác với các agent khác theo nhiều dạng: trao đổi thông điệp (*message*) dựa vào các giao thức (*protocol*) định nghĩa sẵn, tạo thành nhóm làm việc hướng tới mục tiêu chung.

2.1.7. Môi trường (Enviroment)

Có thể phân chia thuộc tính của môi trường như sau [16 - lecture02]:

a. Có thể truy cập và không thể truy cập (*Accessible and Inaccessible*): Một môi trường có thể truy cập là một môi trường mà agent có thể thu được những thông tin chính xác, đầy đủ, mới nhất về môi trường đó. Môi trường thế giới thực (Internet hay môi trường vật lý) là những môi trường không thể truy cập.

b. Xác định và không xác định (*Deterministic and Non-deterministic*): Một môi trường xác định là một môi trường mà mỗi hành động của agent đều cho một kết quả duy nhất đối với môi trường. Trên thực tế, hầu hết các môi trường là không xác định.

c. Tĩnh và động (*Dynamic and Static*): Môi trường tĩnh là môi trường mà không đổi ngoại trừ được agent tác động vào. Ngược lại, môi trường động là môi trường mà thay đổi vượt qua tầm kiểm soát của agent (môi trường vật lý, Internet).

d. Rời rạc và liên tục (*Discrete and Continuous*): Môi trường rời rạc là môi trường có hữu hạn các trạng thái. Ví dụ trò chơi cờ tướng là một môi trường rời rạc (mặc dù số trạng thái rất lớn).

e. Môi trường phân đoạn và không phân đoạn (*Episodic and Nonnepisodic*): Với một môi trường phân đoạn thì kinh nghiệm của Agent được chia thành các đoạn. Mỗi đoạn bao gồm nhận thức và hành động của Agent. Chất lượng của các hành động của Agent phụ thuộc vào chính các đoạn của nó. Bởi vì dãy các đoạn không phụ thuộc vào hành động xảy ra ở đoạn trước. Môi trường phân đoạn thì đơn giản đối với Agent vì nó không cần "suy nghĩ" trước.

Ví dụ: Tính chất của một số môi trường.

Môi trường	Accessible	Deterministic	Episodic	Static	Discrete
Chơi cờ có tính giờ	Có	Có	Không	Semi	Có
Chơi cờ không tính giờ	Có	Có	Không	Có	Có
Chơi bài	Không	Không	Không	Có	Có
Cờ thỏ cáo	Có	Không	Không	Có	Không
Taxi	Không	Không	Không	Không	Không
Chẩn đoán y học	Không	Không	Không	Không	Không

2.1.8. Hệ thống Multi-Agent (MAS)

Hệ thống multi-agent nghiên cứu hành vi của tập hợp các agent làm việc cùng nhau để giải quyết một nhiệm vụ nào đó. Tương tác giữa các agent này không biết trước mà chỉ xác định trong quá trình hệ thống hoạt động. Agent trong các hệ thống có thể thuộc về những cá nhân hay tổ chức khác nhau với những mục tiêu khác nhau. Quan hệ giữa các agent chỉ là phối hợp chứ không nhất thiết là hợp tác. Các agent tự chủ riêng lẻ được trang bị thêm cơ chế liên lạc, thay đổi cơ chế suy diễn cho phép thực hiện phối hợp để tạo thành MAS.

Hệ thống multi-agent có những đặc điểm chính sau:

- Thông tin hoặc khả năng giải quyết vấn đề của từng agent là hạn chế, không đầy đủ.
- Không có sự điều khiển tập trung cho toàn hệ thống.
- Dữ liệu được phân tán trên những thành phần khác nhau của hệ thống.
- Quá trình tính toán được thực hiện không đồng bộ.

Những ưu điểm chính của MAS bao gồm:

- Cho phép giải quyết các vấn đề phức tạp đối với từng agent riêng lẻ, những bài toán đòi hỏi tài nguyên tính toán quá lớn, hoặc những vấn đề đòi hỏi phân tán để tránh tắc nghẽn và tăng tính ổn định.
- Cho phép tích hợp các hệ thống, chương trình có sẵn. Để có thể thích ứng với sự phát triển công nghệ và yêu cầu thực tế, các hệ thống có sẵn cần được nâng cấp, thay đổi, cần có khả năng tích hợp với các hệ thống mới (có thể không tương thích). Việc sửa đổi hoàn toàn các hệ thống cũ rất tốn kém hoặc thậm chí không thực hiện được. MAS cho phép tích hợp các hệ thống cũ bằng cách tạo ra các agent bao. Các agent bằng cơ chế liên lạc của mình có thể đóng vai trò giao diện, cầu nối, cho phép các hệ thống không tương thích tích hợp với nhau.
- Cho phép mô hình hóa một cách tự nhiên và trực giác nhiều vấn đề có cấu trúc giống với MAS. Ví dụ như hệ thống tra cứu kiến thức toán học có thể hình dung một cách tự nhiên như MAS, mỗi agent đại diện cho một tri thức

lĩnh vực toán khác nhau như Toán rời rạc, Toán đồ thị, Đại số tuyến tính, Hình học, Giải tích, ...

- Cho phép giải quyết các vấn đề trong đó thông tin hoặc tri thức có tính phân tán.

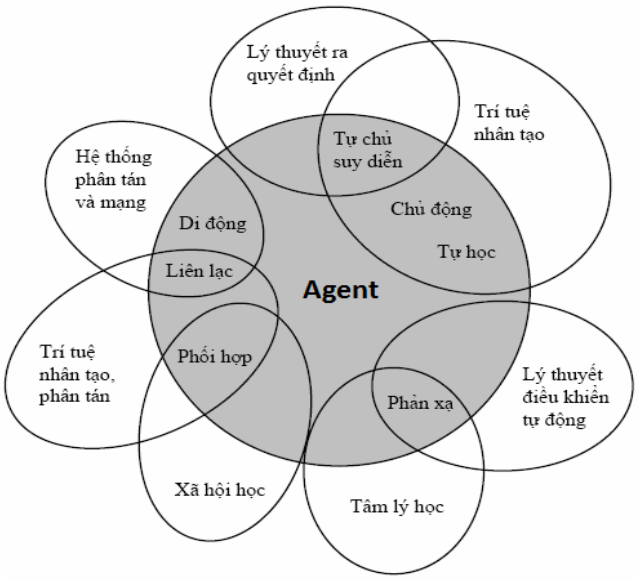
Những vấn đề liên quan đến MAS:

Mặc dù có nhiều ưu điểm, việc thiết kế và cài đặt MAS nảy sinh nhiều vấn đề cần giải quyết. Các vấn đề thường được nhắc đến và cũng là mục đích các nghiên cứu liên quan đến MAS bao gồm:

- Cách biểu diễn, phân rã bài toán, phân chia bài toán con đến các agent trong hệ thống cũng như tổng hợp kết quả thành phần.
- Tổ chức tương tác, liên lạc giữa các agent, sử dụng giao thức liên lạc và ngôn ngữ giao tiếp nào, nội dung trao đổi là gì và thời gian liên lạc ra sao.
- Trong khi từng agent ra quyết định và hành động cục bộ vẫn phải đảm bảo tính thống nhất, hiệu quả của cả hệ thống.
- Để có thể hợp tác với nhau, mỗi agent phải có thông tin và hình dung nhất định về các agent khác.
- Giải quyết các vấn đề có thể có trong quá trình hoạt động của hệ thống.
- Cân bằng giữa khối lượng tính toán cục bộ và phí tổn liên lạc giữa các agent trong hệ thống.
- Xây dựng công cụ, phương pháp thiết kế và cài đặt các MAS cho các ứng dụng thực tiễn.

2.1.9. Agent và các lĩnh vực nghiên cứu liên quan

Cũng giống như nhiều nghiên cứu và công nghệ khác, agent không phải là một công nghệ hay một hướng nghiên cứu độc lập và hoàn toàn mới mà được hình thành từ nhiều hướng nghiên cứu, nhiều công nghệ khác, đồng thời sử dụng nhiều kết quả của những nghiên cứu này. Agent cũng không phải là những ứng dụng mới, được sử dụng biệt lập mà thường được sử dụng kết hợp với những ứng dụng khác, cho phép bổ trợ, tăng khả năng, hiệu quả của những ứng dụng này. Các đặc điểm của agent được hình thành nhờ sử dụng những kỹ thuật từ nhiều khoa học khác nhau [05].



Hình 2.4: Agent và các lĩnh vực nghiên cứu liên quan

- **Trí tuệ nhân tạo:** Trí tuệ nhân tạo là lĩnh vực nghiên cứu có ảnh hưởng quan trọng nhất đến sự hình thành và phát triển của công nghệ agent. Trên thực tế, agent thông minh được xem như một nhánh nghiên cứu của hệ thống trí tuệ nhân tạo còn MAS là đối tượng nghiên cứu chủ yếu của trí tuệ nhân tạo phân tán.
- **Mạng máy tính và Internet:** Việc phát triển mạng máy tính nói chung và internet nói riêng được coi là động lực quan trọng dẫn đến sự hình thành và sử dụng công nghệ agent. Tuy nhiên MAS có nhiều đặc điểm và khả năng riêng mà hệ thống phân tán thông thường không có. Thứ nhất, khác với hệ thống phân tán thông thường, cơ chế phối hợp và đồng bộ của MAS không được xác định từ trước, quan hệ giữa các thành phần trong hệ thống chỉ được xác định trong thời gian chạy, hệ thống phân tán truyền thống không có đặc điểm này. Thứ hai, mỗi agent có thể có mục đích riêng, hoạt động vì mục đích riêng của mình. Trong khi đó các thành phần của hệ thống phân tán thông thường được thiết kế để cộng tác với nhau và có chung một mục tiêu. MAS do vậy cần có phương pháp phối hợp, đồng bộ, giải quyết mâu thuẫn riêng, dựa trên thương lượng và một số cơ chế khác.
- **Công nghệ phần mềm:** Khái niệm agent – những thực thể phần mềm với khả năng hoạt động tự chủ và tương tác – có thể rất có ích cho công nghệ phần mềm. Thay vì sử dụng đối tượng làm khái niệm chính để thực hiện trừu tượng

và môđun hóa các hệ thống phần mềm, hệ thống sử dụng agent làm khái niệm chính. Phương pháp phân tích thiết kế phần mềm trong đó agent được sử dụng như khái niệm trừu tượng chính được gọi là phương pháp phân tích thiết hướng agent. Khái niệm agent cho phép thực hiện trừu tượng hóa cao hơn so với đối tượng. Nếu MAS được cài đặt bằng một ngôn ngữ lập trình hướng đối tượng thì mỗi agent thông thường được tạo thành từ nhiều đối tượng.

2.1.10. Ưu nhược điểm của Agent và Hệ thống multi-agent

Agent và MAS là giải pháp phù hợp cho các hệ thống trong những trường hợp sau:

- Hệ thống có cấu trúc phức tạp, có thể phân tích thành những thành phần tự chủ hoặc bán tự chủ tương tác với nhau.
- Dữ liệu, thông tin, tri thức có tính phân tán và chi phí để chuyển đổi thành dạng tập trung là tương đối lớn hoặc việc chuyển đổi rất khó khăn.
- Yêu cầu sử dụng lại và tích hợp vào hệ thống những phần mềm có sẵn có thể không tương thích nhau.

Tuy nhiên, Agent và MAS không thích hợp khi ứng dụng cho các trường hợp sau:

- Trong những hệ thống thời gian thực hoặc thời gian đáp ứng có ý nghĩa quan trọng. Nói chung thời gian đáp ứng của MAS là rất khó xác định trước.
- Trong những trường hợp cần có sự điều khiển tập trung và thống nhất, cũng như yêu cầu thỏa mãn ràng buộc chung và tìm giải pháp tối ưu tổng thể. Do từng agent không có cái nhìn tổng thể về vấn đề cần giải quyết, giải pháp của MAS thường chỉ là giải pháp tối ưu cục bộ.
- Trong những trường hợp yêu cầu mức độ trách nhiệm cao đối với quyết định và hoạt động. Đòi hỏi người sử dụng có độ tin tưởng cao vào agent và dám ủy thác cho agent hoạt động tự chủ. Để có mức độ tin tưởng cần thiết, người sử dụng cần có thời gian theo dõi và làm việc với agent để đảm bảo agent không vượt quá giới hạn quyền hạn của mình.

2.1.11. Phạm vi ứng dụng của agent

- **Các ứng dụng của agent có thể được chia thành 2 nhóm chính:**

- Các hệ thống phân tán (distributed systems): Trong đó các agent trở thành các nút xử lý. Nhấn mạnh ở khía cạnh "nhiều" của các MAS.
- Phần mềm giúp đỡ cá nhân (personal software assistants): Trong đó agent đóng vai trò giúp đỡ người dùng làm việc trong một ứng dụng nào đó. Nhấn mạnh về tính riêng lẻ của agent.

- **Các lĩnh vực áp dụng của agent:**
  - Agents với các giác quan phân tán (Agents for Distributed Sensing)
  - Quản lý tiến trình thương mại và công việc (Agents for Workflow and Business Process Management)
  - Agent thu thập và quản lý thông tin (Agents for Information Retrieval and Management)
  - Agent trong lĩnh vực thương mại điện tử (Agents for Electronic Commerce)
  - Agent giao diện giữa người và máy (Agents for Human-Computer Interfaces)
  - Agent trong môi trường ảo (Agent for Virtual Environment)
  - Agent để mô phỏng xã hội (Agent for Social Simulation)

2.1.12. Các ngôn ngữ cài đặt agent

- AgentSpeak (Rao, 96)
- 3APL (Hindriks et al. 1998)
- AGENT0 (Shoham)
- GOLOG
- CONGOLOG
- Concurrent MetateM
- JACK Developer
- Ngoài ra, cũng có thể sử dụng: Java, C#, Maple, XML, …

2.2. ONTOLOGY

2.2.1. Định nghĩa

Đã có rất nhiều định nghĩa về ontology, tùy theo quan điểm của những nhà nghiên cứu là những nhà triết học hay nhà khoa học máy tính [02]. Ontology là một thuật ngữ được dùng trong trí tuệ nhân tạo có nguồn gốc từ triết học.

Theo quan điểm triết học thì ontology nghiên cứu về sự tồn tại, về sự phân loại và cấu trúc của các đối tượng, các thuộc tính, biến cố, quy trình và các quan hệ trong mọi lĩnh vực [24]. Ontology còn dùng để phân loại: vật chất và hiện tượng, đầy đủ và rút gọn.

Theo quan điểm của các nhà khoa học máy tính thì thuật ngữ 'ontology' được sử dụng để mô tả việc xây dựng các phân loại. Theo ngữ nghĩa này thì ontology là một từ điển các thuật ngữ tạo nên cú pháp và từ vựng để biểu diễn tri thức nhằm chia sẻ tri thức giữa các hệ thống thông tin khác nhau (như một tiếng nói chung). Trên cơ sở đó thì hoàn loạt những định nghĩa về ontology được đưa ra trong [02], [08], [13], [15], [25], [26], [27], [28], [29], [31], [32], [34]. Theo các định nghĩa trên, một ontology phải có những tính chất sau:

+ Được sử dụng để mô tả một phạm vi ứng dụng cụ thể.
+ Các khái niệm và quan hệ được định nghĩa rõ ràng trong phạm vi ứng dụng đó.
+ Có cơ chế tổ chức các khái niệm (thường là phân cấp).
+ Có sự đồng thuận về mặt ý nghĩa các khái niệm của những người cùng sử dụng ontology.

Từ những định nghĩa trên, tác giả tóm lược và rút ra một định nghĩa về ontology như sau: **"Ontology định nghĩa các *khái niệm* (concepts) dùng để *mô tả* và *biểu diễn* một lĩnh vực của tri thức".** Các loại khái niệm được *mô tả* gồm:

- Lớp đối tượng trong lĩnh vực (classes)
- Đối tượng cụ thể (instance)
- Mối quan hệ giữa các đối tượng (relationships)
- Thuộc tính, tính chất của các đối tượng (properties)
- Hàm (functions) và quy trình (processes) liên quan đến các đối tượng.
- Ràng buộc (constraints) và luật (rules) liên quan đến các đối tượng.

Biểu diễn một mô tả là mã hóa mô tả đó sao cho người và máy có thể sử dụng được. Để biểu diễn, người ta sử dụng các từ ngữ (terms) và câu (sentences), tạo thành một ngôn ngữ biểu diễn tri thức. Ngôn ngữ dùng để biểu diễn cần phải rõ ràng, chính xác, không nhập nhằng.

Có thể nói các ngôn ngữ ontology hiện nay thuộc các nhóm sau:

- *Logic vị từ cấp 1*: CycL, KIF, …
- *Dựa trên frames*: XOL, Ontolingua, SHOE, OKBC, Flogic, ...
- *Logic mô tả*: LOOM, OIL, OWL, DAML+OIL, …
- *Đồ thị khái niệm*: OML/CKML, …
- *Mạng ngữ nghĩa*: RDF

2.2.2. Các thành phần chính của ONTOLOGY

2.2.2.1. Các khái niệm (concepts)

Những khái niệm được tổ chức phân loại để định nghĩa tập hợp các thuộc tính hoặc tập hợp các thao tác vốn đặc trưng của bất cứ thành phần nào của khái niệm.

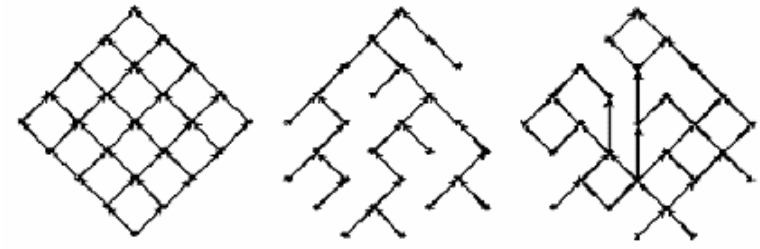
Ví dụ: trong ontology về toán đồ thị, đỉnh và cạnh là 2 khái niệm.

2.2.2.2. Quan hệ (relation)

Kiểu tương tác giữa các khái niệm.

Ví dụ: "Khái niệm cây có hướng" còn là khái niệm con của "khái niệm cây", "song song" là một quan hệ.

Đặc biệt hoá của quan hệ là các quan hệ phân cấp. Một số quan hệ phân cấp thường dùng là: lưới, cây và xoắn.



Hình 2.5: Đồ thị lưới, cây, xoắn

Mỗi nút đại diện cho một khái niệm. Mũi tên trong các đồ thị bắt nguồn từ nút chỉ ra nút cha của nút đó. Nút mà không có nút cha nào được gọi là khái niệm cơ bản. Ví dụ: trong ontology toán đồ thị, đỉnh là khái niệm cơ bản.

2.2.2.3. Hàm (function)

Các thao tác thực hiện trên ontology.

Ví dụ: cấp của đồ thị có thể được tính toán bằng tổng số đỉnh của đồ thị.

2.2.2.4. Tiên đề (axiom)

Tiên đề có thể phân tích thành các luật, các luật thể hiện các tri thức mang tính phổ quát trên các khái niệm và các loại sự kiện khác nhau. Mỗi luật cho ta một qui tắc suy luận để đi đến một sự kiện mới từ sự kiện nào đó, và về mặt cấu trúc nó gồm 2 thành phần chính là: phần giả thuyết và phần kết luận của luật. Phần giả thiết và phần kết luận đều là các tập hợp sự kiện trên các đối tượng nhất định. Như vậy, một luật r có thể được mô hình dưới dạng r: $\{sk_1, sk_2, \ldots, sk_n\} \Rightarrow \{sk_1, sk_2, \ldots, sk_m\}$

Ví dụ: trong đồ thị số đỉnh bậc lẻ là một số chẵn.

2.2.2.5. Thể hiện (instance)

Đại diện cho những phần tử riêng biệt của khái niệm hay quan hệ.

Ví dụ: tam giác được kí hiệu ABC là thể hiện của khái niệm tam giác.

2.2.3. Phân loại ONTOLOGY

2.2.3.1. Theo cách phân loại của John F. Sowa

Có 2 loại:

- **Ontology hình thức (formal ontology)**: Là ontology mô tả các khái niệm một cách chi tiết đến các tiên đề và định nghĩa mà không quan tâm đến các mô tả này có thực hiện dễ dàng trong máy tính hay không. Ontology hình thức thường có xu hướng nhỏ, nhưng các tiên đề và định nghĩa thường rất phức tạp trong suy luận và tính toán. Những ontology này thường do các nhà triết học thiết kế.
- **Ontology thuật ngữ (terminological ontology)**: Là ontology mô tả các khái niệm theo hướng tiên đề và định nghĩa được phát biểu dạng logic hoặc trong một vài ngôn ngữ hướng đối tượng để cho máy tính thực hiện việc chuyển đổi theo dạng logic. Dạng logic này không có sự hạn chế về việc phát biểu các tiên đề và định nghĩa cho máy tính thực hiện dễ dàng. Các tiên đề và định

nghĩa chỉ mô tả đến các vấn đề mà ứng dụng quan tâm. Ontology thuật ngữ lớn, nhưng các tiên đề và định nghĩa thường rất dễ dàng trong suy luận và tính toán. Những ontology này thường do các nhà tin học thiết kế.

2.2.3.2. Theo cách phân loại của D. Fensel

Có 7 loại:

- **Knowledge Representation ontology**: Dựa trên các cách biểu diễn tri thức truyền thống.

  Ví dụ: Frame-Ontology.

- **General/Common ontology**: Từ vựng liên quan đến mọi thứ, sự kiện, thời gian, không gian, ...

  Ví dụ: Ontology về bảng trao đổi giữa meter và inch.

- **Meta-ontology**: Định nghĩa các ontology.

  Ví dụ: Registry Ontology, dùng để quản lý các ontology khác.

- **Domain ontology**: Từ vựng của các khái niệm trong một phạm vi.

  Ví dụ: Ontology về lý thuyết hoặc các nguyên lý cơ bản của một miền.

- **Task ontology**: Hệ thống các từ vựng của các thuật ngữ để giải quyết các vấn đề kết hợp liên quan đến nhiệm vụ mà có thể cùng hoặc không cùng phạm vi ứng dụng cụ thể.

  Ví dụ: Ontology về kế hoạch phân công nhiệm vụ.

- **Domain-task ontology**: Task ontology được sử dụng lại trong một phạm vi ứng dụng cụ thể.

  Ví dụ: Ontology về kế hoạch phân công nhiệm vụ của các chuyến bay.

- **Application ontology**: Chứa các kiến thức cần thiết của một ứng dụng trong phạm vi ứng dụng nhất định.

  Ví dụ: Ontology hình học.

2.2.3.3. Một cách phân loại khác

Ontologies được chia thành 2 loại chính: Các ontologies chủ yếu chỉ gồm phép phân loại và mô hình chi tiết một lĩnh vực, cung cấp nhiều ràng buộc hơn về mặt

ngữ nghĩa của lĩnh vực. Cộng đồng ontology gọi đó là "lightweight ontology" và "heavyweight ontology" [02], [14].

- *Lightweight ontology* bao gồm khái niệm, các phân loại khái niệm, mối quan hệ đơn giản giữa các khái niệm và các thuộc tính mô tả khái niệm.
- *Heavyweight ontology* là sự mở rộng của lightweight ontology. Heavyweight ontology có thêm *hàm*, các *tiên đề* và *ràng buộc*, vì vậy biểu diễn tường minh và bao quát hơn.

Phần lớn các ontologies hiện nay đều tập trung vào các lĩnh vực như web ngữ nghĩa, thương mại điện tử, sinh học… Ontologies biểu diễn tri thức phức tạp thì rất ít, và hầu như không có ontologies nào biểu diễn tri thức của các ứng dụng giải toán, ngoại trừ mô hình COKB trong [07]. Có thể nói bản thân mô hình COKB đã là một ontology, vì mô hình COKB thỏa mãn tất cả các đặc điểm của một ontology. Mô hình COKB có các ưu điểm sau:

- Thích hợp cho việc thiết kế một CSTT với các khái niệm có thể được biểu diễn bởi các đối tượng tính toán C-Object.
- Cấu trúc tường minh giúp dễ dàng thiết kế các module truy cập CSTT.
- Tiện lợi cho việc thiết kế các module giải toán tự động.
- Thích hợp cho việc đặt ra một ngôn ngữ khai báo bài toán và đặc tả bài toán một cách tự nhiên.

2.2.4. Ngôn ngữ ONTOLOGY

Ngôn ngữ ontology là ngôn ngữ dùng để mô tả các thành phần của ontology. Để mô tả mọi thứ về các khái niệm của ontology đòi hỏi phải có một ngôn ngữ chung nhất để diễn đạt. Có 2 loại ngôn ngữ chung nhất:

- **Ngôn ngữ tự nhiên**: Mọi thứ kinh nghiệm của con người đều có thể diễn đạt được bằng một ngôn ngữ tự nhiên (Ví dụ: tiếng Anh, Pháp, Việt Nam, …). Nó là một siêu ngôn ngữ có thể diễn đạt các ngôn ngữ khác như: ký số toán học, ngôn ngữ lập trình, …
- **Ngôn ngữ dựa trên logic**: Mọi thứ được phát biểu chính xác, rõ ràng trong bất kì ngôn ngữ tự nhiên nào đều có thể biểu diễn được trong logic. Nhiều thứ khó có thể nắm được ý nghĩa một cách rõ ràng thì khó có thể biểu diễn

được trong logic (Ví dụ: tình yêu, thơ ca, câu nói đùa, …). Tuy nhiên, bất cứ thứ gì được thực hiện trên máy tính bằng ngôn ngữ lập trình thì đều có thể biểu diễn trong logic. Và để giảm thiểu tính hình thức của logic người ta sử dụng một số ngôn ngữ dựa trên logic.

Các ngôn ngữ biểu diễn thường được dùng trong ontology

- **Conceptual Graphs (CGs)**: Được thiết kế nhằm biểu diễn ngôn ngữ tự nhiên vào ngôn ngữ được trình bày như logic mà con người có thể đọc dễ dàng.
- **Knowledge Interchange Format (KIF)**: Được thiết kế nhằm dễ dàng phân tách cú pháp và một tập kí tự giới hạn để có thể trao đổi được trong các hệ máy tính khác nhau.
- **Conceptual Graph Interchange Form (CGIF)**: Là mức trung gian của CGs và KIF. Giúp cho CGs và KIF được chuyển đổi dễ dàng thông qua CGIF.
- **Các ngôn ngữ biểu diễn web ngữ nghĩa**: XML, XML Schema, RDF, RDF Schema, OIL, DAML+OIL, OWL.
- **Logic mô tả (description logic)**: là dạng logic dùng để mô tả các khái niệm trong phạm vi ứng dụng nhất định. Lý thuyết của logic mô tả tập trung chủ yếu vào 2 thành phần:

  TBox (Terminological Box): mô tả các khái niệm
  ABox (Assertional Box): mô tả các thể hiện

Đa số ngôn ngữ ontology hiện nay đều sử dụng các luật của logic mô tả.

2.3. PHƯƠNG PHÁP LUẬN PROMETHEUS

2.3.1. Tại sao chúng ta cần phải có phương pháp mới?

Có rất nhiều phương pháp đang tồn tại cho việc thiết kế phần mềm. Đặc biệt là phương pháp phân tích và thiết kế hướng đối tượng đã được nghiên cứu và phát triển rộng rãi. Nhưng nó thật sự không thích hợp để xây dựng hệ thống Agent. Mặc dù Agent và đối tượng có điểm giống nhau nhưng chúng khác nhau đáng kể.

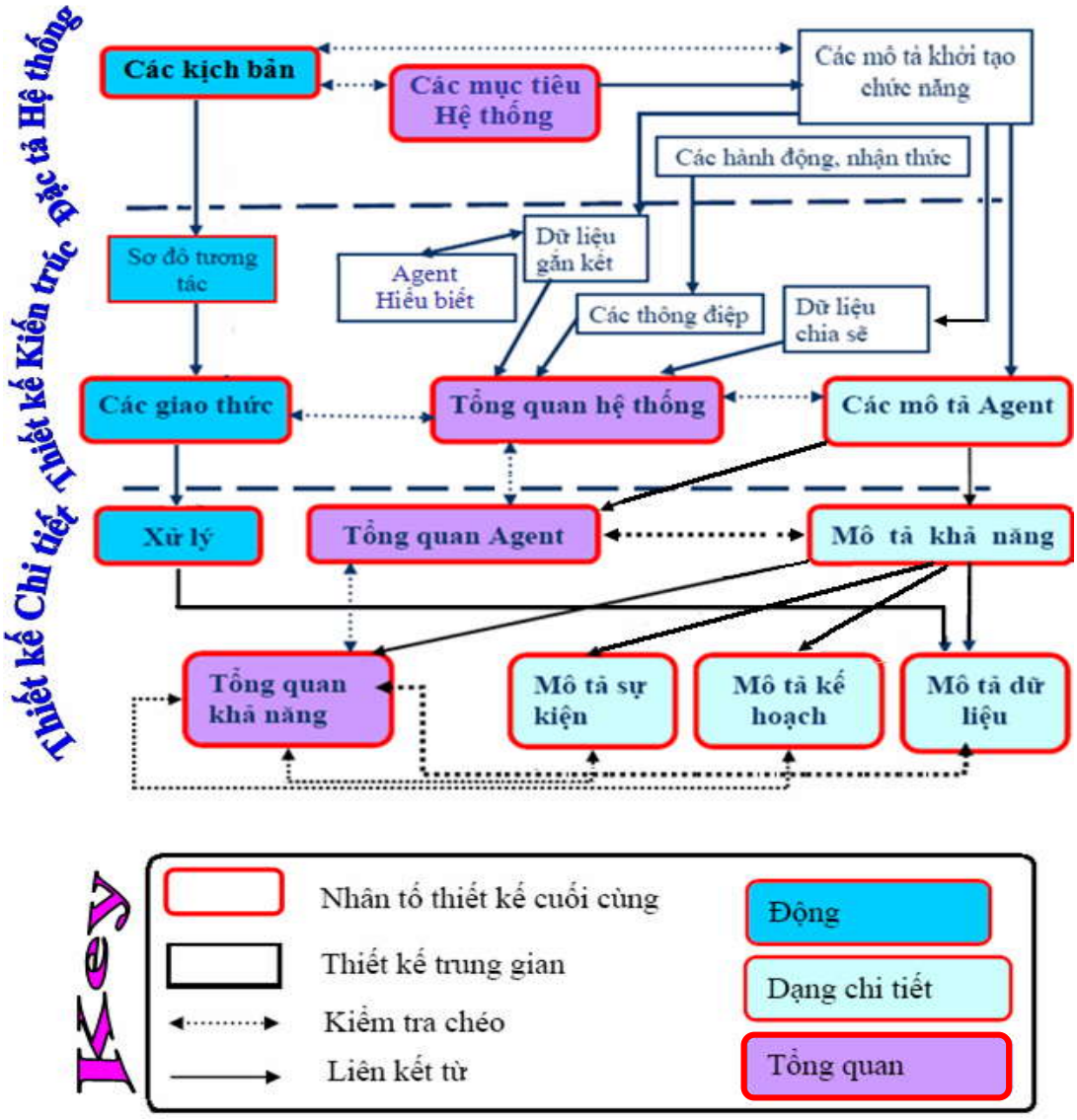
Cho nên ta có thể dùng phương pháp hướng đối tượng để thiết kế Agent nhưng nó không tự nhiên và sử dụng không tốt.

Một phương pháp hỗ trợ tính tiên phong của Agent cần hỗ trợ việc mô hình các mục tiêu một cách rõ ràng mà phương pháp hướng đối tượng không có. Ngược lại, phương pháp Prometheus mô hình các mục tiêu cho nên nó đã hỗ trợ việc thiết kế Agent có tính tiên phong thực hiện.

Một vài điểm khác của phương pháp Prometheus so với phương pháp hướng đối tượng:

- Prometheus cung cấp cách thức xác định loại Agent trong hệ thống.
- Prometheus xử lý thông điệp như những thành phần. Điều này cho phép một thông điệp (hoặc sự kiện) được giải quyết bằng nhiều kế hoạch (plan) mà chủ yếu để đạt được kết quả một cách linh động và đáng tin cậy.
- Prometheus phân biệt giữa hành động và nhận thức từ thông điệp và xem xét quy trình nhận thức một cách rõ ràng. Quá trình nhận thức rất quan trọng để xác định Agent ở vị trí nào trong thế giới thực và lấy nhận thức của chúng.
- Prometheus phân biệt những thành phần bị động (dữ liệu, lòng tin) từ những thành phần chủ động (Agent, khả năng, kế hoạch).
- Prometheus gán cho Agent những thuộc tính thuộc về trí tuệ như lòng tin và điều đáng mong muốn đến Agent, nó nắm những thuộc tính thuộc về trí tuệ trong suốt quá trình thiết kế và phân tích.
- Prometheus cho phép lập mô hình hệ thống mà có thể tích hợp với những hệ thống con khác không theo mô hình agent và không theo phương pháp luận Prometheus.

Như vậy mặc dù Prometheus có nhiều điểm khác so với phương pháp hướng đối tượng nhưng chúng cũng có liên quan, Prometheus vẫn dựa trên những phương pháp và ký hiệu của hướng đối tượng.



Hình 2.6: Các giai đoạn thiết kế của phương pháp luận Prometheus

2.3.2. Giới thiệu tổng quan về Prometheus

Phương pháp luận Prometheus định nghĩa một quy trình chi tiết cho việc đặc tả, thiết kế, hiện thực và kiểm thử những hệ thống phần mềm hướng Agent [22]. Trong quá trình chi tiết, nó định nghĩa một dãy nhiều yếu tố (artifact) được tạo ra trong suốt quá trình. Một vài artifact được giữ lại, một vài cái chỉ sử dụng như là bước đệm. Một vài artifact là đồ họa trong khi những cái khác là văn bản có cấu trúc.

Artifact của Prometheus liên hệ với những khái niệm Agent. Ví dụ hành động và nhận thức được nắm vững trong giai đoạn đặc tả hệ thống; giai đoạn thiết kế

chi tiết cho kết quả là những kế hoạch, sự kiện và lòng tin và những thực thể sử dụng trong những sơ đồ tổng quan khác nhau tương ứng trực tiếp với những khái niệm.

Prometheus gồm ba giai đoạn:

- **Giai đoạn đặc tả hệ thống (System specification)**: Tập trung vào việc nhận dạng mục tiêu và những chức năng cơ bản của hệ thống, cùng với đầu vào (inputs: nhận thức) và đầu ra (outputs: hành động).
- **Giai đoạn thiết kế kiến trúc (Architectural design)**: Sử dụng đầu ra của giai đoạn trước để xác định các loại Agent mà hệ thống sẽ chứa và chúng sẽ tương tác với nhau như thế nào.
- **Giai đoạn thiết kế chi tiết (Detailed design)**: Xem xét bên trong mỗi Agent và nó sẽ hoàn thành những nhiệm vụ bên trong toàn bộ hệ thống như thế nào.

2.3.2.1. Đặc tả hệ thống

Giai đoạn này đề cập đến 4 vấn đề sau:

- Đặt tả mục tiêu của hệ thống tạo ra một danh sách các mục tiêu (có thể phân tích thành các mục tiêu con) cùng với sự mô tả liên kết giữa các mục tiêu này.
- Định nghĩa tập hợp các chức năng mà được liên kết bởi một hoặc nhiều mục tiêu và nắm bắt một giới hạn hành vi của hệ thống.
- Phát triển các kịch bản thích hợp bao trùm các mục tiêu và cung cấp hướng xử lý của hệ thống để được phát triển.
- Mô tả giao diện giữa hệ thống Agent và môi trường mà nó tồn tại, dưới dạng các nhận thức vào, các hành động ra và các kho thông tin ngoài nơi mà hệ thống sẽ tương tác.

1. Đặc tả các mục tiêu của hệ thống

Đặt tả mục tiêu rất quan trọng trong giai đoạn đặt tả hệ thống vì từ đó ta sẽ xây dựng các chức năng, phát triển kịch bản và đặt tả các giai đoạn chi tiết sau. Cho biết hệ thống sẽ phải làm gì. Mô tả ngắn gọn về hệ thống, khi đặc tả hệ thống phải luôn luôn tập trung vào các lý do xây dựng hệ thống. Các mục tiêu nên hướng đến chức năng của agent.

- Nhận dạng mục tiêu khởi tạo: Mô tả tóm tắt khởi tạo ứng dụng và đưa ra tập hợp các mục tiêu của hệ thống được rút ra từ mô tả này.

- Cải tiến mục tiêu: Cải tiến các mục tiêu sử dụng kỹ thuật đơn giản là câu hỏi "how" và vấn đề quan tâm đối với mỗi mục tiêu là "mục tiêu đạt được như thế nào?". Cách giải quyết là sẽ đưa ra các mục tiêu con từ các mục tiêu đã khởi tạo.

2. Xác định các chức năng cơ bản của hệ thống

Sau khi cải tiến các mục tiêu, chúng ta có thể sắp xếp lại chúng, tìm những mục tiêu giống nhau và gộp chúng lại thành một nhóm để tạo thành một chức năng cho hệ thống. Vì thế có những mục tiêu con lúc ban đầu không nằm chung trong mục tiêu khởi tạo nhưng được gộp chung thành một nhóm để tạo thành một chức năng.

Chức năng là tập hợp những hành vi, bao gồm một nhóm các mục tiêu có liên hệ, cùng với các cảm nhận, hành động và dữ liệu thích hợp với hành vi.

3. Phát triển kịch bản use-case minh họa cho hoạt động của hệ thống

Kịch bản mô tả các bước diễn ra trong hệ thống. Khi phát triển mục tiêu, ta thường tạo ra các kịch bản cho thấy các mục tiêu này sẽ là một phần của các quá trình khác nhau trong hệ thống. Trong quá trình phát triển kịch bản có thể nhận ra các mục tiêu mới và bổ sung vào những chức năng thích hợp.

4. Đặc tả giao diện giữa hệ thống và môi trường

MAS ở trong môi trường động hoặc môi trường thay đổi mà có thể bị ảnh hưởng. Vấn đề cần giải quyết là MAS sẽ tương tác với môi trường như thế nào? Đặc biệt là cái gì sẽ nhập vào MAS trong khi nó đang thực thi và MAS sẽ tương tác gì với môi trường. Theo Russell và Norvig (1995) ta có thể gọi thông tin nhập vào là nhận thức (cảm nhận) "*percepts*" và cơ chế tương tác với môi trường là hành động "*action*".

Bốn bước này không nhất thiết phải theo một trình tự nhất định. Đặc tả hệ thống là một quy trình lặp: bắt đầu bằng việc nhận dạng mục tiêu, sau đó tiếp tục với kịch bản, mục tiêu và chức năng, cùng với nhận dạng hành động, cảm nhận và dữ liệu.

2.3.2.2. Thiết kế kiến trúc

Giai đoạn này đề cập đến 3 vấn đề sau:

- Quyết định những loại Agent nào được sử dụng trong hệ thống.
- Mô tả tương tác giữa các Agent sử dụng sơ đồ và các giao thức tương tác.
- Thiết kế toàn bộ cấu trúc hệ thống (mô tả việc sử dụng một sơ đồ khái quát hệ thống).

1. Đặc tả loại Agent

a. Việc quyết định loại Agent (Deciding on the Agent types)

Sự quyết định đơn giản quan trọng nhất trong thiết kế kiến trúc là hệ thống sẽ chứa những Agent nào. Hệ thống của những Agent nên được kết nối một cách linh hoạt ngay khi có thể.

Các bước cần thực hiện trong tiến trình xác định loại Agent là:

- Nhóm những chức năng vào các Agent bằng việc xem xét lựa chọn các nhóm.
- Xem lại tính kết nối (sử dụng sơ đồ hiểu biết Agent) và quyết định một nhóm thích hợp hơn.
- Phát triển việc mô tả Agent.

b. Nhóm những chức năng

Lý do chính cho việc quyết định nhóm những chức năng chắc chắn với nhau trong một Agent đơn là:

- Những chức năng xem như có liên hệ - nó tạo ra cảm biến tập hợp chúng lại.
- Chức năng đòi hỏi rất nhiều thông tin giống nhau. Nếu được nhóm trong Agent đơn, thì điều này có thể được biểu diễn bên trong bản thân cấu trúc dữ liệu Agent. Nếu các Agent tách rời được sử dụng thì thông tin sẽ phải thông qua các thông điệp trừ khi kho dữ liệu được chia sẻ không là một quyết định thiết kế tốt.

Những lí do để không nhóm những chức năng:

- Chức năng không có quan hệ rõ ràng.
- Chức năng tồn tại trên nền phần cứng khác nhau.
- Những con số khác nhau của những chức năng được yêu cầu ở thời gian chạy. Dẫn đến tính kết nối và kết dính là sơ đồ kết nối dữ liệu.

Những nhóm tiềm năng có thể ước lượng và lọc bằng việc sử dụng một sơ đồ hiểu biết Agent.

c. Xem xét lại tính kết nối Agent – Sơ đồ hiểu biết (Review Agent coupling- Acquaitance)

Một tiêu chuẩn thiết kế là nhằm mục đích cho một hệ thống có ít tính kết nối khi có thể. Để đánh giá một nhóm tiềm năng cho tính kết nối, ta sử dụng một sơ đồ Agent hiểu biết miêu tả mỗi loại Agent. Thông tin về sự tương tác Agent được trích từ việc mô tả chức năng và mỗi Agent kết nối với những Agent khác mà nó tương tác.

d. Phát triển các mô tả Agent

Trong khi quyết định loại Agent nào sẽ có bên trong hệ thống, ta cần phải giải quyết một số câu hỏi:

- Mỗi loại nên có bao nhiêu agent?
- Vòng đời của mỗi agent? Biến cố nào tạo ra và kết thúc agent khi hệ thống hoạt động?
- Khởi tạo agent: cần phải thực hiện những gì?
- Khi agent kết thúc: cần dọn dẹp những gì?
- Mục tiêu của agent là gì?
- Agent cần phản ứng lại các cảm nhận nào?
- Agent có những hành động nào?
- Agent sử dụng và tạo ra những dữ liệu nào?

2. Đặc tả tương tác giữa các agent

Một khi mà loại Agent được quyết định, khía cạnh kế tiếp của thiết kế kiến trúc là "đặc tả sự tương tác giữa những Agent, nắm bắt được những khía cạnh động của hệ thống". Việc xây dựng này dựa trên hầu hết các loại Agent và cũng như các kịch bản mô tả từ giai đoạn đặc tả hệ thống.

Sự phát triển của giai đoạn đặc tả là sự tương tác giữa những Agent gồm những phần sau:

- Sử dụng các kịch bản use-case để phát triển *sơ đồ tương tác*.
- Tổng quát hóa sơ đồ tương tác thành các *giao thức tương tác*.
- Mô tả các giao thức và thông điệp.

3. Thiết kế cấu trúc toàn bộ hệ thống

Trong bước cuối cùng của giai đoạn thiết kế kiến trúc, thực hiện:

- Xác định ranh giới của hệ thống Agent và sự tương tác giữa những hệ thống con khác.
- Mô tả nhận thức và hành động, quan hệ giữa nhận thức và hành động, những mối liên hệ giữa những Agent.
- Xác định tất cả các loại dữ liệu chia sẽ, cả những dữ liệu tồn tại bên ngoài và những dữ liệu chia sẽ bên trong hệ thống.
- Phát triển khái quát sơ đồ hệ thống.

2.3.2.3. Thiết kế chi tiết

Thiết kế chi tiết tập trung vào phát triển cấu trúc của các Agent và làm thế nào để nó đạt được tác vụ bên trong hệ thống.

Chúng ta tiếp tục cải tiến Agent bằng định nghĩa khả năng (các modules bên trong Agent), các sự kiện, kế hoạch và các CTDL chi tiết bên trong. Quá trình này bắt đầu bằng cách mô tả bên trong Agent dưới dạng các khả năng. Các khả năng tiếp tục được chi tiết hóa. Ở mức thấp nhất, khả năng được định nghĩa dưới dạng các kế hoạch, sự kiện và dữ liệu. Bên cạnh đó, chúng ta tiếp tục phát triển những mặt động của hệ thống bằng cách cải tiến những giao thức tương tác trong đặc tả quy trình. Quá trình thiết kế chi tiết gồm hai phần:

1. Agent, khả năng và xử lý

Trong phần đầu của thiết kế chi tiết, những khía cạnh được đề cập như sau:

- Quyết định yêu cầu khả năng cần thiết của mỗi Agent để thực hiện nhiệm vụ của nó.
- Mô tả quan hệ giữa các khả năng sử dụng sơ đồ tổng quan Agent.
- Phát triển đặc tả giao thức để xác định một số xử lý bên trong của bản thân từng Agent.
- Tổng hợp thông tin về mỗi khả năng trong mô tả khả năng.

a. Khả năng

Trong thiết kế chi tiết, Ta xác định những chức năng nào được thực hiện bởi mỗi Agent, và danh sách những chức năng này là hệ quả tất yếu cho khả năng Agent.

- Có thể chia khả năng thành nhiều khả năng nhỏ hơn.
- Mỗi khả năng nên xác định mục tiêu nó cần đạt được trong hệ thống.

b. Sơ đồ tổng quan Agent

Sơ đồ này cho chúng ta biết mối quan hệ giữa các khả năng, vì thế cung cấp mức cao hơn về cái nhìn bên trong Agent và thông điệp theo nó giữa những khả năng này cũng như dữ liệu bên trong Agent.

c. Đặc tả xử lý

Ta nhận thấy sơ đồ xử lý được phát triển dựa vào giao thức cũng như kịch bản được phát triển và mục đích của Agent. Một vài sơ đồ xử lý sẽ không được nhận biết từ giao thức khi chúng không bao gồm tín hiệu tương tác trong Agent khác.

d. Phát triển khả năng và mô tả xử lý

Chúng ta phát triển mô tả cho mỗi khả năng, và từng sơ đồ xử lý. Thông tin cho mô tả khả năng có thể được rút ra hầu hết toàn bộ từ thông tin có sẵn, nhưng nó hữu ích để có thể tập hợp lại ở một nơi.

2. Khả năng, kế hoạch và sự kiện

Trong phần cuối cùng của chi tiết thiết kế, mỗi khả năng được chia ra thành những khả năng nhỏ hơn hoặc những sự kiện, thiết lập kế hoạch mà cung cấp chi tiết của việc tác động lại tình huống hoặc đạt được mục đích. Ở bước này, số chi tiết thiết kế thực thi nền trở nên quan trọng. Ta tập trung vào nền **BDI** (*Belief-Desire-Intentio*n: cố gắng mong muốn đạt mục tiêu) trong phần đặc biệt, mà được mô tả những đặc điểm bằng trình bày có thứ tự kế hoạch với những kích hoạt, và mô tả cho mỗi kế hoạch được xác định phạm vi nó ứng dụng. Hệ thống BDI chọn giữa kế hoạch mà có thể ứng dụng, quay lại để thử lại kế hoạch khác nếu một khởi tạo đã chọn không thành công.

Số các vấn đề quan tâm trong phần này là được đặc tả theo hệ thống làm việc BDI, nguyên tắc là dễ dàng để sắp xếp những Agent khác trong hệ thống. Đặc biệt, bất kỳ hệ thống sử dụng Agent với những kế hoạch tác động lại đến sự kiện có thể dễ dàng trở nên thích hợp hơn.

a. Sơ đồ tổng quan khả năng

Ta đã phát triển sơ đồ tổng quan Agent. Đi sâu vào mức chi tiết được cung cấp bởi sơ đồ khả năng lấy từ những khả năng đơn giản và mô tả bên trong của chính nó. Ở mức cuối cùng, sẽ cung cấp kế hoạch, với thông điệp vào cung cấp kết nối giữa kế hoạch, ngay khi chúng thực hiện giữa những khả năng. Tại mức này, sơ đồ có thể có nhiều khả năng liên tiếp hoặc trộn lẫn của khả năng và kế hoạch. Mỗi kế hoạch phải có một thông điệp kích hoạt một cách chính xác.

b. Nhiệm vụ con và lựa chọn kế hoạch

Mức thấp nhất của chi tiết thiết kế, mỗi thông điệp vào đến khả năng phải có một hoặc nhiều kế hoạch để trả lời các thông điệp đó. Kế hoạch trả lời những thông điệp cung cấp nhiều cách để tác động lại tình huống. Nó cho phép mỗi bản thân kế hoạch trở nên đơn giản và không phức tạp, trả lời những thông điệp chính xác cho một tình huống đặc biệt.

Mỗi kế hoạch có thể được chia thành những tác vụ con, mỗi nhiệm vụ được đại diện bằng một thông điệp vào.

c. Sự kiện và thông điệp

Khi xác định, ta dùng sự kiện gốc để biểu diễn mọi thứ mà có thể kích hoạt lựa chọn và thực thi kế hoạch. Sự kiện nguồn có thể là thông điệp mà được nhận dạng giữa các Agent, nhận thức vào từ môi trường, hoặc thông điệp bên trong đến một Agent, xác định tác vụ con hoặc thông điệp giữa kế hoạch và (hoặc) khả năng.

Thông điệp giữa các Agent cũng quan trọng để đặc tả có hay không câu trả lời là mong đợi và chi tiết thiết kế của câu trả lời.

d. Thiết kế chi tiết hành động và nhận thức

Hành động và nhận thức được xác định trong giai đoạn trước. Nếu có xử lý dữ liệu vào để đạt được nhận thức, nó cần được thiết kế và đặc tả.

Nhận thức và hành động có thể đơn giản hoặc phức tạp. Chúng có thể là hệ thống đơn giản gọi là chức năng hỗ trợ và phổ biến. Đặc biệt trong hệ thống vật lý, hành động thường yêu cầu việc quan trọng để đạt được phản ứng hiệu quả.

e. Dữ liệu

Trình bày dữ liệu là phần quan trọng trong bất kỳ hệ thống hoạt động thiết kế nào. Tuy nhiên, trình bày dữ liệu phải được quyết định và đặc tả, sử dụng một vài phương pháp thích hợp.

Một vài môi trường phát triển của Agent đưa ra một vài hỗ trợ đặc biệt cho việc trình bày dữ liệu, như là dữ liệu "beliefsets (đáng tin cậy)" có sẵn trong JACK.

f. Kiểm tra tính toàn diện và tính nhất quán

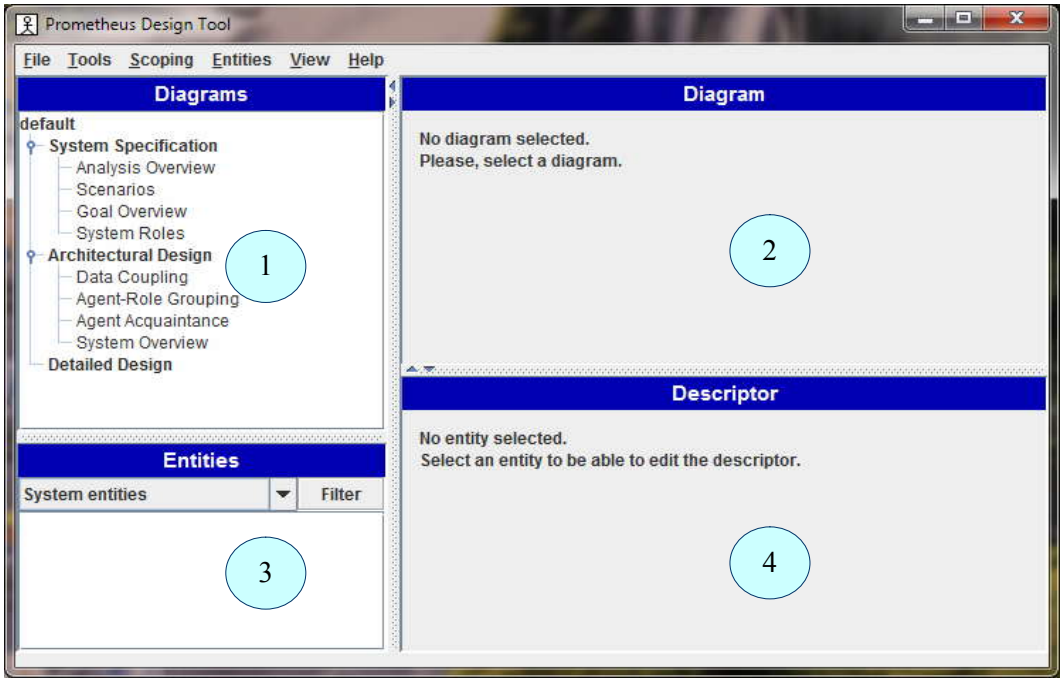
Đối với đặc tả hệ thống và thiết kế cấu trúc, thiết kế chi tiết cũng được kiểm tra tính toàn diện và tính chắc chắn. Ở giai đoạn trước, định rõ tính chắc chắn là rất quan trọng, nhưng ở đây không đi sâu vào vấn đề này. Ở giai đoạn này, có sự gia tăng các thực thể và sẽ có liên hệ đích danh, vì thế nó nhận định để biết rằng chỉ có tên giống chính xác mới chuyển đến cùng thực thể.

- Kiểm tra tính toàn diện cho mỗi Agent: Agent là thật sự chứa đầy đủ các chức năng mà có trong thiết kế cấu trúc.
- Không có lỗi và thiếu sót: Bảo đảm tất cả dữ liệu đều được sử dụng và tạo ra. Bảo đảm tất cả các thông điệp đều được gửi và nhận.
- Kiểm tra tính chắc chắn giữa các nhân tố thiết kế cuối cùng. Kiểm tra việc đảm bảo tính chắc chắn của giao diện và giữa các mức, tính chắc chắn giữa sơ đồ tổng quan và mô tả, tính chắc chắn của các chi tiết kế hoạch.

2.3.3. Công cụ hỗ trợ thiết kế MAS

Đề tài sử dụng Prometheus Design Tool – PDT do RMIT Intelligent Agents Group [22] phát triển. PDT được dùng để thiết kế các hệ thống theo tiếp cận hướng agent. PDT hỗ trợ ba giai đoạn thiết kế trong Prometheus, từ đặc tả hệ thống, thiết kế kiến trúc cho đến thiết kế chi tiết. Thiết kế chi tiết do PDT tạo ra có thể chuyển trực tiếp cho JACK Development Enviroment (JDE) [21] để hiện thực.

2.3.3.1. Cửa sổ giao diện của PDT

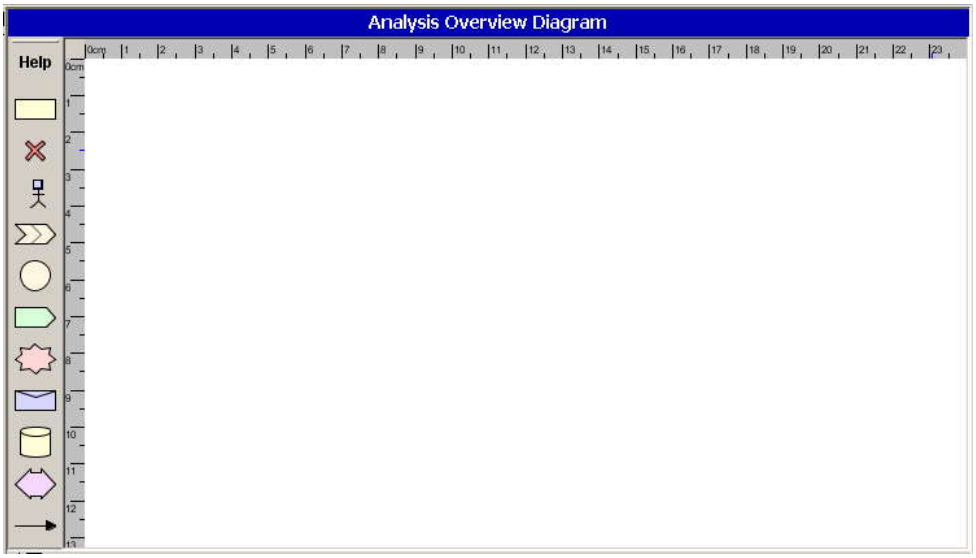


Hình 2.7: Màn hình giao diện của PDT

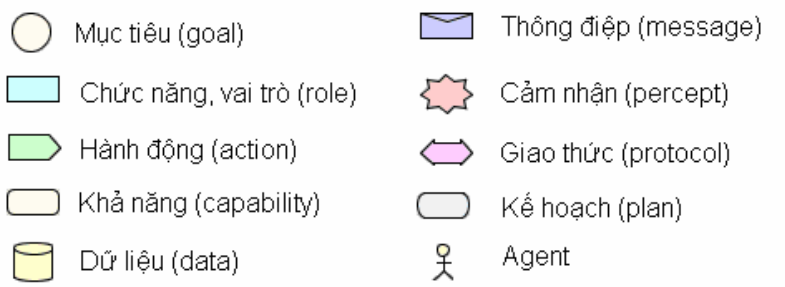
1. Danh sách các sơ đồ (Diagrams List): liệt kê tất cả các sơ đồ thiết kế tương ứng với ba giai đoạn trong phương pháp luận Prometheus.
2. Xem sơ đồ (Diagram View): trình bày sơ đồ được chọn trong Diagram List. Có thể thêm, xóa, hoặc hiệu chỉnh các thực thể, tạo liên kết giữa các thực thể trong sơ đồ.
3. Danh sách các thực thể (Entities List): được dùng để chọn một thực thể để xem mô tả của nó trong Entity Descriptor View. Sau khi được chọn, chi tiết của thực thể sẽ hiển thị trong phần mô tả.
4. Mô tả các thực thể (Entity Descriptor View): hiển thị mô tả của thực thể được chọn. Nội dung mô tả của các thực thể khác nhau. Có thể xóa một thực thể từ Entity Descriptor View. Khi xóa, toàn bộ phần mô tả cùng với các đường liên kết đến thực thể đó đều bị xóa.

2.3.3.2. Tạo sơ đồ

Trước tiên, chọn loại sơ đồ từ Diagrams List. Sau đó sử dụng thanh công cụ để thêm các thực thể vào sơ đồ.



Hình 2.8: Sơ đồ toàn cảnh hệ thống



Hình 2.9: Ý nghĩa các ký hiệu sử dụng

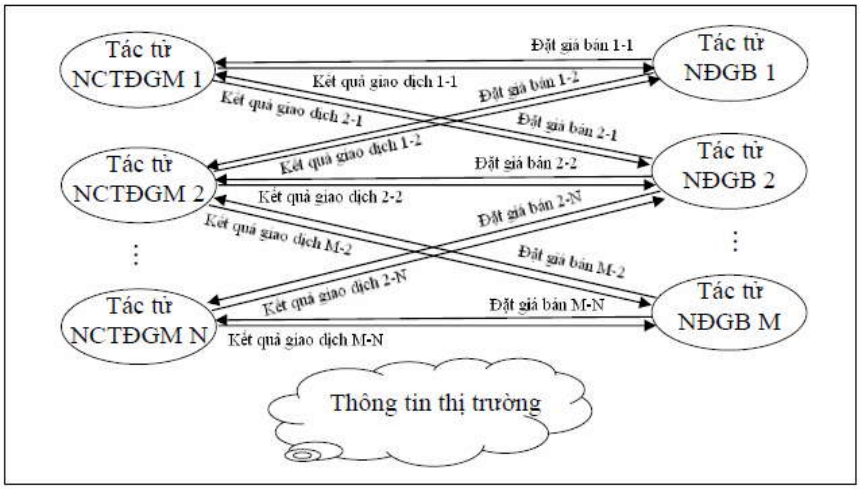
Mỗi sơ đồ có một số loại thực thể tương ứng. Ví dụ sơ đồ toàn cảnh hệ thống có các loại thực thể sau: Agent, Hành động, Dữ liệu, Thông điệp, Cảm nhận, Giao thức, Mũi tên….

2.4. NHỮNG ĐỀ TÀI LIÊN QUAN

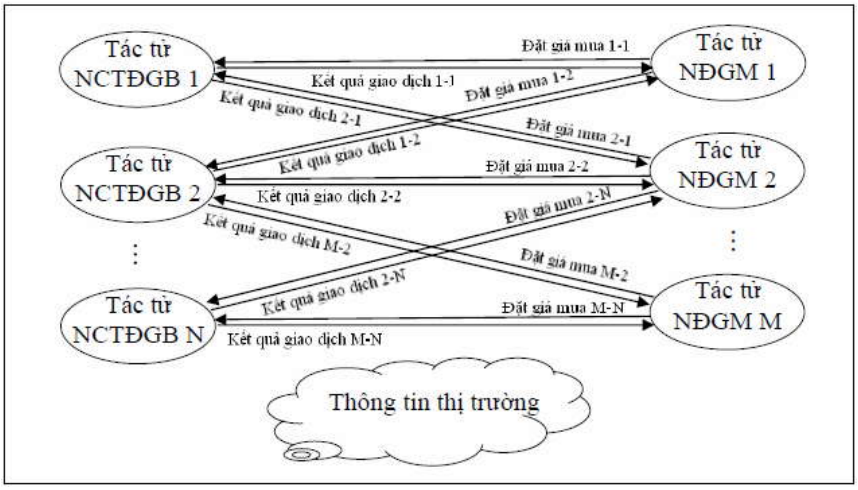
Một số đề tài luận văn Thạc sỹ ứng dụng hệ thống MAS gần đây:

1. **Ứng dụng tác tử thông minh cho đấu giá hàng hóa trong giao dịch thương mại** của ThS Nguyễn Trọng Hòa [05]. Đề tài đã đề xuất một mô hình MAS nhằm hỗ trợ cho việc đấu giá ngược, đấu giá nhiều-nhiều trong giao dịch thương mại. Hệ thống này rất đơn giản chỉ có 4 loại Agent là: Agent cho "Người chủ trì đấu giá bán - NCTĐGB", Agent cho "Người chủ trì đấu giá mua - NCTĐGM", Agent cho "Người đấu giá bán - NĐGB" và Agent cho "Người đấu giá mua - NĐGM". Tác giả xây dựng mô hình chưa theo một

phương pháp luận chính qui nào hỗ trợ cho việc thiết kế hệ thống MAS. Hệ thống được cài đặt thử nghiệm bằng ngôn ngữ C# (VS2005).



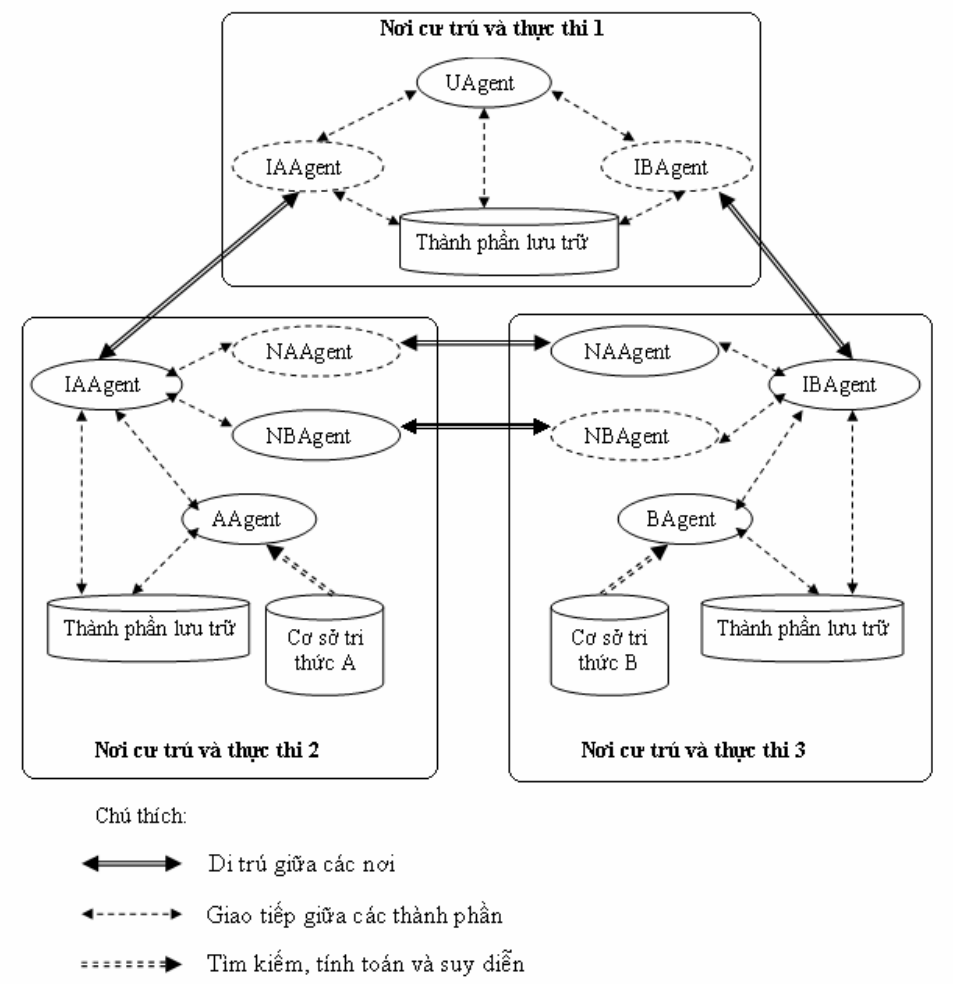
Hình 2.10: Mô hình đấu giá nhiều - nhiều (Người mua chủ trì)



Hình 2.11: Mô hình đấu giá nhiều - nhiều (Người bán chủ trì)

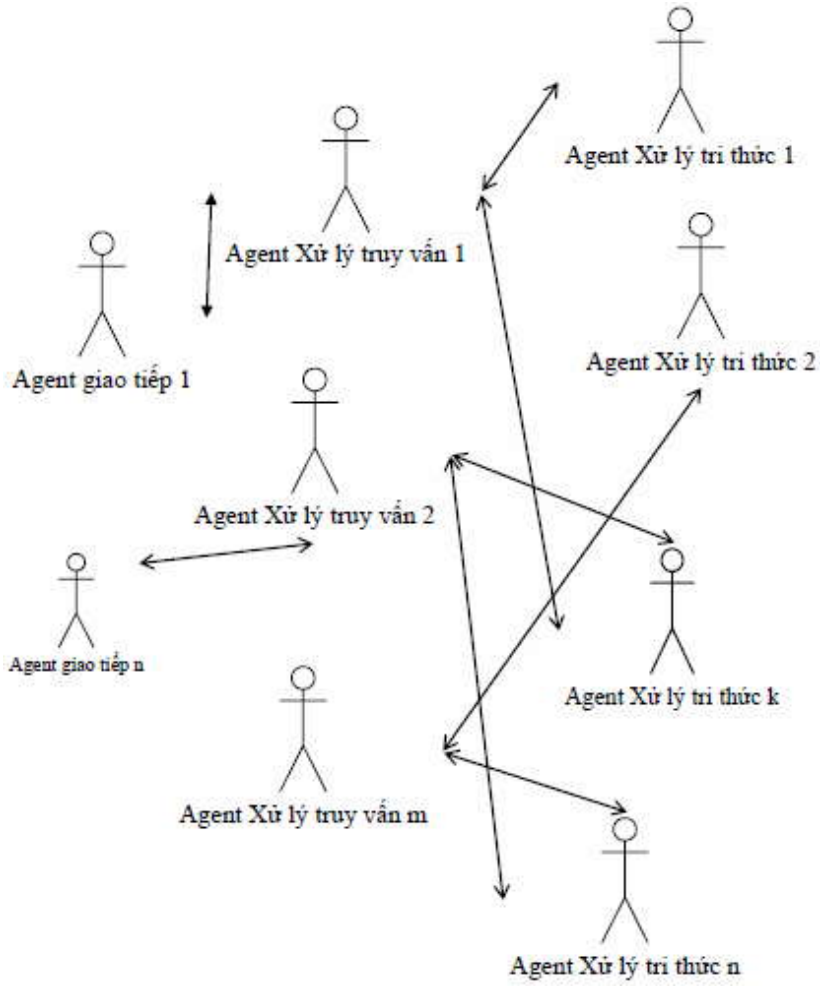
2. **Xây dựng một mô hình hệ thống Multi-Agent và ứng dụng trong E-Learning** của ThS Nguyễn Trần Minh Khuê [03]. Đề tài đã đề xuất một mô hình Multi-agent nhằm hỗ trợ E-Learning. Hệ thống này gồm có 4 agent: Agent giao tiếp người dùng (Intelligent Agent), Agent trung gian (Mobile Agent), Agent tri thức (Intelligent Agent) và Agent báo tin (Mobile Agent). Nhìn chung hệ thống đã được tác giả khảo sát một số mô hình hỗ trợ E-Learning và đề xuất một mô hình mới cho hệ thống của mình. Tuy nhiên, tác giả chỉ xây dựng mô hình dựa trên những suy luận cảm tính mà chưa theo một phương pháp luận chính qui nào hỗ trợ cho việc thiết kế hệ thống MAS. Hệ

thống được cài đặt thử nghiệm bằng ngôn ngữ lập trình C#, một ngôn ngữ mạnh về hỗ trợ lập trình hướng đối tượng.



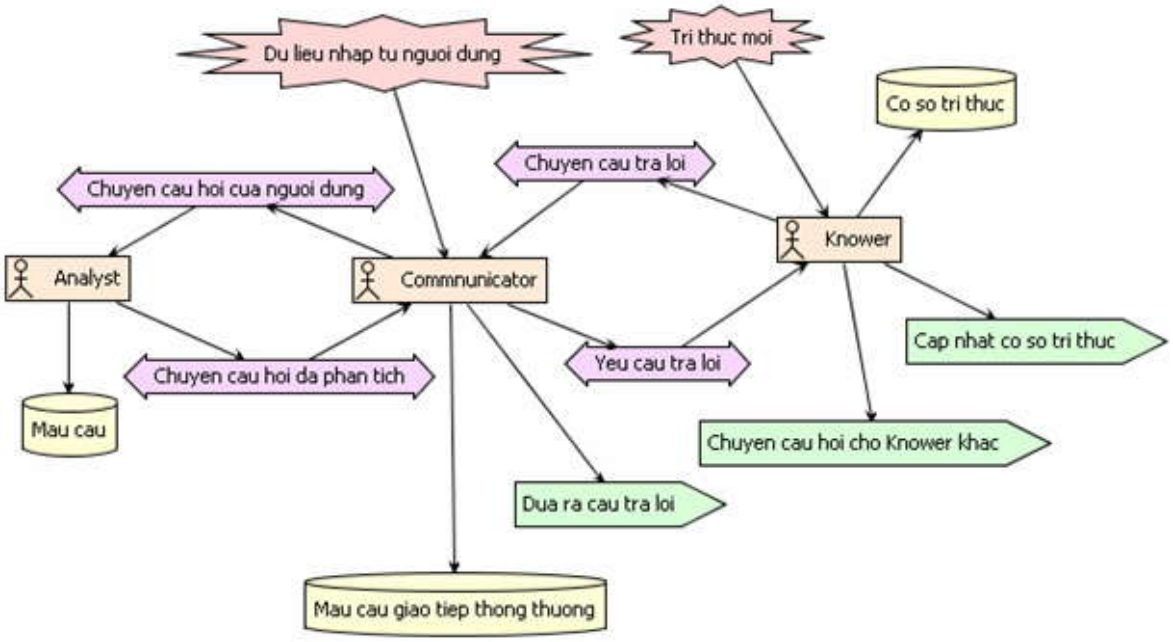
Hình 2.12: Mô hình hệ thống MAS ứng dụng trong E-Learning

3. **Một mô hình MAS cho hệ truy vấn kiến thức** của ThS Phạm Thi Vương [04]. Đề tài đã đề xuất một mô hình Multi-agent nhằm hỗ trợ truy vấn kiến thức Hình học phẳng và Hình học giải tích 2 chiều. Hệ thống này gồm có 3 loại agent: Agent giao tiếp người dùng, Agent xử lý truy vấn và Agent xử lý tri thức. Tác giả đã có những tìm hiểu sâu về tri thức lĩnh vực từ đó đề xuất một mô hình MAS cho hệ thống của mình. Tuy nhiên, tác giả thiết kế và xây dựng mô hình mà không áp dụng một phương pháp luận nào để hỗ trợ cho việc thiết kế hệ thống MAS, hệ thống MAS còn khá đơn giản chỉ dừng lại ở cấp độ truy vấn kiến thức. Hệ thống được cài đặt thử nghiệm bằng ngôn ngữ Maple và Java.



Hình 2.13: Mô hình hệ thống MAS cho hệ tra cứu kiến thức

4. **Một mô hình hệ thống Multi-Agent và Ontology cho hệ truy vấn cơ sở tri thức COKB** của ThS Nguyễn Thị Trâm Anh [02]. Đề tài đã đề xuất một mô hình Multi-agent nhằm hỗ trợ hỏi-đáp kiến thức Toán đại số tuyến tính. Hệ thống này gồm có 3 agent: Agent giao tiếp người dùng (Communicator Agent), Agent trung gian (Analyst Agent) và Agent tri thức (Knower Agent). Tác giả đã có những tìm hiểu sâu về tri thức lĩnh vực từ đó đề xuất một mô hình MAS cho hệ thống của mình. So với ba tác giả trên, ở đây tác giả đã xây dựng mô hình dựa trên theo phương pháp luận Prometheus hỗ trợ cho việc thiết kế hệ thống MAS, nhưng cũng mới chỉ ở cấp độ tìm hiểu, hệ thống MAS còn khá đơn giản chưa cho thấy hết được các đặc tính nỗi bật của các loại agent và MAS. Hệ thống được cài đặt thử nghiệm bằng ngôn ngữ lập trình JDE và Java, một ngôn ngữ hỗ trợ lập trình hướng agent.



Hình 2.14: Mô hình hệ thống MAS cho hệ truy vấn cơ sở tri thức COKB