Đại Học Quốc Gia Thành Phố Hồ Chí Minh
Trường Đại Học Công Nghệ Thông Tin

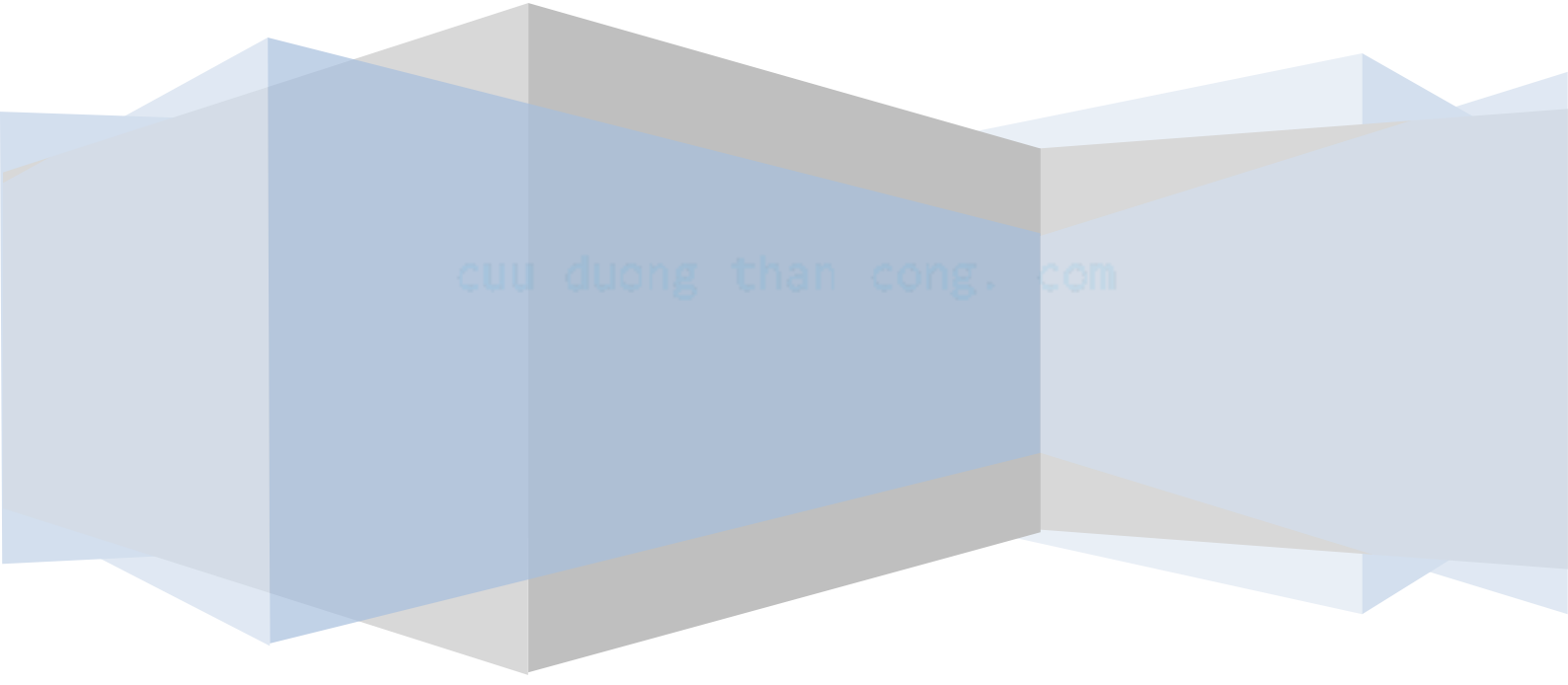
Tổng Quan Về Ontology

Kỹ Thuật Lập Trình Trình Trí Tuệ Nhân Tạo

Lớp: CNTN02

GVHD: TS. Nguyễn Tuấn Đăng

SVTH: Nguyễn Hữu Nhật – MSSV: 07520437



Mục Lục

1. Tổng quan về Ontology:

Thuật ngữ "Ontology" đã xuất hiện từ rất sớm. Trong cuốn sách "Siêu hình" (Metaphysics) của mình, Aristotle đã định nghĩa: "Ontology là một nhánh của triết học, liên quan đến sự tồn tại và bản chất các sự vật trong thực tế". Hay nói cách khác, đối tượng nghiên cứu chủ yếu của Ontology xoay quanh việc phân loại các sự vật dựa trên các đặc điểm mang tính bản chất của nó. Ontology là một thuật ngữ mượn từ triết học được tạm dịch là "bản thể học", nhằm chỉ khoa học mô tả các loại thực thể trong thế giới thực và cách chúng liên kết với nhau.

Trong ngành khoa học máy tính và khoa học thông tin, Ontology mang ý nghĩa là các loại vật và quan hệ giữa chúng trong một hệ thống hay ngữ cảnh cần quan tâm. Các loại vật này còn được gọi là khái niệm, thuật ngữ hay từ vựng có thể được sử dụng trong một lĩnh vực chuyên môn nào đó. Ontology cũng có thể hiểu là một ngôn ngữ hay một tập các quy tắc được dùng để xây dựng một hệ thống Ontology. Một hệ thống Ontology định nghĩa một tập các từ vựng mang tính phổ biến trong lĩnh vực chuyên môn nào đó và quan hệ giữa chúng. Sự định nghĩa này có thể được hiểu bởi cả con người lẫn máy tính. Một cách khái quát, có thể hiểu Ontology là "một biểu diễn của sự khái niệm hoá chung được chia sẻ" của một miền hay lĩnh vực nhất định. Nó cung cấp một bộ từ vựng chung bao gồm các khái niệm, các thuộc tính quan trọng và các định nghĩa về các khái niệm và các thuộc tính này. Ngoài bộ từ vựng, Ontology còn cung cấp các ràng buộc, đôi khi các ràng buộc này được coi như các giả định cơ sở về ý nghĩa mong muốn của bộ từ vựng, nó được sử dụng trong một lĩnh vực mà có thể được giao tiếp giữa người và các hệ thống ứng dụng phân tán khác.

Ontology được sử dụng trong trí tuệ nhân tạo, công nghệ Web ngữ nghĩa (Semantic Web), các hệ thống kỹ thuật, kỹ thuật phần mềm, tin học y sinh và kiến trúc thông tin như là một hình thức biểu diễn tri thức về thế giới hoặc một số lĩnh vực cụ thể. Việc tạo ra các lĩnh vực về Ontology cũng là cơ sở để định nghĩa và sử dụng của cơ cấu một tổ chức kiến trúc(an enterprise architecture framework).

2. Thành phần của một Ontology:

Các thành phần thường gặp của Ontology bao gồm:

- Các cá thể(individuals): các thực thể hoặc các đối tượng (các đối tượng cơ bản hoặc cấp độ nền).
- Các lớp(classes): các tập hợp, các bộ sưu tập, các khái niệm, các loại đối tượng, hoặc các loại khác.
- Các thuộc tính(attributes): các khía cạnh, đặc tính, tính năng, đặc điểm, hoặc các thông số mà các đối tượng (và các lớp) có thể có.
- Các quan hệ(relations): cách thức mà các lớp và các cá thể có thể liên kết với nhau.
- Các thuật ngữ chức năng(function terms): cấu trúc phức tạp được hình thành từ các mối quan hệ nhất định có thể được sử dụng thay cho một thuật ngữ cá thể trong một báo cáo(statement).
- Các sự hạn chế(restrictions): những mô tả chính thức được tuyên bố về những điều phải chính xác cho một số khẳng định được chấp nhận ở đầu vào.
- Các quy tắc(rules): tuyên bố có hình thức như một cặp nếu-thì (if-then) mô tả suy luận logic có thể được rút ra từ một sự khẳng định trong từng hình thức riêng.
- Các tiên đề(axioms): các khẳng định (bao gồm các quy tắc) trong một hình thức hợp lý với nhau bao gồm các lý thuyết tổng thể mà ontology mô tả trong lĩnh vực của ứng dụng.

- Các sự kiện(events): sự thay đổi các thuộc tính hoặc các mối quan hệ.

Sau đây chúng ta sẽ tìm hiểu một số thành phần quan trọng nhất của một Ontology.

2.1 Các cá thể (Individuals):

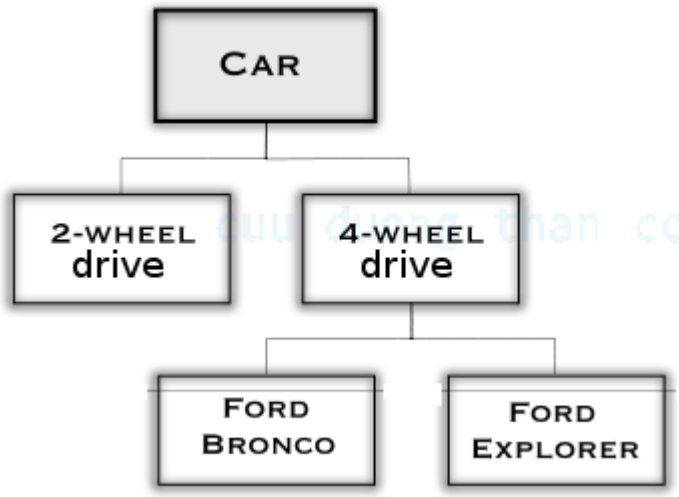
Các cá thể là các thành phần cơ bản, nền tảng của một ontology. Các cá thể trong một ontology có thể bao gồm các đối tượng cụ thể như con người, động vật, cái bàn… cũng như các cá thể trừu tượng như các thành viên hay các từ. Một ontology có thể không cần bất kỳ một cá thể nào, nhưng một trong những lý do chính của một ontology là để cung cấp một ngữ nghĩa của việc phân lớp các cá thể, mặc dù các cá thể này không thực sự là một phần của ontology.

2.2 Các lớp (Classes):

Các lớp là các nhóm, tập hợp các đối tượng trừu tượng. Chúng có thể chứa các cá thể, các lớp khác, hay là sự phối hợp của cả hai. Một số ví dụ về các lớp học:

- *Người (person),* các lớp của tất cả mọi người, hay đối tượng trừu tượng có thể được mô tả bởi các tiêu chí để trở thành một con người.
- *Phương tiện xe cộ (vehicle),* các lớp của tất cả các phương tiện xe cộ, hay đối tượng trừu tượng có thể được mô tả bởi các tiêu chí để được một phương tiện xe cộ.
- *Xe hơi (cars),* các lớp của tất cả các xe ô tô, hoặc đối tượng trừu tượng có thể được mô tả bởi các tiêu chí để được một chiếc xe hơi.
- *Lớp học (class),* đại diện cho lớp của tất cả các lớp học, hay đối tượng trừu tượng có thể được mô tả bởi các tiêu chí để được một lớp học.
- *Đồ vật (Thing),* đại diện cho lớp của tất cả mọi đồ vật, hoặc đối tượng trừu tượng có thể được mô tả bởi các tiêu chí để được đồ vật.

Các ontology biến đổi tuỳ thuộc vào cấu trúc và nội dung của nó: Một lớp có thể chứa các lớp con, có thể là một lớp tổng quan (chứa tất cả mọi thứ), có thể là lớp chỉ chứa những cá thể riêng lẻ, Một lớp có thể xếp gộp vào hoặc bị xếp gộp vào bởi các lớp khác. Mối quan hệ xếp gộp này được sử dụng để tạo ra một cấu trúc có thứ bậc các lớp, thường là với một lớp thông dụng nhất ở trên đỉnh và các lớp có kiểu rõ ràng cụ thể ở phía dưới cùng. Ví dụ mô hình phân lớp xe hơi như hình sau:



Hình – Mô hình phân lớp xe hơi.

Nếu quy tắc phân vùng đảm bảo rằng một *xe* duy nhất không thể ở cả hai lớp, các phân vùng này được gọi là một phân vùng phân chia. Nếu các quy tắc phân vùng đảm bảo rằng mỗi đối tượng cụ thể trong siêu lớp là một thể hiện của ít nhất một trong các lớp phân vùng, các phân vùng này được gọi là một phân vùng toàn bộ.

2.3 Các thuộc tính (Attributes):

Các đối tượng trong ontology có thể được mô tả thông qua việc khai báo các thuộc tính của chúng. Mỗi một thuộc tính đều có tên và giá trị của thuộc tính đó. Các thuộc tính được sử dụng để lưu trữ các thông tin mà đối tượng có thể có. Ví dụ, đối với một cá nhân có thể có các thuộc tính: Họ tên, ngày sinh, quê quán, số CMND… Giá trị của một thuộc tính có thể là một kiểu dữ liệu phức tạp.

2.4 Các mối quan hệ (Relationships):

Mối quan hệ (còn gọi là quan hệ) giữa các đối tượng trong ontology định rõ như thế nào các đối tượng này có liên quan đến các đối tượng khác. Đặc trưng là một mối quan hệ loại riêng biệt (hay lớp) mà quy định cụ thể trong chiều hướng các đối tượng này có liên quan đến các đối tượng khác trong ontology.

Chủ yếu sức mạnh của ontology đến từ khả năng mô tả các mối quan hệ. Cùng với nhau, tập hợp các mối quan hệ mô tả ngữ nghĩa trong một lĩnh vực nào đó. Các thiết lập của các loại quan hệ được sử dụng(các lớp của các quan hệ) và hệ thống phân cấp của nó mô tả sức mạnh biểu hiện của ngôn ngữ trong đó ontology được thể hiện.

Một kiểu quan hệ quan trọng là kiểu quan hệ xếp gộp (subsumption). Kiểu quan hệ này mô tả các đối tượng nào là các thành viên của các lớp nào của các đối tượng. Ontology có thể phân biệt giữa các loại khác nhau của các quan hệ.

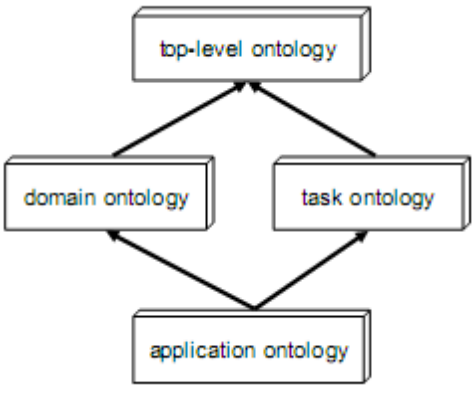
Ví dụ:

- loại quan hệ dành cho các quan hệ giữa các lớp
- loại quan hệ dành cho các quan hệ giữa các cá thể
- loại quan hệ dành cho các quan hệ giữa một cá thể và một lớp
- loại quan hệ dành cho các quan hệ giữa một đối tượng đơn lẻ và một bộ sưu tập (collection)
- loại quan hệ dành cho các quan hệ giữa các bộ sưu tập

Loại quan hệ đôi khi là một lĩnh đặc trưng và sau đó được dùng để lưu trữ các loại đặc trưng của sự kiện hoặc trả lời từng loại câu hỏi riêng biệt. Nếu các định nghĩa của các loại quan hệ được bao gồm trong một ontology, sau đó ontology định nghĩa riêng ngôn ngữ ontology của nó.

3. Phân loại Ontology:

Từ khi bắt đầu nghiên cứu về Ontology trong Khoa học máy tính, Ontology đã được quan tâm với mong muốn tăng cường việc sử dụng lại hệ thống cơ sở tri thức bên trong, và nó cũng đưa ra nhiều dạng Ontology mô tả khả năng khác nhau cho việc tái sử dụng hệ thống cơ sở tri thức. Sự phân loại Ontology có thể được tạo ra theo chủ đề



Hình – Phân loại Onlotogy.

của sự khái niệm hóa.Trong khía cạnh này, phần cốt lõi nhất được tổng kết theo hình bên cạnh.

- *Top-level Ontology:* còn gọi là Ontology lớp cao, nhằm diễn tả những khái niệm tổng quan và trừu tượng có thể được chia sẻ qua nhiều lĩnh vực và ứng dụng. Nó mượn các ý niệm triết học mô tả những khái niệm lớp cao cho mọi vật về sự tồn tại của chúng, như đối tượng vật chất hay đối tượng trừu tượng như là các ý niệm có đặc điểm chung về tri thức nhận thức thông thường về hiện tượng như thời gian, không gian, các tiến trình ... Do sự tổng quan của đó, nó không sử dụng trực tiếp trong các ứng dụng, mà thông qua các Ontology khác.
- *Domain Ontology và task Ontology:* các loại Ontology này lấy tri thức từ trong những lĩnh vực xác định, như trong y khoa, địa lý hay tri thức về một tác vụ riêng biệt như sự chẩn hoặc sự cấu hình. Về mặt ý tưởng thì Ontology loại này thu hẹp hơn và xác định hơn so với Top-level Ontology. Sự khái niệm hóa trong một Domain Ontology là giữ các tác vụ độc lập, khi những ý niệm trong một tác vụ Ontology được miêu tả không có tính chất rõ rệt với chi tiết cụ thể đến một lĩnh vực. Sự phát triển của Domain Ontology được thực hiện nhiều ở các lĩnh vực: y học, di truyền, địa lý, du lịch, thông tin môi trường. Còn Task Ontology được phát minh cho các tác vụ xây dựng, sắp xếp kế hoạch làm việc, giám sát trong một lĩnh vực khoa học, cơ sở tri thức máy tính dạy học, sự theo dõi phóng tên lửa, các tác vụ hướng dẫn điều trị bệnh ...
- *Application Ontology:* càng thu hẹp về phạm vi, Application Ontology cung cập một bộ từ vựng xác định được yêu cầu để mô tả sự ban hành các tác vụ chắc chắn trong một ngữ cảnh ứng dụng cụ thể. Đặc biệt, nó sử dụng cả Domain Ontology và Task Ontology và mô tả vai trò của chúng trong một tác cụ thể.

Chúng ta có thể thấy hệ thống phân cấp của Ontology thông qua sự trình bày ở trên: Ontology ở lớp thấp hơn kế thừa và chuyên môn hóa các khái niệm và mối quan hệ từ Ontology lớp trên. Ontology lớp thấp cụ thể hơn và phạm vi ứng dụng thu hẹp hơn, còn Ontology ở lớp cao có khả năng rộng hơn, chủ yếu dành cho việc kế thừa và sử dụng lại.

4. Phương thức xây dựng Ontology:

Có nhiều phương pháp khác nhau để xây dựng một Ontology, nhưng nhìn chung các phương pháp đều thực hiện hai bước cơ bản là: xây dựng cấu trúc lớp phân cấp và định nghĩa các thuộc tính cho lớp. Trong thực tế, việc phát triển một Ontology để mô tả lĩnh vực cần quan tâm là một công việc không đơn giản, phụ thuộc rất nhiều vào công cụ sử dụng, tính chất, quy mô, sự thường xuyên biến đổi của miền cũng như các quan hệ phức tạp trong đó. Những khó khăn này đòi hỏi công việc xây dựng Ontology phải là một quá trình lặp đi lặp lại, mỗi lần lặp cải thiện, tinh chế và phát triển dần sản phẩm chứ không phải là một quy trình khung với các công đoạn tách rời nhau. Công việc xây dựng Ontology cũng cần phải tính đến khả năng mở rộng lĩnh vực quan tâm trong tương lai, khả năng kế thừa các hệ thống Ontology có sẵn, cũng như tínhlinh động để Ontology có khả năng mô tả tốt nhất các quan hệ phức tạp trong thế giới thực.

Một số nguyên tắc cơ bản của việc xây dựng Ontology thông qua các công đoạn sau đây:

- Xác định miền quan tâm và phạm vi của Ontology.
- Xem xét việc kế thừa các Ontology có sẵn.
- Liệt kê các thuật ngữ quan trọng trong Ontology.
- Xây dựng các lớp và cấu trúc lớp phân cấp.

- Định nghĩa các thuộc tính và quan hệ cho lớp.
- Định nghĩa các ràng buộc về thuộc tính và quan hệ của lớp.
- Tạo các thực thể cho lớp.

4.1 Xác định lĩnh vực quan tâm và phạm vi của Ontology:

Thông thường, các yêu cầu đối với một hệ thống Ontology là mô tả lĩnh vực quan tâm nhằm phục vụ cơ sở tri thức trong việc giải quyết những mục đích chuyên biệt. Công việc đặc tả để xác định, phân tích, nhận diện chính xác yêu cầu được thực hiện bằng cách trả lời những câu hỏi sau:

- Ontology cần mô tả lĩnh vực nào?
- Ontology phục vụ cho mục đích chuyên biệt gì?
- Cơ sở tri thức trong Ontology sẽ giải quyết những câu hỏi gì?
- Ontology nhằm vục vụ đối tượng nào?
- Ai là người sẽ xây dựng, quản trị Ontology?

Nhìn chung, câu trả lời cho các câu hỏi dạng này có thể sẽ thường xuyên thay đổi trong suốt quá trình xây dựng một Ontology. Nhất là khi có sự thay đổi về mục đích hoặc cần bổ sung tính năng trong việc sử dụng cơ sở tri thức. Tuy nhiên, việc trả lời chính xác các câu hỏi trên tại mỗi bước lặp sẽ giúp giới hạn phạm vi của mô hình cần mô tả và dự trù các kỹ thuật sẽ sử dụng trong quá trình phát triển. Lấy ví dụ, nếu dự trù khả năng xảy ra sự khác biệt về ngôn ngữ giữa người phát triển và người sử dụng thì Ontology phải được bổ sung cơ chế ánh xạ (mapping)qua lại các thuật ngữ giữa các ngôn ngữ khác nhau. Hoặc giả sử Ontology cần xây dựng có chức năng xử lý ngôn ngữ tự nhiên, ứng dụng dịch tài liệu tự động thì cũng cần thiết phải có kỹ thuật xác định từ đồng nghĩa...

Sau khi đã phát thảo phạm vi Ontology dựa trên việc trả lời những câu hỏi trên, người thiết kế sẽ trả lời các câu hỏi mang tính đánh giá, qua đó tiếp tục tinh chỉnh lại phạm vi của hệ thống cần xây dựng. Các câu hỏi dạng này thường dựa trên cơ sở tri thức của Ontology và được gọi là câu hỏi kiểm chứng khả năng (competency question):

- Ontology đã có đủ thông tin để trả lời cho các câu hỏi được quan tâm trên cơ sở tri thức hay không?
- Câu trả lời của cơ sở tri thức đã đáp ứng được mức độ, yêu cầu nào của người sử dụng?
- Các ràng buộc và quan hệ phức tạp trong miền quan tâm đã được biểu diễn hợp lý chưa

4.2 Xem xét việc kế thừa các Ontology có sẵn:

Đây là một công đoạn thường hay sử dụng để giảm thiểu công sức xây dựng một Ontology. Bằng cách kế thừa các Ontology tương tự có sẵn, người xây dựng có thể thêm hoặc bớt các lớp, quan hệ giữa các lớp, thực thể... để tinh chỉnh tùy theo mục đích của mình. Việc sử dụng lại các Ontology có sẵn cũng rất quan trọng khi cần sự tương tác giữa các ứng dụng khác nhau, các ứng dụng sẽ cần phải hiểu các lớp, thực thể, quan hệ... của nhau để thuận tiện trong việc trao đổi hoặc thống nhất thông tin.

Vấn đề xây dựng một Ontology mới bằng cách kế thừa các hệ thống có sẵn liên quan đến một bài toán rất phức tạp là trộn (merging)các Ontology. Ví dụ trong trường hợp tên các khái niệm được định nghĩa trong các Ontology này có thể giống nhau trong khi chúng được dùng để mô tả các loại vật hoàn toàn khác nhau, và cũng có thể xảy ra trường hợp ngược lại, khi tên các khái niệm khác nhau nhưng cùng mô tả một sự vật, vàmột vấn đề nữa là làm thế nào để bổ sung các quan hệ, thuộc tính có sẵn vào một hệ thống mới. Tuy nhiên, hầu hết các Ontology sử dụng trong ngành khoa học máy tính nói chung và Web ngữ nghĩa nói riêng đều

được xây dựng trên các hệ thống xây dựng và quản trị Ontology. Tên một số công cụ như: Sesame, Protégé, Ontolingua, Chimaera, OntoEdit, OidEd... Hiện nay, đa số các phần mềm này đều hỗ trợ chức năng tự động trộn các Ontology cùng hoặc thậm chí khác định dạng với nhau. Mặc dù vậy, ở mức nào đó, người xây dựng cũng cần phải kiểm tra lại một cách thủ công, nhưng đây không phải là một công việc phức tạp.

Hiện có rất nhiều Ontology được chia sẻ trên Web nổi tiếng như: UNSPSC (www.unspsc.org)do chương trình phát triển của Liên Hiệp Quốc hợp tác với tổ chức Dun & Bradstreet nhằm cung cấp các thuật ngữ của các sản phẩm và dịch vụ thương mại. Các Ontology trong lĩnh vực thương mại khác như: RosettaNet (www.rosettanet.org), DMOZ (www.dmoz.org), eClassOwl... Open Biological, BioPax trong lĩnh vực sinh vật học, UMLS trong lĩnh vực mạng ngữ nghĩa, GO (Gene Ontology), WordNet (đại học Princeton)...

4.3 Liệt kê các thuật ngữ quan trọng trong Ontology:

Đây là bước rất hữu ích, làm tiền đề cho hai bước tiếp theo là xây dựng cấu trúc lớp phân cấp và định nghĩa các thuộc tính cho lớp. Công đoạn này bắt đầu bằng việc liệt kê tất cả các thuật ngữ xuất hiện trong miền quan tâm (có thể đồng nghĩa hoặc chồng nhau)như tên khái niệm, quan hệ, thuộc tính... Thông thường, các thuật ngữ là danh từ sẽ trở thành các lớp, tính từ sẽ trở thành thuộc tính, còn động từ sẽ là quan hệ giữa các lớp.

4.4 Xây dựng các lớp và cấu trúc lớp phân cấp:

Đây là một trong hai bước quan trọng nhất của công việc xây dựng một Ontology. Nhiệm vụ của bước này là định nghĩa các lớp từ một số thuật ngữ đã liệt kê trong bước trên, sau đó xây dựng cấu trúc lớp phân cấp theo quan hệ lớp cha-lớp con. Lớp ở vị trí càng cao sẽ có mức độ tổng quát càng cao. Vị trí đầu tiên thuộc về lớp gốc, tiếp theo là các lớp trung gian, và cuối cùng là lớp lá. Lớp lá là lớp không thể triển khai được nữa và chỉ được biểu hiện bằng các thực thể.

Hình - Cấu trúc lớp phân cấp

Quan hệ giữa thực thể của lớp con với lớp cha là quan hệ "is-a", nghĩa là một thực thể của lớp con cũng "là-một" thực thể của lớp cha. Có nhiều hướng tiếp cận khác nhau cho vấn đề xây dựng cấu trúc lớp phân cấp. Có thể kể ra ba hướng như sau:

- Hướng xây dựng từ trên xuống (top-down): bắt đầu bằng các lớp có mức độ tổng quát cao nhất, sau đó triển khai dần đến lớp lá.
- Hướng xây dựng từ dưới lên (bottom-up): ngược với hướng xây dựng cấu trúc lớp phân cấp từ trên xuống, hướng này bắt đầu bằng việc xác định các lớp được cho là cụ thể nhất, sau đó tổng quát hóa đến khi được lớp gốc.
- Cách kết hợp (combination): cách này kết hợp cả hai hướng xây dựng trên. Đầu tiên chọn các lớp nổi bật nhất trong lĩnh vực quan tâm, sau đó tổng quát hóa và cụ thể hóa cho đến khi được cấu trúc mong muốn.

4.5 Định nghĩa các thuộc tính và quan hệ cho lớp:

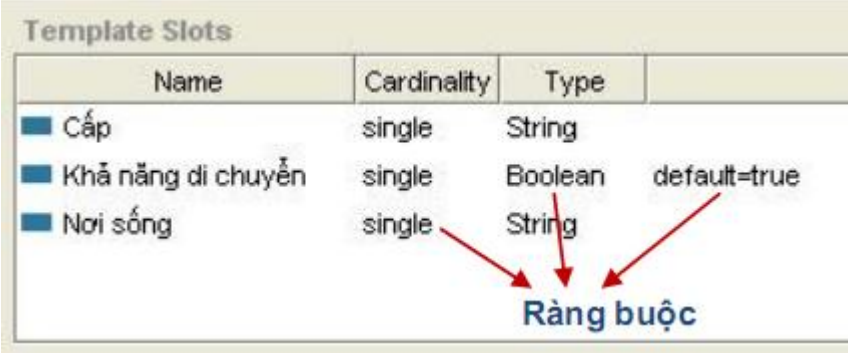
Bản thân các lớp nhận được ở bước trên chỉ mới là những thuật ngữ phân biệt với nhau bằng tên gọi. Về cơ bản, chúng chưa đủ để phục vụ cho việc biểu diễn tri thức. Muốn như vậy, các thuộc tính của lớp cần được định nghĩa. Thuộc tính của lớp là các thông tin bên trong của lớp, mô tả một khía cạnh nào đó của lớp và được dùng để phân biệt với các lớp khác. Thuộc tính được chia làm nhiều loại khác nhau:

- Về mặt ý nghĩa, các thuộc tính có thể được chia làm hai loại: thuộc tính bên trong (intrinsic property)và thuộc tính bên ngoài (extrinsic property). Thuộc tính bên trong mô tả các tính chất nội tại bên trong sự vật, ví dụ: chất, lượng, cấu tạo... Trong khi đó, thuộc tính bên ngoài mô tả phần biểu hiện của sự vật, ví dụ: màu sắc, hình dạng...
- Về mặt giá trị, các thuộc tính cũng được chia làm hai loại: thuộc tính đơn (simple property)và thuộc tính phức (complex property). Thuộc tính đơn là các giá trị đơn ví dụ: chuỗi, số... còn thuộc tính phức có thể chứa hoặc tham khảo đến một đối tượng khác.

Một chú ý quan trọng nữa trong bước này là việc một lớp sẽ kế thừa toàn bộ các thuộc tính của tất cả các cha nó. Do đó cần phải xem xét một thuộc tính đã được định nghĩa ở các lớp thuộc mức cao hơn hay chưa. Thuộc tính chỉ nên được định nghĩa khi nó là tính chất riêng của lớp đang xét mà không được biểu hiện ở các lớp cao hơn.

4.6 Định nghĩa các thuộc tính và quan hệ cho lớp:

Các ràng buộc giới hạn giá trị mà một thuộc tính có thể nhận. Hai ràng buộc quan trọng nhất đối với một thuộc tính là số yếu tố (cardinality)và kiểu (type). Ràng buộc số yếu tố quy định số giá trị mà một thuộc tính có thể nhận. Hai giá trị thường thấy của ràng buộc này là đơn trị (single)và đa trị (multiple). Nhưng một số phần mềm còn cho phép định nghĩa chính xác khoảng giá trị của số yếu tố. Ràng buộc thứ hai là về kiểu. Về cơ bản, các kiểu mà một thuộc tính có thể nhận là: chuỗi, số, boolean, liệt kê và kiểu thực thể. Riêng kiểu thực thể có liên quan đến hai khái niệm gọi là: miền (domain)và khoảng (range). Khái niệm miền được dùng để chỉ lớp (hay các lớp)mà một thuộc tính thuộc về. Trong khi đó, khoảng chính là lớp (hay các lớp)làm kiểu cho giá trị thuộc tính kiểu thực thể. Lấy ví dụ, lớp động vật trong hình sau có thuộc tính "loại thức ăn" thuộc kiểu thực thể (cỏ cây, côn trùng...). Lúc này, miền của loại



Hình – Ràng buộc.

thức ăn là động vật, sinh vật... Còn khoảng của nó chính là lớp cỏ, cây...

4.7 Tạo các thực thể cho lớp:

Đây là bước cuối cùng khép lại một vòng lặp xây dựng Ontology. Công việc chính lúc này là tạo thực thể cho mỗi lớp và gán giá trị cho các thuộc tính. Nhìn chung, các thực thể sẽ tạo nên nội dung của một cơ sở tri thức.

5. Ngôn ngữ Ontology:

Ngôn ngữ Ontology (Ontology languages)là ngôn ngữ hình thức được sử dụng để xây dựng Ontology. Nó cho phép việc mã hóa tri thức trong một lĩnh vực cụ thể và thường bao gồm các quy tắc suy luận cung cấp cho việc xử lý các yêu cầu dựa trên tri thức đó. Ngôn ngữ Ontology thường là ngôn ngữ khai báo, và hầu hết là những sự tổng hợp của ngôn ngữ cấu trúc, và thường được xây dựng dựa trên Logic thủ tục (First-Order Logic)hoặc dựa trên Logic mô tả (Description Logic). Có rất nhiều ngôn ngữ Ontology đã được thiết kế và đưa ra tuân theo sự tiêu chuẩn hóa, ta sẽ tìm hiểu một số ngôn ngữ Ontology thông dụng nhất trong ngữ cảnh của Web ngữ nghĩa và biểu diễn tri thức hiện nay.

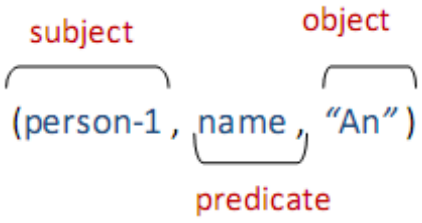
5.1 RDF (Resource Description Framework):

5.1.1 Tổng quan RDF:

RDF là nền tảng cho việc biểu diễn dữ liệu trong lĩnh vực Web ngữ nghĩa. Thông tin biểu diễn theo mô hình RDF là một phát biểu (statement)ở dạng cấu trúc bộ ba gồm ba thành phần cơ bản là: subject, predicate, object. Trong đó:

- Subject chỉ đối tượng đang được mô tả đóng vai trò là chủ thể.
- Predicate (còn được gọi là property)là kiểu thuộc tính hay quan hệ.
- Object là giá trị thuộc tính hay đối tượng của chủ thể đã nêu. Object có thể là một giá trị nguyên thủy như số nguyên, chuỗi...hoặc cũng có thể là một tài nguyên.

Ví dụ:



Hình - Object có thể là tài nguyên hoặc giá trị nguyên thủy, nhưng subject và predicate thì bắt buộc phải là các tài nguyên.

Đây là phát biểu mô tả một chủ thể person-1 có kiểu thuộc tính name với giá trị là An. Phát biểu có thể được tạm hiểu là: person-1 có tên An.

Có thể liệt kê một số ưu điểm của việc lưu trữ dữ liệu RDF so với dữ liệu truyền thống là:

- Tổ chức dữ liệu đơn giản, đồng nhất nên thông tin dễ dàng thêm bớt chỉnh sửa.
- Cấu trúc bộ ba giúp cho thông tin dễ truy xuất bởi các hệ thống suy luận, tìm kiếm ngữ nghĩa. Cũng nhờ vậy mà những bộ xử lí RDF có thể suy luận ra những thông tin mới không có trong hệ dữ liệu.
- Chia sẻ dữ liệu trên mạng dễ dàng nhờ sự đồng nhất...

5.1.2 Chia sẻ dữ liệu RDF:

Mô hình RDF thể hiện được nhiều ưu điểm trong việc biểu diễn thông tin. Chính vì vậy cần phải có một cách thức chung để truyền tải dữ liệu RDF trên internet. Đó là RDF/XML syntax do W3C đưa ra năm 1999. Đây là một ngôn ngữ dựa trên XML, nó bao gồm một tập các quy tắc và từ vựng để hỗ trợ cho biểu diễn thông tin RDF.

RDF/XML syntax: RDF/XML có thể gây khó khăn cho người học bởi vì nó có thể có nhiều cách khi cùng biểu diễn một phát biểu, và một phần là do URI (Uniform Resource Identifier)dùng để định danh cho một tài nguyên thì tương đối dài và khó đọc, khó viết. Tuy nhiên vấn đề này có thể được xử lí bằng cách dùng XML namespace.

Khai báo namespace: việc sử dụng namespace giúp cho tài liệu RDF ngắn gọn và dễ đọc hơn đối với người thiết kế. Chẳng hạn như ta có một địa chỉ là "http://www.semantic.vn/2009/01/rdf-syntax-ns#". Nếu ta gán nó cho một namespace, ví dụ như xmlns: rdf, thì từ nay về sau ta chỉ việc dùng rdf: phone thay cho "http://www.semantic.vn/2009/01/rdf-syntax-ns#phone".

Định danh một chủ thể: Chúng ta dùng cú pháp sau để biểu diễn một bộ ba {subject,predicate,object}, ví dụ:

{person-1, name, An}

{person-1, phone, 0909213456}

Và biểu diễn ví dụ trên trong tài liệu RDF :

```
<rdf :RDF xmlns:rdf=„http://www.semantic.vn/2009/01/rdf-syntax-ns#">
  <rdf :Description rdf:about= „#person-1">
    <rdf :Name rdf:literal= „An">
    <rdf :Phone rdf:literal= „0909213456">
  </rdf :Description>
</rdf :RDF>
```

Thuộc tính rdf : about dùng để chỉ định URI của một resource, còn #person-1 cho ta biết rằng person-1 đã được khai báo trong cùng tài liệu này (trong ví dụ trên chúng ta không chỉ ra person-1 đã được khai báo ở đâu). Nếu person-1 không được khai báo thì chúng ta phải đưa URI của nó vào để sử dụng hoặc là dùng namespace để đại diện cho URI đó. Chúng ta dùng thuộc tính rdf :ID để gán định danh cho một tài nguyên :

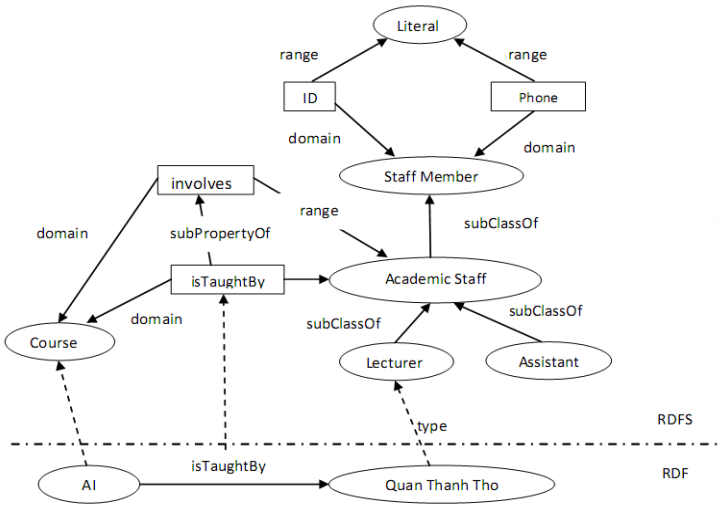
```
<rdf :Description rdf :ID= "person-1">
  <rdf :type rdf :resource= "http://www.semantic.vn/2009/01/rdf-syntax-ns#person-1"/>
</rdf :Description>
```

Ở đây, tài nguyên http://www.semantic.vn/2009/01/rdf-syntax-ns#person-1 đã được định danh bởi person-1. Thuộc tính ID giúp cho chúng ta sử dụng tài nguyên person-1 mà không phải dùng lại URI của nó dài dòng mà phức tạp.

5.2 RDFS (RDF-Schema):

RDF-Schema là một ngôn ngữ Ontology cơ bản. Nó được phát triển ở tầng trên của RDF cho nên bản thân RDF-Schema cũng chính là RDF, nó được mở rộng từ RDF và bổ sung thêm các tập từ vựng để hỗ trợ cho việc xây dựng các Ontology được dễ dàng. Như chúng ta đã biết, ngôn ngữ RDF chỉ giúp cho thông tin được thể hiện ở dạng bộ ba theo đúng mô hình RDF chứ thông tin vẫn chưa thể hiện gì về mặt ngữ nghĩa. Do đó, xây dựng RDFS là điều cần thiết để hình thành nên ngữ nghĩa cho thông tin, là cơ sở để xây dựng các công cụ tìm kiếm

ngữ nghĩa. RDFS và RDF có mối liên hệ tương đối gần gũi nên đôi lúc ta gọi ngôn ngữ này là RDF/RDFS.



Hình – So sánh giữa RDF và RDFS

Trong hình vẽ chúng ta thấy, ở tầng RDF chỉ biểu diễn được thông tin ở dạng bộ ba. Đến tầng RDFS, thông tin đã được phân loại rõ ràng. Chẳng hạn như Quan Thanh Tho có kiểu là Lecturer và Lecturer là lớp con của Academic Staff…

5.2.1 Các lớp và thuộc tính trong RDF/RDFS:

❖ RDF/RDFS định nghĩa 13 lớp cơ bản:

rdfs :Resource (Chỉ định đây là một tài nguyên)

rdfs :Class (Dùng để khởi tạo một lớp)

rdfs :Literal (Lớp các giá trị nguyên thủy: chuỗi, số nguyên…)

rdfs :XMLLiteral

rdfs :Container

rdfs :ContainerMembershipProperty

rdf :Property (Cho biết tài nguyên thuộc lớp này là một thuộc tính - property)

rdf :Statement (Cho biết tài nguyên thuộc lớp này là một phát biểu - statement)

rdf :Bag

rdf :Seq

rdf :Alt

rdf :list (Lớp danh sách RDF)

rdf :Container

❖ Và một số thuộc tính :

rdf :type (Xác định kiểu cho một tài nguyên)

rdfs :subClassOf (Cho biết là 'lớp con của')

rdfs:subPropertyOf ('Thuộc tính con của' thuộc tính)

rdfs:domain (Chỉ định vùng)

rdfs :range (Chỉ định tầm vực)

rdfs:label (Gán nhãn cho một tài nguyên)

rdfs:comment (Chú thích)

rdf :member (Thành viên của một lớp chứa - container)

rdf :first (Phần tử đầu tiên trong một danh sách RDF)

rdf :rest (Danh sách các phần tử còn lại)

rdfs :seeAlso (Các thông tin bổ sung)

rdfs :isDefinedBy (Được định nghĩa bởi)

rdf :value (Gán một giá trị nào đó cho chủ thể)

rdf :subject (Chủ thể của một phát biểu)

rdf :predicate (Thuộc tính của một phát biểu)

rdf :object (Giá trị thuộc tính của một phát biểu)

Với những lớp và thuộc tính được liệt kê trên đây, RDF/RDFS đủ mạnh để xây dựng các Ontology. Tuy nhiên bản thân nó còn chứa đựng nhiều hạn chế như là chưa hỗ trợ tốt về mặt suy luận, cũng như chưa có ràng buộc về kiểu và số yếu tố... mà các ngôn ngữ thế hệ sau sẽ khắc phục (chúng ta sẽ bàn về điều này trong phần OWL).

Các lớp và thuộc tính thường dùng trong RDF/RDFS chủ yếu là: Resource, Class, Property, type, label, subClassOf, subPropertyOf, domain, và range. Trong đó domain là miền tài nguyên sử dụng thuộc tính đó, còn range là phạm vi giá trị có thể gán cho thuộc tính đó. Ví dụ ta có lớp thuộc tính là hasChild thì domain của hasChild là lớp Person, còn range là lớp số nguyên dương chẳng hạn(số con từ 1 trở lên).

5.2.2 Ví dụ xây dựng Ontology với RDFS:

Chúng ta sẽ dùng cú pháp cơ bản sau để biểu diễn cho một phát biểu:

{subject, predicate, object}

Nếu có nhiều hơn một thuộc tính cho một chủ thể thì:

{subject,

{predicate1, object1}

{predicate2, object2}

}

Nếu có nút trống thì biểu diễn như sau:

{subject, predicate,

{predicate-of-bnode,object-of-bnode}

}

Một tài nguyên bắt đầu bởi dấu # chẳng hạn #resource_1 thì có nghĩa là tài nguyên đó đã được khai báo trong cùng một tài liệu mà chúng ta đang xây dựng ontology. Sau này muốn sử dụng tài nguyên đó thì ta dùng #resource _1 chứ không cần phải lấy URI của nó cho phức tạp, khó nhìn.

Bây giờ chúng ta sẽ đi vào xây dựng một Ontology đơn giản, Ontology này sẽ xây dựng tóm gọn sơ đồ như hình so sánh giữa RDF và RDFS ở trên:

{rdfs:Class
{rdf:ID,"StaffMember"}
{rdf:resource, "http://www.semantic.vn/2009/01/rdf-schema#StaffMember"}
}

{rdfs:Class
{rdf:about,"AcademicStaff"}
{rdfs:subClassOf, "# StaffMember"}
}

{rdfs:Class
{rdf:about,"Lecturer"}
{rdfs:subClassOf, #AcademicStaff}
}

{rdfs:Class
{rdf:ID,"Course"}
{rdf:resource, "http://www.semantic.vn/2009/01/rdf-schema#Course"}
}

{rdf:Property
{ rdf:ID,"phone"}
{rdfs:domain, #staffMember}
{rdfs:range ,"http://www.semantic.vn/
2009/01/rdf-schema#Literal"}
}

{rdf:Property
{ rdf:ID,"ID"}
{rdfs:domain, #staffMember}
{rdfs:range ,"http://www.semantic.vn/
2009/01/rdf-schema#Literal"}
}

{rdf:Property

{ rdf:ID,"involve"}

{rdfs:domain, #Course}

{rdfs:range ,#AcademicStaff}

}

{rdf:Property

{ rdf:ID,"isTaughtBy"}

{rdfs:domain, #Course}

{rdfs:range ,#AcademicStaff}

{rdf:subPropertyOf, "involve"}

}

Từ Ontology được xây dựng như trên ta có thể vẽ ra sơ đồ phân cấp các tài nguyên như sau:

Staff Member

Academic Staff

Lecturer

Course

Nhìn sơ đồ ta thấy có hai tài nguyên là StaffMember và Course. Các lớp AcademicStaff và Lecturer chỉ là thừa kế từ StaffMember.

5.3 OWL (Ontology Web Language):

OWL là ngôn ngữ ontology khá mạnh, nó ra đời sau RDFS nên biết kế thừa những lợi thế của ngôn ngữ này đồng thời bổ sung thêm nhiều yếu tố giúp khắc phục được những hạn chế của RDFS. OWL giúp tăng thêm yếu tố logic cho thông tin và khả năng phân loại, ràng buộc kiểu cũng như số yếu tố tương đối mạnh.

Ta hãy tìm hiểu sơ qua các lớp và thuộc tính của OWL để thấy được những ưu điểm của OWL so với RDFS.

Classes	**Properties**	
owl:AllDifferent	owl:allValuesFrom	owl:TransitiveProperty
owl:Class	owl:backwardCompatibleWith	owl:inverseOf
owl:DataRange	owl:cardinality	owl:maxCardinality
owl:DataTypeProperty	owl:complementOf	owl:minCardinality
owl:DeprecatedProperty	owl:distinctMembers	owl:oneOf
owl:DeprecatedClass	owl:differentFrom	owl:onProperty
owl:FunctionalProperty	owl:disjointWith	owl:priorVersion
owl:InverseFunctionalProperty	owl:equivalentClass	owl:sameAs

owl:Nothing	owl:equivalentProperty	owl:sameIndividualAs
owl:ObjectProperty	owl:hasValue	owl:someValuesFrom
owl:Ontology	owl:imports	owl:subClassOf
owl:Restriction	owl:incompatiblewith	owl:unionOf
owl:SymmetricProperty	owl:intersectionOf	owl:versionInfo

Trong OWL có thêm một số thuộc tính hỗ trợ suy luận và ràng buộc.

- ❖ Hỗ trợ suy luận:
  - Tính chất bắt cầu:

Nếu như chúng ta có một lớp thuộc tính "contain" và gán cho nó thuộc tính *owl:transitivePropertive* thì thuộc tính "contain" này sẽ có tính chất bắt cầu. Giả sử ta có thông tin A contain B và B contain C, thì hệ thống sẽ tự suy luận ra một thông tin khác là A contain C, và đây là biểu diễn thuộc tính contain trong OWL:

{owl:TransitiveProperty

{rdf:ID,"contain"}

}

  - Tính chất đảo ngược:

Nếu A có thuộc tính hasParent là B, thì suy ra B có thuộc tính hasChild là A.

{owl:OjectProperty

{Rdf:ID, 'hasChild'}

{owl:inverseOf, #hasParent}

}

- ❖ Hỗ trợ ràng buộc:
  - Ràng buộc kiểu:

Giả sử ta đã có lớp Human, thừa kế từ hai lớp này là Man và Woman. Ta muốn một đối tượng thuộc lớp Man thì không thể thuộc lớp Woman và ngược lại.

{owl :Class

{rdf :about, #Man}

{owl :disjointWith, #Woman}

}

  - Ràng buộc số yếu tố:

Ta muốn một người thì chỉ có một cha và một mẹ. Tức thuộc tính hasParent luôn được gán số yếu tố là 2.

{owl :Restriction

{owl :onProperty, #hasParent}

{owl :cardinality, 2}

}

Trên đây là một vài ví dụ mà trong RDFS không thể làm được. Rõ ràng OWL có nhiều ưu điểm hơn trong việc xây dựng hệ thống ontology thông minh và có phân loại tốt. Với những đặc điểm đó, OWL ngày nay đã trở thành ngôn ngữ ontology chính thức cho việc xây dựng và phát triển các hệ thống Semantic Web.

5.4 DAML + OIL (DARPA Agent Markup Language + Ontology Inference Layer):

DAML + OIL là kết quả từ hai dự án nghiên cứu độc lập với nhau là DAML và OIL nhằm khắc phục những hạn chế về kiểu dữ liệu trong các ngôn ngữ Ontology trước đó là RDF, RDFS. DAML + OIL (gọi tắt là DAML)là ngôn ngữ đánh dấu cho các tài nguyên trên Web, có hỗ trợ suy luận. Ngôn ngữ này được xây dựng có kế thừa từ các chuẩn của W3C như XML, RDF, RDFS... Một số điểm đáng chú ý của ngôn ngữ này là:

- Cho phép giới hạn các kiểu dữ liệu được định nghĩa trong XML Schema hay bởi người dùng. Trong DAML, một thuộc tính có thể nhận giá trị trên nhiều khoảng khác nhau, tạo nên tính uyển chuyển trong việc mô tả dữ liệu.
- Cho phép định nghĩa thuộc tính unique để xác định các đối tượng.
- Cho phép mô tả các quan hệ như hoán đổi và bắc cầu.

DAML sau đó tiếp tục trở thành nền tảng cho một ngôn ngữ Ontology khác là OWL. DAML về cơ bản rất giống với OWL (ngoại trừ tên một số ít thuật ngữ, cú pháp được sửa đổi), tuy nhiên khả năng mô tả các ràng buộc kém hơn. Theo thống kê, đã có khoảng 5 triệu phát biểu DAML từ hơn 20.000 Website trên Internet vào năm 2002.

6. Công cụ phát triển Ontology:

Về lý thuyết, người xây dựng và phát triển Ontology có thể không cần các công cụ hỗ trợ, có thể thực hiện trực tiếp bằng các ngôn ngữ. Tuy nhiên, sẽ không khả thi khi Ontology có kích thước lớn và cấu trúc phức tạp. Thêm vào đó, việc xây dựng và phát triển Ontology không chỉ đòi hỏi việc tạo cấu trúc lớp phân cấp, định nghĩa các thuộc tính, ràng buộc... mà còn bao hàm việc giải quyết các bài toán liên quan trên nó. Có rất nhiều bài toán liên quan đến một hệ thống Ontology như:

- Trộn hai hay nhiều Ontology.
- Chuẩn đoán và phát hiện lỗi.
- Kiểm tra tính đúng đắn và đầy đủ.
- Ánh xạ qua lại giữa các Ontology.
- Suy luận trên Ontology.
- Sao lưu và phục hồi một Ontology.
- Xóa, sửa và tinh chỉnh các thành phần bên trong Ontology.
- Tách biệt Ontology với ngôn ngữ sử dụng (DAML, OWL...).

Những khó khăn trên đã khiến các công cụ trở thành một thành phần không thể thiếu, quyết định đến chất lượng của một hệ thống Ontology. Hiện có rất nhiều công cụ có khả năng hỗ trợ người thiết kế giải quyết những bài toán liên quan. Có thể kể ra một số như: Sesame , Protégé , Ontolingua , Chimaera , OntoEdit , OidEd , Apollo , RDFedt , WebODE , KAON , ICOM , DOE , WebOnto... Chúng ta sẽ tìm hiểu một số công cụ phổ biến nhất.

6.1 Protégé:

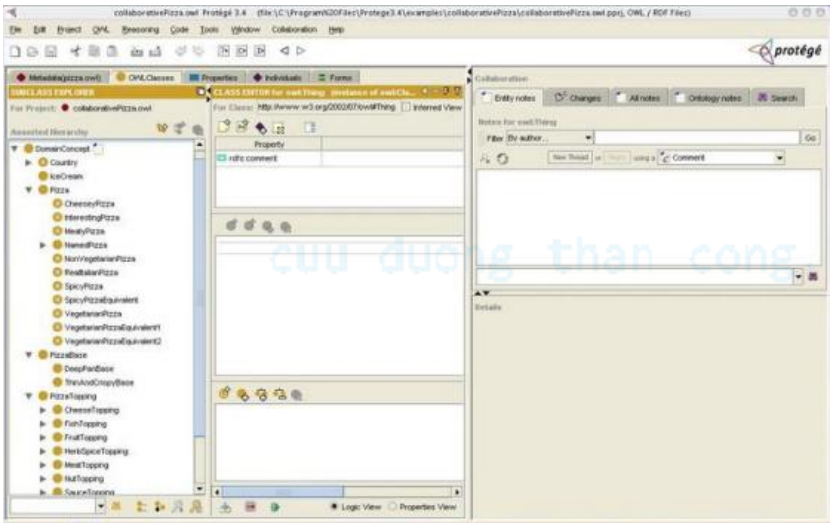
Protégé là bộ phần mềm mã nguồn mở Java nổi tiếng. Protégé được nghiên cứu và phát triển từ năm 1998 bởi nhóm nghiên cứu của Mark Musen, ĐH. Stanford nhằm quản lý các thông

tin trong lĩnh vực sinh y học. Đây là dự án được nhận được sự quan tâm và tài trợ từ rất nhiều tổ chức, trong đó có Bộ Quốc Phòng Mỹ. Mã nguồn Protégé có thể được tìm thấy tại website: http://smi-protege.stanford.edu/repos/protege/owl/trunk/src/edu/stanford/smi/.

Các ưu điểm của Protégé là:

- Hỗ trợ đầy đủ ba phiên bản của ngôn ngữ OWL là OWL-Full, OWL-Lite và OWL-DL.
- Nhờ sử dụng mô hình hướng đối tượng của ngôn ngữ Java, Protégé tỏ ra rất hiệu quả trong việc mô hình các lớp, thực thể, quan hệ...
- Giao diện thiết kế trực quan có tính tương tác cao. Người sử dụng có thể định nghĩa các thành phần của Ontology trực tiếp từ các form.
- Cho phép biểu diễn trực quan Ontology dưới dạng các sơ đồ.
- Cho phép xây dựng Ontology từ nhiều nguồn khác nhau.
- Protégé tự động lưu một bản tạm của Ontology. Nếu có lỗi phát sinh trong quá trình thao tác thì Ontology cũ sẽ tự động được phục hồi. Người thiết kế cũng có thể chuyển qua lại giữa hai bản Ontology này bằng chức năng Revert to a Previous Version và Active Current Version.
- Cung cấp chức năng tìm kiếm lỗi, kiểm tra tính nhất quán và đầy đủ của Ontology. Để sử dụng, người thiết kế chọn chức năng Run Ontology Test và Check Consistency.
- Cho phép các lớp và thuộc tính của Ontology này có thể được sử dụng trong một Namespace khác mà chỉ cần sử dụng các URL để tham khảo. Để sử dụng, chọn chức năng Move Resource to Namespace.
- Hỗ trợ suy luận trực tiếp trên Ontology dựa trên Interface chuẩn DL Implementation Group (DIG).
- Hỗ trợ sinh mã tự động. Protégé cho phép chuyển Ontology thành mã nguồn RDF/XML, OWL, DIG, Java, EMF Java Interfaces, Java Schema Classes... Các mã này có thể được nhúng trực tiếp vào ứng dụng và là đầu vào cho các thao tác trên Ontology khi cần.
- Cung cấp đầy đủ chuẩn giao tiếp cho các Plug-in.

Tuy nhiên, Protégé cũng thể hiện một số hạn chế như không cho phép truy vấn từng phần một cơ sở tri thức dẫn tới việc không quản lý hiệu quả các cơ sở tri thức có kích thước lớn, hoặc chưa hỗ trợ kết nối trực tiếp với một số hệ quản trị cơ sở tri thức phổ biến như Sesame.



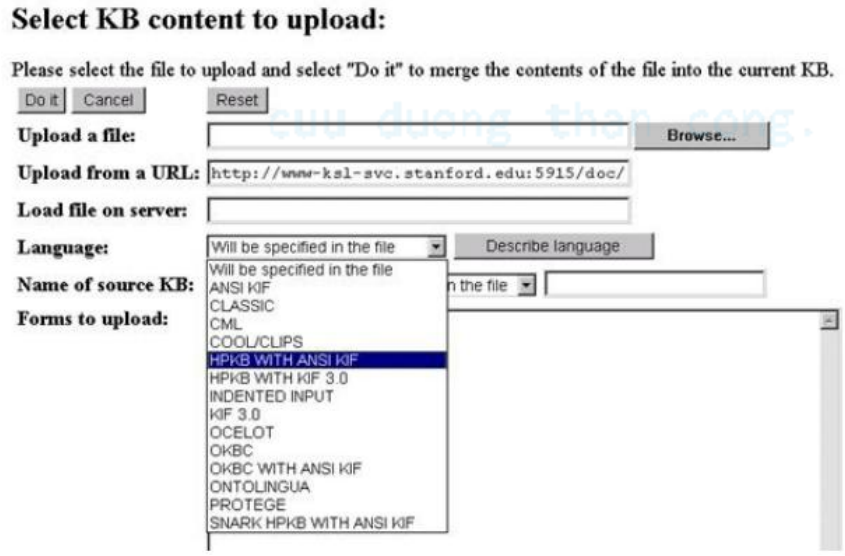
Hình – Giao diện Protégé 3.4

6.2 Chimaera:

Chimaera cũng là một ứng dụng khác được phát triển bởi đại học Stanford, với mục đích ban đầu nhằm giải quyết hai vấn đề là: trộn các Ontology và chuẩn đoán lỗi, phân tích tính nhất quán giữa các Ontology phân tán. Có thể tìm các thông tin liên quan đến Chimaera tại địa chỉ: http://www.ksl.stanford.edu/software/chimaera/.

Một số điểm đáng chú ý của Chimaera là:

- Chimaera là ứng dụng chạy trên nền Web, hỗ trợ thao tác với hơn 15 định dạng Ontology bao gồm: ANSI KIF, Ontolingua, Protégé, CLASSIC, iXOL, OKBC... Riêng hai chuẩn RDF và DAML sẽ được hỗ trợ trong thời gian tới.
- Chimaera tích hợp sẵn chức năng chỉnh sửa Ontology, đặc biệt có thêm chức năng kéo thả và phím tắt nhờ sử dụng các đoạn mã Javascript nhúng vào các trình duyệt. Tuy nhiên, so với các ứng dụng GUI trên Windows/UNIX thì vẫn còn nhiều hạn chế.
- Chimaera có chức năng phân tích, hỗ trợ người dùng chuẩn đoán và kiểm tra các Ontology. Việc kiểm tra này bao gồm kiểm tra tính đầy đủ (thuộc tính hoặc thực thể tham khảo đến một lớp hoặc quan hệ chưa được định nghĩa trong cơ sở tri thức), kiểm tra cú pháp, kiểm tra ngữ nghĩa, phát hiện chu trình (các lớp tham khảo lẫn nhau theo một chu trình)... Kết quả sẽ được thể hiện dưới dạng các test log. Hiện Chimaera đang cố gắng tích hợp thêm lớp ngôn ngữ dưới dạng luật để cho phép người dùng đặc tả các phương thức kiểm tra theo ý muốn.



Hình – Chimaera hỗ trợ hầu hết các định dạng Ontology

6.3 KAON:

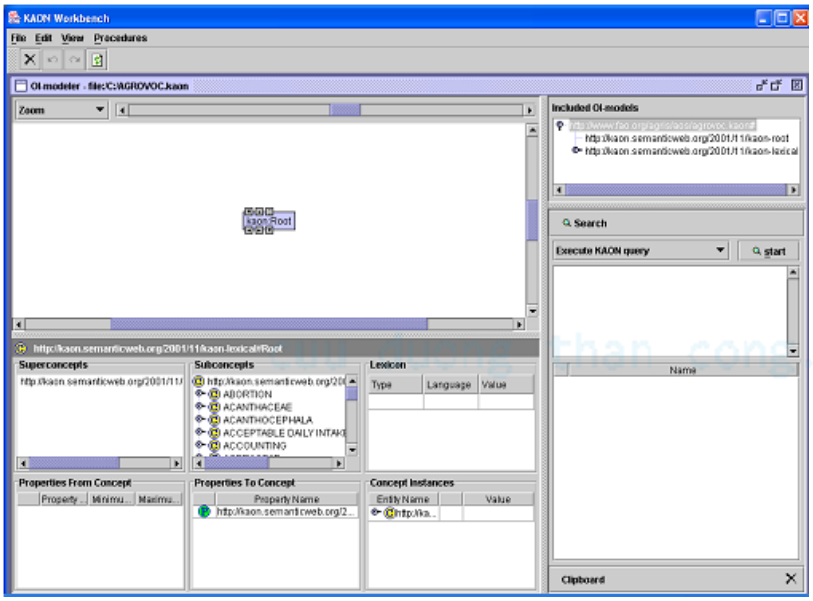
KAON là một hệ thống quản lý Ontology mã nguồn mở nhằm cho các ứng dụng thương mại. Nó là một bộ công cụ đầy đủ cho phép dễ dàng tạo mới và quản lý một Ontology, và nó cung cấp một nền tảng (framework)cho việc xây dựng các ứng dụng Ontology cơ sở. KAON cung cấp ứng dụng cho hai cấp người dùng (user-level): OiModeler và KAON PORTAL, tất cả môđun KAON khác dành cho việc phát triển phần mềm. OiModeler là một chương trình tinh chỉnh Ontology và hỗ trợ cho việc tạo mới cũng như bảo trì Ontology. KAON PORTAL cung cấp một nền tảng đơn giản cho việc truy cập và tìm kiếm của Ontology trên trình duyệt web.

KAON chủ yếu là một nền tảng cho sự phát triển các ứng dụng Ontology cơ sở. Nó có các Môđun sau:

- Ngoại vi (front-end): phần lớn được trình bày bởi hai ứng dụng: OI-modeler và KAON Portal.
- Lõi của KAON (core of KAON): là hai APIs cho RDF và ngôn ngữ Ontology KAON.
- Thư viện (libraries): để cung cấp chức năng của KAON.

Các đặc tính của KAON:

- Nhập/xuất với định dạng RDFS.
- Không hỗ trợ quan sát đồ họa.
- Kiểm tra tính thay đổi.
- Hỗ trợ Web thông qua KAON PORTAL.
- Hỗ trợ đa người dùng.
- Không hỗ trợ việc trộn các Ontology.
- Việc suy luận luận hiệu quả với các Ontology.
- Mở rộng RDFS với các quan hệ đối xứng, bổ sung và nghịch đảo.



Hình – Giao diện của công cụ KAON

7. Tài liệu tham khảo:

- Wikipedia:
  - http://en.wikipedia.org/wiki/Ontology_(information_science)
  - http://en.wikipedia.org/wiki/Ontology_components
  - http://en.wikipedia.org/wiki/Ontology_engineering
  - http://en.wikipedia.org/wiki/Ontology_language
- Blog và Forum:
  - http://my.opera.com/Alibobo/blog/index.dml/tag/WEB%20SEMANTIC
  - http://ngo2uochung.blogspot.com/2009/04/quy-trinh-xay-dung-ontology.html
  - http://www.aiti-aptech.edu.vn/forums/showthread.php?t=1934&goto=nextnewest
- Tài liệu, giáo trình, tiểu luận, luận văn:
  - GIỚI THIỆU SEMANTIC WEB & ONTOLOGY - Võ Hoàng Nguyên, Hoàng Lê Quân – Tháng 5/2009.
  - Knowledge Representation and Ontologies - Logic, Ontologies and Semantic Web Languages - FZI Research Center for Information Technologies, University of Karlsruhe, Germany & Institute AIFB, University of Karlsruhe, Germany.
  - Survey about Ontology Development Tools for Ontology-based Knowledge Management - Seongwook Youn, Anchit Arora, Preetham Chandrasekhar, Paavany Jayanty, Ashish Mestry and Shikha Sethi - University of Southern California.
  - Building Ontology from Knowledge Base Systems - Faten Kharbat, Zarqa Private University - Haya El-Ghalayini, Petra University, Jordan.
  - A software engineering approach to ontology building - Antonio De Nicola, Michele Missikoff, Roberto Navigli.
  - Slide: Ontologies in the Semantic Web – TS. Hoàng Hữu Hạnh, ĐH Khoa Học Huế.
  - Slide: OWL – W3C's Web Ontology Language - TS. Hoàng Hữu Hạnh, ĐH Khoa Học Huế.