

LỜI GIỚI THIỆU

Trong những năm gần đây, sự phát triển mạnh mẽ của các công nghệ truyền thông và internet đã ảnh hưởng sâu rộng đến mọi mặt của cuộc sống từ kinh tế, khoa học đến văn hóa và xã hội. Rõ ràng sự phát triển của phần cứng đóng vai trò rất quan trọng trong quá trình tiến hóa này nhưng yếu tố then chốt đã ảnh hưởng mạnh mẽ đến xã hội tri thức ngày nay chính là bản thân phần mềm. Khi mà mạng máy tính và Internet trở thành phổ biến thì việc xử lý thông tin phân tán, chia sẻ và tích hợp thông tin thông qua đường truyền giữa các máy với những cơ sở dữ liệu có những khuôn dạng khác nhau càng ngày càng trở nên phổ biến. Điều này dẫn đến một thách thức mới đối với giới phát triển phần mềm khi phải đổi mới với những yêu cầu thực tế của các hệ phần mềm phức tạp, mở và phân tán.

Những nghiên cứu và công nghiệp phát triển phần mềm trong những cuối năm 80 và đầu thập niên 90 xoay quanh cách tiếp cận hướng đối tượng tiến hóa từ phương pháp luận phần mềm cấu trúc truyền thống. Phương pháp hướng đối tượng có ưu điểm so với phương pháp cấu trúc là khả năng sử dụng lại mã nguồn, dễ đọc mã nguồn và xử lý lỗi. Ý tưởng cơ bản của nó là xem hệ phần mềm như tập hợp các thực thể tương tác gọi là “đối tượng” trong đó mỗi đối tượng được xác định bởi ba yếu tố: *Định danh, trạng thái và hành vi*¹. Như vậy, phát triển phần mềm dựa trên cách tiếp cận này có nghĩa là tiến hành xây dựng mô hình của hệ thống cần được phát triển (cả trong các pha phân tích và thiết kế) dựa trên khái niệm đối tượng và những khái niệm liên quan như thành viên, phương thức, quan hệ... Ngôn ngữ UML đã được sử dụng rộng rãi để mô hình các hệ phần mềm này dưới dạng use case, biểu đồ lớp, biểu đồ tương tác...

Tuy nhiên, cách tiếp cận hướng đối tượng tỏ ra không đáp ứng được nhu cầu phát triển các hệ phần mềm mở, phân tán, phức tạp như quản lý mạng viễn thông, thương mại điện tử, trợ giúp văn phòng, tìm kiếm/ lọc thông tin... Là một phát triển tiếp theo của hướng đối tượng, cách tiếp cận hướng agent được xem là công nghệ hứa hẹn cho phát triển các hệ phần mềm phức tạp này. Ý tưởng cơ bản của hệ đa agent là xem hệ phần mềm như một cấu trúc xã hội bao gồm các agent có khả năng *tự chủ* cùng với các tương tác “có tính chất tri thức” hay “mang ngữ nghĩa” giữa chúng.

Giống như đối tượng, các agent cũng có định danh, trạng thái và hành vi nhưng những khái niệm này được mô tả một cách tinh tế hơn:

¹ *Trạng thái* được mô tả bởi bộ giá trị của các biến, *hành vi* được mô tả theo các phương thức có thể được thực hiện từ trong chính đối tượng đó hay gọi từ những đối tượng khác. Tương tác giữa các đối tượng được mô tả theo một số các quan hệ khác nhau có được giữa chúng.

- Trạng thái có thể bao gồm tri thức, lòng tin, đích cần phải thoả mãn, các trách nhiệm được gán cho từng agent;
- Hành vi là những vai trò mà agent có thể đảm nhiệm, những công việc cần phải tiến hành, các sự kiện cần phải được quan sát...

Công nghệ phần mềm hướng agent đã thu hút nhiều quan tâm nghiên cứu vì nó được xem là cách tiếp cận tiến hoá từ công nghệ phần mềm hướng đối tượng và công nghệ tri thức. Nó đã tỏ ra có nhiều hứa hẹn cho phát triển các hệ phần mềm trong môi trường phân tán và mở. Thập niên 90 đã chứng kiến sự nở rộ của nhiều ứng dụng và thử nghiệm thành công trong các lĩnh vực khác nhau như viễn thông, quản lý không lưu, các dịch vụ trên Internet...Những năm 2000, các nghiên cứu về agent tập trung vào xây dựng các phương pháp luận phát triển phần mềm bao gồm xây dựng quy trình, công cụ cùng các kỹ thuật phân tích và thiết kế hệ đa agent.

Như vậy, công nghệ agent đã được nghiên cứu và phát triển mạnh mẽ trên thế giới và đã được áp dụng trong nhiều lĩnh vực khác nhau. Tuy nhiên, những nghiên cứu ở trong nước về agent mới chỉ ở giai đoạn bắt đầu và theo hiểu biết của chúng tôi nghiên cứu về công nghệ phần mềm hướng agent chưa được quan tâm nhiều.

Nhằm đáp ứng nhu cầu nghiên cứu và phát triển các hệ phần mềm đa agent, đề tài đã tập trung xem xét quy trình phát triển và các kỹ thuật cho các bước trong các pha phân tích và thiết kế hệ này. Thuật ngữ *quy trình trong đề tài này được hiểu là bao gồm các bước trong các pha phân tích và thiết kế phần mềm*. Mặc dù có nhiều phương pháp luận và công cụ phát triển hệ đa agent đã được xây dựng nhưng phương pháp luận MaSE (chi tiết sẽ được trình bày trong Chương 2) đã được lựa chọn vì hai lý do sau đây:

- a. Phương pháp luận MaSE kế thừa từ phương pháp luận hướng đối tượng và do đó sẽ dễ dàng hơn cho những người phát triển phần mềm đã quen thuộc với cách tiếp cận hướng đối tượng phổ biến hiện nay;
- b. Phương pháp luận này có công cụ đi kèm agentTool có thể hỗ trợ phát triển từ phân tích, thiết kế đến sinh mã nguồn. Hơn nữa, trong khi các công cụ khác tách biệt khâu phát triển ontology thì agentTool đã tích hợp khâu này vào trong quá trình phát triển và do đó đã tạo điều kiện dễ dàng cho người phát triển hơn vì không phải sử dụng các công cụ khác để phát triển ontology và hơn nữa nó lại được sinh ra trong quá trình sinh mã nguồn hệ thống.

Nội dung nghiên cứu của đề tài bao gồm:

- Nghiên cứu các đặc trưng của agent và hệ đa agent; một số vấn đề cơ bản liên quan đến phát triển hệ phần mềm agent bao gồm xây dựng ontology và tương tác;

- Nghiên cứu các bước trong phân tích và thiết kế hệ đa agent và sử dụng công cụ agentTool trong các bước này.
- Nghiên cứu áp dụng phương pháp luận MaSE trong phân tích và thiết kế hệ dịch vụ thương mại điện tử **TraNeS**.

Tài liệu được tổ chức thành 2 phần bao gồm 7 chương như sau:

Phần 1 Cơ sở phát triển hệ đa agent

Chương 1: Hệ đa agent

Chương này trình bày một cách tổng quan về agent, hệ đa agent và các cách tiếp cận trong nghiên cứu xây dựng các phương pháp luận phát triển hệ đa agent. Nội dung của chương này tập trung xem xét các cách tiếp cận khi xây dựng các phương pháp luận phát triển hệ phần mềm đa agent.

Chương 2: Tương tác trong hệ đa agent

Chương này trước hết trình bày tổng quan vấn đề tương tác trong hệ đa agent bao gồm các dạng tương tác, tương tác với agent trung gian và thương lượng trong hệ đa agent. Một mô hình thương lượng song phương dựa trên ràng buộc mờ sẽ được trình bày nhằm cơ sở cho phát triển hệ dịch vụ du lịch sẽ được đề cập đến trong các chương tiếp theo.

Chương 3: Ontology trong hệ đa agent

Ontology là khái niệm quan trọng nhằm biểu diễn ngữ nghĩa của thông tin được truyền đi giữa các agent trong quá trình tương tác. Nội dung của chương này tập trung xem xét khái niệm ontology và vai trò của nó trong tương tác giữa các agent. Phần kỹ thuật xây dựng ontology trong hệ đa agent sẽ được đề cập trong Chương 4.

Chương 4: Quy trình phát triển hệ phần mềm hướng agent

Nội dung chương này tập trung trình bày quy trình phát triển hệ phần mềm hướng agent dựa trên phương pháp luận MaSE cùng với các bước tương ứng trong quá trình phát triển dựa trên công cụ agentTool. Các bước phát triển ontology của hệ thống cũng được gói gọn trong chương này. Một áp dụng của quy trình này cho phát triển hệ dịch vụ thương lượng tự động sẽ được mô tả chi tiết trong các chương còn lại.

Phần 2: Áp dụng phát triển hệ dịch vụ du lịch

Chương 5: Phân tích hệ dịch vụ

Chương này nhằm trình bày chi tiết một áp dụng của quy trình phát triển hệ đa agent cho phân tích hệ dịch vụ du lịch **TraNeS**. Nội dung các bước phân tích này được trình bày gắn liền với công cụ phát triển agentTool.

Chương 6: Thiết kế hệ dịch vụ

Nội dung chính của chương này là trình bày một áp dụng của quy trình phát triển hệ đa agent trong thiết kế cho thiết kế hệ dịch vụ du lịch **TraNeS**.

Chương 7: Cài đặt và tích hợp hệ dịch vụ

Nội dung của chương này trình bày các vấn đề liên quan đến cài đặt và tích hợp hệ dịch vụ thương lượng.

Chương 8: Giới thiệu hệ TraNeS

Nội dung nhằm điểm qua một số đặc trưng và cách tiến hành cài đặt của hệ dịch vụ du lịch TraNeS đã được phát triển trong các Chương 5, 6 và 7.

Kết luận

Phần cuối cùng là kết luận và một số vấn đề cần quan tâm nghiên cứu hơn nữa trong phát triển các ứng dụng.

Tài liệu này được viết với giả thiết rằng người đọc đã quen thuộc với phương pháp luận phát triển phần mềm hướng đối tượng. Do đó, nhiều khái niệm không được nhắc lại như use case, biểu đồ tương tác, biểu đồ trạng thái. Mặc dù nhóm đề tài đã có nhiều nỗ lực để hoàn thiện tài liệu nhưng không thể tránh khỏi những thiếu sót. Rất mong nhận được những ý kiến đóng góp và chỉ bảo của các đồng nghiệp.

MỤC LỤC

LỜI GIỚI THIỆU	1
MỤC LỤC	5
PHẦN 1 CƠ SỞ PHÁT TRIỂN HỆ ĐA AGENT	8
CHƯƠNG 1 HỆ ĐA AGENT	9
1.1 Agent	10
1.1.1 Khái niệm agent	10
1.1.2 Agent và đối tượng	12
1.2 Hệ đa agent	13
1.2.1 Khái niệm hệ đa agent	13
1.2.2 Môi trường tính toán thích hợp cho hệ đa agent	14
1.2.3 Các ứng dụng của hệ đa agent	15
1.3 Các phương pháp luận phát triển hệ đa agent	16
1.3.1 Các cách tiếp cận phát triển hệ đa agent	17
1.3.1.1 Các phương pháp mô hình yêu cầu	18
1.3.1.2 Các cách tiếp cận trong phân tích thiết kế hệ thống đa agent	19
1.4 Phương pháp luận Gaia	22
1.4.1 Giới thiệu chung	22
1.4.2 Pha phân tích	23
1.4.3 Pha thiết kế	23
1.5 Phương pháp luận MAS-CommonKADS	24
1.5.1 Giới thiệu chung	24
1.5.2 Pha khái niệm hoá	25
1.5.3 Pha phân tích	25
1.5.4 Pha thiết kế	27
1.4 Kết luận	28
CHƯƠNG 2 TƯƠNG TÁC TRONG HỆ ĐA AGENT	29
2.1 Tổng quan về tương tác trong hệ đa agent	30
2.1.1 Ngôn ngữ truyền thông giữa các agent	31
2.1.2 Các mô hình tương tác	33
2.1.3 Tương tác với agent trung gian	37
2.2 Thương lượng trong hệ đa agent	40
2.3 Mô hình thương lượng song phương	42
2.3.1 Cơ sở toán học cho thương lượng song phương	42
2.3.2 Chiến lược thương lượng cho agent bán	45
2.3.3 Chiến lược thương lượng cho agent mua	47
2.4 Kết luận	52
CHƯƠNG 3 ONTOLOGY TRONG HỆ ĐA AGENT	53
3.1 Khái niệm Ontology	54
3.1.1 Khái niệm	54
3.1.2 Ontology và cơ sở tri thức	55

3.1.3 Phân loại ontology	56
3.1.4 Vai trò của ontology trong tương tác giữa các agent.....	57
3.2 Biểu diễn ontology	58
3.2.1 Biểu diễn ontology theo kiểu hình thức.....	59
3.2.2 Biểu diễn ontology theo kiểu không hình thức.....	65
3.3 Phương pháp luận xây dựng ontology tổng quát	67
3.4 Kết luận	69
CHƯƠNG 4 QUY TRÌNH PHÁT TRIỂN HỆ PHẦN MỀM HƯỚNG AGENT ...	70
4.1 Đặc điểm của phương pháp luận MaSE	71
4.2 Quy trình phát triển hệ phần mềm hướng agent	72
4.2.1 Khái quát các bước phát triển.....	72
4.2.2 Phân tích	73
4.2.3 Thiết kế.....	93
4.3 Kết luận	103
PHẦN 2 ÁP DỤNG PHÁT TRIỂN HỆ DỊCH VỤ DU LỊCH	104
CHƯƠNG 5 PHÂN TÍCH HỆ DỊCH VỤ	105
5.1 Mô hình sở thích người sử dụng	106
5.1.1 Bài toán dịch vụ du lịch.....	106
5.1.2 Mô hình sở thích người sử dụng	107
a. Ràng buộc các thuộc tính.....	107
b. Ràng buộc giữa các mặt hàng.....	109
5.2 Phân tích hệ thống	110
5.2.1 Xác định đích của hệ thống	110
5.2.2 Xây dựng các use case	112
5.2.3 Xây dựng ontology.....	114
5.2.4 Hoàn thiện các role.....	116
5.3 Kết luận	120
CHƯƠNG 6 THIẾT KẾ HỆ DỊCH VỤ	121
6.1 Một số vấn đề về thiết kế hệ đa agent	122
6.2 Thiết kế hệ đa agent	122
6.2.1 Xây dựng các lớp agent	122
6.2.2 Xây dựng các phiên hội thoại	124
6.2.3 Hoàn thiện các agent.....	129
6.2.4 Triển khai hệ thống	133
6.3 Kết luận	133
CHƯƠNG 7 CÀI ĐẶT VÀ TÍCH HỢP HỆ THỐNG	134
7.1 Vài nét về agentMom	135
7.2 Mô hình tích hợp hệ thống	137
7.2.1 UserAgent.....	137
7.2.2 HotelAgent và TrainAgent.....	137
7.2.3 MatchAgent	138
7.2.4 Hoạt động của hệ thống	139

7.3 Cài đặt các lớp agent.....	140
7.3.1 UserAgent.....	140
7.3.2 HotelAgent.....	146
7.3.3 TrainAgent.....	150
7.3.4 MatchAgent	153
7.4 Kết luận	156
CHƯƠNG 8 GIỚI THIỆU HỆ TRAMES.....	157
8.1 Đặc trưng của Hệ TraNeS	158
8.2 Các mô hình hoạt động của hệ TraNeS	158
8.3 Các nhóm chức năng của Hệ TraNeS.....	162
8.4 Cài đặt Hệ TraNeS.....	179
8.5 Bài học từ phát triển hệ TraNeS	179
8.6 Kết luận	180
KẾT LUẬN	183
TÀI LIỆU THAM KHẢO	184

PHẦN 1

CƠ SỞ PHÁT TRIỂN HỆ ĐA AGENT

CHƯƠNG 1

HỆ ĐA AGENT

- Agent
- Hệ đa agent
- Một số vấn đề cơ bản khi nghiên cứu và phát triển hệ đa agent
- Các phương pháp luận phát triển hệ đa agent

Nội dung chương này trược hết trình bày một cách khái quát về agent, hệ đa agent, môi trường thích hợp cho ứng dụng hệ đa agent, và ba vấn đề cơ bản cần quan tâm khi nghiên cứu và phát triển hệ đa agent là ontology, tương tác và phương pháp luận phát triển hệ đa agent. Phần tiếp theo của chương tập trung trình bày tổng quan các phương pháp luận trong phát triển hệ đa agent nhằm làm cơ sở cho xây dựng quy trình phát triển hệ đa agent sẽ trình bày trong Chương 4.

1.1 Agent

1.1.1 Khái niệm agent

Trong những năm gần đây, sự phát triển của các công nghệ Internet đã dẫn tới việc áp dụng rộng rãi của công nghệ thông tin vào nhiều lĩnh vực khác nhau của cuộc sống như tìm kiếm truy xuất thông tin, quản lý mạng viễn thông, thương mại điện tử, hỗ trợ ra quyết định, giải trí,... Sự đa dạng của các lĩnh vực áp dụng khiến cho việc phát triển phần mềm càng ngày càng trở nên phức tạp và sự phức tạp này thể hiện ở một số đặc điểm sau đây:

- *Khối lượng công việc cần xử lý ngày càng lớn:* Các hệ phần mềm ngày nay phải xử lý một khối lượng dữ liệu rất lớn hoặc thao tác trên một số lượng lớn các nguồn thông tin. Bên cạnh đó, quá trình phát triển hệ thống thường xuyên phải đối mặt với các bài toán có độ phức tạp lớn (nhiều bài toán thuộc dạng NP đầy đủ) đặc biệt là với các ứng dụng thương mại điện tử hay điều khiển phức tạp.
- *Yêu cầu về tính chính xác ngày càng cao:* Yêu cầu này xuất hiện cùng với sự ra đời của các hệ thống đòi hỏi độ chính xác và thời gian thực như các hệ điều khiển không lưu, điều khiển thiết bị viễn thông, các bài toán quản lý lưu lượng, quản lý tiến trình công việc... Đặc biệt, việc xây dựng và triển khai các ứng dụng thời gian thực đang ngày càng trở thành nhu cầu tất yếu và là một trong những hướng phát triển của công nghệ thông tin và truyền thông nói chung.
- *Yêu cầu về tính mở và phân tán:* Yêu cầu này xuất hiện cùng với sự phát triển của các hệ thống mạng, đặc biệt là hệ thống trên mạng Internet. Ngày nay, hầu hết các hệ thống thông tin đều gắn bó chặt chẽ với môi trường mạng. Internet đã trở thành một phần quan trọng trong cuộc sống con người và do đó các phần mềm cũng cần phải đáp ứng ngày càng tốt hơn các nhu cầu của con người như tìm kiếm thông tin, hỗ trợ người mua và người bán đưa ra quyết định,... và phải có tính mở, tức là có thể được cập nhật, thay đổi hay bổ sung các dịch vụ vào hệ thống.
- *Yêu cầu tính độc lập giữa các thành phần trong hệ thống:* Yêu cầu này thể hiện rõ nhất trong các hệ ra quyết định và các hệ thương mại điện tử. Các hệ thống này yêu cầu các thành phần phải hoạt động độc lập và chủ động tương tác với các thành phần khác nhằm hướng tới đích riêng của mình. Nhất là trong các hệ thống mà mục đích riêng của các thành phần là không thống nhất với nhau, thậm chí tranh chấp nhau thì yêu cầu này càng trở nên quan trọng.

Những yêu cầu này đã dẫn đến sự nghiên cứu và phát triển mạnh mẽ của công nghệ phần mềm trong những năm gần đây. Cách tiếp cận dựa trên cấu trúc chiêm ưu thế vào những năm 70-80 đã dần dần bị thay thế bởi phương pháp luận hướng đối tượng với

tập kí hiệu chuẩn UML mà ngày nay đã trở thành phổ biến trong phân tích, thiết kế và xây dựng các hệ phần mềm.

Tuy nhiên, khi hệ thống thông tin càng ngày càng phức tạp thì người ta cũng nhận ra sự hạn chế của cách tiếp cận này. Nguyên nhân cơ bản là do tính thụ động của các đối tượng nghĩa là *các đối tượng chỉ thực sự hoạt động khi nhận được một thông điệp từ đối tượng khác*. Với các hệ thống có yêu cầu về tính phân tán như các hệ thương lượng trong thương mại điện tử, các hệ quản lý mạng viễn thông... thì tương tác thụ động như vậy tỏ ra không phù hợp. Các thành phần phần mềm trong hệ thống như vậy phải phục vụ cho những dịch vụ khác nhau và do đó cần phải chủ động theo mục đích của riêng mình đồng thời phải tương tác với các thành phần khác để chia sẻ tài nguyên, yêu cầu hỗ trợ tính toán...

Ta thử xét một ví dụ sau đây. Trong hệ dịch vụ du lịch, người sử dụng thường có nhiều yêu cầu khác nhau cho các gói du lịch của mình như vé máy bay, vé tàu, chỗ ở... Do đó, thành phần phần mềm thay mặt người dùng cần phải tương tác, thương lượng với nhiều dịch vụ khác một cách tự động và sau đó tích hợp kết quả gửi lại cho người sử dụng, mỗi thành phần như thế được gọi là một *agent*. Mặc dù cho đến nay chưa có một định nghĩa thống nhất về khái niệm này, nhiều nghiên cứu cho rằng:

Agent là một hệ tính toán hoàn chỉnh hay chương trình được đặt trong một môi trường nhất định, có khả năng hoạt động một cách tự chủ và mềm dẻo trong môi trường đó nhằm đạt được mục đích đã thiết kế.

Các đặc trưng cơ bản của agent sau đây đã được nhiều người thừa nhận ([9], [13], [14], [20]):

- *Tính tự chủ (autonomy)*: Mỗi agent có một trạng thái riêng, độc lập với các agent khác (*tự chủ ở trạng thái bên trong*) đồng thời nó có thể tự quyết định các hành động của mình (*tự chủ về hành động*). Tự chủ ở trạng thái bên trong thể hiện ở chỗ: mỗi agent chứa một trạng thái nào đó của riêng nó, các agent khác không truy nhập được vào các trạng thái này. Còn tính tự chủ về hành động thể hiện ở chỗ agent có thể tự quyết định các hành động của mình (có thể là một hành động đơn hoặc là một chuỗi các hành động) dựa trên trạng thái hiện thời mà không có sự can thiệp của con người hay các agent khác. Tính tự chủ chính là đặc trưng quan trọng nhất của agent.
- *Khả năng phản ứng (reactivity)*: Tính phản ứng là khả năng agent có thể nhận biết được môi trường (qua bộ phận cảm nhận nào đó) và dựa qua nhận biết đó, agent đáp ứng kịp thời những thay đổi xảy ra trong môi trường. Tính phản ứng thể hiện rõ nhất ở các agent hoạt động trên các môi trường có tính mở và hay thay đổi như Internet, môi trường mạng phân tán, môi trường vật lý, ... Phản ứng của mỗi một agent đối với môi trường bên ngoài đều hướng tới việc thực hiện mục tiêu (đích) của agent đó.

- *Tính chủ động (pro-activeness)*: Khi có sự thay đổi của môi trường, agent không chỉ phản ứng một cách đơn giản mà còn xác định một chuỗi hành động cần thực hiện, bản thân mỗi agent sẽ chủ động trong việc khởi động và thực hiện chuỗi hành động này.
- *Khả năng xã hội (social ability)*: Các agent không chỉ hướng tới đích riêng của mình mà còn có khả năng tương tác với các agent khác trong hệ thống để hướng tới đích chung của hệ thống. Các hoạt động tương tác này rất đa dạng bao gồm phối hợp, thương lượng, cạnh tranh...

1.1.2 Agent và đối tượng

Để hiểu rõ hơn khái niệm agent, chúng ta hãy so sánh agent và đối tượng. Trong phương pháp hướng đối tượng, các đối tượng được định nghĩa là các thực thể tính toán đóng gói bao gồm các trạng thái, các hành động hay phương thức trong trạng thái đó và các đối tượng liên lạc với nhau thông qua việc gửi các thông điệp (message).

Xét theo quan điểm hệ thống, có thể xem mỗi agent cũng là một đối tượng nhưng ở mức trừu tượng cao hơn. Vì vậy, khái niệm đối tượng được sử dụng trong phần này là để chỉ các đối tượng chuẩn (standard object) trong phương pháp hướng đối tượng. Với định nghĩa agent đã được đề cập ở Mục 1.1.1 thì các đối tượng và các agent có các điểm khác biệt sau:

- Agent có tính tự chủ cao hơn đối tượng. Điều này thể hiện ở chỗ các agent có thể tự quyết định hành động của mình mà không phải là thực hiện hành động theo yêu cầu của agent khác. Ngược lại, các đối tượng chỉ thực sự hoạt động khi nhận được lời gọi hàm từ các đối tượng khác. Trong các ngôn ngữ lập trình hướng đối tượng như Java chẳng hạn, các đối tượng có các thành phần riêng kiểu *private* không thể truy nhập từ các đối tượng khác. Tuy nhiên, các đối tượng lại không thể tự chủ về mặt *hành vi* của mình, một đối tượng với thành phần *public*, có thể được truy nhập bởi các đối tượng khác và chỉ khi một đối tượng khác sử dụng các lời gọi tới các thành phần *public* của đối tượng này thì nó mới thực sự hoạt động.
- Agent có tính hướng đích, mỗi agent có một đích riêng và đích của các agent trong một hệ thống có thể thống nhất hay không tương thích với nhau. Trong khi đó các đối tượng không có mục đích riêng, chúng cùng chia sẻ mục đích chung của cả hệ thống. Do đó, các agent thường phải thương lượng với nhau trong quá trình tương tác.
- Agent có các hành vi linh hoạt dựa trên các đặc trưng như tính chủ động, khả năng phản ứng và khả năng xã hội đã trình bày ở trên. Còn các đối tượng thì không có các kiểu hành vi này.

- Mỗi agent có một hoặc nhiều luồng điều khiển (thread) riêng. Trong hệ hướng đối tượng cũng có điều khiển theo kiểu luồng (thread) nhưng không yêu cầu mỗi đối tượng là có thread riêng mà ngược lại có thể có nhiều đối tượng chung một thread. Bản chất của sự khác nhau này cũng là đặc trưng quan trọng về mức độ tự chủ của agent so với đối tượng.

1.2 Hệ đa agent

1.2.1 Khái niệm hệ đa agent

Khả năng của mỗi agent thể hiện ở năng lực giải quyết vấn đề của riêng agent đó. Trong một hệ thống cụ thể, thông thường tài nguyên dành cho mỗi agent là hạn chế do đó khả năng hành động của mỗi agent cũng là hạn chế. Mỗi agent chỉ tập trung giải quyết một vấn đề tại một vị trí cụ thể nào đó chứ không thể giải quyết được hết các vấn đề đặt ra cho hệ thống. Trong các hệ phân tán phức tạp, hệ đa agent được xem là hệ xử lý thông tin có nhiều hứa hẹn.

Có thể hiểu hệ đa agent là *một tập các agent cùng hoạt động trong một hệ thống, mỗi agent có thể có đích khác nhau nhưng toàn bộ hệ agent cùng hướng tới mục đích chung thông qua tương tác*.

Quá trình tính toán và xử lý thông tin trong hệ đa agent được xem là có nhiều ưu điểm hơn so với các hệ thống khác như hệ đối tượng [20]:

- *Khả năng tính toán hiệu quả*: Hệ đa agent cung cấp khả năng tính toán hiệu quả hơn nhờ quá trình tính toán được phân chia cho các agent khác nhau và khả năng phối hợp cùng xử lý của nhiều agent.
- *Độ tin cậy cao*: Do có nhiều agent cùng tham gia giải bài toán và các agent có cơ chế trao đổi, kiểm tra kết quả nên độ tin cậy tính toán trong hệ đa agent được cho là cao hơn.
- *Khả năng mở rộng*: Hệ đa agent là hệ mở vì có thể có thêm các agent mới hoặc bớt đi các agent khi các agent hoàn thành nhiệm vụ. Khả năng này phù hợp với tính mở của yêu cầu các hệ phần mềm hiện nay.
- *Tính mạnh mẽ*: Hệ đa agent có thể xử lý được các bài toán ra quyết định phức tạp hoặc các bài toán dựa trên thông tin không chắc chắn như các bài toán thương lượng trong thương mại điện tử, các bài toán điều khiển tự động...
- *Khả năng bảo trì*: Do hệ đa agent gồm nhiều agent, mỗi agent là một module có tính tự chủ cao nên hệ đa agent là hệ dễ bảo trì.
- *Khả năng phản ứng*: Hệ đa agent kế thừa khả năng phản ứng của các agent đơn nên khi nhận biết được một thay đổi của môi trường thì các agent trong hệ thống sẽ phối hợp với nhau để đưa ra hành động tương ứng với thay đổi đó.

- *Tính linh hoạt:* Các agent trong hệ đa agent có khả năng khác nhau có thể tương tác với nhau để cùng giải quyết một vấn đề chung. Một agent trong hệ thường không phải chờ agent khác mà chủ động tương tác để tìm ra thông tin cần thiết để giải quyết vấn đề đặt ra cho riêng mình.
- *Khả năng sử dụng lại:* Hệ đa agent có khả năng sử dụng lại vì mỗi agent có khả năng riêng và có thể dùng lại cho nhiều ứng dụng khác nhau.

1.2.2 Môi trường tính toán thích hợp cho hệ đa agent

Hệ đa agent tỏ ra có nhiều ưu điểm trong việc giải quyết các bài toán phức tạp hiện nay dựa trên tính năng của từng agent và sự phối hợp giữa các agent. Các môi trường và dạng bài toán thích hợp cho hệ đa agent bao gồm [20]:

- Hệ đa agent có thể giải quyết một bài toán vượt quá khả năng của một agent đơn. Trong hệ sử dụng một agent đơn, hệ thống thường tập trung tất cả các xử lý cho một agent duy nhất. Nhưng do tài nguyên của một agent đơn là hạn chế (chẳng hạn như đường truyền hay bộ nhớ...) nên các hệ thống như vậy thường có những “nút cốt chai”, gây nghẽn mạng hoặc tình trạng bế tắc khi có quá nhiều yêu cầu tập trung về một agent. Hệ đa agent giải quyết vấn đề này thông qua cơ chế phối hợp, cộng tác giữa các agent.
- Hệ đa agent cung cấp phương pháp giải quyết các bài toán phân tán trong đó có nhiều thành phần tự chủ cùng hoạt động trong một xã hội agent (society of agent) và cùng tuân theo các luật xã hội (social law) trong xã hội đó. Các thành phần trong các hệ đa agent không phải luôn luôn có cùng chung một đích. Để thực hiện các đích riêng của mình, các agent có thể tương tác với các agent khác theo các giao thức tương tác như: phối hợp, cộng tác, hoặc trong trường hợp mục đích riêng mâu thuẫn nhau thì có thể là cạnh tranh, thương lượng.
- Hệ đa agent cung cấp phương pháp giải quyết các bài toán mà thông tin được thu thập từ nhiều nguồn khác nhau. Các nguồn thông tin này có bản chất phân tán trong một hệ thống rất lớn. Ví dụ cụ thể cho dạng bài toán này chính là bài toán truy xuất thông tin trên internet, các bài toán tích hợp và xử lý thông tin ...
- Một dạng bài toán khác rất phù hợp với hệ đa agent là bài toán tích hợp hệ chuyên gia. Mỗi hệ chuyên gia là một hệ thống tập trung giải quyết một vấn đề xác định dựa trên tri thức của chuyên gia về vấn đề đó. Thực tế có thể có nhiều hệ chuyên gia tuy hướng tới giải quyết cùng một vấn đề nhưng lại phân tán ở những “vị trí” rất xa nhau. Hệ đa agent cung cấp khả năng phối hợp giữa các hệ chuyên gia này để nâng cao khả năng xử lý của hệ thống.
- Cách tiếp cận hướng agent phù hợp khi hệ thống yêu cầu các kiểu liên lạc phức tạp, đa dạng. Ví dụ như các hệ thống sử dụng cơ chế liên lạc của con người hoặc tương tác giữa các thực thể *hỗn tạp*.

- Cách tiếp cận hướng agent phù hợp các hệ thống cần phải thực hiện tốt trong tình huống không thể mô tả hành vi của các thành phần trong hệ thống một cách rõ ràng theo dạng từng trường hợp (case-by-case).
- Cách tiếp cận hướng agent cũng tỏ ra phù hợp trong tình huống có sự thương lượng, cộng tác hay cạnh tranh giữa các thực thể khác nhau trong hệ thống. Ví dụ như các tác vụ khác nhau với các đích xung đột nhau có thể cần phải thực hiện đồng thời, khi đó sẽ có các quá trình cạnh tranh hay thương lượng giữa các thành phần.
- Cách tiếp cận hướng agent cũng phù hợp khi hệ thống phải hành động một cách tự chủ để thay mặt người dùng, ví dụ như trong các quá trình thương lượng giữa các thành phần bên trong hệ thống để đạt tới những mục đích khác nhau.

1.2.3 Các ứng dụng của hệ đa agent

Trong những năm gần đây, các hệ đa agent đã ngày càng trở nên phổ biến và được áp dụng trong nhiều hệ thống khác nhau. Theo Jennings et al. [23], các ứng dụng của hệ đa agent có thể chia thành các nhóm sau:

Các hệ ứng dụng trong công nghiệp

Các ứng dụng hệ đa agent trong công nghiệp là những ứng dụng đầu tiên của lĩnh vực nghiên cứu này. Hiện nay, agent đã được áp dụng rộng rãi trong các dạng hệ thống như:

- *Hệ sản xuất*: trong các hệ đa agent ứng dụng trong sản xuất, công việc sẽ được phân chia thành các nhóm công việc hoặc các công việc nhỏ hơn vào giao cho các agent thực hiện. Các agent cần có cơ chế lập kế hoạch và phối hợp (tương tác) lẫn nhau để hoàn thành công việc được giao.
- *Hệ thống điều khiển tiến trình*: Các hệ điều khiển tiến trình có vai trò rất lớn trong công nghiệp. Hệ đa agent trong hệ thống này sẽ được xem như một bộ điều khiển tiến trình (process controller) với tính tự chủ và linh hoạt để điều khiển hoạt động của tiến trình đó.
- *Hệ thống viễn thông*: các hệ thống viễn thông thường là các hệ thống lớn, phân tán, yêu cầu quá trình giám sát và quản lý theo thời gian thực (như quản lý mạng viễn thông, giám sát hoạt động của thiết bị). Các ứng dụng này rất phù hợp với hệ đa agent.
- Ngoài ra, hệ đa agent cũng đã được áp dụng trong các *hệ thống quản lý không lưu* và *quản lý lưu lượng giao thông*. Đây là các hệ quản lý yêu cầu tính thời gian thực cao, các thành phần trong hệ thống phải có tính tự chủ và linh hoạt trong xử lý tình huống.

Các ứng dụng trong thương mại

Trong thời gian gần đây, hệ đa agent ngày càng được áp dụng nhiều trong thương mại điện tử. Với các hệ ứng dụng này, việc trao đổi mua bán diễn ra thuận lợi và hiệu quả

hơn cho cả người bán, người mua cũng như các nhà sản xuất. Các hệ ứng dụng agent trong thương mại bao gồm:

- *Hệ quản lý thông tin:* Các hệ thống này thực hiện việc lọc, tách và thu thập thông tin cần thiết dùng trong thương mại. Hệ thống thường xuyên phải xử lý một khối lượng thông tin rất lớn nhằm cung cấp cho người dùng những thông tin cần thiết.
- *Các hệ thương mại điện tử:* Các agent trong các hệ thương mại điện tử sẽ đại diện cho người bán, người mua cũng như người môi giới trong các giao dịch điện tử. Các agent này tự trao đổi với nhau thông qua các chiến lược thương lượng của mình. Đây chính là xu hướng phát triển của thương mại điện tử hiện nay.
- *Các ứng dụng quản lý tiền kinh doanh:* Quản lý tiền kinh doanh nhằm giúp cho người quản lý ra quyết định trong một tình huống cụ thể hoặc thực hiện một công việc cụ thể nào đó với sự hỗ trợ của tất cả các thành viên trong đơn vị. Các agent sẽ đại diện cho các thành viên thực hiện các vai trò xác định. Công việc sẽ được phân chia cho các nhóm thành viên dựa trên cơ chế tương tác giữa các agent.

Các ứng dụng giải trí

Các hệ đa agent cũng đã được sử dụng để xây dựng các ứng dụng giải trí như các trò chơi điện tử và các ứng dụng khác như nhà hát hay rạp chiếu phim tương tác (Interactive Theatre and Cinema).

Các ứng dụng trong y tế

- *Ứng dụng giám sát bệnh nhân:* Các agent hoạt động như các chuyên gia để theo dõi hoặc chẩn đoán bệnh cho người bệnh. Việc chẩn đoán bệnh được thực hiện thông qua cơ chế lập luận của agent.
- *Các ứng dụng chăm sóc sức khoẻ (Health Care):* Hệ đa agent được thiết kế để thực hiện các nhiệm vụ của mạng lưới y tế cộng đồng.

1.3 Các phương pháp luận phát triển hệ đa agent

Trong nghiên cứu phát triển hệ đa agent ba vấn đề sau đây đã được quan tâm xem xét:

Tương tác giữa các agent

Trong các hệ đa agent, mỗi agent là một thành phần chủ động và hướng tới đích riêng nên cần phải trao đổi thông tin-trí thức với nhau và thương lượng với nhau khi cần thiết. Quá trình trao đổi, tương tác giữa các agent không thể giống với dạng tương tác thụ động (qua các lời gọi hàm) trong hệ hướng đối tượng. Vấn đề tương tác giữa các agent được xem là then chốt trong phát triển hệ đa agent và sẽ được trình bày chi tiết trong Chương 2.

Ontology

các agent tương tác với nhau thông qua việc gửi và nhận các thông điệp truyền thông giống như các đối tượng. Tuy nhiên, các thông điệp này không biểu diễn các

lời gọi hàm đơn giản mà cần phải biểu diễn được thông tin và tri thức trao đổi giữa các agent. Mỗi agent trong hệ đa agent có thể có một miền quan tâm riêng, để các agent hiểu nhau trong quá trình trao đổi cần một cấu trúc ngữ nghĩa gọi là ontology. Ontology và xây dựng ontology để biểu diễn thông tin và tri thức trong hệ thống sẽ được trình bày chi tiết trong Chương 3.

Phương pháp luận phát triển hệ đa agent

Xây dựng hệ đa agent cần phải theo quan điểm nào và các bước nào? Do xuất phát điểm của các nhà nghiên cứu khác nhau (hoặc là từ cộng đồng trí tuệ nhân tạo hoặc là từ giới nghiên cứu hướng đối tượng) nên có những quan điểm khác nhau về phát triển hệ đa agent. Điều này dẫn tới việc có nhiều phương pháp luận phát triển hệ đa agent khác nhau; hoặc dựa trên phương pháp luận truyền thống hướng đối tượng hoặc dựa trên công nghệ tri thức hoặc cả hai. Tổng quan các phương pháp luận phát triển hệ đa agent sẽ được đề cập tiếp theo trong phần này.

1.3.1 Các cách tiếp cận phát triển hệ đa agent

Nhu cầu phát triển các ứng dụng phần mềm dựa trên công nghệ agent trong những năm gần đây đã dẫn đến sự ra đời của nhiều phương pháp luận dựa vào ba cách tiếp cận sau đây: (1) *cách tiếp cận dựa trên agent và công nghệ agent*, (2) *cách tiếp cận phát triển từ hướng đối tượng* và (3) *cách tiếp cận dựa trên công nghệ tri thức*. Nội dung phần còn lại của chương này trước hết trình bày các cách tiếp cận xây dựng hệ đa agent và một số phương pháp luận đã được phát triển dựa trên các cách tiếp cận đó. Sau đó, chúng tôi sẽ tập trung trình bày khái quát hai phương pháp luận đại diện cho hai cách tiếp cận (1) và (2) là Gaia và MAS-CommonKADS.

Quá trình phát triển một hệ thống phần mềm thông thường bao gồm các pha chính sau đây:

- *Xác định yêu cầu*
- *Phân tích*
- *Thiết kế*
- *Cài đặt và tích hợp*

Trong các pha trên thì phân tích và thiết kế hệ thống được xem là các pha chính thể hiện quan điểm của người phát triển về hệ thống của mình. Phần tiếp theo sẽ trình bày một số phương pháp mô hình yêu cầu của người sử dụng, các cách tiếp cận trong phân tích và thiết kế hệ đa agent; phần cài đặt và tích hợp cho một áp dụng cụ thể sẽ được trình bày trong chương 7 của tài liệu này.

1.3.1.1 Các phương pháp mô hình yêu cầu

Việc xác định yêu cầu hệ thống là công việc đầu tiên cần thực hiện khi xây dựng một hệ đa agent. Phương pháp mô hình yêu cầu nhằm mô hình và phân tích các yêu cầu chức năng cũng như các yêu cầu phi chức năng của hệ cần phát triển. Tập các yêu cầu cần phải có khả năng biểu diễn đầy đủ và chính xác các ràng buộc của hệ thống trên thực tế; nó đóng vai trò quan trọng trong việc giám sát các thay đổi có thể có trong toàn bộ quá trình phân tích thiết kế sau này.

Theo Weiss [37], có hai hướng khác nhau trong việc mô hình hoá yêu cầu hệ thống: (1) *mô hình yêu cầu hướng agent* và (2) *mô hình yêu cầu hướng đích (goal)*. Chúng ta sẽ lần lượt xem xét hai kỹ thuật này.

Phương pháp mô hình hoá yêu cầu hướng agent

Mô hình yêu cầu hướng agent dựa trên hai đặc điểm:

- Mỗi agent là một phần mềm cụ thể có khả năng hoạt động tự chủ và hướng tới đích riêng của mình.
- Agent được xây dựng dựa trên việc mô hình quá trình nhận thức và lập luận của con người.

Như vậy, mỗi yêu cầu cần phải xác định được: hệ đa agent nhằm mục đích giải quyết những vấn đề gì; cần phải mô hình hoá những tri thức nào và nhất là mô hình hoá cơ chế lập luận của agent dựa trên những cơ sở nào. Các phương pháp đại diện cho kiểu mô hình này gồm:

*i**: đây là cơ sở để mô hình hoá tập yêu cầu thông qua các thuộc tính ý định (*intention*) như mục đích (*goal*) hay thoả thuận (*commitment*). Các yêu cầu sẽ được nhóm theo các thuộc tính ý định này và thông qua quá trình mô hình hoá để chuyển sang giai đoạn phân tích (và đặc tả) yêu cầu.

ALBERT (Agent-oriented Language for Bulding and Eliciting Real-Time requirement) và ALBERTII. Đây là các kỹ thuật xác định yêu cầu tập trung vào khái niệm agent. ALBERT đồng thời cũng là một ngôn ngữ để đặc tả yêu cầu theo kiểu hình thức.

Hai kiểu mô hình hoá *i** và ALBERT có thể áp dụng riêng biệt hoặc kết hợp với nhau. Mô hình yêu cầu trong phương pháp luận Tropos ([2]) chính là ví dụ của việc kết hợp hai kiểu này.

Phương pháp mô hình hoá yêu cầu hướng đích

Đích (goal) là khái niệm để trả lời câu hỏi *hệ thống hướng tới cái gì*. Phương pháp mô hình hóa yêu cầu hướng đích sẽ xác định các yêu cầu chức năng và phi chức năng theo phương pháp sau:

- Yêu cầu chức năng sẽ được thu thập và nhóm theo trả lời của các câu hỏi *cái gì*. Tức là cần phải trả lời các câu hỏi như *hệ thống hướng tới thực hiện các công việc gì? các thành phần nào cần có trong hệ thống? các ràng buộc của hệ thống là gì?*.
- Các yêu cầu phi chức năng nhằm mô hình hoá các câu hỏi *tại sao, thế nào* như *hệ thống thực hiện nhiệm vụ của mình như thế nào, tại sao cần hệ đa agent?*

Các ví dụ của phương pháp mô hình hoá yêu cầu hướng đích:

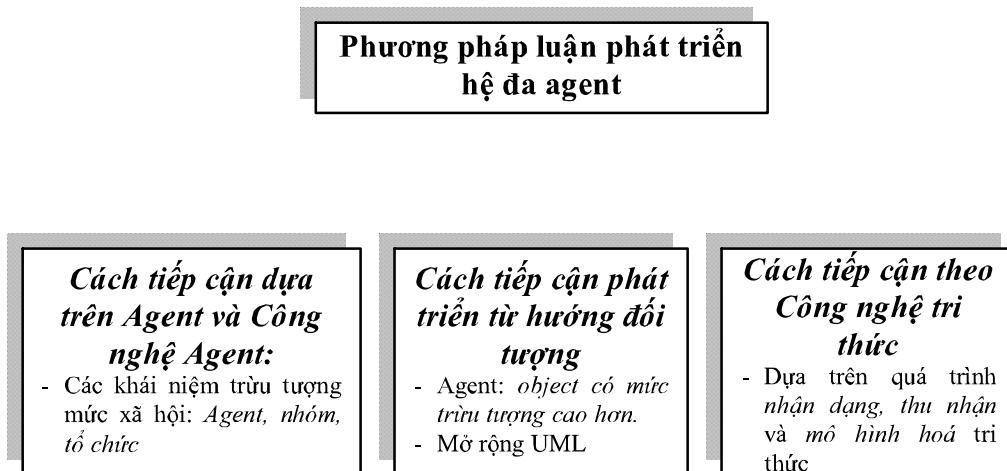
KAOS (Knowledge Acquistion in automated Specification): là một khuôn mẫu chung để mô hình hoá yêu cầu dựa trên tính hướng đích của agent.

NFR (Non-Functional Requirement): Tập trung vào việc đặc tả và lập luận liên quan đến các yêu cầu phi chức năng. NFR cũng xây dựng phương pháp để biểu diễn các yêu cầu softgoal, ví dụ như các goal không định nghĩa một cách rõ ràng được, các goal mô tả yêu cầu thoả mãn ràng buộc...

1.3.1.2 Các cách tiếp cận trong phân tích thiết kế hệ thống đa agent

Theo [37] các phương pháp luận phân tích và thiết kế hệ đa agent đều được xây dựng dựa trên một trong ba cách tiếp cận (Xem Hình 1.1):

- Dựa trên agent và công nghệ agent
- Phát triển từ phương pháp hướng đối tượng
- Dựa trên công nghệ tri thức.



Hình 1.1: Các cách tiếp cận xây dựng phương pháp luận đa agent

Cách tiếp cận theo agent và công nghệ agent

Các nghiên cứu xây dựng phương pháp luận phát triển hệ đa agent theo các đặc trưng của agent và công nghệ agent xuất phát từ các nhận định sau:

- Agent có những đặc trưng riêng như tính tự chủ, tính chủ động và khả năng phản ứng. Các đặc trưng này là khác hoàn toàn khác với đối tượng và tạo cho agent khả năng tương tác chủ động và khả năng suy luận mà các đối tượng không có. Vì vậy, không thể mô hình hóa agent bằng cách sử dụng các phương pháp luận dành cho hướng đối tượng.
- Agent hoạt động như một xã hội với các luật chi phối riêng nên có thể xem xét agent từ khía cạnh xã hội và sử dụng các khái niệm trừu tượng mức xã hội để mô hình hóa các agent trong hệ thống.

Các khái niệm trừu tượng mức xã hội được sử dụng để mô hình hóa agent theo cách tiếp cận này bao gồm *agent, nhóm (group), tổ chức (organization)* ... Mức trừu tượng xã hội được xem như là một mức trừu tượng cao hơn so với mức đối tượng thông thường và phù hợp với việc biểu diễn các agent vì xã hội các agent bao gồm các thành phần có tính phản ứng, linh hoạt và tương tác chủ động.

Dựa trên các khái niệm trừu tượng mức xã hội mà phương pháp luận này định nghĩa các bước, các pha của toàn bộ quá trình phát triển hệ thống. Thông thường, với cách tiếp cận này, quá trình phát triển hệ thống được phân thành hai pha tách biệt: pha phân tích và pha thiết kế. Pha phân tích tương ứng với mô

hình tổ chức (organization model) còn pha thiết kế ứng với mô hình agent (agent model). Tiêu biểu cho các phương pháp luận thuộc loại này là các phương pháp luận Gaia, SODA, AALAADIN... Phương pháp luận Gaia sẽ được giới thiệu chi tiết hơn trong phần 1.4.

Cách tiếp cận phát triển từ phương pháp hướng đối tượng.

Khác với cách tiếp cận trên, cách tiếp cận phát triển từ phương pháp hướng đối tượng dựa trên quan điểm cho rằng các kỹ thuật đã được phát triển và được áp dụng rộng rãi cho công nghệ phần mềm hướng đối tượng có thể được mở rộng cho các phần mềm hướng agent.

Cách tiếp cận này xem mỗi agent cũng là một đối tượng nhưng ở mức trừu tượng cao hơn. Các công cụ sử dụng để biểu diễn đối tượng và phân tích thiết kế hệ thống hướng đối tượng như UML hoàn toàn có thể mở rộng để biểu diễn các đặc trưng riêng của agent. Các nghiên cứu xây dựng phương pháp luận phát triển từ phương pháp luận hướng đối tượng tập trung vào các công việc sau:

- Phát triển các công cụ dựa trên nền tảng sẵn có của hướng đối tượng để áp dụng cho hệ đa agent, ví dụ như mở rộng UML thành AUML (Agent UML).
- Bổ sung các bước, pha mang tính đặc trưng của agent và công nghệ agent như đích (goal), vai trò (role) và ontology.

Tiêu biểu cho các phương pháp luận này là các phương pháp luận MaSE, MASSIVE, KGR...

Cách tiếp cận dựa trên công nghệ tri thức

Dựa trên quan điểm cho rằng quá trình mô hình tri thức riêng của mỗi agent và sự trao đổi tri thức giữa các agent trong hệ thống chính là cơ sở của mọi hoạt động của hệ thống, các nghiên cứu theo cách tiếp cận này sử dụng các khái niệm và quá trình trong công nghệ tri thức để xây dựng nên phương pháp luận cho hệ đa agent.

Các phương pháp luận này tập trung vào các quá trình *thu thập tri thức* (*Knowledge acquisition*), *mô hình tri thức* (*Knowledge modelling*) và lập luận được sử dụng bởi các thành phần agent trong hệ thống phần mềm. Tiêu biểu cho cách tiếp cận này là các phương pháp luận như CoMoMAS, MAS-ComonKADS.

Như vậy, có ba cách tiếp cận trong phân tích và thiết kế hệ đa agent. Mỗi cách tiếp cận lại có những ưu điểm riêng. Để hiểu những điểm khác biệt của các phương pháp luận theo các cách tiếp cận này, phần còn lại của chương này dành trình bày về phương

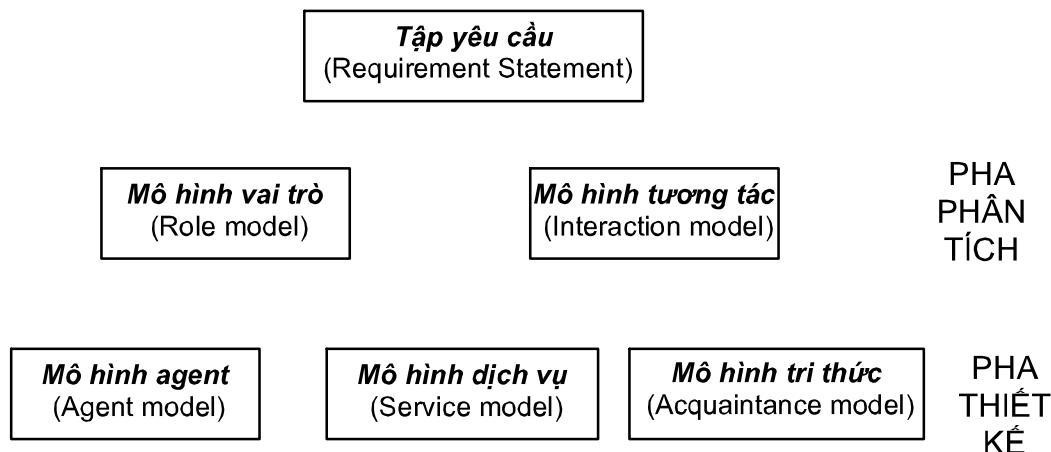
pháp luận Gaia (theo cách tiếp cận agent và công nghệ agent) và phương pháp luận MAS-CommonKADS (theo cách tiếp cận dựa trên công nghệ tri thức). Riêng với MaSE, đại diện cho cách tiếp cận phát triển từ hướng đối tượng, là phương pháp luận được lựa chọn để xây dựng quy trình phát triển hệ phần mềm hướng agent sẽ được trình bày chi tiết trong chương 4 của tài liệu.

1.4 Phương pháp luận Gaia

1.4.1 Giới thiệu chung

Gaia [38] là một phương pháp luận được xây dựng theo cách tiếp cận agent và công nghệ agent. Phương pháp luận này phân biệt hoàn toàn hai pha phân tích và thiết kế đồng thời kết hợp các mô hình khác nhau trong các pha đó. Gaia tập trung xem xét hệ đa agent trên khía cạnh tổ chức với các khái niệm như: vai trò (role), tương tác (interaction), và thu thập tri thức (acquaintance). Phương pháp luận Gaia tiến hành phát triển hệ thống dựa trên cả hai quan điểm: *quan điểm vi mô* với *mức xã hội (social level)* và *quan điểm vi mô* với *mức agent (agent level)*.

Các bước phân tích thiết kế trong Gaia là sự chuyển tiếp giữa các mô hình trong quá trình xây dựng hệ thống. Mỗi quan hệ giữa các bước này được biểu diễn như trong Hình 1.2 :



Hình 1.2: Các bước phát triển của Gaia

Gaia cũng *mượn* một số khái niệm và ký hiệu từ phân tích thiết kế hệ hướng đối tượng. Tuy nhiên, Gaia không áp dụng các phương pháp này để phát triển hệ đa agent mà cung cấp một tập các khái niệm mang tính đặc trưng của agent giúp cho người phát triển có thể hiểu và mô hình hóa được các hệ thống phức tạp. Gaia giúp người phát triển hệ thống xem quá trình phát triển hệ agent như là một quá trình thiết kế tổ chức (*organisational design*). Các khái niệm chính trong Gaia được phân thành hai nhóm

chính là trùu tượng và cụ thể. Các khái niệm trùu tượng bao gồm *role*, *quyền hạn* (*permission*), *trách nhiệm* (*responsibilities*), *giao thức* (*protocol*), *hoạt động* (*activities*) ... còn các khái niệm cụ thể là *kiểu agent* (*agent type*), *dịch vụ* (*service*) hay *thu thập tri thức* (*acquaintance*).

Ta sẽ lần lượt xem xét chi tiết các pha phân tích và thiết kế trong Gaia.

1.4.2 Pha phân tích

Mục đích của pha phân tích là phát triển một hiểu biết cơ bản về hệ thống và cấu trúc của nó (không xét chi tiết đến việc cài đặt hệ thống). Hiểu biết này được biểu diễn theo cấu trúc tổ chức của hệ thống. Chúng ta có thể xem cấu trúc tổ chức của một hệ thống bao gồm một tập các role, các mối quan hệ giữa các role và tương tác giữa các role đó.

Khái niệm role trong Gaia bao gồm bốn thuộc tính: *trách nhiệm* (*responsibilities*), *quyền hạn* (*permissions*), *hoạt động* (*activities*) và *giao thức* (*protocols*). *Trách nhiệm* xác định các yêu cầu chức năng ứng với role đó. *Quyền hạn* của một role giúp nhận ra các trách nhiệm thông qua việc xác định các tài nguyên sẵn sàng cho role đó. Các *hoạt động* của role là các tính toán liên kết với role mà agent có thể có thể tiến hành mà không cần tương tác với agent khác. Mỗi role cũng được định nghĩa bởi một số *giao thức*, mỗi giao thức xác định role đó phải tương tác với các role khác như thế nào.

Dựa trên yêu cầu, pha phân tích sẽ tiến hành xây dựng *mô hình role* và *mô hình tương tác* giữa các agent trong hệ thống.

Mô hình role nhằm xác định các role của hệ thống và được biểu diễn một cách trùu tượng theo hai thuộc tính:

- Các quyền hạn ứng với role đó
- Các trách nhiệm của role đó

Mô hình tương tác xác định sự phụ thuộc và các mối quan hệ giữa các role trong tổ chức đa agent. Mỗi giao thức tương tác kết nối 2 role sẽ được định nghĩa cụ thể trong mô hình này. Tương tác trong mô hình này được xem xét theo bản chất tự nhiên và mục đích của tương tác chứ không phải dựa trên việc gửi và nhận các thông điệp.

1.4.3 Pha thiết kế

Mục đích của pha thiết kế trong Gaia là chuyển các mô hình trong pha phân tích sang mức trùu tượng thấp hơn (tương tự như trong thiết kế hướng đối tượng) nhằm hướng tới việc cài đặt hệ thống. Nói cách khác, pha thiết kế nhằm trả lời câu hỏi làm thế nào để xã hội các agent thông qua tương tác đạt được đích của hệ thống, và riêng với các agent thì cần những gì để đạt được điều đó.

Pha thiết kế trong Gaia bao gồm ba mô hình: *mô hình agent* xác định các kiểu agent trong hệ thống và các agent cụ thể trong hệ thống sẽ là các thể hiện của các kiểu agent này; *mô hình dịch vụ* xác định các dịch vụ chính cần có để thực hiện các role trong kiểu agent tương ứng; còn *mô hình thu thập tri thức* nhằm biểu diễn liên lạc giữa các agent khác nhau. Ta sẽ xem xét cụ thể từng mô hình:

Mô hình agent

Mục đích là xác định các kiểu agent có thể sử dụng trong quá trình phát triển hệ thống. Một kiểu agent là một tập các role. Trên thực tế có thể tồn tại các tương ứng 1-1 giữa các role và các kiểu agent. Tuy nhiên, điều này là không bắt buộc, người thiết kế có thể chọn một số role một số role có liên quan vào trong cùng một kiểu agent cho phù hợp.

Với mỗi kiểu agent, người thiết kế có thể xác định số agent thể hiện có thể có thông qua một chú giải ghi bên dưới tên lớp agent như sau. Nếu số chú giải có dạng là một số n thì sẽ có chính xác n thể hiện, nếu là $m..n$ thì sẽ có ít nhất là m và nhiều nhất là n thể hiện, nếu là * có thể có 0 đến nhiều thể hiện, còn nếu là + thì có nghĩa là có từ 1 đến nhiều thể hiện.

Mô hình dịch vụ

Xác định các dịch vụ kết hợp với mỗi role và đặc tả các thuộc tính chính của dịch vụ đó. Có thể xem mỗi dịch vụ là một *chức năng* của agent. Mỗi dịch vụ sẽ có các thuộc tính như *đầu vào*, *đầu ra*, *các điều kiện đầu vào (pre-conditions)* và *các điều kiện đầu ra (post-conditions)*. Các điều kiện đầu vào và đầu ra xác định các ràng buộc cho dịch vụ đó.

Mô hình thu thập tri thức

Xác định các liên lạc giữa các kiểu agent đã tồn tại. Mô hình không xác định thông điệp gì được gửi đi và khi nào gửi đi, mà chỉ đơn giản là chỉ ra các phiên liên lạc đang tồn tại. Mục đích của mô hình này là nhằm phát hiện ra các “*nút cỏ chai*” có thể có trong thời gian hoạt động của hệ thống. Một mô hình có dạng như một đồ thị với các node là các kiểu agent.

1.5 Phương pháp luận MAS-CommonKADS

1.5.1 Giới thiệu chung

MAS-CommonKADS [21] là phương pháp luận mở rộng từ CommonKADS để áp dụng cho phân tích và thiết kế hệ đa agent. Đây là một phương pháp luận theo cách tiếp cận dựa trên công nghệ tri thức. Phương pháp luận này thêm vào CommonKADS các kỹ thuật của phân tích thiết kế hướng đối tượng như *kỹ thuật mô hình hóa đối tượng (Object Modelling Technique: OMT)*, *thiết kế hướng trách nhiệm (Responsibility Driving Design: RDD)* và các kỹ thuật để mô tả giao thức agent như

ngôn ngữ mô tả và đặc tả (Specification and Description Language: SDL) và biểu đồ thông điệp tuần tự (Message Sequence Charts). MAS-CommonKADS đưa ra các mô hình sau:

- *Mô hình agent (Agent model: AM):* biểu diễn các đặc tính của agent như: khả năng lập luận, các dịch vụ, các nhóm agent và các sơ đồ phân cấp các agent.
- *Mô hình task (Task model: TM):* biểu diễn các task mà agent có thể tiến hành, các đích (goal), các thành phần...
- *Mô hình chuyên gia (Expert model: EM):* biểu diễn tri thức cần cho agent để đạt được đích của nó đặt ra.
- *Mô hình tổ chức (Organisation model: OM):* biểu diễn cấu trúc tổ chức xã hội trong hệ đa agent cần xây dựng.
- *Mô hình phối hợp (CoM):* biểu diễn các phiên hội thoại giữa các agent, các tương tác và các giao thức tương tác tương ứng.
- *Mô hình liên lạc (Communication model: CM):* biểu diễn chi tiết các tương tác giữa con người với hệ thống đa agent.
- *Mô hình thiết kế (Design model: DM):* tập hợp từ các mô hình trước đó để thiết kế hệ thống. Bao gồm ba mô hình nhỏ: thiết kế mạng, thiết kế agent và thiết kế nền.

Các bước phát triển hệ đa agent theo phương pháp luận MAS-CommonKADS bao gồm 3 pha chính là: pha khái niệm hoá, pha phân tích và pha thiết kế. Chúng ta sẽ lần lượt xem xét các pha này.

1.5.2 Pha khái niệm hoá

Nhiệm vụ chính của pha này là mô tả bài toán thông qua các biểu đồ use cases và scenario có dạng tương tự như trong phân tích hướng đối tượng. Các biểu đồ này giúp chúng ta hiểu rõ các yêu cầu phi hình thức của hệ thống và kiểm tra hệ thống về sau. Mỗi thành phần tham gia vào các use case và biểu đồ tuần tự được gọi là một role.

1.5.3 Pha phân tích

Mục đích của pha phân tích là đặc tả các yêu cầu của hệ thống vào trong các mô hình được chỉ ra trong phần 1.5.1 trừ mô hình thiết kế. Quá trình xây dựng các mô hình này được mô tả ngắn gọn như sau.

Xây dựng Mô hình agent

Các agent sẽ được xác định theo các chiến lược sau:

- Phân tích các thành phần tham gia vào các use case đã xác định trong pha khái niệm hoá để nhóm các role tương tự nhau vào trong một agent nhằm đơn giản hoá các trao đổi, liên lạc trong hệ thống.
- Phân tích các câu trong mô tả bài toán để tìm ra các chủ ngữ là các đối tượng chủ động và gán chúng thành các agent. Các hành động của các chủ ngữ này sẽ trở thành đích của agent nếu như hành động đó do agent khởi tạo, và sẽ trở thành dịch vụ của nó nếu như hành động đó được thực hiện theo yêu cầu bên ngoài.
- Sử dụng chiến lược heuristics: các agent sẽ được xác định thông qua các khái niệm về khoảng cách như: sự phân tán tri thức, phân tán về địa lý, phân tán về mặt logic hoặc phân tán về tổ chức.
- Sử dụng mô hình chuyên gia để xác định các chức năng và các yêu cầu về năng lực xử lý tri thức, thông qua đó để định nghĩa các agent.
- Sử dụng thẻ CRC và kỹ thuật RDD giống như trong thiết kế hướng đối tượng.

Xây dựng Mô hình Task

Các task được xác định theo cách tiếp cận từ trên xuống và được biểu diễn theo dạng hình cây. Mô tả một task bao gồm tên của task, đầu vào và đầu ra của task, cấu trúc của task, các điều kiện thực hiện, ...

Mô hình này giúp người phát triển hệ thống dễ dàng quản lý các thay đổi trong các bước còn lại của pha phân tích cũng như trong pha thiết kế.

Xây dựng Mô hình phối hợp

Mô hình phối hợp được xây dựng theo 2 pha:

- Định nghĩa các kênh truyền thông giữa các agent và xây dựng một bản mẫu (prototype).
- Phân tích các tương tác và chỉ ra các tương tác phức tạp cùng với các giao thức phối hợp tương ứng.

Mỗi pha bao gồm một loạt các bước nhỏ, trong đó sử dụng các ký hiệu theo MSC (Message Sequence Chart), SDL (Specification and Description Language) hoặc biểu diễn thông qua các cấu trúc tri thức (tham khảo thêm trong [21]).

Xây dựng Mô hình tri thức

Người phát triển sẽ sử dụng mô hình chuyên gia để mô hình hoá khả năng lập luận của các agent trong việc thực hiện các task và hướng tới đích của nó. Ở đây chỉ có một số mô hình chuyên gia được xây dựng nhằm mô hình hoá khả năng lập luận trong miền tri thức và mô hình hoá khả năng suy luận của agent.

Mô hình chuyên gia biểu diễn các tri thức ứng dụng (bao gồm tri thức miền, tri thức lập luận, và tri thức tác vụ) cùng với tri thức giải quyết bài toán. Quá trình xây dựng mô hình tri thức sẽ lần lượt xem xét và biểu diễn các tri thức ứng dụng và các tri thức giải quyết bài toán.

Xây dựng Mô hình tổ chức

Tương tự như trong CommonKADS, mô hình tổ chức trong MAS-CommonKADS cũng được xây dựng để biểu diễn các mối quan hệ tĩnh hoặc được cấu trúc hoá giữa các agent (trong khi mô hình phối hợp biểu diễn các mối quan hệ động).

Mô hình tổ chức cũng sử dụng tập ký hiệu theo OMT nhưng với ngữ nghĩa thay đổi cho phù hợp với hệ đa agent.

1.5.4 Pha thiết kế

Dựa trên tập các agent cùng với các mô hình đã được xây dựng trong pha phân tích, pha thiết kế sẽ tiến hành xây dựng mô hình thiết kế. Pha này bao gồm các bước nhỏ sau:

- Thiết kế kiến trúc mạng agent (agent network design)
- Thiết kế agent (agent design)
- Thiết kế nền (platform design)

Ta sẽ lần lượt xem xét từng bước này.

Thiết kế kiến trúc mạng agent

Bước này xác định cơ sở hạ tầng cho hệ đa agent bao gồm các điều kiện cần thiết về mạng, về tri thức và các sự phối hợp giữa các thành phần trong hệ thống. Các agent cũng sẽ được định nghĩa trên cơ sở hạ tầng của hệ thống phụ thuộc vào một số điều kiện cần thiết theo yêu cầu sau:

- *Các điều kiện về mạng*: bao gồm dịch vụ đặt tên agent, dịch vụ đăng ký, mức độ bảo mật, mã hoá và chứng thực, các giao thức truyền thông và ứng dụng...
- *Các điều kiện về tri thức*: bao gồm các ontology server, các bộ chuyển đổi ngôn ngữ mô tả tri thức...
- *Các điều kiện về sự phối hợp giữa các thành phần trong hệ thống*: bao gồm các giao thức phối hợp, các dịch vụ quản lý nhóm agent...

Kết quả của bước thiết kế kiến trúc mạng agent là các agent phải liên lạc được với nhau thông qua các giao thức tương tác và ontology của hệ thống.

Thiết kế Agent

Bước này xác định kiến trúc phù hợp nhất cho mỗi agent dựa theo các module: *liên lạc với người dùng (user-communication)*, *liên lạc giữa các agent (agent communication)*, *năng lực xử lý và các dịch vụ của hệ thống*.

Thiết kế nền

Bước này lựa chọn môi trường phát triển cho hệ đa agent và các phần cứng cần thiết cho hệ thống nếu cần.

1.4 Kết luận

Chương này trước hết tập trung trình bày những nét khái quát về agent, hệ đa agent, những ứng dụng của hệ đa agent. Một sự so sánh giữa agent/hệ đa agent và đối tượng/hệ đối tượng cũng đã được điểm qua nhằm làm sáng tỏ hơn các khái niệm này. Tiếp theo đó là điểm qua một số vấn đề quan trọng khi nghiên cứu và phát triển hệ agent: ontology, tương tác và các phương pháp luận. Phần quan trọng của chương này là trình bày tổng quan các phương pháp luận phát triển hệ đa agent nhằm làm cơ sở cho xây dựng quy trình phát triển hệ đa agent dựa trên phương pháp luận MaSE mà sẽ được trình bày chi tiết trong Chương 4.

CHƯƠNG 2

TƯƠNG TÁC

TRONG HỆ ĐA AGENT

- Tổng quan về tương tác trong hệ đa agent
- Thương lượng trong hệ đa agent
- Mô hình thương lượng song phương

Chương này tập trung trình bày vấn đề tương tác trong hệ đa agent. Trước hết sẽ đề cập tổng quan về vai trò tương tác trong hệ đa agent và ngôn ngữ truyền thông được sử dụng trong quá trình trao đổi thông tin và tri thức giữa các agent. Phản tiếp theo sẽ trình bày các mô hình tương tác trong hệ đa agent và đặc biệt tập trung xem xét mô hình thương lượng song phương với ràng buộc mờ nhằm làm cơ sở cho phát triển hệ dịch vụ du lịch TraNeS sẽ được trình bày trong các chương 5, 6, 7, 8.

2.1 Tổng quan về tương tác trong hệ đa agent

Hệ đa agent bao gồm nhiều agent tự chủ có thể hoạt động trên những máy tính khác nhau. Tuy nhiên, các agent thường phải trao đổi, tương tác với nhau và chính các tương tác trong hệ đa agent quyết định kiến trúc của hệ thống đó. Các dạng tương tác này phức tạp hơn rất nhiều so với các tương tác trong hệ đối tượng. Các agent tương tác với nhau bằng cách gửi thông điệp và bản chất của các thông điệp này cũng là những lời gọi hàm như trong hệ các đối tượng nhưng các lời gọi trong tương tác giữa các agent có nhiều khác biệt so với tương tác giữa các đối tượng:

- Các tham số có thể có kiểu được định nghĩa trong một cấu trúc ngữ nghĩa gọi là ontology.
- Các tham số được viết theo một dạng thông điệp truyền thông được định nghĩa bởi một ngôn ngữ truyền thông agent (như KQML hoặc FIPA-ACL).
- Nội dung của thông điệp trong tương tác đa agent có thể rất phức tạp như một chuỗi các hành động hoặc các yêu cầu...

Ngoài sự khác nhau về dạng của các đối số, tương tác trong hệ đa agent cũng khác tương tác giữa các đối tượng do bản chất khác nhau giữa đối tượng và agent. Agent là thành phần có tính tự chủ và hành động hướng đích chứ không thụ động như các đối tượng.

Với mỗi hệ agent cụ thể được xây dựng thì mục đích chung của hệ thống và mục đích riêng của từng agent có thể khác nhau, thậm chí không tương thích nhau. Ví dụ trong hệ thương mại điện tử, nếu agent mua có nhiệm vụ mua được hàng với giá càng rẻ càng tốt thì agent bán lại có mục đích là bán với giá càng cao càng tốt. Sự thống nhất và mâu thuẫn về mục đích của các agent trong hệ thống dẫn đến sự đa dạng của các mô hình tương tác trong hệ đa agent. Như vậy, tương tác trong hệ đa agent có những đặc trưng riêng khác biệt so với tương tác đa đối tượng. Vai trò của tương tác trong hệ đa agent có thể được tổng kết như sau:

- Thông qua tương tác, mỗi agent sẽ thu thập thông tin và tri thức nhằm đạt được đích (goal) riêng của mình và hướng tới đích chung của cả hệ thống.
- Tương tác tạo nên tính động cho hệ đa agent. Qua tương tác, hệ thống có thể được mở rộng hay thu hẹp một cách dễ dàng, nhất là với các hệ đa agent sử dụng agent trung gian.

- Quá trình tương tác không chỉ diễn ra giữa các agent mà còn có thể diễn ra giữa các hệ agent khác nhau. Khi đó, khả năng phối hợp giữa các hệ thống để giải quyết các vấn đề phức tạp tăng lên nhiều lần.
- Tương tác giữa các agent quyết định kiến trúc và hoạt động của hệ đa agent đó. Thông qua việc xem xét các tương tác cần có giữa các agent, người thiết kế hệ thống có thể xây dựng kiến trúc hệ thống và phân tách nhiệm vụ một cách rõ ràng cho từng agent.
- Tương tác giữa các agent giúp tích hợp các nguồn thông tin trong hệ thống. Trong hệ tích hợp thông tin, mỗi agent đại diện cho một nguồn thông tin nhất định. Các nguồn thông tin này thường là không đồng nhất, được biểu diễn theo những cách khác nhau. Thông qua tương tác, thông tin giữa các nguồn đó sẽ được tích hợp để thu được những thông tin cần thiết.

Ba vấn đề sau đây cần quan tâm xem xét khi nghiên cứu về tương tác trong hệ đa agent:

1. *Mô hình tương tác*: Tuỳ thuộc vào mục đích của hệ thống cụ thể mà người phát triển hệ thống phải lựa chọn một mô hình tương tác phù hợp, mô hình tương tác này sẽ quy định kiến trúc của hệ thống cũng như hành vi của các agent trong hệ thống.
2. *Ngôn ngữ truyền thông sử dụng trong các thông điệp*: Khi hoạt động trong cùng một hệ thống với nhau các agent phải sử dụng chung một ngôn ngữ truyền thông. Ngôn ngữ này không chỉ quy định cấu trúc thông điệp mà còn quy định các dạng thông điệp hỏi và trả lời trong các phiên hội thoại.
3. *Ontology và sử dụng ontology trong tương tác đa agent*: Mỗi agent trong hệ thống là một thành phần phần mềm riêng biệt, do đó, miền tri thức quan tâm của các agent trong một hệ thống có thể khác nhau. Để các agent có thể hiểu nhau trong quá trình trao đổi thì hệ thống phải sử dụng ontology nhằm biểu diễn các khái niệm mô tả miền và mối quan hệ giữa các khái niệm đó.

Phần 2.1.1 sẽ trình bày những nét chung về ngôn ngữ truyền thông đa agent. Các mô hình tương tác sẽ được trình bày trong phần 2.3. Ontology sẽ được trình bày trong chương 3.

2.1.1 Ngôn ngữ truyền thông giữa các agent

Các agent trao đổi với nhau thông qua các thông điệp. Khác với hệ hướng đối tượng, thông điệp trong hệ đa agent không chỉ biểu diễn các lời gọi hàm mà còn phải biểu diễn thông tin và tri thức cần trao đổi giữa các agent. Các thông điệp này được biểu diễn theo

các ngôn ngữ truyền thông agent (ACL: Agent Communication Language) nhằm mục đích:

- Định nghĩa khuôn dạng các thông điệp để trao đổi giữa các agent trong hệ thống.
- Thiết lập một giao thức trao đổi giữa các agent, bao gồm: định nghĩa các kiểu thông điệp gửi và nhận, các mô hình trao đổi thông điệp giữa các agent.

Các ngôn ngữ truyền thông đều dựa trên lý thuyết hành động - lời nói (speech-act) ([11]). Mỗi thông điệp bao giờ cũng phải mô tả đầy đủ người gửi, người nhận, mục đích của lời nói và ngữ nghĩa của lời nói. Một hành động - lời nói đầy đủ không chỉ định nghĩa cấu trúc lời nói mà còn xác định hành động liên quan đến lời nói đó. Có nhiều ngôn ngữ truyền thông đa agent đã được đưa ra trong đó hai ngôn ngữ truyền thông được sử dụng rộng rãi nhất là KQML ([11]) và FIPA-ACL ([10]).

KQML (Knowledge Query and Manipulation Language)

Đây là một ngôn ngữ được phát triển theo dự án DARPA trong khoảng thời gian đầu những năm 1990 [11]. KQML định nghĩa ngôn ngữ và giao thức cho quá trình chuyển đổi thông tin và tri thức trong hệ đa agent.

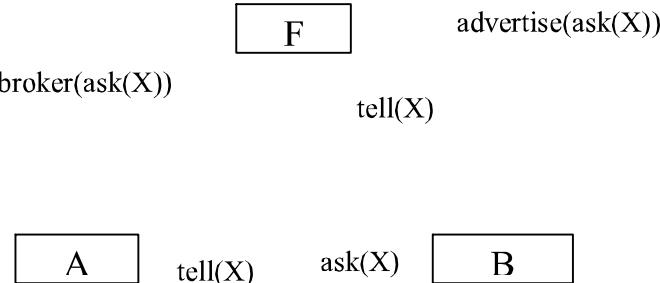
KQML định nghĩa ba mức là mức nội dung, mức thông điệp và mức truyền thông. Mỗi thông điệp KQML định nghĩa một hành động thoại, ngữ nghĩa đi kèm hành động thoại đó, giao thức và một tập các thuộc tính. Cấu trúc chung của một thông điệp KQML như sau

```
(performative-name
  : sender      A
  : receiver    B
  : content     X
  : language    L
  : ontology   N
  : reply-with  W
  : in-reply-to P)
```

Mỗi một thông điệp KQML tương ứng với một dạng tương tác trong trường performative-name. Có tới 25 dạng tương tác (performative) đã được định nghĩa bao gồm ask-one, advertise, broadcast, insert... Đồng thời, KQML cho phép mở rộng và định nghĩa thêm các dạng tương tác khác khi cần thiết.

Trường content mô tả nội dung của thông điệp. Nội dung này có thể rất phức tạp tùy thuộc vào nhu cầu trao đổi thông tin của hai agent trong phiên liên lạc đó. Agent nhận sẽ hiểu được nội dung trong trường content bằng cách tham chiếu vào

trường ontology của thông điệp mà nó nhận được. KQML cũng định nghĩa các giao thức truyền thông bao gồm cả thứ tự các thông điệp, các perormative. Ví dụ một giao thức truyền thông được định nghĩa trong KQML như Hình 2.1:



Hình 2.1: Một giao thức truyền thông trong KQML

FIPA-ACL (Foundation for Intelligent Physical Agent)

FIPA-ACL (*Foundation Intelligent Physical Agent*) là ngôn ngữ truyền thông agent được phát triển năm 1997. FIPA-ACL cũng dựa trên lý thuyết hành động - lời nói và có cấu trúc tương tự như KQML. FIPA-ACL sử dụng XML theo dạng như sau [10]:

```

<fipa-message act = " " >
  <sender>           </sender>
  <receiver>          </receiver>
  <content>           </content>
  <language>          </language>
  <ontology>          </ontology>
  <conversation-id> </conversation-id>
</fipa-message>
  
```

So với KQML, FIPA-ACL linh động hơn và có thể dễ dàng thêm vào các dạng tương tác mới. Tuy nhiên, FIPA-ACL không định nghĩa các performative theo kiểu sử dụng thành phần trung gian (facilitator) như trong KQML. Đó là các performative như broker hay advertise.

2.1.2 Các mô hình tương tác

Phân loại mô hình tương tác

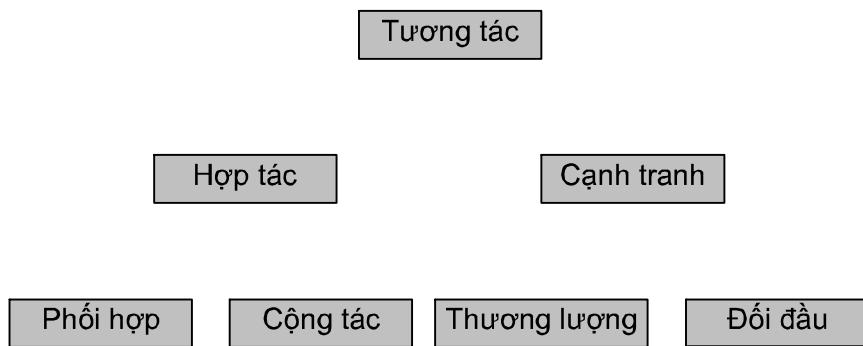
Dựa vào mục đích của các bên tham gia tương tác, có thể chia các hình thức tương tác thành hai loại chính: Hợp tác và cạnh tranh ([20]).

- **Hợp tác:** Hai bên cùng thực hiện một công việc chung (cộng tác) hoặc công việc của bên này là bước tiền đề cho bên kia (Phối hợp). Hình thức tương tác này thường xuất

hiện khi các agent có chung mục đích, nhiệm vụ hoặc cùng thực hiện một tiến trình phức tạp nhất định.

- **Cạnh tranh:** Hai bên cạnh tranh nhau về thông tin, hoặc quyền lợi (thương lượng) hoặc hoàn toàn trái ngược nhau về lợi ích (đối đầu).

Như vậy, có thể có các loại hình tương tác như sau:



Hình 2.2: Các loại hình tương tác

Phần 2.3.2 sẽ trình bày một số mô hình tương tác theo cách phân loại này.

Một số mô hình hợp tác

Các giao thức phối hợp

Trong các môi trường phân tán và hạn chế về tài nguyên cho các agent thì các agent thường phải phối hợp với nhau. Như trình bày trong phần 2.2.1, mô hình tương tác được coi là phối hợp khi công việc của agent này là tiền đề cho công việc của agent kia.

Để các agent phối hợp với nhau, các nghiên cứu cho rằng cần xây dựng kỹ thuật phân tán công việc cần thực hiện, bao gồm cả phân tán về điều khiển (control) và phân tán dữ liệu (data). Phân tán về điều khiển tức là các agent có thể tự chủ trong việc sinh ra các hành động mới và quyết định mục đích kế tiếp để hướng tới việc thực hiện công việc chung. Tri thức của hệ thống trong trường hợp này cần được biết bởi tất cả các thành phần trong hệ thống. Dựa trên tri thức này, các agent sẽ xác định hành động tiếp theo cần thực hiện trong một chuỗi công việc cần thiết để hoàn thành mục tiêu chung của hệ thống.

Liên quan đến mô hình phối hợp còn nhiều vấn đề khác như *sự thoả thuận* (*commitment*), *các quy ước* (*conventions*) và việc biểu diễn các thoả thuận hay các quy ước này ([20])

Các giao thức cộng tác

Chiến lược chung của các giao thức cộng tác là phân rã nhiệm vụ cần thực hiện của cả hệ thống và sau đó phân tán các tác vụ (task) cụ thể cho các thành viên. Các agent cùng hướng tới đích chung thông qua việc thực hiện các tác vụ mà mình được giao.

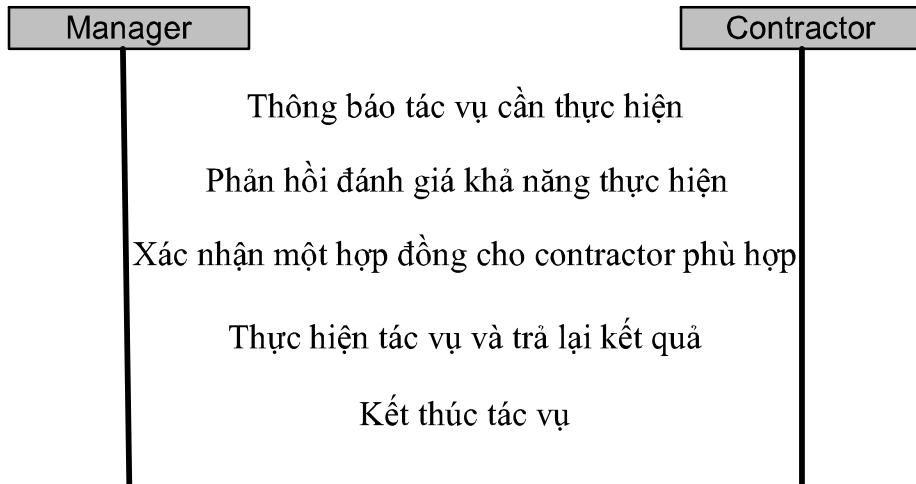
Việc phân rã các task như thế nào được thực hiện bởi người thiết kế hệ thống và tuân theo các giao thức cụ thể. Các tác vụ được phân rã phải thoả mãn các yêu cầu sau:

- Tránh xung đột tài nguyên
- Các tác vụ phải phù hợp với khả năng của agent
- Tạo ra một agent có nhiệm vụ phân phối tác vụ cho các agent khác trong hệ thống.
- Xác định các tác vụ có độ độc lập cao để giảm thiểu việc truyền thông và đồng bộ hoá kết quả.
- Định nghĩa lại các tác vụ nếu cần thiết để hoàn thành một tác vụ “khẩn cấp”.

Phản tiếp theo sẽ trình bày hai giao thức cộng tác tiêu biểu là giao thức mạng hợp đồng và giao thức bảng đen.

Giao thức Mạng hợp đồng

Giao thức mạng hợp đồng là một giao thức tương tác kiểu cộng tác. Giao thức này sẽ kết hợp các kết quả từ các agent khác nhau thông qua việc mô hình hoá hệ thống theo cơ chế hợp đồng sử dụng trong thương mại để trao đổi sản phẩm và dịch vụ. Mạng hợp đồng sẽ cung cấp giải pháp cho bài toán: *tìm một agent phù hợp cho một tác vụ cho trước*. Giả sử có một agent có một tác vụ cần xử lý. Agent này sẽ được gọi là *manager*, và agent có khả năng xử lý tác vụ này gọi là *contractor*. Tương tác giữa manager và contractor sẽ diễn ra theo biểu đồ tương tác sau:

**Hình 2.3:** Giao thức mạng hợp đồng

Manager sẽ gửi thông báo về tác vụ cần thực hiện cho tất cả các agent khác trong hệ thống. Khi nhận được yêu cầu, các agent sẽ gửi trả lại cho *manager* thông báo về khả năng thực hiện tác vụ của mình. *Manager* sẽ đánh giá và chọn ra agent phù hợp nhất để thực hiện tác vụ đó và xác nhận agent đó thành *contractor*. *Contractor* sẽ có nhiệm vụ thực hiện tác vụ và trả lại kết quả cho *manager*.

Trên lý thuyết, *manager* có thể là bất kỳ agent nào trong hệ thống khi có task cần thực hiện. Vì vậy mỗi agent có thể nhận được nhiều task, nếu agent đó là agent có khả năng xử lý cao thì rất nhiều *manager* sẽ chọn agent đó làm *contractor*. Khi đó, *contractor* sẽ lựa chọn task “hấp dẫn” nhất và mô hình mạng hợp đồng sẽ trở nên phức tạp hơn nhiều.

Giao thức Bảng đen

Phương pháp giải quyết bài toán dựa trên giao thức Bảng đen được mô tả như sau:

Giả sử có một nhóm chuyên gia hoặc agent cùng ngồi cạnh một bảng đen lớn. Các chuyên gia sẽ cộng tác với nhau để giải quyết bài toán thông qua việc sử dụng bảng đen để phát triển lời giải. Quá trình giải bài toán bắt đầu khi bài toán và dữ liệu đầu vào được viết lên bảng đen. Các chuyên gia sẽ quan sát bảng đen và cố gắng đưa ra ý kiến để phát triển lời giải của bài toán. Khi tìm ra được một thông tin/ý kiến phù hợp, chuyên gia này sẽ viết ý kiến (thông tin) đó lên bảng đen. Các chuyên gia khác sẽ sử dụng thông tin này để tiếp tục tìm ra lời giải. Quá trình cứ tiếp tục như vậy cho đến khi bài toán được giải quyết hoàn toàn.

Áp dụng giao thức bảng đen cho hệ đa agent ta sẽ có mô hình tương tác kiểu bảng đen. Khi đó, hệ thống này có các đặc điểm sau:

- Tính độc lập về giải pháp: Các chuyên gia có thể đưa ra các ý kiến độc lập với nhau.
- Tính đa dạng trong kỹ thuật giải bài toán: Thông qua bảng đen và các phương pháp biểu diễn tri thức thì một bài toán có thể có rất nhiều hướng giải quyết khác nhau.
- Cho phép biểu diễn thông tin một cách linh hoạt trên bảng đen.
- Sử dụng ngôn ngữ tương tác chung.

Một giao thức kết hợp cộng tác và cạnh tranh là thương lượng sẽ được trình bày chi tiết trong phần 3.2 và 3.3 của tài liệu. Phần tiếp theo dành để trình bày kiến trúc agent trung gian.

2.1.3 Tương tác với agent trung gian

2.1.3.1 Vai trò của agent trung gian

Trên quan điểm chú trọng đến các mô hình có sử dụng agent trung gian, ta có thể chia các mô hình tương tác trong hệ đa agent thành: (i) *tương tác với agent trung gian* và (ii) *tương tác không sử dụng agent trung gian*.

Các mô hình tương tác không sử dụng agent trung gian như mô hình bảng đen, mạng hợp đồng... có ưu điểm là đơn giản, dễ xây dựng và phù hợp với những hệ thống đa agent đóng. Các mô hình này yêu cầu các agent phải biết được khả năng của các agent khác trong hệ thống mà nó muốn tương tác. Do đó, hệ thống với các mô hình này khó mở rộng cho agent khác tham gia như trong môi trường Internet.

Khác với các mô hình bảng đen hay hợp đồng, mô hình tương tác với agent trung gian ([24], [35], [40]) sử dụng một agent trung gian *MidAgent* nhằm quản lý khả năng của các agent khác. Trong mô hình này, *Agent Yêu cầu (Requester Agent)* sẽ tương tác với *MidAgent* để biết được khả năng của các agent (*Agent Cung cấp: Provider Agent*) trong hệ thống có thể giải quyết được yêu cầu của mình. Vai trò của *MidAgent* trong những mô hình cụ thể có thể khác nhau nhưng lớp agent này đều có chung những đặc trưng sau:

- Cung cấp các phương tiện dịch vụ cơ bản để quản lý xã hội các agent;
- Phối hợp các dịch vụ được cung cấp theo một giao thức xác định nào đó;

- Đảm bảo quản lý các agent bên trong xã hội agent và quản lý việc thêm hay bớt các agent tham gia vào hệ thống.

Trong [24], lớp mô hình tương tác sử dụng agent trung gian được chia ra thành 3 mô hình nhỏ gồm mô hình tương tác kiểu *Agent Trung tâm (Mediator Agent)*, mô hình tương tác kiểu *Môi giới (MatchMaker)* và mô hình tương tác kiểu *Điều phối (Broker)*.

2.1.3.2 Các mô hình tương tác với agent trung gian

Mô hình tương tác với Agent Trung tâm (Mediator Agent)

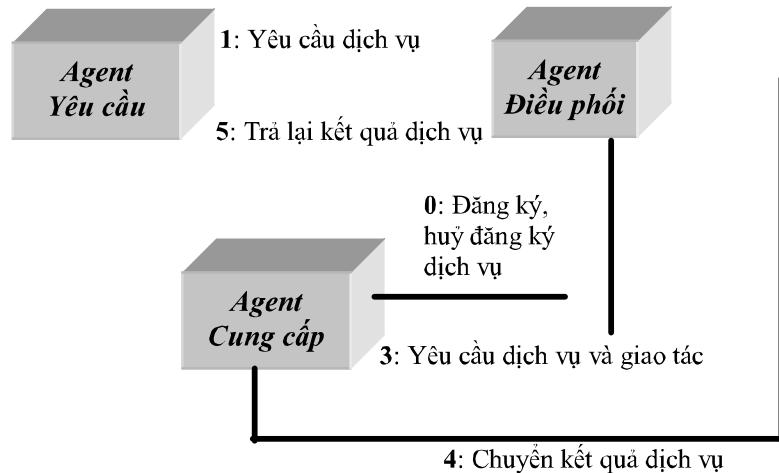
Trong mô hình này, nhiệm vụ của *Agent Trung tâm* là chủ động liên lạc với các agent khác có dữ liệu hay tri thức cần thiết trong hệ thống. Các dịch vụ mà *Agent Trung tâm* có thể cung cấp là:

- Tự động xác định các dịch vụ thông tin;
- Xác định vai trò của các agent trong hệ thống;
- Tự thu thập và tạo ra thông tin từ các *Agent Cung cấp (Provider Agent)* sau đó gửi trả lại cho các agent yêu cầu.

Để thực hiện nhiệm vụ trên, *Agent Trung tâm* sử dụng *mô hình thông tin toàn cục* bằng cách thu thập và tích hợp các thông tin cần thiết để giải quyết các yêu cầu hoặc có thể chuyển yêu cầu cho các agent phù hợp trong hệ thống để giải quyết. Như vậy, *Agent Trung tâm* đóng vai trò vừa là agent trực tiếp quản lý các agent khác lại vừa tự tìm ra thông tin cần thiết để giải quyết và gửi trả lại kết quả cho các agent yêu cầu. Công việc của *Agent Trung tâm* là rất nhiều và hiệu quả hoạt động của hệ thống phụ thuộc hoàn toàn vào khả năng của agent này. Vai trò của *MidAgent* sẽ giảm đi trong hai mô hình còn lại được trình bày sau.

Mô hình tương tác với Agent Điều phối (Broker Agent)

Trong mô hình này, *MidAgent* đóng vai trò là *Agent Điều phối (Broker Agent)*. Công việc mà *Agent Điều phối* cần thực hiện là một phần công việc của *Agent Trung tâm* và được biểu diễn như trong Hình 2.4. Khi có một agent mới tham gia vào hệ thống, thì nó phải đăng ký khả năng cung cấp dịch vụ của mình cho *Agent Điều phối*. Dịch vụ, tên và địa chỉ của *Agent Cung cấp* này sẽ được cập nhật vào cơ sở tri thức của *Agent Điều phối*.



Hình 2.4: Mô hình tương tác sử dụng Agent Điều phối

Khi có một agent *Agent Yêu cầu* gửi cho *Agent Điều phối* một yêu cầu dịch vụ nào đó, *Agent Điều phối* sẽ tìm kiếm trong cơ sở tri thức của mình xem có *Agent Cung cấp* nào có thể giải quyết được yêu cầu của *Agent Yêu cầu* không và sau đó liên lạc trực tiếp với agent đó để giải quyết yêu cầu. Sau cùng, *Agent Điều phối* sẽ gửi lại kết quả cho *Agent Yêu cầu* và kết thúc quá trình tương tác.

Như vậy, trong mô hình tương tác này, bất cứ một liên lạc nào giữa *Agent Yêu cầu* và *Agent Cung cấp* đều phải thông qua *Agent Điều phối*. Trong một số tài liệu, *Agent Điều phối* còn được gọi là Facilitator ([40]). Ưu điểm của mô hình này là khả năng mở rộng hệ thống. Một agent mới muốn tham gia vào hệ thống thì agent đó chỉ cần đăng ký dịch vụ với *Agent Điều phối*. Tuy nhiên, nhược điểm của mô hình này là *Agent yêu cầu* phải gửi đi toàn bộ yêu cầu của mình cho *Agent Điều phối* mà điều này thường không thực tế đặc biệt trong thương mại điện tử.

Mô hình tương tác với Agent Môi giới (MatchMaker Agent)

Trong mô hình này, *MidAgent* đóng vai trò như một *Agent Môi giới* mà nhiệm vụ chính của nó là tạo ra cơ chế liên lạc trực tiếp giữa *Agent Yêu cầu* và *Agent Cung cấp* như minh họa trong Hình 2.5.



Hình 2.5: Mô hình tương tác sử dụng Agent Môi giới

Tương tự như mô hình *Agent Điều phối*, khi muốn tham gia vào hệ thống, mỗi agent phải đăng ký dịch vụ với *Agent Môi giới* và trở thành nhà cung cấp dịch vụ (*Agent Cung cấp*). *Agent Môi giới* sẽ cập nhật vào cơ sở tri thức của nó tên và khả năng dịch vụ của *Agent Cung cấp*. Trong một tương tác cụ thể, khi có một *Agent Yêu cầu* yêu cầu một dịch vụ, nó sẽ gửi yêu cầu đó đến *Agent Môi giới*. *Agent Môi giới* sẽ xem xét trong cơ sở tri thức của nó để tìm ra *Agent Cung cấp* có thể thực hiện yêu cầu và sẽ gửi cho *Agent Yêu cầu* tên, địa chỉ của *Agent Cung cấp* đó. Quá trình tương tác sau đó sẽ diễn ra trực tiếp giữa *Agent Yêu cầu* và *Agent Cung cấp*.

Như vậy, công việc mà *Agent Môi giới* phải thực hiện là một phần công việc của *Agent Điều phối*. Trong mô hình này, *Agent Yêu cầu* chỉ cần gửi đi yêu cầu nào liên quan đến việc tìm ra *Agent Cung cấp* phù hợp.

2.2 Thương lượng trong hệ đa agent

Khác với các mô hình tương tác kiểu cộng tác như mạng hợp đồng hay bảng đen, thương lượng là một tiến trình tương tác vừa cộng tác vừa cạnh tranh được diễn ra giữa hai hay nhiều bên tham gia, bắt đầu bằng những mục tiêu (đích) khác nhau, dần dần đi đến một thoả thuận chung có lợi cho tất cả các bên.

Trong bài toán thương lượng, tuỳ vào số bên tham gia, người ta chia làm bốn nhóm là: thương lượng 1-1, thương lượng 1- n, thương lượng n-1 và thương lượng n-n. Sự phân chia này được minh họa như Hình 2.6.

Số người mua

	1-1	1-n
Số người bán	n-1	n-n

Hình 2.6: Các dạng thương lượng

- *Thương lượng 1-1*: còn gọi là thương lượng song phương, chỉ có một người bán thương lượng với một người mua.
- *Thương lượng n-1*: nhiều người mua một người bán. Đây chính là hình thức đấu giá (Auction). Đấu giá là một trong những hình thức mua bán phổ biến trong thương mại. Trong mô hình đấu giá (nhiều người mua, một người bán), người mua sẽ trả giá theo một cách thức nào đó, ví dụ như ai trả giá cao nhất sẽ thắng.
- *Thương lượng 1-n*: Một người mua có thể thương lượng đồng thời với nhiều người bán. Hình thức này còn gọi là đấu giá ngược (Reverse-Auction).
- *Thương lượng n-n*: còn gọi là thương lượng đa phương hay *chợ* (*market*). Mỗi người mua có thể thương lượng đồng thời với nhiều người bán và mỗi người bán cũng có thể thương lượng đồng thời với nhiều người mua.

Khi hệ thống có các agent đại diện cho người mua và người bán thì các agent này sẽ tự động thương lượng với nhau. Trong thương lượng song phương người ta chia ra ba cách tiếp cận [5]:

Cách tiếp cận dựa trên lý thuyết trò chơi

Đây là cách tiếp cận sử dụng chiến lược tương tác giữa các *agent ích kỷ* (*self-interested agent*) theo các luật trò chơi. Trong cách tiếp cận này, các nhà nghiên cứu cố gắng xác định một chiến lược tối ưu bằng cách phân tích mối tương tác giữa các agent giống như trong một trò chơi và tìm ra điểm cân bằng của quá trình tương tác giữa các agent đó.

Cách tiếp cận dựa trên Heuristic

Trong mô hình này, các hàm quyết định dựa trên heuristic được sử dụng để ước lượng và sinh ra các yêu cầu hay đề nghị mới trong quá trình thương lượng.

Cách tiếp cận dựa trên lập luận (Argumentation-based)

Cách tiếp cận này cho phép các agent có thể chuyển các thông tin thêm hoặc lập luận cho các giá trị tinh thần như niềm tin hay ý định trong quá trình thương lượng.

Phần 3.4 của tài liệu sẽ tìm hiểu một kiến trúc thương lượng sử dụng agent trung gian và áp dụng kiến trúc này cho thương lượng song phương cũng như thương lượng đa phương.

2.3 Mô hình thương lượng song phương

2.3.1 Cơ sở toán học cho thương lượng song phương

Nội dung phần này trình bày các khái niệm cơ bản về logic mờ nhằm làm cơ sở cho việc mô hình hóa chiến lược thương lượng dựa trên ràng buộc mờ sẽ được trình bày trong phần 2.3.2.

Tập mờ

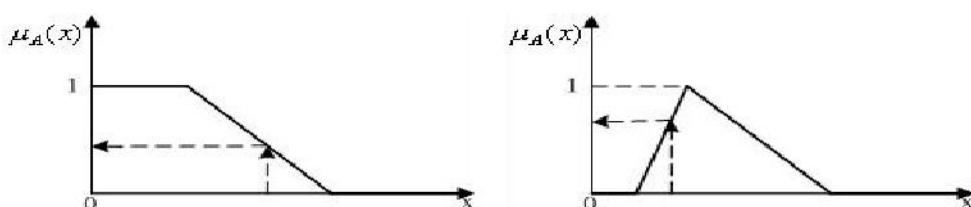
Khái niệm tập mờ nhằm biểu diễn và tính toán với các dữ liệu liên quan đến biến ngôn ngữ.

Định nghĩa 2.1 ([12]): Cho X là một tập khác rỗng. Một tập mờ A trong X được đặc trưng bởi hàm liên thuộc

$$\mu_A : X \rightarrow [0,1]$$

và $\mu_A(x)$ được hiểu là mức độ phụ thuộc của phần tử x vào tập mờ A , với mọi phần tử $x \in X$.

Các hàm liên thuộc có đồ thị dạng hình thang hoặc tam giác thường được chọn cho nhiều ứng dụng thực tế.

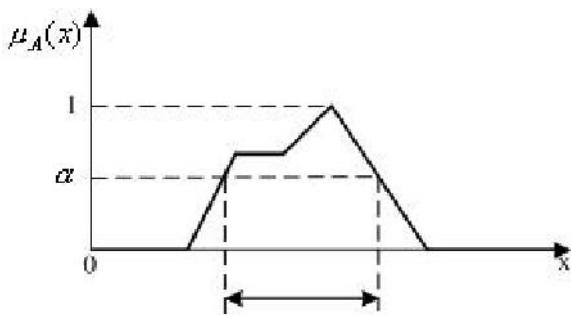


Hình 2.7: Các phương pháp ước lượng mờ: hình thang và tam giác

Lát cắt của tập mờ

Định nghĩa 2.2 ([26]): Gọi A là một tập mờ trong một tập không rỗng X với hàm thuộc μ_A . Một lát cắt α của tập mờ A được xác định là tập các phần tử $x \in X$ sao cho $\mu_A(x) \geq \alpha$.

Hình 2.8 minh họa một cách xác định lát cắt của một tập mờ.



Hình 2.8: Xác định lát cắt của tập mờ

Các toán tử mờ

Định nghĩa 2.3 ([26]): Một hàm $G : [0,1] \times [0,1] \rightarrow [0,1]$ được gọi là hàm ưu tiên nếu nó thoả mãn bốn điều kiện.

1. $G(u_1, a) \geq G(u_2, a)$ nếu $u_1 \leq u_2$,
 2. $G(u, a_1) \leq G(u, a_2)$ nếu $a_1 \leq a_2$,
 3. $G(1, a) = a$,
 4. $G(0, a) = 1$.

Định nghĩa 2.4 ([25]): Một hàm $O : [0,1] \times [0,1] \rightarrow [0,1]$ được gọi là hàm đồng nhất ưu tiên nếu nó thoả mãn các điều kiện.

1. $O(u_1, \lambda) \leq O(u_2, \lambda)$ nêu $u_1 \leq u_2$,
 2. $O(u, \lambda_1) \leq O(u, \lambda_2)$ nêu $\lambda_1 \leq \lambda_2$,
 3. $O(1, \lambda) = \lambda$.

Định nghĩa 2.5 ([26]): Một toán tử hai ngôi $\oplus:[0,1]\times[0,1]\rightarrow[0,1]$ được gọi là toán tử đồng nhất nếu nó đơn điệu tăng, đối xứng và tồn tại một phần tử $\theta\in[0,1]$ sao cho $\forall a\in[0,1]: a\oplus\theta=a$. Phần tử θ khi đó được gọi là phần tử đơn vị của toán tử đồng nhất \oplus .

Một toán tử đồng nhất \oplus với phần tử đơn vi θ có các tính chất sau đây [26]:

- $$l, a \oplus a = a,$$

2. $a \oplus b \geq \max(a, b)$ nếu $a, b > \theta$,
3. $a \oplus b \leq \min(a, b)$ nếu $a, b < \theta$,
4. $\min(a, b) \leq a \oplus b \leq \max(a, b)$ nếu $a < \theta, b > \theta$,
5. $0 \oplus a = a$.

Định nghĩa 2.6 ([26]): Một hàm $F : [0,1] \times [0,1] \times [0,1] \rightarrow [0,1]$ được gọi là hàm chấp nhận tổng thể của người mua đối với một mặt hàng nếu nó thoả mãn các điều kiện.

1. $F(0, \beta, \gamma) = 0$ và $F(\delta, 0, \gamma) = 0$,
2. $F(\delta_1, \beta_1, \gamma_1) \leq F(\delta_2, \beta_2, \gamma_2)$ nếu $\delta_1 \leq \delta_2, \beta_1 \leq \beta_2, \gamma_1 \leq \gamma_2$,
3. $F(\delta, \beta, \gamma) \geq \min(\delta, \beta)$,
4. $\delta < F(\delta, \beta, \gamma) < 1$ nếu $\delta < \theta, \gamma > 0, \theta$ là ngưỡng chấp nhận của người dùng.

Toán tử OWA

Toán tử OWA (Ordered Weighted Average) là một kỹ thuật kết hợp dựa trên các trọng số được sắp xếp, do Ronald R. Yager giới thiệu năm 1988.

Định nghĩa 2.7 ([12]): Một toán tử OWA n chiều là một ánh xạ $F : R^n \rightarrow R$ xác định bởi vector $w = (w_1, w_2, \dots, w_n)^T$ sao cho $w_i \in [0,1], 1 \leq i \leq n$ và $\sum_{i=1}^n w_i = w_1 + w_2 + \dots + w_n = 1$.

$$F(a_1, a_2, \dots, a_n) = \sum_{j=1}^n w_j b_j = w_1 b_1 + \dots + w_n b_n$$

với b_j là phần tử lớn thứ j của tập $A = (a_1, \dots, a_n)$.

Một số trường hợp đặc biệt của toán tử OWA [12].

- Max: trong trường hợp $w^* = (1, 0, \dots, 0)^T$ thì

$$\text{Max}(a_1, \dots, a_n) = \max\{a_1, \dots, a_n\}.$$

- Min: trong trường hợp $w_* = (0, 0, \dots, 1)^T$ thì

$$\text{Min}(a_1, \dots, a_n) = \min\{a_1, \dots, a_n\}.$$

- Average: trong trường hợp $w_A = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})^T$ thì

$$F_A(a_1, \dots, a_n) = \frac{a_1 + \dots + a_n}{n}.$$

Một ứng dụng quan trọng của toán tử OWA là trong *kết hợp theo các lượng từ ngôn ngữ* [12]. Giả sử

$$\{A_1, A_2, \dots, A_n\}$$

là một tập các tiêu chí để đánh giá một lớp các đối tượng nào đó và x là một đối tượng thuộc lớp đó sao cho với mỗi tiêu chí A_i , giá trị hàm $A_i(x) \in [0,1]$ chỉ ra độ thỏa mãn của đối tượng x đối với thuộc tính A_i . Nếu muốn tìm mức độ thỏa mãn “*tất cả các tiêu chí*” của đối tượng x , kí hiệu là $D(x)$, thì

$$D(x) = \text{Min}\{A_1(x), A_2(x), \dots, A_n(x)\}$$

Nếu muốn tìm mức độ thỏa mãn “*ít nhất một tiêu chí*” của đối tượng x , kí hiệu là $E(x)$, thì

$$E(x) = \text{Max}\{A_1(x), A_2(x), \dots, A_n(x)\}$$

Trong các trường hợp còn lại, tùy theo lượng từ ngôn ngữ mà phải xây dựng các hàm tìm mức độ thỏa mãn của đối tượng x theo dạng các lượng từ đơn điệu không giảm hoặc lượng từ đơn điệu không tăng (xem chi tiết [12]).

2.3.2 Chiến lược thương lượng cho agent bán

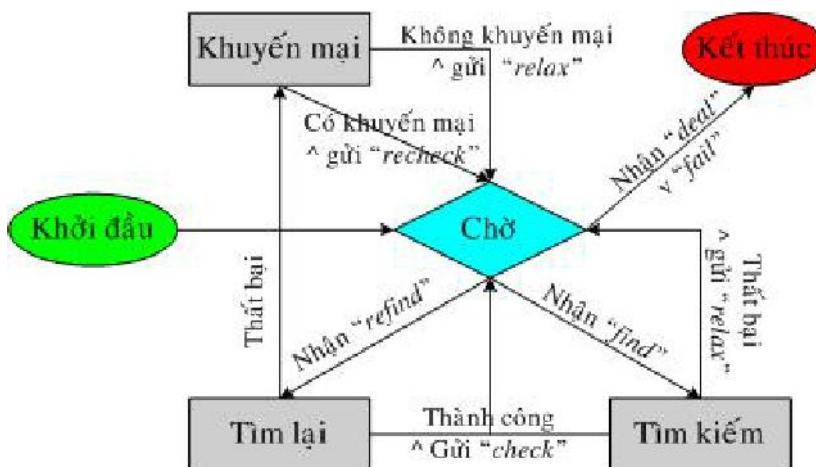
Trong mô hình thương lượng song phương, agent bán có nhiệm vụ quản lý việc giao dịch đồng thời với các khách hàng của mình. Mỗi agent bán được trang bị một tập các tri thức sau đây:

- Tập $O = \{O_1, O_2, \dots, O_m\}$ các đơn vị hàng hóa có thể cung cấp, mỗi đơn vị hàng hóa O_i (của cùng một mặt hàng - mỗi agent bán chỉ bán một mặt hàng) được mô tả thông qua n thuộc tính $\{o_1^i, o_2^i, \dots, o_n^i\}$ và có một giá trị lợi ích g_i nếu bán được đơn vị hàng đó.
- Một tập $R = \{r_1, r_2, \dots, r_m\}$ các ràng buộc đối với người mua của các đơn vị hàng hóa. Tập này có miền giá trị kiểu boolean (có hoặc không).
- Một tập $C = \{c_1, c_2, \dots, c_n\}$ các hình thức khuyến mại cho các đơn vị hàng hóa tương ứng, tập này cũng có miền giá trị kiểu boolean.

- d. Một tập $B = \{B_1, B_2, \dots, B_k\}$ các khách hàng đang thương lượng với agent bán. Mỗi khách hàng được biểu diễn qua các thông tin sau: Tập các đơn vị hàng hoá đã bị từ chối, đơn vị hàng hoá vừa mới giới thiệu đi, các thuộc tính và giá trị các thuộc tính đã yêu cầu.

Khi đó, hoạt động thương lượng của agent bán diễn ra theo sơ đồ được mô tả trong Hình 2.9.

- **Khởi đầu:** sau khi đăng kí với hệ thống, agent bán sẽ chuyển ngay vào trạng thái chờ để chờ các kết nối đến từ các agent mua.
- **Chờ:** tại đây, agent bán sẽ chờ đợi các kết nối đến từ các agent mua. Tuỳ thuộc vào nội dung thông điệp nhận được mà agent bán chuyển vào các trạng thái phù hợp.
- **Tìm kiếm:** là trạng thái mà agent bán chuyển vào khi nhận được thông điệp “find”. Tại đây, nó cập nhật thêm yêu cầu của khách hàng vừa nhận được và tiến hành tìm kiếm các đơn vị hàng hoá thoả mãn các yêu cầu mà nó đã nhận được. Nếu không tìm được đơn vị hàng nào, nó sẽ gửi đi thông điệp “relax” với mong muốn agent bán sẽ nhượng bộ trên thuộc tính nào đó. Nếu tìm thấy, nó chọn đơn vị hàng hoá nào đem lại giá trị lợi ích g_i cao nhất để giới thiệu với agent mua kèm theo thông điệp “check”. Đồng thời cập nhật đơn vị hàng hoá mới nhất vừa được gửi đi cho khách hàng tương ứng.



Hình 2.9: Mô hình chiến lược thương lượng của agent bán

- **Tìm lại:** là trạng thái agent bán chuyển vào khi nhận được thông điệp “refind”. Khi đó, nó sẽ tiến hành tìm kiếm theo các yêu cầu cũ do không có yêu cầu mới bổ sung. Nếu tìm thấy các đơn vị hàng hoá mới, agent bán sẽ chọn đơn vị hàng nào cho giá trị lợi ích g_i cao nhất để gửi đến agent mua với thông điệp “check”. Đồng thời cập nhật đơn vị hàng

hoa mới nhất vừa được giới thiệu. Nếu không tìm thấy hàng hóa mới, agent bán chuyển vào trạng thái khuyến mại để xem có thể bổ sung các hình thức khuyến mại hay không.

- **Khuyến mại:** tại đây, agent bán lấy lại đơn vị hàng hóa mới nhất đã được giới thiệu cho agent mua để kiểm tra xem đơn vị hàng đó có hình thức khuyến mại nào không. Việc này luôn đảm bảo có đơn vị hàng đã giới thiệu, vì trạng thái này chỉ được chuyển đến sau trạng thái tìm lại, trạng thái tìm lại chỉ xảy ra khi agent bán nhận được thông điệp “refind”, tức là trước đó agent mua đã nhận được một đơn vị hàng hóa do chính agent bán này giới thiệu. Nếu đơn vị hàng này có kèm theo khuyến mại, agent bán sẽ gửi các hình thức khuyến mại này kèm theo thông điệp “recheck”. Nếu đơn vị hàng này không có khuyến mại hoặc có khuyến mại nhưng đã được giới thiệu trước đó, agent bán sẽ gửi thông điệp “relax”, đồng thời cập nhật đơn vị hàng hóa này vào tập các hàng hóa đã bị từ chối.
- **Kết thúc:** là trạng thái kết thúc cho một phiên thương lượng với agent mua mà không phải kết thúc cho bản thân agent bán. Trạng thái này đạt được khi nó nhận được thông điệp “deal” báo chấp nhận hoặc “fail” báo thất bại từ phía agent mua.

Quá trình trên có thể diễn ra nhiều lần với một hoặc nhiều agent mua khác nhau. Khác với agent mua là sẽ kết thúc nhiệm vụ sau khi thương lượng, agent bán chỉ kết thúc nhiệm vụ khi đã bán hết các mặt hàng mà nó quản lý.

2.3.3 Chiến lược thương lượng cho agent mua

Trong mô hình thương lượng song phương, mỗi agent mua đại diện cho một khách hàng, sẽ thương lượng với một đối tác duy nhất và do đó mỗi agent mua được trang bị các tri thức sau:

- a. Tập $A = \{A_1, A_2, \dots, A_n\}$ các thuộc tính của hàng hóa. Trong trường hợp nhiều mặt hàng thì mỗi mặt hàng sẽ được xem xét với tập thuộc tính riêng của mặt hàng đó. Mỗi thuộc tính A_i có độ ưu tiên u_i , tương ứng. Trong trường hợp thuộc tính A_i có giá trị khoảng thì mỗi giá trị (hoặc khoảng con các giá trị) trong khoảng đó có một mức độ thỏa mãn người dùng a_i^j .
- b. Mỗi thuộc tính A_i có một giá trị λ_i gọi là ngưỡng nhượng bộ của thuộc tính. Giá trị của ngưỡng này cho biết thuộc tính tương ứng có thể được nhượng bộ thêm nếu giá trị độ thỏa mãn a_i^j của nó chưa thấp hơn λ_i .
- c. Một giá trị θ được gọi là ngưỡng chấp nhận, đặc trưng cho khả năng chấp nhận của người dùng. Nếu khả năng chấp nhận của người dùng càng cao thì ngưỡng chấp

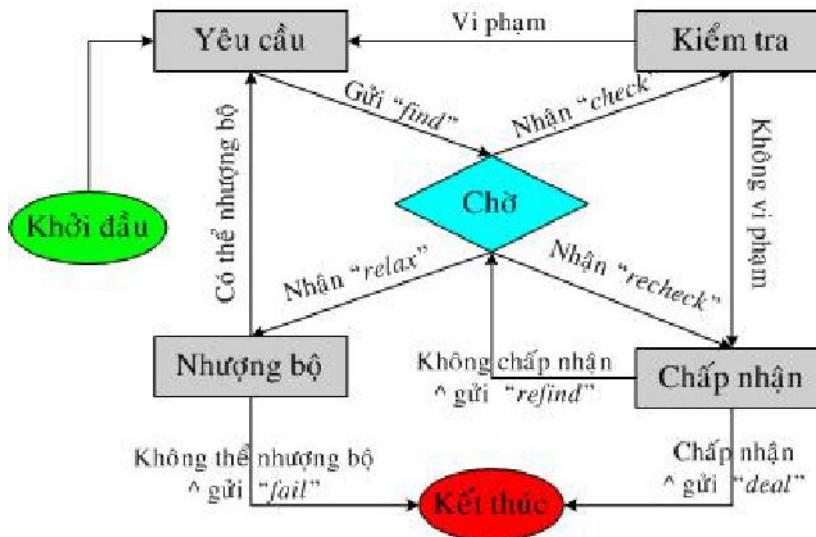
nhận θ có giá trị càng thấp. Nghĩa là mặt hàng chỉ được chấp nhận nếu độ thoả mãn tổng thể của nó không nhỏ hơn ngưỡng này. Ngưỡng chấp nhận θ được ước lượng từ các độ quan trọng u_i và ngưỡng nhượng bộ λ_i của mỗi thuộc tính như sau:

- Ước lượng ngưỡng chấp nhận trên mỗi thuộc tính $\theta_i = O(u_i, \lambda_i)$. Hàm O được chọn thoả mãn các điều kiện của định nghĩa 3.4.
- Kết hợp các giá trị θ_i lại với nhau bằng toán tử OWA. Nhận xét rằng, với yêu cầu của người dùng thì nếu có ít nhất một thuộc tính không thoả mãn thì mặt hàng sẽ không được chấp nhận, nghĩa là trường hợp này thuộc dạng “ít nhất một” và dạng toán tử OWA được áp dụng là toán tử *Max*.

$$\theta = \text{Max}(\theta_1, \theta_2, \dots, \theta_n).$$

- d. Một tập D các cặp thuộc tính và giá trị của các thuộc tính đã gửi đi cho bên agent bán. Tại thời điểm đầu, agent mua chỉ gửi đi các yêu cầu của các thuộc tính có độ ưu tiên cao nhất. Sau đó, trong suốt quá trình thương lượng, agent mua sẽ gửi thêm yêu cầu chỉ khi giá trị thuộc tính tương ứng không thoả mãn yêu cầu của nó. Do vậy, khi có yêu cầu nhượng bộ từ phía agent bán, agent mua chỉ tiến hành chọn lựa khả năng nhượng bộ trên các thuộc tính đã được yêu cầu lưu trong tập D này.

Với tập các tri thức này, quá trình thương lượng của agent mua diễn ra theo sơ đồ được minh họa trong Hình 2.10.

**Hình 2.10:** Mô hình chiến lược thương lượng của agent mua

- Khởi đầu:** Trong trạng thái khởi đầu của phiên thương lượng này, agent mua tìm ra thuộc tính có độ ưu tiên u_i cao nhất để gửi đi. Trong trường hợp có nhiều thuộc tính có độ ưu tiên u_i cao nhất, nó sẽ gửi đi toàn bộ các thuộc tính này.
- Yêu cầu:** Khi muốn gửi đi yêu cầu về thuộc tính mới hoặc giá trị mới cho thuộc tính đã yêu cầu, agent mua sẽ chuyển vào trạng thái **yêu cầu** để thực hiện việc gửi các yêu cầu đó cho phía đối tác. Đồng thời với việc gửi yêu cầu đi, agent mua còn phải cập nhật lại nội dung các thuộc tính được gửi đi (tập D) trong bộ nhớ hoạt động của mình.
- Chờ:** Là trạng thái để agent mua chờ đợi một thông điệp phúc đáp từ phía đối tác. Khi nhận được thông điệp phúc đáp, tùy thuộc vào nội dung của thông điệp mà agent mua chuyển vào các trạng thái tương ứng.
- Kiểm tra:** Khi nhận được thông điệp kiểu "check" từ phía agent bán thì agent mua chuyển vào trạng thái **kiểm tra**. Tại đây, nó tiến hành kiểm tra xem có thuộc tính nào bị vi phạm hay không. Một thuộc tính bị coi là vi phạm nếu giá trị của nó kém hơn giá trị yêu cầu của thuộc tính đó đang được lưu giữ trong bộ nhớ động của agent mua. Khái niệm **kém hơn** là tuỳ thuộc vào thuộc tính của đối tượng. Chẳng hạn, với thuộc tính **giá phòng** thì kém hơn là khi **giá phòng** của khách sạn nhận được cao hơn **giá** được yêu cầu.

Nếu có ít nhất một thuộc tính bị vi phạm, agent mua sẽ chuyển sang trạng thái **yêu cầu** để bổ sung các yêu cầu mới. Trong trường hợp ngược lại, không có thuộc tính nào bị vi phạm, nó sẽ chuyển vào trạng thái **chấp nhận** để kiểm tra xem đối tượng có

thể chấp nhận được không; bởi vì khi không có thuộc tính nào bị vi phạm thì chưa thể chắc chắn rằng mặt hàng sẽ được chấp nhận bởi người dùng.

- **Chấp nhận:** Trong trường hợp không có thuộc tính nào bị vi phạm sau trạng thái kiểm tra hoặc nhận được thông điệp “recheck”, agent mua sẽ chuyển vào trạng thái chấp nhận. Các bước tiến hành để tính độ thoả mãn tổng thể của đối tượng đối với người dùng, dựa trên các kỹ thuật ước lượng mờ được tiến hành như sau.

- Tính độ phù hợp tương đương b_i của thuộc tính A_i dựa vào độ ưu tiên u_i và độ phù hợp a_i ; $b_i = G(u_i, a_i)$. Trong đó, a_i được ước lượng mờ theo phương pháp hình thang từ giá trị thực của mặt hàng so với các giới hạn của người dùng trên thuộc tính tương ứng. Hàm G được chọn thoả mãn các điều kiện của định nghĩa 2.3. Trong áp dụng được trình bày ở chương 5, hàm chuyển đổi độ phù hợp tương đương được sử dụng là $G(u, a) = (a - 1)u + 1$.
- Toán tử OWA được áp dụng để tính độ thoả mãn ràng buộc δ của mặt hàng từ các độ thoả mãn ràng buộc b_i trên các thuộc tính A_i . Nhận xét rằng một mặt hàng muốn thoả mãn được các ràng buộc của người dùng thì nó phải thoả mãn các ràng buộc trên tất cả các thuộc tính; vậy toán tử OWA được áp dụng là dạng “tất cả mọi”, tức là toán tử dạng *Min*.

$$\delta = \text{Min}(b_1, b_2, \dots, b_n).$$

- Tính độ thoả mãn ràng buộc β của bên bán: bên bán có thể có một số ràng buộc đối với người mua, chẳng hạn về độ tuổi hay cấm mang theo vật nuôi... Khi đó, độ thoả mãn β được tính là phần bù (phủ định mờ) của giá trị độ yêu thích của khách hàng về điều kiện tương ứng ($\beta = \bar{a}_i$ với a_i là độ yêu thích của người dùng, $a_i \in [0,1]$).

Ví dụ: nếu khách sạn cấm đưa súc vật vào phòng mà người dùng muốn đưa theo súc vật với mức độ là 40% (0.4) thì độ thoả mãn ràng buộc $\beta = \bar{0.4} = 0.6$.

- Tính độ thoả mãn khuyến mại γ : Với mỗi hình thức khuyến mại, độ thoả mãn sẽ bằng độ yêu thích của khách hàng đối với hình thức khuyến mại đó ($\gamma_i = a_i$ với a_i là độ yêu thích của người dùng, $a_i \in [0,1]$).

Nếu có nhiều hơn một hình thức khuyến mại cho cùng một mặt hàng, các độ thoả mãn γ_i được kết hợp với nhau bởi một phép toán đối xứng hai ngôi (ký hiệu là o) thoả mãn điều kiện:

$$\max\{\gamma_1, \gamma_2\} \leq \gamma_1 \circ \gamma_2 \leq 1.$$

Trong áp dụng ở chương 5, phép toán được sử dụng là $\gamma_1 \circ \gamma_2 = \gamma_1 + \gamma_2 - \gamma_1\gamma_2$.

- Tính khả năng chấp nhận mặt hàng ψ từ các giá trị độ thoả mãn yêu cầu δ , độ thoả mãn ràng buộc β và độ thoả mãn khuyến mại γ : $\psi = F(\delta, \beta, \gamma)$. Hàm F được chọn thoả mãn các điều kiện của định nghĩa 2.6. Trong áp dụng ở chương 4, hàm tính khả năng chấp nhận của người dùng được sử dụng là $F(\delta, \beta, \gamma) = \min(\delta, \beta) \oplus ((1-\theta)\gamma + \theta)$. Trong đó \oplus là toán tử đồng nhất với phần tử đơn vị θ , thoả mãn các điều kiện của định nghĩa 2.5. Trong áp dụng ở chương 4, toán tử đồng nhất được áp dụng là

$$a \oplus b = \frac{(1-\theta)ab}{(1-\theta)ab + \theta(1-a)(1-b)}.$$

với θ là ngưỡng chấp nhận của người dùng.

Sau các bước này, nếu khả năng chấp nhận ψ lớn hơn ngưỡng chấp nhận θ thì mặt hàng được chấp nhận và agent mua gửi thông điệp “deal” đến bên bán và chuyển vào trạng thái kết thúc (thành công). Nếu ψ nhỏ hơn θ thì mặt hàng không được chấp nhận và agent mua sẽ gửi thông điệp “refind” yêu cầu tìm lại với các yêu cầu cũ mà không cần gửi thêm yêu cầu bổ sung.

- Nhượng bộ:** Khi nhận được thông điệp “relax” từ agent bán thì agent mua chuyển vào trạng thái này. Tại đây, nó kiểm tra xem trong các thuộc tính đã gửi yêu cầu, có thuộc tính nào còn nhượng bộ được nữa hay không, một thuộc tính được coi là còn nhượng bộ được nếu giá trị hiện tại của nó vẫn lớn hơn ngưỡng nhượng bộ λ của thuộc tính đó. Sau khi kiểm tra, nếu không có thuộc tính nào có thể nhượng bộ thêm, agent mua sẽ gửi thông điệp “fail” và chuyển sang trạng thái kết thúc (thất bại).

Ngược lại, nếu có thuộc tính còn có thể nhượng bộ thêm, agent mua sẽ chọn ra một thuộc tính để nhượng bộ sao cho giá trị lợi ích của mình bị giảm đi là nhỏ nhất. Giá trị lợi ích bị mất được ước lượng dựa trên độ thoả mãn bị giảm đi Δa_i và độ ưu tiên u_i của thuộc tính đó. Hàm tính giá trị lợi ích bị mất được chọn thoả mãn điều kiện của định nghĩa 3.4 và có thể sử dụng hàm $L(u, a) = ua$.

Khi đó, agent mua sẽ tìm xem thuộc tính A_i nào cho giá trị l_i nhỏ nhất. Giá trị thuộc tính ấy sẽ được nhượng bộ thêm một mức và agent mua chuyển sang trạng thái yêu cầu để gửi đi. Trong trường hợp có nhiều hơn một thuộc tính cùng đạt giá trị nhỏ nhất, agent mua sẽ chọn ngẫu nhiên một trong số các thuộc tính này để nhượng bộ.

Như vậy, agent mua có thể có hai trạng thái kết thúc là thành công hoặc thất bại, việc thành công hay thất bại đều do bản thân agent này quyết định. Điều này là phù hợp với thực tế khi quá trình thương lượng kết thúc là do người mua quyết định mua hoặc không.

2.4 Kết luận

Chương này đã trình bày một cách khái quát về tương tác trong hệ đa agent, ngôn ngữ truyền thông dùng trong tương tác và một số mô hình tương tác. Các kiến trúc hệ đa agent với agent trung gian đã được nghiên cứu và áp dụng rộng rãi trong phát triển các hệ dịch vụ thương mại điện tử. Cho đến nay đã có nhiều mô hình thương lượng được đưa ra, mô hình thương lượng song phương dựa trên ràng buộc mờ đã được trình bày nhằm áp dụng cho phát triển hệ dịch vụ du lịch **TraNeS** sẽ được trình bày trong Chương 5.

CHƯƠNG 3

ONTOLOGY TRONG HỆ ĐA AGENT

- Khái niệm ontology
- Biểu diễn ontology
- Phương pháp luận xây dựng ontology

Ontology là một thuật ngữ trong triết học nhằm mô tả bản chất của sự tồn tại và đã được sử dụng rộng rãi trong lĩnh vực trí tuệ nhân tạo. Trong hệ đa agent, ontology biểu diễn thông tin và tri thức về miền quan tâm của các agent nhằm hỗ trợ tương tác. Mỗi ontology là một cấu trúc phân lớp các *khái niệm* (*concepts*), các *thuật ngữ* (*term*) và các *mối quan hệ* (*relations*) giữa các khái niệm, thuật ngữ đó. Thông qua ontology, các agent sẽ hiểu được nội dung các thông điệp truyền thông mà nó nhận được trong quá trình tương tác. Nội dung chương này nhằm trình bày khái niệm ontology, vấn đề biểu diễn ontology, vai trò của ontology trong tương tác đa agent và phương pháp luận tổng quát để xây dựng ontology.

3.1 Khái niệm Ontology

3.1.1 Khái niệm

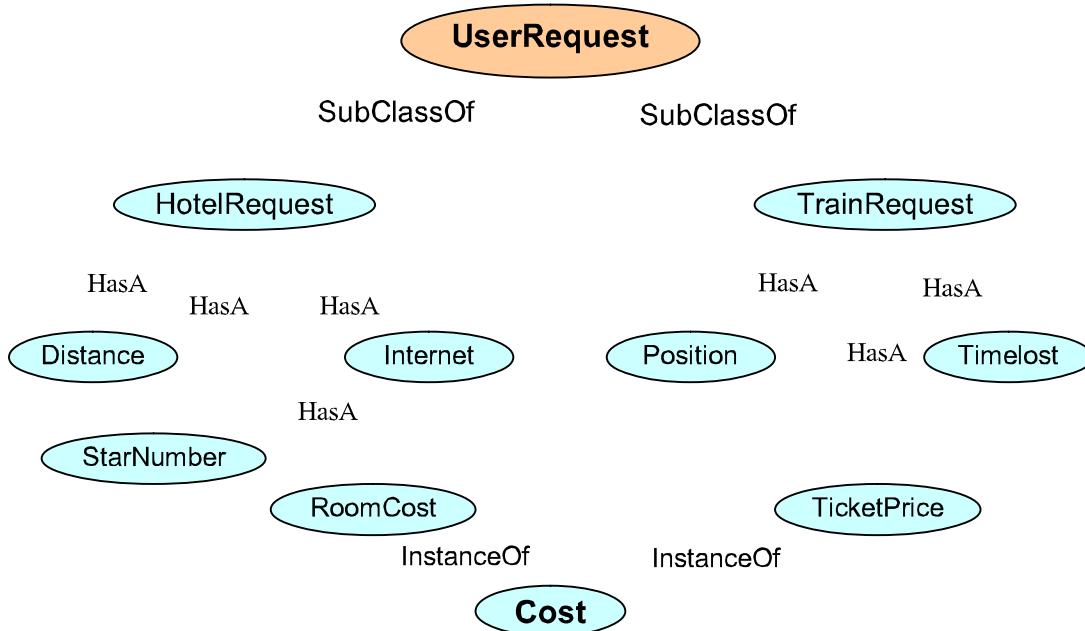
Ontology là một thuật ngữ của triết học đã được sử dụng một cách rộng rãi trong lĩnh vực trí tuệ nhân tạo và đã có nhiều định nghĩa khác nhau về ontology ([1], [4], [15], [17], [20], [36]), trong đó định nghĩa của T. Gruber là được chấp nhận rộng rãi nhất. Theo T. Gruber ([15]), *Ontology là một đặc tả hình thức về khái niệm*. Các yêu cầu cho biểu diễn ontology là:

- Các khái niệm được dùng trong ontology và các ràng buộc giữa các khái niệm đó phải được định nghĩa một cách rõ ràng.
- Ontology phải là dạng thông tin sao cho máy có thể hiểu được.
- Thông tin biểu diễn trong ontology phải có tính phổ quát nghĩa là thông tin đó không chỉ cho một thành phần mà cần được chấp nhận bởi một nhóm các thành phần khác nhau.

Có thể nói, ontology xác định tập các thuật ngữ dùng để mô tả và biểu diễn các khái niệm dựa trên mối quan hệ qua lại (hoặc ràng buộc) giữa các khái niệm đó. Các khái niệm trong ontology giúp cho việc dùng chung và chia sẻ tri thức giữa hai miền tri thức khác nhau.

Khi sử dụng trong hệ đa agent, ontology là một cấu trúc ngữ nghĩa được tham chiếu trong quá trình truyền thông giữa các agent, giúp các agent, với các miền tri thức quan tâm khác nhau, có thể hiểu nhau trong quá trình trao đổi (qua việc hiểu các message mà nó nhận được).

Để minh họa cho định nghĩa trên, ta xem xét ví dụ trong Hình 4.1. Các khái niệm như UserRequest, HotelRequest, TrainRequest ... được biểu diễn dưới dạng các nút trong một đồ thị có hướng. Mỗi cạnh của đồ thị đều có nhãn biểu diễn mối quan hệ giữa các khái niệm trong đồ thị. Ở đây ta có ba mối quan hệ là *SubClassOf* (khái niệm này là lớp con của khái niệm kia), *InstanceOf* (khái niệm này là thể hiện của khái niệm kia) và *HasA* (khái niệm này có một thuộc tính là khái niệm kia).



Hình 3.1: Ví dụ về ontology

3.1.2 Ontology và cơ sở tri thức

Ontology biểu diễn tri thức của các agent chứ không phải về dữ liệu của một miền nào đó. Nó cung cấp tập các khái niệm biểu diễn tri thức trong miền quan tâm của các agent. Các agent sẽ thông qua ontology để hiểu nhau trong quá trình trao đổi, tương tác. Giữa ontology và cơ sở tri thức cũng có sự khác biệt ở vai trò của việc biểu diễn tri thức. Các đặc điểm thể hiện sự khác biệt này bao gồm:

- Ontology hướng tới việc mô tả tri thức như xã hội con người. Trong khi cơ sở tri thức hướng tới việc đặc tả tri thức cho một lĩnh vực cụ thể nào đó mà hệ thống cần giải quyết.
- Ontology liên quan đến tri thức miền tĩnh. Cơ sở tri thức biểu diễn các thông tin có thể bị thay đổi trong quá trình suy luận. Do đó, các luật và các cơ chế suy luận là thành phần rất quan trọng của cơ sở tri thức. Tri thức biểu diễn trong ontology là không đổi trong quá trình suy luận. Ontology biểu diễn các khái niệm như các hành động, các quá trình, các tài nguyên, các khái niệm đó không bị thay đổi khi có suy luận. Trong khi tri thức biểu diễn trong cơ sở tri thức là các hoạt động được thực hiện trong một hệ riêng biệt, các tài nguyên dùng để tạo nên các sản phẩm cụ thể nào đó. Những tri thức như vậy sẽ bị thay đổi trong quá trình hoạt động của hệ thống (khi có suy luận).
- Tri thức trong ontology hướng tới việc sử dụng lại và dùng chung trong các trình ứng dụng.

- Mặc dù ontology hướng tới việc miêu tả các tri thức có tính chất tinh nhưng nó vẫn phụ thuộc chặt chẽ vào trình ứng dụng. Nếu hai trình ứng dụng cùng chung một miền tri thức nhưng nhiệm vụ lại khác nhau thì biểu diễn ontology trong miền đó cho mỗi ứng dụng là khác nhau. Điểm này khác với cơ sở tri thức vì trong cơ sở tri thức, tri thức biểu diễn một miền xác định là nhất quán.

Trong một hệ thống gồm tập hợp nhiều agent, mỗi agent thường có cơ sở tri thức riêng của mình và hệ thống cũng có thể dùng nhiều ontology. Xét về mối quan hệ, ontology mô tả các khái niệm để từ đó mỗi agent (qua tương tác) cập nhật tri thức vào trong cơ sở tri thức của mình. Tuy nhiên, *mục đích chính khi xây dựng ontology trong hệ đa agent là giúp cho quá trình tương tác và truyền thông giữa các agent*.

3.1.3 Phân loại ontology

Ontology có thể được phân loại dựa trên phương pháp hình thành khái niệm. Theo H. Beck et al ([1]), có các kiểu ontology như sau:

- Ontology biểu diễn (Representation ontologies* hay còn gọi là *meta ontologies*). Kiểu ontology này nhằm khái niệm hoá và biểu diễn các tri thức theo kiểu hình thức. Ontology kiểu này định nghĩa các khái niệm như là các lớp, các mối quan hệ, các hàm hay các tiên đề được định danh (*named-axiom*).
- Ontology chung (General ontology)* hay còn gọi là ontology mức cao (*upper ontology*). Kiểu ontology này nhằm phân loại một tập các thực thể tồn tại trong thế giới vật chất. Nó thường biểu diễn các khái niệm chung, không phụ thuộc vào vấn đề hay miền cụ thể nào được mô tả trong ontology đó. Tri thức định nghĩa trong kiểu ontology này thường là sự vật (*thing*), sự kiện (*event*), thời gian (*time*), không gian (*space*) hoặc các khái niệm chung khác.
- Ontology miền (Domain ontologies)*. Tri thức được định nghĩa trong kiểu ontology này là tri thức xác định một miền nào đó. Từ vựng, khái niệm được mô tả trong kiểu ontology này có mối quan hệ gần gũi với các miền tri thức tổng quát, ví dụ như hàng không, y tế...
- Ontology ứng dụng (Application ontology)*. Mô tả các phần tri thức phụ thuộc vào một miền tri thức riêng biệt cũng như nhiệm vụ cụ thể nào đó. Một ontology ứng dụng thường biểu diễn các khái niệm liên quan trực tiếp đến việc giải quyết bài toán.

Tuy được phân thành nhiều loại, nhưng thường thì ontology của một hệ thống bao gồm nhiều kiểu ontology chứ không chỉ có một kiểu ontology nhất định nào đó. Ví dụ, trong

ontology của một hệ thương mại điện tử luôn có các khái niệm thuộc kiểu ontology chung ở mức cao nhất, dưới đó sẽ là các khái niệm ứng với miền tri thức của ứng dụng thương mại điện tử đó. Trong ontology này cũng có thể có các khái niệm chỉ liên quan đến ứng dụng như các khái niệm liên quan đến cơ chế hoạt động của agent.

3.1.4 Vai trò của ontology trong tương tác giữa các agent

Ontology có thể được xem như một khuôn mẫu chung để biểu diễn ngữ nghĩa của thông tin trong một miền xác định thông qua một tập khái niệm (concept), các mối quan hệ (relation) và một tập các luật (axiom) được sử dụng để ràng buộc các khái niệm và mối quan hệ khi cần thiết. Theo [31], ontology được sử dụng cho các mục đích sau đây:

- Hỗ trợ truyền thông giữa con người, giữa con người với máy tính và giữa các hệ thống máy tính độc lập với nhau.
- Cho phép các hệ thống sử dụng lại miền tri thức
- Làm cho miền tri thức trở nên rõ ràng hơn
- Phân tách hay kết hợp các miền tri thức nhờ các phép toán

Trong các trường hợp sử dụng ontology để hỗ trợ truyền thông giữa các agent phần mềm thì các ontology sẽ biểu diễn các khái niệm và các thuật ngữ được sử dụng trong quá trình tương tác giữa các agent cùng với các mối quan hệ cũng như các luật ràng buộc giữa các khái niệm, thuật ngữ đó. Để thực hiện được truyền thông giữa các agent, mỗi một agent trong hệ thống phải biết các thông tin:

- Khả năng và dịch vụ của các agent khác (có thể chỉ biết về một agent, hoặc biết tất cả các agent trong hệ thống).
- Ontology để định nghĩa các khái niệm trao đổi.
- Ngôn ngữ truyền thông agent (*ACL: Agent Communication Language*) mà các agent dùng để biểu diễn thông điệp.

Như vậy các agent trong hệ thống phải dùng chung một ngôn ngữ truyền thông để biểu diễn các thông điệp và trong mỗi thông điệp phải có ontology tương ứng mà agent gửi thông điệp đó sử dụng.

Có hai dạng ngôn ngữ truyền thông phổ biến là KQML (*Knowledge Query and Manipulation Language*) ([11]) và FIPA-ACL (*Foundation Intelligent Physical Agent*) ([10]). Cấu trúc các thông điệp truyền thông biểu diễn theo hai ngôn ngữ này đều có trường ontology trong đó chỉ rõ ontology mà thông điệp đó sử dụng. Agent nhận thông

điệp sẽ dựa trên nội dung trường này để xác định ontology mà agent gửi sử dụng và dựa trên các khái niệm, các thuật ngữ được biểu diễn trong ontology đó để hiểu nội dung của thông điệp.

Bảng 2.1 biểu diễn cấu trúc chung của hai ngôn ngữ KQML và FIPA-ACL. Tuy biểu diễn theo hai dạng khác nhau (KQML biểu diễn theo dạng khai báo lớp còn FIPA-ACL biểu diễn theo dạng thẻ XML) nhưng cấu trúc hai ngôn ngữ này vẫn có những điểm chung. Các trường trong KQML tương ứng với các thẻ trong FIPA-ACL.

Bảng 3.1 Cấu trúc chung của KQML và FIPA-ACL

KQML	FIPA ACL
(KQML-performative sender: <word> receiver: <word> language: <word> ontology: <word> content: <expression>)	<fipa-message act = " "> <sender> </sender> <receiver> </receiver> <content> </content> <language> </language> <ontology> </ontology> <conversation- id></conversation-id> </fipa-message>

Trong phần 3.2.3, chúng ta sẽ xem xét chi tiết hơn các agent sử dụng ontology như thế nào trong quá trình tương tác.

3.2 Biểu diễn ontology

Một trong những vấn đề quan trọng khi nghiên cứu và phát triển các hệ tương tác dựa trên ontology là biểu diễn ontology. Có hai cách tiếp cận để biểu diễn ontology:

- *Cách tiếp cận biểu diễn ontology theo kiểu hình thức:* Theo cách tiếp cận này, ontology sẽ được biểu diễn dựa trên đồ thị từ đó ta có thể xây dựng được cấu trúc đại số với các phép toán trên ontology.
- *Cách tiếp cận biểu diễn ontology theo kiểu không hình thức.* Theo cách này, ontology sẽ được biểu diễn và sử dụng cho phát triển các hệ thống theo một ngôn ngữ nào đó được xây dựng cho mục đích riêng (ví dụ như OIL, RDF, DAML, ...) hoặc sử dụng trực tiếp ngôn ngữ lập trình (ví dụ như Java).

Tài liệu sẽ lần lượt xem xét hai kiểu ontology này trong hai phần 3.2.1 và 3.2.2.

3.2.1 Biểu diễn ontology theo kiểu hình thức

3.2.1.1 Các khái niệm

Theo P. Mitra và G. Weiderhold [28], ontology $O = (G, R)$ được biểu diễn một cách hình thức bởi một đồ thị có hướng G và một tập các luật R , trong đó đồ thị $G = (V, E)$ bao gồm một tập hữu hạn các đỉnh V và một tập hữu hạn các cạnh E .

Nhãn của mỗi đỉnh trong V là một xâu ký tự không rỗng biểu diễn các danh từ. Đó chính là các *khái niệm* (*concept*) hoặc các *thuật ngữ* (*term*). Nhãn của các cạnh trong E là tên của các mối quan hệ ngữ nghĩa giữa các cạnh trong V . Nhãn này có thể rỗng (null) khi mỗi quan hệ giữa hai khái niệm là không xác định. Mỗi cạnh có thể được biểu diễn theo dạng Statement (câu) với cấu trúc chung là Subject-R-Object (SRO). Subject và Object là các khái niệm trong V và R là mối quan hệ giữa hai khái niệm đó.

Luật trong ontology được biểu diễn dựa trên logic vị từ. Theo [28], lớp ngôn ngữ logic vị từ được lựa chọn để biểu diễn luật trong ontology là Horn Clause. Một luật r sẽ có dạng như sau:

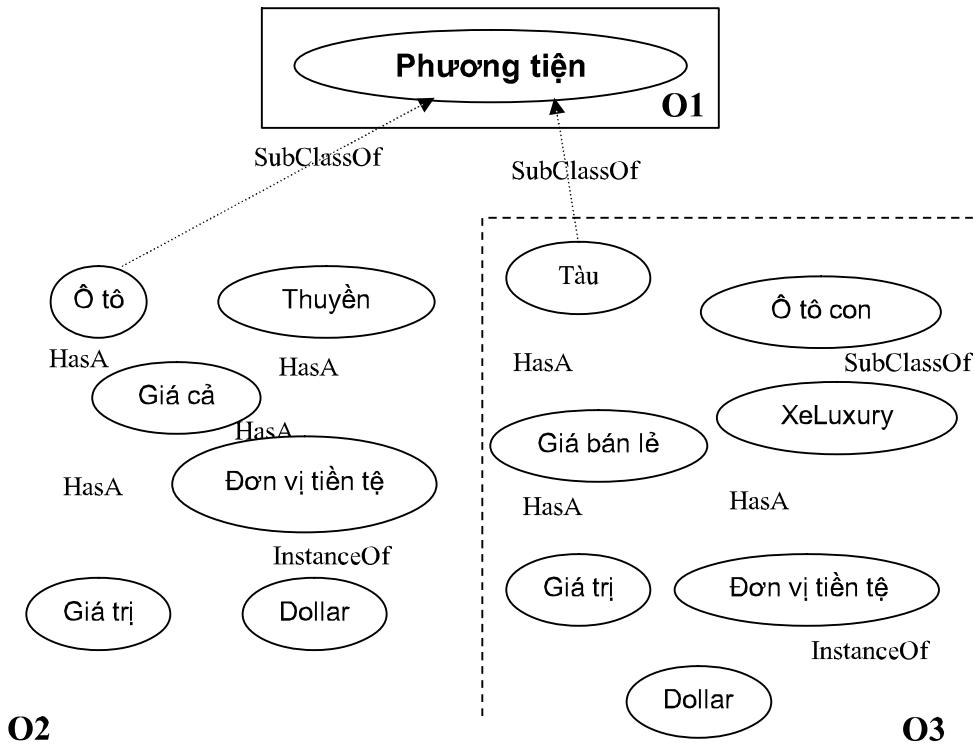
$$\text{CompoundStatement} \Rightarrow \text{Statement}$$

Statement chính là một câu SRO (Subject R Object) như trình bày ở trên, trong đó subject và object có thể là nhãn của các nút trong sơ đồ ontology hoặc là giá trị của một hay nhiều nút trong ontology. Tiền đề của luật r , *CompoundStatement*, có thể là một giá trị Boolean (đúng hay sai) hoặc một sự liên kết các *Statement*.

Để minh họa cho các khái niệm trong phần biểu diễn ontology theo kiểu hình thức, chúng ta xem xét ví dụ như trong Hình 3.1. Ontology O1 chỉ có một khái niệm duy nhất là *Phuong tiện*. Các ontology còn lại có nhiều hơn một khái niệm như O2 có các khái niệm *Ôtô*, *Thuyền*.... Mỗi khái niệm được biểu diễn bởi một đỉnh có hình ovan với nhãn là tên của khái niệm. Các mối quan hệ là các cạnh có hướng trong đồ thị. *SubclassOf* biểu diễn khái niệm này là lớp con của khái niệm kia, *HasA* biểu diễn khái niệm này có thuộc tính là khái niệm kia, *InstanceOf* biểu diễn khái niệm này là thể hiện của khái niệm kia. Như vậy, ontology được biểu diễn giống như mạng ngữ nghĩa trong biểu diễn tri thức.

Trong các phép toán trên tập ontology, các khái niệm được biểu diễn là *Tên_ontology.Tên_khai_niệm* (ví dụ như O2.Ôtô hay O3.XeLuxury).

Với mỗi tập các luật R sẽ có một tập các đỉnh ứng với luật đó, ký hiệu là *Nodes(R)*. Ví dụ $R = \{(O2.\text{Ôtô } \text{SubclassOf } O1.\text{Phuong tiện}), (O2.\text{Thuyền } \text{SubclassOf } O1.\text{Phuong tiện})\}$ thì $\text{Nodes}(R) = \{O2.\text{Ôtô}, O1.\text{Phuong tiện}, O2.\text{Thuyền}\}$.



Các luật kết hợp:

1. **True** => (O2.Ôtô SubclassOf O1.Phương tiện)
2. (X InstanceOf O3.Ôtô), (X HasA X.Giá cả), (Y InstanceOf X.Giá cả) (Y HasA Y.Giá trị), (Z InstanceOf Y.Value), (Y.Value > 40,000) => (X InstanceOf O3.XeLuxury)

Hình 3.2: Ví dụ biểu diễn ontology

3.2.1.2 Luật kết hợp và hàm sinh luật kết hợp

Thông thường, để biểu diễn tri thức miền trong một hệ thống lớn cần một hệ nhiều ontology. Để giải quyết tính chất không đồng nhất giữa các ontology, Mitra [28] cho rằng cần có một hàm sinh các quan hệ tương ứng giữa các khái niệm trong các ontologies khác nhau. Mỗi quan hệ như vậy gọi là một luật kết hợp (articulation rule). Mỗi hàm sinh các quan hệ như vậy gọi là hàm sinh luật kết hợp.

Đầu vào của hàm sinh luật kết hợp là hai ontology (trong tập các ontology có thể có O) và đầu ra của hàm này là một tập các luật kết hợp (trong một tập tất cả các luật có thể R) giữa hai ontology đó. Như vậy, một cách hình thức hóa sinh luật kết hợp là hàm $f: O \times O \rightarrow 2^R$, trong đó O là tập tất cả ontology có thể có và 2^R là tập tất cả các tập con của những luật trong R. Chúng ta giả sử rằng các hàm sinh kết hợp là một hàm hoàn chỉnh,

tức là với 2 ontology cho trước, hàm này sẽ luôn luôn kết thúc và trả lại các luật kết hợp tương ứng. Như vậy, có thể định nghĩa hàm sinh luật kết hợp như sau:

Định nghĩa 1: Cho hai ontology $O1, O2$. Một hàm f nói hai đỉnh $n_1 \in O1$ và $n_2 \in O2$ được gọi là hàm sinh luật kết hợp nếu f sinh ra một cạnh $(n_1 R n_2)$ hoặc $(n_2 R n_1)$, với R là một quan hệ nào đó. Ký hiệu: $f(O1, O2)$.

Hình 3.1 biểu diễn 2 luật kết hợp. Một luật kết hợp có thể có dạng đơn giản như luật số 1 hoặc phức tạp hơn nhiều như trong luật số 2 (**true** là mệnh đề **hằng đúng**). Có một số dạng hàm sinh luật kết hợp như hàm sinh bắc cầu hoặc hàm sinh nhất quán (xem chi tiết trong [28]).

3.2.1.3 Các phép toán trên tập ontology

Từ các định nghĩa ontology, các quan hệ, các luật kết hợp và các hàm sinh luật kết hợp, ta có thể xây dựng các phép toán trên ontology bao gồm các phép lựa chọn (Select), phép giao (intersection), phép hợp (union) và phép hiệu (difference).

Phép chọn

Phép chọn dùng để lựa chọn một phần trong ontology cần quan tâm. Ví dụ, khi một người chỉ quan tâm đến việc đặt chỗ khách sạn mà không quan tâm đến việc mua vé tàu hoả thì người đó chỉ cần xem xét phần ontology mô tả yêu cầu về khách sạn và bỏ đi các khái niệm mô tả yêu cầu liên quan đến các chuyến tàu. Phép Select được định nghĩa theo 2 dạng như sau:

Định nghĩa 2: Cho một ontology $O = ((N, E), R)$ và một đỉnh $n \in N$, phép chọn $select(O, n) = (G_n, R_n)$ với $G_n = (N_n, E_n)$ là một đồ thị con của G sao cho với tất cả các đỉnh n' trong N_n , tồn tại một đường đi từ n đến n' trong G . Trong đó, $E_n = \{e = (n_1 R n_2) \in E | n_1, n_2 \in N_n\}$ là tập sao cho mỗi cạnh trong E_n biểu diễn một quan hệ giữa n_1 và n_2 trong N_n ; $R_n = \{r \in R | Nodes(r) \in N_n\}$ là một tập các luật lấy ra từ tập R tất cả các luật trong O ứng với các khái niệm trong N_n .

Định nghĩa 3: Cho một ontology $O = ((N, E), R)$ và một tập các đỉnh V . Phép chọn được định nghĩa là $Select(O, V) = (G, R_v)$ với $G = (V, E_v)$, $E_v = \{e = (n_1 R n_2) \in E | n_1, n_2 \in V\}$ và $R_v = \{r \in R | Nodes(r) \subseteq V\}$.

Xem xét lại ví dụ đã trình bày trong Hình 3.1. Ontology O3 chứa các cạnh (O3.XeLuxury SubClassOf O3.Ôtôcon) và (O3.XeLuxury HasA O3.Giábánlè). Phép Select(O3, Ôtôcon) sẽ chọn ra tất cả các đỉnh có thể thu được từ đỉnh Ôtôcon trong

ontology O3 (bao gồm Xe Luxury, Giá chi tiết, Đơn vị tiền tệ, Giá trị và Dollar) cùng với tất cả các cạnh giữa chúng. Phép Select(O3, {Ô tô con, Xe Luxury}) chỉ chọn ra hai đỉnh O3. Ô tô con và O3. Xe Luxury cùng với cạnh *Xe Luxury SubClassOf O3*. Ô tô con giữa hai đỉnh đó.

Phép giao

Phép giao thực hiện trên hai ontology cần kết hợp và sinh ra ontology bằng các luật kết hợp được sinh ra từ các hàm sinh kết hợp. Phép giao được định nghĩa như sau:

Định nghĩa 4: Giao của hai ontology $O1 = ((N1, E1), R1)$ và $O2 = ((N2, E2), R2)$ bởi một hàm sinh kết hợp f là $OI_{1,2} = O1 \cap_f O2$, trong đó

$$OI_{1,2} = (NI, EI, RI),$$

$$NI = Nodes(f(O1, O2)),$$

$$EI = Edges(E1, NI \cap N1) + Edges(E2, NI \cap N2) + Edges(f(O1, O2)) \text{ và}$$

$$RI = Rules(O1, NI \cap N1) + Rules(O2, NI \cap N2) + f(O1, O2)).$$

Các đỉnh có trong ontology giao chính là các đỉnh có mặt trong các luật kết hợp. Một cạnh trong ontology giao được tạo nên từ các cạnh hoặc có mặt trong cả hai ontology nguồn hoặc là đầu ra của các hàm sinh kết hợp theo một luật kết hợp. Luật trong ontology giao chính là các luật kết hợp giữa hai ontology nguồn mà trong đó chỉ có các khái niệm có mặt trong ontology giao.

Phép hợp

Định nghĩa 5: Phép hợp giữa hai ontology $O1 = (V1, E1, R1)$ và $O2 = (V2, E2, R2)$ biểu diễn bởi tập $OU = O1 \cup_{AR} O2 = (VU, EU, RU)$ với

$$VU = V1 \cup V2 \cup VI_{1,2}$$

$$EU = E1 \cup E2 \cup EI_{1,2} \text{ và}$$

$$RU = R1 \cup R2 \cup RI_{1,2},$$

và $OI_{1,2} = O1 \cap_{AR} O2 = (VI_{1,2}, EI_{1,2}, RI_{1,2})$ là giao của hai ontology $O1, O2$.

Tập $VI_{1,2}$ chỉ không rỗng khi các luật kết hợp không sinh ra đỉnh mới. Tuy nhiên, trong trường hợp hai nút trong các ontology khác nhau cần kết hợp lại có chung một nhãn (ví dụ O1.x và O2.x) và tồn tại một hàm f chỉ ra rằng chúng cùng biểu diễn một giá trị (giả sử hàm f : Equals: $O1.x Equals O2.x$) thì trong các ontology giao và hợp vẫn có cả hai node O1.x và O2.x với cạnh tương ứng giữa hai khái niệm đó.

Hiệu của hai ontology

Định nghĩa 6: Hiệu của hai ontology $O1$ và $O2$, ký hiệu là $O1 - O2$, với $O1 = ((V1, E1), R1)$ và $O2 = ((V2, E2), R2)$ được định nghĩa là $OD = ((VD, ED), RD)$, với

$$VD = V1 - VI_{I,2}$$

$$ED = E1 - EI_{I,2}$$

$$RD = R1 - RI_{I,2}$$

và với $OI_{I,2} = O1 \cap_f O2 = (VI_{I,2}, EI_{I,2}, RI_{I,2})$ là tập giao của hai ontology với hàm sinh kết hợp f .

Có thể hiểu hiệu của hai ontology chính là phần trong ontology đầu tiên mà không phô biến trong ontology thứ hai. Các đỉnh, cạnh và luật không có trong phần giao của hai ontology (theo định nghĩa phép giao) mà lại có trong ontology thứ nhất sẽ bao gồm các đỉnh, cạnh và luật trong hiệu của hai ontology.

3.2.1.4 Các tính chất của các phép toán

Phần này dành trình bày một số tính chất của các phép toán trên ontology [28].

Tính chất bất biến

Đây là một tính chất quan trọng của các phép toán như hợp và giao. Một cách trực giác, tính chất bất biến yêu cầu hợp (hoặc giao) của một ontology với chính nó sẽ cho kết quả là ontology giống với ontology ban đầu. Trong phép hợp, tính chất bất biến yêu cầu $O1 \cup_f O1 = O1$ (với f là một hàm sinh luật kết hợp). Đặt $OU = O1 \cup_f O1$ và $AR = f(O1, O2)$. Theo định nghĩa phép hợp ta đã có $Nodes(OU) = Nodes(AR)$. Theo tính bất biến ta cần có $Nodes(OU) = Nodes(O1) = Nodes(AR)$. Như vậy, hàm f phải sinh ra các luật kết hợp bao hàm được tất cả các đỉnh trong $O1$.

Tuy nhiên, không phải lúc nào OU cũng hoàn toàn giống $O1$. Ngoài những cạnh và các luật đã có trong $O1$, OU còn có các cạnh kết nối các đỉnh trong $O1$ với chính nó (self-edges) và các luật chỉ áp dụng cho một đỉnh (self-rules). Khi đó, điều kiện chặt chẽ của tính bất biến không thỏa mãn; trong [28] giới thiệu một khái niệm khác gọi là *tính bất biến về ngữ nghĩa* như sau:

Một phép toán là bất biến về ngữ nghĩa nếu và chỉ nếu phép toán đó áp dụng trên hai bản sao của cùng một ontology, bỏ đi các cạnh kết nối một đỉnh với chính nó và các luật chỉ áp dụng cho một đỉnh, thì ontology kết quả giống với ontology ban đầu.

Tính chất bất biến trong phép giao hoàn toàn tương tự như trong phép hợp. Dễ dàng nhận ra là phép hiệu không thể có tính chất bất biến.

Tính chất giao hoán

Tính chất giao hoán cũng là một tính chất quan trọng, tính chất này cho phép đổi thứ tự các ontology trong một phép toán giúp cho quá trình thực hiện tính toán thuận lợi hơn. Tương tự như với tính chất bất biến, trong [28] cũng định nghĩa tính chất giao hoán về ngữ nghĩa.

Định nghĩa 7: Một hàm sinh kết hợp f được gọi là *hàm sinh giao hoán* về ngữ nghĩa nếu và chỉ nếu

$$f(O_1, O_2) \Leftrightarrow f(O_2, O_1)$$

với mọi ontology O_1, O_2 .

Tính chất giao hoán về ngữ nghĩa của phép giao (hoặc phép hợp): Một phép giao (hoặc phép hợp) được gọi là *giao hoán* về ngữ nghĩa nếu hàm sinh kết hợp được sử dụng trong phép giao (hay phép hợp) đó là *hàm sinh giao hoán* về ngữ nghĩa.

Tính chất giao hoán không áp dụng được cho phép hiệu.

Tính chất kết hợp

Nếu như một phép toán có tính chất kết hợp, chúng ta có thể thay đổi thứ tự các ontology nhằm tối ưu hóa quá trình tính toán.

Trong phép giao, tính chất kết hợp yêu cầu

$$(O_1 \cap_{ARules} O_2) \cap_{fARules} O_3 = O_1 \cap_{ARules} (O_2 \cap_{ARules} O_3)$$

trong đó ARules là một hàm sinh luật luật kết hợp. Tính chất kết hợp của phép giao phát biểu như sau:

Một phép giao sử dụng một hàm sinh luật kết hợp f có tính chất kết hợp nếu hàm f là nhát quán và kết nối bắc cầu.

Nếu như hàm f không nhát quán và không kết nối bắc cầu thì trong nhiều trường hợp tính chất kết hợp sẽ không được thoả mãn (xem thêm trong [28]). Tính chất kết hợp trong phép hợp cũng được phát biểu hoàn toàn tương tự như trong phép giao. Như vậy, với việc sử dụng đồ thị và lý thuyết đồ thị để biểu diễn ontology, chúng ta có thể xây dựng được đại số trên các ontology với các phép toán tương ứng. Đây chính là cơ sở cho các nghiên cứu về tổng hợp và sử dụng lại ontology.

3.2.2 Biểu diễn ontology theo kiểu không hình thức

Khác với biểu diễn ontology theo kiểu hình thức, biểu diễn ontology theo kiểu không hình thức gắn bó chặt chẽ với việc xây dựng hệ thống. Theo cách tiếp cận này, các mối quan hệ trong ontology sẽ được định nghĩa rõ ràng bởi các ngôn ngữ riêng, hoặc sử dụng trực tiếp các mối quan hệ sẵn có như lớp-lớp con, lớp-đối tượng-giá trị...trong các ngôn ngữ đó. Có rất nhiều phương pháp biểu diễn ontology không hình thức khác nhau và có thể phân chia thành hai dạng chính:

- *Cách biểu diễn ontology trực tiếp bằng ngôn ngữ lập trình*
- *Cách biểu diễn ontology sử dụng các ngôn ngữ riêng*

Trong cách biểu diễn thứ nhất, ontology được biểu diễn trực tiếp bằng ngôn ngữ lập trình như Java hay C++. Cách biểu diễn này có ưu điểm là hỗ trợ phát triển nghĩa là có thể sử dụng trực tiếp kết quả biểu diễn cho phát triển các trình ứng dụng và khi đó các thành phần trong hệ thống dễ dàng hiểu được nội dung và cấu trúc của ontology. Để minh họa cho cách biểu diễn này, ta xem xét lại ví dụ trong Hình 3.1. Giả sử các lớp UserRequest và Cost đã được định nghĩa trước. Khi đó, các khái niệm HotelRequest, TrainRequest sẽ được định nghĩa thành các lớp, các khái niệm khác sẽ được định nghĩa thành các thuộc tính tương ứng với hai lớp trên và được biểu diễn trong ngôn ngữ Java như sau:

```
public class HotelRequest extends UserRequest implements java.io.Serializable{

    private Cost roomCost;
    private int starNumber;
    private int distance;
    private boolean internet;

    public int getroomCost() {
        return roomCost;
    }

    public int getstarNumber() {
        return starNumber;
    }

    public int getdistance() {
        return distance;
    }

    public boolean getinternet() {
        return internet;
    }

    public void setroomCost(Cost newcost) {
        this.roomCost = newcost;
    }
}
```

```

public void setstarNumber(int newstarNumber) {
    this.starNumber = newstarNumber;
}

public void setdistance(int newdistance) {
    this.distance = newdistance;
}

public void setinternet(boolean newinternet) {
    this.internet = newinternet;
}
}

public class TrainRequest extends UserRequest implements java.io.Serializable{

    private int possition;
    private int timelost;
    private Cost ticketPrice;

    public int getpossition(){
        return possition;
    }

    public int gettimelost(){
        return timelost;
    }

    public int getticketprice(){
        return ticketPrice;
    }

    public void setpossition(int newpossition){
        this.possition = newpossition;
    }

    public void settimelost(int newtimelost){
        this.timelost = newtimelost;
    }

    public void setticketprice(Cost newprice){
        this.ticketPrice = newprice;
    }
}

```

Như vậy, trong cách biểu diễn ontology theo ngôn ngữ lập trình, các mối quan hệ được định nghĩa theo *quan hệ kế thừa* và *quan hệ lớp - thuộc tính* cùng với các hàm trả lại giá trị trong các lớp và thuộc tính đó.

Để hỗ trợ cho phát triển các hệ đa agent trên Internet, nhiều nghiên cứu đã tiến hành và đưa ra các ngôn ngữ dành riêng cho biểu diễn ontology như *Topic Maps*, *RDF*, *OIL*, *DAML* (xem chi tiết trong [1]). Hầu hết các phương pháp này đều tập trung vào việc biểu diễn các tài nguyên trên Internet thông qua các địa chỉ URL và khi đó các mối quan

hệ giữa các tài nguyên sẽ được cấu trúc giống như mạng ngữ nghĩa trong biểu diễn tri thức.

3.3 Phương pháp luận xây dựng ontology tổng quát

Có rất nhiều cách biểu diễn ontology không hình thức khác nhau và mỗi cách biểu diễn lại tương ứng với một công cụ hay một phương pháp xây dựng ontology cụ thể. Tuy nhiên, đặc điểm chung của các cách biểu diễn này là đều xem ontology như một cấu trúc phân lớp các khái niệm và các thuộc tính của khái niệm đó [1]. Cấu trúc phân lớp này sẽ được biểu diễn như một *mạng ngữ nghĩa* với các mối quan hệ giữa các khái niệm.

Dựa trên đặc điểm này, các phương pháp xây dựng ontology cuối cùng đều chuyển các khái niệm, các thuật ngữ, các mối quan hệ trong ontology thành tập các lớp, các thuộc tính... và biểu diễn ontology theo một phương pháp biểu diễn nào đó. Các phương pháp xây dựng ontology đều có đầu vào là các yêu cầu của hệ thống, yêu cầu của tương tác trong hệ đa agent cần xây dựng... và đầu ra là một sơ đồ phân cấp ontology biểu diễn các khái niệm cần thiết cho tương tác của các agent trong hệ đa agent đó.

Tuy có nhiều điểm khác nhau nhưng do mục đích, đầu vào và đầu ra giống nhau nên ta có thể quy tất cả các phương pháp xây dựng ontology theo một phương pháp chung tổng quát nhất gọi là *vòng đời phát triển ontology*. Theo H. Beck và H. Pinto [1], việc xây dựng ontology là một vòng đời hoàn chỉnh theo một tiến trình (*process*) cụ thể từ tập yêu cầu vào đến khi có cây phân cấp ontology và sử dụng ontology trong tương tác của hệ đa agent. Một cách tổng quát, tiến trình đó bao gồm một dãy các hành động, mỗi hành động được xem như một pha trong vòng đời phát triển ontology. Các pha trong quá trình xây dựng ontology được biểu diễn như trong Hình 3.3.

Đặc tả (Specification)

Pha này nhằm xác định mục đích và phạm vi của ontology. Mục đích là trả lời cho câu hỏi “*Tại sao ontology lại được xây dựng?*”. Còn phạm vi của ontology nhằm trả lời cho câu hỏi “*Ontology hướng tới cái gì, đối tượng sử dụng ontology là người dùng hay trình ứng dụng?*”

Hình thành khái niệm (Conceptualization)

Bước này tiến hành tập hợp các khái niệm cần có trong ontology và mỗi khái niệm được mô tả trong một mô hình khái niệm (*conceptual mode*). Mô hình khái niệm này phải đảm bảo bao gồm tất cả các đặc tả về mục đích và phạm vi của ontology đã có trong bước trước.

Hình thức hoá (Formalization)

Bước này nhằm hình thức hoá các khái niệm trong ontology. Bước này được thực hiện thông qua việc chuyển đổi mô hình khái niệm trong bước trước vào một mô hình hình thức.

Cài đặt (Implementation)

Mỗi thành phần trong mô hình ontology hình thức đã có trong bước trước sẽ được cài đặt theo một ngôn ngữ biểu diễn ontology nào đó.

Bảo trì (Maintenance)

Mỗi thành phần sẽ được cập nhật và sửa đổi để tạo nên ontology hoàn chỉnh.

Ngoài các pha trên, trong vòng đời phát triển ontology tổng quát còn có các hoạt động tồn tại song song trong toàn bộ quá trình phát triển ontology. Các hoạt động đó bao gồm:

Thu thập tri thức (knowledge acquisition)

Mỗi tri thức liên quan đến ontology sẽ được thu thập nhờ sử dụng các kỹ thuật suy luận hoặc tham chiếu đến một danh mục tri thức phù hợp.

Đánh giá (evaluation)

Trong hoạt động này, người phát triển sử dụng một kỹ thuật ước lượng cụ thể để đánh giá các pha xây dựng ontology.

Viết tài liệu (Documentation)

Hoạt động viết tài liệu cần thiết trong tất cả các pha của vòng đời phát triển ontology tổng quát. Với mỗi công việc, người phát triển phải viết tài liệu để trả lời các câu hỏi như *đã làm công việc gì, làm như thế nào và tại sao lại thực hiện công việc đó*.

Sử dụng lại (reuse)

Người phát triển cũng có thể sử dụng lại các ontology đã có trong quá trình xây dựng ontology của mình. Đây chính là vấn đề tích hợp ontology và đang thu hút nhiều mối quan tâm trong thời gian gần đây.



Thu thập tri thức

Đánh giá

Viết tài liệu

Hình 3.3: Các bước xây dựng ontology

Trên đây là vòng đời phát triển ontology tổng quát. Có rất nhiều phương pháp xây dựng ontology cụ thể đã ra đời như TOVE, ENTERPRISE, METHONTOLOGY, ... Các phương pháp này thường tập trung vào việc xây dựng ontology từ đầu với vòng đời gồm các bước khác nhau. Tuy nhiên, về cơ bản thì các phương pháp này vẫn tuân theo vòng đời phát triển ontology tổng quát đã nói đến ở trên.

Phương pháp luận MaSE đã tích hợp quá trình xây dựng ontology thành một bước trong pha phân tích. Như vậy, MaSE với agentTool đã hỗ trợ quá trình phát triển một hệ đa agent hoàn chỉnh. Kỹ thuật xây dựng ontology trong MaSE dựa trên các bước trong vòng đời phát triển ontology tổng quát và sẽ được trình bày chi tiết hơn trong Chương 5 của tài liệu này.

3.4 Kết luận

Chương này đã trình bày khái quát về ontology, các phương pháp biểu diễn ontology và phương pháp luận phát triển ontology tổng quát. Việc áp dụng ontology trong hệ đa agent còn rất nhiều vấn đề như biểu diễn ngữ nghĩa thông tin, tổng hợp và sử dụng lại ontology.... cần được tiếp tục nghiên cứu phát triển đặc biệt cho tích hợp thông tin các hệ thống lớn ([33], [34]). Kỹ thuật xây dựng ontology trong MaSE sẽ được trình bày chi tiết hơn trong Chương 4.

CHƯƠNG 4

QUY TRÌNH PHÁT TRIỂN HỆ PHẦN MỀM HƯỚNG AGENT

- Pha phân tích
- Pha thiết kế
- Kết luận

MaSE là một trong những phương pháp luận phát triển phần mềm hướng agent được xây dựng dựa trên phương pháp phát triển phần mềm hướng đối tượng. So với các phương pháp luận khác, phương pháp luận này có ưu điểm là dễ dàng hơn cho những người đã quen thuộc với phát triển phần mềm hướng đối tượng và tích hợp được phát triển ontology trong quá trình phát triển hệ đa agent. Nội dung chương này tập trung trình bày quy trình phát triển hệ phần mềm hướng agent cùng với các bước tương ứng phát triển dựa trên công cụ agentTool. Các bước phát triển này được minh họa với hệ dịch vụ du lịch TraNeS như đã đề cập trong các chương trước.

4.1 Đặc điểm của phương pháp luận MaSE

MaSE (Multiagent System Engineering) là phương pháp luận để phân tích và thiết kế các hệ phần mềm hướng agent được phát triển bởi nhóm nghiên cứu thuộc Viện Công nghệ Hàng không Hoa Kỳ (Air Force Intistute of Technology - AFIT) ([5], [6], [7], [8], [9]).

Trong các cách tiếp cận xây dựng phương pháp luận phát triển hệ đa agent đã được trình bày trong Chương 2, MaSE là một phương pháp luận được phát triển dựa trên cách tiếp cận hướng đối tượng. Quan điểm xây dựng của phương pháp luận này là xem agent như mức trừu tượng cao hơn của một đối tượng: *mỗi agent được xem là một đối tượng đặc biệt*. Khác với một đối tượng truyền thống trong đó các phương thức có thể được gọi bởi các đối tượng khác, các agent tương tác với nhau thông qua hội thoại và hành động một cách tự chủ để hoàn thành mục đích của riêng mình cũng như mục đích chung của hệ thống. Ngoài ra, các agent được xem như là một sự khái quát hoá đối tượng phù hợp với bài toán cụ thể, nó có thể có hoặc không có khả năng thông minh. Do đó, các agent và các thành phần không thông minh trong hệ thống được xử lí tương tự như nhau. Như DeLoach đã khẳng định [9], việc xem agent là một mức trừu tượng cao hơn của đối tượng khiến cho việc phân tích thiết kế hướng agent có thể kế thừa từ các phương pháp luận phát triển phần mềm hướng đối tượng.

Quá trình phát triển hệ đa agent theo MaSE bao gồm hai pha: pha phân tích và pha thiết kế:

- Pha phân tích bao gồm các bước: *Xác định Goal*, *Xác định các Use Case*, *Xây dựng Ontology* và *Hoàn thiện Role*.
- Pha thiết kế bao gồm các bước: *Xác định Agent*, *Xây dựng hội thoại*, *Hoàn thiện Agent* và *Triển khai hệ thống*.

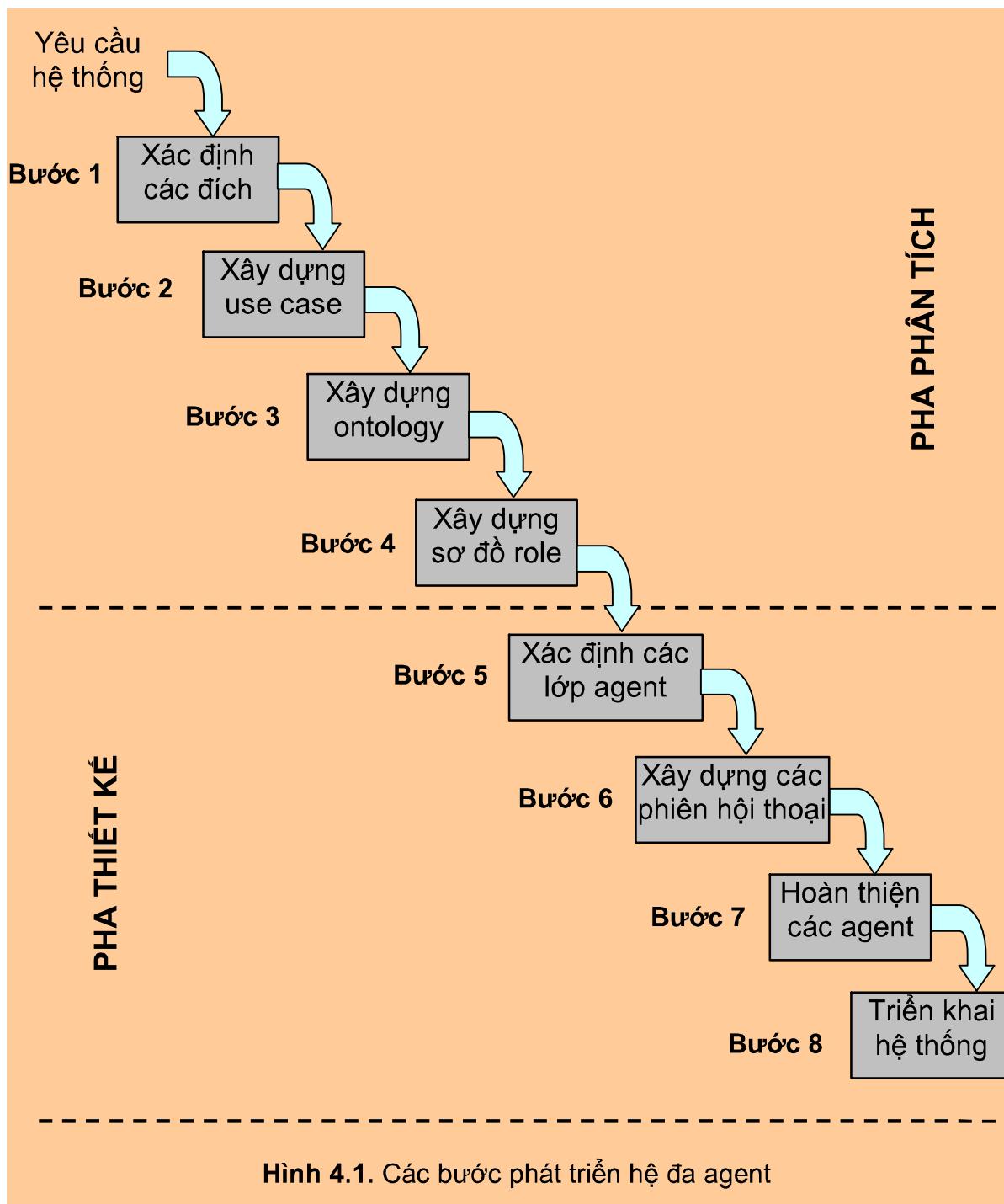
Tuy nhiên, trong quá trình tiến hành, người thiết kế có thể chuyển đổi qua lại giữa các bước một cách tự do để có thể bổ sung các thiếu sót hay điều chỉnh lại các bước để đảm bảo tính nhất quán và toàn vẹn của hệ thống đang thiết kế. Toàn bộ quá trình phân tích thiết kế hệ thống theo phương pháp luận MaSE được hỗ trợ bởi bộ công cụ agentTool. Mỗi bước đều được biểu diễn bởi các sơ đồ tương ứng trong agentTool. Bên cạnh đó, bộ công cụ này còn hỗ trợ người thiết kế kiểm thử tương tác giữa các agent và sinh mã tự động cho hệ thống.

Phản tiếp theo của tài liệu sẽ trình bày quy trình phát triển hệ phần mềm hướng agent theo phương pháp luận MaSE. Tương ứng với mỗi bước, tài liệu sẽ trình bày trình tự thực hiện bước đó dựa trên bộ công cụ agentTool.

4.2 Quy trình phát triển hệ phần mềm hướng agent

4.2.1 Khái quát các bước phát triển

Các bước phát triển hệ phần mềm hướng agent được biểu diễn như trong Hình 4.1.



4.2.2 Pha phân tích

Bước 1: Xác định các đích



Hình 4.2: Bước xác định các đích

Đích (goal) là một khái niệm để chỉ mục đích mà hệ thống cần đạt được. Mục đích của hệ thống ở đây được nhìn từ quan điểm của hệ thống nghĩa là các dịch vụ mà hệ thống có thể cung cấp. Đích sẽ được phân rã thành các đích con, các đích con này lại được tiếp tục phân rã và các đích ở mức thấp hơn này sẽ không được coi là đích mà chỉ được xem xét để đưa vào các bước sau của pha phân tích.

Nhiệm vụ của bước này là chuyên toàn bộ đặc tả yêu cầu của hệ thống vào tập các đích có cấu trúc. Như vậy, có hai bước trong việc xây dựng cây đích: tập hợp các đích và xây dựng cây phân cấp các đích.

- **Tập hợp đích**

Bước này thực hiện trích các yêu cầu chức năng có trong tài liệu đặc tả hệ thống, mỗi yêu cầu chức năng được mô tả bằng một đích. Các yêu cầu chức năng được xác định bằng cách trả lời câu hỏi “*Hệ thống phải làm cái gì*” mà chưa cần quan tâm đến cách thức thực hiện nhiệm vụ đó như thế nào.

Các đích đầu tiên được xác định một cách trực quan thông qua việc xác định mục tiêu cần đạt được của hệ thống. Chẳng hạn với hệ dịch vụ đặt chỗ khách sạn thì ta có thể xác định ngay được đích đầu tiên là *đặt chỗ*. Các đích tiếp theo được xác định thông qua đích trước bằng cách trả lời câu hỏi “*Muốn đạt được đích X thì cần phải có cái gì?*”. Quá trình này được gọi là quá trình phân rã, các đích được phân rã từ các đích ban đầu sẽ trở thành các đích con. Sự phân rã sẽ diễn ra với tất cả các đích đã được phát hiện nhưng chưa được phân rã.

Quá trình phân rã sẽ dừng lại khi các chức năng con sinh ra không phải là nhiệm vụ mức hệ thống, nghĩa là không thể đóng vai trò đích của hệ thống. Các đích không cần phân rã thêm có đặc điểm là khi cố gắng phân rã đích này ta sẽ phải trả lời câu hỏi “*muốn hoàn thành việc này thì cần làm cái gì?*”, tức là tìm ra một cách thức thực hiện đích đó chứ không phải là một đích con.

- **Tổ chức cây đích**

Bước con này có nhiệm vụ tổ chức các đích đã xác định trong bước con trước vào một sơ đồ phân cấp đích (Goal Hierarchy Diagram). Một sơ đồ phân cấp đích là một đồ thị có hướng và không có chu trình (dạng tựa hình cây). Trong đó:

- Các đỉnh biểu diễn các đích, có tên trùng với tên của đích mà nó biểu diễn.
- Các mũi tên chỉ ra quan hệ đích cha – con và quan hệ với các đích khác.

Có hai trường hợp xảy ra: (i) Nếu đã xác định được đích tổng thể của hệ thống thì đặt nó ở gốc của cây đích; (ii) Nếu đích tổng thể không xác định được trực tiếp từ yêu cầu thì phải kết hợp các đích ở mức cao nhất lại thành một đích tổng thể cho hệ thống. Các đích còn lại có thể phân cấp thành các quan hệ cha - con hoặc ngang hàng bằng cách lặp các thủ tục sau:

Bước 1: Các đích được phân rã từ các đích khác trong bước con trước phải là đích con với đích cha tương ứng.

Bước 2: Nếu các đích không được phân rã từ bất kì một đích nào (các đích được xác định ngay ban đầu), để xác định quan hệ cha – con, thì trả lời câu hỏi “*chúng có thực hiện một phần nhiệm vụ cho một đích nào đó không?*”. Nếu có, nó sẽ trở thành đích con của đích mà nó hỗ trợ. Nếu không, phải xem xét lại rằng đích đó có cần thiết cho hệ thống hay không. Nếu không cần thiết thì nó sẽ bị loại bỏ và ngược lại, nếu cần thiết thì nó sẽ tạo thành một nhánh mới ngay từ nút gốc của cây đích.

Trong cây đích có thể có bốn loại đích sau:

Đích chung (Summary goal): là một đích được tạo ra từ các đích ngang hàng (thường là đích tổng thể của hệ thống).

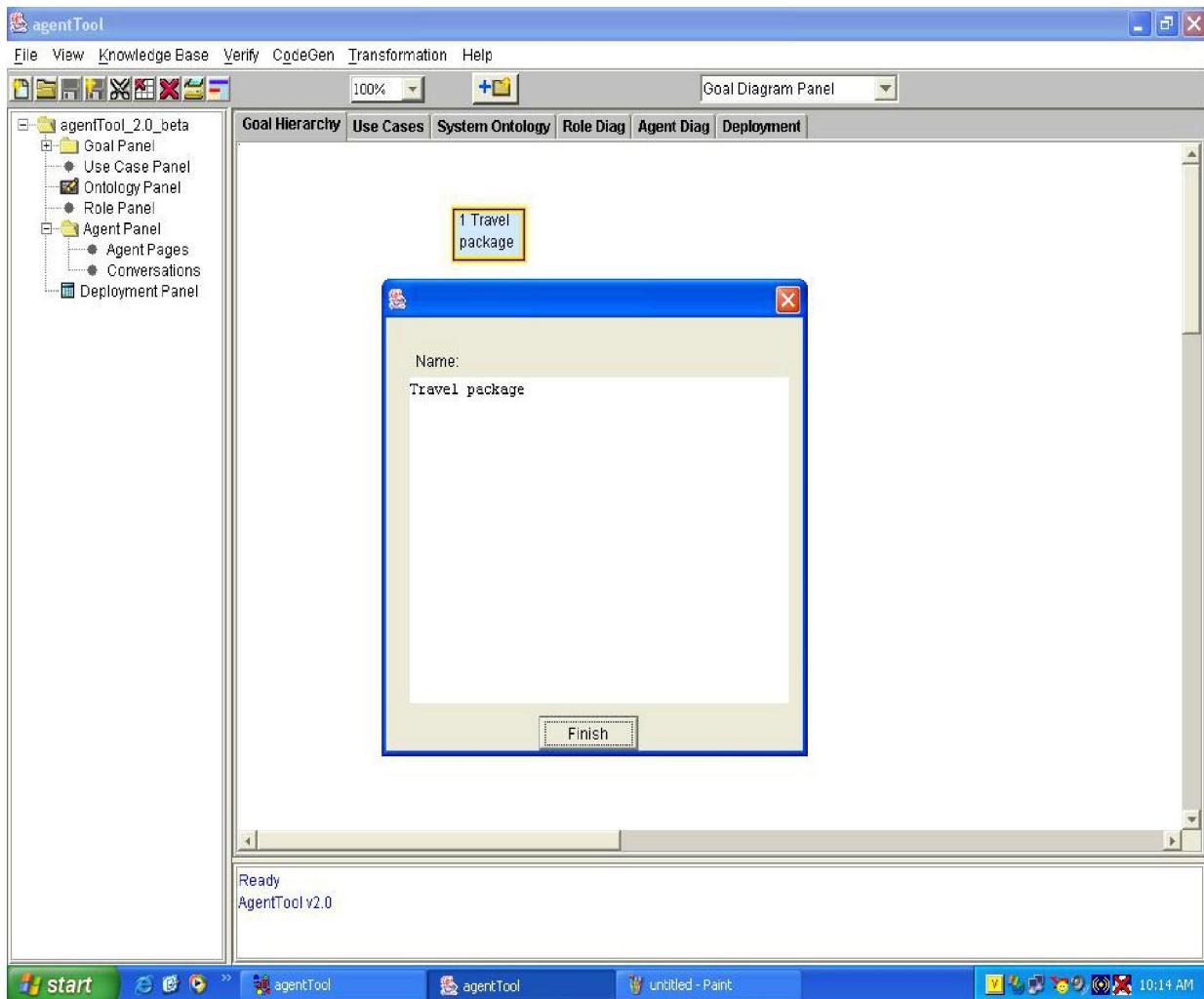
Đích phi chức năng (Non-functional goal): là các đích không thực hiện trực tiếp một chức năng nào của hệ thống, nhưng là nhân tố kiểm tra tính đúng đắn của hệ thống. Các đích này thường xuất hiện từ các yêu cầu phi chức năng chẳng hạn như độ tin cậy hay yêu cầu thời gian thực cho hệ thống.

Đích được kết hợp (Combined goal): là các đích được tạo thành khi kết hợp hai hay nhiều đích có chức năng giống nhau hoặc tương tự nhau.

Đích bị phân hoạch (Partitioned goal): là đích được phân hoạch hoàn toàn. Theo đó, nếu tất cả các đích con của nó được hoàn thành thì bản thân nó cũng được hoàn thành mà không cần thực hiện thêm nhiệm vụ nào nữa.

Biểu diễn cây phân cấp đích trong agentTool

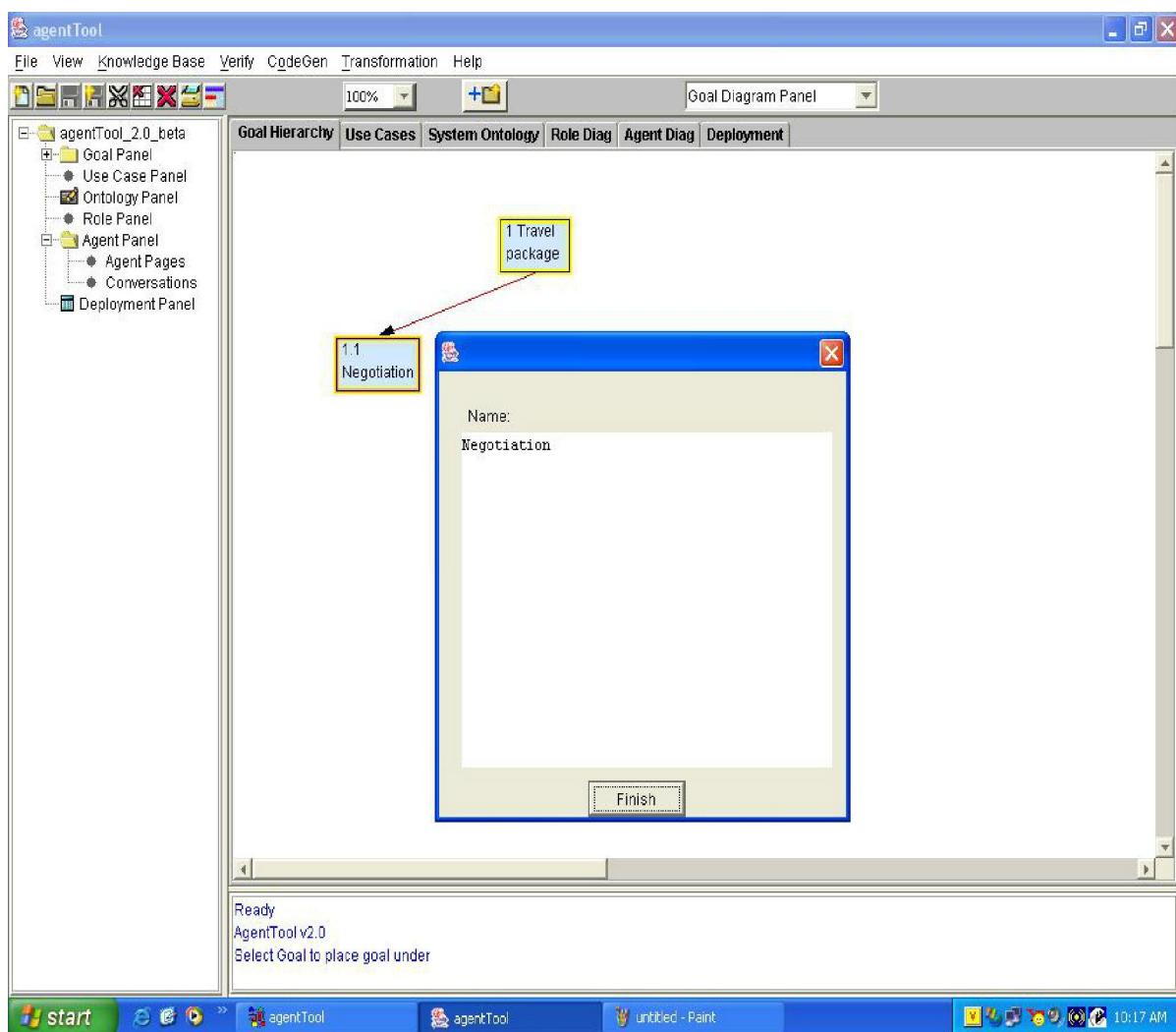
Để biểu diễn cây phân cấp đích trong agentTool, trước hết phải biểu diễn đích tổng thể của hệ thống bằng cách nhấn nút Add Goal trên thanh công cụ trong Goal Panel. Đích tổng thể sẽ được đánh số thứ tự là 1 và người phát triển sẽ phải đặt tên cho đích này sao cho mô tả khái quát được mục tiêu chung của hệ thống (Hình 4.3).



Hình 4.3: Xây dựng đích tổng thể

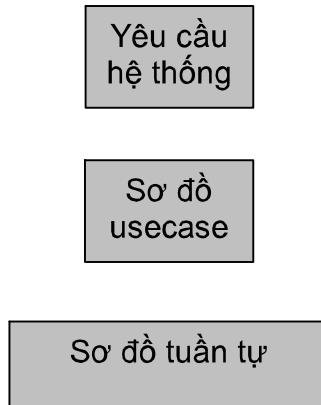
Sau khi xây dựng đích tổng thể, người phát triển hệ thống có thể biểu diễn các đích con bằng cách nhấp vào đích cha và nhấp nút Add Goal, sau đó đặt tên cho goal con đó có phần giống với goal tổng thể (Hình 4.4). Quá trình này tiếp diễn cho đến khi hoàn thành toàn bộ cây phân cấp đích.

Tương ứng với mỗi đích trong sơ đồ phân cấp đích, người phát triển phải xác định rõ kiểu của đích đó. Nếu đích đó là đích bị phân hoạch (*partitioned goal*) thì người phát triển phải nhấp chuột phải vào đích đó và chọn chức năng *set partitioned*. Trong Hình 4.4, Travel package là một đích bị phân hoạch



Hình 5.4: Xây dựng đích con

Bước 2: Xây dựng use case



Use case có thể hiểu là các mô tả về hành vi mà hệ thống cần thực hiện trong một trường hợp cụ thể. Các hành vi này được xuất phát từ mong muốn của người dùng.

Mục đích của bước này là tạo ra một tập các use case và các sơ đồ dãy (sequence diagram) tương ứng nhằm hỗ trợ cho người phân tích hệ thống phát hiện được tập các role ban đầu và các đường truyền thông có thể có trong hệ thống.

Hình 4.5: Xây dựng use case

Việc sử dụng use case trong MaSE được kế thừa từ phương pháp phân tích hướng đối tượng. Có hai loại use case khác nhau dựa vào hoàn cảnh mà chúng mô tả:

- *Use case chủ động*: mô tả hành vi của hệ thống trong trường hợp lý tưởng, nghĩa là các điều kiện đều thỏa mãn.
- *Use case bị động*: mô tả hành vi mà hệ thống cần thực hiện trong trường hợp có lỗi, có ngoại lệ hoặc có sự cố nghiêm trọng.

Bước này cũng bao gồm hai bước con là tạo các use case và xây dựng biểu đồ tương tác tuần tự.

• Tạo các use case

Các use case có thể được trích ra từ nhiều nguồn khác nhau: (i) Đặc tả yêu cầu của hệ thống; (ii) Mong muốn của người dùng; (iii) Bản mẫu nhanh (nếu có). Về nguyên tắc, mỗi một đích được xác định trong Bước 1 sẽ tương ứng với một use case, ngoại trừ các đích bị phán hoạch (vì đối với các đích này, khi hoàn thành các use case của các đích con thì bản thân nó cũng được hoàn thành).

Trong các use case tương ứng của mỗi đích cha, dãy hành động thuộc use case của đích con sẽ được coi là một hành động đơn, nghĩa là nó được xem tương đương với một hành động đơn bình thường khác và cũng được biểu diễn bằng một sự kiện đầu vào và một hành động được thực hiện.

Các use case ứng với các đích thuộc lá của cây đích sẽ được trích dẫn trước, sau đó ngược dần lên phía gốc của cây đích và dừng lại khi gặp một đích bị phân hoạch. Đích bị phân hoạch sẽ không cần use case bởi vì use case tương ứng với nó chính là phép hợp đơn giản của các use case tương ứng với các đích con của nó mà không cần bổ sung thêm bất kì sự kiện hay hành động nào.

- **Xây dựng biểu đồ tuần tự**

Một biểu đồ tuần tự mô tả một dãy các sự kiện diễn ra giữa các role đã được xác định trong các use case. Nó được xây dựng dựa trên các role và quan hệ tương tác giữa các role này với nhau. Một biểu đồ tuần tự bao gồm:

- Các role liên quan đến use case cần biểu diễn, các role này được đặt phía trên của biểu đồ.
- Các đường nối từ các role thẳng xuống dưới là các đường biểu thị cho thời gian hoạt động của các role.
- Các mũi tên nối từ role này đến role kia biểu thị một tương tác giữa hai role, theo chiều mũi tên. Nhắn kèm theo mũi tên sẽ biểu thị tên của sự kiện tương tác đó.
- Tuần tự các sự kiện xảy ra trong use case sẽ là tuần tự các mũi tên đi từ trên xuống dưới.

Việc chuyển từ một use case sang một biểu đồ tuần tự có thể tiến hành trực tiếp như sau. Mỗi use case có thể tương ứng với một biểu đồ tuần tự. Tuy nhiên, trong một số trường hợp mà use case quá phức tạp, nó có thể cần đến nhiều hơn một biểu đồ dãy để mô tả hoạt động. Trong trường hợp này, cũng có thể tách use case thành các use case nhỏ hơn và đơn giản hơn, mỗi use case chỉ cần một biểu đồ dãy như trường hợp lí tưởng ban đầu.

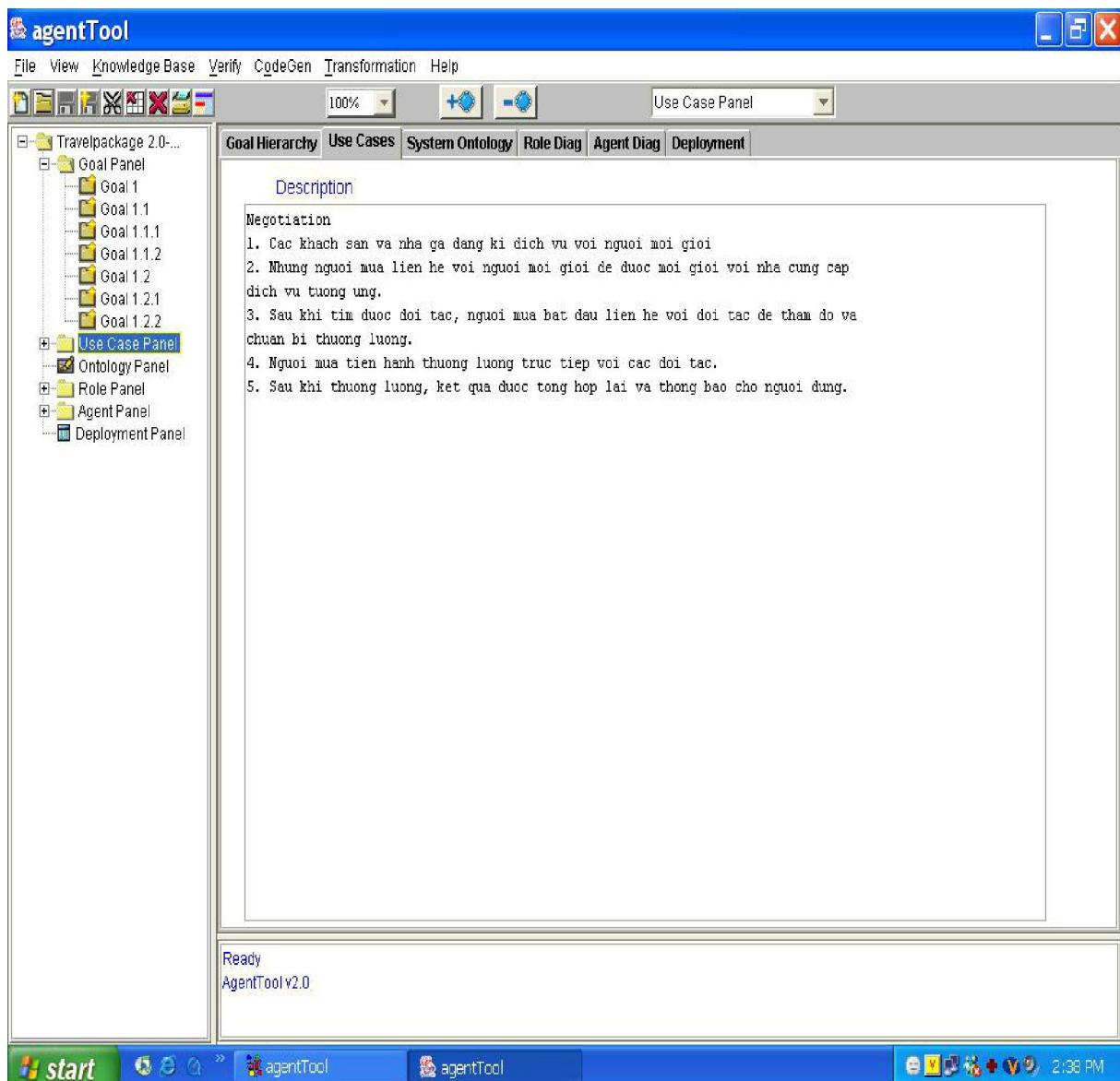
Việc xây dựng biểu đồ tuần tự được bắt đầu bằng việc xác định các role cần thiết phải tham gia vào biểu đồ. Role có thể hiểu là một tập các nhiệm vụ có liên quan chặt chẽ đến nhau mà các agent trong hệ thống cần đảm nhiệm để đạt tới đích. Mỗi role có thể thực hiện một hay nhiều đích của hệ thống.

Để xác định các role, người phát triển có thể sử dụng kỹ thuật trích danh từ. Từ các use case đã có trong bước trước, người phát triển sẽ tiến hành duyệt và tìm ra các danh từ chỉ các đối tượng có chức năng cụ thể nào đó trong use case đó. Tiếp theo, người phát triển sẽ xem xét lại danh mục các danh từ này và loại đi các

danh từ chỉ cùng một đối tượng. Các danh từ còn lại sẽ là các role sẽ có mặt trong sơ đồ tuần tự tương ứng với use case đó. Sau khi đã có các role thì trong bước tiếp theo người phát triển sẽ tiến hành biểu diễn tuần tự các sự kiện của sơ đồ tuần tự trong AgentTool.

Biểu diễn use case và các sơ đồ tuần tự trong agentTool

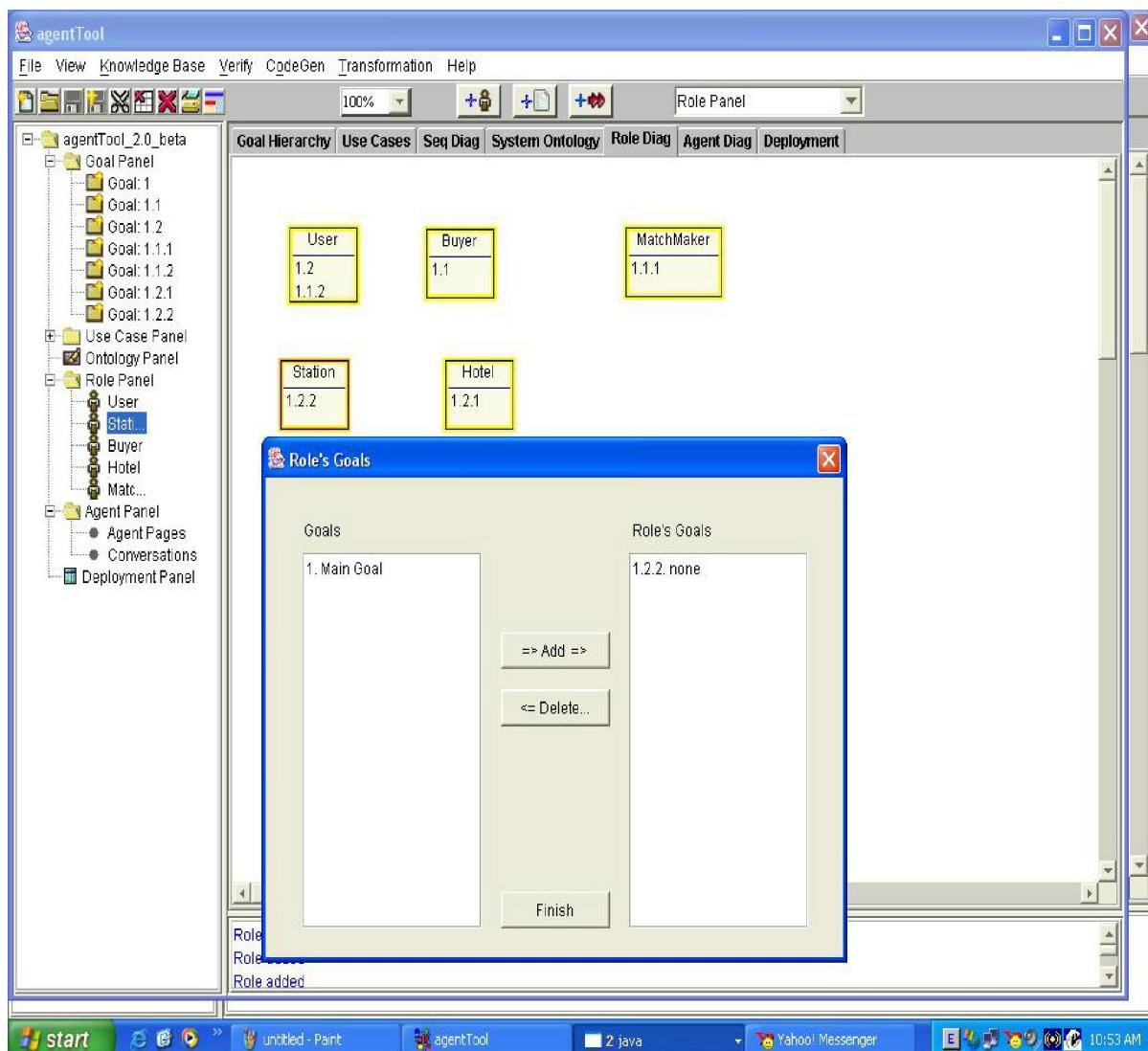
Người phát triển hệ thống sử dụng Use Case Panel trong agentTool để xác định các use case cần có của hệ thống và được biểu diễn như trong Hình 4.6.



Hình 4.6: Biểu diễn Use case

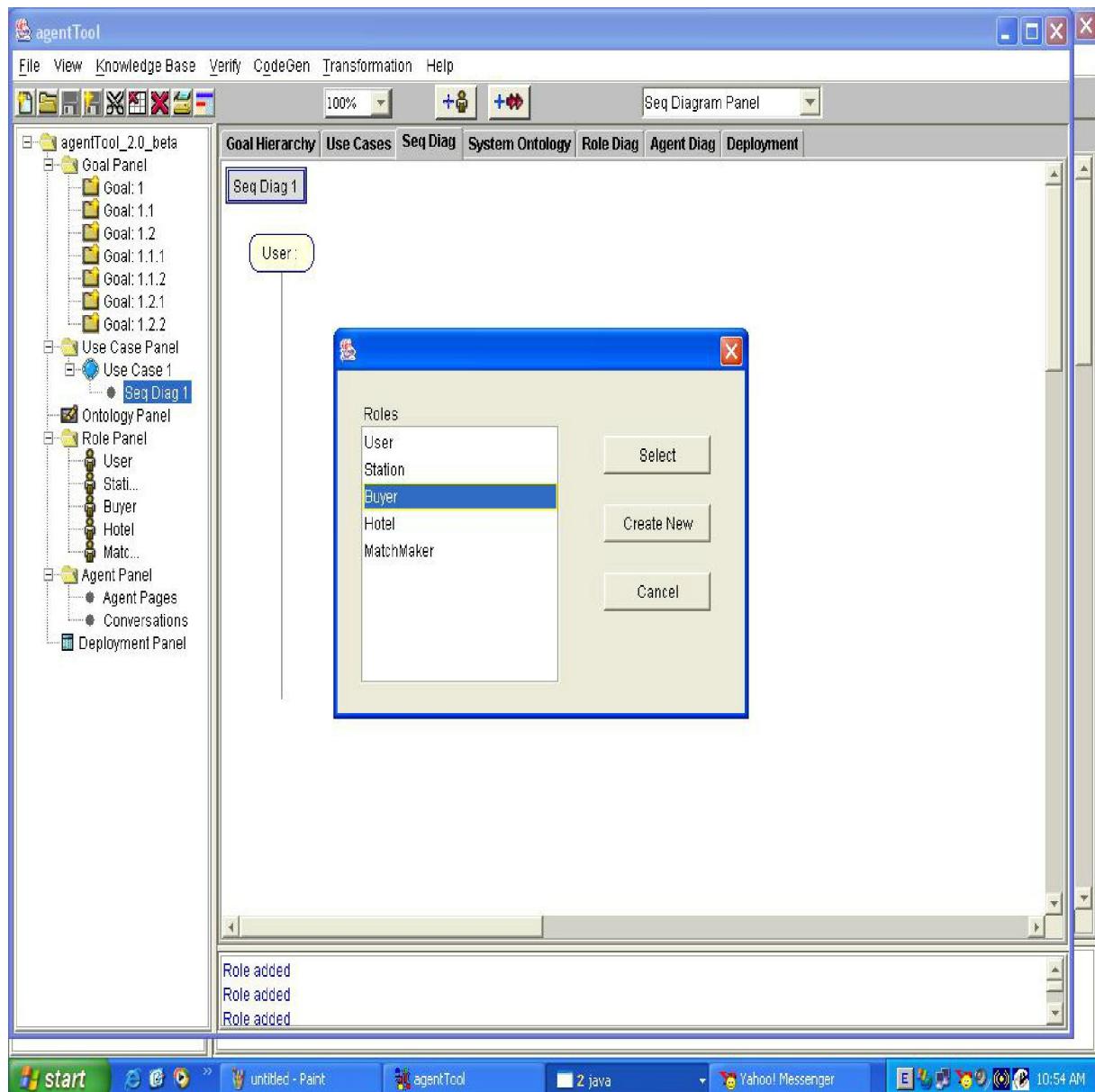
Mỗi use case sẽ có các sơ đồ tuần tự tương ứng. Để biểu diễn được các sơ đồ tuần tự này, hệ thống cần phải có các role, chính là các thành phần tham gia trong sơ đồ tuần tự. Việc xác định các role được thực hiện trong Role panel theo Hình 4.7. Tương ứng với mỗi role sẽ thực hiện một hoặc nhiều goal.

Trong Hình 4.7, các role được biểu diễn dưới dạng các bảng chữ nhật có hai phần. Phần trên là tên role, phần dưới là số thứ tự các goal tương ứng mà role đó đảm nhiệm.



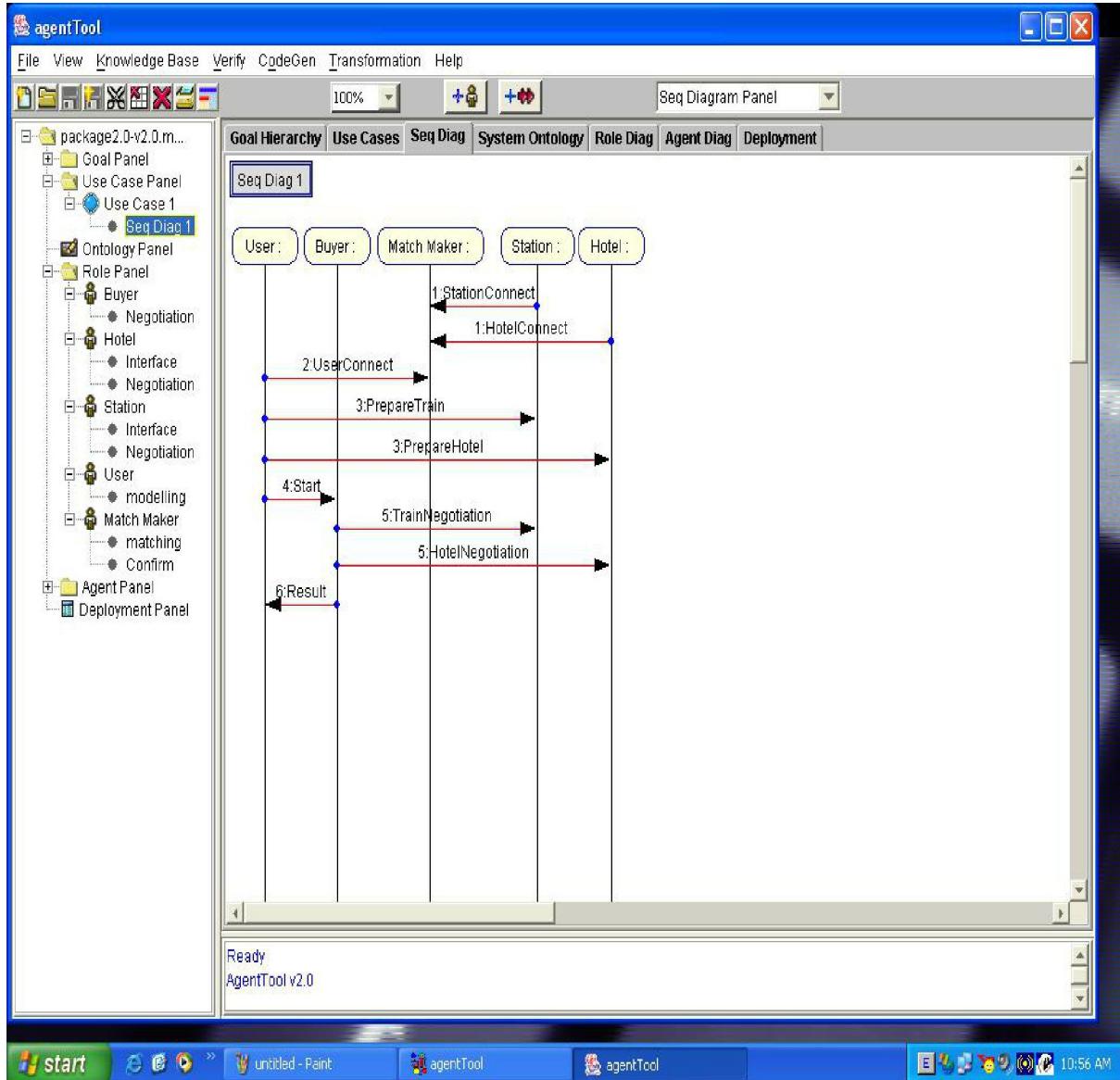
Hình 4.7: Biểu diễn Role

Các role xác định trong bước này chỉ có ý nghĩa tạm thời. Sơ đồ role sẽ được biểu diễn đầy đủ và chi tiết trong các bước sau. Dựa trên các role này, người phát triển sẽ lần lượt biểu diễn các sơ đồ tuần tự như trong Hình 4.8 và Hình 4.9 bằng cách sử dụng hai nút Add Goal và Add Role trên thanh công cụ.



Hình 4.8: Biểu diễn sơ đồ tuần tự

Mỗi sơ đồ tuần tự sẽ bao gồm các role cùng với các protocol. Các protocol chính là các tương tác giữa các thành phần. Một sơ đồ tuần tự hoàn chỉnh có dạng như trong Hình 4.9.



Hình 5.9: Sơ đồ tuần tự hoàn chỉnh

Bước 3: Xây dựng ontology

Như đã trình bày trong Chương 4 của tài liệu, ontology có thể xem là một tập các khái niệm để biểu diễn thông tin và tri thức trong trao đổi giữa các agent. Nói cách khác, ontology là một hệ tri thức chung giữa các agent, bao gồm các khái niệm được dùng trong một miền tri thức nhất định. Theo quan điểm MaSE, ontology là tập các khái niệm có thể sử dụng như là các tham số chứa trong các thông điệp trao đổi giữa các agent.

Trong MaSE, xây dựng ontology bao gồm bốn bước chính ([6],[8]):

- *Xác định mục đích và phạm vi của ontology*
- *Thu thập dữ liệu*
- *Xây dựng ontology khởi đầu*
- *Hoàn thiện ontology*

Kết quả của quá trình này là một sơ đồ phân cấp các khái niệm có thể đáp ứng yêu cầu hoạt động của hệ thống. Khi áp dụng kỹ thuật này trong các hệ thống tích hợp thông tin thì cần có những thay đổi cho phù hợp. Các bước tiến hành xây dựng ontology được cho trong Hình 4.10.

Ontology được sử dụng trong hầu hết tất cả các bước tiếp theo của quá trình phân tích thiết kế hệ phần mềm đa agent. Các tham số truyền qua lại giữa các agent trong bước xây dựng được xác định một cách tường minh bởi các kiểu dữ liệu cơ bản của Java hoặc các kiểu được định nghĩa trong ontology của hệ thống. Chính vì vậy, ở các bước sau của phần thiết kế, ta sẽ sử dụng ontology và qua đó hiệu chỉnh lại ontology nếu thấy cần thiết. Sau đây, ta sẽ xem xét cụ thể từng bước của quá trình xây dựng ontology.

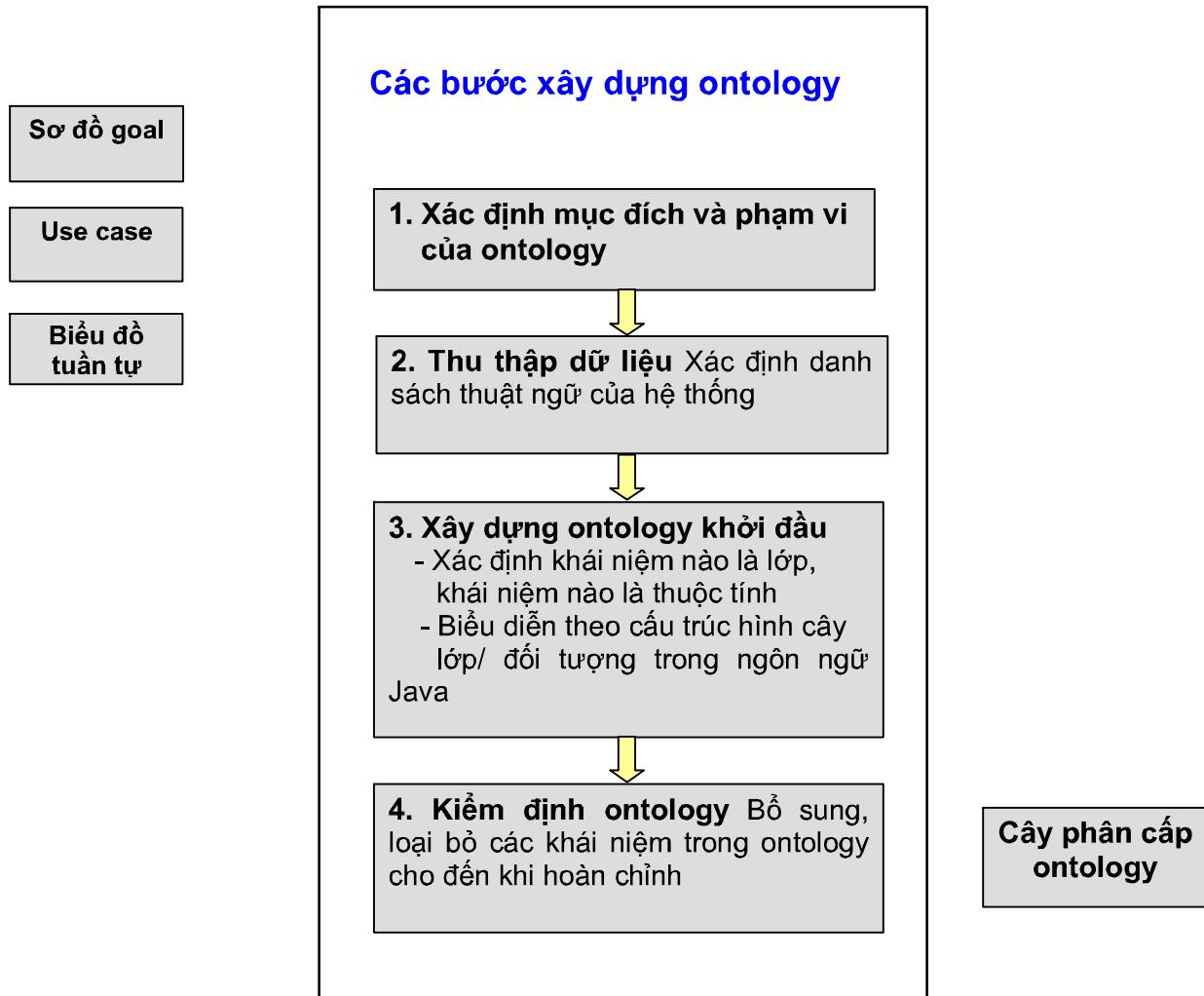
Bước 1: Xác định mục đích và phạm vi của Ontology

Bước thứ nhất trong việc xây dựng ontology là xác định được lĩnh vực mà ontology cần biểu diễn. Nghĩa là người phân tích phải trả lời câu hỏi tại sao ontology được xây dựng và phạm vi ý định sử dụng ontology của hệ thống. Khi một hệ thống kế thừa ontology từ một hệ thống khác, ontology được kế thừa đó phải được mô tả chi tiết và chính xác phạm vi biểu diễn của nó. Từ đó, người phân tích so sánh phạm vi này với phạm vi lĩnh vực của hệ thống mình đang xây dựng:

- Nếu phạm vi của mình rộng hơn thì phải bổ sung thêm các khái niệm
- Nếu phạm vi hẹp hơn thì phải loại bỏ các khái niệm không cần thiết ra khỏi ontology được kế thừa.

Để xác định phạm vi của ontology cần thực hiện các bước như sau:

- a. Xem xét toàn bộ các use case đã được mô tả trong Bước 2 và sơ đồ đích của hệ thống trong Bước 1;
- b. Xác định toàn bộ các thông tin và kiểu dữ liệu mà hệ thống sẽ dùng, đồng thời xác định được mức độ chi tiết cần mô tả của các kiểu dữ liệu này.

**Hình 5.10:** Các bước xây dựng ontology

Người phân tích có thể sử dụng kỹ thuật khoanh vùng và thu hẹp dần các miền tri thức để xác định phạm vi của ontology. Để xác định *mức độ chi tiết* của các tri thức và khái niệm, người phân tích tiếp tục sử dụng các kỹ thuật:

- Phân rã miền tri thức ban đầu thành các miền con cho đến khi không thể phân rã thêm nữa.
- Khoanh vùng các miền tri thức có liên quan đến bài toán và loại bỏ các vùng không liên quan.

Như vậy bước này tương ứng với pha *đặc tả* trong vòng đời phát triển ontology tổng quát đã trình bày trong Chương 3.

Bước 2: Thu thập dữ liệu

Sau khi xác định được các loại dữ liệu và mức độ chi tiết của chúng, người thiết kế phải lập ra một danh sách liệt kê tất cả các danh từ có mặt trong cây phân cấp đích (đã có ở bước xác định Goals) và các use case. Danh sách này được gọi là “*danh sách thuật ngữ*” của hệ thống.

Từ tập thuật ngữ này, người thiết kế phải chọn ra các danh từ có thể biểu diễn một kiểu dữ liệu cần thiết cho hệ thống bằng cách xem xét tất cả các kiểu dữ liệu cần truyền đi trong các use-case. Bước này thực chất là tìm ra các khái niệm có trong các message và các khái niệm ở mức nhỏ hơn để các agent có thể hiểu được các message mà nó nhận được. Sau bước này ta nhận được các khái niệm cần có trong ontology.

Bước này ứng với pha *hình thành khái niệm* trong phương pháp luận xây dựng ontology tổng quát.

Bước 3: Xây dựng ontology khởi đầu

Để xây dựng được ontology khởi đầu, người thiết kế phải chuyển toàn bộ các khái niệm đã thu được trong phần trước vào tập các lớp và các thuộc tính của chúng. Vấn đề chính là làm sao xác định được khái niệm nào là lớp và khái niệm nào nên là thuộc tính, điều này phụ thuộc vào việc xác định mức độ chi tiết của mỗi khái niệm. Thông thường, các khái niệm biểu diễn mức độ chi tiết thấp nhất thì nên lấy làm thuộc tính, các khái niệm là tổ hợp của nhiều khái niệm con thì nên xem là lớp.

Tuy nhiên, không phải mỗi khái niệm đều phải mô tả đầy đủ các thuộc tính như ý nghĩa thực tế của nó. Trong ontology, chỉ các thuộc tính cần thiết để thực hiện các đích con và đích tổng thể của hệ thống mới được đưa vào làm thuộc tính cho các đối tượng tương ứng. Để xây dựng được ontology khởi đầu, người phân tích có thể lựa chọn một trong hai cách:

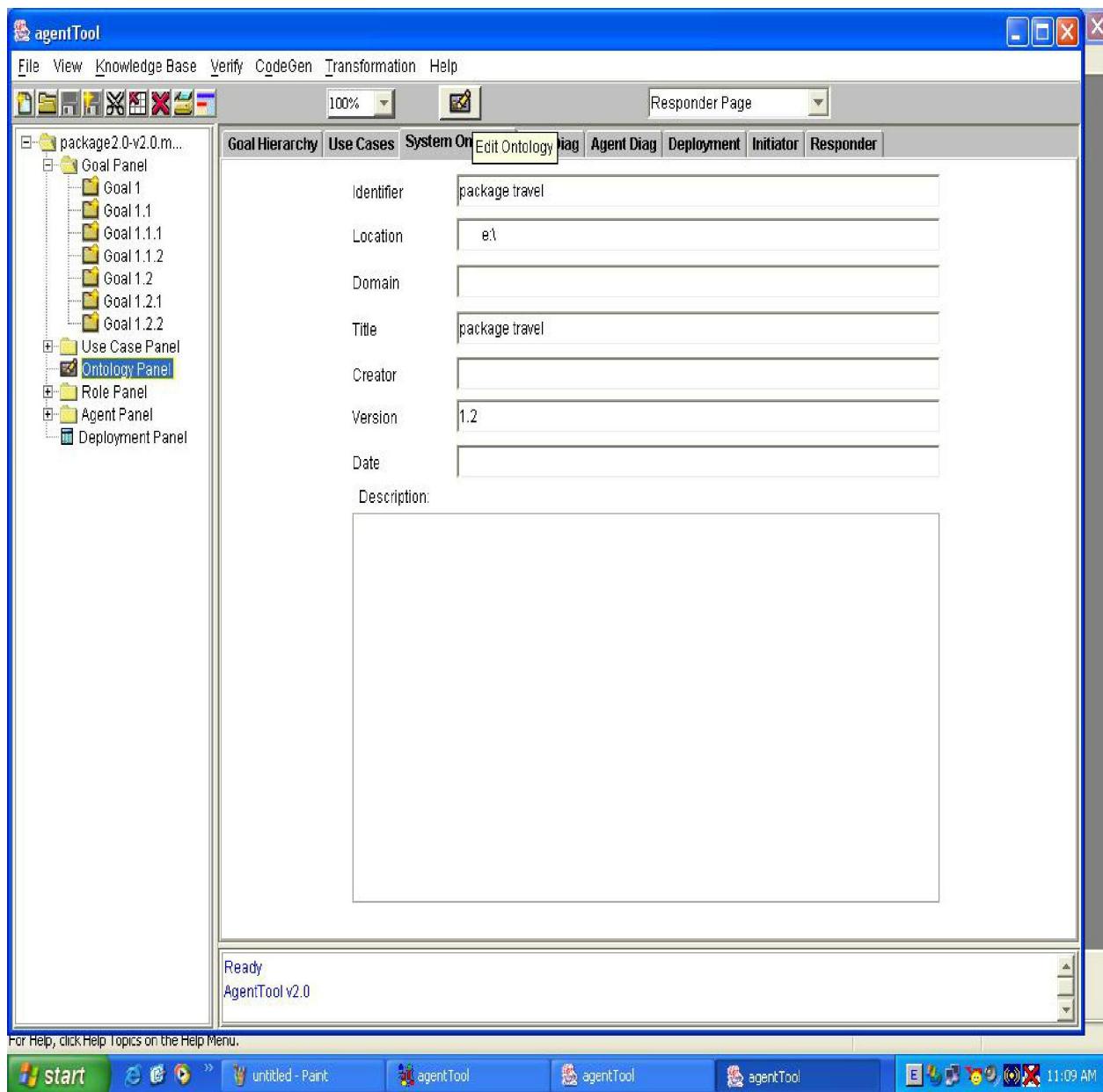
- **Kế thừa từ một ontology của một hệ thống khác:** người phân tích phải xác định rõ phạm vi biểu diễn của ontology của hệ thống được kế thừa và phạm vi tri thức của hệ thống mình đang xây dựng. Nếu phạm vi tri thức của hệ thống được kế thừa rộng hơn, thì loại bỏ các thông tin dư thừa và bổ sung các thông tin chi tiết hơn. Nếu phạm vi tri thức của hệ thống được kế thừa nhỏ hơn thì phải bổ sung các tri thức về miền lĩnh vực mới.
- **Xây dựng ontology ngay từ đầu:** người phân tích phải chuyển toàn bộ các khái niệm đã thu được trong phần trước vào tập các lớp và các thuộc tính của chúng. Thông

thường, các khái niệm biểu diễn mức độ chi tiết thấp nhất thì nên lấy làm thuộc tính, các khái niệm là tổ hợp của nhiều khái niệm con thì nên tạo thành lớp.

Bước này ứng với pha *hình thức hóa* và pha *cài đặt* trong phương pháp luận xây dựng ontology tổng quát.

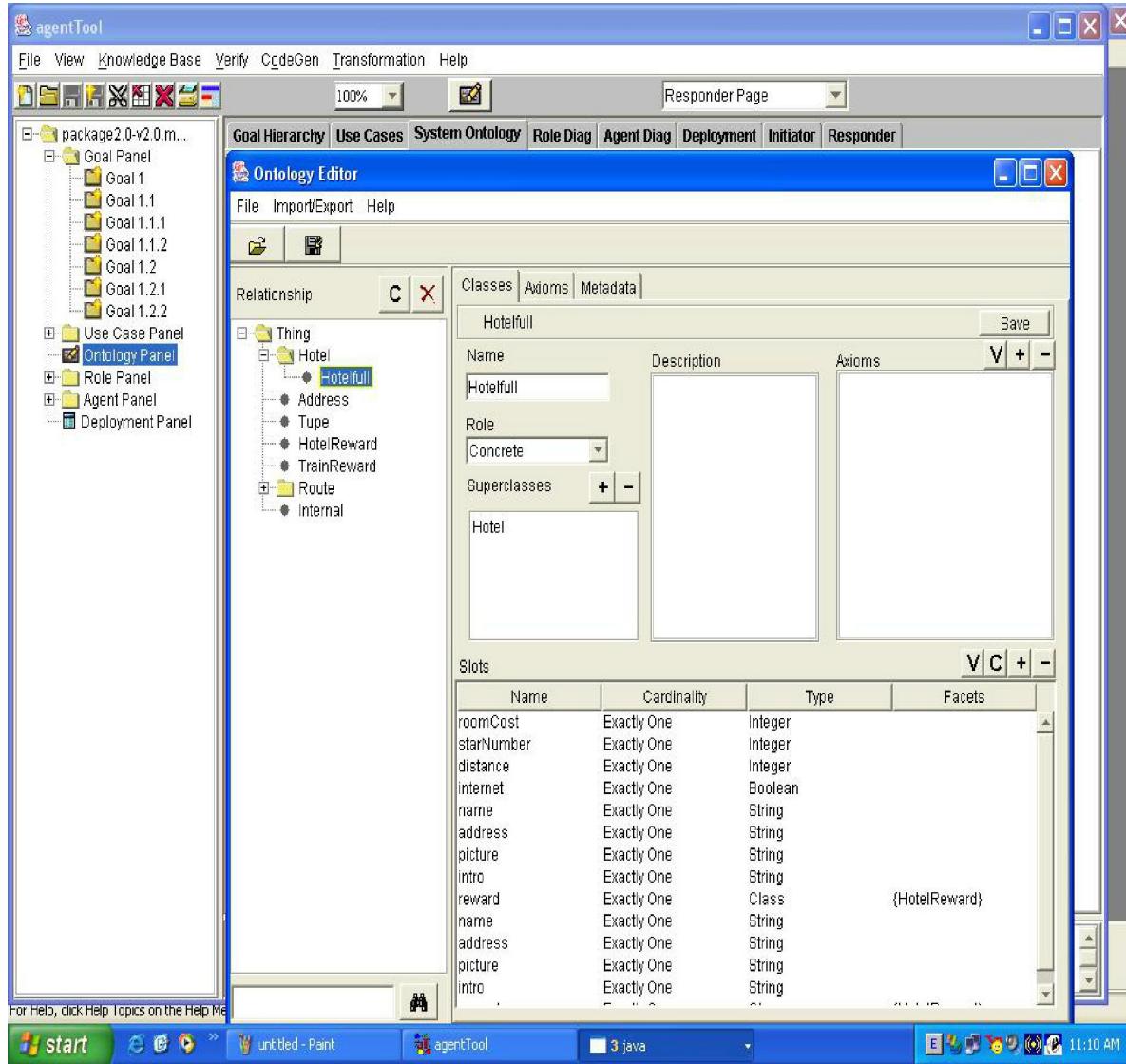
Biểu diễn ontology khởi đầu trong AgentTool

Để biểu diễn ontology khởi đầu trong agentTool, người phát triển cần sử dụng công cụ Ontology Editor được tích hợp trong Ontology panel. Hình 4.11 biểu diễn các thông tin cần xác lập cho ontology của hệ thống.



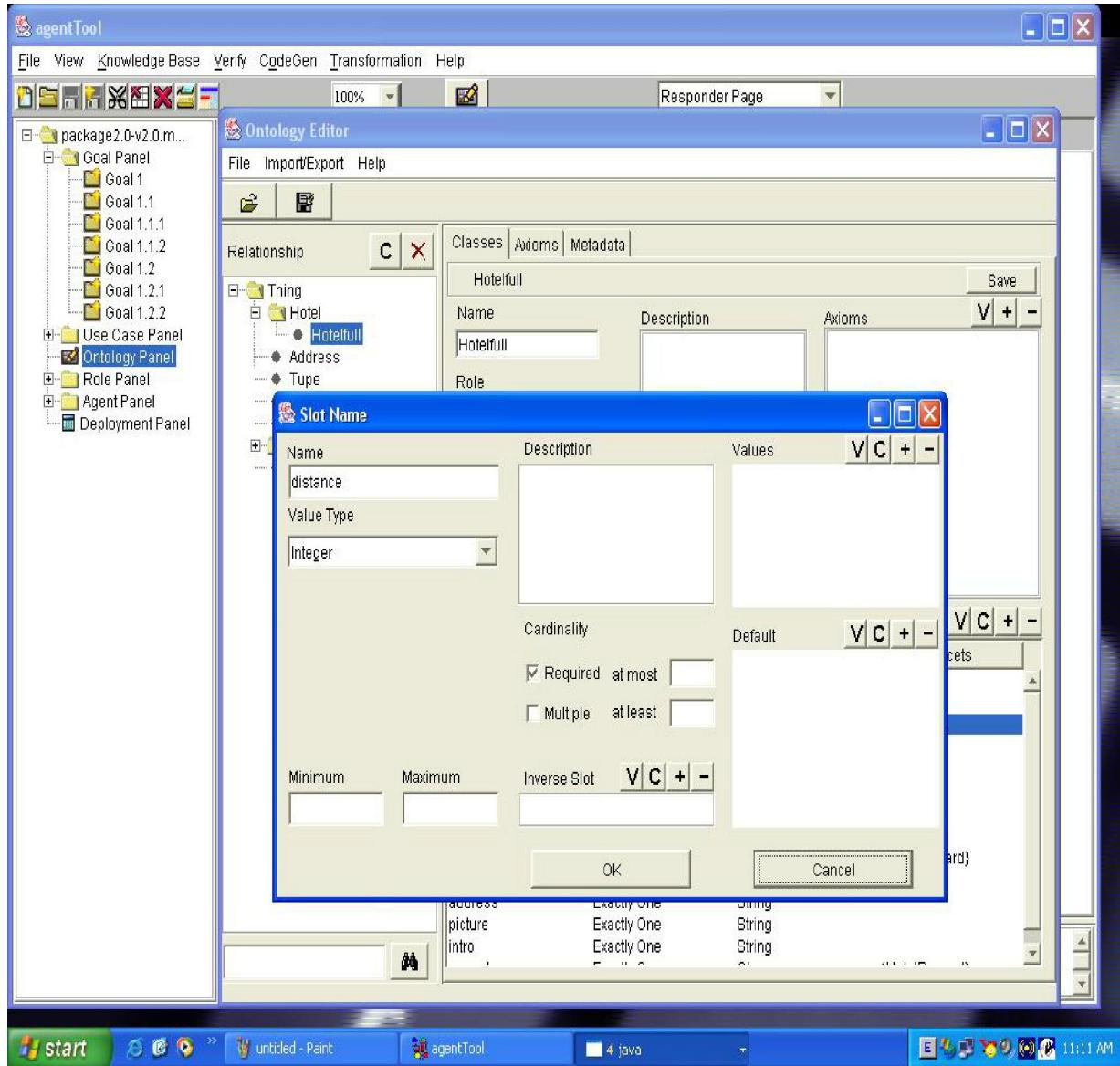
Hình 4.11: Xây dựng ontology khởi đầu (1)

Để bắt đầu biểu diễn các lớp, các thuộc tính trong ontology khởi đầu, người phát triển cần chọn nút Edit Ontology trên thanh công cụ. Khi đó, màn hình Ontology Editor sẽ hiện ra phía trên của AgentTool như trong Hình 4.12 sau đây:



Hình 4.12: Xây dựng ontology khởi đầu (2)

Hình 4.13 biểu diễn việc định nghĩa một Slot trong Ontology Editor (mỗi slot thường là một thuộc tính của một lớp trong ontology).



Hình 4.13: Xây dựng ontology khởi đầu (3)

Bước 4. Kiểm định ontology

Trong bước cuối cùng này, người phân tích phải chứng thực được rằng ontology vừa xây dựng đã thỏa mãn yêu cầu của hệ thống bằng cách xem lại toàn bộ các tình huống đã được mô tả trong các use case và sơ đồ tuần tự (sequence diagram):

- Nếu phát hiện thấy một thiếu sót nào đó thì khái niệm tương ứng phải được bổ sung ngay vào ontology;
- Nếu phát hiện có khái niệm nào không cần thiết phải loại bỏ ngay ra khỏi ontology.

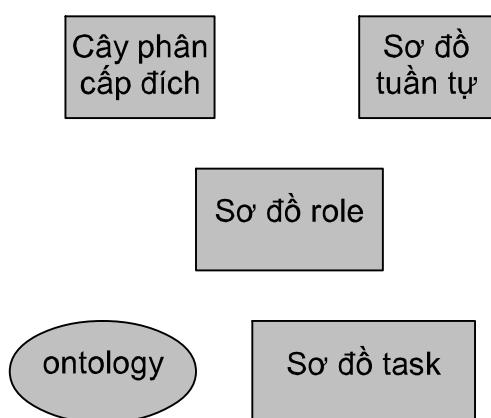
Quá trình này được lặp lại hoặc cho đến pha cài đặt hệ thống hoặc cho đến khi người thiết kế thấy rằng, ontology đã hoàn toàn thỏa mãn yêu cầu hoạt động của hệ thống.

Bước 4 tương ứng với pha *bảo trì* trong phương pháp luận xây dựng ontology tổng quát đã nói đến trong Chương 3.

Kết quả của bước 4 này là một cây phân cấp ontology hoàn chỉnh, có thể đáp ứng được yêu cầu sử dụng trong các bước tiếp theo của quá trình phân tích thiết kế và đảm bảo đầy đủ các khái niệm cần cho tương tác của hệ đa agent.

Những bước xây dựng ontology theo phương pháp luận MaSE là khá rõ ràng và tách biệt. Tuy nhiên, trên thực tế các bước này đều nằm trong vòng đời phát triển ontology nên có thể được thực hiện lồng vào nhau. Mặt khác, do đặt bước xây dựng ontology trong vòng đời phát triển chung của hệ thống nên nếu có thay đổi trong các bước đầu tiên như Xác định yêu cầu, Xác định cây phân cấp đích, Xây dựng tập use case và các biểu đồ dây thì toàn bộ quá trình xây dựng ontology cũng phải thay đổi theo. Và ngược lại, nếu cây phân cấp ontology thay đổi thì các bước sau của phân tích thiết kế cũng cần thay đổi cho phù hợp.

Bước 4: Xây dựng sơ đồ role



Role là khái niệm dùng để chỉ các thành viên (đối tượng) thực hiện một (hoặc một số) vai trò nhất định trong hệ thống.

Mục tiêu của bước này là chuyển tập các đích của hệ thống vào tập các role cùng với các nhiệm vụ của nó. Thông thường, việc chuyển đổi từ đích sang role là tương ứng 1-1, nghĩa là mỗi role thực hiện một đích.

Hình 4.14: Xây dựng sơ đồ role

Tuy nhiên, trong một số trường hợp, một role có thể tương ứng với nhiều đích khi các đích này có quan hệ chặt chẽ hoặc gần giống nhau. Trong bước này, người phát triển hệ thống sẽ phải xem xét lại các role đã có trong bước xây dựng sơ đồ tuần tự, thêm, bớt, sửa đổi cho phù hợp.

Các giao tiếp với bên ngoài sẽ được xây dựng thành một role riêng biệt và hoạt động tương tự như một tương tác từ một nguồn tài nguyên bên ngoài với phần còn lại của hệ thống. Các thành phần được coi là nguồn tài nguyên bên ngoài bao gồm: cơ sở dữ liệu, các file, hệ thống bổ sung và con người. Sau khi xác định được các role phù hợp của hệ thống, phải xác định được tập các giao thức giao tiếp ban đầu giữa các role này. Các tương tác này được trích từ các use case của Bước 2.

Nếu giữa các role trong biểu đồ dãy có một hoặc một số sự kiện liên quan với nhau thì giữa hai role tương ứng trong biểu đồ này sẽ có một giao thức tương tác, bên khởi đầu trùng với bên khởi đầu các sự kiện trong use case. Sơ đồ role ban đầu này sẽ được chi tiết hóa bằng cách gán các task cho các role. Task là một nhiệm vụ cụ thể, chi tiết mà các role phải thực hiện nhằm đạt được đích mà nó có trách nhiệm phải thực hiện.

Thông thường, mỗi đích nên cho tương ứng với một task, bởi vì muốn đạt được một đích, thì ít nhất một task cần phải hoàn thành. Trong trường hợp đặc biệt, đối với các đích phức tạp thì có thể cần đến hơn một task để giải quyết nó. Nên tách đích đó thành các đích bé hơn, đơn giản hơn sao cho một task đã có thể giải quyết được. Dựa vào các đích mà mỗi role phải thực hiện, các task được tạo ra bằng cách trả lời câu hỏi “*Role sẽ thực hiện các đích đó như thế nào?*”. Sau khi các task được gán vào các role xác định, các giao thức giữa các role trong sơ đồ role ban đầu sẽ được xác định cụ thể bởi các task liên quan.

Biểu diễn sơ đồ role trong AgentTool

Trong agentTool, người phát triển hệ thống cần biểu diễn các task tương ứng cho mỗi role (dạng hình ô van) và các tương tác giữa các task đó. Sơ đồ role đã gán task trong agentTool được biểu diễn như trong Hình 4.15.

Sơ đồ bên trong task

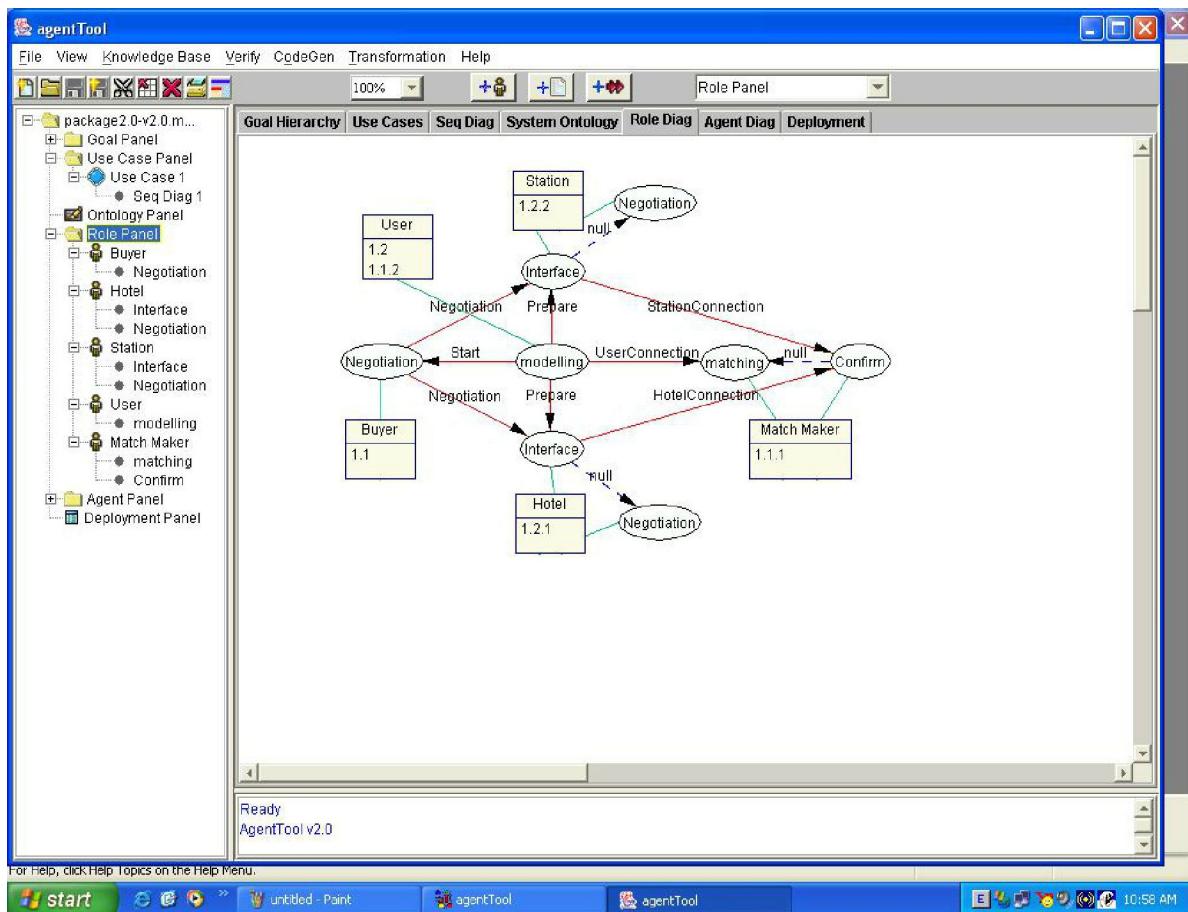
Sau khi xác định được các role, các task sẽ được gán cho mỗi role để mô tả cách mỗi role cư xử và hoạt động nhằm đạt được đích mà nó hướng tới. Các role có thể hoạt động với hai loại tương tác:

- (i) *Tương tác trong*: các tương tác giữa các task trong cùng một role;
- (ii) *Tương tác ngoài*: các tương tác giữa các task của các role khác nhau.

Trong sơ đồ role chi tiết, các tương tác trong được biểu diễn bằng các mũi tên nét đứt, còn các tương tác ngoài được biểu diễn bằng mũi tên nét liền. Các tương tác này sẽ định

hướng cho việc thiết kế các conversation trong pha thiết kế sau này. Có hai loại task khác nhau:

- Task *chủ động* là task tự động bắt đầu khi role được sinh ra. Task này không cần điều kiện kích hoạt ban đầu, nó tự động đi đến trạng thái ban đầu và thực hiện các hành động cho đến khi gặp trạng thái kết thúc hoặc role tương ứng của nó bị huỷ bỏ.
- Task *bị động* là task chỉ được thực hiện khi có sự kiện kích hoạt từ bên ngoài (thường là một thông điệp từ task khác). Task này hoạt động cho đến khi gặp trạng thái kết thúc hoặc có một task khác yêu cầu nó kết thúc.



Hình 4.14: Sơ đồ role có task

Hoạt động bên trong của các task được mô tả theo sơ đồ máy trạng thái hữu hạn (finite state machine), bao gồm các trạng thái và các chuyển tiếp giữa các trạng thái. Tại các trạng thái, các task được thực hiện thông qua các hàm cụ thể hoặc chờ cho đến khi một sự

kiện nào đó xảy ra. Cú pháp của các trạng thái chuyển tiếp là tuỳ thuộc vào kiểu tương tác mà chuyển tiếp đó tham gia là tương tác trong hay tương tác ngoài.

Các chuyển tiếp tham gia vào tương tác được mô tả theo cú pháp:

<trigger><[guard]><^transmissions>

Nghĩa là, nếu có sự kiện *trigger* xảy ra mà nó đang giữ điều kiện *guard* thì nó sẽ gửi đi các thông điệp *transmissions* và chuyển sang trạng thái tiếp theo.

Trong các sơ đồ task, có một số trường hợp các điều kiện chuyển tiếp đồng loạt được thoả mãn, nghĩa là các chuyển tiếp có thể diễn ra đồng thời thì thứ tự ưu tiên các chuyển tiếp cần thực hiện được xác định như sau:

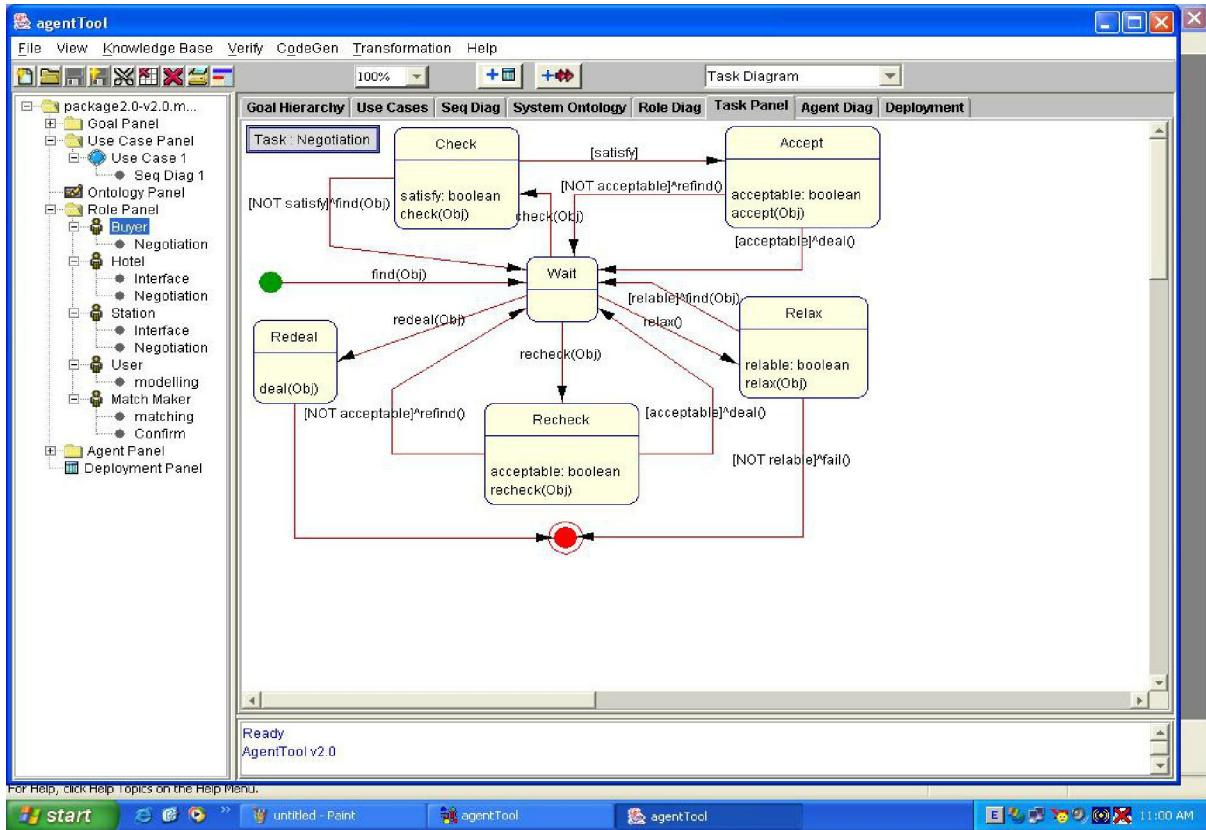
- Chuyển tiếp có điều kiện *[guard]* là một sự kiện thuộc loại tương tác trong;
- Chuyển tiếp có *transmission* là một sự kiện thuộc loại tương tác trong;
- Chuyển tiếp có *receive()* từ các role khác (tương tác ngoài);
- Chuyển tiếp có *send()* gửi thông điệp đến role khác;
- Chuyển tiếp chỉ có điều kiện *[guard]*.

Như vậy, ưu tiên trước hết được dành cho các chuyển tiếp có trao đổi thông tin với task khác, mức thứ hai là các tương tác thuộc loại tương tác trong, mức thứ ba là các tương tác có nhận thông điệp được ưu tiên hơn tương tác gửi thông điệp.

Kết quả của pha phân tích là một tập các role và các task tương ứng nhằm hoàn thành mục đích của hệ thống. Hoạt động bên trong và các tương tác bên ngoài giữa các task là nhân tố quan trọng cho pha thiết kế sau này.

Biểu diễn sơ đồ bên trong task

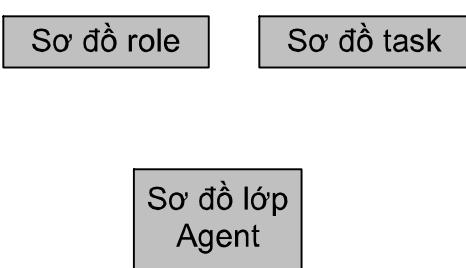
Trong AgentTool, sơ đồ bên trong các task được biểu diễn như trong Hình 4.15. Các trạng thái và các chuyển tiếp được thêm vào sơ đồ bằng cách sử dụng hai nút công cụ Add State và Add Transition trên thanh công cụ.



Hình 4.15: Sơ đồ bên trong task

4.2.3 Pha thiết kế

Bước 5: Xác định các lớp agent



Hình 4.16: Xác định các lớp agent

Trong bước này, các lớp agent sẽ được tạo ra từ các role đã được xác định trong bước 4 của pha phân tích, bao gồm:

- Phân chia các role cho các lớp agent
- Xác định các phiên hội thoại xuất hiện giữa các lớp agent.

Kết quả cần thu được của bước này là sơ đồ các lớp agent (agent classes diagram). Sơ đồ này mô tả hệ thống một cách tổng thể theo các lớp agent và các phiên hội thoại liên lạc giữa chúng.

Để đảm bảo các đích hệ thống đều được thực hiện trong pha thiết kế, mỗi role phải được đảm nhiệm bởi ít nhất một lớp agent, đôi khi cũng có các role được đảm nhiệm bởi nhiều lớp agent. Trong trường hợp một lớp agent đảm nhiệm nhiều role thì các agent của lớp đó sẽ luân phiên đảm nhiệm công việc của từng role. Trường hợp này xảy ra khi các role phục vụ cho các đích liên quan chặt chẽ với nhau hoặc có khối lượng tương tác với nhau lớn, chúng được xếp vào cùng một lớp agent nhằm giảm chi phí truyền thông. Người thiết kế có thể dễ dàng thay đổi các role giữa các lớp agent, điều này cho phép họ điều chỉnh hệ thống theo mục đích của mình như sau:

- Nếu giữa hai role có lưu lượng trao đổi thông tin lớn thì nên xếp chúng vào cùng một lớp agent để giảm chi phí truyền thông.
- Nếu cả hai role có khối lượng tính toán lớn và phức tạp thì nên xếp chúng vào hai lớp agent khác nhau để giảm khối lượng tính toán đồng thời tận dụng được khả năng xử lí đồng thời giữa các agent.

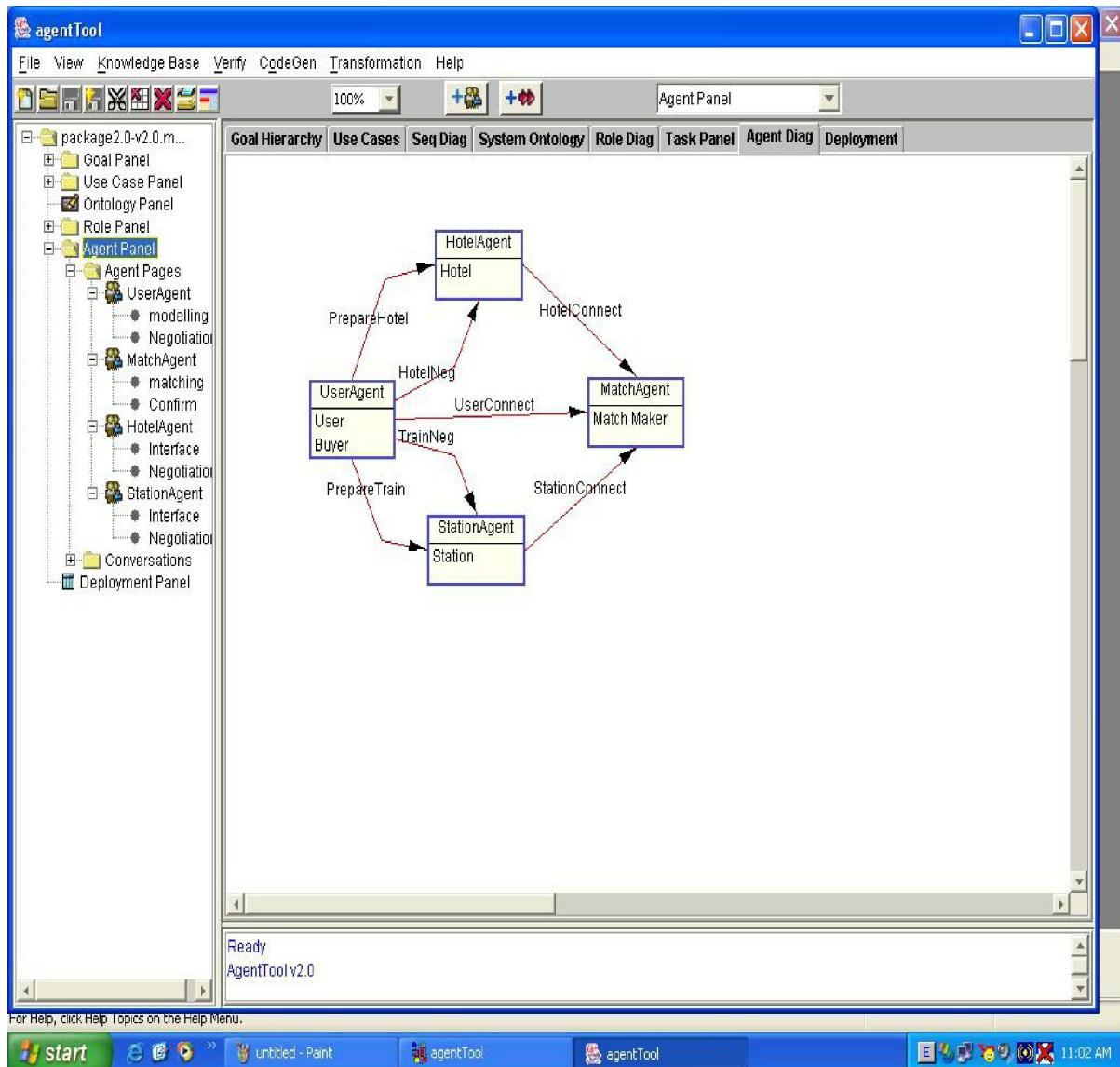
Nhiệm vụ còn lại của bước này là xác định các phiên hội thoại xuất hiện giữa các lớp agent để hoàn thiện sơ đồ lớp agent của hệ thống. Các phiên hội thoại được xác định từ các quan hệ giữa các role mà các lớp agent tương ứng cần thực hiện. Chẳng hạn, hai lớp agent A và B lần lượt thực hiện các role 1 và 2 tương ứng. Nếu giữa role 1 và 2 có một giao thức được xác định trong sơ đồ role thì giữa hai lớp agent A và B cũng xuất hiện một phiên hội thoại theo chiều tương ứng.

Các phiên hội thoại biểu diễn tương tác giữa các lớp agent diễn ra ở mức ngữ nghĩa thông qua ontology. Dựa vào ontology đã được xây dựng trong bước 3, các lớp agent có cách biểu diễn tri thức khác nhau sẽ trao đổi và hiểu được lẫn nhau dựa trên tri thức chung trong ontology.

Việc chi tiết hoá các phiên hội thoại là nội dung của bước 6 và chi tiết hoá kiến trúc bên trong của các lớp agent là nội dung của bước 7. Hai bước tiếp theo này có thể được tiến hành song song.

Biểu diễn sơ đồ lớp agent

Quá trình biểu diễn sơ đồ lớp agent được thực hiện trên Agent panel của AgentTool. Mỗi lớp agent được biểu diễn bằng một bảng chữ nhật có hai ô, trong đó ô trên biểu diễn tên lớp agent, ô dưới là role tương ứng với lớp agent đó. Các phiên hội thoại được biểu diễn bởi các mũi tên hướng từ lớp agent khởi tạo hội thoại đến lớp agent kết thúc hội thoại. Hình 4.17 mô tả một ví dụ về biểu diễn sơ đồ lớp agent trong AgentTool.



Hình 4.17: Biểu diễn sơ đồ lớp agent

Bước 6: Xây dựng các phiên hội thoại

Sơ đồ tuần tự Sơ đồ Task Sơ đồ Role

Các phiên
hội thoại

Hình 4.18: Bước xây dựng các phiên hội thoại

Một phiên hội thoại biểu diễn một phiên liên lạc giữa hai agent, nghĩa là các agent sẽ tiến hành gửi đi các yêu cầu và đáp ứng lại các yêu cầu từ phía agent kia. Trong MaSE, mọi liên lạc giữa hai agent bất kỳ đều phải thông qua các phiên hội thoại được hoạt động theo cơ chế socket và cổng (port), với các chuẩn của giao thức TCP/IP.

Nhiệm vụ của bước này là thiết kế chi tiết kiến trúc bên trong của các phiên hội thoại đã xác định được trong bước 5. Đối với mỗi phiên hội thoại, phải làm sáng tỏ được cách thức và cơ chế hoạt động của mỗi bên agent tham gia vào phiên hội thoại đó.

Mỗi phiên hội thoại bao gồm hai sơ đồ lớp truyền thông (Communication class diagram), một cho bên khởi xướng và một cho bên đáp ứng phiên hội thoại. Mỗi một sơ đồ lớp truyền thông được biểu diễn như một sơ đồ máy trạng thái hữu hạn tương tự như các task đồng thời trong Bước 4.

Performative là một khái niệm liên quan đến các thông điệp được trao đổi trong các phiên hội thoại. Nó là một tập các từ thuộc thể loại mệnh lệnh thức (lời nói mang ý nghĩa hành động), được quy định cho mỗi hệ thống. Chẳng hạn trong hệ du lịch, khi UserAgent gửi đến HotelAgent một thông điệp có performative là “*find*” thì có nghĩa rằng UserAgent yêu cầu HotelAgent tìm kiếm một khách sạn thỏa mãn các yêu cầu kèm theo trong nội dung của thông điệp đó.

Trong Bước 5, các phiên hội thoại được xác định từ các tương tác giữa các role của các lớp agent khác nhau. Do vậy, các sơ đồ lớp truyền thông cũng được xác định tương ứng với các task mà các role đó thực hiện. Thông thường, mỗi sơ đồ task sẽ tương ứng với một sơ đồ phiên hội thoại cho bên tham gia tương ứng.

Tại các trạng thái của phiên hội thoại, người thiết kế phải xác định chính xác tên các hàm, các biến và các tham số cho các hàm mà không được để một cách khái quát như các hàm và thủ tục trong các trạng thái của task.

Tại các chuyển tiếp của sơ đồ phiên hội thoại, người thiết kế phải chi tiết hóa dựa trên các chuyển tiếp trong sơ đồ task tương ứng. Cùng dựa trên cú pháp của chuyển tiếp $<receive(message)><[guard]><send(message)>$, nhưng được cụ thể hóa nội dung các thông điệp (message) dựa vào việc sử dụng ontology của hệ thống đã được thiết kế trong Bước 3.

Sau khi các thông tin từ sơ đồ task được ánh xạ vào như một phần của phiên hội thoại, người thiết kế phải kiểm tra lại để đảm bảo các điều kiện khả thi cho phiên hội thoại: mỗi message gửi đi phải có ít nhất một bên nhận và xử lý, không có vòng lặp vô hạn trong các sơ đồ trạng thái... Mỗi phiên hội thoại được coi là hợp lệ nếu nó thoả mãn các điều kiện sau.

- Trong mỗi sơ đồ của các bên không có vòng lặp vô hạn (deadlock), nghĩa là nếu trong sơ đồ có xuất hiện chu trình thì chu trình này phải có điều kiện dừng và điều kiện dừng này là có khả năng đạt được.
- Thứ hai, với mỗi lớp agent liên quan đến phiên hội thoại, tại mỗi thời điểm nó chỉ được ở trong một trạng thái xác định.
- Thứ ba là các trạng thái đều có thể được sử dụng trong một điều kiện nhất định (không có trạng thái không bao giờ được sử dụng).
- Thứ tư là mỗi thông điệp được gửi đi thì bên đối diện phải có khả năng nhận được thông điệp đó.

Việc kiểm tra các điều kiện này có thể tiến hành một cách thủ công hoặc bán tự động với sự trợ giúp của agentTool. Chẳng hạn kiểm tra theo phương pháp thủ công điều kiện một với phiên hội thoại *HotelNeg*, xuất hiện chu trình

(Wait – Check – Accept – Wait2 – Relax - Wait)

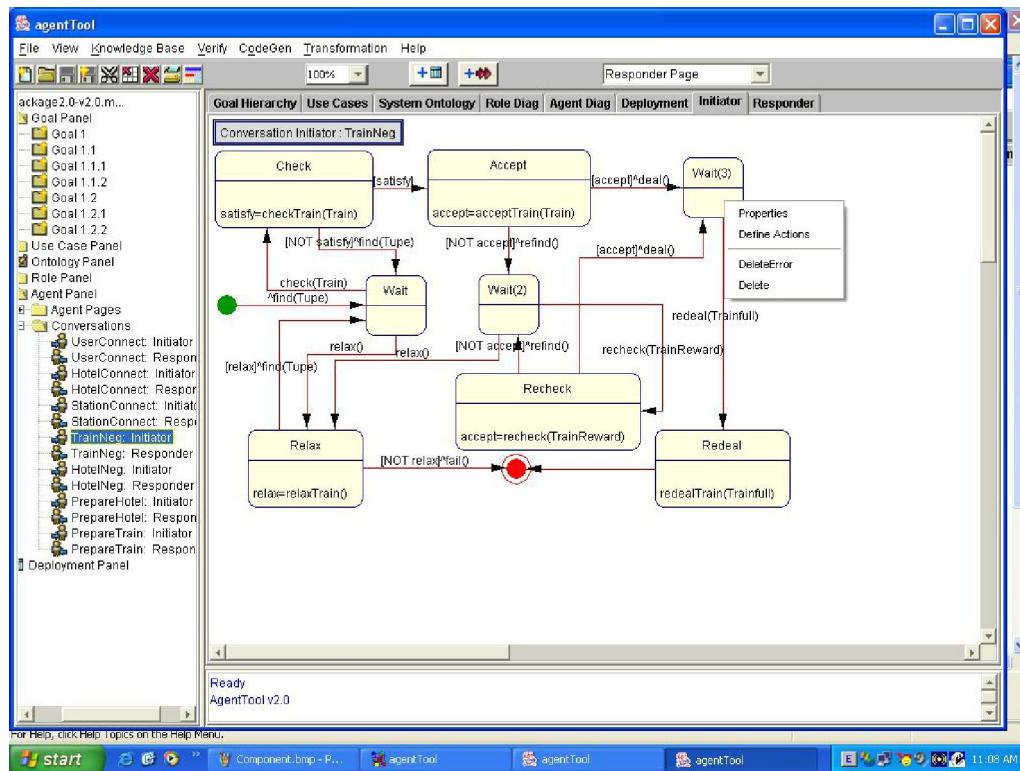
của bên *UserAgent* (bên khởi tạo phiên hội thoại này). Sau mỗi vòng lặp này, yêu cầu về các thuộc tính đều bị giảm đi một mức do *UserAgent* tự nhượng bộ, nên dãy yêu cầu của *UserAgent* là “đơn điệu giảm”. Hơn nữa, sự nhượng bộ sẽ dừng lại khi độ thoả mãn các thuộc tính thấp hơn ngưỡng nhượng bộ (*relaxThreshold*), ngưỡng này được ước lượng khi mô hình hoá sở thích người dùng. Do vậy, dãy yêu cầu của *UserAgent* sẽ đơn điệu giảm và bị chặn dưới nên sẽ dừng, nghĩa là vòng lặp không vô hạn.

Trong việc thiết kế các phiên hội thoại, người thiết kế phải đối mặt với vấn đề là nên lựa chọn theo hướng ít phiên hội thoại dạng phức tạp hoặc nhiều phiên hội thoại dạng đơn giản.

Việc sử dụng các phiên hội thoại đơn giản có ưu điểm là dễ theo dõi, quản lí và đồng thời tiết kiệm được tài nguyên mạng như kênh truyền, công, kết nối... Lý do là khi kết thúc, các tài nguyên này được giải phóng và khi cần thiết, chúng mới bị chiếm dụng trở lại. Nhưng nhược điểm của cách tiếp cận này là làm chậm tốc độ do phải tốn nhiều thời gian cho việc thiết lập và giải phóng các kết nối truyền thông.

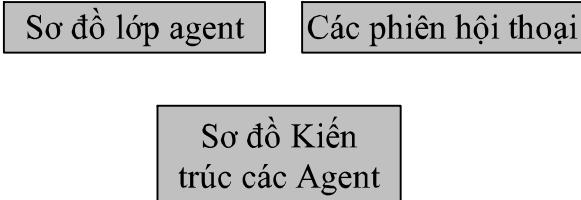
Kiểu các phiên hội thoại phức tạp cho hiệu quả ngược lại, tăng tốc độ nhưng tốn tài nguyên do nó vẫn giữ đường truyền ngay cả khi không truyền thông điệp mà đang tính toán tại các trạng thái.

Kiểu hội thoại phức tạp được sử dụng cho các phiên hội thoại *TrainNeg* và *HotelNeg*, các phiên hội thoại còn lại được thuộc kiểu đơn giản. Các phiên hội thoại được biểu diễn trong Hình 4.19.



Hình 4.19: Biểu diễn các phiên hội thoại

Bước 7: Hoàn thiện các agent



Hình 4.20: Hoàn thiện các agent

Bước này nhằm thiết kế chi tiết các tương tác bên trong (self-interaction) của mỗi lớp agent trong hệ thống. Bao gồm hai bước con: *thiết kế kiến trúc bên trong agent* và *thiết kế các thành phần* trong kiến trúc đó.

Thiết kế kiến trúc

Để xác định kiến trúc bên trong của các agent, người thiết kế có thể tự thiết kế kiến trúc của mình hoặc có thể chọn một kiến trúc có sẵn như BDI [9]. Trong trường hợp không sử dụng kiến trúc có sẵn, người thiết kế có thể xây dựng các thành phần từ các nhiệm vụ của các role mà lớp agent tương ứng đảm nhiệm. Việc này được hỗ trợ tự động bởi agentTool.

Sau khi xác định được các thành phần trong kiến trúc của mỗi agent, nhiệm vụ tiếp theo là thiết kế quan hệ giữa các thành phần này sao cho phù hợp với sơ đồ role của hệ thống đã được phân tích trong Bước 4. Để đảm bảo quan hệ giữa các thành phần thông nhất với quan hệ giữa các nhiệm vụ trong sơ đồ role, các quan hệ này sẽ được thiết kế dựa trên các tương tác giữa các nhiệm vụ trong sơ đồ role:

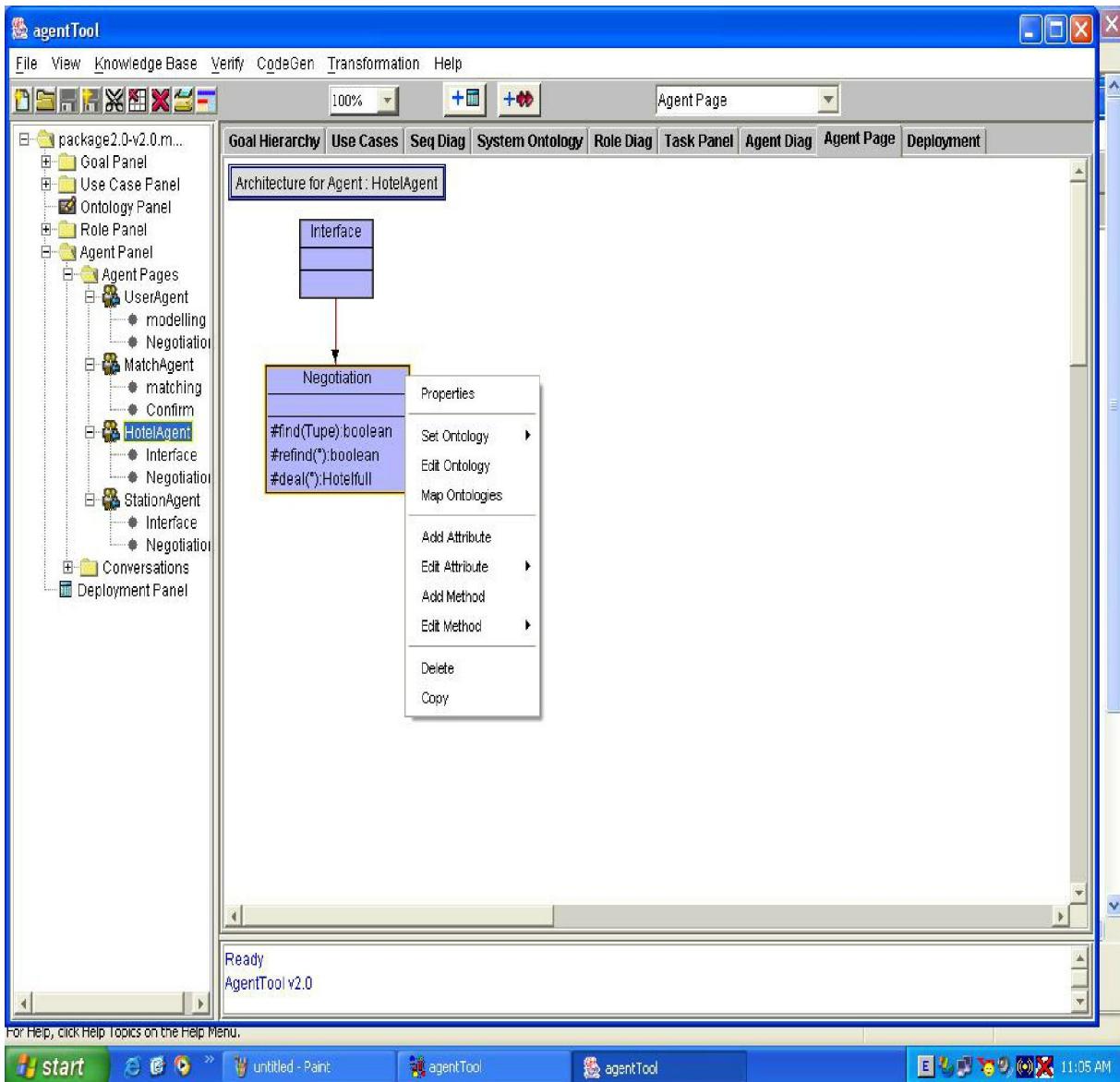
- Các tương tác giữa các nhiệm vụ sẽ trở thành một quan hệ trong kiến trúc này và được biểu diễn bằng một mũi tên nét liền, chiều của mũi tên trùng với chiều của tương tác tương ứng trong sơ đồ role (Một số tương tác bên ngoài cũng có thể trở thành các quan hệ bên trong nếu các role của các nhiệm vụ tương ứng được thực hiện bởi một lớp agent).
- Các tương tác bên ngoài giữa các role được đảm nhiệm bởi các lớp agent khác nhau sẽ trở thành các quan hệ bên ngoài.

Sau khi xác định được kiến trúc bên trong của mỗi lớp agent, nhiệm vụ của bước con tiếp theo là thiết kế chi tiết bên trong mỗi thành phần của kiến trúc này. Biểu diễn sơ đồ kiến trúc lớp agent trong AgentTool được cho trong Hình 4.21.

Thiết kế các thành phần

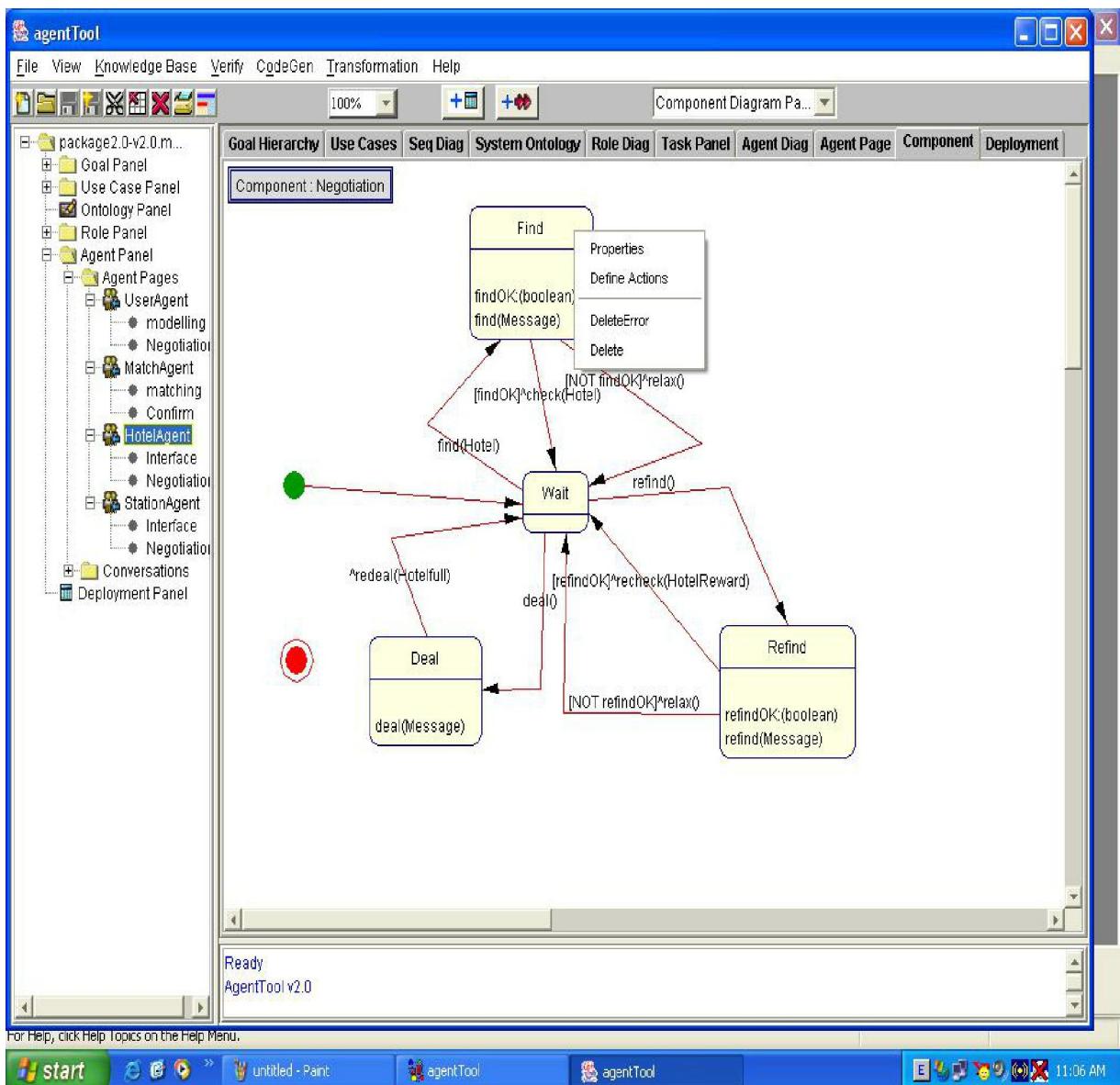
Nhiệm vụ của bước con này là thiết kế chi tiết bên trong các thành phần của kiến trúc đã được thiết kế ở bước trên. Việc này được hoàn thành bằng cách gán các thuộc tính và các hàm (thủ tục) chính cho mỗi thành phần.

Để xác định được các hàm và thuộc tính, người thiết kế phải duyệt trong tất cả các trạng thái của các phiên hội thoại mà thành phần tương ứng tham gia như trong Bước 6 đã xác định cụ thể tên các hàm cần thiết. Các hàm này sẽ trở thành tên các hàm cho mỗi thành phần được cho trong Hình 4.22.



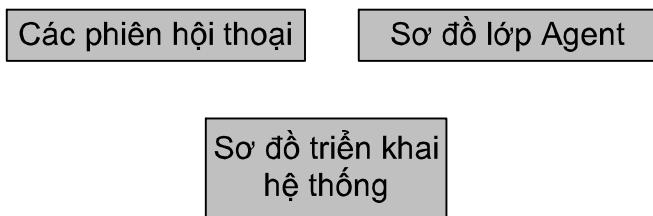
Hình 4.22: Thiết kế kiến trúc agent

Trước khi kết thúc bước thiết kế kiến trúc bên trong của agent này, người thiết kế phải hoàn thành việc thiết kế ontology riêng cho mỗi thành phần của mỗi lớp agent nếu điều đó là cần thiết. Một lớp agent cần được thiết kế thêm phần ontology riêng nếu như ontology chung của hệ thống không giúp nó biểu diễn đầy đủ được tri thức của mình.



Hình 5.20: Thiết kế thành phần

Bước 8: Triển khai hệ thống



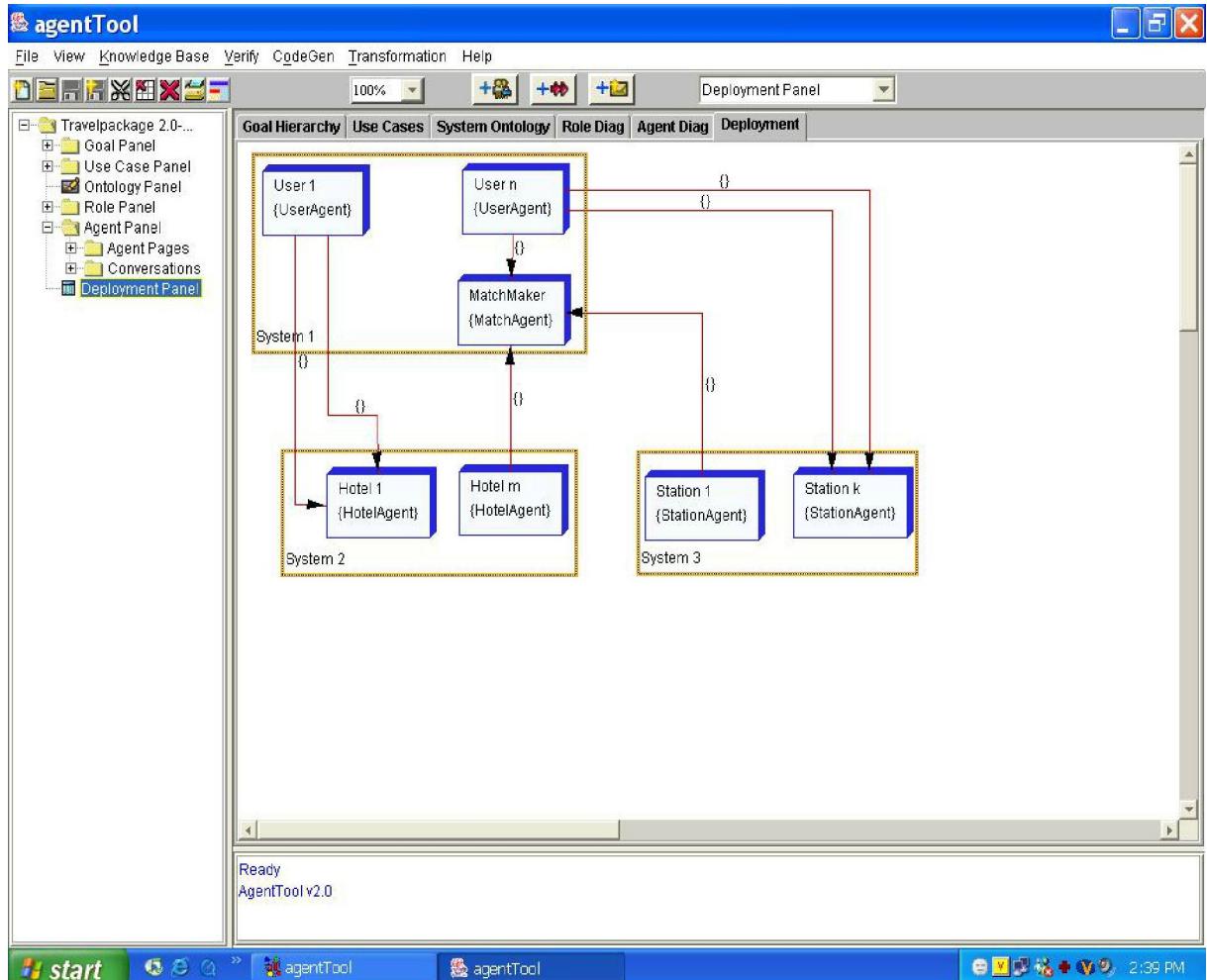
Hình 4.21: Bước triển khai hệ thống

Nhiệm vụ của bước cuối cùng này là xây dựng được sơ đồ triển khai hệ thống (Deployment diagram) nhằm mô tả số lượng, các kiểu và vị trí của các agent trong hệ thống.

Dựa vào sơ đồ này, người thiết kế có thể cấu hình hệ thống sao cho tối ưu được sức mạnh tính toán cũng như băng thông của mạng:

- Nếu chú trọng đến chi phí truyền thông, có thể xếp nhiều agent trên một máy tính, nhưng tốc độ xử lý sẽ bị chậm lại và làm mất tính ưu việt của hệ đa agent trong môi trường phân tán.
- Ngược lại, muốn giảm khối lượng tính toán tại mỗi máy, có thể xếp các agent khác nhau trên các máy khác nhau và phải tăng phần chi phí truyền thông.

Mỗi hệ thống con được hiểu là một máy chủ (server) có khả năng hoạt động liên tục trong thời gian chạy ứng dụng và được biểu diễn bằng một hình chữ nhật nét rời. Trong mỗi hệ thống con, có thể thiết kế số lượng các agent theo mục đích riêng của người thiết kế. Khi đó, các agent của các lớp agent khác nhau sẽ tương tác với nhau thông qua các phiên hội thoại giữa các lớp agent tương ứng. Các phiên hội thoại này được biểu diễn bằng các mũi tên nét liền theo đúng chiều phiên hội thoại đã thiết kế trong bước 3 của pha thiết kế. Trong AgentTool, sơ đồ triển khai hệ thống được biểu diễn như trong hình 4.22.



Hình 4.22: Sơ đồ triển khai hệ thống

4.3 Kết luận

Nội dung của chương này tập trung trình bày quy trình phát triển hệ phần mềm đa agent dựa trên phương pháp luận MaSE. Quy trình này bao gồm 8 bước và được phân thành hai pha: pha phân tích và pha thiết kế. Mục đích của pha phân tích là tiến hành xây dựng các dịch của hệ thống và dựa trên đó để xây dựng các use case, ontology và sơ đồ role. Pha thiết kế bao gồm xác định các lớp agent, xây dựng các phiên hội thoại, hoàn thiện các agent và triển khai hệ thống. Các bước trong quy trình phát triển đã được trình bày với agentTool hỗ trợ khi phát triển hệ dịch vụ du lịch.

PHẦN 2

ÁP DỤNG PHÁT TRIỂN HỆ DỊCH VỤ DU LỊCH

CHƯƠNG 5

PHÂN TÍCH HỆ DỊCH VỤ

- Bài toán dịch vụ du lịch
- Mô hình sở thích người sử dụng
- Phân tích hệ thống
- Kết luận

Chương này nhằm trình bày áp dụng quy trình phát triển trong Chương 4 trong xây dựng hệ dịch vụ du lịch **TraNeS**. Vấn đề quan trọng trước hết đối với hệ này là mô hình sở thích/nhu cầu người sử dụng trong các hệ dịch vụ sẽ được đề cập. Phản trọng tâm là áp dụng các kỹ thuật và các bước trong pha phân tích hệ đa agent vào một ứng dụng cụ thể.

5.1 Mô hình sở thích người sử dụng

Mô hình sở thích và nhu cầu người sử dụng là một trong những vấn đề then chốt của phát triển các hệ thống và đặc biệt các hệ dịch vụ thương mại điện tử. Mục này dành trình bày bài toán du lịch mà chúng tôi chọn làm mẫu cho quá trình phát triển hệ thống và sau đó xem xét vấn đề mô hình sở thích người sử dụng.

5.1.1 Bài toán dịch vụ du lịch

Phát biểu bài toán

Để thực hiện một chuyến du lịch giữa các địa điểm/thành phố trong nước, một khách du lịch muốn tự mình xây dựng một tour theo ý muốn. Anh ta muốn đặt vé và đặt chỗ ở khách sạn theo yêu cầu và sở thích của mình.

Một số vấn đề khách du lịch sẽ gặp phải:

- Không đủ thông tin về các tuyến du lịch để lựa chọn.
- Không đủ thời gian để tìm hiểu và chọn lựa tuyến phù hợp.
- Không tự quyết định để lựa chọn chuyến tàu hay khách sạn nào là phù hợp nhất với thời gian, sở thích và túi tiền của mình.
- Tốn thời gian và công sức khi phải trực tiếp đặt mua vé tàu và đặt phòng khách sạn...

Hệ dịch vụ du lịch TraNeS (Travel Negotiation Service System) được phát triển nhằm hỗ trợ khách hàng đặt vé tàu và đặt chỗ khách sạn theo sở thích và yêu cầu của họ. Chức năng của hệ TraNeS bao gồm:

- Thay mặt khách hàng thương lượng một cách tự động để tìm ra giải pháp tốt nhất dựa trên các ràng buộc của họ.
- Thay mặt các nhà cung cấp, đưa các dịch vụ đến với khách hàng một cách thuận tiện và thân thiện. Các nhà cung cấp dịch vụ bao gồm: dịch vụ tàu hoả và dịch vụ khách sạn.
- Đối với mỗi khách hàng, hệ thống phải quản lý các thông tin cá nhân, các sở thích, nguyện vọng của họ về dịch vụ mà họ mong muốn.
- Quá trình thương lượng diễn ra trong hệ thống là “trong suốt” đối với khách hàng. Tuy nhiên, khách hàng có thể điều chỉnh lại các yêu cầu của mình khi chưa đến hạn nhận kết quả.
- Đối với các nhà cung cấp dịch vụ tàu hoả và khách sạn, hệ thống phải quản lý thông tin về cá nhân và thông tin dịch vụ của nhà cung cấp.

5.1.2 Mô hình sở thích người sử dụng

Đối với lớp các bài toán trong thương mại điện tử cũng như nhiều lớp các bài toán khác, vấn đề quan trọng cần giải quyết trước tiên đó là mô hình sở thích của người sử dụng được xác định bởi tập các ràng buộc bao gồm:

- Ràng buộc các thuộc tính của một mặt hàng.
- Ràng buộc giữa các mặt hàng với nhau.

a. Ràng buộc các thuộc tính

Sở thích của người dùng về các đối tượng (các mặt hàng) có thể dựa trên việc đánh giá đối tượng hoặc dựa trên thuộc tính của đối tượng.

Phương pháp dựa trên đánh giá đối tượng

Để đánh giá được sở thích của khách hàng đối với mỗi đối tượng, khách hàng phải chỉ ra mình thích cái nào trong tập các đối tượng đó. Phương pháp này mang tính trực quan và cảm tính.

Một cách tiếp cận khác trong phương pháp đánh giá đối tượng này là sử dụng kiến thức chuyên gia: các chuyên gia sẽ đánh giá đối tượng nào là phù hợp với khách hàng hoặc so sánh từng cặp đối tượng để chọn ra đối tượng phù hợp hơn. Dựa trên các kết quả đánh giá của các chuyên gia, người ta tổng hợp lại để có được kết quả cuối cùng và chọn ra đối tượng phù hợp nhất đối với khách hàng.

Phương pháp dựa trên các thuộc tính

Theo phương pháp này, mỗi đối tượng được đặc trưng bởi một tập các thuộc tính. Do đó, thay vì phải đánh giá sự phù hợp trên toàn đối tượng, người ta đánh giá sự phù hợp với khách hàng trên từng thuộc tính của đối tượng dựa trên biến ngôn ngữ. Các kết quả trên từng thuộc tính này sẽ được tổng hợp lại để được kết quả cuối cùng của mỗi đối tượng [26].

Trong bài toán hệ dịch vụ du lịch, mô hình sở thích người sử dụng được tiếp cận theo phương pháp này:

- Khách hàng không thể biết chi tiết toàn bộ các khách sạn (hoặc tàu hoả) trong khu vực mình định đến để đánh giá cái nào là phù hợp với mình hơn.
- Khách hàng chỉ biết các ràng buộc nào đó của mặt hàng mà mình có thể chấp nhận được. Ví dụ, các ràng buộc về khách sạn mà khách hàng quan tâm có thể là các thuộc tính như: giá phòng, điều kiện phục vụ, khoảng cách tới trung tâm thành phố...

- Trong điều kiện nhất định, khách hàng sẽ không thể quyết định được việc phải chọn một trong hai phương án, ví dụ như đi tàu chất lượng cao và phải ở khách sạn hạng thấp hoặc ngược lại, chịu khó đi tàu chất lượng trung bình để dành tiền ở khách sạn chất lượng tốt hơn...

Các khách sạn được đặc trưng bởi các thuộc tính:

- Giá phòng
- Chất lượng phục vụ (số sao của khách sạn)
- Khoảng cách tới trung tâm
- Có kết nối dịch vụ internet hay không

Các chuyến tàu được đặc trưng bởi các thuộc tính:

- Giá vé.
- Chất lượng chỗ ngồi/năm.
- Tốc độ của tàu (thời gian trên tàu).

Các khách hàng khác nhau thường có những yêu cầu khác nhau về các thuộc tính này. Chẳng hạn, có người quan tâm đến giá cả, có người lại quan tâm đến chất lượng của dịch vụ. Do đó, để đặc tả được sự khác nhau giữa các thuộc tính, cần đến tham số, được gọi là *độ quan trọng* của mỗi thuộc tính. Khách hàng càng quan tâm đến thuộc tính nào thì giá trị của *độ quan trọng* cho thuộc tính đó càng cao.

Bảng 5.1 mô tả một cách biểu diễn độ quan trọng của các thuộc tính đối với dịch vụ khách sạn.

	Giá phòng	Số sao	Khoảng cách đến trung tâm	Internet
Tuyệt đối quan trọng	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Rất quan trọng	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Khá quan trọng	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Quan trọng vừa phải	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Không quan trọng lắm	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ít quan trọng	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Hoàn toàn không quan trọng	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Bảng 5.1: Ví dụ về biểu diễn độ ưu tiên

Một vấn đề nữa là tính mềm dẻo trong yêu cầu của khách hàng: họ không thể khẳng định được chính xác một giá trị cụ thể của mỗi ràng buộc mà chỉ có thể xác định được một khoảng nào đó là có thể chấp nhận được. Chẳng hạn với thuộc tính giá phòng của khách sạn, khách hàng có thể chấp nhận mức giá xê dịch trong khoảng từ 250 đến 300 ngàn đồng/phòng/ngày.

Để đặc tả được các yêu cầu mềm dẻo như vậy, mỗi thuộc tính của mặt hàng cần đến hai tham số: *min* là giá trị thấp nhất, *max* là giá trị cao nhất có thể chấp nhận được.

Như vậy, mỗi thuộc tính của mỗi mặt hàng sẽ được đặc trưng bởi các tham số:

- *Min*: giá trị thấp nhất có thể chấp nhận.
- *Max*: giá trị cao nhất có thể chấp nhận.
- *Priory*: độ quan trọng của thuộc tính đó đối với khách hàng.

Phương pháp mô hình sở thích của người dùng này dựa trên các kỹ thuật tính toán mềm và lập luận theo logic mờ trong việc ra quyết định của agent ([19], [25],[26]).

b. Ràng buộc giữa các mặt hàng

Trong bài toán dịch vụ du lịch, có hai ràng buộc cần giải quyết:

Ràng buộc về chi phí

Với tổng chi phí có hạn, việc đi tàu chất lượng cao (giá vé cao) sẽ dẫn đến việc phải ở trong các khách sạn rẻ tiền, chất lượng thấp.

Ràng buộc về thời gian

Thời gian thuê khách sạn phải phụ thuộc vào thời gian của chuyến tàu đi và về. Chẳng hạn nếu khách hàng muốn đi vào ngày mồng 5 nhưng ngày đó hết vé tàu thì phải dịch sang ngày mồng 6 hoặc đi từ ngày mồng 4, điều này dẫn đến thay đổi thời gian và chi phí trong việc thuê khách sạn.

Các ràng buộc này sẽ không được kiểm soát trong quá trình thương lượng của mỗi mặt hàng bởi vì việc thương lượng mỗi mặt hàng của mỗi thành phần là độc lập với nhau trong bản thân UserAgent (sẽ được trình bày trong chương 7). Để giải quyết các ràng buộc này, có hai cách giải quyết như sau:

- **Tiền ước lượng:** chỉ áp dụng được đối với các ràng buộc chỉ phụ thuộc vào khách hàng, chẳng hạn ràng buộc chi phí. Nghĩa là tiền ước lượng chi phí cho mỗi dịch vụ có được từ giá trị ràng buộc của khách hàng. Ràng buộc chi phí của bài toán dịch vụ du lịch được giải quyết theo cách này. Theo đó, chi phí của dịch vụ khách sạn và tàu hỏa sẽ được ước lượng trước sau khi tham khảo giá dịch vụ từ các agent cung cấp các dịch vụ đó. Quá trình thương lượng chính thức diễn ra sau khi việc tiền ước lượng kết thúc.
- **Hậu ước lượng:** sau khi thương lượng xong mỗi mặt hàng, nếu khi tổng hợp có ít nhất một ràng buộc không thỏa mãn thì sẽ tiến hành thương lượng lại. Việc thương lượng lại có thể được thực hiện theo một trong hai giải pháp sau:
 - Chỉ thương lượng lại một mặt hàng khi chấp nhận các mặt hàng kia.

- Thương lượng lại tất cả các mặt hàng. Giải pháp này thực tế tăng độ phức tạp tính toán lên rất lớn.

Trong cài đặt hệ **TraNeS**, giải pháp thứ nhất cho việc giải quyết ràng buộc về thời gian đã được áp dụng. Theo đó, nếu sau khi thương lượng, ràng buộc này không thỏa mãn, hệ thống sẽ chấp nhận dịch vụ tàu hoả và thương lượng lại dịch vụ khách sạn.

5.2 Phân tích hệ thống

Như đã trình bày trong Chương 4, phát triển hệ đa agent bao gồm hai pha: *pha phân tích* và *pha thiết kế*. Chi tiết các bước trong mỗi pha được minh họa trong Hình 4.1. Pha phân tích bao gồm các bước:

1. Xác định các dịch
2. Xác định Use Case
3. Xây dựng Ontology
4. Xây dựng sơ đồ role

Chi tiết các bước trong pha phân tích hệ dịch vụ du lịch **TraNeS** sẽ được giới thiệu trong các phần tiếp theo.

5.2.1 Xác định dịch của hệ thống

Có hai bước nhỏ trong việc xây dựng cây dịch của hệ thống là tập hợp các dịch và tổ chức cây dịch. Trong ngữ cảnh của hệ dịch vụ TraNeS, hai bước nhỏ này được thực hiện như sau:

Tập hợp dịch

Trước hết cần xác định các yêu cầu chức năng để từ đó chỉ ra các dịch ban đầu của hệ thống. Với dịch vụ du lịch TraNeS, có thể xác định ngay được hai dịch ban đầu là *thương lượng* (negotiation) và *đặt chỗ* (reservation).

Tiếp theo cần thực hiện việc phân rã các dịch. Với hai dịch ban đầu đã được phát hiện là *đặt chỗ* và *thương lượng*, thì dịch *thương lượng* sẽ được phân rã thành hai dịch con là *tìm đối tác* và *thông báo kết quả*. Việc *đặt chỗ* sẽ hoàn thành nếu các việc *đặt chỗ khách sạn* và *đặt vé tàu* được hoàn thành. Do vậy, dịch *đặt chỗ* được phân rã thành hai dịch con là *đặt vé tàu* và *đặt chỗ khách sạn*.

Với dịch *tìm đối tác*, ta sẽ không thể phân rã thêm vì khi trả lời câu hỏi “muốn hoàn thành việc tìm đối tác thì nên làm cái gì?” sẽ đưa ra cách thức *tìm đối tác* mà không phải là *một cái gì đó* để tìm đối tác. Do vậy, dịch này không cần phân rã thêm.

Như vậy, các dịch của hệ thống được xác định như sau:

- Đích *thương lượng*, muốn thương lượng thành công thì trước hết phải có đối tác phù hợp, tức là phải tìm kiếm đối tác. Nhận xét rằng việc *tìm kiếm đối tác* cũng là một task mức hệ thống, cho nên nó sẽ đóng vai trò một đếch, gọi là *Search seller*. Hơn nữa, sau khi thương lượng thì khách hàng phải biết được kết quả thương lượng, do đó, việc *thông báo kết quả* cũng là một đếch, gọi là *Report result*.
- Đích *đặt chỗ* sẽ hoàn thành nếu các việc *đặt chỗ khách sạn* và *đặt vé tàu* được hoàn thành. Do vậy, đếch *đặt chỗ* được phân rã thành hai đếch con là *đặt vé tàu* (*Train ticket booking*) và *đặt chỗ khách sạn* (*Hotel reservation*).

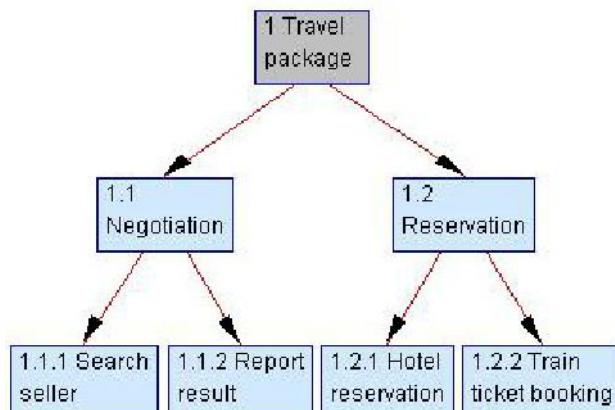
Tóm lại, hệ thống có các đếch là *thương lượng* (*Negotiation*), *đặt chỗ* (*Reservation*), *tìm đối tác* (*Search seller*), *thông báo kết quả* (*Report result*), *đặt chỗ khách sạn* (*Hotel reservation*) và *đặt vé tàu* (*Train ticket booking*).

Tổ chức cây đếch

Bước con này có task tổ chức các đếch đã xác định trong bước con trước vào một sơ đồ phân cấp đếch. Trong hệ dịch vụ du lịch, hai đếch ở cùng mức cao nhất là *thương lượng* và *đặt chỗ*. Nhận xét rằng mặc dù hai đếch này không trực tiếp hỗ trợ nhau nhưng cùng hỗ trợ cho một task chung, chính là mục đich của bài toán: *dịch vụ du lịch trọn gói*. Cho nên đếch tổng thể của hệ thống được xác định là *dịch vụ du lịch trọn gói* (*travel package*). Trong hệ dịch vụ du lịch các đếch được đưa vào cây phân cấp đếch như sau:

- Các đếch *thương lượng* và *đặt chỗ* là nút con trực tiếp của nút gốc.
- Các đếch *tìm đối tác* và *thông báo kết quả* là nút con của đếch *thương lượng*.
- Các đếch *đặt vé tàu* và *đặt chỗ khách sạn* là nút con của đếch *đặt chỗ*.

Cây đếch của hệ thống được mô tả như Hình 5.2.



Hình 5.2: Cây Đếch của hệ thống

Trong cây đích này, *travel package* là một thuộc loại *đích tổng hợp* và *bị phân hoạch*.

5.2.2 Xây dựng các use case

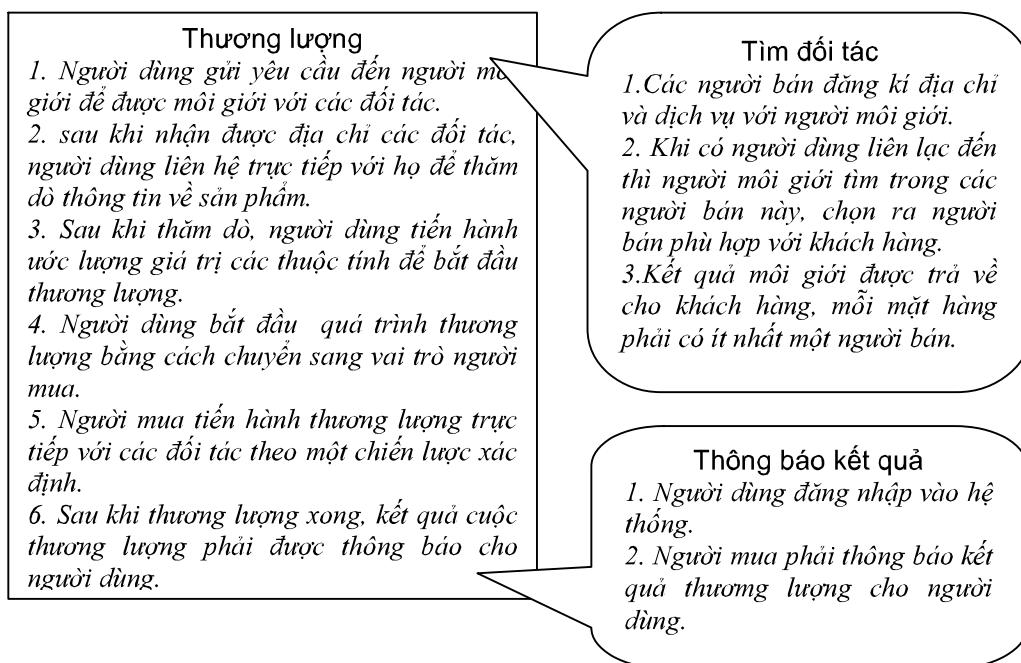
Bước này cũng bao gồm hai bước con là tạo các use case (creating use case) và xây dựng biểu đồ tương tác tuần tự (creating sequence diagram).

Tạo các use case

Với nhánh có đích *thương lượng* trong cây đích của bài toán dịch vụ du lịch, quá trình xây dựng các use case được diễn ra như sau:

- Trích dẫn use case tương ứng với đích *tìm đối tác*
- Trích dẫn use case tương ứng với đích *thông báo kết quả*
- Trích dẫn use case tương ứng với đích *thương lượng* khi xem các use case tương ứng với các đích con của nó tương đương với một hành động đơn.

Quá trình trích dẫn use case của nhánh này sẽ dừng lại ở đây bởi vì nút cha của đích *thương lượng* nút *travel package* bị phân hoạch như đã được xác định trong bước 1. Quá trình này và chi tiết các use case trong nhánh này được mô tả trong Hình 5.3.



Hình 5.3: Quan hệ giữa các use case của đích cha và đích con

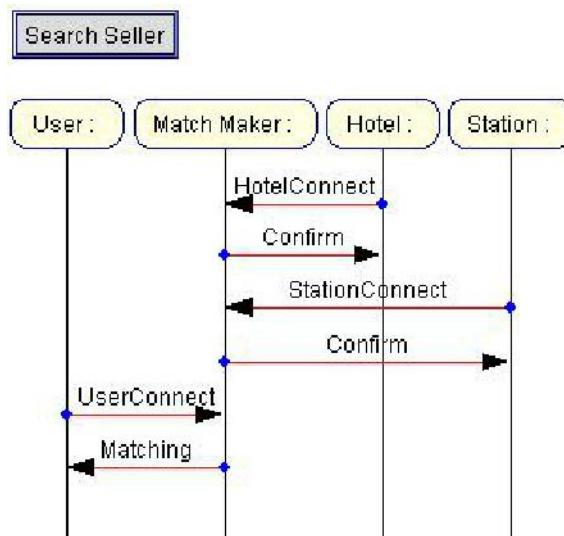
Xây dựng biểu đồ tuần tự

Trong use case *tìm đối tác*, có các khái niệm có thể tạo thành role là *người mua*, *người bán* (bao gồm khách sạn và nhà ga) và *người môi giới*. Do đó, có thể cần đến các role là: Buyer, Hotel, Station và Matchmaker.

Với use case *thông báo kết quả* chỉ cần hai role là *người bán* (Buyer) và *người sử dụng* (User).

Đối với use case *thương lượng*, cần tất cả các role có mặt trong các use case con (*tìm đối tác* và *thông báo kết quả*) và trong bản thân nó. Do đó, cần các role: User, Buyer, Hotel, Station và Matchmaker.

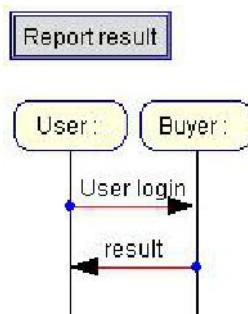
- Tìm kiếm đối tác**
- Một khách sạn muốn đăng ký dịch vụ với người môi giới trong hệ thống, nó gửi các thông tin dịch vụ đến người môi giới.
 - Người môi giới lưu thông tin này lại. Đồng thời xác nhận lại với khách sạn là đã nhận được thông tin đăng ký.
 - Một nhà ga muốn tham gia cũng đăng ký thông tin dịch vụ với người môi giới.
 - Người môi giới cũng lưu lại thông tin dịch vụ này và xác nhận với nhà ga là đã nhận được đăng ký.
 - Khi người dùng muốn tìm người bán phù hợp với mình, nó gửi một yêu cầu đến người môi giới.
 - Khi nhận được yêu cầu từ người dùng, người môi giới tìm ra người bán theo yêu cầu và gửi thông tin về cho người dùng.



Hình 5.4: Use case *tìm đối tác* và sơ đồ tương ứng

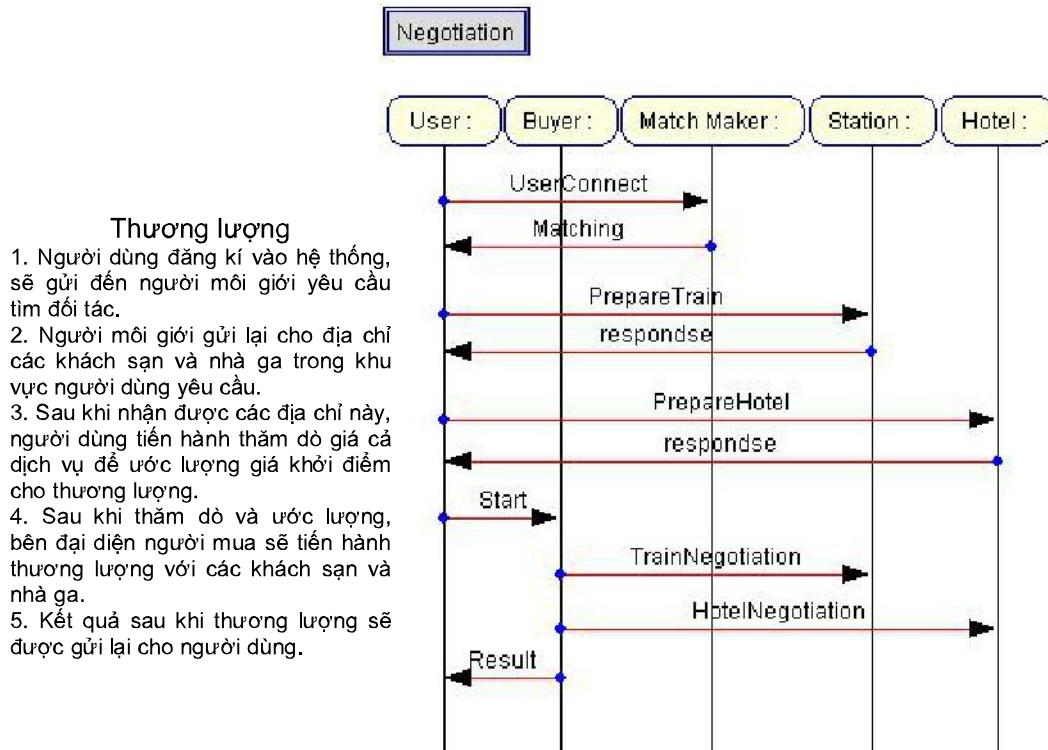
Công việc tiếp theo là sắp xếp các sự kiện diễn ra trong use case theo đúng tuần tự vào biểu đồ. Trong use case *tìm đối tác*, sự kiện người mua liên hệ với người môi giới để tìm đối tác sẽ được biểu diễn bằng một mũi tên đi từ role User đến role Matchmaker và có nhãn là “yêu cầu đối tác”.

- Thông báo kết quả**
- Để xem kết quả, người sử dụng đăng nhập vào hệ thống với tài khoản của mình.
 - Khi có kết quả thương lượng, hệ thống sẽ thông báo cho người dùng đó.



Hình 5.5: Use case *Thông báo kết quả*

Các biểu đồ dây tương ứng với các use case trong Hình 5.3 được minh họa trong các Hình 5.4, 5.5 và 5.6.



Hình 5.6: Sơ đồ tuần tự cho Use case *Thương lượng*

Sau bước này, ta đã chỉ ra được một tập các role ban đầu cho hệ thống.

5.2.3 Xây dựng ontology

Trong MaSE, việc xây dựng ontology bao gồm bốn bước chính [8]: Xác định mục đích và phạm vi của ontology, thu thập dữ liệu, xây dựng ontology khởi đầu và bước cuối cùng là hoàn thiện ontology.

Xác định mục đích và phạm vi của ontology

Để xác định mục đích và phạm vi của ontology trong hệ TraNeS, người phân tích sử dụng kỹ thuật khoanh vùng và thu hẹp dần các miền tri thức để xác định phạm vi của ontology. Một miền tri thức ban đầu được xác định là tri thức về *thương mại điện tử*. Tuy nhiên, bài toán chỉ xem xét bài toán thương mại điện tử thực hiện theo mô hình thương lượng song phương tự động. Do đó, phạm vi được thu hẹp lại một mức, chỉ gồm các tri thức về *hoạt động thương lượng song phương tự động trong thương mại điện tử*. Trong thương lượng song phương, các miền tri thức cần biểu diễn bao gồm:

- Tri thức về người bán: các khách sạn và các nhà ga.
- Tri thức về người mua: khách du lịch

- Tri thức về mặt hàng được trao đổi: sản phẩm du lịch.
- Tri thức về người môi giới.

Thu thập dữ liệu

Trong hệ dịch vụ du lịch, miền tri thức của người bán bao gồm:

- Tên và địa chỉ của người bán.
- Tên mặt hàng và các thuộc tính của mỗi mặt hàng của người bán.

Miền tri thức của người mua bao gồm:

- Tên và địa chỉ của người mua.
- Yêu cầu về sở thích của người mua đối với mặt hàng.
- Các thông tin về cá nhân để liên hệ.

Miền tri thức của người môi giới:

- Tên, địa chỉ và khả năng cung cấp dịch vụ của các agent bán.
- Tên, địa chỉ, và yêu cầu của các agent mua.

Xây dựng ontology khởi đầu

Với hệ TraNeS, quá trình phân loại đối với mỗi miền tri thức con được tiến hành như sau.

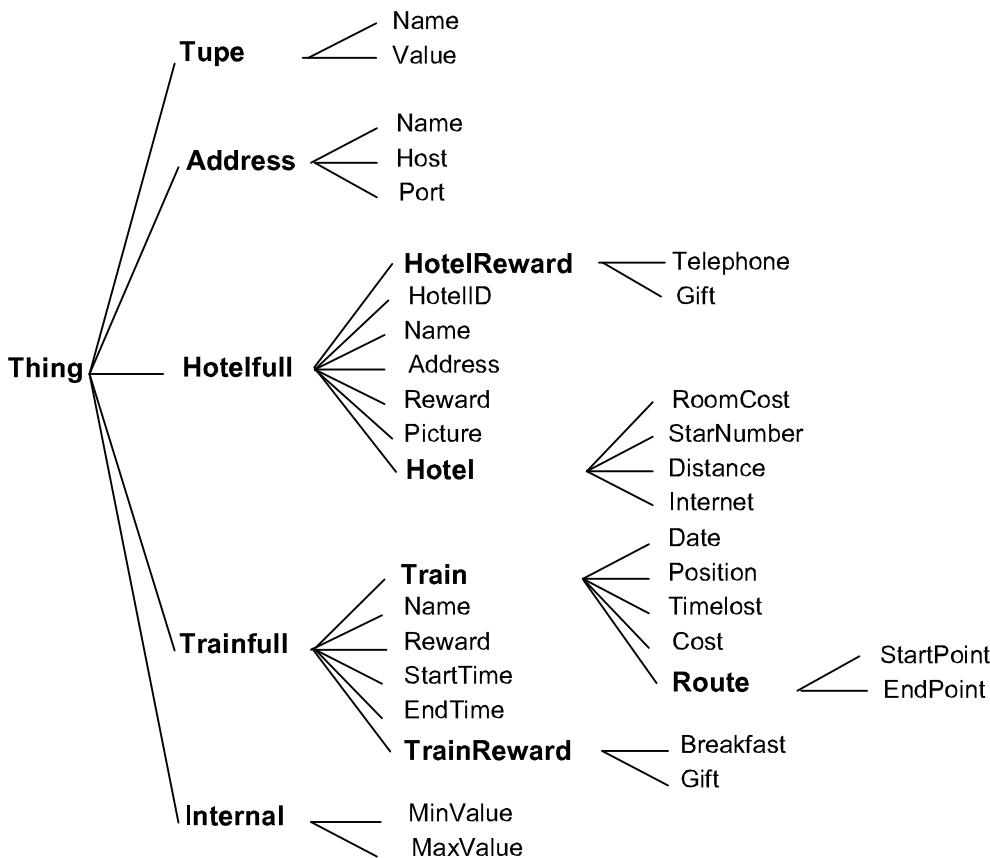
- Đối với miền tri thức của người bán và người môi giới, có các khái niệm là {địa chỉ (*Address*), tên (*name*), số cổng (*port*)}.
- Đối với miền tri thức của người mua, có hai khái niệm là {giá nhỏ nhất (*minValue*), giá lớn nhất (*maxValue*)}. Hai khái niệm này có cùng mức độ chi tiết là thấp nhất, có thể xếp chúng làm thuộc tính cho một lớp mới tạo ra, gọi là một khoảng (*Internal*).
- Đối với miền tri thức của mặt hàng, có các khái niệm chính là {khách sạn (*Hotel*), khách sạn đầy đủ (*Hotelfull*), chuyến tàu (*Train*), chuyến tàu đầy đủ (*Trainfull*), bộ thuộc tính (*Tupe*), khuyến mại khách sạn (*HotelReward*), khuyến mại tàu (*TrainReward*), tuyến đi (*Route*)}. Các khái niệm này sẽ trở thành lớp và các khái niệm được khai phá từ các khái niệm này đều trở thành thuộc tính tương ứng của chúng. Khái niệm *Hotel* có các thuộc tính là {số sao (*starNumber*), giá phòng (*roomCost*), khoảng cách (*distance*), *internet*}. Lớp *Hotelfull* được kế thừa từ lớp *Hotel* và có thêm các thuộc tính là {tên khách sạn (*name*), địa chỉ khách sạn (*address*)}. Lớp *Route* có hai thuộc tính là {nơi đi (*startPoint*), nơi đến (*endPoint*)}. Lớp *Train* kế thừa từ lớp *Route*, có thêm các thuộc tính để thương lượng là {giá vé (*cost*), loại vé (*position*), loại tàu (*timelost*)}. Lớp *Trainfull* kế

thì từ lớp *Train* và có thêm các thuộc tính là {tên tàu (*name*), giờ đi (*startTime*), giờ đến (*endTime*)}.

Sơ đồ lớp ontology của hệ thống được minh họa trong hình 5.7.

Hoàn thiện và kiểm định ontology

Kết quả của bước 3 này là một sơ đồ phân cấp khái niệm của ontology hoàn chỉnh, có thể đáp ứng yêu cầu hoạt động của hệ thống. Sơ đồ ontology của hệ thống như Hình 5.7.



Hình 5.7: Sơ đồ ontology của hệ thống

5.2.4 Hoàn thiện các role

Bài toán dịch vụ du lịch đã có các role tương ứng với các dịch như sau:

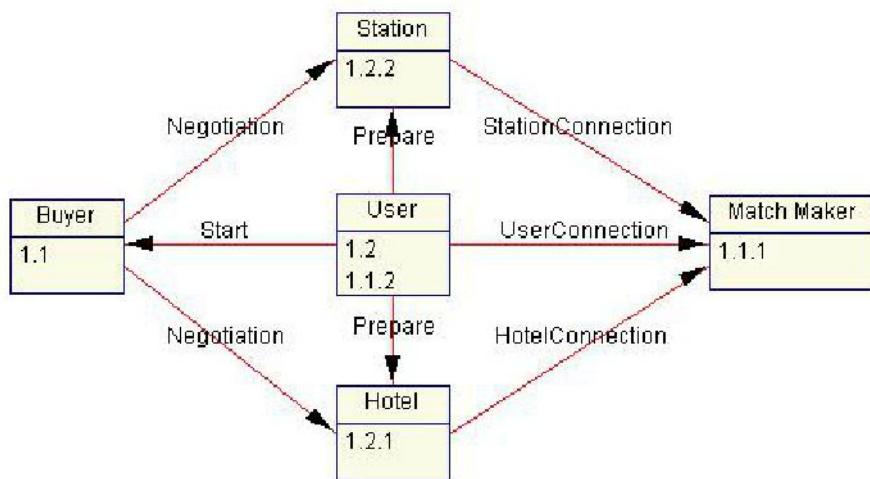
- Role Matchmaker thực hiện đích *tìm đối tác*
- Role Buyer thực hiện đích *thương lượng*
- Role User thực hiện đích *thông báo kết quả*.
- Role Hotel thực hiện đích *đặt chỗ khách sạn*
- Role Station thực hiện đích *đặt vé tàu*.

- Với đích *đặt chỗ* (1.2), đích này phần lớn task đã được thực hiện bởi hai đích con là *đặt chỗ khách sạn* và *đặt vé tàu*, phần việc còn lại là tích hợp thông tin theo yêu cầu người sử dụng nên được thực hiện bởi role User.

Việc giao tiếp với khách hàng được giao cho role User bởi vì bản thân role này đã thực hiện các đích *thông báo kết quả*, tích hợp thông tin trong đích *đặt chỗ*, là các đích giao tiếp với người dùng. Do vậy, không cần thiết phải tạo ra một role hoàn toàn mới để đảm nhiệm việc này.

Trong bài toán này, ngoài tài nguyên con người đã được giải quyết như trên, các nguồn tài nguyên bên ngoài còn lại bao gồm các khách sạn và các nhà ga quản lý các chuyến tàu. Việc giao tiếp với các đối tượng này được giao cho các role tương ứng là Hotel và Station, bởi các role này đã thực hiện các đích *đặt chỗ khách sạn* và *đặt vé tàu*, các các đích mang tính chất giao tiếp với các đối tượng đã nêu. Do vậy, cũng không cần thiết phải tạo ra các đích mới để đảm nhiệm các công việc này.

Với các sự kiện của use case *tìm đối tác*, xuất hiện giữa hai role User và Matchmaker sẽ trở thành giao thức *UserConnect* giữa hai role tương ứng. Hình 5.8 là sơ đồ role ban đầu của hệ thống.



Hình 5.8: Sơ đồ Role chưa gán Nhiệm vụ

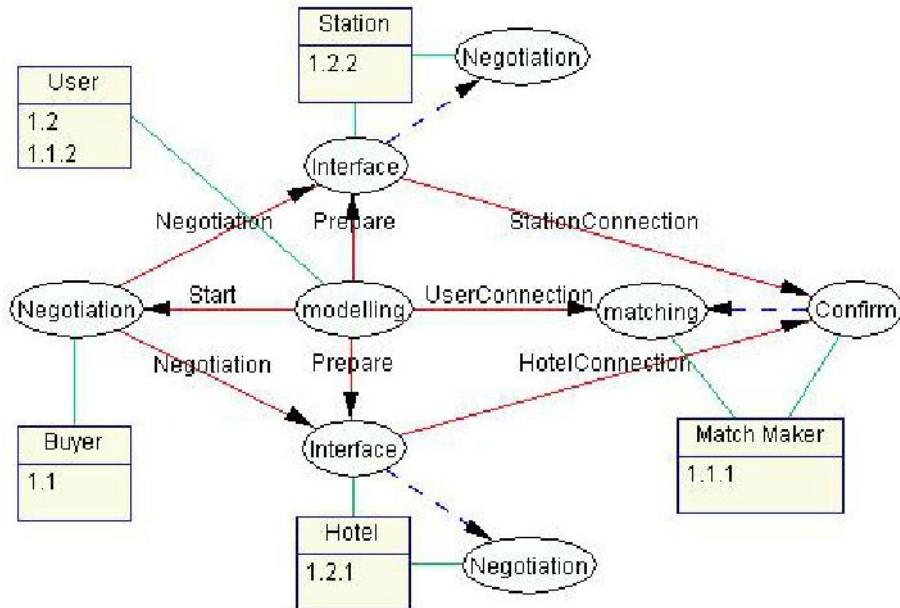
Trong bài toán này, có ba task trùng tên là thương lượng (*negotiation*) của ba role tương ứng là Buyer, Hotel và Station. Mặc dù trùng tên và có chức năng tương tự nhau, nhưng mỗi role lại có task thương lượng theo thuật toán khác nhau (người bán và người mua) và nhằm đem lại lợi ích cho các đối tượng có quyền lợi khác nhau (chi tiết các task này được mô tả trong phần tiếp theo). Các task của các role còn lại được tổ chức như Hình 5.9.

Role User chỉ có một task là *modelling*, thực hiện mô hình hoá yêu cầu của người dùng và chuẩn bị cho các hoạt động thương lượng. Mặc dù role User phải thực hiện hai đích là *thông báo kết quả* và *đặt chỗ*, nhưng các đích này có công việc gần giống nhau và đều liên quan đến việc mô hình hoá sở thích người dùng; do vậy chỉ cần một task *modelling* là đủ.

Role Buyer cũng chỉ có một task là *negotiation*, thay mặt người dùng thương lượng với các đối tác. Tuy nhiên, đích *thương lượng* thuộc vào loại đích đặc biệt, cần nhiều bên tham gia nhưng không thể chia thành các đích nhỏ hơn được nữa. Do đó, nó được chia ra làm ba task cùng tên *negotiation* nằm ở ba role tương ứng là Buyer, Hotel và Station.

Role Station và Hotel có hai task: *interface* có task giao tiếp (truyền thông) với các khách hàng và người môi giới, task *negotiation* thay mặt người bán thương lượng với khách hàng.

Role Matchmaker mặc dù chỉ cần thực hiện một đích là *tìm đối tác*, nhưng để hoàn thành được task này, nó cần phải thu thập được địa chỉ và khả năng của các agent khác trong hệ thống để khi cần thiết, môi giới cho các người mua có yêu cầu. Do đó, nó cần đến hai nhiệm vụ: task *confirm* xác nhận các đăng ký dịch vụ từ các nhà cung cấp dịch vụ, nghĩa là thu thập tri thức về các người bán tham gia vào hệ thống. Task *matching* môi giới các khách hàng với các nhà cung cấp dịch vụ phù hợp với họ, việc môi giới này được dựa trên các tri thức mà nó đã thu thập được trong task *confirm*.



Hình 5.9: Sơ đồ Role đã gán Nhiệm vụ

Sau khi các task được gán vào các role xác định, các giao thức giữa các role trong sơ đồ role ban đầu sẽ được xác định cụ thể bởi các task liên quan. Chẳng hạn,

trong sơ đồ role ban đầu, role Matchmaker có ba giao thức kết nối tới, đến bước này chúng được cụ thể hoá như sau:

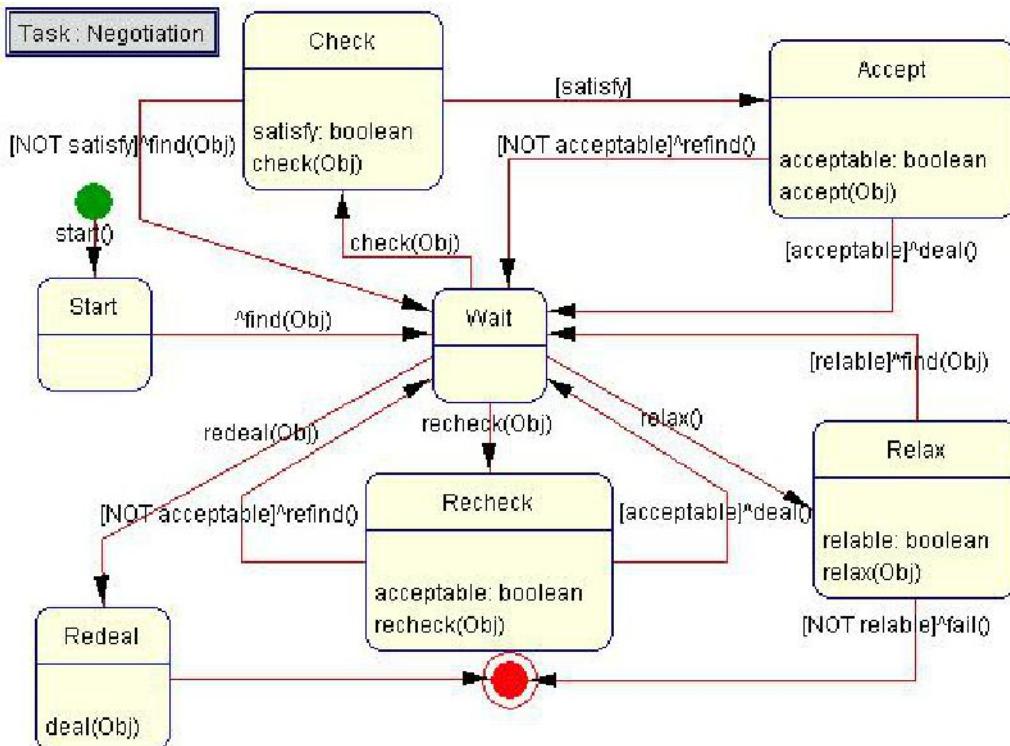
- Các giao thức *StationConnection* và *HotelConnection* liên quan đến task *confirm* do task này chịu trách nhiệm thu thập tri thức về các người bán.
- Giao thức còn lại là *UserConnection* liên quan đến task *matching* do task này trực tiếp thực hiện môi giới người mua với người bán phù hợp.

Phần việc còn lại của bước này là xây dựng cách thức hoạt động bên trong của mỗi nhiệm vụ.

Sơ đồ bên trong task

Hình 5.10 là sơ đồ của task *thương lượng* thuộc role Buyer. Vì role này chỉ có một task nên tất cả các chuyển tiếp trong sơ đồ task này đều tham gia vào tương tác ngoài. Theo đó, nó bước vào trạng thái *Start* khi nhận được thông điệp “*start*” từ task *modelling* của role User, do đó task này thuộc loại task bị động. Các tương tác còn lại đều được tiến hành với task *thương lượng* của role Station hoặc role Hotel.

- Tại trạng thái *Start*, nó tìm ra thuộc tính về hàng hoá có độ ưu tiên cao nhất, gửi đi theo thông điệp “*find*” và chuyển vào trạng thái *Wait* để chờ các đáp ứng từ task *negotiation* của đối tác.
- Tại trạng thái *Wait*, nếu nhận được thông điệp “*check*” thì nó chuyển vào trạng thái *Check*, nếu nhận được thông điệp “*recheck*” thì nó chuyển vào trạng thái *Recheck*, nếu nhận được thông điệp “*relax*” thì nó chuyển vào trạng thái *Relax*, nếu nhận được thông điệp “*redeal*” thì nó chuyển vào trạng thái *Redeal*.
- Tại trạng thái *Recheck*, nó kiểm tra lại xem mặt hàng cũ với thêm các hình thức khuyến mãi có thể chấp nhận được hay không.
- Tại trạng thái *Relax*, nếu còn nhượng bộ được, nó chọn ra thuộc tính để nhượng bộ sao cho giá trị lợi ích bị giảm đối với nó là nhỏ nhất, gửi đi theo thông điệp “*find*” và chuyển vào trạng thái *Wait*, nếu không nhượng bộ được nữa thì nó gửi thông điệp “*fail*” báo thất bại và chuyển vào trạng thái kết thúc.
- Tại trạng thái *Redeal*, nó lưu lại thông tin mặt hàng vừa thương lượng được.



Hình 5.10: Sơ đồ hoạt động của task Negotiation thuộc role Buyer

Như vậy, kết quả của pha phân tích là một tập các role và các task tương ứng nhằm hoàn thành mục đích của hệ thống. Hoạt động bên trong và các tương tác bên ngoài giữa các task là nhân tố quan trọng cho pha thiết kế sau này.

5.3 Kết luận

Nội dung chương này đã trình bày chi tiết các bước trong pha phân tích hệ đa agent trong hệ dịch vụ du lịch **TraNeS**. Pha phân tích bao gồm bốn bước: Xác định đích, xác định use case, xây dựng Ontology và xây dựng sơ đồ role. Kết quả của pha này sẽ tạo điều kiện cho việc xây dựng các bước trong pha thiết kế và sẽ được trình bày một cách chi tiết trong chương đến.

CHƯƠNG 6

THIẾT KẾ HỆ DỊCH VỤ

- Một số vấn đề thiết kế hệ đa agent
- Thiết kế hệ dịch vụ du lịch
- Kết luận

Nhiệm vụ của pha thiết kế là chuyển các role được xác định trong pha phân tích thành tập các agent và xác định số lượng, vị trí các agent trong hệ thống. Như đã trình bày trong Chương 4, pha này bao gồm các bước:

1. Xác định các lớp agent
2. Xây dựng các phiên hội thoại
3. Hoàn thiện các agent
4. Triển khai hệ thống

Chương này nhằm tập trung trình bày chi tiết các bước này trong ngữ cảnh hệ dịch vụ du lịch **TraNeS**.

6.1 Một số vấn đề về thiết kế hệ đa agent

Trong quá trình thiết kế một hệ đa agent, người thiết kế phải trả lời các câu hỏi sau [39]:

- Cần có những kiểu agent nào? Đây chính là câu hỏi giúp xác định các lớp agent trong hệ thống.
- Cần có các agent nào? Mỗi agent nhằm thực hiện một chức năng xác định và thuộc một lớp agent nào đó.
- Cấu trúc chung của hệ thống như thế nào?
- Các hành vi của hệ thống trong những trường hợp cụ thể như thế nào?
- Các agent hướng tới các đích của mình như thế nào?
- Các agent phản ứng lại các sự kiện như thế nào?
- Các agent cần có những thông tin gì để giải quyết vấn đề đặt ra cho mình?

Sau khi đã trả lời đầy đủ các câu hỏi trên thì vấn đề còn lại của pha thiết kế là mô hình hoá hệ thống dựa trên một phương pháp luận và một công cụ nào đó.

Các phương pháp luận khác nhau mô hình hoá các lớp agent, các agent và các thành phần của agent theo các cách khác nhau. Chương này tập trung trình bày các bước trong pha thiết kế hệ đa agent như đã trình bày trong Chương 4. Nhiệm vụ của pha thiết kế là chuyển toàn bộ các role và các nhiệm vụ của chúng đã được xác định trong pha phân tích vào tập các lớp agent phù hợp. Đồng thời thiết kế số lượng và vị trí của mỗi loại agent trong hệ thống. Pha thiết kế bao gồm bốn bước:

1. Xác định các lớp agent
2. Xây dựng các phiên hội thoại
3. Hoàn thiện các agent
4. Triển khai hệ thống

Nội dung các phần tiếp theo sẽ trình bày chi tiết các bước này.

6.2 Thiết kế hệ đa agent

6.2.1 Xây dựng các lớp agent

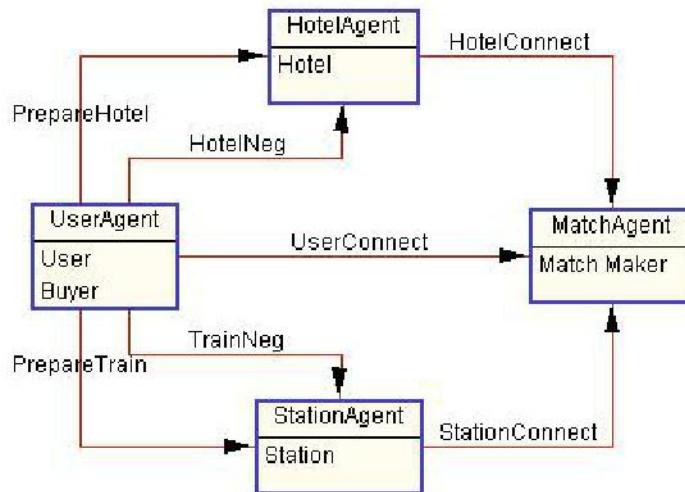
Để đảm bảo các đích hệ thống đều được thực hiện trong pha thiết kế, mỗi role phải được đảm nhiệm bởi ít nhất một lớp agent, đôi khi cũng có các role được đảm nhiệm bởi nhiều lớp agent. Chẳng hạn, lớp UserAgent trong bài toán du lịch đảm nhiệm hai role là User và Buyer. Lý do là hai role này cùng liên quan đến thỏa mãn yêu cầu của người dùng và tương tác giữa chúng ngoài mục đích trao đổi thông tin để hoàn thành nhiệm vụ, còn có mục đích bảo mật thông tin cá nhân cho khách hàng. Do vậy, hai role này được đặt vào cùng một lớp agent.

Theo cách này, các lớp agent để thực hiện các role tương ứng được xác định như sau:

- Lớp UserAgent thực hiện vai trò các role User và Buyer
- Lớp HotelAgent thực hiện vai trò role Hotel
- Lớp StationAgent thực hiện vai trò role Station
- Lớp MatchAgent thực hiện vai trò role Matchmaker.

Nhiệm vụ còn lại của bước này là xác định các phiên hội thoại (conversation) xuất hiện giữa các lớp agent để hoàn thiện sơ đồ lớp agent của hệ thống. Các phiên hội thoại được xác định từ các quan hệ giữa các role mà các lớp agent tương ứng cần thực hiện. Tất cả các tương tác giữa các task đã được xác định trong Bước 4 (Pha phân tích) đều trở thành các phiên hội thoại ngoại trừ tương tác giữa task *modelling* của role User với task *Negotiation* của role Buyer bởi vì hai role này được thực hiện bởi chỉ một lớp agent là UserAgent.

- Tương tác giữa task *Modelling* với task *Interface* của role Hotel trở thành phiên hội thoại *PrepareHotel*
- Tương tác giữa task *Negotiation* của role Buyer với task *Interface* của role Hotel trở thành phiên hội thoại *HotelNeg*.
- Tương tác giữa lớp UserAgent với lớp StationAgent có hai phiên hội thoại là *PrepareTrain* và *TrainNeg*
- Tương tác giữa task *Modelling* của role User với task *Matching* của role Matchmaker trở thành phiên hội thoại *UserConnect*
- Tương tác giữa task *Interface* của role Hotel với task *Confirm* của role Matchmaker trở thành phiên hội thoại *HotelConnect*
- Tương tác giữa task *Interface* của role Station với task *Confirm* của role Matchmaker là phiên hội thoại *TrainConnect* giữa hai lớp StationAgent và MatchAgent. Sơ đồ lớp agent của hệ thống được cho trong Hình 6.1.



Hình 6.1: Sơ đồ lớp agent

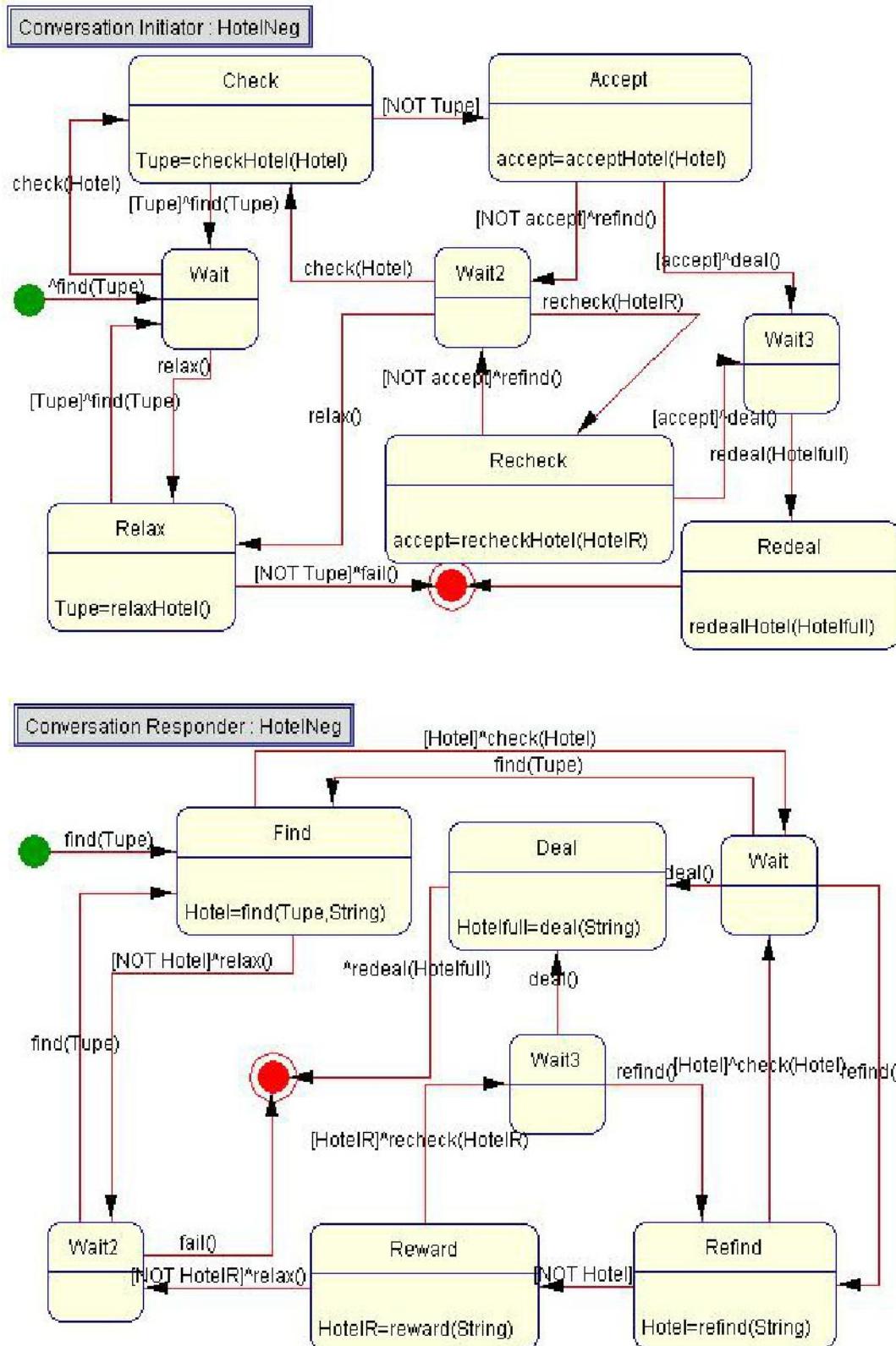
Các phiên hội thoại biểu diễn tương tác giữa các lớp agent diễn ra ở mức ngữ nghĩa thông qua ontology. Dựa vào ontology đã được xây dựng trong Bước 3 (Pha phân tích), các lớp agent có cách biểu diễn tri thức khác nhau sẽ trao đổi và hiểu được lẫn nhau dựa trên tri thức chung trong ontology.

Chẳng hạn, với khái niệm giá tiền cho chuyến đi, UserAgent biểu diễn bằng “*price*”, trong khi HotelAgent biểu diễn giá phòng là “*RoomCost*” và StationAgent biểu diễn giá vé tàu là “*TrainCost*”. Khi đó, để thương lượng với HotelAgent, UserAgent phải dùng khái niệm “*RoomCost*” và khi thương lượng với StationAgent, nó lại dùng khái niệm “*TrainCost*”. Cuối cùng, khi trao đổi kết quả với khách hàng nó lại dùng khái niệm “*price*”, nhưng nó hiểu rằng, “*price*” là tương đương với tổng giá trị của “*RoomCost*” và “*TrainCost*”.

6.2.2 Xây dựng các phiên hội thoại

Nhiệm vụ của bước này là thiết kế chi tiết kiến trúc bên trong của các phiên hội thoại đã xác định được trong bước trước.

Mỗi phiên hội thoại bao gồm hai sơ đồ lớp truyền thông (Communication class diagram), một cho bên khởi xướng và một cho bên đáp ứng phiên hội thoại. Thông thường, mỗi sơ đồ nhiệm vụ sẽ tương ứng với một sơ đồ phiên hội thoại cho bên tham gia tương ứng.



Hình 6.2: Phiên hội thoại HotelNeg cho UserAgent và HotelAgent

Trước hết ta xem xét sơ đồ phiên hội thoại *Negotiation* của *UserAgent*. Tại trạng thái *Check*, nhiệm vụ của agent là phải kiểm tra xem mặt hàng có thỏa mãn yêu cầu hiện tại của *Buyer* hay không:

- Nếu thỏa mãn, trả lại một biến boolean có giá trị true
- Nếu không thì phải trả lại thông tin các thuộc tính không thỏa mãn để gửi yêu cầu lại cho bên đối tác.

Nghĩa là cần một biến là *satisfy* có kiểu boolean và một biến có thể lưu các cặp thuộc tính – giá trị của mặt hàng đang thương lượng. Dựa vào các nhiệm vụ này, trong sơ đồ phiên hội thoại hàm *checkHotel()* được thiết kế như sau: Chỉ cần trả về một biến có kiểu là *Tupe*, kiểu này đã được xác định trong bước xây dựng ontology cho hệ thống:

- Nếu giá trị này là NULL thì có nghĩa là khách sạn thỏa mãn yêu cầu (không có thuộc tính nào bị vi phạm)
- Nếu giá trị này khác NULL thì có nghĩa là có yêu cầu không thỏa mãn và giá trị của biến này đã xác định được tên và giá trị các thuộc tính để yêu cầu lại cho bên đối tác.

Về thiết kế các tham số cho hàm này, tham số đầu vào phải là một giá trị có kiểu *Hotel* đã được định nghĩa trong ontology. Như vậy trong trạng thái *Check* này chỉ cần một hàm *checkHotel(Hotel)*: *Tupe* là đủ. Các hàm trong các trạng thái khác được thiết kế một cách tương tự. Sơ đồ phiên hội thoại cho *UserAgent* được mô tả trong Hình 6.2.

- Tại trạng thái *Check* có hàm *checkHotel(Hotel):Tupe*. Nếu *Tupe* khác NULL thì gửi đi thông điệp *find* và quay lại trạng thái *Wait*. Nếu *Tupe* là NULL thì chuyển sang trạng thái *Accept*.
- Tại trạng thái *Accept* có hàm *acceptHotel(): boolean*. Nếu trả về TRUE thì chấp nhận khách sạn và gửi đi thông điệp *deal*. Nếu trả về FALSE thì không chấp nhận khách sạn, gửi đi thông điệp *refind* và chuyển vào trạng thái *Wait2*.
- Tại trạng thái *Recheck* có hàm *recheckHotel(HotelReward): boolean*. Nếu trả về TRUE thì chấp nhận khách sạn, gửi đi thông điệp *deal* và chuyển vào trạng thái *Wait3*. Nếu trả về FALSE thì không chấp nhận khách sạn, gửi yêu cầu *refind* và chuyển về trạng thái *Wait2*.
- Tại trạng thái *Relax* có hàm *relaxHotel():Tupe*. Nếu *Tupe* khác NULL thì vẫn còn nhượng bộ được, gửi đi yêu cầu *find* và quay lại trạng thái *Wait*. Nếu *Tupe* bằng NULL thì không thể nhượng bộ thêm, gửi thông báo *fail* và kết thúc thát bại.

- Tại trạng thái *Redeal* có hàm *redealHotel(Hotelfull)*. Lưu lại toàn bộ thông tin về khách sạn đã thương lượng được.

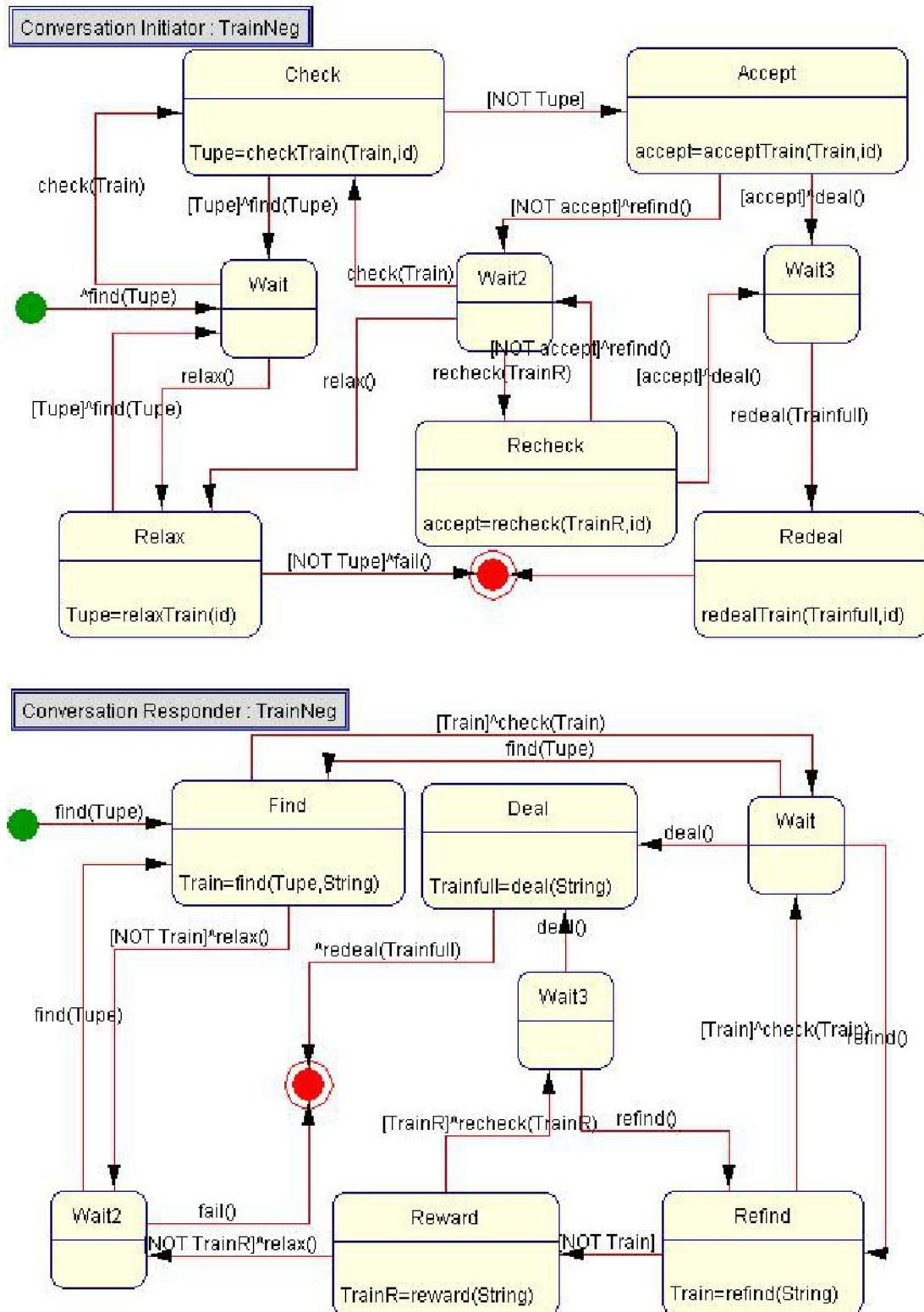
Thiết kế được thực hiện tương tự cho phiên hội thoại *Trainneg* giữa *UserAgent* và *StationAgent*. Tuy nhiên, có một điểm khác biệt giữa thương lượng với *HotelAgent* và thương lượng với *StationAgent*. *UserAgent* có thể thương lượng đồng thời với hai *StationAgent* khác nhau cho các chuyến tàu đi và về; do đó, để xác định được đang thương lượng với *StationAgent* cho chuyến đi hay chuyến về, các hàm trong tất cả các trạng thái của phiên hội thoại này, về phía *UserAgent* phải có thêm một tham số đầu vào là *id* để xác định chuyến tàu đang thương lượng là đi hay về. Sơ đồ cho phiên hội thoại này được mô tả trong Hình 6.3.

Đối với các sơ đồ phiên hội thoại cho *HotelAgent* hoặc *StationAgent* cũng được thiết kế từ các nhiệm vụ *Negotiation* của các role tương ứng. Tuy nhiên, cũng phải bổ sung thêm một tham số đầu vào là *userName*, có kiểu *String* để xác định xem hiện tại nó đang thương lượng với khách hàng nào. Sơ đồ tương ứng cho các phiên hội thoại này được mô tả trong Hình 6.3.

Đối với phiên hội thoại của *HotelAgent*:

- Tại trạng thái *Find* có hàm *find(User, Type): Hotel*. Nếu *Hotel* bằng NULL thì không có khách sạn nào thoả mãn yêu cầu; nó gửi đi thông báo *relax* và chuyển vào trạng thái *Wait2*. Nếu *Hotel* khác NULL thì có khách sạn thoả mãn, nó gửi thông báo *check* và chuyển vào trạng thái *Wait*.
- Tại trạng thái *Refind* có hàm *refind(User): Hotel*. Nếu *Hotel* là NULL thì không có khách sạn thoả mãn; nó chuyển sang trạng thái *Reward*. Nếu *Hotel* khác NULL thì có khách sạn thoả mãn, nó gửi đi thông báo *check* và chuyển sang trạng thái *Wait*.
- Tại trạng thái *Reward* có hàm *reward(User): HotelReward*. Nếu trả về NULL thì không có khuyến mại, nó gửi thông báo yêu cầu *relax* và chuyển sang trạng thái *Wait2*. Nếu trả về khác NULL thì có khuyến mại, nó gửi thông báo *recheck* và chuyển sang trạng thái *Wait3*.
- Tại trạng thái *Deal* có hàm *deal(User): Hotelfull*. Trả về thông tin đầy đủ mà khách hàng vừa đồng ý, nó gửi đi thông điệp *redeal* và kết thúc.

Đối với *TrainAgent* cũng tương tự và được mô tả trong Hình 6.3.



Hình 6.3: Phiên hội thoại TrainNeg cho UserAgent và StationAgent

Tại các chuyển tiếp của sơ đồ phiên hội thoại, người thiết kế cũng phải chi tiết hoá dựa trên các chuyển tiếp trong sơ đồ nhiệm vụ tương ứng. Trong phiên hội thoại *HotelNeg* của *UserAgent*, chuyển tiếp từ trạng thái *Check* sang trạng thái *Wait* được miêu tả bằng cú pháp

[*Tupe not Null*] ^ *find(Tupe)*

Nghĩa là sau khi kiểm tra bằng hàm *checkHotel(Hotel)*, nếu có vi phạm thì giá trị trả về *Tupe* khác NULL, tức là điều kiện [*Tupe not Null*] có giá trị *true*, nên *UserAgent* sẽ gửi đi một thông điệp có performative là “*find*” và nội dung thông điệp có kiểu là *Tupe*. Sơ đồ phiên hội thoại hoàn chỉnh cho mỗi bên được mô tả như các Hình 6.2 và 6.3.

Sau khi các thông tin từ sơ đồ *concurrent task* được ánh xạ vào như một phần của phiên hội thoại, người thiết kế phải kiểm tra lại để đảm bảo các điều kiện khả thi cho phiên hội thoại: mỗi message gửi đi phải có ít nhất một bên nhận và xử lí, không có vòng lặp vô hạn trong các sơ đồ trạng thái... Việc kiểm tra các điều kiện này có thể tiến hành một cách thủ công hoặc bán tự động với sự trợ giúp của *agentTool*. Chẳng hạn kiểm tra theo phương pháp thủ công điều kiện một với phiên hội thoại *HotelNeg*, xuất hiện chu trình

(*Wait – Check – Accept – Wait2 – Relax - Wait*)

của bên *UserAgent* (bên khởi tạo phiên hội thoại này). Sau mỗi vòng lặp này, yêu cầu về các thuộc tính đều bị giảm đi một mức do *UserAgent* tự nhượng bộ, nên dãy yêu cầu của *UserAgent* là “đơn điệu giảm”. Hơn nữa, sự nhượng bộ sẽ dừng lại khi độ thoả mãn các thuộc tính thấp hơn ngưỡng nhượng bộ (*relaxThreshold*), ngưỡng này được ước lượng khi mô hình hoá sở thích người dùng. Do vậy, dãy yêu cầu của *UserAgent* sẽ đơn điệu giảm và bị chặn dưới nên sẽ dừng, nghĩa là vòng lặp không vô hạn.

6.2.3 Hoàn thiện các agent

Bước này hai bước con: *thiết kế kiến trúc bên trong agent* và *thiết kế các thành phần* trong kiến trúc đó.

Thiết kế kiến trúc

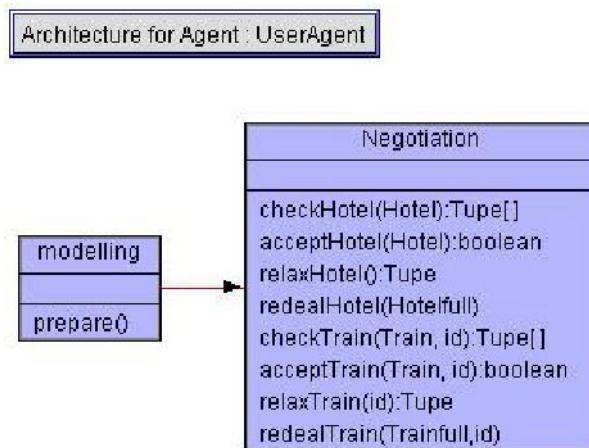
Kiến trúc bên trong của các agent trong bài toán du lịch được thiết kế như sau:

- Lớp *UserAgent* đảm nhiệm hai role *User* và *Buyer* nên các thành phần là *Modelling* và *Negotiation* tương ứng với hai nhiệm vụ của hai role đó.
- Lớp *HotelAgent* và *StationAgent*, mỗi lớp có hai thành phần là *Interface* và *Negotiation* tương ứng với các nhiệm vụ của role mà chúng đảm nhiệm.
- Lớp *MatchAgent* đảm nhiệm một role *Matchmaker* có hai nhiệm vụ nên cũng có hai thành phần tương ứng là *Confirm* và *Matching*.

Sau khi xác định được các thành phần trong kiến trúc của mỗi agent, nhiệm vụ tiếp theo là thiết kế quan hệ giữa các thành phần này cho phù hợp với sơ đồ role của hệ thống đã được phân tích trong Bước 4. Để đảm bảo quan hệ giữa các thành phần là thống nhất với quan hệ giữa các task trong sơ đồ role, các quan hệ này sẽ được thiết kế dựa trên các tương tác giữa các nhiệm vụ trong sơ đồ role. Theo đó, tương tác giữa task *Negotiation* của role *Buyer* với task *Negotiation* của role *Hotel*, sẽ trở thành một quan hệ bên ngoài. Sơ đồ kiến trúc của lớp *UserAgent* được biểu diễn trong Hình 6.4.

Thiết kế các thành phần

Nhiệm vụ của bước con này là thiết kế chi tiết bên trong các thành phần của kiến trúc đã được thiết kế ở bước trên. Việc này được hoàn thành bằng cách gán các thuộc tính và các hàm (thủ tục) chính cho mỗi thành phần.



Hình 6.4: Sơ đồ kiến trúc lớp *UserAgent*

Với kiến trúc lớp *UserAgent*:

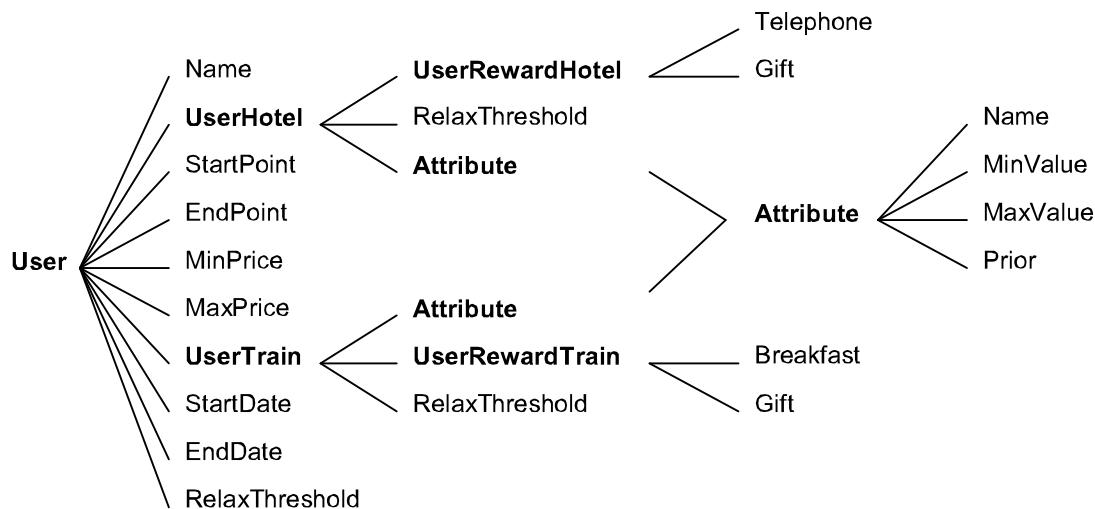
- Thành phần *modelling* chỉ tham gia một phiên hội thoại *UserConnect* đến *MatchAgent*, nên các hàm chỉ có một là *prepare()*.
- Đối với thành phần *Negotiation* của kiến trúc này, nó tham gia đồng thời hai phiên hội thoại phức tạp là *HotelNeg* và *TrainNeg*, nên có một dãy các hàm đã xác định được trong bước 6.

Tất cả các hàm này đều trở thành hàm của các thành phần tương ứng. Sơ đồ kiến trúc chi tiết của *UserAgent* được mô tả trong Hình 6.4.

Trước khi kết thúc bước thiết kế kiến trúc bên trong của agent này, người thiết kế phải hoàn thành việc thiết kế ontology riêng cho mỗi thành phần của mỗi lớp agent nếu điều đó là cần thiết. Các agent loại *HotelAgent*, *StationAgent* và *MatchAgent* không cần ontology riêng, vì tri thức của agent này đều được sử dụng để trao đổi với các

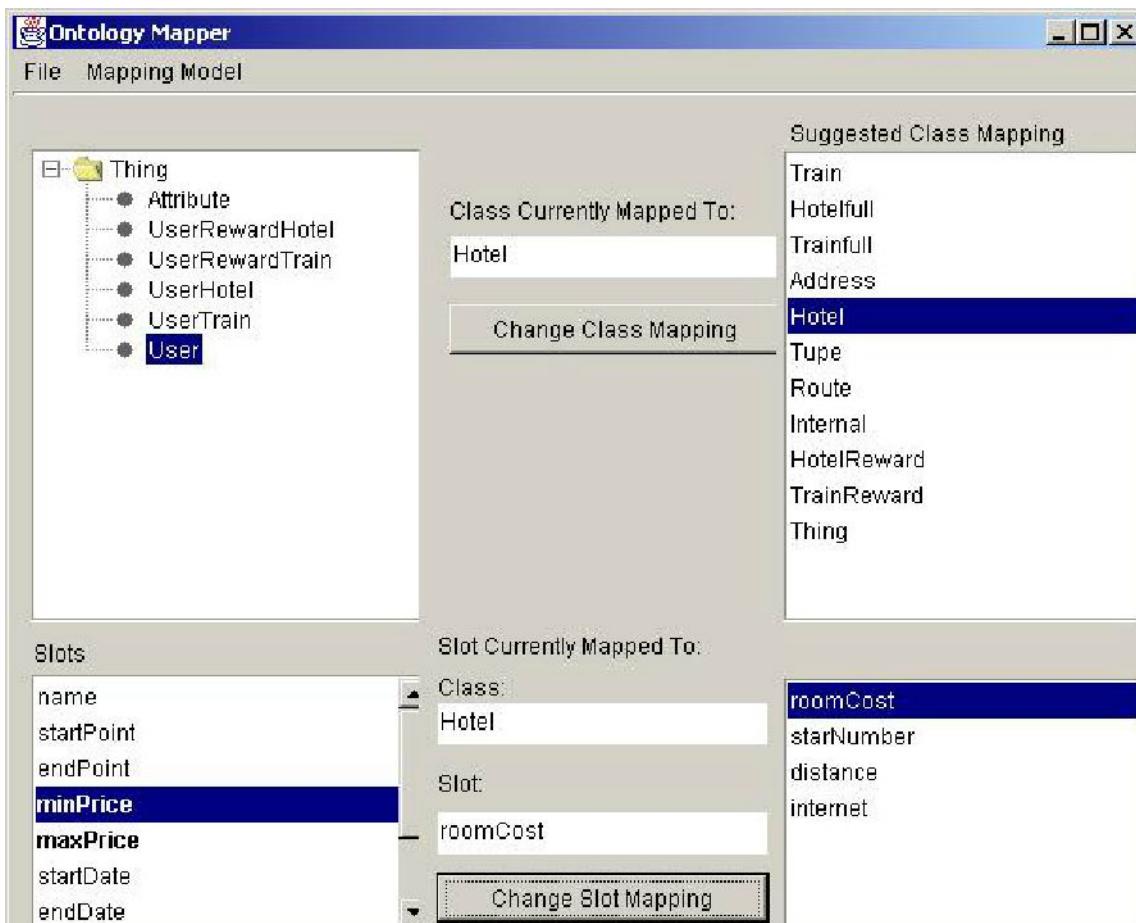
agent còn lại trong hệ thống và do đó, đã được mô tả trong ontology chung của hệ thống.

Với UserAgent, ngoài phần thông tin cần trao đổi với agent khác, nó còn có thêm các tri thức về sở thích và mong muốn của người dùng đã được mô hình hoá theo chiến lược thương lượng. Các thông tin này không phải bao giờ cũng được trao đổi với các agent khác vì mục đích bảo mật thông tin khách hàng. Do đó, các thông tin này không được đưa vào ontology chung của hệ thống mà sẽ được thiết kế như ontology riêng cho UserAgent.



Hình 6.5: Ontology riêng của UserAgent

Các bước xây dựng ontology cho riêng các agent cũng được tiến hành như việc xây dựng ontology cho hệ thống đã được trình bày chi tiết trong Bước 3. Ngoại trừ một khác biệt là tập các tri thức được xem xét chỉ giới hạn trong phạm vi của agent tương ứng. Hơn nữa, các tri thức liên quan đến agent đó đã được mô tả trong ontology của hệ thống sẽ không cần nhắc lại trong ontology riêng. Theo cách này, ontology riêng của UserAgent mô tả phần tri thức chưa được biểu diễn còn lại có cấu trúc như Hình 6.5.

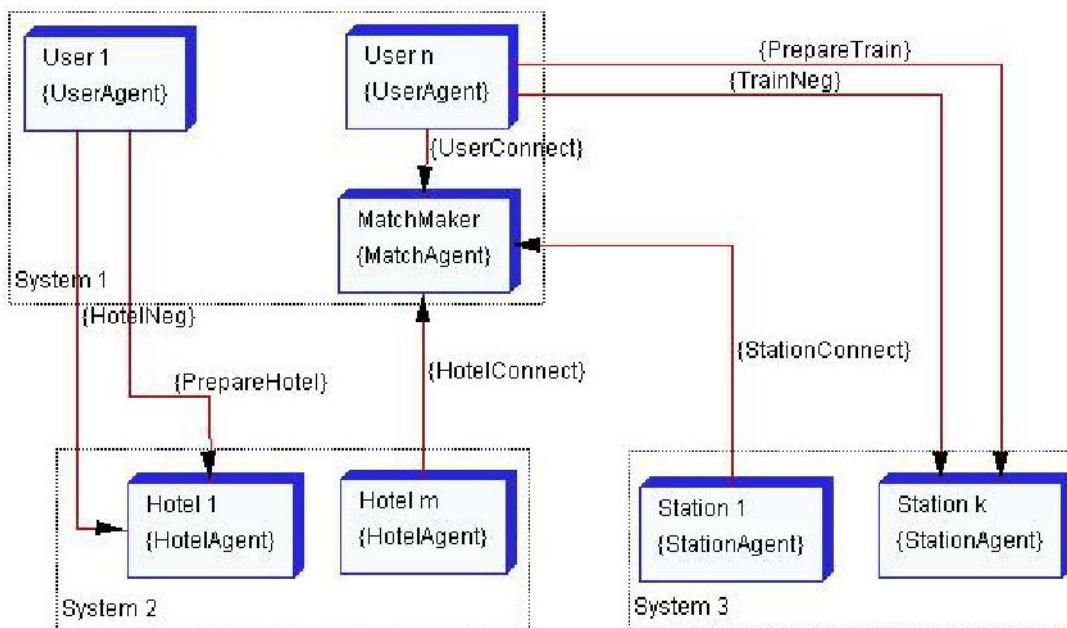


Hình 6.6: Ánh xạ ontology của UserAgent trong agentTool

Sau khi hoàn thành việc thiết kế ontology riêng cho các agent cần thiết, người thiết kế phải tiến hành ánh xạ ontology riêng của các agent này vào ontology chung của hệ thống. Việc này được hỗ trợ bởi cơ chế bán tự động của agentTool. Ví dụ với việc ánh xạ ontology riêng của UserAgent vào ontology chung của hệ thống. Trong UserAgent, khái niệm giá trọn gói của chuyến đi được biểu diễn thông qua cặp khái niệm $\{minPrice, maxPrice\}$. Trong ontology của hệ thống không có khái niệm biểu diễn giá trọn gói của chuyến đi mà chỉ có các khái niệm chỉ giá phòng khách sạn *RoomCost* và khái niệm chỉ giá vé tàu *TrainCost* là gần tương đương với cặp khái niệm trên. Do đó, cặp khái niệm $\{minPrice, maxPrice\}$ được ánh xạ là tương đương với các khái niệm *RoomCost* và *TrainCost*. Hình 6.6 mô tả việc ánh xạ này dưới sự hỗ trợ bán tự động của agentTool.

6.2.4 Triển khai hệ thống

Nhiệm vụ của bước cuối cùng này là xây dựng sơ đồ triển khai hệ thống (Deployment diagram).



Hình 6.7: Mô hình triển khai hệ thống

Hình 6.7 là mô hình triển khai của hệ dịch vụ du lịch. Hệ thống có một agent trung gian (MatchAgent), các agent còn lại đều có thể có số lượng nhiều hơn. Các UserAgent khi sinh ra đều được chạy trên máy chủ của MatchAgent (do máy cá nhân không đảm bảo kết nối mạng và hoạt động liên tục trong thời gian dài). Các agent HotelAgent và StationAgent có thể chạy trên các máy chủ khác nhau hay trên cùng một máy. Sau khi triển khai hệ thống, agentTool hỗ trợ sinh ra khung mã nguồn cho các lớp agent và các hàm đã được thiết kế. Nhờ đó, việc cài đặt chương trình sẽ thuận lợi hơn.

6.3 Kết luận

Chương này đã trình bày một cách khái quát về một số vấn đề thiết kế hệ đa agent và sau đó tập trung vào áp dụng các bước trong pha thiết kế để phát triển hệ dịch vụ du lịch. Chi tiết việc cài đặt và tích hợp hệ dịch vụ du lịch này sẽ được trình bày trong chương tiếp theo.

CHƯƠNG 7

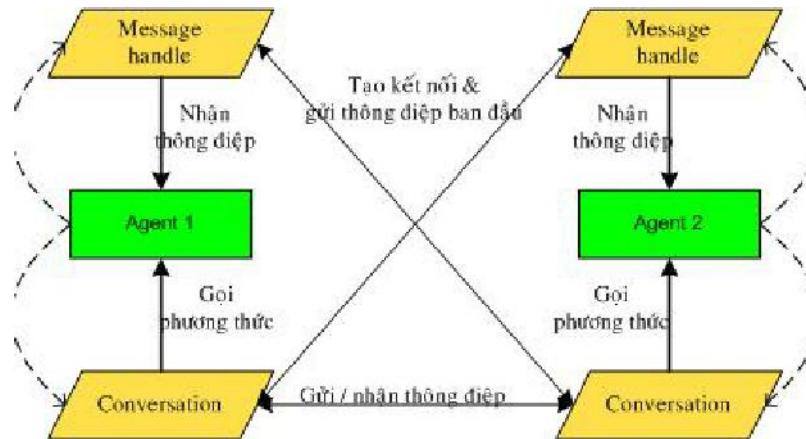
CÀI ĐẶT VÀ TÍCH HỢP HỆ THỐNG

- Vài nét về agentMom
- Mô hình tích hợp hệ thống
- Cài đặt các lớp agent
- Kết quả thử nghiệm
- Kết luận

Chương này trước hết trình bày các lớp agentMom mà nó được xem là cơ sở hạ tầng để hỗ trợ tương tác giữa các agent. Phần chủ yếu của chương dành trình bày kết quả cài đặt và tích hợp hệ thống dựa trên quá trình phân tích và thiết kế đã được trình bày trong Chương 5 và Chương 6.

7.1 Vài nét về agentMom

AgentMom là một tập các lớp làm nhân cho các ứng dụng phát triển hệ đa agent theo phương pháp luận MaSE và agentTool [5]. AgentMom hỗ trợ việc phát triển các lớp agent, các phiên hội thoại diễn ra giữa các agent và các thông điệp được trao đổi qua lại giữa các agent.



Hình 7.1: AgentMom

Hoạt động của các lớp trong agentMom được minh họa như Hình 7.1. Theo đó, mỗi agent sẽ được phép giao tiếp với các agent khác trong hệ thống sau khi nó khởi tạo phương thức `receiveMessage()` trong lớp `MessageHandle`, lớp này có nhiệm vụ giám sát và lắng nghe các kết nối từ các agent khác trên một cổng được dành riêng cho mỗi agent. Khi mỗi agent nhận được một kết nối từ agent khác và có khả năng giao tiếp được, nó sẽ khởi tạo một phiên hội thoại mới để bắt đầu cuộc trao đổi. Trong phiên hội thoại này, các thông điệp được trao đổi qua lại nhằm truyền đạt nội dung yêu cầu hoặc đáp ứng cho mỗi bên tham gia. Khuôn dạng thông điệp được mô tả tại lớp `Message`, kế thừa từ cấu trúc thông điệp chuẩn của ngôn ngữ giao tiếp truyền thông KQML (Knowledge Query and Manipulation Language). Phần này sẽ giới thiệu sơ lược nội dung của các lớp trong agentMom, chi tiết có thể tham khảo tại [5].

Lớp Agent, được kế thừa từ lớp `Runnable` của Java nên có khả năng chạy một cách độc lập như một phân tuyến (thread) riêng biệt trên máy chủ thông qua phương thức `run()`. Tất cả các lớp agent được thiết kế trong các ứng dụng đều phải kế thừa từ lớp này. Đáng chú ý nhất của lớp agent là phương thức ảo (abstract) `receiveMessage()`, dành cho các lớp agent kế thừa khả năng xử lý các thông điệp nhận được. Trong phương thức này, tuỳ vào nội dung của thông điệp nhận được mà agent khởi tạo các phiên hội thoại tương ứng cho phù hợp. Phương thức khởi tạo của lớp agent sẽ khởi

tạo một thê hiện của lớp `MessageHandler` để giám sát cổng kết nối riêng của nó, nhằm xác định các thông điệp do các agent khác gửi đến. Khi có thông điệp gửi đến, lớp này mới gọi lại phương thức `receiveMessage()` để xử lí thông điệp theo cách đã được thiết kế.

Lớp `Conversation` cũng là một lớp trừu tượng cho phép các lớp `conversation` trong ứng dụng kế thừa. Lớp này cung cấp các kết nối đến các agent khác và các hàm để gửi và nhận các thông điệp được trao đổi qua lại giữa các agent. Các kết nối được trang bị thông qua các cổng tương ứng của agent khởi tạo nó, khi đó, nó tạo ra các luồng đọc để nhận thông điệp và các luồng ghi giúp việc gửi thông điệp đi một cách thuận lợi.

Lớp `MessageHandler` được khởi tạo trong các phương thức khởi dụng cho lớp `agent`. Phương thức quan trọng của lớp này là phương thức `run()`, chạy liên tục cho đến khi agent tương ứng bị huỷ bỏ. Phương thức này chuyên lắng nghe các kết nối đến từ các agent khác trên một cổng xác định của agent. Khi có kết nối đến, nó gọi phương thức `receiveMessage()` của lớp `Agent` để xử lí.

Lớp `Message` định nghĩa cấu trúc thông điệp được trao đổi trong các phiên hội thoại giữa các agent. Cấu trúc lớp này được định nghĩa dựa trên cấu trúc thông điệp chuẩn của ngôn ngữ giao tiếp truyền thông KQML. Trong số các trường của lớp này, quan trọng là các trường như: `performative` quy định một loại yêu cầu cho bên nhận thông điệp, trường `content` chứa nội dung thông điệp, trường `ontology` xác định kiểu ontology được sử dụng trong giao tiếp đó. Hình 7.2 minh họa một số trường của lớp này.



Hình 7.2: Một thông điệp trong các Phiên hội thoại

7.2 Mô hình tích hợp hệ thống

Hệ thống được thiết kế như Hình 7.3, bao gồm bốn lớp agent: UserAgent, HotelAgent, TrainAgent và MatchAgent. Phần này sẽ trình bày tổng quan vai trò nhiệm vụ của các lớp agent này trong hệ thống. Chi tiết về các lớp agent này sẽ được trình bày trong các mục tiếp theo.

7.2.1 UserAgent

UserAgent là lớp agent đại diện cho chức năng và mục đích của khách hàng. Nó thay mặt khách hàng thương lượng với các agent cung cấp dịch vụ tương ứng là HotelAgent và TrainAgent. Việc thương lượng này được tiến hành một cách độc lập với khách hàng, nhằm tìm ra giải pháp thỏa mãn tốt nhất các ràng buộc của họ.

Lớp UserAgent có bốn thành phần chính: General Control, Hotel Negotiator, Train Negotiator và Preparer. Chức năng chính của mỗi thành phần như sau:

General Control

Điều khiển tổng quan các chức năng của UserAgent, bao gồm các hoạt động như:

- Tiếp xúc với khách hàng để trao đổi thông tin về các yêu cầu và ràng buộc về mặt hàng.
- Mô hình hóa yêu cầu người sử dụng để phù hợp với chiến lược thương lượng được sử dụng.
- Điều khiển tiến trình thương lượng của các thành phần Hotel Negotiator và Train Negotiator.
- Tổng hợp kết quả thương lượng của hai thành phần đó khi quá trình thương lượng kết thúc.
- Quyết định chấp nhận giải pháp vừa thương lượng xong hoặc không chấp nhận. Trong trường hợp không chấp nhận, nó phải quyết định thương lượng lại.

Hotel Negotiator

Thương lượng về dịch vụ khách sạn với các HotelAgent.

Train Negotiator

Thương lượng về dịch vụ tàu hỏa với TrainAgent.

Preparer

Nhằm thăm dò thông tin về các dịch vụ trước khi chính thức thương lượng. Thành phần này xuất hiện là do sử dụng chiến lược tiền ước lượng, sẽ được trình bày chi tiết trong phần UserAgent.

7.2.2 HotelAgent và TrainAgent

HotelAgent và TrainAgent là hai agent có mô hình tương tự nhau, bao gồm hai thành phần chính: phần điều khiển chung và phần thương lượng.

General Control

Điều khiển các hoạt động của Agent cung cấp dịch vụ, bao gồm các chức năng như:

- Quản lý các nhà cung cấp dịch vụ
- Quản lý các thông tin về dịch vụ.
- Điều khiển tiến trình thương lượng với các agent khách hàng (UserAgent).

Negotiator

Thay mặt nhà cung cấp dịch vụ thương lượng và cung cấp các dịch vụ cho khách hàng trong phạm vi khả năng của mỗi nhà cung cấp.

Chi tiết về hai lớp agent này sẽ được trình bày trong các phần về HotelAgent và TrainAgent.

7.2.3 MatchAgent

MatchAgent làm nhiệm vụ môi giới các UserAgent tìm được các agent cung cấp dịch vụ mà nó cần thương lượng. Nó có hai thành phần chính: điều khiển chung và phần môi giới.

General control

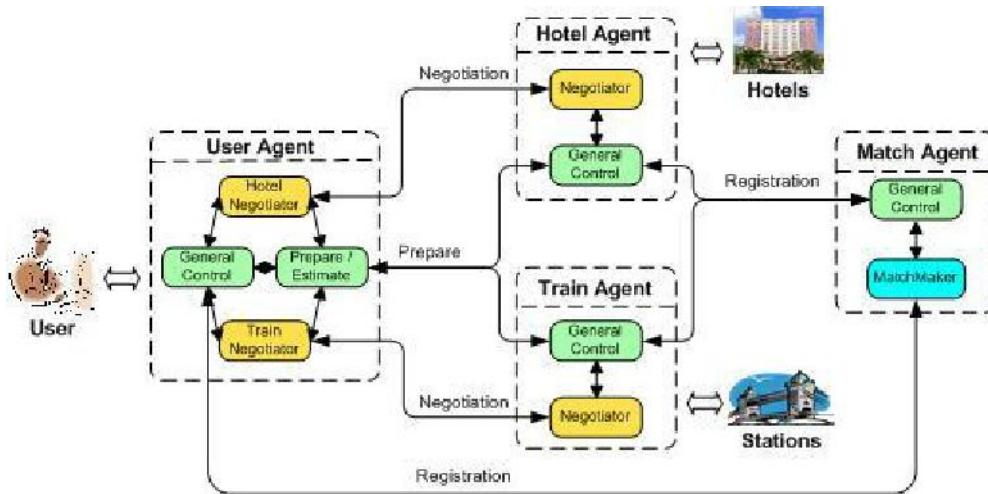
Điều khiển hoạt động của bản thân nó, bao gồm các hoạt động:

- Giám sát và quản lý trạng thái hoạt động của các agent trong hệ thống bằng cách lưu địa chỉ và khả năng của mỗi agent khi chúng tham gia vào hệ thống.
- Điều khiển quá trình môi giới và tương tác với các agent khác

MatchMaker

Môi giới các UserAgent với các HotelAgent và TrainAgent cần thiết cho UserAgent đó.

Trong bốn loại agent của hệ thống, chỉ có duy nhất MatchAgent là không có giao tiếp với con người. Nó được khởi động khi khởi tạo hệ thống và chạy nền trong suốt thời gian hoạt động của hệ thống này. Các agent còn lại có thể tham gia hay thoát khỏi hệ thống một cách tự do.



Hình 7.3: Kiến trúc tổng quan hệ thống

7.2.4 Hoạt động của hệ thống

Hệ thống hoạt động theo nguyên tắc như sau:

- MatchAgent được khởi động khi khởi tạo hệ thống và được chạy ngầm trong suốt thời gian hoạt động của hệ thống.
- Các HotelAgent và TrainAgent được tạo ra khi có các nhà cung cấp dịch vụ tham gia vào hệ thống và bị huỷ đi khi họ chấm dứt hợp tác với hệ thống.
- Các UserAgent được tạo ra mỗi khi có người dùng mới tham gia dịch vụ, nó bị huỷ đi khi đã hết hạn với người dùng.
- Trong số bốn loại agent trên, chỉ có MatchAgent là không trực tiếp tương tác với con người, các agent còn lại đều có thể tương tác với con người trong quá trình hoạt động của mình.
- Mỗi khi có một HotelAgent hoặc TrainAgent tham gia vào hệ thống, nó phải đăng ký hoạt động với MatchAgent bằng cách gửi đến MatchAgent một thông báo đăng ký bao gồm: tên, địa chỉ và dịch vụ mà nó có thể cung cấp. Các thông tin này được MatchAgent giữ lại để quản lý các agent có mặt trong hệ thống.
- Mỗi khi có một HotelAgent hoặc TrainAgent rút ra khỏi hệ thống, nó phải thông báo với MatchAgent để MatchAgent loại nó ra khỏi danh sách quản lý của mình.
- Mỗi khi có một UserAgent được tạo ra (ngay sau khi có một khách hàng mới tham gia vào hệ thống), nó phải đăng ký với MatchAgent bằng cách gửi đi một thông báo đăng ký bao gồm tên, địa chỉ của mình và các yêu cầu dịch vụ mà nó mong muốn.
- Khi nhận được một thông báo đăng ký từ UserAgent, MatchAgent lưu lại các thông tin này để quản lý. Đồng thời dựa trên các yêu cầu dịch vụ của UserAgent

để tìm ra các HotelAgent và TrainAgent phù hợp với UserAgent đó. Nghĩa là MatchAgent sẽ giới thiệu với UserAgent các agent có khả năng cung cấp dịch vụ theo yêu cầu của UserAgent.

- Sau khi tìm được các agent phù hợp cho UserAgent, MatchAgent sẽ gửi lại cho UserAgent các thông tin về địa chỉ của các agent đó.
- Sau khi nhận được địa chỉ các agent cung cấp dịch vụ khách sạn và tàu hoả, UserAgent tiến hành thăm dò sản phẩm (tiền ước lượng) và thương lượng trực tiếp với các agent đó.
- Kết quả thương lượng, nếu đã có, sẽ được gửi đến khách hàng trong lần đăng nhập tiếp theo vào hệ thống.
- Nếu muốn đặt chỗ tự động, khách hàng có thể ủy quyền cho hệ thống thực hiện. Khi đó, hệ thống tiến hành đặt vé tàu và đặt chỗ khách sạn theo yêu cầu của khách hàng.
- Sau khi khách hàng đã đăng ký đặt chỗ tự động hoặc hết hạn nhận kết quả, các UserAgent sẽ bị huỷ đi, nhường tài nguyên cho các Agent khác.

7.3 Cài đặt các lớp agent

Nội dung phần này sẽ trình bày việc cài đặt chi tiết các lớp agent đã được thiết kế trong chương 5. Mỗi lớp agent sẽ được trình bày theo các nội dung: Sơ đồ kiến trúc chi tiết bên trong, sơ đồ hoạt động và cuối cùng là các phương thức chính của mỗi lớp đó.

7.3.1 UserAgent

UserAgent được thiết kế để thay mặt khách hàng thương lượng với HotelAgent và TrainAgent để đặt chỗ trọn gói cho chuyến đi. Như vậy, UserAgent có các nhiệm vụ và hoạt động chính như sau:

- Giao tiếp với người dùng để thu thập yêu cầu và thông báo kết quả.
- Tương tác với MatchAgent để yêu cầu môi giới.
- Thương lượng với HoetlAgent để đặt chỗ khách sạn
- Thương lượng với TrainAgent để đặt vé tàu.

Nội dung chi tiết phần cài đặt cho lớp UserAgent như sau.

a. Sơ đồ cấu trúc và chức năng

Dựa vào kiến trúc bên trong đã được thiết kế trong chương 5. Sơ đồ chức năng của UserAgent được xây dựng như Hình 7.4. UserAgent có các thành phần chính: phần điều khiển tổng quan, phần tiền ước lượng, hai thành phần thương lượng khách sạn và tàu hoả là tương tự nhau.

Phần điều khiển tổng quan (General Control)

Bộ xử lí (http://)

hoạt động dựa trên giao thức http và TCP/IP để tương tác với khách hàng thông qua giao diện web. Bao gồm:

- Nhận thông tin quản lý cá nhân và sở thích, mong muốn của khách hàng về dịch vụ họ muốn được cung cấp.
- Thông báo kết quả thương lượng cho khách hàng.
- Tương tác với khách hàng để có thể thay mặt họ đặt chỗ tự động các dịch vụ đã thỏa thuận.
- Cho phép người dùng có thể điều chỉnh các yêu cầu của mình trong thời gian **UserAgent** của người đó còn hiệu lực.

Các thông tin nhận được từ khách hàng đồng thời cũng được lưu vào cơ sở dữ liệu để tiện cho việc quản lý khách hàng. Phần này được cài đặt bằng JSP, chạy trên máy chủ Jrun 3.0.

Bộ đăng ký với hệ thống (Register)

Sau khi nhận thông tin của khách hàng từ bộ xử lí http, bộ đăng ký hệ thống sẽ tạo một thông điệp đăng ký chứa địa chỉ của agent đó và yêu cầu về các loại dịch vụ nó cần. Thông điệp này sẽ được gửi đến MatchAgent để được môi giới với các đối tác phù hợp cho việc thương lượng sắp tới.

Bộ duyệt sản phẩm (Adapter)

Có nhiệm vụ xem xét các sản phẩm sau khi thương lượng có thỏa mãn ràng buộc lẫn nhau hay không. Các ràng buộc này là không kiểm tra được trong quá trình thương lượng mà chỉ có thể hậu xử lí, chẳng hạn các ràng buộc về thời gian của đi tàu và thời gian thuê khách sạn là phải tương thích với nhau. Nếu có ít nhất một ràng buộc không thỏa mãn, bộ này sẽ điều khiển việc thương lượng lại bằng cách chấp nhận dịch vụ tàu hoả và thương lượng lại dịch vụ khách sạn cho tương thích. Nếu không có vi phạm ràng buộc nào, bộ này sẽ lưu kết quả thương lượng và quá trình thương lượng chính thức kết thúc.

Thành phần tiền thương lượng (Preparer)

Sau khi gửi đi thông điệp đăng ký vào hệ thống, có kèm theo các yêu cầu về đối tác. **UserAgent** sẽ được MatchAgent môi giới với các đối tác phù hợp. Địa chỉ của các đối tác này sẽ được gửi đến cho **UserAgent** ngay sau khi MatchAgent môi giới xong. Sau khi nhận được địa chỉ của các đối tác bao gồm **HotelAgent** và **TrainAgent**, **UserAgent** bắt đầu tiến trình thăm dò tiền thương lượng.

Bộ tạo yêu cầu thăm dò

Tạo ra các thông điệp thăm dò đến HotelAgent và TrainAgent. Thông điệp này chứa các câu hỏi thăm dò về giá của các dịch vụ tương ứng.

Bộ tiên ước lượng

Sau khi nhận được các thông điệp trả lời từ HotelAgent và TrainAgent, bộ tiên ước lượng (Pre-Estimator) bắt đầu hoạt động. Nó dựa vào các thông tin giá dịch vụ vừa nhận được này và ràng buộc yêu cầu của khách hàng để ước lượng giá khởi điểm phù hợp cho từng dịch vụ: khách sạn và vé tàu. Sau khi ước lượng xong, giai đoạn thương lượng chính thức bắt đầu.

Thành phần thương lượng

Bộ xử lý thông điệp (Message Processor)

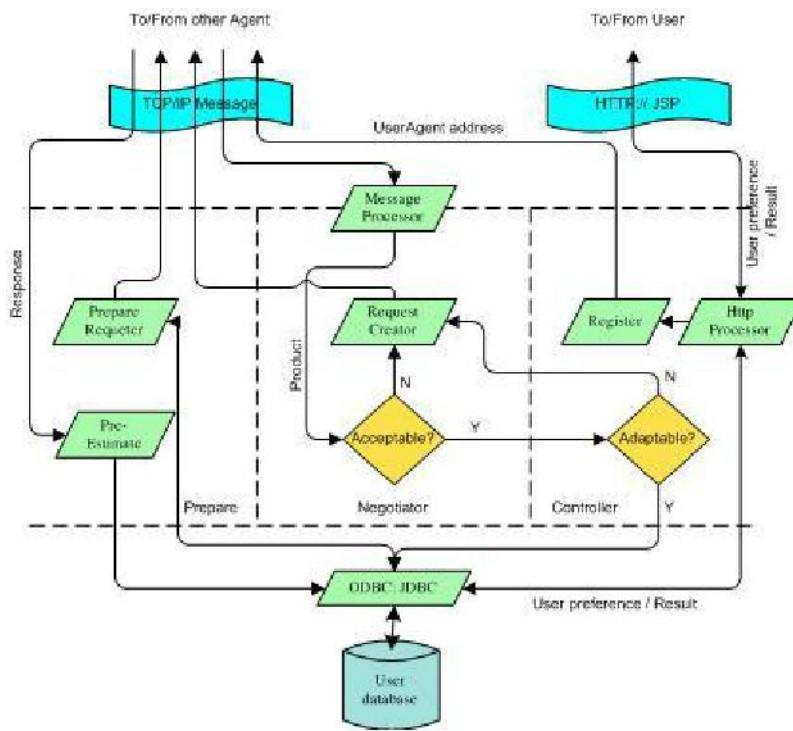
Nhận và lọc các thông điệp nhận được, Lấy ra các thông tin được các đối tác gửi đến. Các thông tin này chính là các thông tin về sản phẩm dịch vụ mà các nhà cung cấp có thể có. Thông tin này sẽ được đưa vào bộ kiểm tra chấp nhận xem có chấp nhận được hay không. Các thông điệp trao đổi giữa các agent được tuân theo chuẩn của ngôn ngữ truyền thông ACL.

Bộ tạo thông điệp (Message Creator)

Tạo ra các thông điệp để yêu cầu đối tác tìm kiếm các sản phẩm thỏa mãn một số yêu cầu mới bổ sung hoặc sau khi nhượng bộ...

Bộ quyết định chấp nhận (Acceptable)

Có nhiệm vụ đánh giá xem sản phẩm của đối tác là có thể chấp nhận được hay không.



Hình 7.4: Kiến trúc UserAgent

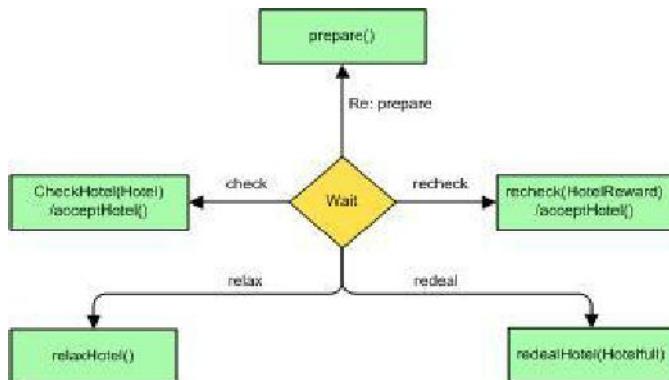
b. Sơ đồ hoạt động

Tương tự như sơ đồ thiết kế trong chương 7, sơ đồ hoạt động của UserAgent bao gồm thương lượng với HotelAgent và TrainAgent. Phần này trình bày mối liên hệ và tương tác giữa các phương thức chính của lớp UserAgent.

Thương lượng với HotelAgent bao gồm các phương thức có quan hệ như Hình 7.5. Các phương thức này được gọi trong thủ tục thương lượng khách sạn như sau:

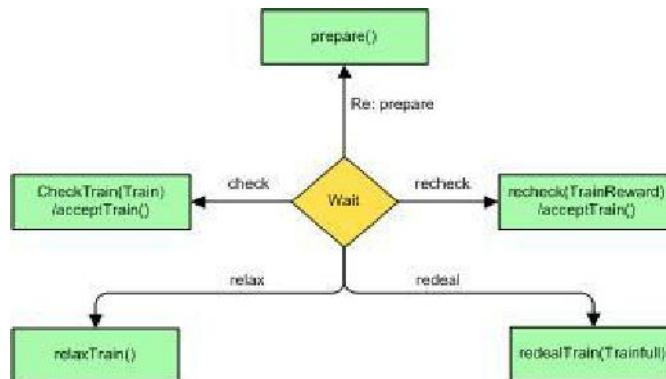
- Nếu nhận được thông điệp check thì gọi phương thức checkHotel(Hotel) để kiểm tra xem khách sạn đó có vi phạm ràng buộc nào hay không. Có thể dùng tới phương thức acceptHotel() để kiểm tra xem khách sạn có chấp nhận được hay không nếu như nó không có vi phạm ràng buộc.
- Nếu nhận được thông điệp recheck thì gọi phương thức recheckHotel(HotelReward) để kiểm tra xem khách sạn vừa bị từ chối trước đó được bổ sung các hình thức khuyến mại thì có chấp nhận được hay không. Có thể dùng đến phương thức acceptHotel().
- Nếu nhận được thông điệp relax thì gọi phương thức relaxHotel() để xem xét việc còn có thể nhượng bộ trên thuộc tính nào đấy hay không..

- Nếu nhận được thông điệp redeal thì gọi phương thức redealHotel(Hotelfull) để lưu lại kết quả thương lượng là thông tin đầy đủ về khách sạn mà nó vừa chấp nhận.



Hình 7.5: Thương lượng với HotelAgent

Thương lượng với TrainAgent cũng có các phương thức được sử dụng cho Train và được gọi hoàn toàn tương tự như thương lượng với HotelAgent. Việc gọi các phương thức thương lượng với TrainAgent được mô tả trong Hình 7.6.



Hình 7.6: Thương lượng với TrainAgent

c. Các phương thức chính của lớp UserAgent

Lớp UserAgent có các phương thức chính được tóm tắt như sau:

`Void prepare()`

Tiền uớc lượng giá dịch vụ khách sạn và tàu hoả. Tham số vào và tham số ra của phương thức này đều thông qua các biến toàn cục của lớp UserAgent.

Tupe checkHotel (Hotel hotel)

Kiểm tra xem khách sạn có vi phạm ràng buộc nào không. Nếu không có vi phạm, trả về NULL. Nếu có vi phạm, trả về các cặp Tupe(< thuộc tính >, < giá trị >) của khách sạn theo ràng buộc. Các tham số bao gồm các thuộc tính để thương lượng: Hotel: hotel là kiểu dữ liệu khách sạn.

Tupe checkTrain (Train train) tương tự.

boolean acceptHotel (Hotel hotel)

Kiểm tra xem khách sạn có thẻ được người dùng chấp nhận hay không. Nếu có, trả về TRUE, nếu không, trả về FALSE. Tham số vẫn là Hotel.

boolean acceptTrain (Train train) tương tự.

boolean recheckHotel (HotelReward reward)

Kiểm tra xem khách sạn có bồ sung khuyến mại thì có thẻ chấp nhận được hay không. Nếu có thẻ, trả về TRUE, ngược lại, trả về FALSE. Tham số: HotelReward reward: Kiểu dữ liệu mô tả các hình thức khuyến mại của khách sạn.

boolean recheckTrain (TrainReward reward) tương tự.

Tupe relaxHotel ()

Kiểm tra xem có thẻ nhượng bộ được nữa hay không. Nếu có thẻ nhượng bộ được, trả về các cặp Tupe(< Thuộc tính >, < giá trị >) đối với thuộc tính còn nhượng bộ được. Nếu không thẻ nhượng bộ thêm, trả về NULL.

Tupe relaxTrain () tương tự.

void redealHotel (Hotelfull hotel)

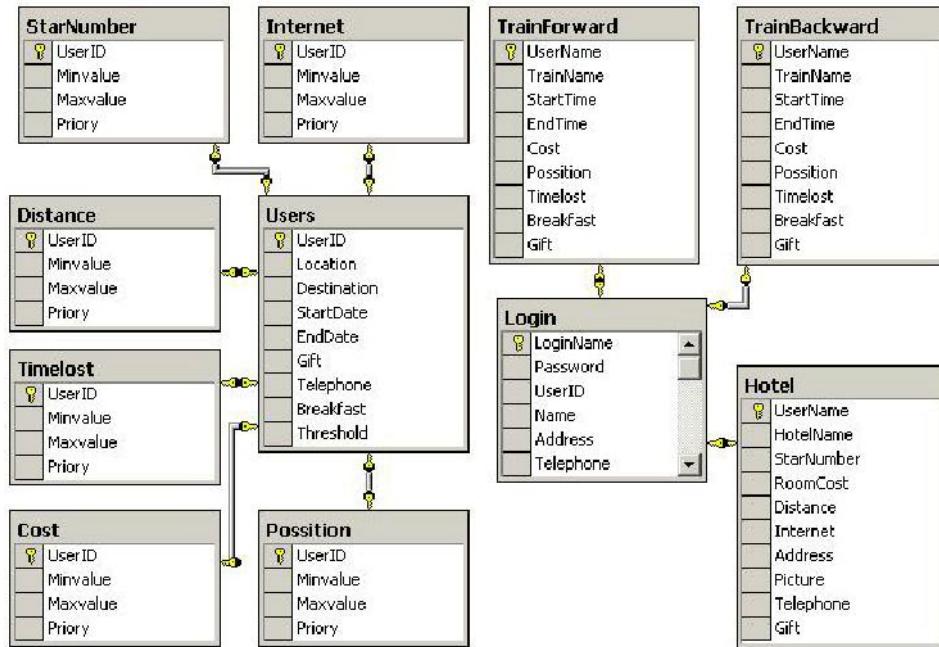
Xử lý kết quả thương lượng. Tham số:

Hotelfull hotel: kiểu dữ liệu mô tả đầy đủ thông tin về khách sạn vừa thương lượng xong.

void redealTrain (Trainfull train) tương tự.

d. Cơ sở dữ liệu của UserAgent

Cơ sở dữ liệu của UserAgent được tổ chức thành các bảng có quan hệ như Hình 7.7.



Hình 7.7: Cơ sở dữ liệu của UserAgent

Trong cơ sở dữ liệu này có các bảng như sau:

Bảng Login: lưu các thông tin đăng nhập của người dùng, bao gồm các thông tin về tên đăng nhập, mật khẩu, ngày đăng kí...

Bảng Users: lưu các thông tin cá nhân của khách hàng để phục vụ cho việc liên hệ khi đặt chỗ tự động và thông báo kết quả. Bao gồm tên thật, địa chỉ, số điện thoại...

Các bảng StartNumber, Internet, Cost, TimeLost, Distance và Position: Lưu các thông tin về sở thích của khách hàng về các điều kiện của dịch vụ. Mỗi bảng đều có cấu trúc bao gồm: ID khách hàng, độ quan trọng của thuộc tính, giá trị nhỏ nhất và cao nhất về thuộc tính đó mà khách hàng chấp nhận được.

Các bảng TrainForward, TrainBackward và Hotel: lưu thông tin về chuyến tàu đi, chuyến tàu về và khách sạn đã thương lượng được cho khách hàng đó. Các trường thông tin trong các bảng này là tương ứng với cấu trúc của các lớp Trainfull và Hotelfull đã được đề cập trong pha thiết kế.

7.3.2 HotelAgent

HotelAgent đại diện cho các khách sạn thương lượng với khách hàng để cung cấp dịch vụ khách sạn cho khách hàng có nhu cầu. Khác với UserAgent, HotelAgent chỉ quan tâm thương lượng trên một thuộc tính là giá cả của dịch vụ khách sạn. Do đó, kiến trúc và chiến lược thương lượng của nó cũng đơn giản hơn UserAgent.

a. Mô hình kiến trúc

Như đã trình bày trong phần kiến trúc tổng quan, HotelAgent có hai thành phần chính: phần điều khiển tổng quan và phần thương lượng. Mỗi quan hệ bên trong của hai thành phần này được mô tả trong Hình 7.8.

Thành phần điều khiển chung

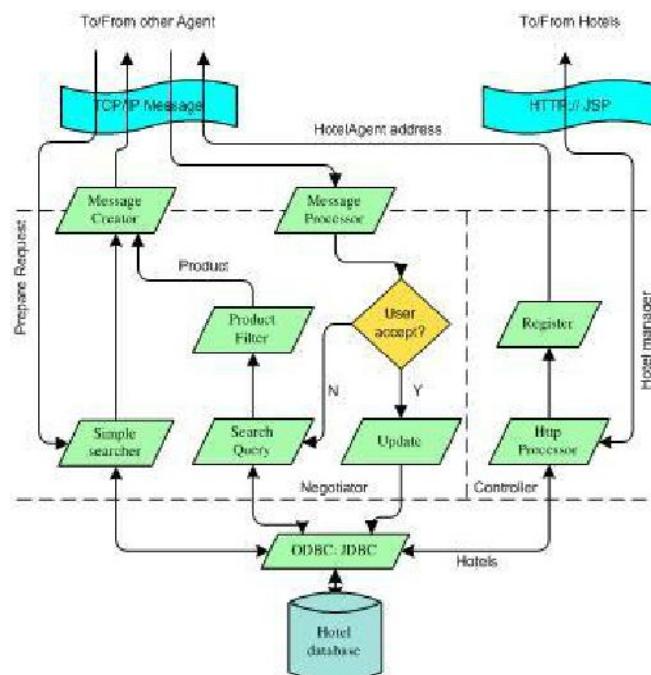
Thành phần này bao gồm bộ xử lí (<http://>) và bộ đăng kí hệ thống.

Bộ xử lí http

tương tác với các khách sạn thông qua giao diện web, bao gồm các hoạt động: (i) Thu nhận và quản lí các thông tin đăng kí, đăng nhập của các khách sạn; (ii) Cập nhật thông tin về dịch vụ của các khách sạn.

Bộ đăng kí

Các khách sạn được quản lí theo khu vực (các thành phố). Do đó, khi một khách sạn đăng kí tham gia vào hệ thống nằm ở khu vực do một agent nào đó đang quản lí, thì nó sẽ chịu sự quản lí của agent này. Nếu khu vực của khách sạn đó là mới, sẽ có một HotelAgent tạo ra để quản lí các khách sạn theo khu vực đó. Khi đó, bộ đăng kí sẽ gửi đến MatchAgent các thông tin về địa chỉ và dịch vụ của mình để đăng kí tham gia vào hệ thống.



Hình 7.8: Kiến trúc HotelAgent

Thành phần thương lượng

Bộ tạo thông điệp Message creator

Tạo ra các thông điệp để gửi đến các agent khác trong hệ thống. Bộ xử lý thông điệp có nhiệm vụ lọc và phân loại các thông điệp được gửi cho HotelAgent.

Bộ tìm kiếm đơn giản Simple search

Dùng để tìm kiếm giá dịch vụ khách sạn cho các yêu cầu tiền ước lượng từ phía UserAgent.

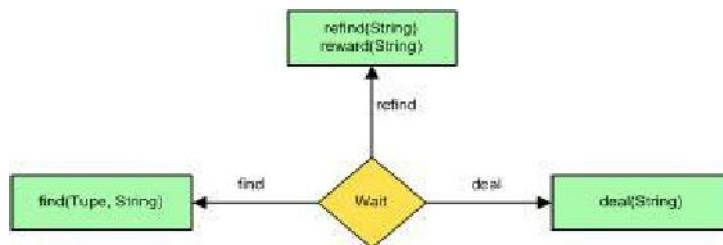
Bộ kiểm tra sự chấp nhận của khách hàng User accept

Nếu khách hàng chấp nhận thì cập nhật lại thông tin dịch vụ qua bộ cập nhật Update. Nếu không, phải thông qua bộ tìm kiếm Search Query và bộ lọc Hotel filter để tìm ra khách sạn phù hợp với các ràng buộc của khách hàng.

b. Sơ đồ hoạt động

Sơ đồ 7.9 mô tả mối liên hệ và tương tác giữa các phương thức của lớp HotelAgent:

- Nếu nhận được thông điệp find thì gọi phương thức find(Tupe, String) để tìm ra một khách sạn phù hợp với các ràng buộc trong Tupe.
- Nếu nhận được thông điệp refind thì gọi phương thức refind(String) và/hoặc phương thức reward(String) để tìm lại một khách sạn khác và/hoặc tìm khuyến mại bổ sung cho khách sạn trước đó.
- Nếu nhận được thông điệp deal thì gọi phương thức deal(String) để tìm toàn bộ thông tin liên quan đến khách sạn mà UserAgent vừa chấp nhận.



Hình 7.9: Quan hệ giữa các phương thức của lớp HotelAgent

c.Tóm tắt các biến và hàm của lớp HotelAgent

Lớp HotelAgent có một số phương thức cơ bản sau đây:

Hotel find(Tupe tupe, String userAgentName)

Tìm kiếm khách sạn phù hợp với các ràng buộc bổ sung trong tham số tuple. Tham số thứ hai là tên của agent khách hàng. Nếu tìm được, trả về một khách sạn, nếu không sẽ trả về NULL.

`Hotel refind(String userAgentName)`

Tìm kiếm khách sạn mà không bổ sung thêm ràng buộc nào. Tham số duy nhất là tên của agent khách hàng. Nếu tìm được, trả về một khách sạn, nếu không, trả về NULL.

`HotelReward reward(String userAgentName)`

Tìm kiếm khuyến mại của một khách sạn vừa giới thiệu cho khách hàng được định danh trong tham số *userAgentName*. Nếu có, trả về kiểu khuyến mại của khách sạn, nếu không sẽ trả về NULL.

`Hotelfull deal(String userAgentName)`

Tìm kiếm thông tin đầy đủ về khách sạn vừa được khách hàng chấp nhận, khách hàng này được định danh qua tham số *userAgentName*.

d. Cơ sở dữ liệu

Cơ sở dữ liệu của HotelAgent được tổ chức như Hình 7.10. Trong đó:

Bảng Hotel: lưu giữ toàn bộ thông tin về các khách sạn mà nó quản lý, bao gồm các thông tin có thể thương lượng như: StarMunber, Distance, Cost... và các thuộc tính bổ sung như: Address, picture...

Bảng Reward: lưu thông tin lưu các thông tin về khuyến mại của mỗi khách sạn. Ví có thể có hai hình thức khuyến mại được chấp nhận là “Miễn phí điện thoại nội hat” và “Tặng quà”, nên mỗi hình thức khuyến mại của mỗi khách sạn sẽ được lưu dưới dạng biến boolean: có hoặc không.

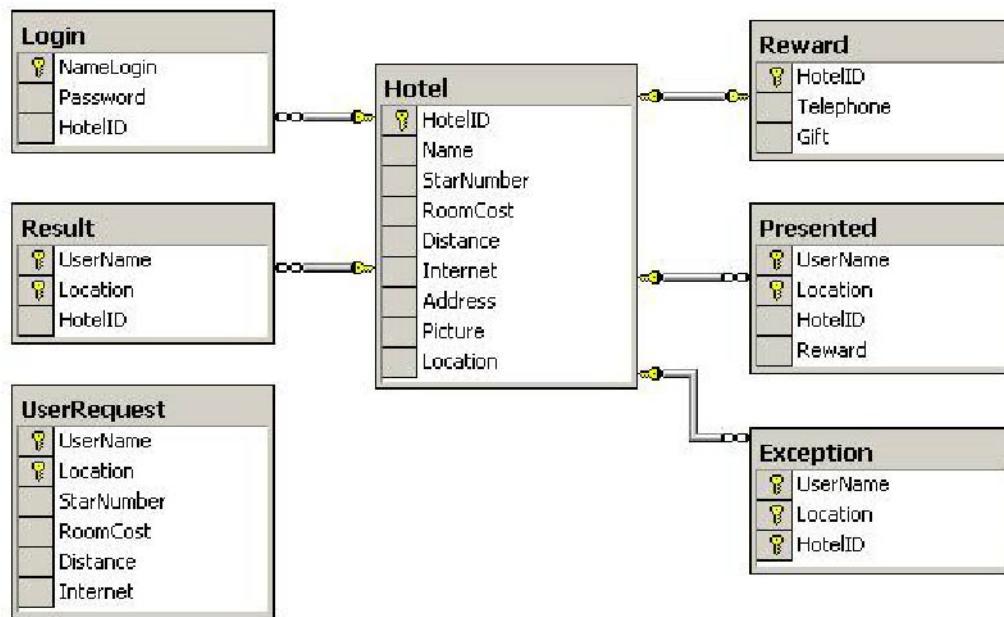
Bảng Login: lưu thông tin đăng nhập của mỗi khách sạn, bao gồm: tên đăng nhập, mật khẩu, ID của khách sạn...

Bảng UserRequest: lưu các thông tin về giá trị các thuộc tính mà mỗi UserAgent thương lượng với nó đã gửi đến. Giá trị thuộc tính nào chưa gửi đến sẽ được mặc định là NULL.

Bảng Presented: lưu danh sách khách sạn mới nhất vừa được giới thiệu cho mỗi khách hàng đang thương lượng với nó.

Bảng Exception: lưu danh sách các khách sạn đã bị UserAgent từ chối trong quá trình thương lượng, tính cho đến thời điểm hiện tại.

Bảng Result: lưu thông tin về khách sạn nào đã được UserAgent chấp nhận.



Hình 7.10: Cơ sở dữ liệu của HotelAgent

7.3.3 TrainAgent

TrainAgent đại diện cho các nhà ga thương lượng với khách hàng để cung cấp dịch vụ khách sạn cho khách hàng có nhu cầu. Sơ đồ kiến trúc và hoạt động của TrainAgent tương tự như HotelAgent, chỉ khác là nó làm đại diện cho các nhà ga mà thôi.

a. Mô hình kiến trúc

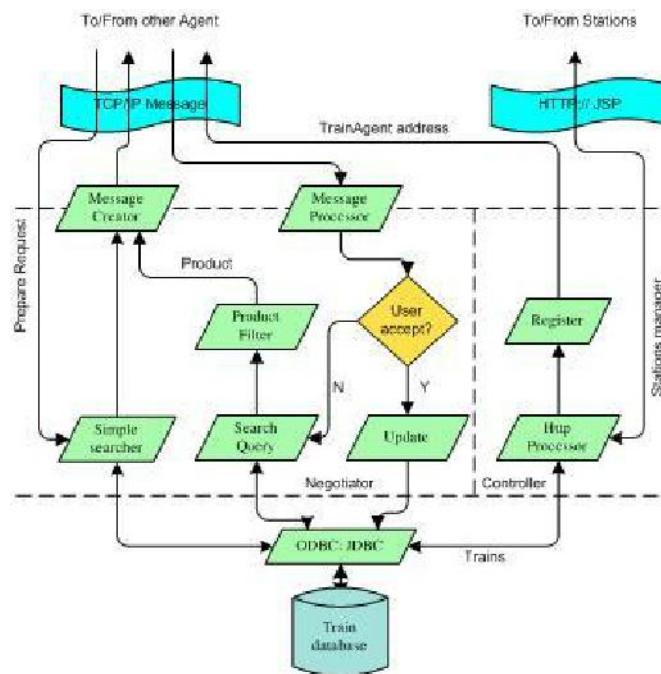
Như đã trình bày trong phần kiến trúc tổng quan, TrainAgent có hai thành phần chính: phần điều khiển tổng quan và phần thương lượng. Mỗi quan hệ bên trong của hai thành phần này được mô tả trong Hình 7.11.

Thành phần điều khiển chung

Thành phần này bao gồm bộ xử lí (<http://>) và bộ đăng ký hệ thống.

Bộ xử lí http: tương tác với các nhà ga thông qua giao diện web, bao gồm các hoạt động: (i) Thu nhận và quản lý các thông tin đăng ký, đăng nhập của các nhà ga; (ii) Cập nhật thông tin về dịch vụ của các nhà ga.

Bộ đăng ký: Các nhà ga được quản lý theo khu vực (các thành phố). Do đó, khi có nhà ga đăng ký tham gia hệ thống, nếu nhà ga nằm ở khu vực do một agent nào đó đang quản lý, thì nó sẽ chịu sự quản lý của agent này. Nếu khu vực của nhà ga đó là mới, sẽ có một TrainAgent tạo ra để quản lý các nhà ga theo khu vực đó. Khi đó, bộ đăng ký sẽ gửi đến MatchAgent các thông tin về địa chỉ và dịch vụ của mình để đăng ký tham gia vào hệ thống.



Hình 7.11: Kiến trúc TrainAgent

Thành phần thương lượng

Bộ tạo thông điệp Message creator: tạo ra các thông điệp để gửi đến các agent khác trong hệ thống. Bộ xử lý thông điệp lọc và phân loại các thông điệp được gửi cho TrainAgent.

Bộ tìm kiếm đơn giản Simple search: dùng để tìm kiếm giá vé tàu cho các yêu cầu tiền ước lượng từ phía UserAgent.

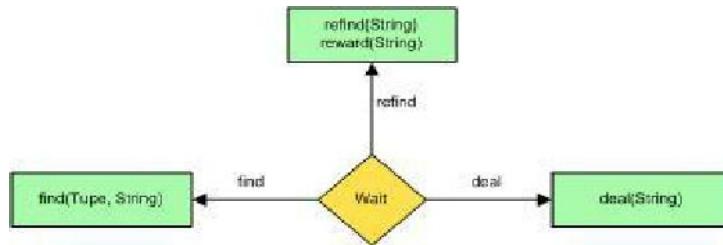
Bộ kiểm tra sự chấp nhận của khách hàng User accept: nếu khách hàng chấp nhận thì cập nhật lại thông tin dịch vụ qua bộ cập nhật Update. Nếu không, phải thông qua bộ tìm kiếm Search Query và bộ lọc Train filter để tìm ra chuyến tàu phù hợp với các ràng buộc của khách hàng.

b. Sơ đồ hoạt động

Hình 7.12 mô tả mối liên hệ giữa các phương thức chính của lớp TrainAgent:

- Nếu nhận được thông điệp find thì gọi phương thức find(Tupe, String) để tìm chuyến tàu thỏa mãn các ràng buộc trong Tupe.
- Nếu nhận được thông điệp refind thì gọi phương thức refind(String) để tìm lại chuyến tàu mới và / hoặc gọi phương thức reward(String) để bổ sung khuyến mại cho chuyến tàu vừa bị từ chối.

- Nếu nhận được thông điệp deal thì gọi phương thức deal(String) để tìm toàn bộ thông tin liên quan đến chuyến tàu mà UserAgent vừa chấp nhận.



Hình 7.12: Quan hệ giữa các phương thức của lớp TrainAgent

c. Tóm tắt các biến và hàm của lớp TrainAgent

Lớp TrainAgent có một số phương thức cơ bản sau:

Train find(Tupe tuge, String userAgentName)

Tìm kiếm chuyến tàu phù hợp với các ràng buộc bổ sung trong tham số tuge. Tham số thứ hai là tên của agent khách hàng. Nếu tìm được, trả về một chuyến tàu, nếu không sẽ trả về NULL.

Train refind(String userAgentName)

Tìm kiếm chuyến tàu mới mà không bổ sung thêm ràng buộc nào. Tham số duy nhất là tên của agent khách hàng. Nếu tìm được, trả về một chuyến tàu, nếu không, trả về NULL.

TrainReward reward(String userAgentName)

Tìm kiếm khuyến mại của một chuyến tàu vừa giới thiệu cho khách hàng được định danh trong tham số userAgentName. Nếu có, trả về kiểu khuyến mại của chuyến tàu, nếu không sẽ trả về NULL.

Trainfull deal(String userAgentName) :

Tìm kiếm thông tin đầy đủ về chuyến tàu vừa được khách hàng chấp nhận, khách hàng này được định danh qua tham số userAgentName.

d. Cơ sở dữ liệu

Cơ sở dữ liệu của TrainAgent được tổ chức như Hình 7.13. Trong đó:

Bảng Login: quản lý các thông tin đăng nhập hệ thống và tài khoản trong hệ thống của các nhà ga.

Bảng Route: Thông tin về chặng đường (nơi đi, nơi đến) của mỗi chuyến tàu.

Bảng Train: Thông tin về mỗi chuyến tàu: tên, thời gian chạy trên mỗi chặng...

Bảng Cost: Thông tin về các vị trí chỗ ngồi trên mỗi chuyến tàu: hạng của ghế, giá vé...

Bảng Capacity: Lưu thông tin về số chỗ trống của mỗi hạng vé trên mỗi chuyến tàu.

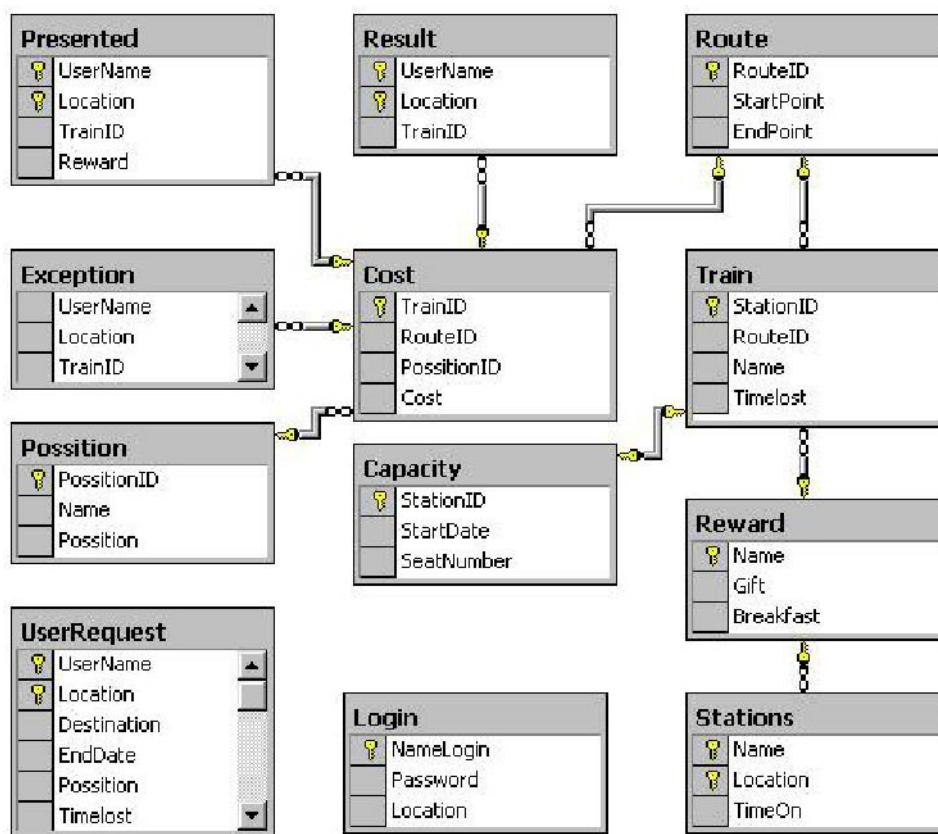
Bảng Reward: Hình thức khuyến mại có thể có của mỗi chuyến tàu.

Bảng Presented: Lưu các chuyến tàu đã giới thiệu cho khách hàng.

Bảng Exception: Lưu các chuyến tàu đã bị từ chối của mỗi khách hàng.

Bảng UserRequest: Lưu các yêu cầu ràng buộc của khách hàng về chuyến tàu họ muốn đi.

Bảng Result: Lưu kết quả thương lượng thành công với khách hàng.



Hình 7.13: Cơ sở dữ liệu của TrainAgent

7.3.4 MatchAgent

MatchAgent là một dạng agent đặc biệt:

- Không trực tiếp giao tiếp với con người hay các nguồn điều khiển trực tiếp từ bên ngoài khác, chỉ giao tiếp với các agent trong hệ thống
- Chỉ chạy nền cho hệ thống, được sinh ra khi khởi tạo hệ thống và chỉ chết đi khi hệ thống ngừng hoạt động.

MatchAgent có hai chức năng chính:

- Quản lý các agent trong hệ thống.

- Môi giới cho thương lượng giữa UserAgent với HotelAgent và TrainAgent.

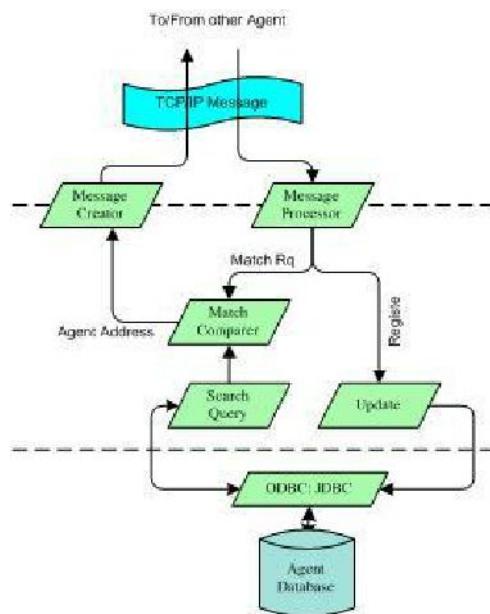
a. Mô hình kiến trúc

Kiến trúc bên trong của MatchAgent được mô tả trong Hình 7.14, bao gồm:

Bộ tạo thông điệp Message Creator: tạo ra các thông điệp để gửi đi các agent khác. Bộ xử lí thông điệp message processor xử lí và lọc các thông điệp nó nhận được.

Bộ đối sánh sự phù hợp Match Compare: làm nhiệm vụ đối sánh sự khà hợp giữa UserAgent với HotelAgent và TrainAgent. Bộ này cần sự trợ giúp của bộ xử lí truy vấn Search Query.

để cập nhật các thông tin về các agent khi chúng gửi thông báo đăng kí đến.



Hình 7.14: Kiến trúc MatchAgent

b. Sơ đồ hoạt động

Hình 7.15 mô tả mối quan hệ giữa các phương thức chính của lớp MatchAgent:

- Nếu nhận được thông tin đăng kí của HotelAgent thì gọi phương thức saveHotel(HotelAddress) để lưu lại địa chỉ và khả năng của agent đó, phục vụ công việc môi giới sau này.
- Tương tự, khi nhận được thông điệp đăng kí của các TrainAgent thì gọi phương thức saveTrain(TrainAddress) để lưu lại khả năng và địa chỉ của các agent đó.

- Nếu nhận được yêu cầu môi giới của UserAgent thì gọi hai phương thức: searchHotel(Location) và searchTrain(Location). Phương thức searchHotel(Location) dùng để tìm ra HotelAgent có thể cung cấp dịch vụ Hotel cho UserAgent. Còn phương thức searchTrain(Location) dùng để tìm ra TrainAgent có thể cung cấp dịch vụ tàu hoả cho UserAgent.



Hình 7.15: Quan hệ giữa các phương thức của lớp MatchAgent

c. Các biến và hàm của lớp MatchAgent

Lớp MatchAgent có các phương thức chính như sau:

Void saveHotelAgentAddress (Address hotelAddress)

Lưu địa chỉ của một HotelAgent vào cơ sở dữ liệu. Tham số đầu vào là địa chỉ của một HotelAgent, bao gồm tên agent, địa chỉ hostname mà agent đó hoạt động và số cổng port để kết nối với agent đó.

void saveTrainAgentAddress (Address trainAddress)

Dùng để lưu địa chỉ của các TrainAgent.

Address searchHotelAgent (String location)

Dùng để tìm địa chỉ của một HotelAgent quản lý các khách sạn tại khu vực được chỉ ra bởi tham số đầu vào location. Phương thức này trả lại một giá trị có kiểu địa chỉ của agent.

Address searchTrainAgent (String location)

Dùng để tìm ra địa chỉ của các TrainAgent.

d. Cơ sở dữ liệu

Cơ sở dữ liệu của MatchAgent chỉ bao gồm hai bảng có cấu trúc tương tự nhau: bảng HotelAddress lưu trữ địa chỉ các HotelAgent và bảng TrainAddress lưu trữ địa chỉ các TrainAgent. Mỗi bảng đều có cấu trúc 3 trường thông tin như sau:

- AgentName*: Tên của agent.
- HostName*: Tên máy mà agent đó hoạt động.
- Port*: Số cổng mà agent đó làm việc với hệ thống

7.4 Kết luận

Chương này đã trình bày các bước cài đặt, tích hợp và triển khai hệ dịch vụ du lịch **TraNeS**. Hệ đã được thử nghiệm trên hai môi trường khác nhau: môi trường tập trung và môi trường phân tán.

CHƯƠNG 8

GIỚI THIỆU HỆ TRAMES

- Đặc trưng của hệ TraNeS
- Các nhóm chức năng của Hệ TraNeS
- Sử dụng Hệ TraNeS

Quá trình phân tích, thiết kế và cài đặt hệ dịch vụ du lịch **TraNeS** đã được trình bày trong các chương 5, 6, 7. Mục đích của chương 8 nhằm giới thiệu các đặc trưng và các nhóm chức năng của hệ thống. Chương này cũng sẽ trình bày khái quát về giao diện của hệ thống và quá trình cài đặt, sử dụng hệ thống này.

8.1 Đặc trưng của Hệ TraNeS

Trong các chương trước, tài liệu đã trình bày quá trình phân tích, thiết kế hệ dịch vụ du lịch **TraNeS** theo phương pháp luận MaSE sử dụng bộ công cụ AgentTool. Chương này sẽ tập trung vào việc giới thiệu các nhóm chức năng của hệ **TraNeS**, cách cài đặt và triển khai hệ thống. Trước hết, tài liệu sẽ tổng kết lại những đặc trưng chính của Hệ **TraNeS**. Các đặc trưng này bao gồm:

- Mô hình hóa người sử dụng dựa trên logic mờ. Trong hệ dịch vụ **TraNeS**, yêu cầu của khách hàng được mô hình theo các thuộc tính của loại mặt hàng cần mua. Mỗi thuộc tính được đặc trưng bởi giá trị lớn nhất, giá trị nhỏ nhất và *độ quan trọng* tương ứng. Độ quan trọng này được biểu diễn dựa trên các biến mờ ngôn ngữ (như *Tuyệt đối quan trọng*, *rất quan trọng*, *không quan trọng lắm*, ...). (Chi tiết xem trong chương 5).
- **TraNeS** sử dụng mô hình thương lượng song phương dựa trên ràng buộc mờ. Mô hình này đã được trình bày chi tiết trong chương 2 của tài liệu. Hệ thống sẽ bao gồm nhiều HotelAgent đại diện cho các khách sạn, nhiều TrainAgent đại diện cho các nhà ga tàu hỏa. Khi khách hàng gửi yêu cầu, hệ thống sẽ sinh ra một UserAgent để tiến hành thương lượng với các HotelAgent, Train Agent nhằm tìm ra một khách sạn và một chuyến tàu phù hợp với khách hàng. Quá trình thương lượng ở phía UserAgent sẽ tuân theo chiến lược thương lượng cho agent mua, ngược lại, quá trình thương lượng của các HotelAgent và TrainAgent sẽ tuân theo chiến lược thương lượng cho agent bán. (Xem chi tiết trong phần 2.3).
- Đặc trưng thứ ba là về công cụ phát triển hệ thống: Hệ **TraNeS** được phân tích thiết kế sử dụng bộ công cụ AgentTool, ngôn ngữ phát triển hệ thống được lựa chọn là ngôn ngữ Java và JSP trên nền ứng dụng Web. Cơ sở dữ liệu của hệ thống được xây dựng sử dụng SQL Server.

8.2 Các mô hình hoạt động của hệ TraNeS

Hệ **TraNeS** đã được cài đặt để có thể hoạt động theo hai mô hình *mô hình tập trung* và *mô hình phân tán*.

a. Mô hình tập trung

Mô hình tập trung chính là mô hình client-server. Trong mô hình này, tất cả các agent đều chạy trên một máy server. Hệ thống sẽ được cài đặt và đưa lên một máy chủ duy nhất. Các khách hàng khi gửi yêu cầu đến hệ thống từ các Client đều sinh ra một UserAgent đại diện cho khách hàng đó. Agent này cũng chạy trên server và trả kết quả về cho khách hàng ở client. Tuy nhiên mô hình này có những nhược điểm sau:

- Mô hình tập trung này không phản ánh ưu điểm phân tán của hệ đa agent.
- Đòi hỏi tài nguyên hệ thống cao: yêu cầu về bộ nhớ, tốc độ xử lí...do các agent cùng chạy trên một nền phần cứng, cùng phải chia sẻ tài nguyên trên một máy.
- Không thể hiện bản chất của một hệ xử lí song song khi agent phải thương lượng nhiều mặt hàng cùng lúc.

b. Mô hình phân tán

Mô hình này có nhiều server, mỗi server đại diện cho một khu vực địa lý (chẳng hạn như một thành phố) để lưu trữ liệu liên quan đến các nhà ga và các khách sạn ở khu vực đó. Trong mô hình này, HotelAgent và TrainAgent sẽ hoạt động ở các server khác nhau còn UserAgent và MathAgent hoạt động trên một máy chủ xác định. Mô hình này khắc phục được các nhược điểm như đã nêu trên của mô hình tập trung.

Trong cả hai mô hình, tại mỗi server, hệ thống luôn có các agent chạy nền và luôn hoạt động để lắng nghe các yêu cầu gửi đến nó. Khách hàng hoàn toàn không biết sự tồn tại của các agent này. Trong mô hình tập trung, cả MatchAgent, TrainAgent và HotelAgent đều hoạt động trên cùng một server. Trong mô hình phân tán, các HotelAgent và TrainAgent khác nhau hoạt động trên các server khác nhau. Riêng UserAgent sẽ được khởi động khi khách hàng gửi yêu cầu đến hệ thống một cách tự động và hoàn toàn chạy ngầm cho đến khi hoàn thành nhiệm vụ.

Người sử dụng phải kích hoạt các agent chạy nền trên các server. Sau khi được kích hoạt, các agent sẽ hoạt động và khi có một sự kiện xảy ra thì agent sẽ ghi lại sự kiện đó trên giao diện của agent đó.

Hình 8.1 mô tả các sự kiện ghi lại trong các agent sau khi được khởi động trên server. Agent trung gian (MiddleAgent) sẽ lắng nghe các kết nối trên cổng 2000. Khi các HotelAgent, TrainAgent khởi động thì nó sẽ kết nối tới MiddleAgent. Trên hình 8.1, có 6 HotelAgent và 6 TrainAgent khởi động nên MiddleAgent nhận được 12 kết nối.

The screenshot displays three separate command-line windows (cmd.exe) running on a Windows system. Each window shows log output from different agents starting their message handlers.

- Top Window:** Shows "Agent Middler Agent messageHandler started on port 2000" followed by a series of "Connection X received from cntt" messages for X from 1 to 12.
- Middle Window:** Shows "Agent IP HCM messageHandler started on port 3001" and other agents starting on ports 3002 through 3006. A callout bubble points to this window with the text: "Agent trung gian - lắng nghe các kết nối ở cổng 2000".
- Bottom Window:** Shows "Agent IP HCM messageHandler started on port 6001" and other agents starting on ports 6002 through 6006. A callout bubble points to this window with the text: "Các HotelAgent khởi động." and "Các TrainAgent khởi động."

Hình 8.1: Các Agent khởi động trên Server

Khi khách hàng gửi yêu cầu đến hệ thống, hệ thống sẽ sinh ra một UserAgent chạy ngầm trên server. Agent này trước hết liên lạc với Agent trung gian để tìm ra *đối tác* của mình theo thuật toán thương lượng song phương. Giao diện của agent trung gian khi đó sẽ có thêm một kết nối thứ 13 (Hình 8.2).

This screenshot shows a single command-line window (cmd.exe) displaying the log output of the "Agent Middler". It includes the initial startup message for port 2000, followed by 12 connection logs from "cntt" and finally a log for a 13th connection.

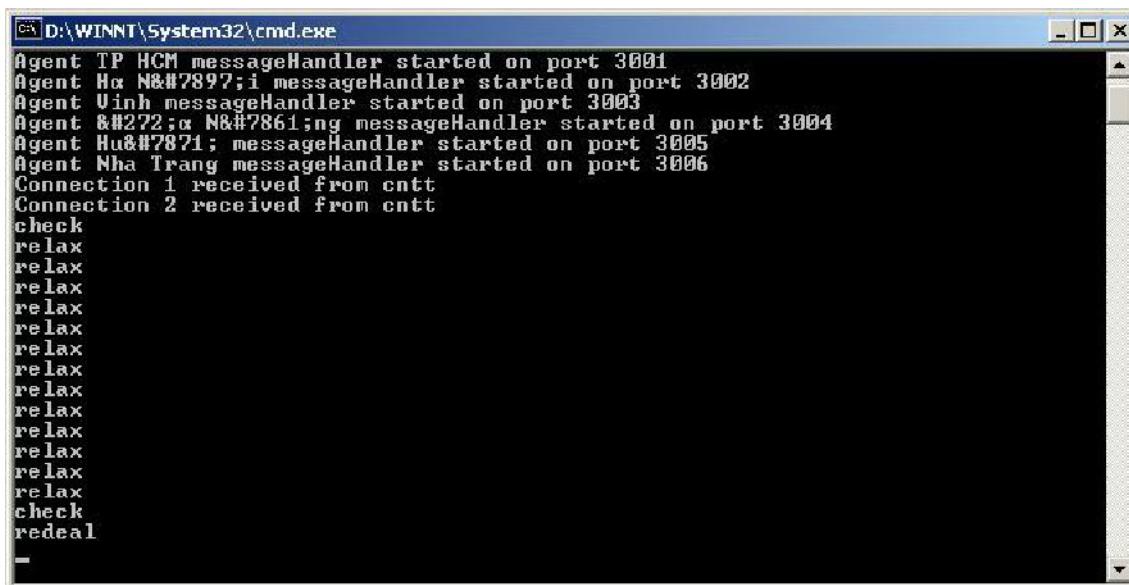
```

Agent Middler Agent messageHandler started on port 2000
Connection 1 received from cntt
Connection 2 received from cntt
Connection 3 received from cntt
Connection 4 received from cntt
Connection 5 received from cntt
Connection 6 received from cntt
Connection 7 received from cntt
Connection 8 received from cntt
Connection 9 received from cntt
Connection 10 received from cntt
Connection 11 received from cntt
Connection 12 received from cntt
Connection 13 received from cntt

```

Hình 8.2: Agent trung gian khi UserAgent khởi động

Tiếp theo, quá trình thương lượng giữa các agent sẽ được tiến hành, các trạng thái thay đổi trong quá trình thương lượng đó đều được lưu vết lại trên HotelAgent và TrainAgent (Hình 8.3 và Hình 8.4).

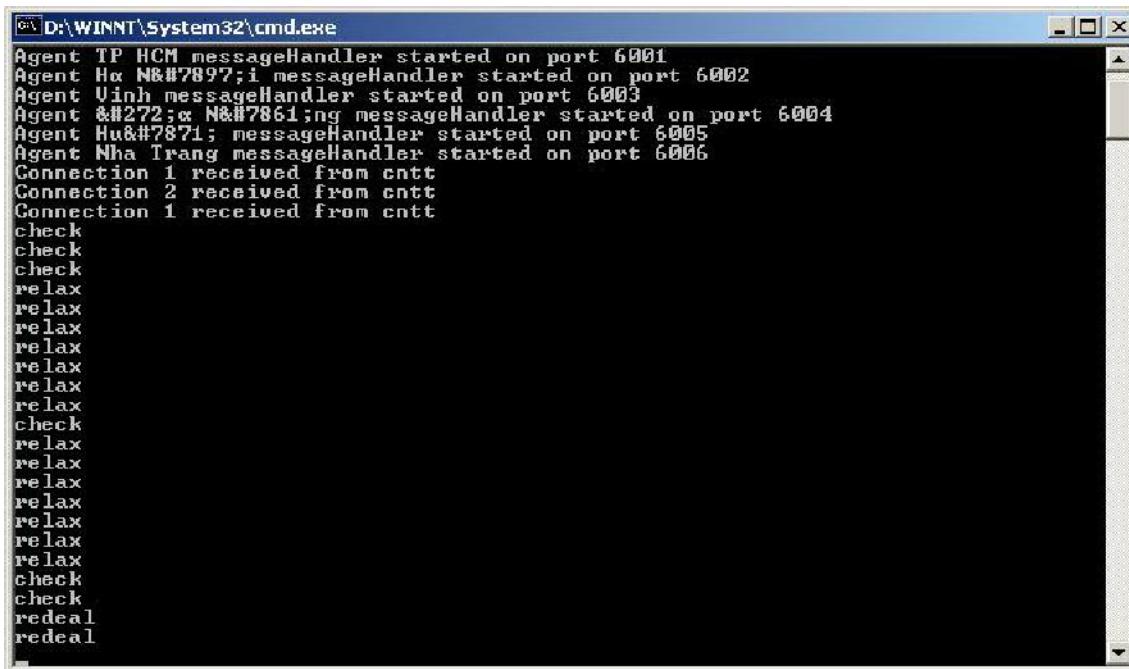


```
D:\WINNT\System32\cmd.exe
Agent TP HCM messageHandler started on port 3001
Agent Ho N&#7897;i messageHandler started on port 3002
Agent Vinh messageHandler started on port 3003
Agent &#272;ox N&#7861;ng messageHandler started on port 3004
Agent Hu&#7871; messageHandler started on port 3005
Agent Nha Trang messageHandler started on port 3006
Connection 1 received from cttt
Connection 2 received from cttt
check
relax
check
redeal
-
```

Hình 8.3: HotelAgent khi thương lượng với User Agent

Trong Hình 8.3 ta thấy quá trình nhượng bộ của HotelAgent cũng diễn ra qua rất nhiều bước (tương ứng với các trạng thái *relax*). Khi hai bên đạt được thỏa thuận thì HotelAgent sẽ chuyển sang trạng thái *redeal*.

TrainAgent sẽ lưu lại vết của cả hai quá trình thương lượng để xác định chuyến tàu đi và chuyến tàu về (Hình 8.4). Hai quá trình này diễn ra gần như đồng thời và kết quả cũng đến gần như cùng một lúc.



```
D:\WINNT\System32\cmd.exe
Agent TP HCM messageHandler started on port 6001
Agent Ho N&#7897;i messageHandler started on port 6002
Agent Vinh messageHandler started on port 6003
Agent &#272;ox N&#7861;ng messageHandler started on port 6004
Agent Hu&#7871; messageHandler started on port 6005
Agent Nha Trang messageHandler started on port 6006
Connection 1 received from cttt
Connection 2 received from cttt
Connection 1 received from cttt
check
check
check
relax
relax
relax
relax
relax
relax
relax
relax
check
relax
relax
relax
relax
relax
relax
relax
check
redeal
-
```

Hình 8.4: TrainAgent khi thương lượng với UserAgent

8.3 Các nhóm chức năng của Hệ TraNeS

Phần này sẽ giới thiệu các nhóm chức năng của hệ **TraNeS**. Trong mỗi nhóm chức năng, tài liệu sẽ giới thiệu sơ lược nội dung và giao diện của từng trang của nhóm chức năng đó. Trong hệ **TraNeS** có thể được chia làm bốn nhóm chức năng như sau:

- **Nhóm các chức năng chung**

Nhóm này bao gồm các chức năng cho mọi đối tượng sử dụng hệ thống. Đây chính là các chức năng khi người sử dụng truy nhập tới hệ thống mà chưa cần đăng nhập. Các chức năng này bao gồm:

Đăng nhập: giới thiệu những thông tin chung về hệ thống. Giao diện của Trang chủ được biểu diễn như trong Hình 8.5.



Hình 8.5: Trang chủ của hệ thống

Tìm kiếm: Giúp người dùng tìm kiếm các khách sạn hoặc chuyến tàu theo các tiêu chí đơn giản như: tên, địa chỉ và giá tiền thuê phòng (hoặc giá vé) giới hạn. Các trang này nhằm mục đích giúp người dùng tham khảo những thông tin khái quát trước khi quyết định đăng ký để trở thành *khách hàng* trong hệ dịch vụ du lịch **TraNeS**. Có ba trang tìm kiếm bao gồm:

- Tìm kiếm khách sạn
- Tìm kiếm các chuyến tàu
- Tìm kiếm nhà ga tàu hỏa

Giao diện của hai trang tìm kiếm chuyến tàu và tìm kiếm khách sạn có dạng như trong Hình 8.6 và Hình 8.7 dưới đây.



Hình 8.6: Trang Tìm kiếm chuyến tàu



Hình 8.7: Trang tìm kiếm khách sạn

Trang đăng ký: Giúp cho người dùng có thể đăng ký sử dụng hệ thống với vai trò là *khách hàng*, là *người quản lý nhà ga* hay *người quản lý khách sạn*:

- Khách hàng của hệ thống chỉ cần đăng ký tên đăng nhập của mình và một số thông tin cá nhân khác
- Những người quản lý khách sạn và các nhà ga phải đăng ký đầy đủ về thông tin và các dịch vụ mà khách sạn và nhà ga của mình có thể cung cấp.

Khi chọn Đăng ký, trang Hướng dẫn đăng ký sẽ chỉ dẫn cho khách hàng các bước để đăng ký với vai trò là khách hàng, người quản lý khách sạn hay người quản lý các nhà ga. Trang này có giao diện như trong hình 8.8.



Hình 8.8: Trang hướng dẫn đăng ký

Trang Khách hàng đăng ký có giao diện như trong hình 8.9 dưới đây. Các trang Khách sạn đăng ký và Nhà ga đăng ký chỉ khác ở một điểm là có thêm các trường thông tin để người quản lý khách sạn (và nhà ga) nhập thông tin về dịch vụ mà khách sạn (hay nhà ga) của mình có thể đáp ứng.



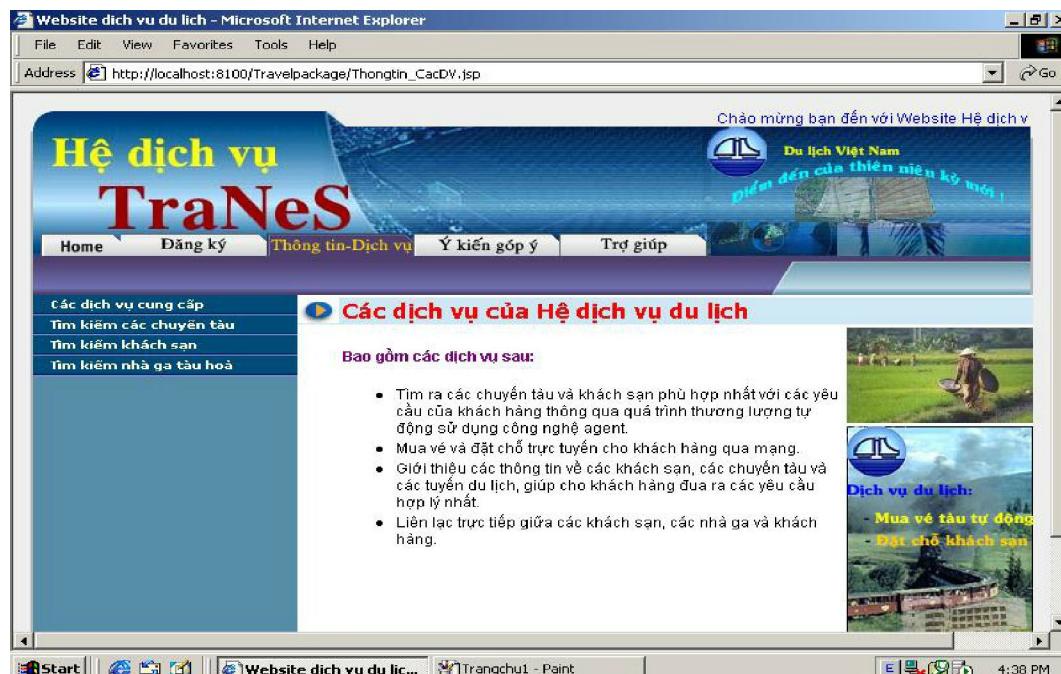
Hình 8.9: Trang Khách hàng đăng ký

Trợ giúp người sử dụng: Hướng dẫn người dùng hệ thống (bao gồm khách hàng, người quản lý nhà ga, người quản lý khách sạn) cách cài đặt và truy nhập hệ thống, cách thức sử dụng các dịch vụ của hệ thống. Hình 8.10 dưới đây là giao diện của trang Trợ giúp cho khách hàng.



Hình 8.10: Trang trợ giúp khách hàng

Hỗ trợ khác: Cung cấp một số thông tin khác liên quan đến lĩnh vực dịch vụ du lịch và liên quan trực tiếp đến hệ **TraNeS**. Hình 8.11 dưới đây là giao diện trang thông tin Các dịch vụ của hệ **TraNeS**.



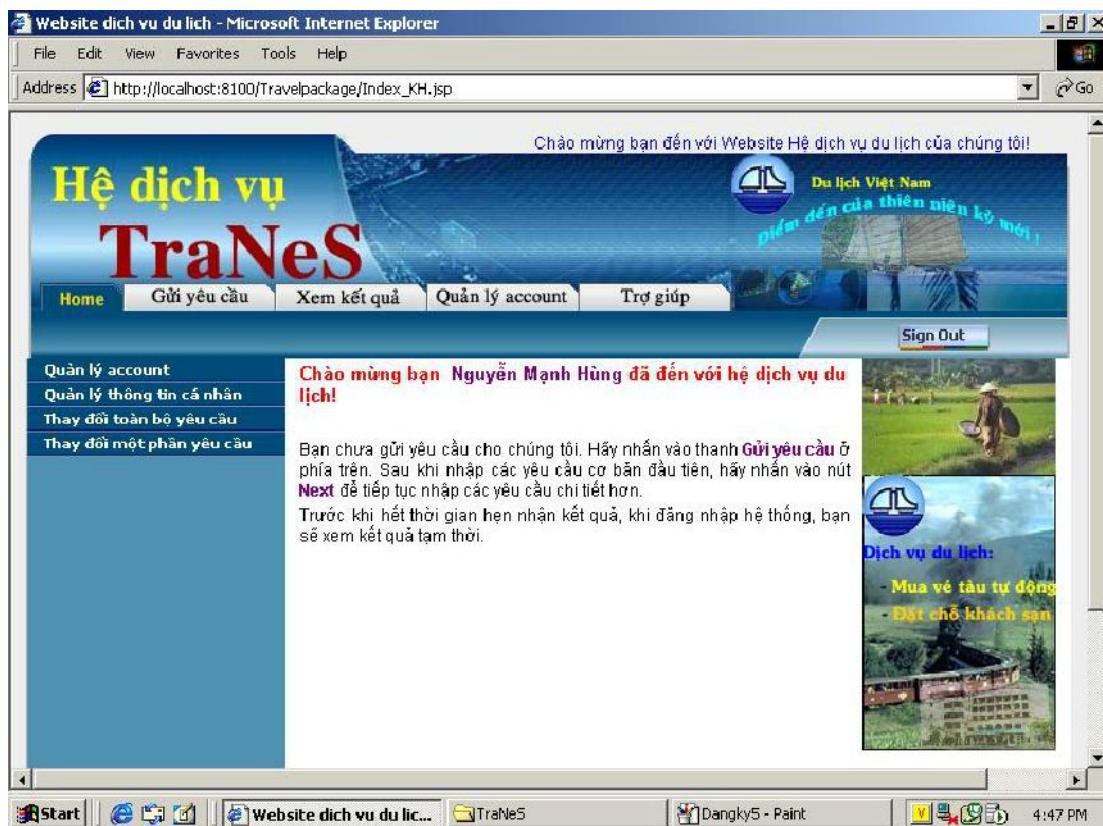
Hình 8.11: Trang thông tin Các dịch vụ của **TraNeS**

- Nhóm chức năng giao tiếp với khách hàng**

Truy nhập cho khách hàng: Đây là trang đầu tiên khi khách hàng thực hiện truy nhập vào hệ thống. Nội dung trang này sẽ khác nhau trong hai trường hợp:

- Nếu khách hàng truy nhập lần đầu, trang này sẽ gợi ý cho khách hàng các bước cần thực hiện để gửi yêu cầu cho chuyến đi của mình.
- Nếu khách hàng đã truy nhập vào hệ thống sau khi đã gửi yêu cầu thì trang này sẽ gợi ý cho khách hàng có thể thay đổi yêu cầu hoặc xem kết quả tìm kiếm tạm thời về khách sạn và chuyến tàu cho chuyến đi của mình.

Hình 8.12 dưới đây là nội dung trang chủ khách hàng khi khách hàng truy nhập lần đầu còn Hình 8.13 là khi khách hàng truy nhập lại sau khi đã gửi yêu cầu trong lần trước đó.



Hình 8.12: Trang chủ khách hàng khi chưa gửi yêu cầu



Hình 8.13: Trang chủ khách hàng khi đã gửi yêu cầu

Phản hồi yêu cầu: Giúp khách hàng gửi yêu cầu đến hệ thống. Các yêu cầu của khách hàng gửi đi có dạng khoảng và có độ ưu tiên tương ứng. Khách hàng sẽ gửi yêu cầu trong ba trang:

- Trang thứ nhất là các yêu cầu chung nhất như ngày đi và ngày về, các sở thích trong chuyến đi và khả năng chấp nhận tổng thể của khách hàng.
- Trang thứ hai là các yêu cầu của khách hàng về khách sạn muốn đặt chỗ với các tiêu chí như hạng khách sạn, khoảng cách đến trung tâm, internet trong phòng và mức độ ưu tiên cho từng tiêu chí.
- Trang thứ ba là các yêu cầu của khách hàng về chuyến tàu bao gồm: loại chỗ, loại tàu và mức độ ưu tiên của các tiêu chí này. Cuối trang thứ tư này sẽ là yêu cầu về mức giá tổng thể của khách hàng. Mức giá này sẽ được tính toán tương đối dựa trên các yêu cầu trước đó của khách hàng và khách hàng sẽ lựa chọn trong một khoảng giá trị có thể chấp nhận được.

Hình 8.10 là trang nhập yêu cầu tổng thể. Trong trang này, ngoài các yêu cầu về địa điểm, thời gian đi và về, khách hàng còn cần đưa vào hệ thống các thông tin yêu cầu về sở thích các dịch vụ trên tàu và trong khách sạn như Điện thoại miễn phí, ăn sáng miễn phí, tặng quà may mắn, ... Giá trị của các thuộc tính này được biểu diễn theo biến mờ ngôn ngữ với các giá trị như *rất thích*, *khá thích*, *không*

thích lắm... Ví dụ trong hình 8.14, khách hàng này rất thích được miễn phí điện thoại nội hat, khá thích chế độ ăn sáng miễn phí và chỉ quan tâm ở mức vừa phải về chế độ tặng quà may mắn.

Ngoài ra, khách hàng còn phải nhập yêu cầu về ngưỡng chấp nhận. Có ba cách lựa chọn cho ngưỡng chấp nhận là *cao*, *vừa phải* và *thấp*. Chi tiết về ngưỡng chấp nhận đã được trình bày trong chương 5.

The screenshot shows a Microsoft Internet Explorer window displaying a website for tourism services. The main header reads "Hệ dịch vụ Du lịch". Below it, there's a banner for "Du lịch Việt Nam" with the tagline "Đến cửa thiên niên kỷ mới". The menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The top navigation bar has links for "Home", "Gửi yêu cầu" (selected), "Xem kết quả", "Quản lý Acount", "Trợ giúp", and "Sign Out". On the left, a sidebar lists options like "Thay đổi yêu cầu", "Xem kết quả tạm thời", and "Giới thiệu các loại vé tàu các tuyến du lịch". The main content area is titled "Khách hàng Gửi yêu cầu". It contains fields for "Địa điểm" (Hà Nội to Đà Nẵng) and "Thời gian" (Ngày 2 Tháng 11 to Ngày 10 Tháng 11). Under "Sở thích", there's a table with three rows: "Rất thích" (radio buttons for free phone and breakfast), "Khá thích" (radio buttons for both), and "Vừa phải" (radio button for breakfast only). Below this is a section for "Khả năng chấp nhận" with three buttons: "Thấp" (radio button), "Vừa phải" (radio button, selected), and "Cao" (radio button).

Hình 8.14: Yêu cầu tổng thể

Hình 8.15 là các yêu cầu chi tiết của khách hàng về khách sạn, bao gồm: hạng khách sạn, khoảng cách từ khách sạn đến trung tâm, internet trong phòng và mức độ ưu tiên của khách hàng cho mỗi tiêu chí đó. Ngoại trừ thuộc tính internet trong phòng được biểu diễn theo giá trị boolean (có hay không), còn lại các tiêu chí khác đều được biểu diễn dưới dạng khoảng.

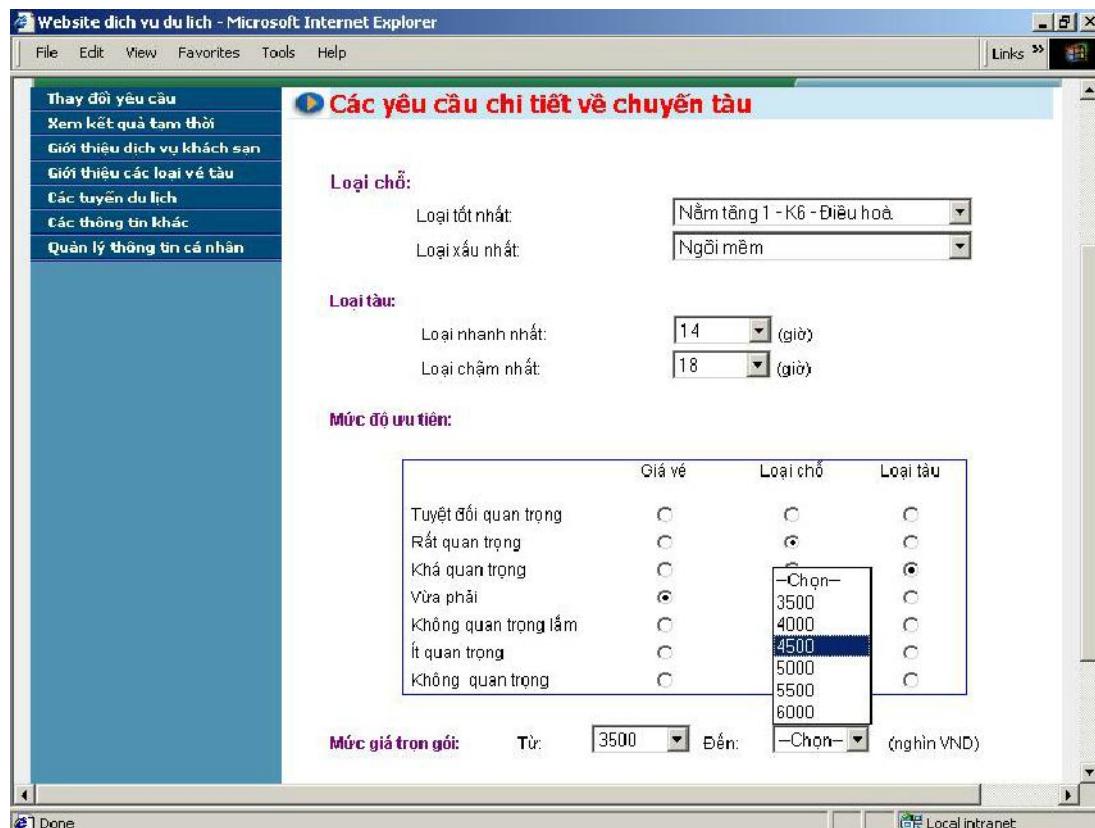
Mức độ ưu tiên cho mỗi tiêu chí được biểu diễn theo các giá trị mờ ngôn ngữ, thể hiện sự quan tâm của khách hàng đối với tiêu chí đó. Ví dụ, trên hình 8.15, khách hàng này

cho rằng có internet trong phòng đối với anh ta là rất quan trọng, còn giá phòng là không quan trọng lắm.

	Giá phòng	Hạng (số sao)	KH	Internet
Tuyệt đối quan trọng	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Rất quan trọng	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Khá quan trọng	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Vừa phải	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Không quan trọng lắm	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Ít quan trọng	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Không quan trọng	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>

Hình 8.15: Các yêu cầu chi tiết về khách sạn

Hình 8.16 là giao diện trang nhập yêu cầu chi tiết về các chuyến tàu. Các yêu cầu này bao gồm loại chỗ trên tàu (chính là loại vé), loại tàu (tính theo tốc độ) và mức độ ưu tiên của khách hàng đối với thuộc tính đó.



Hình 8.16: Yêu cầu chi tiết về các chuyến tàu

Thay đổi yêu cầu: Trang này giúp khách hàng thay đổi yêu cầu của mình khi chưa đồng ý với kết quả tìm kiếm hoặc khi khách hàng thấy cần phải thay đổi cho phù hợp. Khách hàng có thể lựa chọn thay đổi toàn bộ yêu cầu (giống như gửi yêu cầu lại từ đầu) hoặc chỉ thay đổi lại một số thông tin.

Khi khách hàng chọn thay đổi một số thông tin trong yêu cầu, khách hàng chỉ nhấn vào nút thay đổi ứng với thông tin cần thay đổi. Khi đã thay đổi xong thì khách hàng phải nhấn nút thay đổi xong để kích hoạt User Agent tiến hành thương lượng lại với các Hotel Agent và Train Agent. Hình 8.17 là giao diện chung của trang thay đổi một phần yêu cầu.

Website dịch vụ du lịch - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost:8100/Travelpackage/ThaydoiYeucau_KH.jsp

Chào mừng bạn đến với Website Hệ dịch vụ du lịch của chúng tôi!

Hệ dịch vụ TraNeS

Home Gửi yêu cầu Xem kết quả Quản lý account Trợ giúp Sign Out

Khách hàng thay đổi yêu cầu

Địa điểm:

Nơi đi	Hà Nội	Thay đổi
Nơi đến	Đà Nẵng	

Thời gian:

Ngày đi	12/8/2004	Thay đổi
Ngày về	17/8/2004	

Sở thích:

Điện thoại nội hat miễn phí	Khá thích	Thay đổi
Ánh sáng miễn phí	Rất thích	
Tặng quà may mắn	Rất thích	

Khả năng chấp nhận: **Vừa phải** **Thay đổi**

Các yêu cầu về khách sạn:

Tiêu chí	Giá trị	Mức ưu tiên	
Hạng khách sạn	Từ 3 (sao) đến 4 (sao)	Không quan trọng lắm	Thay đổi
K/c đến trung tâm	Từ 1 (km) đến 3 (km)	Khá quan trọng	Thay đổi
Internet	Cần thiết	Rất quan trọng	Thay đổi

Các yêu cầu về chuyến tàu:

Tiêu chí	Giá trị	Mức ưu tiên	
Loại chỗ	Từ Nằm tầng 3 - K6 - Điều hòa đến Nằm tầng 1 - K4 - Điều hòa	Khá quan trọng	Thay đổi
Loại tàu	Từ 14 đến 18	Vừa phải	Thay đổi

Mức giá trọn gói:

Cao nhất	2500 000 VND	Thay đổi
Thấp nhất	1500 000 VND	

Thay đổi xong

Index || Gửi yêu cầu || Kết quả || Hướng dẫn || Copyright by Mạnh Hùng - Mạnh Sơn

Mailto: nmson2881@yahoo.com or nmhufng@yahoo.com

Local intranet

Hình 8.17: Giao diện thay đổi yêu cầu

Xem kết quả tạm thời: Cung cấp cho khách hàng kết quả tìm kiếm tạm thời trước khi hết thời hạn mà khách hàng đặt ra cho hệ thống. Nếu khách hàng đồng ý với kết quả tìm kiếm này thì khách hàng có thể dùng quá trình thương lượng trước thời hạn.

Giao diện của trang xem kết quả tạm thời như trong Hình 8.18. Nếu đồng ý với kết quả này thì khách hàng nhấn vào chữ Đồng ý để chuyển sang trang đặt chỗ.

Chào mừng bạn đến với Website Hệ dịch vụ du lịch của chúng tôi!

Du lịch Việt Nam
Điểm đến của thiên niên kỷ mới

Kết quả

A. Thông tin về các chuyến tàu đi và về

Điểm xuất phát: Hà Nội
Điểm đến: Đà Nẵng

Chi tiết

Tiêu chí	Yêu cầu	Chuyến tàu đi	Chuyến tàu về
Tên chuyến tàu	-	E1	E2
Ngày giờ đi	-	1 - 11 - 2003 (23 h 00)	10 - 11 - 2003 (14 h 00)
Ngày giờ đến	-	2 - 11 - 2003 (12 h 47)	11 - 11 - 2003 (5 h 00)
Loại vé	Nâng tầng 1 - K6 - Điều hòa	Nâng tầng 1 - K6 - Điều hòa	Nâng tầng 1 - K6 - Điều hòa
Tốc độ tàu	14 - 18 h	13 h 47	14 h 59
Giá vé	-	385 000 VND	385 000 VND

B. Thông tin về khách sạn

Tên Khách sạn: Việt Bắc
Địa chỉ: 34 - Nguyễn Chí Thanh - Đà Nẵng
Nghỉ từ ngày 2 - 11 - 2003 đến ngày 10 - 11 - 2003

Chi tiết

Tiêu chí	Yêu cầu	Khách sạn
Hạng KS	3 - 4 Sao	4
K/c đến trung tâm	1 - 4 Km	1
Internet	1	1
Giá phòng/ngày	-	325 000 VND

C. Thông tin về giá

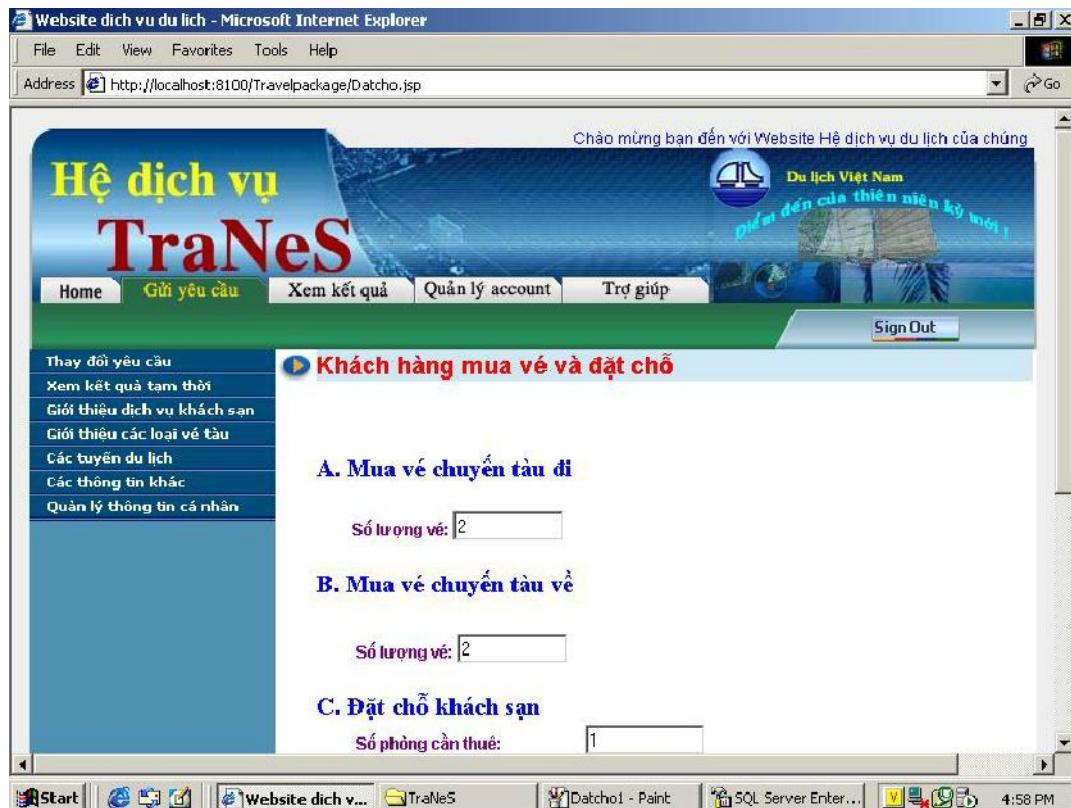
Giá vé tàu đi + về	770 000 VND
Giá thuê phòng KS	2925 000 VND
Tổng số tiền phải trả	3695 000 VND
Số tiền yêu cầu	3500 000 - 4500 000 VND

Nếu các bạn đồng ý với kết quả tìm kiếm trên và quyết định mua vé tàu, đặt chỗ khách sạn, hãy nhấn vào **Đồng ý**

Đồng ý

Hình 8.18: Giao diện kết quả

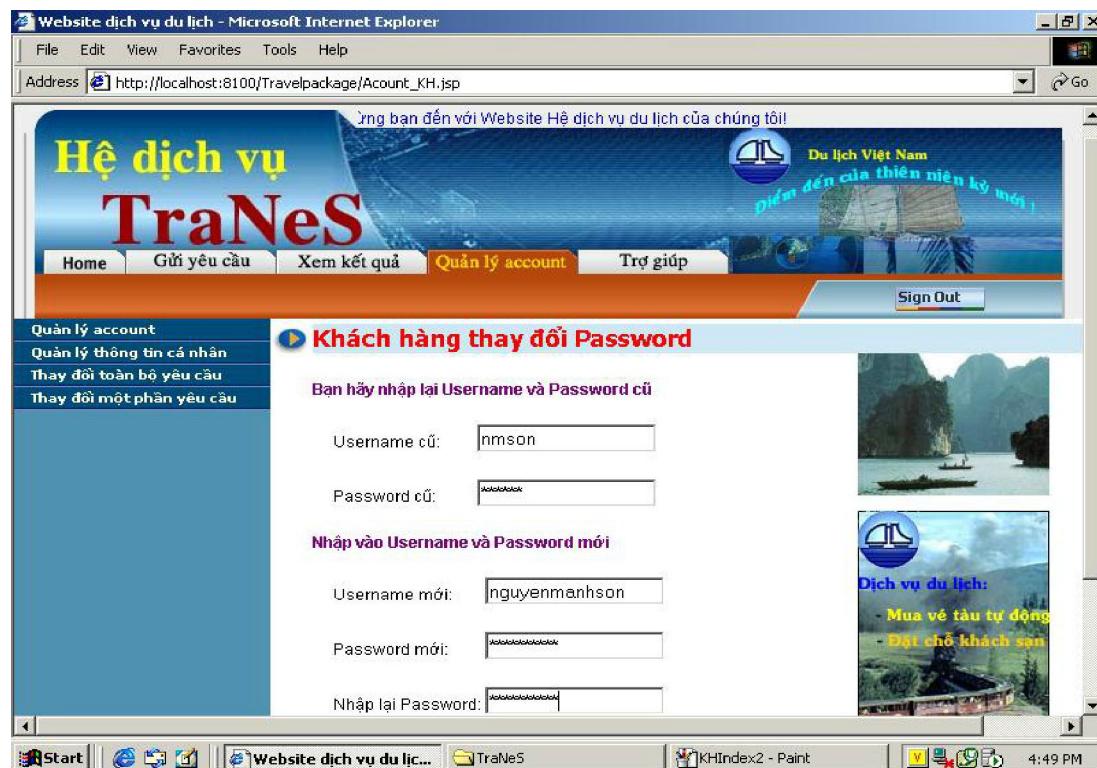
Đặt chỗ: Giúp khách hàng đăng ký đặt chỗ sau khi đồng ý với kết quả thương lượng. Các thông tin này sẽ được gửi đến các nhà ga và các khách sạn tương ứng. Giao diện trang đặt chỗ có dạng như trong hình 8.19. Khách hàng phải nhập vào các thông tin về số lượng vé tàu cần mua, số phòng cần đặt, số người trong mỗi phòng khách sạn, ...



Hình 8.19: Trang mua vé và đặt chỗ

Quản lý tài khoản: Giúp khách hàng quản lý và thay đổi các thông tin liên quan đến tài khoản cá nhân như tên đăng nhập (username) và mật khẩu (password). Giao diện của trang này được biểu diễn như trong hình 8.20.

Cung cấp thông tin: Cung cấp các thông tin liên quan đến dịch vụ du lịch, đến các nhà ga và các khách sạn, ... Đây chỉ là các thông tin giúp khách hàng tham khảo trong quá trình sử dụng hệ thống.



Hình 8.20: Trang khách hàng thay đổi thông tin tài khoản

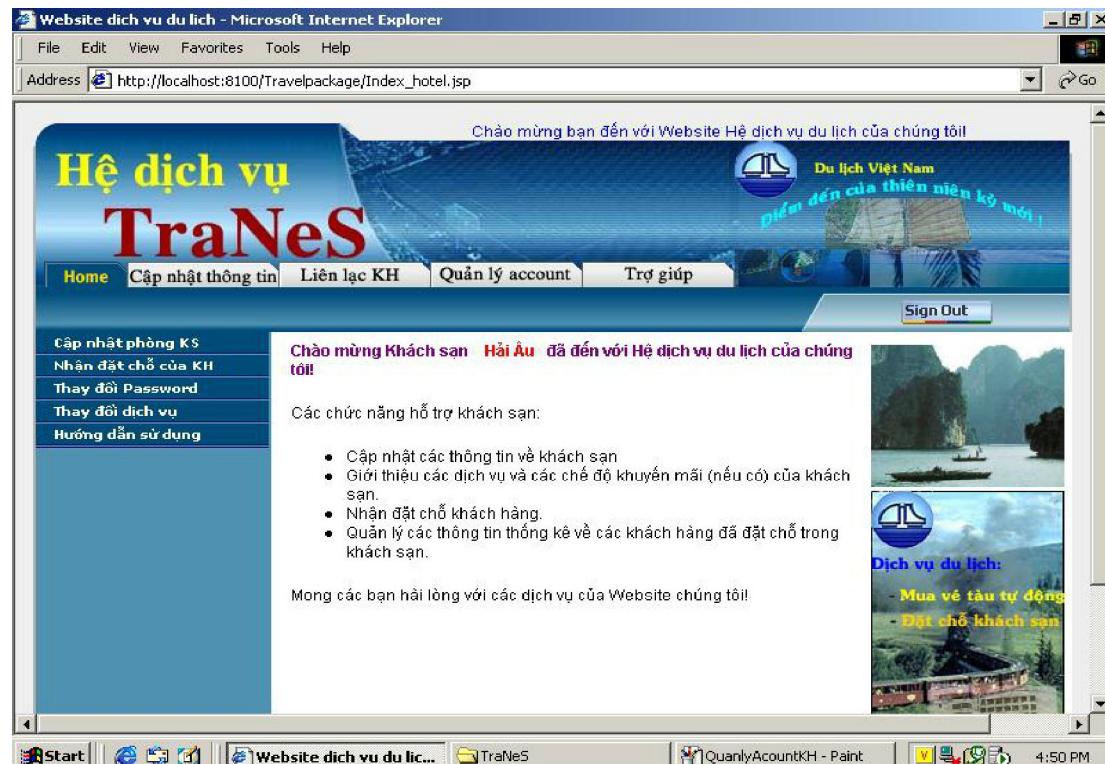
- **Nhóm chức năng giao tiếp với các khách sạn**

Các trang này hỗ trợ cho những người quản lý khách sạn có thể cập nhật thông tin về khách sạn của mình, nhận đặt chỗ của khách hàng và quản lý account. Để vào trang dành cho khách sạn, người quản lý phải login dưới vai trò khách sạn. Nhóm chức năng này bao gồm các trang:

Các chức năng của dịch vụ Khách sạn: Giao diện trang này được biểu diễn như trong hình 8.21.

Quản lý account: Hoàn toàn tương tự như quản lý account cho khách hàng

Nhận đặt chỗ của khách hàng: Người quản lý khách sạn phải nhập vào UserID của các khách hàng đăng ký đặt chỗ ở khách sạn sau đó kiểm tra các thông tin liên quan đến khách hàng trước khi liên hệ trực tiếp với khách hàng để đưa ra thỏa thuận cuối cùng.

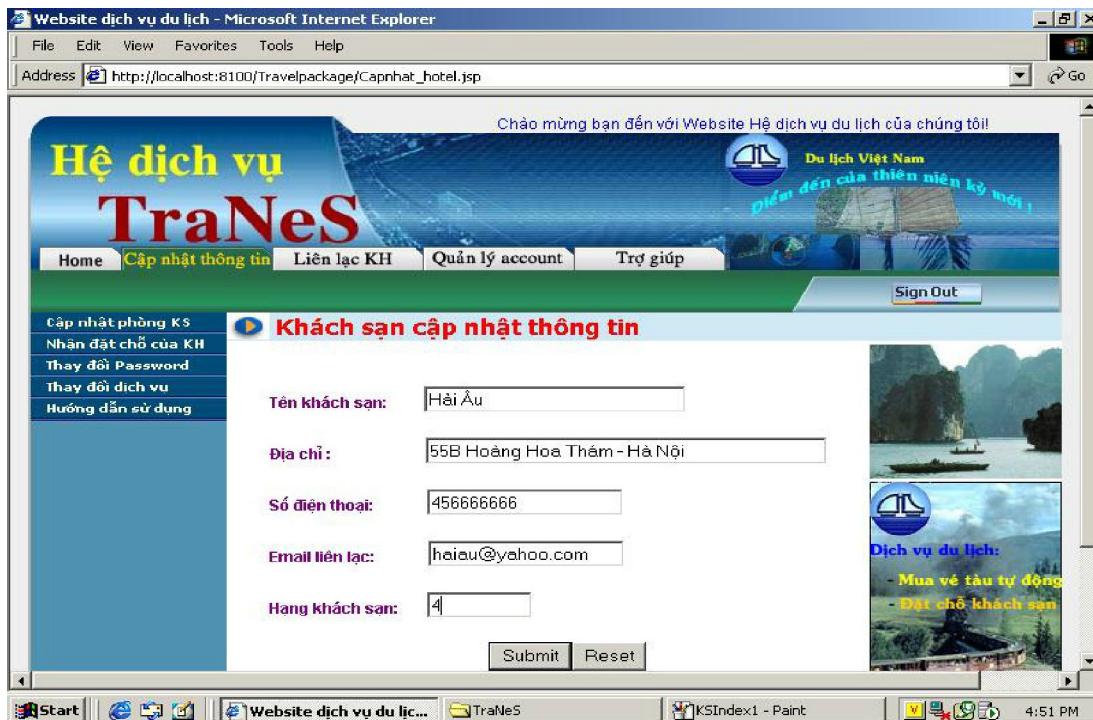


Hình 8.21: Trang chủ khách sạn



Hình 8.22: Khách sạn nhận đặt chỗ

Cập nhật thông tin và thay đổi dịch vụ khách sạn: Người quản lý khách sạn có thể thay đổi toàn bộ các thông tin liên quan đến khách sạn như các thông tin chung (tên, địa chỉ liên lạc, số điện thoại, ...) (như Hình 8.23) các thông tin về phòng khách sạn như số phòng còn trống, các dịch vụ trong phòng.



Hình 8.23: Khách sạn cập nhật thông tin

- Nhóm chức năng giao tiếp với các nhà ga tàu hỏa**

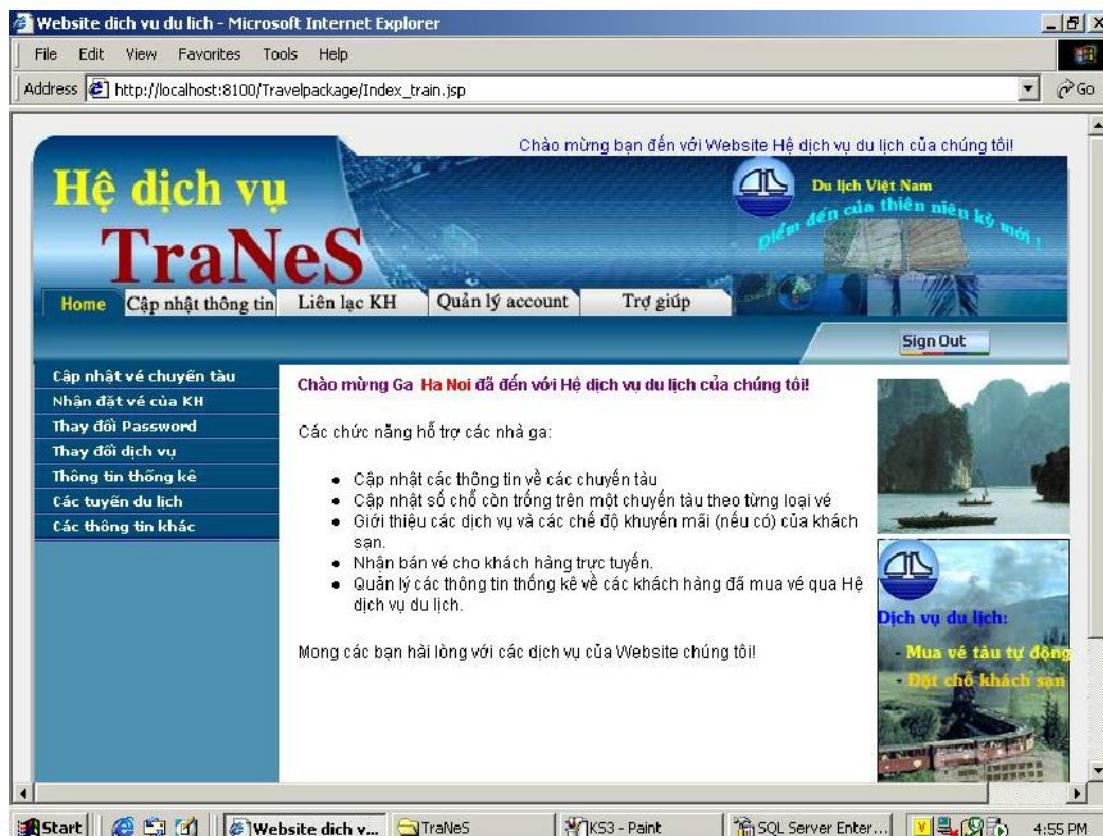
Tương tự như nhóm các trang giao tiếp với khách sạn, nhóm các trang giao tiếp với nhà ga cũng hỗ trợ cho các nhân viên quản lý nhà ga quản lý các thông tin của mình. Nhóm trang này bao gồm:

Hiển thị các dịch vụ của Nhà ga tàu hỏa

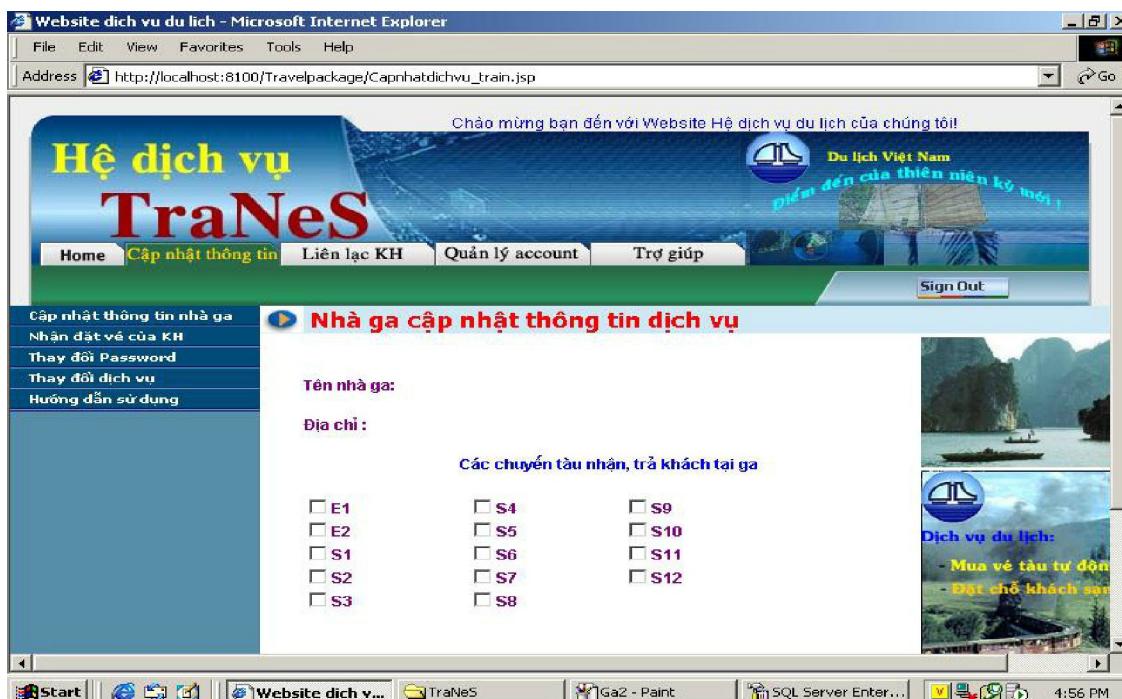
Quản lý account cho nhà ga: Hoàn toàn tương tự như trang quản lý account cho khách hàng và cho khách sạn.

Nhận mua vé: Cũng tương tự như trang nhận đặt chỗ của khách sạn.

Cập nhật thông tin về các chuyến tàu: Người quản lý các nhà ga cũng có thể cập nhật các thông tin liên quan đến các chuyến tàu của nhà ga mình như các chuyến tàu nhận, trả khách tại ga, các dịch vụ, ... hay số vé tàu còn lại. Giao diện trang Nhà ga cập nhật thông tin có dạng như trong Hình 8.25.



Hình 8.24: Trang chủ của nhà ga



Hình 8.25: Nhà ga cập nhật thông tin

8.4 Cài đặt Hệ TraNeS

Phần này trình bày ngắn gọn về cài đặt các phần mềm cần thiết cũng như thiết lập các thông số cho hệ thống.

Cài đặt các phần mềm cần thiết cho hệ thống

Hệ **TraNeS** là một hệ ứng dụng Java, JSP trên môi trường Web, cơ sở dữ liệu xây dựng trên SQL Server 2000. Vì vậy, để truy nhập và sử dụng hệ thống, người sử dụng cần cài đặt các phần mềm sau:

- Trên cả Client và Server cần có trình duyệt Internet Explorer phiên bản 5.0 trở lên.
- Trên Server (hoặc các server trong mô hình phân tán): cần cài đặt các phần mềm sau:
 - Java phiên bản 1.3 trở lên
 - SQL Server 2000 trở lên
 - Máy chủ JRun 3.0 trở lên.

Cài đặt và thiết lập các thông số cho hệ thống

Tại server, người sử dụng phải thực hiện các công việc sau:

- Upload toàn bộ hệ **TraNeS** lên máy chủ JRun theo một tên nào đó: (ví dụ TravelPackage).
- Đăng ký nguồn dữ liệu System DSN sử dụng công cụ Data Sources trong Administrative Tool của Windows.
 - Trong mô hình tập trung: tất cả System DSN đều phải được thiết lập tại server. Các DSN này bao gồm:

HotelNegotiation: cho file cơ sở dữ liệu của khách sạn.

TrainNegotiation: cho file cơ sở dữ liệu của nhà ga

UserNegotiation: cho file cơ sở dữ liệu của khách hàng.

Negotiation: cho file cơ sở dữ liệu biểu diễn tri thức của agent trung gian trong quá trình thương lượng.

Các DSN này cần được trỏ đến đúng các file dữ liệu của Hệ TraNeS kèm theo trong đĩa CD.

- Trong mô hình phân tán: có nhiều server khác nhau, mỗi server lưu trữ dữ liệu cho một khu vực địa lý (chẳng hạn một thành phố) thì tại mỗi server cần thiết lập các DSN cho cơ sở dữ liệu của khách sạn và nhà ga (**HotelNegotiation** và **TrainNegotiation**) tại khu vực đó. Người sử dụng

cần chọn một server làm server chính và cài đặt thêm các DSN **UserNegotiation** và **Negotiation** trong server này.

Sau khi thiết lập đầy đủ các thông số như trên, người quản lý hệ TraNeS phải kiểm tra và thử nghiệm hoạt động của hệ thống trên cả máy client và server.

8.5 Bài học từ phát triển hệ TraNeS

Phần này sẽ trình bày một số bài học kinh nghiệm của quá trình phát triển một hệ đa agent, được rút ra từ quá trình phát triển hệ dịch vụ TraNeS. Các bài học kinh nghiệm này được trình bày trong ba vấn đề chính là áp dụng phương pháp luận MaSE và bộ công cụ agentTool, ứng dụng hệ đa agent trong thương mại điện tử, và xây dựng hệ đa agent trên ngôn ngữ Java và JSP.

- **Áp dụng phương pháp luận MaSE và bộ công cụ agentTool trong phát triển hệ phần mềm hướng agent**

Hệ TraNeS đã được xây dựng sử dụng phương pháp luận MaSE và agentTool. Khi áp dụng để xây dựng một hệ thống cụ thể, phương pháp luận và bộ công cụ này tỏ ra có những ưu điểm sau:

- Các bước được trong MaSE được phân tách rõ ràng với các nhiệm vụ cụ thể và đều có sơ đồ tương ứng trong agentTool. Các sơ đồ trong các bước của MaSE tương tự với các sơ đồ UML trong phân tích thiết kế hướng đối tượng nên thuận lợi cho những người đã quen phát triển các hệ phần mềm hướng đối tượng.
- Các khái niệm mới, đặc trưng của hệ đa agent như đích (goal), role (vai trò), ontology, agent hay phiên hội thoại (conversation) được biểu diễn dưới dạng các sơ đồ rất rõ ràng, dễ hiểu cho cả người phát triển hệ thống và những người nghiên cứu về hệ đa agent. Bước xây dựng ontology là một bước phức tạp nhưng MaSE đã mô hình hóa ontology dưới dạng đơn giản hơn và xác định rõ các bước cần thực hiện, qua đó làm cho quá trình này trở nên sáng sủa và dễ dàng hơn cho người phát triển.
- Tuy tách thành hai pha Phân tích và Thiết kế riêng biệt nhưng trong MaSE, hai pha này có mối quan hệ chặt chẽ với nhau, sơ đồ role, sơ đồ task và ontology trong pha phân tích sẽ là cơ sở để thiết kế các lớp agent trong pha thiết kế. Do đặc điểm này, người phát triển hệ thống có thể dễ dàng chuyển từ pha phân tích sang pha thiết kế.
- Phương pháp luận MaSE và bộ công cụ agentTool cũng hỗ trợ cho việc phát triển các ứng dụng mobile agent. Trong các ứng dụng này, các agent thể hiện tính tự chủ của mình thông qua việc có thể di chuyển từ hệ thống này sang hệ thống khác khi có yêu cầu. MaSE và agentTool hỗ trợ xây dựng các mobile agent nền

và biểu diễn cơ chế di chuyển của mobile agent theo một giao thức xác định (chẳng hạn như FIPA).

- MaSE và agentTool phù hợp với việc phát triển hệ thống từ đầu, dễ dàng áp dụng cho các hệ đa agent nhỏ và có tính chất nghiên cứu. Với các hệ thống lớn, bao gồm nhiều thành phần và xử lý thông tin trong nhiều lĩnh vực khác nhau, cách tiếp cận của MaSE sẽ phân chia hệ thống đó thành nhiều hệ thống nhỏ hơn và phát triển riêng các hệ thống nhỏ này.

- **Ứng dụng hệ đa agent trong thương mại điện tử**

Với một ứng dụng cụ thể là hệ dịch vụ TraNeS, nhóm phát triển thấy việc áp dụng hệ đa agent trong thương mại điện tử có các ưu điểm sau:

- Kiến trúc và nguyên tắc hoạt động của hệ thống được phát biểu một cách rõ ràng và gần với mối quan hệ giữa người mua và người bán trên thực tế.
- Mô hình thương lượng giữa các agent rất phù hợp cho việc mô tả quá trình giao dịch trong thương mại điện tử. Trong hệ đa agent thương mại điện tử, agent mua và agent bán sẽ tiến hành trao đổi theo các thuật toán thương lượng tự động và người mua và người bán có thể hoàn toàn trong suốt với quá trình này trong khi vẫn đạt được mục đích đề ra của mình.
- Hệ đa agent cũng rất phù hợp để mô tả các quá trình khác trong thương mại điện tử như: quảng cáo giới thiệu sản phẩm, thanh toán điện tử...

- **Xây dựng hệ đa agent trên ngôn ngữ Java và JSP**

Cho đến nay, hệ đa agent đã được xây dựng trên nhiều ngôn ngữ khác nhau như C++, Java, ... Nhóm phát triển lựa chọn ngôn ngữ Java để cài đặt hệ thống. Giao diện Web được xây dựng trên JSP. Xây dựng hệ đa agent theo Java và JSP có những điểm thuận lợi sau:

- Java cho phép cài đặt các thread hoạt động độc lập với nhau. Các thread chính là cơ sở để cài đặt các agent. Các agent sẽ luôn hoạt động một cách tự chủ và
- Sử dụng JSP có thể xây dựng hệ thống giống như một Website thương mại. Thông qua Internet, người dùng có thể dễ dàng truy nhập hệ thống và hoàn toàn trong suốt với hoạt động bên trong hệ thống.
- Java hỗ trợ việc cài đặt các lời gọi từ xa (cơ chế RMI) nên có thể xây dựng được các ứng dụng mobile agent, trong đó các agent chỉ di chuyển giữa các hệ thống khác nhau khi có yêu cầu.

8.6 Kết luận

Chương 8 đã giới thiệu các đặc trưng của hệ **TraNeS** được xây dựng dựa trên quy trình phát triển hệ phần mềm hướng agent được trình bày trong Chương 4. Nội dung chương trình bày mô hình hoạt động của hệ **TraNeS**, các nhóm chức năng chính của hệ thống này, hoạt động của hệ thống theo hai mô hình tập trung và phân tán, hoạt động của các agent chạy tại server và cách thức cài đặt và thiết lập các thông số để triển khai hệ thống. Ngoài ra, chương 8 cũng đã trình bày một số bài học kinh nghiệm rút ra từ quá trình phát triển hệ TraNeS.

KẾT LUẬN

Trong tài liệu báo cáo này, chúng tôi đã tập trung trình bày một số vấn đề sau đây:

- Phần thứ nhất trình bày cơ sở cho phát triển hệ đa agent bao gồm:
 - Tổng quan về agent và các phương pháp luận phát triển hệ đa agent: Tài liệu đã trình bày tổng quan agent, hệ đa agent và các cách tiếp cận trong xây dựng phương pháp luận phát triển hệ đa agent: phát triển dựa trên công nghệ agent, phát triển kế thừa từ phương pháp hướng đối tượng hoặc dựa trên công nghệ tri thức.
 - Tương tác trong hệ đa agent: Tương tác dựa trên ngữ nghĩa của các thông điệp truyền đi giữa các agent được xem là yếu tố then chốt phân biệt hệ đa agent và hệ đối tượng. Tài liệu đã trình bày một cách tổng quan các mô hình tương tác trong hệ đa agent và sau đó tập trung vào mô hình thương lượng. Sau đó đã đi sâu nghiên cứu mô hình thương lượng song phương dựa trên ràng buộc mờ nhằm áp dụng cho phát triển hệ dịch vụ du lịch.
 - Ontology trong hệ đa agent: Ontology được xem là biểu diễn ngữ nghĩa của thông điệp truyền đi giữa các agent. Tài liệu đã trình bày khái niệm và các cách biểu diễn ontology sau đó tập trung vào các kỹ thuật xây dựng ontology trong hệ đa agent và áp dụng vào xây dựng ontology cho hệ dịch vụ du lịch.
 - Tài liệu đã tập trung nghiên cứu quy trình phát triển hệ đa agent bao gồm các bước trong các pha từ phân tích, thiết kế đến cài đặt và tích hợp dựa trên phương pháp luận MaSE. Quy trình bao gồm các bước sau đây:

Phân tích

1. Xác định các đích
2. Xây dựng use case
3. Xây dựng ontology
4. Xây dựng sơ đồ role

Thiết kế

5. Xây dựng các lớp agent
6. Xây dựng các phiên hội thoại
7. Hoàn thiện các agent
8. Triển khai hệ thống

- Phần thứ hai là áp dụng quy trình phát triển trên cho việc phát triển hệ dịch vụ du lịch **TraNeS**. Hệ thống đã được cài đặt và thử nghiệm trên mạng cục bộ. Một số vấn đề liên quan đến cài đặt và tích hợp như mô hình kiến trúc hệ thống, tích hợp các lớp agent...cũng đã được đề cập.

Một số vấn đề tiếp tục nghiên cứu

- Các mô hình thương lượng trong thương mại điện tử: Xem xét các kiểu thương lượng trong thương mại điện tử như thương lượng song phương kết hợp tuần tự và đồng thời, các dạng đấu giá như đấu giá kiểu English hay Vickrey và đấu giá tổ hợp...
- Các kiến trúc hệ phần mềm tương ứng với các mô hình thương lượng: Xem xét các kiểu kiến trúc hệ phần mềm tương ứng với các mô hình thương lượng.
- Các kỹ thuật và công nghệ hỗ trợ dịch vụ thương lượng dịch vụ thông qua thiết bị di động.
- Agent và agent di động trong các hệ dịch vụ như thương mại điện tử, cung ứng thông tin, quản lý mạng...
- Các công nghệ hỗ trợ cho phát triển các hệ dịch vụ này.

Những vấn đề này sẽ là những chủ đề được quan tâm nghiên cứu trong thời gian tới.

Các kết quả nghiên cứu liên quan đến đề tài đã được công bố

1. *Agent đưa ra quyết định dựa trên sở thích người dùng*, Báo cáo tại Hội nghị Quốc gia về Tin học và Truyền thông, Thái Nguyên, Việt Nam, Tháng 8, 2003.
2. *Ontology và tương tác giữa các agent*, Báo cáo tại Hội nghị Quốc gia về Tin học và Truyền thông, Thái Nguyên, Việt Nam, Tháng 8, 2003.
3. *Thương lượng tự động trong thương mại điện tử*, Kỷ yếu Hội nghị Khoa học Lần thứ năm của Học Viện Công nghệ Bưu chính Viễn thông, 452-462, Hà Nội, Việt nam, Tháng 9, 2003.
4. *Tích hợp thông tin dựa trên Ontology*, Kỷ yếu Hội nghị Khoa học Lần thứ năm của Học Viện Công nghệ Bưu chính Viễn thông, 401-412, Hà Nội , Việt Nam, Tháng 9, 2003.
5. *Techniques of information integration based on ontology in developing multiagent systems*, Proceedings of Asian Info-communications Council 30th Conference, Kuala Lumpur, Malaysia, April 2004.
6. *Ontology trong thương lượng giữa các agent*, Báo cáo tại Hội nghị Quốc gia về Công nghệ thông tin và Truyền thông lần thứ VII, Đà Nẵng, Việt Nam, Tháng 8, 2004.
7. *Từ role đến các lớp agent trong thiết kế hệ đa agent*, Báo cáo tại Hội nghị Quốc gia về Công nghệ thông tin và Truyền thông lần thứ VII, Đà Nẵng, Việt Nam, Tháng 8, 2004.
8. *Combined concurrent and sequential bilateral negotiations in e-commerce*, Accepted to publish in Proceedings of International Conference M2USIC 2004, Malaysia, October 2004.

TÀI LIỆU THAM KHẢO

- [1] H. Beck, H. S. Pinto (2002), “Overview of approach, Methodologies, Standards, and Tools for Ontologies”, The Agricultural Ontology Service, UN FAO
- [2] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, Anna Perini (2002) “Tropos: An agent-oriented software development methodology” Technical Report#DIT-02-0015
- [3] Andrea Cali, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini (2002), “On the role of Integrity Constraints in Data Integration”, IEEE Computer Society Technical Committee ontology Data Engineering
- [4] B.Chandrasekaran and John R.Josephson, V.Richard Benjamins (1999), “What are Ontologies, and Why do we need them?” IEEE Intelligent Systems, 14(1), 20-26
- [5] S. A. DeLoach (2002), “AgentMom User’s Manual” Online, http://www.cis.ksu.edu/~sdeloach/ai/software/agentMom_2.0/home.html
- [6] S. A. DeLoach (2001), “Analysis and Design using MaSE and agentTool”, 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001), Miami University, Oxford, Ohio, March 31-April 1, 2001
- [7] S. A. DeLoach (2002), “Modeling Organizational Rules in the Multiagent Systems Engineering Methodology”, Proceedings of the 15th Canadian Conference on Artificial Intelligence, Calgary, Alberta, Canada, May 27-29, 2002.
- [8] J. DiLeo, T. Jacobs and S. A. DeLoach (2002), “Integrating Ontologies into Multiagent Systems Engineering”, Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems(AOSI 2002), Bologna(Italy), 15-16 July 2002.
- [9] S. A. DeLoach, Mark F. Wood and Clint H. Sparkman (2001), “Multiagent Systems engineering”, International Journal of Software Engineering and Knowledge Engineering, 11(3), 231-258
- [10] “Foundation for Intelligent Physical Agents. FIPA ACL message representation in string specification.” On line, <http://www.fipa.org/specs/fipa00070>
- [11] T. Finin, Y. Labrou et al (1997). “KQML as an agent communication language”, In J. Bradshaw, editor, Software agents. MIT Press, 291-316.

- [12] Robert Fullér (1996), “OWA Operators in Decision Making”, www.abo.fi/~rfuller/rem961.pdf.
- [13] M. R. Genesereth and Steven P. Ketchpel (1994), “Software Agents”, 37(7)
- [14] M. Georgeff, B. Pell, M. Pollack, M. Tambe and M. Wooldridge, (1999) “The Belief-Desire-Intention Model of Agency”, Proceedings of Agents, Theories, Architectures and Languages (ATAL).
- [15] T. R. Gruber (1994), “Toward Principles for the Design of Ontologies Used for Knowledge Sharing”, In Formal Ontology in Conceptual Analysis and Knowledge Representation, Guarino and Poli (Eds.). Kluwer Academic Publishers.
- [16] N. Guarino (1998), “Formal Ontology and Information Systems”, National Research Cuoncil, LADSEB-CNR, Corso Stati Uniti 4, I-35127 Padova, Italy
- [17] Jeffrey Douglas Heflin, Doctor of Philosophy (2001), “Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment”, Dissertation.
- [18] Minghua He, Nicholas R. Jennings, Ho-fung Leung (2002), “On Agent-Mediated Electronic Commerce”. IEEE Transactions on Knowledge and Data Systems, 15(4), 2003.
- [19]. F. Herrera, E. Herrera – Veidma (1998), “Linguistic Decision Analysis: Steps for Solving Decision Problems under Linguistic Information”, Fuzzy Sets and Systems 115 (2000) 67-82.
- [20] Michael N. Huhns and Larry M. Stephens (1999), “Multiagent Systems and Societies of Agents”, Multiagent systems: a modern approach to distributed artificial intelligence table of contents, pages 79-120, MIT Press Cambridge, MA, USA.
- [21] C.A.Iglesias, M. Garijo, J. C.Gonzalez, J. R. Velasco, “Analysis and Design of Multiagent Systems using MAS-CommonKADS”, In Proceeding of AAAI’97, Workshop on Agent Theories, Architectures and languages, Providence, RI, 1997.
- [22] Nicholas R. Jennings (1999), “On agent-based software engineering” Artificial Intelligence 117 (2000) 277–296.
- [23] Nicholas R. Jennings, Katia Sycara, Michael Wooldridge (1998), “A Roadmap of Agent Research and Development”, Autonomous Agents and Multi-Agent Systems, 1, 7-38 (1998).

- [24] Matthias Klush and Katia Sycara (2001), “Brokering and Matchmaking for Coordination of Agent Societies”, In Coordination of Internet Agents, A. Omicini et al. (eds.), Springer., 2001.
- [25] X. Luo, J. Lee. H. Liung and N. R. Jennings (2002), “Prioritised Fuzzy Constraint Satisfaction Problems: Axioms, Instantiation and Validation”, Journal of Fuzzy Sets and Systems, 136, (2), 155 – 188, 2002.
- [26] X. Luo, N. R. Jennings, N. Shadbolt, H. Liung and J. H. Lee (2003). “A Fuzzy Constraint Based Model for Bilateral, Multi-issue Negotiations in Semi-competitive Environments”, Artificial Intelligence Journal 148 (1-2) 53-102.
- [27] “MESSAGE: Methodology for Engineering Systems of Software Agents, Deliverable 1: Initial Methodology”, July 2000, EURESCOM Project P907-GI.
- [28] P. Mitra and G. Weiderhold (2002), “An Algebra for Composition of Ontologies”, Inforlab, Stanford University, CA 94305, USA.
- [29] T. R. Payne, M. Paolucci, R. Singh, and K. Sycara (2002), “Communicating Agents in Open Multi Agent Systems”, In *Proceedings of First GSFC/JPL Workshop on Radical Agent Concepts (WRAC)* , (365-371), McLean, VA, USA
- [30] A. Preece (2001), “A Mediator-based Infrastructure for Virtual Organisation”, University of Aberdeen, Computing Science Department, Germany.
- [31] H. Scholten, A. J.M. Beulens (2002), “Ontologies to structure models and modeling tasks”, IIASA, Laxenburg, Austria, July 15-17-2002.
- [32] K. B. Shaban (2002), “Information fusion in a cooperative Multi-Agent system for Web information retrieval”, A Thesis Master of Science Presented to The Faculty of Graduate Studies.
- [33] H. Wache, T.Vogele, U.Visser, H.Stuckenschmidt, G.Schuster, H.Neumann and S.Hubner (2001), “Ontology-based Integration of Information – A Survey of Existing Approaches”, Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, 2001, vol. pp.108-117.
- [34] G. Schuster and H. Stuckenschmidt (2001), “Building Shared Ontologies for Terminology Integration”, Workshop on Ontologies, KI-2001, September 18, 2001, Vienna, Austria.
- [35] K. P. Sycara (1998), “Multiagent Systems”, American Association for Artificial Intelligence, AAAI AI Magazine 19(2).

- [36] U. Mike and M. Gruninger (1996), “Ontologies: Principle, Methods and Application”, *The Knowledge Engineering Review*. 1996, 11(2): 93-136.
- [37] Gerhard Weiss (2002) “Agent Orientation in Software Engineering” Revised version for knowledge engineering review, Vol. 16(4), 349-373.
- [38] M. Wooldridge, N. R. Jennings, D. Kinny (2000), “The GAIA Methodology for Agent – Oriented Analysis and Design”, *Autonomous Agents and Multi-Agent Systems*, 3, 285-312, 2000.
- [39] M. Winikoff and L. Padgham RMIT University (2002) “Agent design methodologies: What, How, Tools, and Issues”, Workshop on Agent Oriented Software Engineering, Nov. 2002, Seattle, USA.
- [40] H.C Wong and K. Sycara (2000), “A Taxonomy of middle-agents for the Internet”, Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS 2000), 465-466, 2000.
- [41] Soe-Tsyr Yuan (1999), “Ontologies-based Agent Community for Information Gathering and Integration”, Proc.Natl.Sci.Counc.ROC(A), 23(6), 766-781, June 1999.