**Gaia - Phương pháp luận phân tích và thiết kế hướng Agent**

**Tóm tắt**

Bài viết này trình bày về Gaia: một phương pháp luận để phân tích và thiết kế hướng Agent. Phương pháp luận Gaia được áp dụng cho một loạt các hệ thống đa agent, ở cả hai cấp độ vĩ mô (xã hội) và vi mô (đại lý) của hệ thống. Gaia được hình thành trên quan điểm nhìn nhận một hệ đa agent là một tổ chức tính toán bao gồm các vai trò tương tác khác nhau. Chúng tôi minh họa Gaia thông qua một tình huống (một hệ thống quản lý quy trình kinh doanh dựa trên Agent).

**1. Giới thiệu**

Tiến bộ trong công nghệ phần mềm trong hai thập kỷ qua đã được thực hiện thông qua sự phát triển mạnh mẽ và trừu tượng hóa ở mức cao để mô hình hóa và phát triển các hệ thống phức tạp. Trừu tượng về thủ tục, các kiểu dữ liệu trừu tượng, và gần đây nhất, các đối tượng và thành phần đều là ví dụ về trừu tượng. Chúng tôi tin rằng Agent đại diện cho một bước tiến tương tự trong trừu tượng: chúng có thể được sử dụng bởi các nhà phát triển phần mềm để hiểu tự nhiên hơn, mô hình, và phát triển một lớp quan trọng trong hệ thống phân phối phức tạp.

Nếu các Agent là để nhận ra tiềm năng của chúng như là một mô hình công nghệ phần mềm, thì cần phát triển công nghệ phần mềm được thiết kế đặc biệt cho chúng. Những công nghệ phát triển phần mềm đang tồn tại (ví dụ, phân tích thiết kế hướng đối tượng) không phù hợp cho công việc này. Sự không phù hợp cơ bản là giữa các khái niệm được sử dụng bởi các nhà phát triển hướng đối tượng và các quan điểm hướng Agent [32, 34]. Đặc biệt, Các phương pháp tiếp cận hiện có không nắm bắt đầy đủ tính linh hoạt của Agent, tự chủ hành vi trong giải quyết các vấn đề, sự phong phú trong tương tác của các Agents, và sự phức tạp về cơ cấu tổ chức của một hệ thống Agent. Từ những lý do này, bài viết này giới thiệu một phương pháp gọi là Gaia, được thiết kế đặc biệt cho việc phân tích và thiết kế các hệ thống dựa trên Agent.

Phần còn lại của bài viết này được cấu trúc như sau. Chúng ta bắt đầu bằng việc thảo luận các đặc điểm của ứng dụng phù hợp với Gaia. Phần 2 đưa ra một cái nhìn tổng quan về các khái niệm chính được sử dụng trong Gaia. Phân tích dựa trên Agent được thảo luận trong phần 3, và thiết kế trong phần 4. Cách thức sử dụng Gaia sẽ được minh họa bằng tình huống trong phần 5, nơi mà chúng ta sẽ chỉ ra cách làm thế nào để áp dụng Gaia để thiết kế một hệ thống hướng Agent nhằm quản lý quá trình kinh doanh. Các nghiên cứu có liên quan được thảo luận trong phần 6, và một số kết luận được trình bày trong phần 7.

***Miền lý thuyết***

Trước khi tiếp tục, đây là giá trị về phạm vi nghiên cứu của chúng tôi, và đặc biệt, trên các đặc điểm của lĩnh vực mà chúng tôi tin rằng Gaia là thích hợp. Gaia được dự trù là hướng thích hợp để phát triển của các hệ thống như ADEPT và Archon. Đây là những ứng dụng thực tế quy mô lớn, với những đặc điểm chính sau đây:

+ Agents là hệ thống tính toán thô, được sử dụng tính toán tài nguyên quan trọng (Giống như là mỗi Agent có nguồn tài nguyên của một tiến trình UNIX).

+ Giả định rằng mục tiêu là có được một hệ thống mà tối ưu hóa chất lượng toàn cục, nhưng cũng có thể là tối ưu cục bộ với quan điểm của các hệ thống thành phần. Gaia *không* dành cho các hệ thống thừa nhận khả năng xung đột.

+ Agents rất khác nhau, trong đó các Agent khác nhau có thể được thực hiện bằng cách sử dụng các ngôn ngữ lập trình khác biệt, kiến trúc và kỹ thuật. Chúng tôi không giả định về nền tảng.

+ Cơ cấu tổ chức của hệ thống là tĩnh, trong đó mối quan hệ giữa các Agent bất biến theo thời gian chạy.

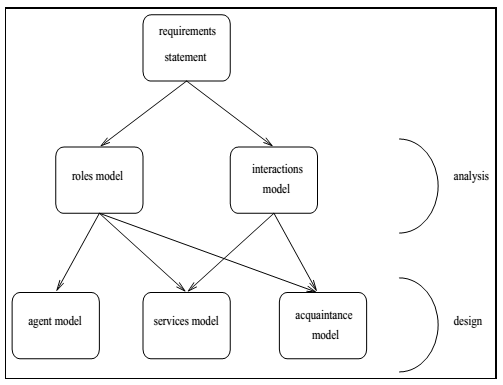
+ Các khả năng của các Agent và các dịch vụ mà chúng cung cấp là tĩnh, trong đó chúng không thay đổi tại thời gian chạy.

+ Các hệ thống tổng thể có chứa một số lượng tương đối nhỏ các loại Agent khác nhau (ít hơn 100).

Gaia giải quyết ở cả cấp vĩ mô (xã hội) và vi mô (Agent) ở khía cạnh thiết kế. Nó thể hiện một bước tiến so với các phương pháp định hướng Agent trước đây ở chỗ, nó trung lập đối với cả lĩnh vực mục tiêu và các kiến trúc agent (xem phần 6 cho so sánh chi tiết hơn).

**2. Khái niệm nền tảng**

Gaia được mong đợi sẽ phép các nhà phân tích có được hệ thống, từ lời phát biểu yêu cầu đến một thiết kế đầy đủ chi tiết mà có thể triển khai trực tiếp. Lưu ý rằng, chúng ta xem giai đoạn xác định yêu cầu là độc lập trong mô hình được sử dụng để phân tích và thiết kế. Khi áp dụng Gaia, các nhà phân tích đi từ trừu tượng đến ngày càng cụ thể hóa khái niệm. Mỗi di chuyển liên tiếp tạo ra sự chênh lệch lớn hơn, và thu nhỏ không gian của khả thi hệ thống sẽ được thực hiện để đáp ứng các yêu cầu ban đầu. Phân tích và thiết kế có thể được coi như là một quá trình phát triển  ngày càng chi tiết  *mô hình* của hệ thống được xây dựng. Các mô hình chính được sử dụng trong Gaia được tóm tắt trong Hình 1



*Hình 1.* Mối quan hệ giữa các mô hình của Gaia

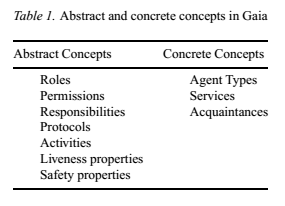
Gaia mượn một số thuật ngữ và ký hiệu từ phân tích và thiết kế hướng đối tượng. Tuy nhiên, nó không chỉ đơn giản là cố gắng để áp dụng phương thức như vậy để phát triển theo hướng Agent. Thay vào đó, nó cung cấp một Agent cụ thể của tập hợp khái niệm, thông qua đó một kỹ sư phần mềm có thể hiểu và mô hình một hệ thống phức tạp. Đặc biệt, Gaia khuyến khích một nhà phát triển suy nghĩ về việc xây dựng các hệ thống như là một quá trình *thiết kế tổ chức.*

Các khái niệm chính của Gaia có thể được chia thành hai loại: *trừu tượng và* cụ thể; khái niệm trừu tượng và cụ thể được tóm tắt trong Bảng 1. Thực thể trừu tượng được sử dụng trong suốt quá trình phân tích để khái quát hóa hệ thống, nhưng nó không nhất thiết phải có *bất* kỳ thực hiện trực tiếp nào trong hệ thống. Thực thể cụ ​​thể, ngược lại, được sử dụng trong quá trình thiết kế, và thông thường sẽ phải bổ sung trực tiếp trong thời gian chạy hệ thống.

**3. Phân tích**

Mục tiêu của giai đoạn phân tích là phát triển hiểu biết về hệ thống và cấu trúc của nó. Trong trường hợp của chúng tôi, sự hiểu biết được nắm bắt trong *tổ chức* của hệ thống. Chúng tôi xem một tổ chức như một tập hợp các vai trò trong mối quan hệ nhất định với nhau và tham gia vào hệ thống, mô hình tương tác với các vai trò khác - xem hình 2.

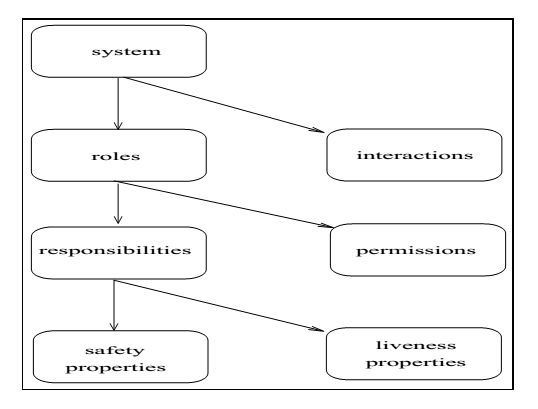
Các thực thể trừu tượng nhất trong phân cấp khái niệm của chúng tôi *là* hệ thống. Mặc dù thuật ngữ "hệ thống" được dùng theo nghĩa riêng, nhưng cũng có ý nghĩa liên quan khi nói về một hệ thống dựa trên agent, có nghĩa là một xã hội hay một tổ chức. Do đó, chúng tôi nghĩ hệ thống dựa trên Agent là một cộng đồng hoặc tổ chức nhân tạo.



Ý tưởng về một hệ thống như một xã hội là hữu ích khi nghĩ về mức độ tiếp theo trong hệ thống phân cấp khái niệm: *vai trò (Roles).* Thật lạ để nghĩ rằng hệ thống máy tính được định nghĩa là một tập hợp các vai trò, nhưng ý tưởng này là khá tự nhiên khi chấp nhận một quan điểm tổ chức của thế giới thực. Hãy xem xét một tổ chức của con người như một công ty. Công ty có vai trò như "chủ tịch", "phó chủ tịch"... Lưu ý rằng trong *thực* *thể cụ thể* của một công ty, các vai trò sẽ *được* khởi tạo với các cá nhân thực tế: sẽ có một người sẽ vào vai trò của chủ tịch, một người có vai trò phó chủ tịch... Tuy nhiên, các thực thể là không nhất thiết phải tĩnh. Ví dụ, trong vòng đời của công ty, nhiều cá nhân có thể đảm nhận vai trò của Chủ tịch công ty. Cũng thế, không nhất thiết phải là một ánh xạ một-một giữa vai trò và cá nhân. Một cách không thông dụng (đặc biệt là trong các tổ chức nhỏ hoặc không chính thức được) một cá nhân có nhiều vai trò. Ví dụ, một cá nhân duy nhất có thể đảm nhận vai trò của " tea maker", "Mail fetcher", và như vậy. Ngược lại, có thể có nhiều cá nhân có cùng một vai trò, ví dụ, "nhân viên bán hàng".

Một vai trò được quy định bởi bốn thuộc *tính:* *Trách nhiệm*, *quyền hạn, hoạt động,* và *giao thức. Trách nhiệm* xác định chức năng và thuộc tính khóa liên kết với một vai trò. Một ví dụ về trách nhiệm liên quan đến vai trò của giám đốc công ty có thể triệu tập đại hội cổ đông hàng năm. Trách nhiệm chia thành thành hai *loại:* thuộc tính thiết yếu *và* thuộc tính *an toàn*. Thuộc tính thiết yếu có thể nhân thức trực giác là "điều tốt đẹp sẽ xảy ra". Chúng mô tả những trạng thái của công việc mà Agent phải đem lại, đưa ra điều kiện môi trường chắc chắn. Ngược lại, thuộc tính an toàn là *bất biến.* Bằng trực giác, một thuộc tính an toàn là "không có gì xấu xảy ra". Một ví dụ "đảm bảo nhiệt độ lò phản ứng luôn luôn nằm trong khoảng 0-100".

Để thực hiện trách nhiệm, mỗi vai trò có một *tập* các quyền. Quyền được gắn với mỗi vai trò. Các quyền cho từng vai trò phải xác định các nguồn tài nguyên có sẵn cho vai trò đó để thực hiện trách nhiệm của mình. Trong các loại hệ thống chúng tôi đã mô hình hóa, quyền hạn có xu hướng là *các nguồn tài nguyên thông tin.* Ví dụ như, một vai trò có thể liên kết với khả năng đọc một mục cụ thể của thông tin hoặc sửa đổi một phần của thông tin. Một vai trò cũng có thể có *khả* năng tạo ra các thông tin.



*Hình 2.* Phân tích các khái niệm

*Các* hoạt động của một vai trò được tính toán liên quan đến vai trò mà Agent có thể thực hiện mà không tương tác với các Agent khác. Các hành động như vậy, được xem như hành động "riêng”. Cuối cùng, vai trò cũng được xác định với một *số* giao thức, trong đó xác định các cách mà nó có thể tương tác với các vai trò khác. Ví dụ, một vai trò "người bán" có thể có các giao thức " đấu giá Hà Lan "và" đấu giá Anh" liên kết với nó. Như vậy, các mô hình tổ chức trong Gaia bao gồm hai mô hình: các *mô hình vai trò* (mục 3.1) và các *mô hình tương tác* (phần 3.2).

*3.1.* M*ô hình vai trò*

Mô hình vai trò xác định vai trò quan trọng trong hệ thống. Ở đây một vai trò có thể được xem như một mô tả trừu tượng các chức năng dự kiến của thực thể. Nói cách khác, một vai trò là nhiều hơn hoặc ít hơn so với khái niệm của *một* văn phòng trong ý nghĩa rằng "Thủ tướng", "luật sư nói chung của Hoa Kỳ ", hoặc" thư ký của nhà nước cho giáo dục " đều là các văn phòng. Như thế vai trò (hay văn phòng) được đặc trưng bởi hai loại thuộc tính

*Các quyền gắn liền với vai trò.*

Một vai trò sẽ gắn kết với quyền nhất định, liên quan đến các loại và số lượng tài nguyên có thể khai thác khi thực hiện vai trò này. Trong trường hợp của chúng ta, những khía cạnh được giữ trong một thuộc tính được gọi là *quyền* *của* vai trò.

*Trách nhiệm của vai trò.*

Một vai trò được tạo ra để *làm* một cái gì đó. Đó là, một vai trò có một chức năng nhất định. Chức năng này được đại diện bởi một thuộc tính được gọi là *trách nhiệm* của vai trò.

*Quyền*  Các quyền liên quan đến một vai trò có hai khía cạnh:

+ Quyền xác định các tài nguyên hợp pháp có thể sử dụng để thực hiện vai trò – trực quan mà nói là “những gì *có thể* dùng khi thực hiện vai trò”;

+ Quyền nêu rõ giới hạn tài nguyên mà trong đó vai trò phải hoạt động - trực giác,

Có thể nói là “*những* gì không thể dùng khi thực hiện vai trò”.

Nói chung, quyền liên quan đến bất kỳ loại tài nguyên nào. Trong một tổ chức con người, một vai trò có thể được cấp một ngân sách tiền tệ, một số tiền nhất định cho nỗ lực con người. Tuy nhiên, trong Gaia, chúng tôi nghĩ rằng các nguồn tài nguyên chỉ liên quan đến các *thông tin* hay *tri thức mà Agent* có. Đó là, để thực hiện một vai trò, một Agent sẽ được truy cập thông tin nhất định. Một số vai trò có thể tạo ra thông tin; số khác cần truy cập vào một phần thông tin nhưng không sửa đổi nó, trong khi những người khác có thể cần sửa đổi các thông tin. Chúng tôi nhận ra rằng phải có một mô hình giàu tài nguyên hơn là cần thiết trong tương lai, mặc dù cho thời điểm này, chúng tôi chỉ đơn giản là thông tin.

Gaia sử dụng một ký hiệu chuẩn để nhấn mạnh quyền, dựa trên ký hiệu FUSION cho lược đồ hoạt động [6, pp26-31]. Để giới thiệu các khái niệm của chúng tôi, chúng tôi sẽ lấy ví dụ về một vai trò CoffeeFller (mục đích của vai trò này là để đảm bảo rằng nồi cà phê đầy đủ cà phê cho nhóm công nhân). Sau đây là một minh họa đơn giản các điều khoản liên quan đến vai trò CoffeeFIller

*reads coffeeStatus*// *Đầy hoặc rỗng*

changes *coffeeStock* // *Cấp cổ phiếu của cà phê*

Đặc điểm kỹ thuật này định nghĩa hai quyền cho CoffeeFller: Nó nói rằng các Agent thực hiện vai trò được quyền truy cập vào giá trị *coffeeStatus*, và có quyền đọc và chỉnh sửa các giá trị *coffeeStock*. Ngoài ra còn có một loại quyền thứ ba, *generates*, để chỉ ra rằng vai trò là nhà sản xuất của một tài nguyên (không có trong ví dụ). Lưu ý rằng các quyền liên quan đến tri thức mà *các Agent có.* Đó là, *coffeeStatus* là biểu diễn của một số giá trị trong thế giới thực.

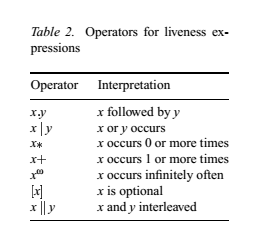
Một số vai trò được tham số hóa (*parameterised)* bởi những giá trị nhất định. Ví dụ, chúng ta có thể khái quát vai trò COFFEEFILLER bởi parameterising nó với máy pha cà phê mà là để được giữ nạp lại.

Đây là định nghĩa quyền bằng cách cung cấp từ khóa, như sau:

Reads supplied coffeeMaker // Tên máy cà phê

Coffee Status // đầy hoặc rỗng

*Changes coffeeStock* // *Cấp cổ phiếu của cà phê*

**

*Trách nhiệm: Chức năng của một* vai trò được định nghĩa *bởi* các trách nhiệm. Trách nhiệm có thể được chia thành hai loại: Trách nhiệm *thiết yếu* và *an toàn.* Trách nhiệm thiết yếu, nói một cách trực giác, đó là "một cái gì đó tốt sẽ xảy ra".

Trách nhiệm thiết yếu được gọi như vậy vì chúng có xu hướng cho rằng "một cái gì đó sẽ được thực hiện ", và vì thế mà các Agent thực hiện vai trò là vẫn còn sống. Trách nhiệm thiết yếu có xu hướng theo những mẫu nhất định. Ví dụ, loại *guaranteed response của mục tiêu* có dạng "một yêu cầu luôn luôn kéo theo một phản hồi". *Vòng*  lặp lại *vô hạn* có dạng *"x* sẽ xảy ra vô hạn". Lưu ý rằng các loại yêu cầu đã được nghiên cứu rộng rãi trong các tài liệu công nghệ phần mềm, nơi mà chúng đã được chứng minh là cần thiết cho việc chụp những thuộc tính của hệ thống *phản ứng* [27].

Để minh họa cho các khái niệm khác nhau liên quan đến vai trò, chúng tôi sẽ tiếp tục với ví dụ của chúng ta về vai trò CoffeeFIller. Ví dụ về các trách nhiệm thiết yếu cho các vai trò CoffeeFIller có thể là:

+ Bất cứ khi nào nồi cà phê là rỗng thì đổ đầy nó;

+ Bất cứ khi nào cà phê tươi được ủ thì chắc chắn các công nhân biết về nó.

Trong Gaia, tính thiết yếu được xác định thông qua *một* liveness expression*,* trong đó xác định "vòng đời "của vai trò. Liveness expression tương tự như các biểu hiện *vòng đời của* FUSION, đó thực chất là biểu thức thông thường. Liveness expression của chúng tôi có một phép toán bổ sung “Ω” cho *vòng lặp vô hạn (xem* Bảng 2 để biết thêm chi tiết). do đó chúng giống với biểu thức chính quy được cho là thích hợp để đại diện tính chất vô hạn [32].

Liveness expression xác định quỹ đạo thực hiện tiềm năng thông qua chuỗi các hành vi và tương tác (tức là, qua các giao thức) kết hợp với vai trò. Hình thức chung của một liveness expression là:

ROLENAME = expression

Với ROLENAME là tên của vai trò mà các thuộc tính thiết yếu đang được định nghĩa, *và* expression là liveness expression định nghĩa thuộc tính thiết yếu của ROLENAME.

Các thành phần nguyên tử của một liveness expression là *activities* hoặc *protocols*. Một *activities* có phần giống protocols về hướng đối tượng, hoặc một thủ tục trong ngôn ngữ lập trình PASCAL. Nó tương ứng với một đơn vị của hành động mà các Agent có thể thực hiện, mà không không liên quan đến sự tương tác với bất kỳ Agent nào khác. Protocol, mặt khác, là những hoạt động *mà* đòi hỏi sự tương tác với các Agent khác. Để cung cấp cho người đọc một số đầu mối trực quan, chúng tôi viết tên giao thức trong một phông chữ sans serif, và sử dụng một phông chữ tương tự, gạch dưới, cho tên activities.

Để minh họa liveness expressions, hãy xem xét lại khái niệm về trách nhiệm ở trên, các vai trò CoffeeFIller:

COFFEEFILLER = (Fill. InformWorkers.CheckStock.AwaitEmty) ω

Biểu thức này cho biết CoffeeFIller bao gồm: thực thi giao thức Fill, tiếp sau là giao thức InformWorkers, tiếp theo là activities CheckStock và giao thức AwaitEmpty. Sự thực hiện tuần tự của các giao thức và các activities được lặp đi lặp lại vô hạn. Đối với thời điểm này, chúng ta phải xem xét với các giao thức đơn giản như các nhãn cho các tương tác và có trách nhiệm không lo lắng về việc làm thế nào xác định được chúng (vấn đề này sẽ được thảo luận trong mục 3.2).

Liveness expressions phức tạp có thể làm cho dễ đọc hơn bằng cách cơ cấu lại chúng. Một ví dụ đơn giản minh họa cách thực hiện:

COFFEEFILLER = (All)ω

ALL = Fill.InformWorkers.CheckStock.AwaitEmpty

Ngữ nghĩa của các định nghĩa như là sự thay thế văn bản.

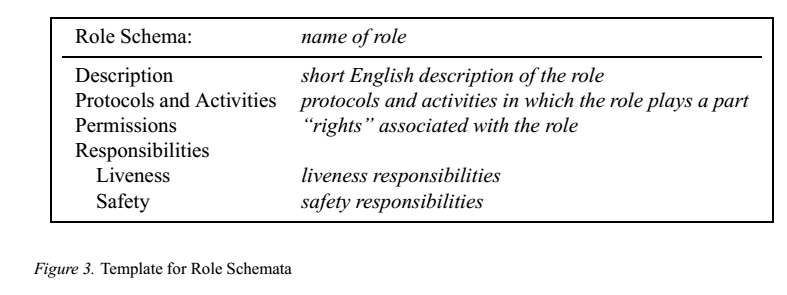
Trong nhiều trường hợp, không đơn giản để xác định trách nhiệm thiết yếu của một vai trò. Điều này là do một Agent, thực hiện vai trò, sẽ được yêu cầu duy trì *bất biến* (invariants) nào đó trong khi thực hiện. Ví dụ, chúng ta có thể yêu cầu một Agent đặc biệt tham gia một ứng dụng thương mại điện tử không bao giờ chi tiêu nhiều tiền hơn nó đã được phân bổ. Những bất biến được gọi là *điều* kiện an toàn, bởi vì chúng thường liên quan đến sự vắng mặt của một số điều kiện không mong muốn phát sinh.

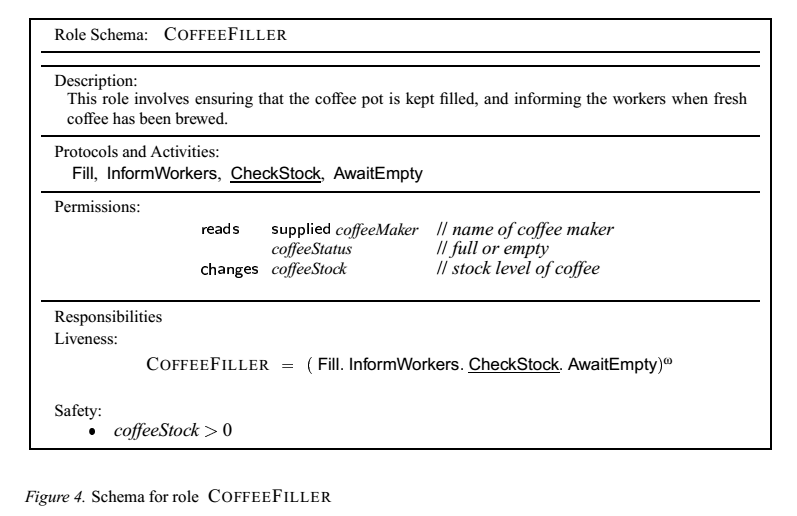
Yêu cầu an toàn trong Gaia được quy định bằng theo nghĩa một danh sách của các vị ngữ. Những vị ngữ thường được thể hiện qua các biến được liệt kê trong điều khoản thuộc tính của một vai trò.

Quay trở lại với vai trò CoffeeFIller của chúng ta, một Agent thực hiện vai trò nói chung sẽ yêu cầu kho cà phê là không bao giờ trống rỗng. Chúng tôi có thể biểu diễn điều kiện này bằng biểu thức an toàn như sau:

*coffeeStock >* 0

Theo quy ước, chúng ta chỉ cần liệt kê các biểu thức an toàn như một danh sách liệt kê, mỗi mục trong danh sách thể hiện một trách nhiệm an toàn riêng. Nó mặc định cho rằng trách nhiệm được áp dụng tới tất cả các trạng thai thực thi của hệ thống. Nếu vai trò là kéo dài vô hạn (như trong ví dụ COffeeFIller), thì invariants *luôn* nhận giá trị True. Bây giờ, có thể xác định chính xác các mô hình vai trò Gaia. Một mô hình vai trò bao gồm một tập hợp các *lược đồ vai trò (role schemata),* một cho mỗi vai trò trong hệ thống. Một *lược đồ vai trò* vai trò kéo theo một chuỗi thuộc tính khác nhau đã thảo luận ở trên vào một nơi duy nhất (hình 3).

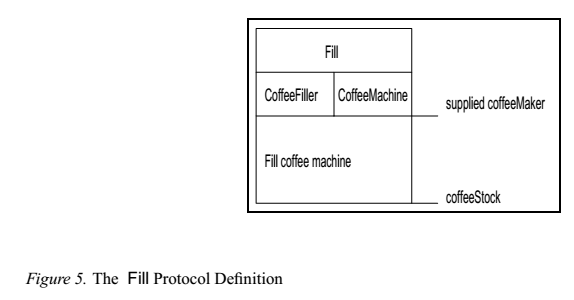




Môt thí dụ mẫu mực được đưa ra cho các vai trò CoffeeFiller trong hình 4. Giản đồ này chỉ ra rằng CoffeeFiller có quyền đọc thông số *máy pha cà phê* (vai trò nhằm giữ đầy máy cà phê), và các *coffeeStatus* (xem máy là đầy hoặc rỗng). Ngoài ra, vai trò có quyền thay đổi các giá trị *coffeeStock.*

*3.2.* Mô hình *Tương tác*

Có sự phụ thuộc và mối quan hệ chặt chẽ giữa các vai trò khác nhau trong một tổ chức đa Agent. Thật vậy, tác động lẫn nhau là cách thức trung tâm trong chức năng hệ thống. Với thực tế này, các tương tác cần phải được hiểu và biểu diễn trong pha phân tích.



Trong Gaia, liên kết như vậy giữa các vai trò được thể hiện *trong* mô hình *tương tác.*

Mô hình này bao gồm một tập hợp các *định nghĩa giao thức,* mỗi cá thể là một loại liên vai trò tương tác. Dưới đây là một giao thức có thể được xem như là một mô hình thể chế của tương tác. Đó là, một mô tương tác được chính thức định nghĩa và trừu tượng hóa từ dãy các bước thực thi. Xem các tương tác theo cách này có nghĩa là tập trung chú ý vào các tính chất cần thiết và mục đích của sự tương tác, chứ không dựa vào trật tự chính xác của các thông điệp trao đổi (các sơ đồ tương tác của OBJECTORY hoặc các kịch bản của FUSION).

Cách tiếp cận này có nghĩa rằng một định nghĩa giao thức thường làm tăng số nút trao đổi thông điệp trong thời gian chạy hệ thống. Ví dụ, hãy xem xét giao thức đấu giá Anh. Điều này liên quan đến nhiều vai trò (người bán và các nhà thầu) và nhiều mẫu tiềm năng của interchange (báo giá cụ thể và giá thầu tương ứng). Tuy nhiên ở

giai đoạn phân tích, độ chính xác một ví dụ chi tiết là không cần thiết, và quá sớm.

Một định nghĩa giao thức bao gồm các thuộc tính sau:

*Mục đích( purpose):*  đoạn ngắn mô tả về bản chất của sự tương tác (ví dụ, "thông tin

yêu cầu "," lập lịch hành vi "và" giao nhiệm vụ ");

*Người khởi xướng (initiator):* vai trò trách nhiệm khởi động sự tương tác;

*Người trả lời (responder):* Các vai trò (s) mà người khởi xướng tương tác;

*Đầu vào:* thông tin được sử dụng bởi vai trò người khởi xướng trong khi ban hành các giao thức;

*Kết quả đầu ra:* thông tin cung cấp để trả lời đến người yêu cầu giao thức trong quá trình tương tác;

*Xử lý (processing):*  văn bản ngắn gọn mô tả các xử lý của các người khởi tạo giao thức thực hiện trong quá trình tương tác.

Minh họa, xem xét giao thức Fill, là một phần của vai trò CoffeeFIller Hình 5). Điều này khẳng định rằng giao thức Fill được khởi tạo bởi vai trò CoffeeFiller và liên quan đến vai trò COffeeMAchine. Giao thức liên quan CoffeeFIller đưa cà phê vào máy có tên coffeeMaker, và kết quả CoffeeMAchine được thông báo về giá trị của *coffeeStock.* Chúng ta sẽ thấy xem xét ví dụ về các giao thức trong phần 5.

*3.3.* *Quá trình phân tích*

Các giai đoạn phân tích của Gaia bây giờ có thể tóm tắt như sau:

1. Xác định *vai* trò trong hệ thống. Vai trò trong một hệ thống thông thường sẽ tương ứng với:

+ Cá nhân, hoặc một tổ chức, một hoạt động độc lập;

+ Phòng ban trong một tổ chức;

+ Tổ chức.

*Output:* Một mô hình vai trò nguyên mẫu - một danh sách các vai trò quan trọng xảy ra trong hệ thống.

2. Đối với mỗi vai trò, xác định và ghi lại *các giao thức* liên quan. Giao thức là các mẫu của sự tương tác xảy ra trong hệ thống giữa các vai trò khác nhau. Ví dụ, một

giao thức có thể tương ứng với một Agent trong vai trò của BUYER nộp thầu (trả giá) khác với Agent trong vai trò của SELLER.

*Output:* Một mô hình tương tác, sẽ chụp các mẫu hình chu kỳ của liên vai trò liên

hành động.

3. Sử dụng mô hình giao thức như là một cơ sở, xây dựng mô hình vai trò.

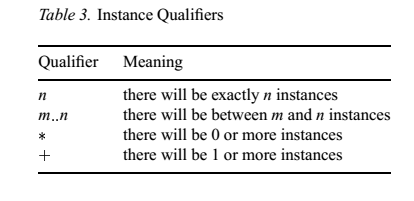
*Output:* Một mô hình vai trò xây dựng đầy đủ, với thông tin về vai trò quan trọng xảy ra trong hệ thống, quyền hạn và trách nhiệm của mình, cùng với các giao thức và các hoạt động trong đó họ tham gia.

4. Giai đoạn Duyệt (1) - (3).

**4. Thiết kế**

Mục đích của quá trình thiết kế "cổ điển" là chuyển đổi các mô hình trừu tượng thu được trong giai đoạn phân tích thành các mô hình trừu tượng ở mức thấp hơn để có thể triển khai chúng dễ dàng. Đây không phải là trường hợp thiết kế theo hướng Agent. Thay vào đó, mục đích của Gaia là biến đổi các mô hình phân tích sang một mức độ trừu tượng đủ thấp để có thể được áp dụng các kỹ thuật thiết kế truyền thống (bao gồm cả kỹ thuật hướng đối tượng) để bổ sung Agent theo thứ tự. Nói một cách khác, Gaia liên quan đến cách một xã hội Agent hợp tác để thực hiện mục tiêu cấp hệ thống, và những gì được yêu cầu của từng Agent riêng lẻ để làm điều này. Trên thực tế *cách thức* một agent thực hiện các dịch vụ của nó vượt quá phạm vi của Gaia và phụ thuộc vào các miền ứng dụng cụ thể.

Quá trình thiết kế Gaia liên quan đến việc tạo ra ba mô hình (xem hình 1). Các mô hình Agent (agent model) *xác* định các loại *Agent*  sẽ tạo thành hệ thống,  các thực thể Agentsẽ được khởi tạo từ các loại trên. Các *services model* xác định các dịch vụ chính mà được yêu cầu vai trò Agent thực hiện. Cuối cùng, các *acquaintance model* cung cấp các đường giao tiếp giữa các Agent khác nhau.



*4.1.* *Mô hình Agent*

Mục đích của mô hình Agent Gaia là cung cấp *các* loại *Agent* khác nhau sẽ được sử dụng trong các hệ thống, *và* các thực thể Agents  thu được từ các loại Agent trên trong thời gian chạy.

Loại Agent được cho là tốt nhất là một tập hợp các vai trò Agents. Thực tế đó là quan hệ 1 – 1 giữa các Roles (như được xác định trong mô hình vai trò - xem mục 3.1) và các loại Agent. Một nhà thiết kế có thể chọn một số Roles liên quan chặt chẽ với nhau trong cùng loại Agent cho thuận tiện. Hiệu quả là mối quan tâm lớn ở giai đoạn này: một nhà thiết kế sẽ gần như chắc chắn muốn tối ưu hóa thiết kế, và một cách để làm điều này là để tổng hợp một số vai trò Agent vào một loại.

Các mô hình Agent được xác định bằng cách sử dụng một *cây chủng loại Agent* đơn giản (agent type tree), trong đó các nút lá tương ứng với vai trò, và các nút khác tương ứng với các loại Agent. Nếu một loại Agent *t*1 có các nút con là t2, t3 thì có nghĩa là t1 bao gồm các vai trò tạo nên t2, t3.

Chúng tôi ghi lại các thực thể Agent xuất hiện trong hệ thống bằng cách chú thích loại Agent trong mô hình Agent. Một chú thích *n* có nghĩa là có chính xác *n* loại Agent thời gian chạy hệ thống. Một chú thích *m* ..*n* có nghĩa là sẽ có không ít hơn *m* và nhiều hơn *n* loại Agent trong thời gian chạy (m <n)*.* Một chú thích \* có nghĩa là sẽ có không một hoặc nhiều thực thể trong thời gian chạy, và + có nghĩa là sẽ có một hoặc nhiều thực thể trong thời gian chạy (xem Bảng 3).

Lưu ý thừa kế mà không đóng một phần trong mô hình Agent Gaia. Quan điểm của chúng tôi là “Agent là hệ thống tính toán thô, và một hệ thống Agent thường sẽ chỉ chứa một số lượng tương đối nhỏ các vai trò và các loại, với ánh xạ một-một giữa

chúng. Vì lý do này, chúng tôi tin rằng người thừa kế không hữu ích trong các thiết kế của loại Agent.

*4.2.* *Mô hình dịch vụ*

Như tên gọi của nó cho thấy, mục đích của các mô hình dịch vụ Gaia là để xác định các *dịch vụ* liên kết với mỗi vai trò của Agent và để xác định các tính chất của các dịch vụ này. Một dịch vụ, theo ý của chúng tôi là một chức năng (function) của Agent. Trong thuật ngữ hướng đối tượng (OO), một dịch vụ sẽ tương ứng với một phương thức; Tuy nhiên, chúng tôi không có ý là các dịch vụ có sẵn cho các Agent trong cùng cách mà các phương thức của một đối tượng có sẵn cho một đối tượng khác để gọi. Thay vào đó, một dịch vụ chỉ đơn giản là một khối thống nhất duy nhất của hoạt động trong đó một Agent sẽ tham gia. Rõ ràng, tất cả hoạt động được xác định trong giai đoạn phân tích sẽ tương ứng với một dịch vụ, mặc dù không phải mọi mọi dịch vụ sẽ tương ứng với một hoạt động.

Đối với mỗi dịch vụ có thể được thực hiện bởi một agent, cần minh chứng thuộc tính của nó. Cụ thể, chúng ta phải xác định các *yếu tố đầu vào,* đầu ra, *điều kiện tiên quyết,* và  *điều kiện* *hậu nghiệm* của từng dịch vụ. Đầu vào và đầu ra với các dịch vụ sẽ được bắt nguồn từ mô hình giao thức. Điều kiện tiên quyết và hậu quyết biểu diẽn ràng buộc của dịch vụ. Chúng được bắt nguồn từ các thuộc tính an toàn của một vai trò. Lưu ý rằng theo định nghĩa, mỗi vai trò sẽ được liên kết với ít nhất một dịch vụ.

Các dịch vụ mà Agent sẽ thực hiện được bắt nguồn từ danh sách các giao thức, hành động, nhiệm vụ và các thuộc tính thiết yếu của một vai trò. Trở lại ví dụ cà phê, có bốn hoạt động và giao thức liên kết với vai trò này: Fill, InformWorkers, CheckStock, và AwaitEmpty. Nói chung, sẽ có ít nhất một dịch vụ kết hợp với mỗi giao thức. Trong trương hợp CheckStock ví dụ, dịch vụ sẽ là đầu vào cấp độ chứng khoán và một số giá trị ngưỡng, và chỉ đơn giản là sẽ so sánh chúng.  điều kiện trước và sau sẽ chỉ ra stock level >0. Đây là một trong những đặc tính an toàn của vai trò COffeeFIller.

Các mô hình dịch vụ Gaia *không* quy định một cách thực hiện dịch vụ mà nó phát sinh. Các nhà phát triển có thể tự do thực hiện các dịch vụ trong bất kỳ khuôn khổ coi là thích hợp. Ví dụ, có thể được quyết định thực hiện các dịch vụ trực tiếp như

phương thức trong một ngôn ngữ hướng đối tượng. Ngoài ra, một dịch vụ có thể được phân tách ra thành một số phương thức.

*4.3.* *The Acquaintance Model*

Mô hình thiết kế Gaia cuối cùng có lẽ là đơn giản *nhất:* The Acquaintance Model*.* The Acquaintance Model chỉ đơn giản định nghĩa các liên kết truyền thông tồn tại giữa các loại Agent. Chúng *không* xác định nội dung thông điệp được gửi đi hoặc khi nào thì thông điệp được gửi đi - chỉ đơn giản là chỉ ra cách thức truyền thông. Đặc biệt, mục đích của một Acquaintance Model là xác định nút tắc nghẽn cổ chai trong truyền thông, trong đó có thể gây ra vấn đề run time (xem phần 5 cho một ví dụ). Nó là thực hành tốt để đảm bảo rằng hệ thống ghép nối lỏng lẻo, và The Acquaintance Model có thể giúp đỡ trong việc này. Trên cơ sở các The Acquaintance Model, có thể tìm thấy sự cần thiết để nhắc lại các giai đoạn phân tích và làm lại thiết kế hệ thống để loại bỏ vấn đề như vậy.

The Acquaintance Model đơn giản chỉ là một đồ thị, với các nút trong đồ thị tương ứng với các loại Agent và cung trong đồ thị tương ứng với con đường truyền thông. Agent Acquaintance Model được *vẽ trực tiếp*, và vì vậy một vòng cung *a 🡪 b* chỉ ra rằng *a* sẽ gửi tin nhắn cho *b, nhưng không nhất thiết rằng b* sẽ gửi tin nhắn đến a. Một The Acquaintance Model có thể bắt đầu trực tiếp từ vai trò, các giao thức, và các mô hình Agnet.

*4.4.* *Quá trình thiết kế*

Giai đoạn thiết kế Gaia bây giờ có thể được tóm tắt:

1. Tạo một *mô hình Agent*:

+ Tổng hợp vai trò thành *các loại Agent*, và tinh chỉnh để tạo thành một hệ thống phân cấp loại Agent;

+ Cung cấp các thực thể của từng loại Agent bằng cách sử dụng các chú thích.

2. Xây dựng một mô hình dịch vụ, bằng cách kiểm tra các hoạt động, các giao thức, thuộc tính thiết yếu và an toàn của vai trò (Roles).

3. Xây dựng acquaintance mode từ các mô hình tương tác và mô hình đại lý.

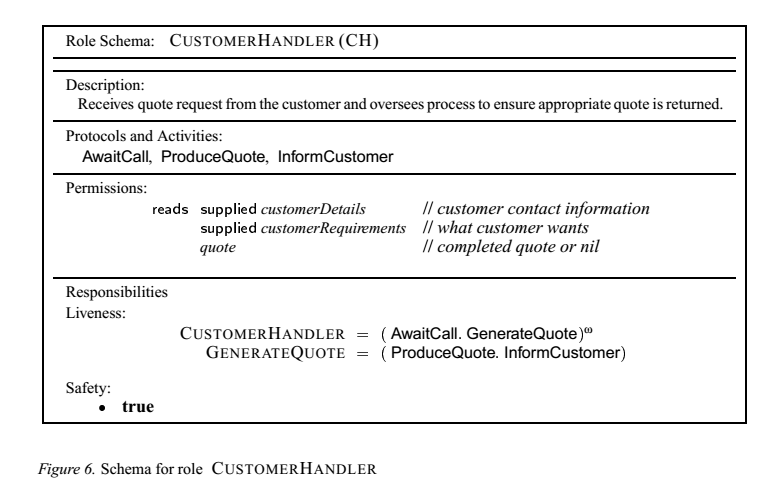
**5. Tình huống: Agent-Based Management Business Process**

Phần vắn tắt này minh hoạ cách áp dụng Gaia, thông qua một tình huống phân tích và thiết kế của một hệ thống dựa trên Agent quản lý quá trình kinh doanh của công ty British Telecom (xem [20] để biết thêm chi tiết). Vì lý do ngắn gọn và nhằm mục đích để cung cấp tư tưởng chung để phân tích và thiết kế chúng ta bỏ qua một số chi tiết.

Các ứng dụng cụ thể cung cấp cho khách hàng một báo giá lắp đặt mạng để cung cấp dịch vụ viễn thông. Hoạt động này bao gồm việc bộ phận sau: *bộ phận dịch vụ khách hàng* (CSD), *bộ phận thiết kế* (DD), *Bộ phận pháp chế* (LD) và nhiều tổ chức khác nhau cung cấp dịch vụ khác nằm ngoài kết quả khảo sát *khách hàng*. Quá trình này được khởi tạo bởi một khách hàng liên hệ với CSD với một tập các yêu cầu. Song song với việc nắm bắt các yêu cầu, các CSD thăm dò khách hàng. Nếu khách hàng không thỏa mãn các thủ tục, quá trình báo giá chấm dứt. Giả sử khách hàng là đạt yêu cầu, yêu cầu của họ được ánh xạ vào trong các danh mục dịch vụ. Nếu họ có thể được đáp ứng bởi một item chuẩn thì một báo giá có thể ngay lập tức được cung cấp.

Trong trường hợp các dịch vụ riêng biệt, tiến trình sẽ phức tạp hơn. DD bắt đầu để thiết kế giải pháp để đáp ứng các yêu cầu của khách hàng và trong khi điều này xảy ra thì LD kiểm tra tính hợp pháp của các dịch vụ được đề xuất. Nếu các dịch vụ mong muốn là bất hợp pháp, quá trình báo giá kết thúc. Giả sử các dịch vụ yêu cầu là hợp pháp, các thiết kế cuối cùng sẽ được hoàn thành và dự trù kinh phí. DD sau đó thông báo các báo giá cho CSD. CSD lần lượt thông báo cho khách hàng. Các quá trình kinh doanh sau đó chấm dứt.

Chuyển từ này mô tả quá trình định hướng cho hoạt động của hệ thống vào ngữ cảnh tổ chức là tương đối đơn giản. Trong nhiều trường hợp có một ánh xạ 1-1 giữa các phòng ban và vai trò. Hành vi của CSD rơi vào hai vai trò khác nhau: một giao diện cho khách hàng (CUSTOMERHANDLER, Hình 6), và một giám sát quá trình bên trong tổ chức (QUOTEMANAGER, Hình 7). Như vậy, VC, LD, và hành vi của DD tương ứng với các vai trò CUSTOMERVETTER(Hình 8), LEGALADVISOR (Hình 9), và NetworkDESIGNER (Hình 10). Vai trò cuối cùng là của CUSTOMER (Hình 11), người yêu cầu báo giá.



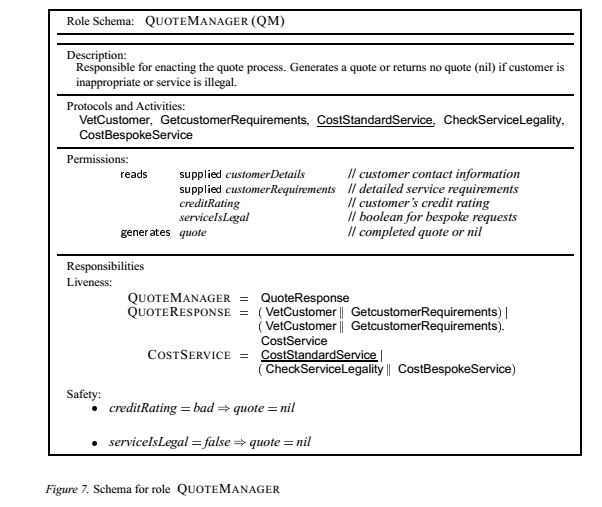
Với các định nghĩa vai trò tương ứng, giai đoạn tiếp theo là xác định các mô hình tương tác liên kết cho những vai trò này. Ở đây chúng ta tập trung vào tương tác liên kết với vai trò QUOTEMANAGER. Vai trò này tương tác với vai trò CUSTOMER để có được các yêu cầu của khách hàng (giao thức GetCustomerRequirements, hình12c) và với vai trò CUSTOMER VETTERđể xác định xem khách hàng có đạt yêu cầu không (giao thức VetCustomer, Hình 12a).

Nếu khách hàng không đạt yêu cầu, thì đây là hai giao thức duy nhất được ban hành. Nếu khách hàng là thỏa mãn thì yêu cầu của họ được dự trù kinh phí. Dự toán kinh phí bao gồm ban hành hành vi CostStandardService cho các dịch vụ thường xuyên được yêu cầu hoặc CheckServiceLegality (Hình 12b) và giao thức CostBespokeService (Hình 12d) cho các yêu cầu phi tiêu chuẩn.

Sau khi hoàn thành pha phân tích, chúng ta chuyển sang pha thiết kế. Mô hình đầu tiên được tạo ra là mô hình Agent (Hình 13). Điều này cho thấy, đối với hầu hết trường hợp, một sự tương ứng một-một giữa vai trò và các loại đại lý. Các ngoại lệ là đối với các vai trò CUSTOMER HANDLER và QUOTE MANAGER, do mức độ phụ thuộc lẫn nhau cao, được nhóm lại thành một loại tác nhân duy nhất.

Mô hình thứ hai là mô hình dịch vụ. Một lần nữa vì những hạn chế không gian chúng ta chú ý vào vai tro QUOTEMANAGER Customer Service Division Agent. Dựa trên vai trò QUOTEMANAGER, bảy dịch vụ có thể được xác định (Bảng 3). Từ giao thức GetCustomerRequirements, chúng ta lấy được các dịch vụ “obtain customer requirements”. Dịch vụ này xử lý các tương tác từ quan điểm của người quản lý báo giá. Nó nhận *CustomerDetails* là đầu vào và trả về *customerRequirements* như đầu ra (Hình 12c). Không có liên kết đến điều kiện tiền định, hậu định.

Dịch vụ liên kết với giao thức VetCustomer là “vet customer”. Đầu vào của nó là những *CustomerDetails* và kết quả đầu ra là creditRating. Dịch vụ này có một điều kiện tiên định là mà một phiếu tham dò khách hàng thích hợp phải luôn sẵn sàng

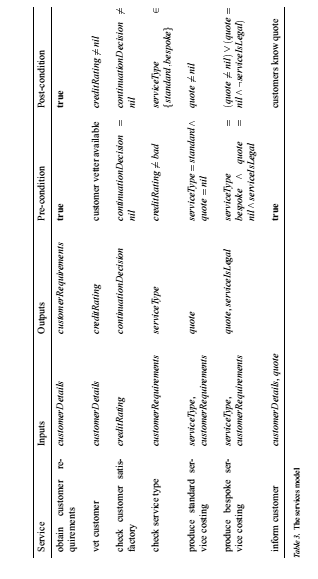


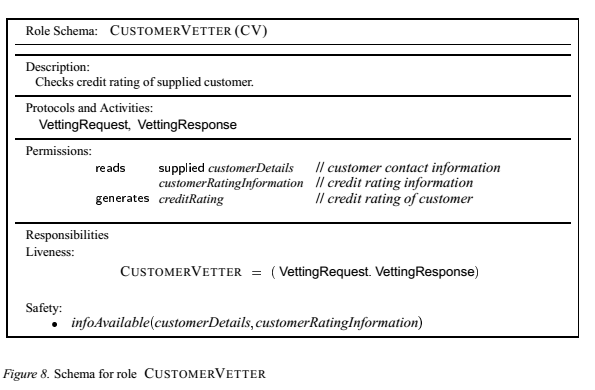
(lấy từ tương tác TenderContract trên giao thức VetCustomer) và một điều kiện hậu nghiệm là giá trị của *creditRating* là không null (vì đây là một phần điều kiện an toàn của vai trò QUOTEMANAGER).

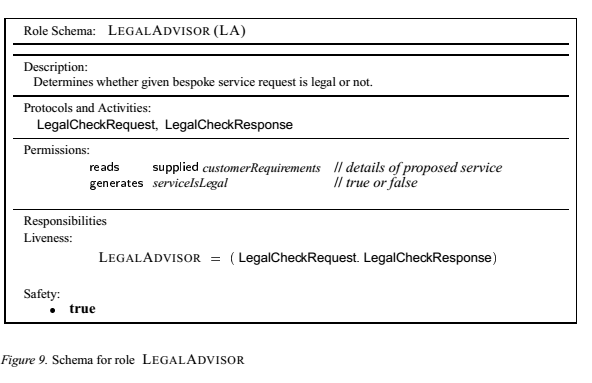
Các dịch vụ thứ ba bao gồm kiểm tra liệu khách hàng có đạt yêu cầu không (các *creditRating,* điều kiện an toàn của QUOTEMANAGER). Nếu khách hàng không đạt yêu cầu thì chỉ có nhánh đầu tiên của điều kiện thiết yếu QuoteRespose (Hình 7) được thực hiện. Nếu khách hàng là thỏa đáng, nhánh thiết yếu CostService được thực thi.

Các dịch vụ tiếp theo sẽ ra quyết định đường dẫn nào của CostService liveness expression được thực thi. Cả các dịch vụ loại tiêu chuẩn (thực hiện các dịch vụ “produce standard costing”) và dịch vụ riêng trong giao thức CheckServiceLegality và CostBespokeService được ban hành. Trong trường hợp sau, các giao thức liên kết với dịch vụ “produce bespoke costing”. Dịch vụ này tạo ra một giá trị không rỗng cho *giá* miễn là điều kiện an toàn *serviceIsLegal*  (Hình 7) không bị xâm phạm.

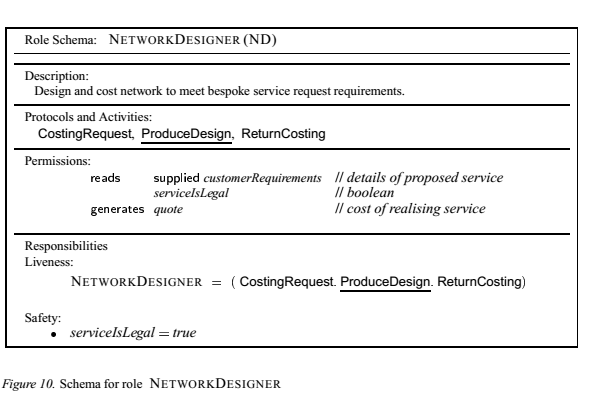
Các dịch vụ cuối cùng bao gồm thông báo cho các khách hàng các báo giá. Điều này, hoàn thành vai trò CUSTOMERHANDLER

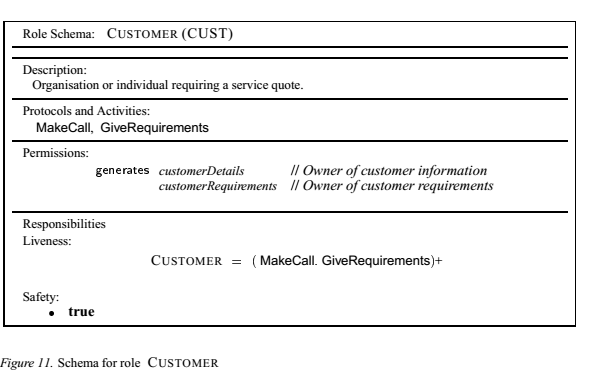




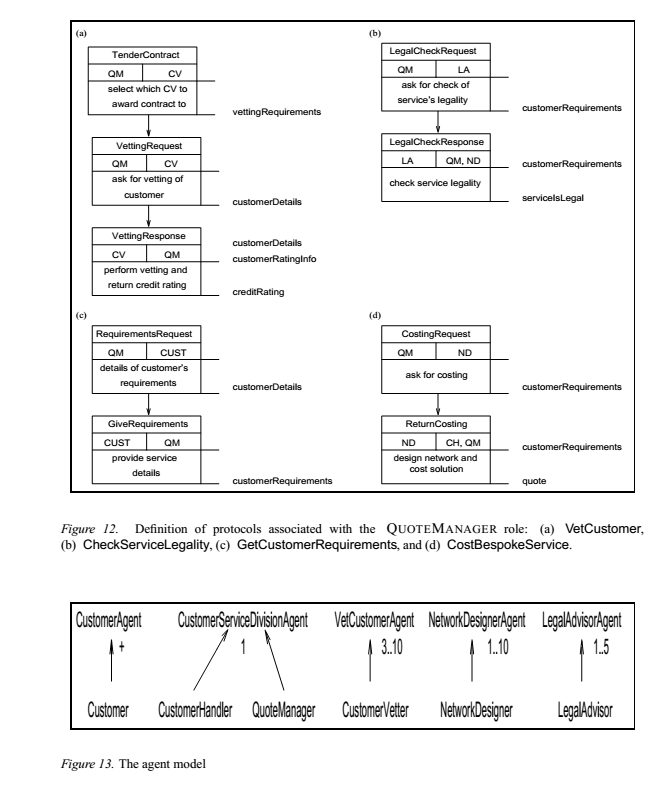


Các mô hình cuối cùng là acquaintance model, trong đó chỉ ra các đường giao tiếp giữa các Agent (Hình 14).

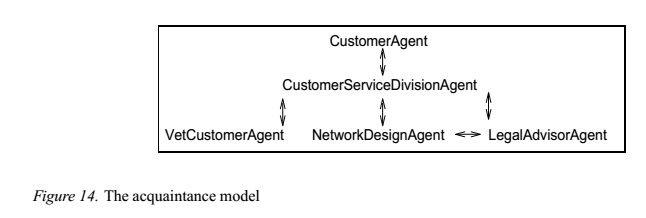


**

**6. Nghiên cứu liên quan**

Trong thời gian gần đây, đã gia tăng quan tâm trong công nghệ và phương pháp luận mô hình hướng Agent. Các phương pháp tiếp cận khác nhau có thể được tạm phân loại như sau: 

Phương pháp [4, 24] lấy kỹ thuật phương pháp luận mô hình hướng đối làm cơ sở, tìm kiếm hoặc để mở rộng mô hình và xác định các phương pháp cần sử dụng, hoặc là trực tiếp mở rộng khả năng ứng dụng của kỹ thuật và phương pháp luận OO, chẳng hạn như các thiết kế mẫu để thiết kế hệ thống Agent,



Phương pháp [3, 17] dựa trên việc mở rộng các kỹ thuật và phương pháp luận của mô hình công nghệ tri thức, cung cấp mẫu, mô hình hóa thành phần ngôn ngữ thích hợp cho các thẩm định cấu trúc và chức năng hệ thống.

Chúng lấy những phương pháp và ngôn ngữ mẫu như Z[31] và cung cấp định nghĩa trong một Framework mà hỗ trợ các đặc điểm kỹ thuật của Agent hoặc hệ thống Agent [26], Và chúng đã cơ bản được phát triển *de novo* đối với các loại cụ thể của hệ thống Agent. CASSIOPEIA [7], là một ví dụ, hỗ trợ các thiết kế hệ thống dựa trên Contract Net [29] áp dụng cho Robot đá bóng.

Những phương pháp thiết kế cũng có thể được phân chia dựa thành phương pháp tiếp cận từ trên xuống, dựa trên phân hủy dần các hành vi, xây dựng (như trong Gaia) trên một số khái niệm về vai trò, và CASSIOPEIA là phương pháp tiếp cận từ dưới lên mà bắt đầu bằng cách xác định hành vi Agent. Một cuộc khảo sát rất hữu ích về phân loại và đánh giá đã nêu ở [16].

Các định nghĩa và sử dụng các khái niệm khác nhau về vai trò, trách nhiệm, tương tác, nhóm và xã hội hoặc tổ chức trong các phương pháp cụ thể để phân tích và thiết kế theo định hướng Agent đã được thừa kế, chuyển thể từ việc sử dụng tổng quát hơn các khái niệm trong hệ thống đa agent, bao gồm phương pháp tiếp cận tập trung vào tổ chức như [14, 9, 18] và cách tiếp cận xã hội học như [5]. Tuy nhiên, nó vượt ra ngoài phạm vi của bài viết này.

Thay vào đó, chúng tôi sẽ tập trung vào mối quan hệ giữa Gaia và các phương pháp tiếp cận khác dựa trên kỹ thuật hướng đối tượng, đặc biệt là cách tiếp cận kgr [24, 23]. Nhưng nó có lẽ là hữu ích để bắt đầu bằng tóm tắt lý do tại sao kỹ thuật mô hình hướng đối tượng và phương pháp thiết kế chúng không thể trực tiếp áp dụng để thiết kế hệ thống đa agent.

*6.1.* *Thiếu sót của kỹ thuật hướng đối tượng*

Vấn đề đầu tiên liên quan đến mô hình của các Agent cá nhân hoặc lớp Agent. Trong khi có rất nhiều sự tương đồng giữa các Agent và đối tượng, Agent như một đối tượng, tức là như một tập hợp các thuộc tính và phương thức, không phải có ích khi biểu diễn Agent quá mịn, hoạt động ở một mức độ không phù hợp của trừu tượng hóa. Một biểu diễn Agent có thể khá kỳ lạ, có lẽ chỉ có một phương thưc public có chức năng nhận tin nhắn từ các Agent khác. Vì vậy, một mô hình đối tượng không nắm bắt được nhiều thông tin hữu ích về một Agent, và các khái niệm mạnh của OO

như kế thừa và tổng hợp trở nên khá vô dụng.

Có một số lý do cho vấn đề này. Một là mô hình Agent được dựa trên một khái niệm về tính đóng gói mạnh hơn so với mô hình đối tượng. Trạng thái nội bộ của một Agent là thường khá mờ.  Điều này là đặc trưng của tính tự chủ: Agent bình thường không thể được tạo ra và phá hủy một cách tự do trong hệ thống đối tượng và họ có tự do để xác định làm thế nào để phản hồi thông điệp, ví dụ, bằng cách thỏa thuận về cách một thực hiện nhiệm vụ. Theo mô hình giao tiếp cơ bản thường không đồng bộ, không có khái niệm xác định trước được luồng điều khiển từ một tác nhân đến tác nhân khác: một Agent có thể tự chủ khởi hành vi nội bộ hoặc ngoại vi bất cứ lúc nào, không chỉ khi nó được gửi một thông điệp. Cuối cùng, trạng thái nội bộ của Agent, bao gồm tri thức của nó, cần phải được thể hiện theo một cách mà có thể không dễ dàng được chuyển sang một tập các thuộc tính; trong mọi trường hợp làm như vậy sẽ tạo ra độ lệch trong giai đoạn tiền thi công.

Vấn đề thứ hai liên quan khả năng của mô hình đối tượng để nắm bắt đầy đủ mối quan hệ giữa các Agent trong một hệ thống đa agent. Trong khi các mô hình thứ cấp được sử dụng phổ biến trong phương pháp luận OO như tình huống và sơ đồ tương tác có thể hữu ích khi điều chỉnh (ngữ nghĩa khác nhau), Mô hình đối tượng, trong đó cấu thành đặc điểm kỹ thuật chính của một hệ thống OO là nắm bắt liên kết giữa các lớp đối tượng khiến cho mô hình phụ thuộc tĩnh tĩnh lớn và đường dẫn truy cập phần lớn là không thích hợp trong một hệ thống đa agent. Chỉ có các mối quan hệ giữa các lớp và thực thể có thể được áp dụng trực tiếp.

Các khía cạnh quan trọng trong mối quan hệ giữa các Agent như biểu diễn sự tương tác, mức độ kiểm soát hoặc ảnh hưởng đến cá nhân khác không dễ dàng để nắm bắt. Những vấn đề thiết yếu ở đây là tính thống nhất và tính chất tĩnh của mô hình hướng đối tượng. Một mô hình Agent đầy đủ cần phải nắm bắt các mối quan hệ giữa các Agent, bản chất động, và cả mối quan hệ giữa các Agent với các yếu tố phi Agent của hệ thống, bao gồm cả thụ động hay trừu tượng của những mô hình đều là tài nguyên.

Cả hai đều liên quan đến vấn đề sự phù hợp của mô hình kỹ thuật OO cho một mô hình hệ thống đa agent. Một vấn đề khác là việc áp dụng các phương pháp OO đến quá trình phân tích và thiết kế một hệ thống đa agent. Phương pháp OO thường bao gồm một chu kỳ lặp đi lặp lại trong việc xác định các class, xác định ngữ nghĩa và mối quan hệ của chúng, xây dựng giao diện và thực hiện. Ở cấp độ trừu tượng này, chúng xuất hiện tương tự phương thức hướng Agent (AO), phương pháp mà thường tiến hành bằng cách xác định vai trò và trách nhiệm và mục tiêu, phát triển một cơ cấu tổ chức, xây dựng tri thức và hành vi liên kết đến một vai trò hoặc Agent.

Tuy nhiên, sự tương đồng này biến mất ở mức độ chi tiết theo yêu cầu của các mô hình, như là những chìa khóa của tính trừu tượng chủ chốt là hoàn toàn khác nhau. Ví dụ, bước đầu tiên của nhận dạng lớp đối tượng là thường xem xét những điều hữu hình, vai trò, tổ chức, các sự kiện và thậm chí tương tác như các ứng viên, trong khi đó cần phải có sự phân biệt rõ ràng và xử lý khác nhau trong cách tiếp cận theo định hướng Agent.

Tính thống nhất và sự cụ thể của mô hình đối tượng là cơ sở của vấn đề; Phương pháp hướng đối tượng cung cấp sự hướng dẫn hay nguồn cảm hứng chứ không phải là một cách tiếp cận trực tiếp hữu ích để phân tích và thiết kế.

*6.2.* *So sánh với các cách tiếp cận KGR*

Các cách tiếp cận KGR [24, 23] được phát triển để đáp ứng nhu cầu có một cách tiếp cận nguyên tắc tới đặc điểm kỹ thuật của hệ thống đa agent phức tạp dựa trên niềm tin, khát vọng chủ định (BDI) công nghệ của Procedural Reasoning System (PRS) Và Distributed Multi-Agent Reasoning System (DMARS) [25, 8]. Một động lực quan trọng của nghiên cứu là đã cung cấp cơ chế hữu ích, quen thuộc cho việc cấu trúc và quản lý một hệ thống phức tạp.

Sự khác biệt đầu tiên và rõ ràng nhất giữa phương pháp tiếp cận được đề xuất ở đây và KG là phạm vi. Phương pháp của chúng tôi không cố gắng thống nhất việc phân tích và thiết kế trừu tượng của một hệ thống đa-agent với thiết kế cụ thể của mình và thực hiện với một công nghệ Agent đặc biệt, liên quan đến kết quả của quá trình phân tích và thiết kế là một đặc điểm kỹ thuật trừu tượng mà phương pháp thiết kế truyền thống ở mức thấp có thể được áp dụng. Ngược lại, KGR làm cho một kiến trúc liên kết mạnh mẽ thành kiến ​​trúc BDI và đề xuất một quy trình thiết kế, xây dựng trực tiếp tạo ra thông số kỹ thuật cho agent. Với sự gia tăng của công nghệ Agent có sẵn, có những lợi thế rõ ràng cho một cách tiếp cận tổng quát hơn, như đề xuất ở đây. Tuy nhiên, nhược điểm là nó không thể cung cấp một tập hợp các mô hình, trừu tượng và thuật ngữ mà có thể được sử dụng thống nhất trong cả vòng đời của hệ thống. Hơn nữa, có thể có một nhu cầu cho sự lặp lại của các quá trình phân tích và thiết kế AO. Nếu quá trình thiết kế cấp thấp hơn cho thấy vấn đề được tốt nhất giải quyết ở cấp AO. Vấn đề tìm kiếm cho cách tiếp cận của chúng tôi và những người khác như trên là làm cách đánh giá độc lập đầy đủ và trọn vẹn các kết quả.

Một khác biệt thứ hai là trong nghiên cứu này sự phân biệt rõ ràng được thực hiện giữa các giai đoạn phân tích, trong đó vai trò và mô hình tương tác được xây dựng hoàn toàn, và các giai đoạn thiết kế, trong đó Agent, dịch vụ và các mô hình acquaintance đang phát triển. Các cách tiếp cận KGR không làm ra sự phân biệt như vậy, đề xuất thay vì việc xây dựng tiến bộ và tinh tế của các mô hình Agent và mô hình tương tác tương ứng vai trò, các Agent và dịch vụ, các tương tác và những người quen biết. Trong khi cả hai phương pháp bắt đầu với việc xác định vai trò và tính chất của chúng, ở đây chúng tôi đã lựa chọn để mô hình Agent trừu tượng riêng (vai trò), Agent cụ thể và các dịch vụ mà chúng cung cấp. KGR thì ngược lại, sử dụng một mô hình Agent đồng bộ hơn, thừa nhận cả hai lớp Agent trừu tượng và lớp Agent cụ thể và cho phép chúng được tổ chức trong một hệ thống phân cấp thừa kế, do đó cho phép nhiều cấp độ trừu tượng và cách trì hoãn xác định các lớp Agent cụ thể cho đến cuối trong quá trình thiết kế.

Trong khi cả hai cách tiếp cận sử dụng trách nhiệm như là một sự trừu tượng được sử dụng để phân hủy các cấu trúc của một vai trò, chúng vẫn khác nhau đáng kể về cách thức này được biểu diễn và phát triển. Ở đây trách nhiệm bao gồm các thuộc tính an toàn và thiết yếu được xây dựng lên từ các tương tác và các hoạt động đã được xác định. Ngược lại, KGR xử lý trách nhiệm như là một mục tiêu trừu tượng, kích hoạt bởi các sự kiện hay các tương tác và thông qua một cách tiếp cận nghiêm ngặt từ trên xuống để phân hủy các thành dịch vụ và mục tiêu ở cấp thấp để các thông số kỹ thuật về hoạt động có thể xây dựng được. Có những điểm tương tự, tuy nhiên, cho dù không có mục tiêu rõ ràng trong cách tiếp cận của chúng tôi, thuộc tính an toàn có thể được xem như là mục tiêu duy trì và thuộc tính thiết yếu như các mục tiêu về thành tích. Tuy nhiên các khái niệm về quyền truy cập, chúng vắng mặt phương pháp tiếp cận KGR, trong khi khái niệm về giao thức có thể được phát triển đến một mức độ lớn hơn nhiều. Ví dụ như trong [22]. Các giao thức được sử dụng như nhiều mô tả chung chung của hành vi mà có thể bao gồm các thực thực thể không được mô hình hóa như một Agetn, chẳng hạn như máy cà phê.

Để tóm tắt những khác biệt chủ chốt, phương pháp tiếp cận KGR bằng cách tạo ra một cam kết để thực hiện với một kiến ​​trúc agent BDI, có thể sử dụng một phương pháp tiếp cận từ trên xuống lặp đi lặp lại để xây dựng một tập hợp các mô hình mô tả một hệ thống đa agent ở cả tầm vi mô và vĩ mô, để làm phong phú hơn sử dụng các kỹ thuật mô hình OO, và để sản xuất chi tiết kỹ thuật thực thi như là kết quả cuối cùng. Các cách tiếp cận chúng tôi đã mô tả ở đây là một tiếp cận hỗn hợp từ trên xuống và từ dưới lên trong đó sử dụng một hoạch nhiều tập mịn và đa dạng của các mô hình chung để nắm bắt các kết quả quá trình phân tích và thiết kế, và cố gắng tránh bất kỳ cam kết sớm, hoặc là kiến ​​trúc, hoặc như các chi tiết thiết kế và thực hiện. Tuy nhiên, chúng tôi dự tính cách tiếp cận của chúng tôi có thể phù hợp với chuyên ngành kiến ​​trúc cụ thể cho Agent hoặc kỹ thuật thực hiện; đây là một chủ đề để nghiên cứu thêm.

**7. Kết luận và hướng phát triển**

Trong bài viết này, chúng tôi đã mô tả Gaia, một phương pháp luận cho việc phân tích và thiết kế các hệ thống hướng Agent. Khái niệm chính trong Gaia là vai trò, trong đó có liên quan đến trách nhiệm, quyền, hoạt độn, và các giao thức. Vai trò có thể tương tác với nhau trong một số cách, được định nghĩa trong các giao thức của các vai trò tương ứng.

Có một số vấn đề còn lại cho công việc tương lai.

*Agent tự quan*tâm.

Gaia không giải quyết được với các hệ thống trong đó các Agent không chia sẻ chung mục tiêu. Lớp hệ thống này là đại diện cho ứng dụng quan trọng nhất đối với hệ thống đa Agent và vì thế điều cần thiết là một phương pháp giải quyết nó.

*Hệ*thống *năng động và mở*.

Hệ thống mở - là hệ thống mà trong đó các thành phần hệ thống có thể tham gia và tách rời trong thời gian chạy, và có thể bao gồm các thực thể mà một nhà thiết kế không có kiến ​​thức về thiết kế tại thời gian thiết kế - Vấn đề này từ lâu đã công nhận là một lớp hệ thống mà rất khó khăn để thiết kế [15, 13].

*Cơ cấu tổ*chức.

Một khía cạnh khác của phân tích thiết kế hướng Agent đòi hỏi phải nghiên cứu nhiều hơn là quan điểm của một cơ cấu tổ chức. Tại thời điểm này, công trình này chỉ ược *ngầm* định trong Gaia- Trong các vai trò và mô hình tương tác. Tuy nhiên, trực tiếp, biểu diễn rõ ràng của cấu trúc như vậy sẽ có giá trị cho nhiều ứng dụng. Ví dụ, nếu các Agentđược sử dụng để mô hình các tổ chức lớn, các tổ chức này sau đó sẽ có một cấu trúc được định nghĩa một cách rõ ràng. Cấu trúc Agent như vậy có thể là cách duy nhất để nắm bắt đầy đủ và hiểu biết tổ chức kết nối và kiểm soát cấu trúc. Nói chung, sự phát triển của *các mẫu thiết kế* *tổ chức* có thể có ích cho việc tái sử dụng các cấu trúc hệ thống đa agent thành công ([12]).

*Giao thức phối hợp Cooperation Protocols*.

Các biểu diến của các giao thức hợp tác liên Agent trong Gaia hiện còn nghèo nà. Các nghiên cứu trong tương lai, chúng tôi sẽ cần phải cung cấp một đặc tả giao thức phong phú hơn nhiều khung công việc.

*Tiêu chuẩn quốc*tế.

Gaia không được thiết kế với bất kỳ tiêu chuẩn cụ thể cho truyền thông Agent (ví dụ như các ngôn ngữ giao tiếp Agent FIPA [11]). Tuy nhiên, trong trường hợp các tiểu chuẩn công nghiệp được phổ biến, nó có thể hữu ích để áp dụng phương pháp của chúng tôi để tương thích với các tiêu chuẩn đó.

Khuôn mẫu *Ngữ nghĩa*.

Cuối cùng, chúng tôi tin rằng một phương pháp thành công, không những chỉ có giá trị thực dụng, mà còn xác định, ngữ nghĩa chính thức rõ ràng. Trong khi các nhà phát triển điển hình thậm chí không bao giờ nhận thức được sự hiện hữu của một ngữ nghĩa, nó vẫn là điều cần thiết để có một sự hiểu biết chính xác về những gì các khái niệm và thuật ngữ trong một phương pháp muốn để cập [33].