

Ontologias e a Web Semântica

Frederico Luiz Gonçalves de Freitas

Programa de Pós-Graduação em Informática
Universidade Católica de Santos - UniSantos
R. Carvalho de Mendonça, 144, Vila Mathias - 11070-906
Santos - SP – Brasil - Fone: +55 13 3205 5500

fred@unisantos.br

Resumo. *A falta de soluções capazes de captar a semântica das páginas da Web criou uma demanda de serviços que se ajusta à aplicação de ontologias. Esta tecnologia desponta como uma forma viável de estruturar informações esparsas, disponíveis na rede. Este trabalho abrange os principais fundamentos teóricos e históricos relativos às ontologias. São abordados seus formalismos de representação, benefícios, ferramentas, princípios e técnicas de construção e aplicações. É apresentada ainda a mais direta aplicação de ontologias: a Web semântica, que, com suas linguagens e padrões, constituirá a base para a efetivação do comércio eletrônico e de um tratamento mais acurado das informações da Web.*

Abstract. *The absence of solutions capable of capturing the semantics of Web pages has created a strong demand for services suitable to the application of ontologies. This technology emerges as a viable alternative to structure sparse information available in the Web. This work comprises the main theoretical and historical topics concerned to ontologies. Its knowledge representation formalisms, gains, tools, building principles and techniques and applications are detailed. It is also presented the main ontology application: the Semantic Web with its languages and standards, which will constitute then basis for the expansion of electronic commerce and a more accurate and intelligent treatment of information in the Web.*

“Conhecimento é a capacidade de atuar num contexto”.

Karl-Erik Sveiby, Ontoprise GmbH, Alemanha

1. A Controvérsia declarativo-procedimental revisitada

As duas abordagens de elaboração de soluções computacionais, aparentemente em oposição, que surgiram logo após o advento das linguagens de alto nível - quando se iniciou a discussão sobre a melhor forma de projetar sistemas - são:

- A abordagem *declarativa*, que descreve fatos e entidades acerca de um determinado domínio (metáfora do “o quê”), fazendo uso de motores de inferência, que deduzem novos fatos a partir dos existentes, entre eles a solução ou ação a ser tomada;
- A abordagem *procedimental*, que descreve o funcionamento de processos em suas nuances mais detalhadas (metáfora do “como”), sem maiores compromissos em descrever as características das entidades a serem representadas.

Os anos 70 e 80 assistiram a um debate entre qual delas seria a melhor forma de conceber e programar sistemas. Na realidade, parece existir complementaridade entre as duas abordagens, já que cada uma se adequa melhor a determinadas tarefas.

Em termos de eficiência e controle, a programação imperativa apresentou melhor performance, tornando-se o paradigma de programação comumente empregado. A chegada da Internet e dos sistemas abertos - ambientes desorganizados e ricos, que requerem a existência de *agentes inteligentes* com forte orientação à cooperação e conseqüentemente à comunicação -, têm demonstrado a necessidade massiva de expressividade e de reuso flexível de conhecimento, provocando uma premente revisão de conceitos. O ressurgimento com vigor de soluções declarativas, baseadas em lógica, acarreta inúmeros benefícios, especialmente para a classe de tarefas relacionada a estes ambientes. As ontologias e o início da implementação da Web Semântica baseada nesta tecnologia, úteis para comércio eletrônico e outras aplicações, modificaram substancialmente as prerrogativas da controvérsia declarativo-procedimental em relação à época em que foi postulada [Winograd 75].

1.1. Abstrações e paradigmas de programação

A evolução da informática está intimamente ligada à elevação do nível das abstrações das soluções propostas. São as abstrações que proporcionam genericidade e abrangência às soluções dos problemas, facilitando o seu reuso em problemas similares. O próprio computador foi projetado com o intuito de ser a máquina que abstrai as outras máquinas, podendo substituí-las, desde que possua conhecimento suficiente para desempenhar as tarefas e dispositivos de entrada e saída adequados.

Aliás, não seria exagero dizer que a história da informática pode ser vista como a abstração crescente de representações. Os códigos binários e endereços das primeiras máquinas foram abstraídos para mnemônicos e rótulos simbólicos e posteriormente para variáveis; as modificações nestes endereços, para atribuições sobre variáveis; saltos, para estruturas de controle; expressões, para funções; comandos para procedimentos; estruturas de armazenamento, para estruturas de dados; “*assembler*” para linguagens de alto nível; consultas a bancos de dados, para linguagens de consulta; modelos físicos de dados, para modelos conceituais; dados e operações, para tipos abstratos de dados ou objetos; tipos, para polimorfismo; compiladores, para gramáticas formais e geradores de compiladores; e, finalmente, as abstrações que deram origem ao paradigma declarativo lógico: funções para relações lógicas e funcionamento ou código para conhecimento explícito expresso num formalismo lógico de representação de conhecimento.

Também é comum que abstrações capazes de resolver problemas complexos sejam subestimadas ou relegadas a segundo plano por problemas teóricos e/ou práticos, para depois serem revistas, aceitas e empregadas em larga escala, como ocorreu, por exemplo com outra sub-área de Inteligência Artificial, a área de redes neurais, virtualmente abandonada até a chegada dos perceptrons com múltiplas camadas [Rumelhart et al 86].

O paradigma de programação declarativo lógico pode ser visto como uma abstração da tradicional programação imperativa. Embora date do fim dos anos 50 e esteja consistentemente fundamentado por teorias de lógica matemática com raízes ainda mais antigas, o paradigma declarativo lógico seguiu uma trajetória pontilhada de

dificuldades (detalhadas na subseção 1.3.1.), até ser redescoberto como uma das soluções mais factíveis aos complexos problemas colocados com a chegada da Internet.

1.2. Nova conjuntura para a controvérsia: a Internet

Os sistemas abertos caracterizam-se principalmente pela capacidade de conectar redes a outras redes, tornando documentos, dados e software acessíveis remotamente por pessoas e outros sistemas, agentes e ferramentas. O surgimento dessa tecnologia acarretou uma forte mudança de paradigma na relação homem-máquina, trazida pela popularização da Internet. Os computadores pessoais, que por aproximadamente 25 anos capacitavam-se apenas a tarefas dentro de um ambiente de informações restrito, controlado e estático, transformaram-se em janelas para um mundo continuamente renovável de informações, pessoas e software. Com isso, a antiga metáfora da manipulação direta da informação – que se sustentava devido à pequena quantidade de informação manipulada – começou a declinar [Maes 97].

O resultado disto terminou por cunhar o termo “sobrecarga de informação” (do inglês “*information overload*”): uma enorme quantidade de documentos disponíveis coloca ao usuário a difícil tarefa de separar o joio do trigo na busca de informação útil. Com o intuito de minimizar este problema, os engenhos de busca – como o *Excite*, o *Yahoo!* e o *AltaVista* - foram então projetados, com base em técnicas desenvolvidas pela área de Recuperação de Informação (RI) [Baeza-Yates & Ribeiro-Neto 99]. Eles indexam as páginas da Internet por palavras-chave, e aplicam métodos e estruturas de dados para recuperá-las, rapidamente devolvendo ao usuário uma lista de endereços de páginas que contém as palavras solicitadas, ordenadas por frequência destas palavras. Dada a variedade de conteúdo da informação disponível, esta era a alternativa viável, uma vez que os dois principais pilares que dão suporte à existência da Internet, o protocolo HTTP (*HyperText Transfer Protocol*) e a linguagem HTML (*HyperText Markup Language*), foram projetados tendo como principal intuito assegurar a apresentação e a navegação na rede. Preocupações sobre como capturar o conhecimento contido nas páginas - ou seja, realizar buscas semânticas - não foram prioritárias. Devido a este fato, os engenhos de busca caracterizam-se por alta cobertura, porém significativa falta de precisão¹, muitas vezes entregando ao usuário uma grande quantidade de endereços de páginas inúteis ou irrelevantes. Utilizando algoritmos matemáticos para atribuir relevância às páginas, estes engenhos não conseguem dotar de semântica a busca, porque possuem capacidade de representar as páginas com análises baseadas apenas no nível léxico.

Basicamente, duas características da Internet dificultam o acesso à informação útil, específica e relevante: o volume e a falta de definição semântica precisa, interpretável por programas e sistemas, para as informações disponibilizadas nas páginas. Por isso, torna-se difícil *agregar valor* à informação disponível, ou seja, transformá-la em informação útil e facilmente acessível, convertendo informação desestruturada ou semi-estruturada em estruturada. Isso permitiria acesso organizado à informação e, o

¹ Medidas de performance típicas de RI: *cobertura (recall)* significa o quociente entre o total de documentos relevantes recuperados sobre o total de documentos relevantes, enquanto *precisão* significa o quociente entre o total de documentos relevantes recuperados sobre o total de documentos recuperados.

mais importante, que agentes inteligentes pudessem inferir sobre as informações capturadas.

A falta de mecanismos capazes de captar a semântica do conteúdo das páginas da Web criou uma forte demanda de serviços que se ajusta adequadamente à classe de serviços estudada em Inteligência Artificial, que passou a ser vista como uma alternativa bastante factível para um melhor tratamento dos problemas relacionados à manipulação de informação na Internet. Basicamente, dois tipos de solução foram propostos, que não são mutuamente exclusivas:

- *Dotar os sistemas de inteligência e autonomia para percorrer e selecionar informação relevante na imensidão da rede, deduzindo ou aprendendo quais as informações úteis. Esta metáfora contribuiu para cunhar termos como *agentes inteligentes, agentes de informação e manipulação cooperativa de informação* [Oates et al 94].*
- *Dotar a própria Internet de inteligência, fazendo com que as páginas possuam uma semântica clara e definida. e que agentes possam raciocinar sobre esta semântica. Essa idéia deu origem ao que chamamos de Web Semântica.*

Com efeito, as ontologias – especificação dos conceitos de um determinado domínio e suas relações, restrições e axiomas, definidos de forma declarativa - representam um papel fundamental em ambas as soluções.

No primeiro caso, as ontologias servem como ferramenta para organização, reuso e disseminação de conhecimento já especificado, facilitando a construção de novos agentes. Porém, para este tipo de solução, as ontologias desempenham um papel ainda mais importante, que motivou a retomada de pesquisas sobre o tema: servir como vocabulário de comunicação entre agentes inteligentes (vide próxima sub-seção).

No segundo caso, linguagens estão sendo desenvolvidas para permitir que páginas sejam anotadas usando formalismos lógicos, que, por sua vez, possam definir ou instanciar ontologias. A Web Semântica será apresentada na seção 7.

1.2.1. A necessidade de cooperação e comunicação entre agentes inteligentes

Face ao tamanho da Internet, qualquer aplicação que se proponha a lidar com toda a rede – seja um engenho de busca, um coletor de estatísticas ou outra aplicação qualquer – deve aproveitar-se de uma característica intrínseca a qualquer rede: a distribuição. A distribuição gera concorrência e paralelismo, e isso faz a diferença em termos de tempo de resposta. Assim, um agente deve ser capaz de pedir, oferecer e receber auxílio de outros agentes, compartilhando com estes os resultados de suas operações.

Para que esta cooperação se torne efetiva, é preciso um modelo de comunicação que lhes permita trocar conhecimento sobre suas tarefas, sobre o domínio de que tratam, sobre suas capacidades e suas intenções em relação aos outros agentes (pedir ou fornecer informação, recrutar outros agentes, etc). Os requisitos de um modelo de comunicação para agentes inteligentes são os seguintes:

- Cada mensagem deve expressar uma intenção pragmática, e um campo especificando essa intenção deve fazer parte da linguagem de comunicação;

- As mensagens devem referir-se a um contexto e vocabulário comuns, sobre o qual a troca de mensagens possa ser efetuada dentro de uma semântica bem definida, segura e sem ambigüidades. Esse contexto é provido por ontologias, que normalmente representam o domínio em que os agentes atuam, como ilustra a figura 1. Nela, os dois agentes se referem a um 777, porque compartilham a ontologia de meio de transportes, e ambos têm em suas bases de conhecimento o fato de que 777 é uma instância de um avião de carreira.

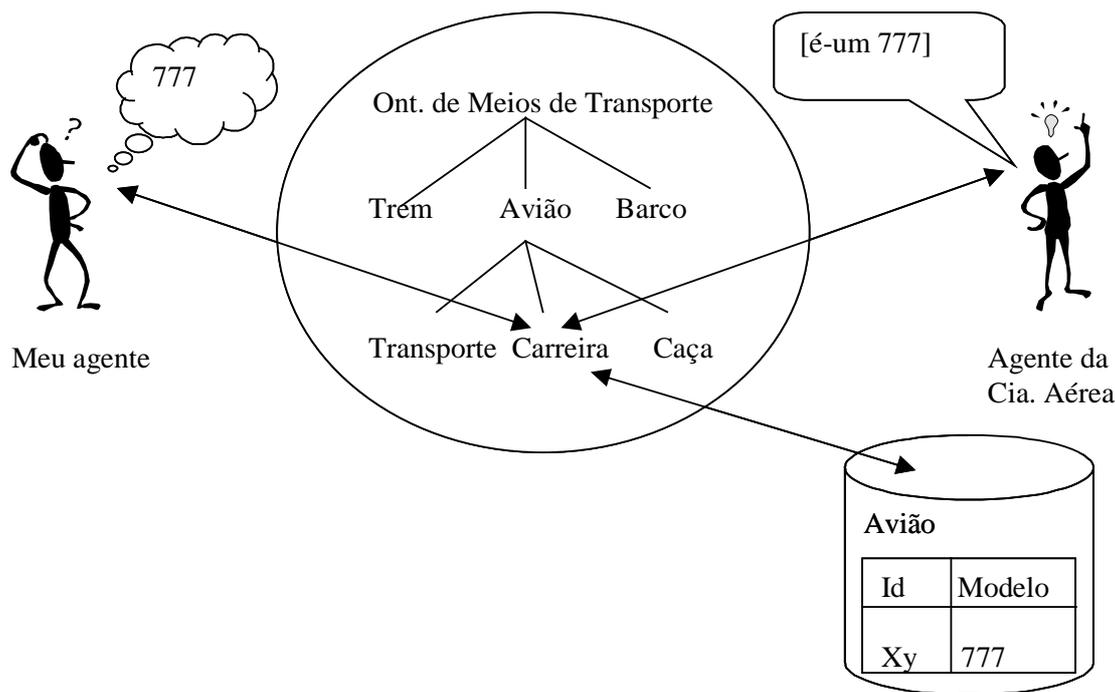


Figura 1. Agentes em comunicação usando uma ontologia como vocabulário compartilhado [Huhns & Singh 97].

- O conhecimento contido na mensagem deve estar escrito em algum formalismo de representação de conhecimento que defina a sintaxe e semântica da mensagem.
- Ambos os agentes são capazes de decodificar mensagens escritas no formalismo escolhido, diretamente ou através de traduções ou mapeamentos.

O modelo de comunicação em nível de conhecimento (ou *peer-to-peer*) satisfaz a estes requisitos. Ele baseia-se na *Teoria dos Atos de Fala* [Austin 62]. A comunicação humana tem sido modelada por esta teoria, que considera que a linguagem falada tem por objetivo engendrar ações e provocar mudanças no ambiente. Os Atos de Fala, anteriormente estudados em Processamento de Linguagem Natural, uma área de Inteligência Artificial, são classificados como *assertivos* (informar), *diretivos* (pedir ou consultar), *comissivos* (prometer ou comprometer-se), *proibitivos*, *declarativos* (causar eventos para o próprio comunicador) e *expressivos* (emoções). A comunicação entre agentes dotados de autonomia e inferência tem seguido este modelo, que sob o ponto de vista de semântica é muito mais claro, portátil, e abrangente do que o modelo Cliente-Servidor. Uma das principais vantagens do modelo em nível de conhecimento reside na proatividade – ou seja, qualquer agente pode iniciar a comunicação a qualquer instante.

Um exemplo de mensagens com intenções e empregando ontologias em agentes de informação encontra-se na seção 4.2.1.

A necessidade de agentes inteligentes, de comunicação em nível de conhecimento e de uma Web Semântica, prementes para a resolução de problemas relativos à manipulação de informação na Internet, trazem à tona de volta a antiga controvérsia declarativo-procedimental.

O presente trabalho visa apresentar não só apresentar as ontologias dentro de uma perspectiva histórica na informática, mas também discorrer sobre aspectos teóricos e práticos de ontologias. O restante desta seção ainda abordará aspectos históricos e teóricos, como motivações, dificuldades e a conceitos iniciais como o nível de conhecimento e componentes de sistemas baseados em conhecimento. Depois vários aspectos sobre ontologias serão vistos: seus formalismos de representação de conhecimento usados para a sua especificação (seção 2), benefícios (seção 3), ferramentas para o seu manuseio e construção (seção 4), além de princípios e técnicas para a sua construção, reuso e manutenção (seção 5). A seção 6 ilustra o uso de ontologias em aplicações práticas. A principal destas aplicações, a Web semântica, é detalhada em suas camadas e linguagens na seção 7. Tópicos abertos de pesquisa em ontologias são discutidos na seção 8, e conclusões são apresentadas na seção 9.

1.3. O nível de conhecimento

A tentativa de criar sistemas diretamente a partir do conhecimento acerca dos processos, isto é, declarativamente, tornou possível aos computadores realizar dedução automática, dando origem à área chamada hoje de Inteligência Artificial Simbólica. A área fundamenta-se basicamente na separação entre o conhecimento e o processo dedutivo ou inferência. A concepção de sistema dessa natureza passa por três especificações consecutivas [Newell 82]:

- O *nível de conhecimento ou epistemológico*, que consiste numa especificação abstrata do conhecimento do domínio ou problema que se deseja modelar;
- O *nível lógico*, que converte as especificações de conhecimento em sentenças de lógica formal;
- O *nível de implementação*, que codifica estas sentenças em linguagem computacional, seja numa linguagem ou num banco de dados dedutivo, dito base de conhecimento.

A existência do nível de conhecimento é de suma importância para os sistemas simbólicos, principalmente por uma razão, que se confunde com a própria história da Inteligência Artificial Simbólica. A motivação principal do paradigma declarativo era *modelar sistemas num nível mais elevado*. O nível de conhecimento guarda uma relação mais próxima e direta com o conhecimento sobre o domínio a ser modelado. O modelo declarativo como um todo consiste numa forma mais flexível de se descrever um domínio, uma vez que, no nível de conhecimento ainda não há compromisso com a implementação.

A motivação de evitar escrever código e tentar abstraí-lo, colocando o conhecimento do lado de fora dos sistemas é extremamente legítima: a maior parte das vantagens do paradigma declarativo – como flexibilidade, legibilidade, fidelidade

semântica, engajamento ontológico, extensibilidade, reuso, abstração das compilações, portabilidade e capacidade de inferência - já era visionada desde essa época, e corroboraram essa idéia. Ainda não se enxergava, contudo, o potencial do uso do conhecimento por entidades de software, que as ontologias viriam a proporcionar, organizando o conhecimento em conceitos e domínios, e nem que elas implementariam o nível de conhecimento apropriadamente, em tese, sem refletir um formalismo de representação de conhecimento específico. Saliente-se que esta prática traz em si a possibilidade do conhecimento ser portátil, *enquanto conhecimento e não enquanto código*, e, portanto, independente de máquina.

1.3.1. Dificuldades iniciais do paradigma declarativo lógico

Contudo, alguns obstáculos eram difíceis de transpor. A dificuldade intrínseca do processo de aquisição - por vezes, o próprio especialista em determinado domínio não sabe porque toma certas decisões -, a falta de metodologias para essa aquisição, a falta de colaboração de especialistas com receio da perda de poder, aliadas à falta de organização de conhecimento em estruturas como as ontologias, fez com que alguns sistemas especialistas simplesmente não obtivessem a performance de acerto aceitável.

O exagerado otimismo com o potencial dos Sistemas Baseados em Conhecimento esbarraria na questão da complexidade computacional, cujos estudos sobre classes de complexidade só viriam a ser colocados em bases concretas no início dos anos 70, quando a indústria já havia se estruturado sobre a computação imperativa. O embevecimento, a falta de objetividade e conseqüente amadorismo da pesquisa gerada, que não se preocupou em firmar produtos e colocá-los no cotidiano, formando indústria, também ajudou a que, durante essa década e a seguinte, se formasse uma imagem negativa da área como um todo. Os resultados de pesquisa limitavam-se a mostrar que alguns caminhos eram viáveis, mas formar indústria e firmar produtos representaria trabalho considerado “braçal”, e, portanto, desinteressante. Ao invés disso, os pesquisadores iam debruçar-se sobre outras áreas sedutoras de Inteligência Artificial.

1.3.2. Motivações do nível de conhecimento

Essa idéia de manter fatos e conhecimento do lado de fora dos programas começou a ser fomentada por um sistema em 1958, o Advice Taker [McCarthy 58], e sedimentou-se com o uso massivo de Lisp na representação de fatos e regras em sistemas especialistas [Winston & Horn 81]. Ao invés de colocarem-se os fatos dentro das regras, percebeu-se que as regras poderiam ser mais genéricas se os fatos estivessem separados. Com isso, o conceito de processo, ao invés de significar uma ação, passou a denotar muito mais uma decisão de ação a ser tomada - no caso de Lisp, através de casamento de padrões, um primeiro esboço de método de dedução, só que ainda sem a presença direta de lógica formal. LISP (*List Processing*, em português, processamento de listas), uma linguagem funcional voltada para o manejo de listas, foi criada justamente com o propósito de representar conhecimento de forma mais abstrata.

Cabe lembrar, ainda, um último argumento favorável às soluções declarativas, que só pôde ser percebido em sua plenitude à luz das ontologias: *o engajamento ontológico ou de conhecimento*, ou seja, a semântica das sentenças codificadas em lógica guarda uma relação mais direta com o domínio modelado e permite o reuso de um

grande volume de informação que desempenha o papel de conhecimento estruturado, disponível para reuso em larga escala por sistemas e programas.

1.4. Os sistemas baseados em conhecimento (SBCs)

Em meados dos anos 70, começaram a aparecer os *sistemas baseados em conhecimento*, à época, também chamados de *sistemas de inferência orientados a padrões* [Hayes-Roth et al 78], que incorporam a programação lógica. Estes sistemas são compostos basicamente de três entidades:

- Um *formalismo* fundamentado em lógica matemática, no qual o código do sistema baseado em conhecimento será escrito,
- Um *método ou estratégia de resolução* para o formalismo escolhido, dito *motor de inferência*,
- Estratégias de controle e escalonamento da inferência ou métodos de resolução de conflitos, implementados dentro dos motores de inferência por módulos chamados *executivos* [Hayes-Roth et al 78]. Os executivos têm a função de guiar e otimizar a inferência, que às vezes não é consistente ou completa.

1.4.1. Novo problema: inadequação do hardware à inferência

As estratégias de controle ou resolução de conflitos encobrem um problema crucial para os sistemas baseados em conhecimento: o hardware, que foi projetado para programação imperativa e não para programação lógica. Como se pode deduzir pelo exposto acima, SBCs devem, tanto quanto possível, explorar concorrência e paralelismo, muito úteis na inferência, com vários predicados podendo ser tratados simultaneamente. Por isso, o processo de inferência (que é simbólico) não “casa” bem com as implementações de hardware e software básico, orientadas à manipulação de números.

Para ilustrar este fato, mencionemos por exemplo o caso da linguagem Prolog, a primeira linguagem baseada em programação relacional ou lógica. Embora tenha sido criada em 1972, Prolog só veio a tornar-se popular a partir de 1977, quando foram criadas as *Máquinas Abstratas de Warren (WAMs)* [Warren 83]. Até então, Prolog não era empregada face à sua lentidão. As WAMs sanaram esta deficiência criando registradores e pilhas virtuais. Na fase de compilação, as sentenças em Prolog são traduzidas para instruções num código intermediário implementado em outra linguagem, (“C”, por exemplo), cujos operadores lançam mão destas pilhas e registradores virtuais, conseguindo uma performance comparável às linguagens mais velozes. Na realidade, é como se toda a programação lógica estivesse sendo “traduzida” para imperativa por restrições de hardware! A

A decisão de continuar mantendo hardwares basicamente imperativos, foi ditada por um mercado crescente baseado em programação imperativa, que também conquistou a maior parte dos fundos de pesquisa. Contudo, houve projetos bem-sucedidos de inferência paralela em nível de hardware, dos quais o projeto japonês PIM (*Paralell Inference Machine*), - parte do projeto de máquinas de 5ª geração que seriam capazes de falar, deduzir e aprender -, foi o mais avançado dentre eles, alcançando 64 MLIPS (milhões de inferências lógicas por segundo) contra 50 KLIPS de Prolog com WAM.

Fica no ar a indagação se teríamos chegado mais rapidamente aos agentes e multiagentes cognitivos de hoje, se este caminho houvesse sido trilhado.

2. Formalismos lógicos de representação de conhecimento

Nesta seção, apresentaremos vários formalismos de representação, divididos em dois grupos de acordo com a abordagem de modelagem de conhecimento. Esta divisão ajuda a perceber a evolução das abordagens de especificação de SBCs. No primeiro grupo, serão apresentados os formalismos orientados a predicados e, no segundo, formalismos orientados a classes e relações.

2.1. Formalismos orientados a predicados

Os dois formalismos a serem apresentados nesta subseção visam, essencialmente, a modelagem de predicados lógicos, abordados como os principais elementos de um domínio. Esta visão funcional dos domínios preocupava-se mais com o conhecimento estratégico - ou seja, como seria feita a inferência e resolvido o problema. A abordagem de desenvolvimento dos primeiros sistemas especialistas compartilhava desta visão, que, por outro lado, dificultava uma especificação do nível de conhecimento mencionado anteriormente, que possui mais analogias com formalismos orientados a relações..

2.1.1. Regras de Produção

A idéia de se empregar regras enquanto formalismo lógico data de 1936, quando foram propostos os sistemas de Post [Bittencourt 2001]. Eis um exemplo de uma sentença lógica transformada em regra orientada ao conseqüente:

$$\begin{array}{ccc} \text{animal}(x) \wedge \text{estimação}(x) \wedge \text{pequeno}(x) & \Rightarrow & \text{doméstico}(x) \\ \swarrow \quad \downarrow \quad \nearrow & & \downarrow \\ \text{premissas} & & \text{conseqüente} \end{array}$$

Esta regra pode ser entendida como: “Para todo x (quantificador universal implícito), se existe *um fato* afirmando que x é animal e *outro fato* afirmando que é de estimação e *outro fato* afirmando que é pequeno, isto implica a validade de um novo fato, o de que x é doméstico”. O formalismo chama-se regras de produção justamente por produzir novos fatos. Um motor de inferência baseado em regras de produção é chamado de motor com encadeamento para a frente (*forward-chaining*).

O primeiro sistema especialista, o Dendral [Feigenbaum et al 71], que inferia estruturas moleculares a partir de dados fornecidos por um espectrômetro, teve seu conhecimento escrito em regras. As regras, enquanto formalismo de representação de conhecimento, tem angariado notório sucesso junto aos usuários, devido à sua simplicidade e facilidade de uso.

A partir da “aprovação” de suas premissas, várias regras podem encontrar-se em condições de serem disparadas ao mesmo tempo, não o conseguindo devido ao funcionamento seqüencial do hardware. A função da resolução de conflito se encarrega de escolher a regra a ser disparada.

Regras consistem no formalismo mais procedimental dentre os existentes, uma vez que, no lado do conseqüente, tanto pode haver a criação de novos fatos na base de

conhecimento quanto a execução de comandos e procedimentos. Com a percepção de que a integração entre as linguagens convencionais, como “C” e Java, e motores de inferência poderia resultar benéfica, foram criados os motores de inferência “plugáveis” a códigos imperativos. Entre os motores de inferência mais populares estão o Jess (*Java Embedded Expert System Shell*) [Friedmann-Hill 97], que possui alguns milhares de usuários, devido à boa integração entre objetos Java, e o CLIPS (“C” *Language Integrated Production System*) [Riley 99] que se integra à linguagem “C”. Mantendo-se o conhecimento fora dos programas, abstraiu-se a compilação, ou seja, não se precisava recompilar os programas quando se alteravam as regras. O conhecimento fora dos programas, torna-se, portanto naturalmente mais extensível e alterável.

2.1.2. Programação em lógica

Neste formalismo, o conhecimento é codificado em conjuntos de sentenças lógicas expressas em cláusulas de Horn invertidas. Uma cláusula de Horn consiste de um conjunto de predicados unidos por conjunções (conectivo “e”) implicando em apenas um predicado, como, por exemplo, em:

$$\text{avô}(x, z) \leftarrow \text{pai}(x, y) \wedge \text{pai}(y, z)$$

traduzido como “x é avô de z, se x é pai de y e y, pai de z”. Esta é uma cláusula invertida, pois o conseqüente aparece antes das premissas.

Os motores de inferência sobre conhecimento especificado dessa forma têm “encadeamento para trás” (*forward-chaining*). A estratégia de inferência, nesse caso, fundamenta-se no algoritmo de resolução [Robinson 65], que encontra a prova por negação (ou refutação)². Embora seja executável por linguagens como Prolog, este formalismo já não possui qualquer traço procedimental, e pode modelar *frames* e redes semânticas (vide próxima subseção).

2.2. Formalismos orientados a classes e relações

Os formalismos aqui apresentados adotam um ponto de vista epistemologicamente distinto dos anteriores: focam nos classes de objetos presentes do domínio, buscando relações entre eles, principalmente a *herança* ou *especialização*, e criando as taxonomias de classes, que ajudam a perceber o domínio como um todo. Isto representa claramente um ganho de declaratividade. Adequados para modelar domínios, não é por acaso que esses formalismos constituem a base para as ontologias

2.2.1. Quadros (*frames*)

Os *frames* [Minsky 75] foram inspirados na forma como as pessoas resolvem problemas, trazendo da memória estruturas de padrões de situações passadas, e tentando instanciá-las no presente. São considerados um dos precursores dos atuais objetos imperativos, guardam substancial semelhança com estes, pois possuem:

² Explicações sobre o algoritmo de resolução podem ser encontradas em [Russel & Norvig 95, Bittencourt 2001]. Outras questões relevantes sobre formalismo lógicos dizem respeito à decidibilidade – se existe um algoritmo de prova para um conjunto de sentenças –, consistência – se um algoritmo de inferência gera apenas sentenças dedutíveis – e completude – se é possível achar a prova de todo predicado dedutível. Estas questões, porém, estão fora do escopo deste trabalho.

- *Classes*, que funcionam como os conceitos das ontologias, normalmente organizados em hierarquias. As classes ligadas com classes mais altas, são subclasses destas classes mais altas, que, por sua vez, são superclasses destas classes mais baixas.
- *Atributos*, que definem as características de membros de uma classe, preenchidos com valores do tipo especificado para o atributo. Este tipo pode ser um dos tipos básicos (inteiro, string, booleano, float, símbolo, etc) ou o tipo especial *instância de classe*, que é responsável por uma função muito importante em sistemas de *frames*: definir relações entre classes. Por exemplo, a classe Professor teria um atributo Orientados, que seria da classe Estudantes.
- *Instâncias* de classes, objetos criados e modelados de acordo com a definição das características da classes (seus atributos), forma de preenchê-los, etc.
- *Herança*, que são relações do tipo *é-um* entre as classes, bem semelhantes ao conceito de subclasse dos objetos. É a herança que define a hierarquia. Diferentemente da maioria dos sistemas orientados a objetos, pode existir herança múltipla.

Além da herança múltipla, *frames* têm uma outra característica própria. São dotados de estruturas pré-definidas, chamadas de *facetras*, para especificar restrições sobre os atributos. As facetras mais comuns, além do *tipo*, presentes em linguagens de *frames* são:

- *Valor default*, que atribui um valor para o atributo, caso nenhum tenha sido assinalado na criação de uma instância ou subclasse;
- *Valores permitidos (allowed-values)* para o atributo,
- *Domínio*, que determina conjuntos de valores esparsos, possíveis para um atributo (por exemplo, 1..100, ou seja, de 1 a 100).
- *Classes permitidas (allowed-classes)* para o atributo, se ele é do tipo instância de classe,³
- *Cardinalidade*, determinando o número máximo e mínimo de elementos de um atributo, pois ele pode ser multivalorado,
- *Documentação*, para descrever o atributo.

Observe-se, ainda, que a faceta *valor default* faz com que instâncias sejam consideradas exceções à classe, o que, por sua vez, faz com que um sistema de *frames* se enquadre como uma *lógica não-monotônica*, uma vez, que, se um dado é acrescentado à base de conhecimento, a conclusão do raciocínio torna-se diferente da normalmente esperada.

Outra característica é a existência de atributos *inversos*, que estabelecem uma correspondência biunívoca entre si. Por exemplo, o atributo Orientados da classe Professor é inverso do atributo Orientador da classe Aluno.

³ Alguns sistemas de *frames* consideram que a faceta *domínio* coincide com a faceta *classes permitidas*.

Porém, a semelhança entre *frames* e objetos é apenas aparente [Farquhar 97]. Enquanto objetos e suas classes visam modelar estruturas de dados e reusar código, *frames* têm por intuito modelar aspectos de um domínio real. Por exemplo, elipse não é, no mundo real, uma sub-classe de círculo, mas para reusar código, modelagens orientadas a objeto podem violar premissas ontológicas, no exemplo abaixo:

```
Class circulo {int x,y; int altura} e
Class elipse extends circulo {int largura}
```

. Também não é necessária em *frames* a inclusão de detalhes de implementação, como tamanho de strings, etc. Em termos de implementação, a diferença principal reside na presença de facetas e pela ausência de métodos e respectivo encapsulamento em *frames*, desnecessária para a declaratividade.

Apesar de a inferência sobre *frames* ser normalmente realizada através de unificação, nada impede que *frames* e regras possam ser combinadas. O sistema de produção CLIPS, para ganhar expressividade, inclui uma linguagem interna (COOL – “C” *Object-Oriented Language*, linguagem orientada a objetos para a linguagem de programação “C”) para definir e manipular *frames*, implementados com todas as suas características. As regras de produção referenciam estes *frames* [Riley 99].

A linguagem de *frames* atualmente mais usada para a definição de ontologias, é a F-Logic [Kifer et al 95], que permite ainda a integração de objetos procedimentais e a definição de axiomas. Axiomas são sentenças consideradas sempre válidas, e que, na prática, servem aos atributos como restrições complexas e que podem envolver classes, instâncias e outros atributos.

2.2.2. Redes semânticas

Redes Semânticas [Woods 75] definem conceitos ligados por relações, representadas graficamente como arcos. Pode-se considerar as Redes Semânticas como um sistema baseado em *frames*, se os arcos da rede forem atributos do *frame* correspondente a essa rede. Ressalve-se, porém, o fato de que as relações de herança e os conceitos de classe constituem o cerne de um *frame*, portanto, se a hierarquia de conceitos for mais importante na representação do domínio tratado do que os atributos, a representação em *frames* ganha em organização. Quando os atributos são mais importantes, as Redes Semânticas são mais adequadas. Por sua expressividade nesses casos, as redes têm sido largamente aplicadas em Processamento de Linguagem Natural. Por exemplo, a linguagem *LIFE (Logics, Inheritance, Functions and Equations)* [Ait-Kaci & Podelski 93], uma linguagem lógica e funcional com tipos e herança, representa redes semânticas. Um exemplo de uso desta linguagem aparece no seguinte código, que especifica, entre outras coisas, que, se duas pessoas são cônjuges, elas têm o mesmo sobrenome.

```
X: pessoa ( nome => id ( primeiro => string, último => Y: string ),
           Cônjuge => pessoa ( nome => id ( último => Y),Cônjuge => X))
```

Note que rótulos (X e Y) expressões restrições complexas sobre atributos. A rede semântica correspondente a este código é mostrada na figura 2.

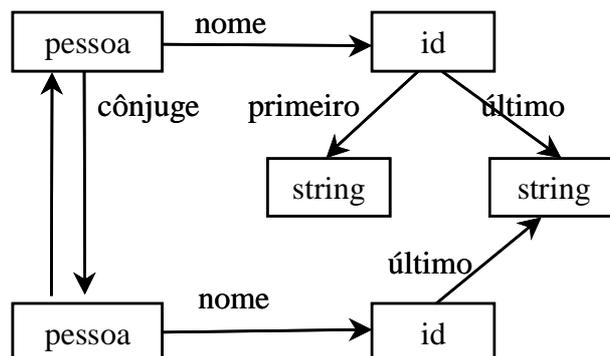


Figura 2. Representação gráfica de uma rede semântica sobre cônjuges.

2.2.3. Lógica de descrições

Baseada em redes semânticas e inicialmente chamada de *lógica terminológica* [Brachman & Schmolze 85], este formalismo propõe a definição de conceitos por meio de descrições esparsas. Não é necessário, como nos *frames*, que todas as definições relativas a uma classe estejam agrupadas; o usuário vai gradativamente entregando-as ao sistema, que, através de um *classificador*, se encarrega de organizar as informações e inferir o que é possível. O formalismo fornece nativamente atributos bastante expressivos, com semântica e inferência associadas, porém cuidadosamente escolhidos de forma a garantir um algoritmo de inferência decidível e eficiente. Alguns deles implementam o seguinte [Fensel 2001]:

- *Herança Múltipla*: e.g., (PRIMITIVE (AND CAR EXPENSIVE-THING) sports-car), denotando que uma instância do conceito carro esporte (*sports-car*) herda as descrições de CAR e EXPENSIVE-THING. Esta não pode ser considerada a única forma de herança para esta lógica, uma vez que existem outras formas de aproveitar descrições já existentes, como pode ser observado em outros atributos logo abaixo e em outras linguagens como OWL (vide subseção 7.5.).
- *Papéis*: e.g., (FILLS thing-driven Corvette), significa que um Corvette desempenha o papel de algo dirigível;
- *Restrição de valores*: e.g., (ALL thing-driven CAR), onde tudo que é dirigível dentro deste domínio deve ser da classe CAR;
- *Restrição de limites*: e.g., (AT-LEAST 3 wheel), descrevendo qualquer objeto relacionado a 3 outros objetos distintos que “desempenham o papel” de roda (*wheel*).
- *Co-referência*: e.g., (SAME-AS (driver) (insurance-payer)), atribuindo a todos os indivíduos que atuam como motoristas (*driver*) o papel de segurados (*insurance-payer*).

Exemplos de linguagens que implementam este formalismo são CLASSIC, FLEX, KL-ONE e LOOM.

2.4. O Elemento de Representação Comum : *Frames*

Os formalismos de representação de conhecimento orientados a classes e relações (*frames*, redes semânticas e lógica de descrições) proporcionaram a preciosa oportunidade de se estruturar conhecimento em ontologias. Até o início dos anos 90, o conhecimento de um sistema especialista não podia ser reusado ou compartilhado, organizando-se em bases de conhecimento monolíticas e isoladas, escritas em diversas linguagens, sem interfaces de acoplamento, e, portanto, sem interoperabilidade, apesar dos formalismos lógicos serem, muitas vezes, similares. Sentiu-se a necessidade de reuso de conhecimento em muitos dos domínios modelados, especialmente em medicina, onde os conceitos são numerosos e cheios de relacionamentos.

Precisava-se encontrar um substrato comum de representação, no qual o conhecimento pudesse ser especificado e facilmente convertido para outros formalismos. O projeto KSE (*Knowledge Sharing Effort* – em português, esforço de compartilhamento de conhecimento) [Farquhar 96] tornou-se pioneiro em escolher os *frames* como base para a construção de ontologias reusáveis e ambientes para a sua especificação e reuso. Os *frames*, por sua simplicidade e expressividade, têm sido, a partir desta decisão, o elemento básico das ontologias, o formalismo comum de interoperabilidade entre bases de conhecimento sobre os mais diversos domínios, escritas sob a forma de redes semânticas, lógica de descrições e até dos próprios *frames*,

3. Ontologias: o ressurgimento das soluções declarativas

Além dos problemas de interoperabilidade citados logo acima, na visão de muitos pesquisadores, herdada dos formalismos orientados a predicados, a especificação de sistemas baseados em conhecimento deveria ter uma visão puramente funcional, com o foco sobre o conhecimento estratégico – regras e funcionamento da inferência –, o que reduzia o volume de conhecimento a ser reusado. Muitos dos conceitos e relações do domínio estavam implícitos, eram tratados como premissas e/ou misturados com conhecimento estratégico. Bases de conhecimento confeccionadas desta forma eram úteis apenas para a resolução de problemas específicos [Gennari et al 2003].

A construção de ontologias, sob essa ótica, pode ser vista como um passo importante de evolução na especificação de conhecimento. A decomposição desse conhecimento em módulos de construção com forte engajamento ontológico – ou seja, similaridade com o conhecimento tanto em relação à forma como ele é organizado, como com a terminologia empregada numa área específica, propiciou a criação de ontologias. A idéia de reuso, ademais, adiciona à especificação de conhecimento a possibilidade de composição, contrapondo-se às bases de conhecimento monolíticas e específicas elaboradas até então.

3.1. Definições de ontologias

A necessidade de melhores ferramentas de busca e tratamento de informação para a Web terminou por disseminar o termo “agente” para praticamente vários outros ramos da informática. Nesta trilha, o novo termo da moda, que vem herdando essa popularidade, face às promessas de melhoria destas atividades, é o termo ontologia, uma vez que elas estão presentes em muitos sistemas, ferramentas e produtos de manipulação de informação e comércio eletrônico, representadas como hierarquias de palavras-chave,

conceitos, e muitas outras formas, chegando a haver, inclusive, vagas no mercado para desenvolvedores de ontologias, ou “ontologistas” [Clark 99]. Contudo, ontologias devem possuir um significado e abrangência muito mais profundos do que as simples hierarquias de conceitos e palavras-chaves empregadas por muitos engenhos de busca.

Apesar da palavra “ontologia” denotar uma teoria sobre a natureza do ser ou existência, em Inteligência Artificial ela pode ser interpretada como o conjunto de entidades com suas relações, restrições, axiomas e vocabulário. Uma ontologia define um domínio, ou, mais formalmente, especifica uma conceitualização acerca dele [Gruber 95]. Normalmente, uma ontologia é organizada em hierarquias de conceitos (ou taxonomias). Pelo fato de, idealmente, não refletirem nenhum formalismo específico, e de representarem com frequência um vocabulário comum entre usuários e sistemas [Clark 99], pode-se considerar as ontologias como a materialização do nível de conhecimento.

Podemos, também, definir o termo ontologia a partir dos requisitos para possibilitar sua aplicação em informática:

“Uma ontologia é uma especificação explícita e formal de uma conceitualização compartilhada”. [Studer et al 98]

Esclarecendo os requisitos desta definição:

- Por especificação explícita, podemos entender as definições de conceitos, instâncias, relações, restrições e axiomas.
- Por formal, que é declarativamente definida, portanto, compreensível para agentes e sistemas.
- Por conceitualização, que se trata de um modelo abstrato de uma área de conhecimento ou de um universo limitado de discurso.
- Por compartilhada, por tratar-se de um conhecimento consensual, seja uma terminologia comum da área modelada, ou acordada entre os desenvolvedores dos agentes que se comunicam.

De fato, ontologias pré-construídas sobre domínios restritos têm sido bastante reutilizadas e podem vir a representar um papel fundamental como fornecedoras de conhecimento para a inferência dinâmica realizada por agentes inteligentes.

3.1. Tipos de ontologias

Diferentes tipos de ontologias, de acordo com seu grau de genericidade podem ser delineados (adaptado de [Gómez-Perez 99]):

- *Ontologias de representação* definem as primitivas de representação - como *frames*, axiomas, atributos e outros - de forma declarativa. Essa idéia [Chandrasekaran & Josephson 96] abstrai os formalismos de representação, porém traz desvantagens (ver subseção 4.2.7.).
- *Ontologias gerais (ou de topo)* trazem definições abstratas necessárias para a compreensão de aspectos do mundo, como tempo, processos, papéis, espaço, seres, coisas, etc.

- *Ontologias centrais (core ontologies) ou genéricas de domínio* definem os ramos de estudo de uma área e/ou conceitos mais genéricos e abstratos desta área. Por exemplo, a ontologia central de direito criada por Valente e Breuker [96], inclui conhecimentos normativos, de responsabilidade, reativos, de agências legais, comportamentos permitidos, etc. Esses conceitos e conhecimentos foram agrupados nesta ontologia para que ela sirva de base para a construção de ontologias de ramos mais específicos do direito, como direito tributário, de família e outros.
- *Ontologias de domínio* tratam de um domínio mais específico de uma área genérica de conhecimento, como direito tributário, microbiologia, etc.
- *Ontologias de aplicação* procuram solucionar um problema específico de um domínio, como identificar doenças do coração, a partir de uma ontologia de domínio de cardiologia. Normalmente, ela referencia termos de uma ontologia de domínio.

Como se pode perceber, os tipos de ontologias estão listados em ordem decrescente de genericidade. Nem todos os tipos são necessários para a construção de uma aplicação. Há ainda que se preocupar em manter as ontologias reusáveis, ou seja, escolher que uma ontologia num nível acima da ontologia em uso esteja ligada a esta ou evitar relacionamentos com uma ontologia específica.

Existe outra classificação quanto ao teor das ontologias, aplicável apenas para os dois últimos tipos citados acima:

- *Ontologias de tarefas* descrevem tarefas de um domínio - como processos, planos, metas, escalonamentos, etc -, com uma visão mais funcional, embora declarativa, de um domínio.
- *Ontologias de domínio* propriamente ditas, tem uma visão mais epistemológica do domínio, focando nos conceitos e objetos do universo de discurso.

3.2. Exemplos de ontologias

As ontologias mais referenciadas na literatura são as gerais, pois muitas aplicações tomam-nas como base. Este tipo de ontologias compartilhou seu nascedouro com a própria lógica [Russel & Norvig 95]: Aristóteles, ao criar a primeira forma de lógica, criou também as *categorias*, uma taxonomia para os objetos do mundo.

Entre as ontologias mais citadas, estão a ontologia de Sowa [Sowa 1999] e a Cyc [Lenat & Guha 90], que visava o raciocínio de senso comum. Caso isto fosse alcançado, SBCs poderiam, com o auxílio de uma grande ontologia, ter um conhecimento generalizado acerca do mundo e fazer inferências úteis, e não apenas as inferências específicas, típicas de um sistema especialista, que trabalha sobre um domínio restrito. Existem duas razões para a existência de ontologias gerais: um grande investimento foi feito para sua existência tentando aproximá-la da realidade, e também a independência ou abrangência expressa nelas, que não necessitam referenciar outras ontologias. A figura 3 mostra as classes primitivas de algumas ontologias de topo, evidenciando diferentes abordagens epistemológicas.

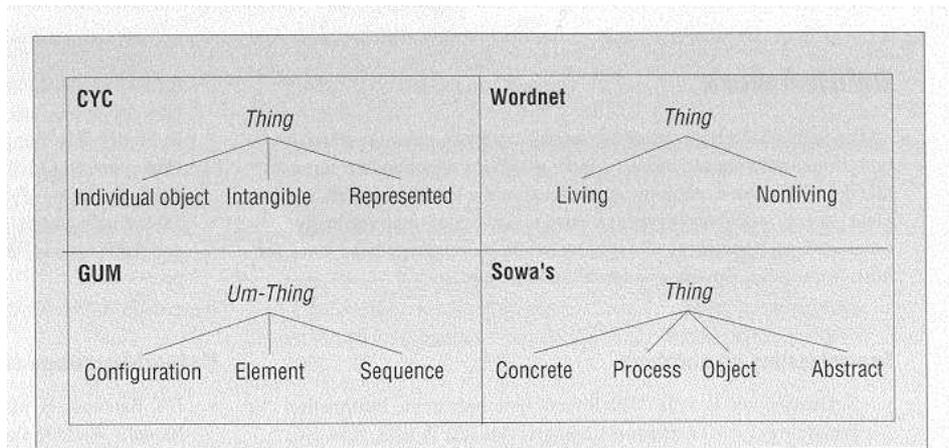


Figura 3. Diferentes primitivas de ontologias de topo [Chandrasekaram 99].

Ontologias lingüísticas, como o WordNet [Miller 95], contém sinônimos, adjetivos e verbos relacionados por redes semânticas, e também são bastante referenciadas por sua aplicação em sistemas de processamento de linguagem natural. Elas se propõem a atacar ambigüidades lingüísticas durante o tratamento sintático e semântico de textos.

3.3. Benefícios das ontologias

À parte dos benefícios advindos de uma abordagem declarativa, outros benefícios mais diretos, ligados à prática de construção de sistemas baseados em conhecimento, têm sido gerados. De início, o projeto KSE e suas ontologias contribuíram para uma maior cooperação entre os grupos de pesquisa responsáveis por manter as ontologias, da mesma forma como mantêm conhecimento, o que, tornando-se uma tendência, pode vir a provocar uma mudança cultural. Desde que foi criado o KSE, estão sendo definidas e mantidas ontologias extensíveis, abrangentes, gerais e muito detalhadas, por grupos de pesquisa, abarcando toda a pesquisa da área cujo conhecimento se deseja representar. Esta orientação ontológica trouxe muitos benefícios, alguns dos quais não previstos, e que só vieram frutificar à época de sua implementação. São eles:

- A oportunidade para os desenvolvedores de reusar ontologias e bases de conhecimento, mesmo com adaptações e extensões. O impacto sobre o desenvolvimento de sistemas baseados em conhecimento é substancial: a construção de bases de conhecimento redonda na tarefa mais cara e demorada de um projeto de sistemas especialistas e/ou agentes. Às ontologias, permite-se ainda aos usuários efetuarem consultas, comparações, integração e checagens de consistência (vide figura 4).
- A disponibilização de uma vasta gama de “ontologias de prateleira”, prontas para uso, reuso e comunicação por pessoas e agentes. Hoje as ontologias mais maduras, algumas com mais de 2.000 definições, incluem metadados de imagens de satélites e para integração de bases de dados de genoma, catálogos de produtos, osciloscópios, robótica, semicondutores, terminologia médica, o padrão IEEE para interconexões entre ferramentas, e outras [Farquhar 96].

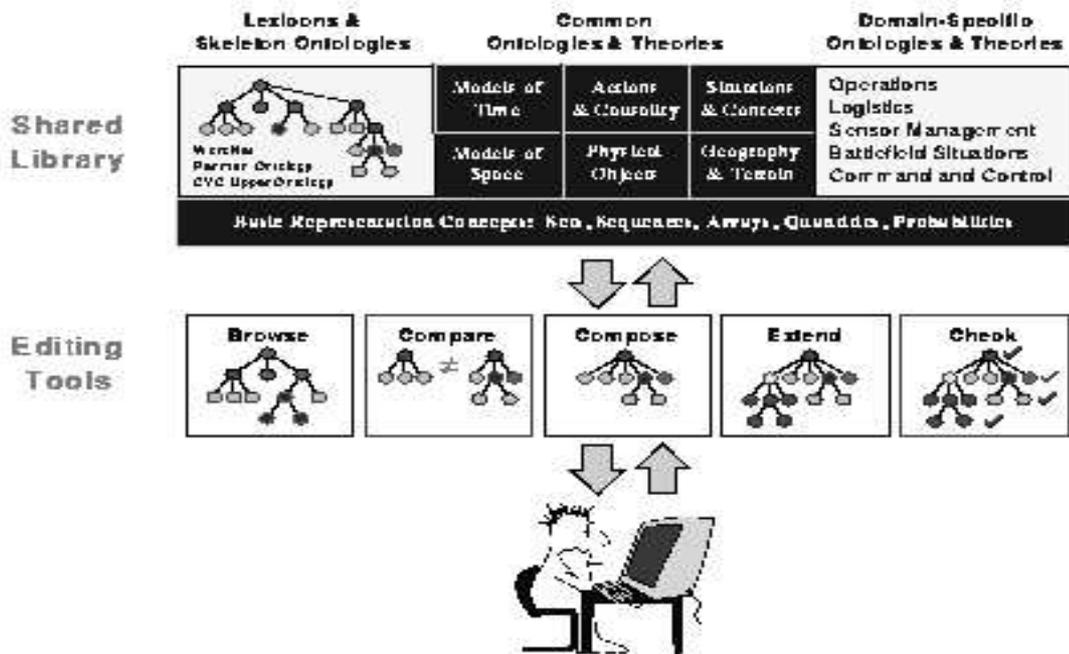


Figura 4. Possibilidades de criação, manutenção e reuso de ontologias através de editores [Fikes 98].

- A possibilidade de tradução entre diversas linguagens e formalismos de representação conhecimento. A tradução concretiza um ideal perseguido por gerações de pesquisadores de Inteligência Artificial. Ela facilita o reuso de conhecimento, e pode vir a permitir comunicação entre agentes em formalismos diferentes, uma vez que este serviço encontra-se disponível um número cada vez maior de formalismos de representação de conhecimento (para os formalismos tratados pela Ontolingua, vide Figura 5). Outra forma de alcançar esse intento são editores de ontologias em que ele pode escolher em que linguagem de representação será escrito o código gerado. No editor Protégé-2000 [Noy et al 2000], podem ser geradas ontologias em CLIPS, Jess, Prolog, XML, RDF, OIL, DAML-OIL e F-Logic.

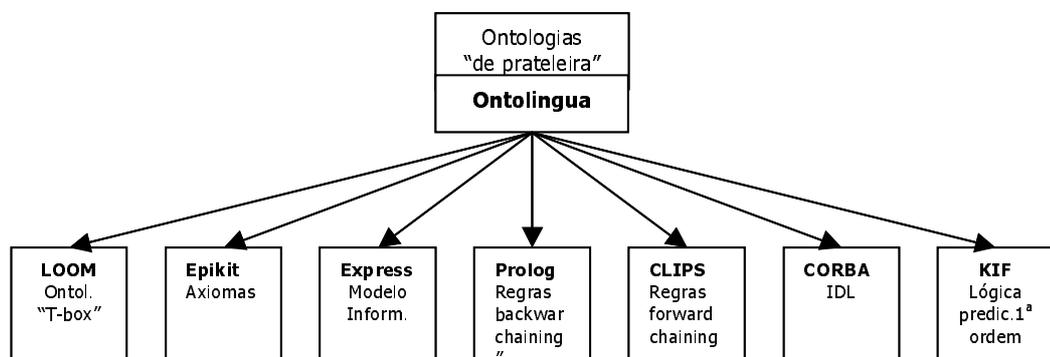


Figura 5. A Ontolingua e os formalismos para os quais podem ser traduzidas as ontologias.

- O acesso *on-line* a servidores de ontologias, capazes de armazenar milhares de classes e instâncias, que serviriam a várias empresas ou grupos de pesquisa, e que

podem funcionar como ferramentas para manter a integridade do conhecimento compartilhado entre elas, garantindo um vocabulário uniforme.

- O mapeamento entre formalismos de representação de conhecimento, que, inspirado no componente de conectividade para sistemas gerenciadores de bancos de dados ODBC (*Open Database Connectivity*), liga dois formalismos criando uma interface interoperável de acesso comum para eles, permitindo a um agente acessar o conhecimento de outro agente. Por isso, o pacote gerado para implementar esta facilidade foi chamado de OKBC (*Open Knowledge Base Connectivity*) [Chaudri et al 98].

4. Ferramentas para Manuseio de Ontologias

A construção de ontologias para a comunicação em nível de conhecimento entre agentes cognitivos tem se tornado alvo de estudos, e propiciou a elaboração de ferramentas com o objetivo de dar suporte não só a esta atividade, mas a atividades-fim de uso de ontologias, como, por exemplo, ferramentas de escritório para gestão de conhecimento (*knowledge management*). Com efeito, vários ambientes para a manipulação de ontologias - que incluem editores, servidores, repositórios e ferramentas para consolidação e tradução de ontologias - estão surgindo, com o objetivo não só de facilitar a sua construção, como também de disponibilizar ontologias públicas para reuso e extensão, integrando diferentes grupos de pesquisa, amiúde distantes geograficamente, que pesquisam sobre as mesmas áreas ou áreas afins. Vale salientar que alguns editores fornecem ferramentas para refazer, desfazer e auditar mudanças (documentando quem as fez e quando), além de verificação de conflitos e tratamento de instâncias e classes órfãs.

4.1. Requisitos para ferramentas e seus formalismos e linguagens de representação

Alguns fatores tornam uma ferramenta popular [Duineveld et al 99]: facilidades de uso, entendimento intuitivo da interface, visibilidade gradativa (característica útil quando se visualizam grandes ontologias), interfaces gráficas, conexão a repositórios, portabilidade, interoperabilidade, organização dos arquivos gerados, documentação de alterações, suporte a trabalho cooperativo, extensibilidade (capacidade de inclusão de componentes) e ferramentas de apoio que facilitem o desenvolvimento. Podemos considerar que a área passa por um processo de maturação, uma vez que já existe um evento, o EON (*Workshop of Evaluation Ontology Tools*) [EON 2002], já em sua 2ª. edição, destinado especificamente à avaliação de ferramentas para ontologias, onde são definidos requisitos desejáveis e fundamentais de editores, como integração e interoperabilidade, e até realizados testes com as ferramentas participantes.

Em relação à avaliação de formalismos e linguagens de representação, não se deve tomar em consideração apenas expressividade, como foi feito em [Corcho & Gómez-Perez 2000]. Outros itens de mais relevância é que garantem a efetiva utilização de um formalismo e/ou linguagem num mercado: a existência de um motor de inferência, a possibilidade de acoplamento do motor a uma ferramenta de edição de ontologias, a independência de uma linguagem ou formalismo específico, e a popularidade do formalismo. O Ontosaurus [McGregor 90], por exemplo, manipula bases de conhecimento apenas na linguagem para lógica de descrições LOOM, o OilEd [OilEd 2003], apenas na linguagem para a Web semântica OIL [Horrocks et al 2000], e o

WebOnto[Domingues 98], na linguagem proprietária OCML (*Operational Conceptual Modeling Language*, linguagem para modelagem conceitual operacional) [Motta 99].

Dentre as linguagens, F-Logic tem se mostrado a mais interoperável e popular entre os editores de ontologia: o OntoEdit [Ontoprise 2003], o WebODE [WebODE 2003] e o Protégé [Noy et al 2000] criam e executam predicados nessa linguagem, este último através de um componente externo.

Nas próximas subseções, serão abordados três ambientes de ferramentas de ontologias: o KSE, o Protégé e o KAON.

4.2. As ferramentas do KSE

O projeto KSE produziu o primeiro conjunto de ferramentas para a manipulação e uso de ontologias a angariar popularidade, por dispor de um leque de funcionalidades bastante amplo. Estas ferramentas tinham por objetivos viabilizar a comunicação de agentes em nível de conhecimento – foi no âmbito do KSE que surgiu esta tecnologia – e proporcionar a edição padronizada e reuso de ontologias, mesmo em formalismos de representação diferentes. As próximas subseções descrevem as ferramentas do KSE.

4.2.1. A linguagem de comunicação entre agentes KQML

A efetivação da troca de conhecimentos entre agentes, independente de formalismos de representação de conhecimento na qual o conhecimento está expresso, foi materializada pela linguagem de comunicação KQML (*Knowledge Query Manipulation Language*, linguagem de manipulação de consultas a conhecimento) [Finin et al 94]. Ela compreende um conjunto extensível de *performativas*, que define as possíveis operações de comunicação em nível de conhecimento entre agentes. Através de mensagens escritas em KQML, os agentes podem trocar conhecimento, enviando sentenças lógicas, fatos e metas, com o intuito de cooperarem e/ou negociarem. Os atos de fala implementados são os atos assertivos (informar), diretivos (pedir ou perguntar), proibitivos e declarativos (causar eventos ao comunicador).

Uma mensagem KQML consiste de uma performativa, seus argumentos obrigatórios – um deles é o conteúdo da mensagem – e outros argumentos opcionais que descrevem o conteúdo, escritos de forma independente da sintaxe da linguagem do conteúdo. No exemplo da figura 6, o Agente CFP, que busca chamadas de trabalhos a conferências – pede (performativa *:ask-all*) ao Agente PPR – que manipula artigos científicos – *links* de páginas cujas âncoras contenham o *conceito* [*call-for-papers*]. Este conceito só é compreendido pelo agente PPR, porque o campo *:ontology* define que a mensagem referencia a ontologia científica *Science*, que contém este conceito. Também os conceitos de *Link*, *anchor* e *URL* estão em ontologias incluídas na *Science*. É importante observar que, neste esquema, para que agentes troquem conhecimento, eles são forçados a referenciar uma *ontologia compartilhada* e conhecida por ambos. Ao achar um link que satisfaz a esta condição, o agente PPR diz, então, (performativa *:tell*) ao agente CFP os dados do *link* encontrado. Como se pode observar, o conteúdo das mensagens foi codificado em JessTab (que, nesse caso, desempenha o papel de linguagem conteúdo), uma adaptação do Jess para a definição de *frames* com todas as suas características.

:

<pre>(ask-all :sender CFP-Agent :receiver PPR-Agent :in-reply-to id0 :reply-with id1 :language JessTab :ontology Science :content (object (is-a Link) (URL ?u) (anchor ?a&: (occurs [call-for-papers] ?a))))</pre>	<pre>(tell :sender PPR-Agent :receiver CFP-Agent :in-reply-to id1 :reply-with id2 :language JessTab :ontology Science :content (object (is-a Link) (URL "http://lcn2002.cs.bonn.edu") (anchor "IEEE Conference on Local Computer Networks (LCN 2002)"))</pre>
--	---

Figura 6: Exemplo de troca de mensagens em KQML.

4.2.2. A *interlingua* KIF e a ontologia de *frames*

A linguagem KIF (*Knowledge Interchange Format*, formato para intercâmbio de conhecimento) foi especificada com dois propósitos: ser uma linguagem conteúdo em mensagens entre agentes, e servir como uma *interlingua* na tradução entre linguagens de formalismos de representação. A tradução via uma *interlingua* economiza a construção de tradutores *ad hoc*, sendo apenas necessários os tradutores de e para a *interlingua*. Assim, se são considerados n formalismos, o número de tradutores cai de $(n-1)^2$ para $2n$ [Baalen & Fikes 93].

KIF representa conhecimento em lógica de 1ª ordem com cálculo de predicados, apresentando a maior expressividade entre as linguagens de ontologia [Corcho & Gómez-Perez 2000]. Ela pode representar meta-conhecimento: através de reificação – ou seja, representando relações como objetos –, é possível, a construção de sentenças sobre sentenças, e, portanto, predicados de 2ª. ordem. Esta forte expressividade, enquanto decisão de projeto, foi tomada para que os outros formalismos pudessem ser facilmente representáveis e, portanto, traduzíveis, nela.

KIF foi feita para ser usada junto com um formalismo de *frames*, que foi definido numa ontologia, a ontologia de *frames* (*Frame-Ontology*). Esta ontologia define classes, instâncias, atributos, facetas, e ainda axiomas, relações ricas e com aridade arbitrária, atributos para instâncias (por exemplo, a classe de animais se extinguem, enquanto suas instâncias - espécimes - morrem), decomposição de classes em partições disjuntas (que não possuem interseções), que podem ser exaustivas (o conjunto de partições forma a classe, por exemplo, os inteiros positivos se dividem em pares e ímpares) ou não.

O binômio KIF-ontologia de *frames*, contudo, apresenta duas desvantagens, que impedem seu uso prático para ontologias. Em primeiro lugar, ainda não existe um motor de inferência para KIF. Além do mais, como todas as ontologias criadas depois instanciam e/ou referenciam a ontologia de *frames*, o código gerado perde a legibilidade (ver sub-seção 4.2.7.).

4.2.3. O editor de *frames* da Ontolingua

Esta ferramenta disponibiliza uma interface de acesso através de navegadores, possibilitando a qualquer usuário da Web a visualização das ontologias criadas e disponíveis, a criação de novas e o trabalho colaborativo. O editor da Ontolingua distingue com bastante propriedade os níveis de conhecimento (ou epistemológico), o nível lógico e o de implementação, criando ainda o nível de apresentação. No nível de conhecimento está o conhecimento propriamente dito em formato texto; o nível lógico pode ser visto como o KIF, pois representa o conhecimento num formalismo, sem a preocupação de como ele será implementado computacionalmente para os processos de inferência; no nível de implementação, a ontologia é representada num determinado formalismo, como CLIPS ou Prolog; e o nível de apresentação resolve conflitos de conceitos homônimos de áreas distintas, que podem causar problemas quando há interconexões entre estas áreas. O nível de apresentação mantém isto transparente ao usuário, dando a real impressão que o sistema “entende” a diferença conceitual entre os conceitos homônimos [Farquhar 96]. Residem nesta característica as raízes do engajamento ontológico de soluções declarativas: na realidade, a forma ideal de interação com os sistemas deve ocorrer no nível de conhecimento e não no de implementação.

4.2.4. Conectividade entre bases de conhecimento: o OKBC

O OKBC (*Open Knowledge Base Connectivity*, em português, conectividade para bases de conhecimento) [Chaudri et al 98] consiste de uma API (*Application Programming Interface*, interface para programação de aplicações) para garantir a interoperabilidade entre diferentes linguagens de representação.

Seguindo o mesmo princípio do ODBC, cada nova linguagem precisa implementar cada método dessa API, de forma que o formalismo fique em conformidade com o OKBC. Feito isto, chamadas OKBC padronizadas a qualquer das linguagens terão a mesma sintaxe e devolverão resultados similares, quando manipularem *frames*. Os métodos da API acessam e retornam dados da base de conhecimento (como o `get-kbs`, que devolve toda a base), classes (por exemplo, `get-classe-supers`, que retorna as superclasses de uma dada classe), atributos (e.g. `get-slot-value`), facetas (e.g. `get-facet-values`), e instâncias (`instance-of`, retornando a classe da instância).

4.2.5. O servidor e o repositório de ontologias da Ontolingua

A arquitetura do servidor Ontolingua está representada na figura 7. Os colaboradores remotos lêem e contribuem com novos conhecimentos para a biblioteca digital de ontologias em texto, mantida em HTML e acessível através de navegadores da Internet. A possibilidade de trabalho colaborativo e de ontologias que servem tanto a pessoas quanto a software consiste num dos maiores benefícios da Ontolingua. As aplicações podem usar as ontologias de duas maneiras: transferindo arquivos de ontologias a partir do repositório – as ontologias podem ser traduzidas para algum dos formalismos com os quais a Ontolingua é capaz de lidar -, ou fazendo consultas e atualizações diretamente às ontologias do servidor, usando o OKBC (na figura, ele ainda se chamava NGFP, ou *Protocolo de Frames Genéricas*).

4.2.6. Consolidando ontologias: o Chimaera

O Chimaera [McGuinness et al 2000], uma ferramenta que surgiu mais adiante, teve por motivação ajudar a resolver discrepâncias entre ontologias similares, evento frequente após a relativa popularização das ontologias. Ela funciona como uma ferramenta para diagnose de ontologias, indicando a ausência de dados obrigatórios, por exemplo, e também para comparação entre ontologias, indicando atributos e classes semelhantes. A ferramenta é mais empregada, no entanto, para combinar ontologias a partir de sua comparação, deixando a cargo do usuário a decisão do que fazer com as semelhanças. Por exemplo, classes com nomes semelhantes podem ser unidas com o mesmo nome, uma delas eliminada ou uma delas considerada como subclasse da outra, etc.

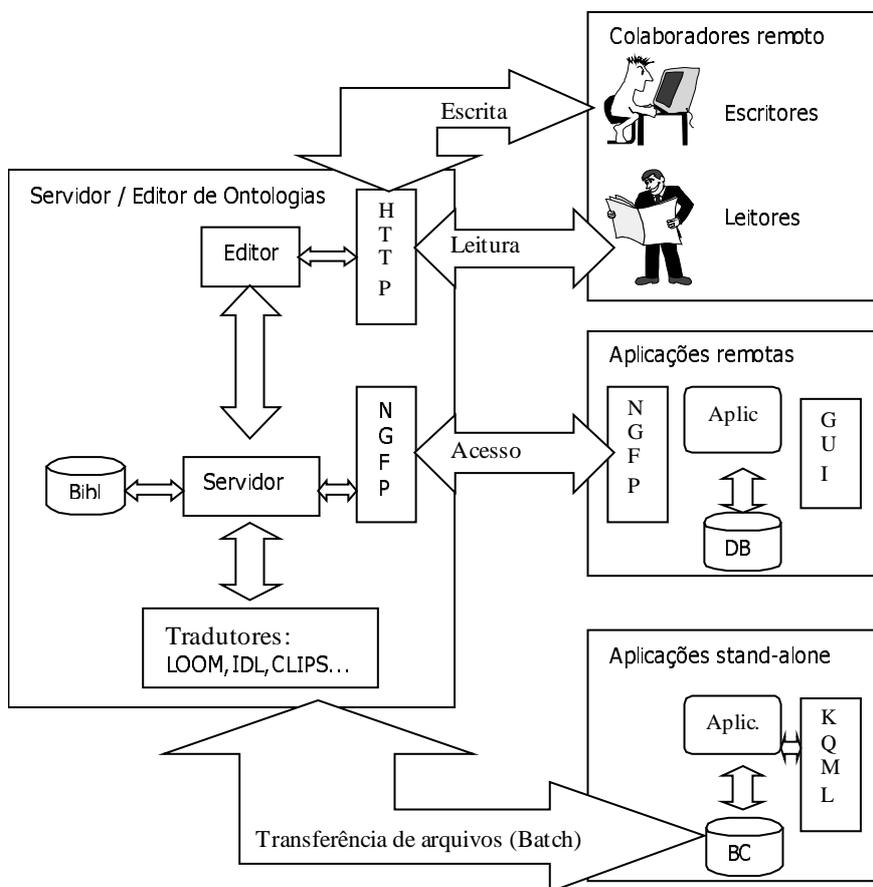


Figura 7. Arquitetura geral do servidor Ontolingua [Farquhar 96].

4.2.7. Problemas da Ontolingua

Mesmo tendo em seu núcleo uma linguagem com alto poder expressivo (KIF), as ferramentas da Ontolingua pecam em relação à usabilidade em vários aspectos. Para permitir definições muito completas, qualquer ontologia criada na Ontolingua deve incluir e referenciar uma ontologia complexa para lidar com *frames*, a *Frame-Ontology* [Gruber 95], e o código gerado ou traduzido pode tornar-se complexo devido às constantes referências às classes desta ontologia. Este erro de requisitos no projeto (a falta de simplicidade na interface e no código gerado), aliados á complexidade das

definições geradas e à falta de um motor de inferência para KIF, impediu que a Ontolingua se difundisse e abriu mercado para o surgimento de ferramentas que sanassem este problema.

Persistem, também, problemas relativos à tradução. Existem diferenças de expressividade entre os formalismos, que dificultam a tradução [Baalen & Fikes 93]. Por exemplo, F-logic [Kifer et al 95] possui restrições para facetas de cardinalidade máxima e mínima [Corcho & Gómez-Pérez 2000], essa informação simplesmente é perdida durante a tradução. Também os diferentes tipos de inferências e premissas das ontologias de topo de muitas linguagens de representação (analogamente à figura 3) normalmente estão refletidas no conhecimento definido, de forma que essas ontologias de topo das linguagens deveriam ser mapeadas entre si para permitir uma tradução e reuso mais seguros, minimizando os erros de tradução [Valente et al 99].

Alem destes problemas, falta à Ontolingua uma boa interface para estações de trabalho, que permitam a usuários, entender, reusar, manusear e verificar ontologias, sem estar necessariamente acessando a Internet. Outros requisitos de uma ferramenta de manipulação de ontologias são a extensibilidade - novas funcionalidades devem poder ser adicionadas ao editor – e a integração, pois nem todas as ferramentas da Ontolingua podem ser usadas e testadas num mesmo ambiente. Aliás, nenhum teste com inferência pode ser rodado, devido à ausência de um motor de inferência, o que faz com que o ambiente como um todo esteja sendo abandonado pela crescente indústria de comércio eletrônico e Web semântica.

4.3. O Ambiente Protégé-2000

Na esteira destes acontecimentos, o ambiente Protégé, que vinha sendo desenvolvido pelo Departamento de Informática Médica da Universidade de Stanford face à necessidade de ontologias médicas desde os anos 80, ocupou o nicho de mercado aberto pela Ontolingua.

Em seu projeto original, o Protégé era uma ferramenta de aquisição de conhecimento limitada a um sistema especialista para oncologia, o Oncocin [Gennari et al 2003]. Ela foi se modernizando para, gradativamente, acompanhar a evolução de tecnologia de SBCs.: servir para aquisição de conhecimento diretamente de especialistas de domínios - com menos dependência de engenheiros de conhecimento -, permitir diversos formalismos e estratégias de inferência, integrar tarefas (aquisição de ontologias e instâncias, ambiente de teste com inferência) num mesmo ambiente, criar automaticamente formulários para entrada de conhecimento, acessar ontologias via OKBC e a combinar de ontologias.

Do ponto de vista de engenharia de software, pode-se considerar que os requisitos corretos foram levados em consideração durante sua evolução. O Protégé sempre procurou crescer em número de usuários, passando por várias reengenharias e reimplementações, e provendo ferramentas simples e configuráveis. A complexidade de seu modelo de conhecimento, por exemplo, é escondida do usuário, que pode ter acesso a ela, se assim o desejar.

A equipe responsável pelo Protégé, ao perceber o potencial de desenvolvimento de seus usuários, optou por abrir seu código. Assim, surgiu uma arquitetura integrável a diversas aplicações, via componentes que podem ser conectados ao sistema. Como

consequência desta decisão e de sua difusão, componentes de vários matizes, elaborados por grupos de pesquisa de usuários, puderam ser adicionados ao sistema, sem necessitar o redesenvolvimento. Foram aproveitados, por exemplo, o Jambalaya, um utilitário com animação e vários outros recursos em visualização de dados, e o Ontoviz, um componente que faz com que o gerador de gráficos Graphviz da AT &T produza gráficos com instâncias, heranças e outros tipos de relacionamento. As figuras 9 a 11 foram geradas pelo OntoViz a partir do Protégé.

A melhor decisão de projeto do Protégé está em seu modelo de conhecimento extensível: é possível redefinir declarativamente as classes primitivas (ou *metaclasses*) de um sistema de representação. Com efeito, um atributo de um *frame* numa classe definida por um usuário instancia uma classe do sistema, a metaclasses `:SLOT`, o mesmo ocorrendo com classes e facetas. O conjunto de metaclasses usados por default pelo sistema implementa características comuns a *frames*, (apresentadas na seção 2.2.1.), tornando-o fácil de usar mesmo para usuários leigos. Todavia, se forem utilizadas metaclasses complexas e distintas das originais - como as para definir classes em RDF, por exemplo - as instâncias alcançarão a expressividade e complexidade desejada. Por isso, o Protégé pode ser adaptado a diversos usos. Esta abordagem verificou-se muito mais flexível do que a ontologia de *frames* da Ontolingua, que é muito complexa, e, se for alterada, muitas ontologias que a referenciam não poderão ser reusadas. Outras características são [Noy et al 2000]:

- A linguagem axiomática PAL (*Protégé Axiomatic Language*), que permite a inserção de restrições e axiomas que incidem sobre as classes e instâncias de uma ou mais ontologias.
- A geração de arquivos de saída alteráveis, permitindo que sejam implementados componentes para traduzir o conhecimento para outros formalismos através das metaclasses. Atualmente podem ser criados classes e instâncias em CLIPS - a base de conhecimento é gerada nativamente para esse motor de inferência, o mais popular quando se iniciou a construção do Protégé, nos anos 80 -, Jess, F-Logic, Prolog, RDF, OIL, XML, Topic Maps (linguagem padrão ISO para descrever estruturas de conhecimento em páginas da Web, nos mesmos moldes de RDF e OIL) – todos estes últimos através de componentes e metaclasses específicas. A saída pode ser armazenada ainda em outros formatos, como bancos de dados relacionais, ainda que as tabelas geradas não sejam normalizadas (vide subseção 8.5.).
- Uma excelente interface para entrada de conhecimento, incluindo um gerador automático de formulários para as classes definidas (vide figura 8), admitindo ainda a reposição da interface original por componentes mais adequados à aplicações específicas. Esta interface facilita sobremaneira o gerenciamento de conhecimento de uma ou mais ontologias.

Do ponto de vista de integração de ontologias, o Protégé oferece várias alternativas. Existem componentes que integram grandes ontologias como o WordNet – um dicionário semântico da língua inglesa -, o GATE (*General Architecture for Text Extraction*, em português, arquitetura geral para extração de texto, um conjunto de ferramentas de linguagem natural para o inglês confeccionadas pela Universidade de Sheffield, Inglaterra), o Cyc e a UMLS (uma volumosa ontologia médica). Está

disponível, também, um repositório de ontologias e um componente-cliente que conecta servidores OKBC, permitindo que todas as ontologias da Ontolingua possam ser acessadas e copiadas (inclusive partes) diretamente para o ambiente Protégé da máquina-cliente.

The screenshot shows a window titled "sc-evt:nr slots & concepts (Case)". The window contains several input fields and lists. On the left, there is a "Description" field with the text "sc-evt:nr slots & concepts". Below it is a checkbox labeled "All-of-Concepts-Ke..." and a "Nr-of-Slots" field with the value "3". There is an "Importance" dropdown menu currently set to "HIGH", with a list of options: "HIGH", "MEDIUM", and "LOW". Below that is a "Keywords" field. On the right side, there are three list boxes: "Concepts", "Concepts-in-the-Beginning", and "Slots". The "Concepts-in-the-Beginning" list contains: "call for participation", "conference", "cfp", "meeting", and "workshop". The "Slots" list contains: "Acceptance-Date", "Accommodation-Places", "Final-Date", "Final-Paper-Due", and "Deadline". Each list box has "V", "C", "+", and "-" buttons. A "Select Instances" button is also visible next to the "Concepts-in-the-Beginning" list. At the bottom right, there is another list box labeled "Slots-in-the-Beginning" which is currently empty.

Figura 8. Formulário automático para entrada de dados gerado pelo Protégé.

4.4. As ferramentas do KAON e da Ontoprise

O grupo de ontologias da cidade de Karlsruhe, Alemanha, que congrega hoje o Instituto de Métodos Formais do Departamento de Economia, o Centro de Pesquisas em Informática (Forschungszentrum Informatik - FZI) e uma empresa, a Ontoprise, é um dos mais prolíficos em produção científica e prática na área de ontologias, tendo inclusive criado o primeiro protótipo do que viria a ser a Web semântica, o Ontobroker [Benjamins et al 98].

Este grupo desenvolveu uma suíte de ferramentas para ontologia e Web semântica chamada de KAON (*the KARlsruhe ONtology and semantic web tool suite*) [KAON 2003], e ferramentas comerciais para a aplicação de ontologias em comércio eletrônico, gestão de conhecimento em empresas e Web semântica. Entre elas estão um editor de ontologias (o OntoEdit), um servidor de ontologias baseado em bancos de dados – que respondeu em no máximo 5 segundos a testes com 100.000 conceitos, 1 milhão de instâncias e 66.000 atributos –, uma ferramenta para criação de ontologias a

partir de texto (o Text-to-Onto [Maedche & Staab 2000]), outra para busca sobre bases de texto baseada em ontologias (SemanticMiner [Ontoprise 2003]), outra para anotação semi-automática de referências a ontologias em páginas para a Web, e outra para agrupamento de textos baseados em ontologias.

Um dos produtos comerciais mais inovadores consiste de um assistente que sugere e provê informações relacionadas a ontologias para ajudar a digitação de textos no Word, o OntoOffice [Ontoprise 2003]. Assim, quando se digita, por exemplo “o gerente da divisão técnica”, o produto automaticamente exibe as informações sobre esse funcionário na tela. Como se vê, o grupo continua a alargar as fronteiras para as aplicações com ontologias.

5. Engenharia de ontologias

A concepção de ontologias deve ser conduzida como qualquer outro projeto de software, no sentido de serem tomadas decisões de projeto que determinam sua qualidade baseada em critérios como eficiência, legibilidade, portabilidade, extensibilidade, interoperabilidade e reuso. Por isso, tal concepção deve basear-se em seu futuro emprego, e não somente em aspectos filosóficos do conhecimento acerca do domínio representado. Nesta seção veremos, em primeiro lugar, princípios para a construção de ontologias visando maximizar o seu reuso, e depois técnicas de desenvolvimento de ontologias.

5.1. Princípios de construção de ontologias

Alguns princípios, se usados com precisão, garantem sua qualidade [Gruber 93]:

- *Clareza*: Os programas usam diferentes modelos e abstrações na resolução de seus problemas. Na definição do conhecimento, deve-se ter a objetividade de definir apenas o que se presume ser útil na resolução da classe de problemas a ser atingida. Definições completas, com condições necessárias e suficientes, devem ter precedência sobre definições parciais.
- *Legibilidade*: As definições devem guardar correspondência com as definições correntes e informais. A ontologia deve usar um vocabulário compartilhável – normalmente o jargão e terminologia usados por especialistas do domínio.
- *Coerência*: As inferências derivadas da ontologia definida devem ser corretas e consistentes do ponto de vista formal e informal com as definições.
- *Extensibilidade*: A ontologia deve permitir extensões e especializações monotonicamente e com coerência, sem a necessidade de *revisão de teoria*, que consiste na revisão lógica automática de uma base de conhecimento em busca de contradições.
- *Mínima codificação*: Devem ser especificados conceitos genéricos independente de padrões estabelecidos para mensuração, notação e codificação, garantindo a extensibilidade. Essa genericidade é limitada pela clareza. Esta característica se contrapõe à programação imperativa orientada a objetos, porque, por exemplo, atributos de objetos trazem tipos básicos e informações sobre a implementação.

- *Mínimo compromisso ontológico*: Para maximizar o reuso, apenas o conhecimento essencial deve ser incluído, gerando a menor teoria possível acerca de cada conceito, e permitindo a criação de conceitos novos, mais especializados ou estendidos.

Esses conceitos podem ser evidenciados através de um exemplo de ontologia para o domínio científico [Freitas 2001], gerada a partir do reuso com vários refinamentos da ontologia do projeto europeu (KA)² (*Knowledge Annotation Initiative of the Knowledge Acquisition Community* – Iniciativa de Anotação de Conhecimento da Comunidade de Aquisição de Conhecimento) [Benjamins et al 98].

O projeto (KA)² teve por objetivo criar uma organização virtual de pesquisadores, universidades, projetos e publicações, entre outros itens, envolvidos com a subárea de Inteligência Artificial conhecida como Aquisição de Conhecimento. Nesta ontologia pode-se verificar o cuidado com que os autores conceberam os conceitos de forma a representar apenas os essenciais, guardando uma correspondência direta - engajamento ontológico - com o domínio e terminologia existentes, satisfazendo os requisitos *clareza e legibilidade*.

Como exemplo de *mínimo compromisso ontológico*, observe-se a definição da classe Documento Científico, apresentada na figura 9. Pode-se perceber que não há restrições desnecessárias na definição do atributo Autores. Poder-se-ia especificar que subclasse da classe Pessoa pode ser autor de artigos, como, por exemplo, a subclasse Pesquisador. Porém, especificou-se exclusivamente o conhecimento do conceito sem definir prematuramente certas decisões.

Scientific-Document		
Publication-Year	Instance	Year
Authors	Instance*	Person
Keywords	String*	
URL-of-Publication	String	
Publication-Title	String	
...		

Figura 9. Parte dos atributos da Classe Documento Científico⁴.

A ontologia do (KA)² possuía *extensibilidade*, por isso, foi alterada de forma que novas classes pudessem ser definidas a partir das já existentes. Como mostra a figura 10, foram criadas as classes abstratas (sem instâncias) Evento Científico ao Vivo (*Live-Scientific-Event*) - com subclasses (não mostradas na figura) Conferência e Workshop – e Evento de Publicação Científica (*Scientific-Publication-Event*), com subclasses Jornal, Revista (*Magazine*) e Evento de Publicação Científica Genérica (*Generic-Scientific-Publication-Event*).

Em relação à *coerência*, foi percebido que a relação meronímica (ou parte-todo) entre artigos de um *proceedings*, ou entre capítulos de um livro, não estava explicitada.

⁴ Nessa e na figura 11, o uso do asterisco (*) após a definição do tipo de um atributo assinala que ele pode assumir múltiplos valores, e não um só como os outros atributos.

Dado que seria importante diferenciar artigos e capítulos e, por outro lado, evidenciar que um determinado conjunto deles pertence a um mesmo livro ou *proceedings*, foram criadas as classes Publicação-Divisível (*Dividable-Publication*) - com as subclasses Livro (*Book*), *Proceedings* e Evento de Publicação Científica (*Scientific-Publication-Event*) - e Publicação-Parte, com subclasses (não mostradas na figura) como Artigo de Workshop, Artigo de Conferência, Capítulo de Livro, etc.

Observe-se o engajamento ontológico causado pelo uso da herança múltipla na classe Evento de Publicação Científica: Jornais e Revistas ao mesmo tempo em que são Publicações Divisíveis, guardam são, também, eventos científicos, pois possuem comitês de programa, datas limite, e outros atributos, embora não ocorram ao vivo.

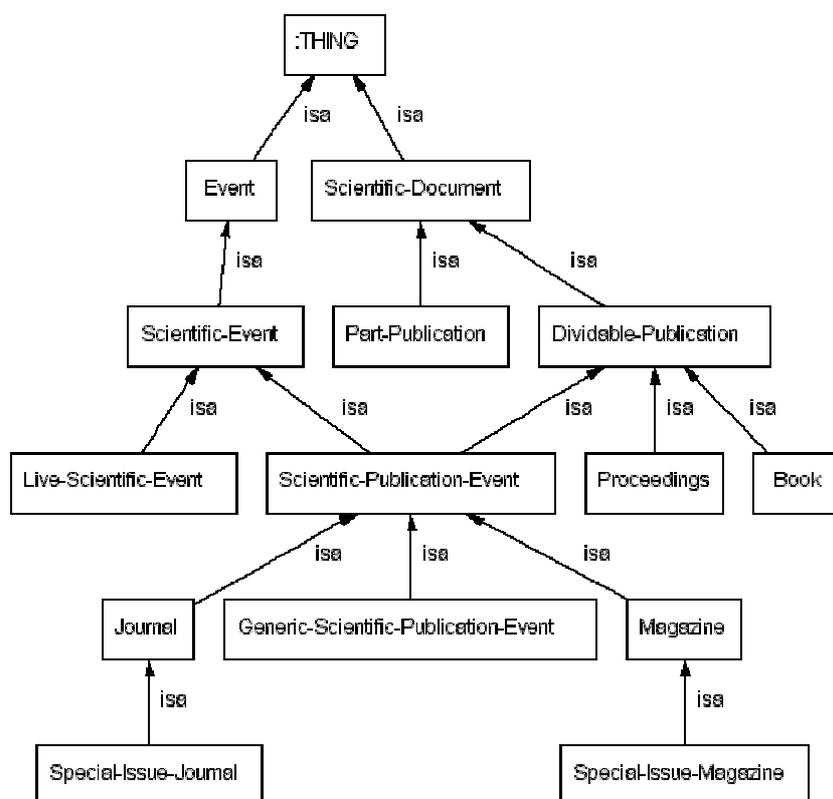


Figura 10. Parte da ontologia Ciência [Freitas 2001], salientando as classes Evento e Publicação, suas subclasses e heranças.

A figura 11 apresenta uma gama maior de relacionamentos, mostrando como se entrelaçam as classes principais da ontologia, incluindo também classes de uma ontologia auxiliar de tempo, como Mês (*Month*) e Ano (*Year*). Constata-se, por exemplo, que os membros de um comitê de programa da classe Evento são instâncias da classe *Researcher* (Pesquisador), através do atributo *member-Of-Program-Committee*.

5.2. Metodologias de desenvolvimento

Começam a despontar as primeiras metodologias, levando sempre em conta que a concepção de ontologias requer um processo iterativo, com revisões constantes. Nas metodologias propostas, são considerados passos similares aos de engenharia de software: *especificação*, *conceitualização*, e *implementação* [Gómez-Perez 99].

À parte deste processo, algumas *atividades de suporte* são executadas concomitantemente com o desenvolvimento, chamadas de atividades de suporte: a *aquisição* – correspondendo à elicitacão de conhecimento necessário à construção da ontologia -, a *avaliação* – que verifica se a ontologia atende aos requisitos e propósitos planejados -, a *documentação* e a *integração* com ontologias existentes. A figura 12 apresenta as etapas de desenvolvimento e sua relação com as atividades de suporte.

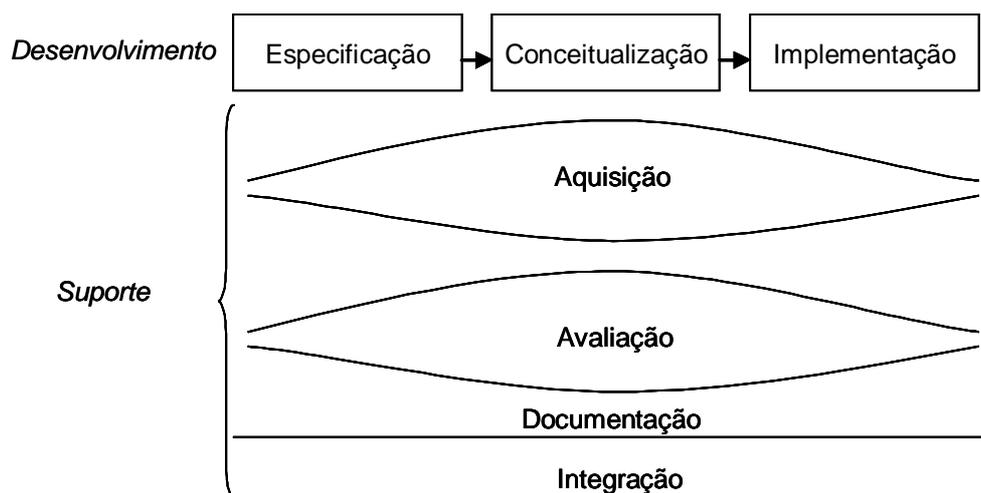


Figura 12. Etapas de desenvolvimento e sua relação com as atividades de suporte [Gómez-Perez 99].

Voltando às fases do desenvolvimento, a primeira fase, a *especificação*, determina o propósito e escopo da ontologia a ser confeccionada. Esta fase deve incluir uma análise para decidir se é possível, necessário ou adequado o reuso de ontologias já existentes. Para auxiliar a determinar o escopo, vale a pena elaborar uma lista de *questões de competência* (*competency questions*) [Uschold & Gruninger 96], que servirão para a avaliação da ontologia durante o desenvolvimento. Por exemplo, “até que ponto Jornais científicos são considerados eventos científicos” é uma questão de competência.

A fase de *conceitualização* é a mais crítica do desenvolvimento, pois, sendo a mais ligada à definição do conhecimento, nela ocorrem a maior parte das atividades de suporte de aquisição e avaliação. O tutorial do Protégé [Noy 97] lista passos e várias dicas úteis sobre esta fase. Algumas estão dispostas abaixo:

- *Enumerar os termos do domínio*, sem preocupar-se com similaridade, repetições e relações entre ele. Costuma-se usar o processo de *brainstorming* para este fim.
- *Definir as classes*. É preciso não confundir nomes de um conceito com o próprio, inclusive existem sistemas que permitem a inclusão de sinônimos e termos associados a conceitos de uma ontologia.
- *Definir a hierarquia das classes*. Ocorrendo junto com a anterior, esse constitui o passo mais capcioso do desenvolvimento, devido às sutilezas das hierarquias. Ao serem criadas subclasses, deve-se observar a clareza e consistência da hierarquia. A clareza refere-se ao fato de uma classe ter subclasses demais –

neste caso é preciso verificar se o uso de classes intermediárias não seria adequado -, ou de menos – caso em que a informação dos atributos pode tornar-se insuficiente para refletir diferenças entre as instâncias. Uma maneira efetiva é considerar se entre a superclasse e a subclasse não é plausível existirem outras classes. Três abordagens para a definição de hierarquias são sugeridas [Uschold & Gruninger 96]: a *top-down*, definindo as classes mais gerais e depois as específicas, a *bottom-up*, que faz o contrário, e a *middle-out*, que começa por classes no meio que vão sendo especializadas (para baixo) e generalizadas (para cima).

- *Definir os atributos e facetas de cada classe.* Este passo deve interagir com os dois anteriores, pois são os novos atributos ou facetas que definem uma classe, exceto em classes terminológicas, como, por exemplo, em medicina. Os atributos podem ser intrínsecos (e.g., número de pernas), extrínsecos (nome de uma pessoa), partes de uma classe (e.g. partes do corpo, cabeça, tronco e membros, vide subseção 8.2. para maiores detalhes), ou relacionamentos, que são instâncias de outras classes, por exemplo, o atributo Pesquisadores da classe Projeto, que são instâncias da classe Pesquisadores. Note-se que, para este exemplo, a faceta de classes-permitidas especificou a classe mais geral, que incluía todas as subclasses de pesquisadores (estudantes de pós-graduação, professores, etc).
- *Criar as instâncias*, tendo como lema que são os conceitos mais específicos de uma ontologia, ou seja, os elementos separados por menor granularidade. Se estes conceitos possuem uma hierarquia natural, é preciso revisar a definição das classes.
- Em relação a *convenções de nomes*, estes devem ser facilmente compreensíveis, recomendando-se convenções diferentes para classes, atributos e instâncias. Devem ser evitadas abreviações, que confundem os usuários, pois a ontologia deve ser legível para as pessoas que as consultam.

A fase de *implementação* transformará a ontologia em algo computável. Corresponde à geração de arquivos numa linguagem de representação de conhecimento, e é seguida pela fase de *avaliação*, que executa testes para verificar se a ontologia atende aos requisitos especificados na fase de especificação. Estas duas fases interagem, de forma que os testes podem provocar mudanças na implementação.

6. Aplicações de Ontologias

Ontologias estão sendo aplicadas com sucesso em áreas em que a necessidade de uso de contexto, e em especial de comunicação contextualizada, se fazem sentir. Com efeito, agentes baseados em ontologias vêm sendo testados em diversos campos, como comércio eletrônico, gestão de conhecimento, *workflow* e tratamento inteligente de informação. Seguem, abaixo, dois exemplos destes agentes, o primeiro para problemas de engenharia em fábricas geograficamente distribuídas, e o segundo para tratamento contextualizado de textos na Web.

6.1. O PACT

O PACT (*Palo Alto Collaboration Testbed*, ou ambiente de teste de colaboração de Palo Alto) [Cutovsky et al 93] converteu-se no primeiro teste prático de uso de ontologias como vocabulário de comunicação, em 1993. Teve por objetivo a resolução negociada de problemas de projeto, fabricação e revisão de projeto de manipuladores robóticos envolvendo sistemas já existentes de fábricas da HP e da Lockheed e de uma empresa de consultoria, responsável pelo desenvolvimento do software (vide figura 13).

Os agentes tinham como vocabulário comum duas ontologias, uma de matemática para engenharia - com classes como quantidades, unidades, dimensões, matrizes e funções - e outra de projeto e configurações - com as classes parâmetros, componentes, restrições e outras. Mais abaixo, uma mensagem KQML trocada entre dois agentes:

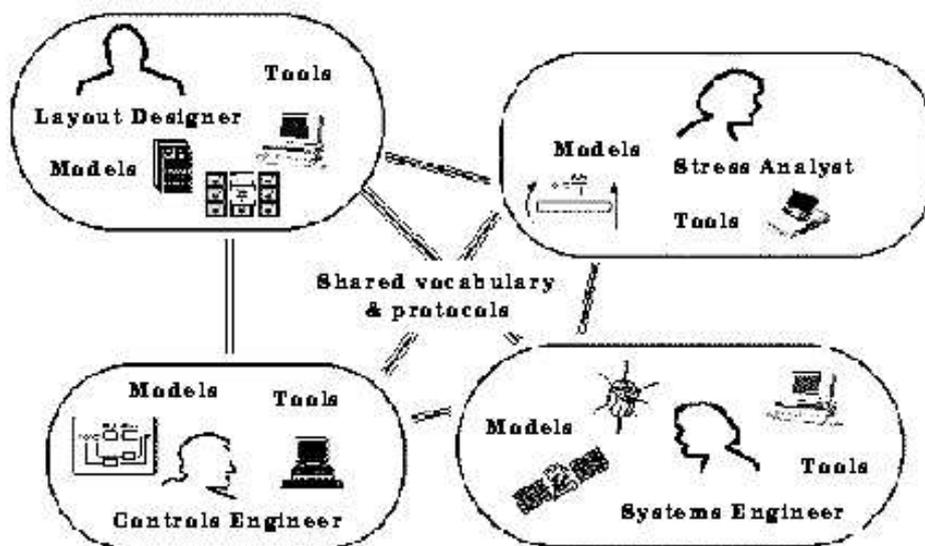


Figura 13. Agentes distribuídos dialogam para resolver problemas de projetos de engenharia [Fikes 98].

```
(monitor :from consumer :to producer :reply-with update-111
:ontology standard-units-and-dimensions
:language KIF
:content (= (q.magnitude (diameter shaft-a) inches) ?x))
```

A intenção "monitor" pede para manter o agente remetente informado sobre mudanças no campo conteúdo, expresso em KIF. Assim, qualquer mudança no diâmetro do eixo da peça q será informada ao agente remetente. Note o uso de conceitos como magnitude, diâmetro e polegada, que é uma instância de medida,

6.2. MASTER-Web: Manipulação integrada de informação através de ontologias

Foi proposta uma arquitetura de sistemas multiagentes [Freitas & Bittencourt 2003] para manipular a informação referente a um conjunto de classes sobre um mesmo grupo, como, por exemplo, o grupo científico, com classes como artigos científicos, eventos, pesquisadores, etc. A arquitetura visa recuperar, classificar e extrair dados de páginas pertencentes às classes a um grupo, e a motivação principal para o emprego de sistemas

multiagentes é beneficiar-se dos relacionamentos entre as classes. A visão geral da arquitetura está ilustrada na figura 14.

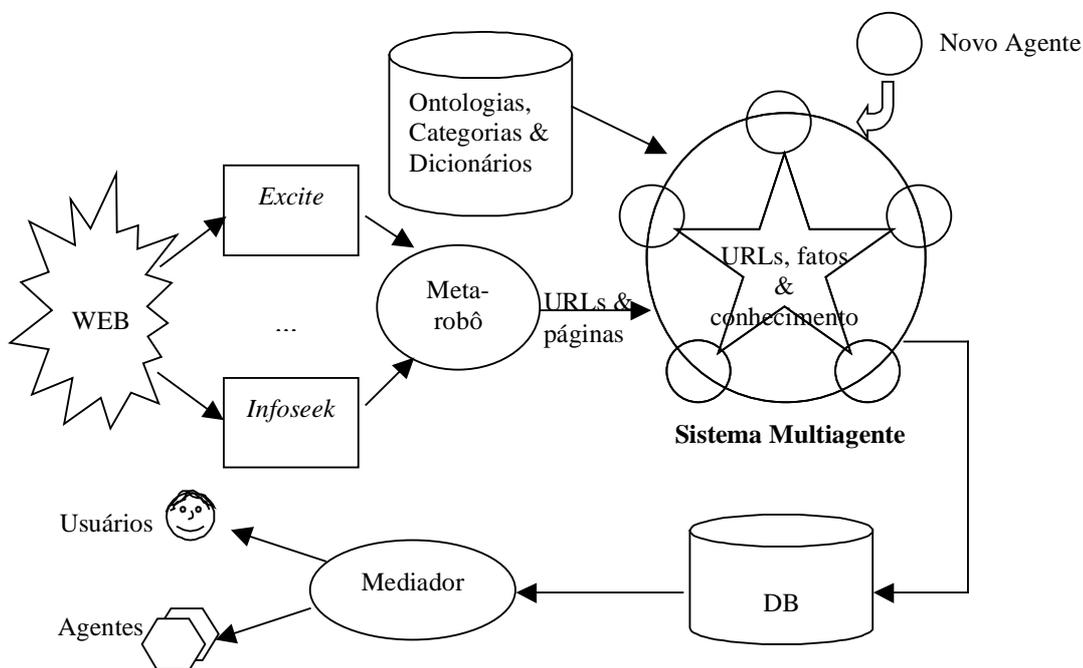


Figura 14. Arquitetura de sistemas multiagentes para manipulação integrada de informação de grupos de classes de páginas.

Na figura, a estrela indica troca de mensagens contendo regras de reconhecimento e fatos (conhecimento dos agentes), além das URLs sugeridas entre os agentes. Cada agente possui um meta-robô, que consulta conecta-se a múltiplos engenhos de busca - como *Altavista*, *Excite*, *Infoseek* e outros - com palavras-chave que garantem cobertura em relação à classe de páginas processada pelo agente. (e.g., os termos '*call for papers*' e '*call for participation*' para o agente CFP). As classificações e dados extraídos das páginas são postos num banco de dados, acessível por um mediador.

As ontologias servem não só como vocabulário de comunicação entre agentes, como também na definição e organização apropriadas de conceitos, relações, e restrições. A ontologia do domínio deve ser bastante detalhada para garantir precisão à classificação por conteúdo. Como estudo de caso, a arquitetura foi testada contra o grupo científico, usando a ontologia mencionada na subseção 5.1. Foram elaborados o agente CFP, que trata páginas de chamadas de trabalhos ("*Call for papers*") de eventos científicos, e o agente PPR, que trata as páginas de artigos científicos. As páginas são classificadas em várias subclasses dentro da ontologia; por exemplo, o agente CFP as classifica em Conferência, Workshop, Jornal, Revista, Evento-Genérico-ao-Vivo, Evento-Genérico-de-Publicação e Edição-Especial-de-Jornal e de Revista.

Outra vantagem de se usar ontologias reside no ganho de expressividade e flexibilidade, uma vez que o conhecimento sobre uma classe não se circunscreve a termos e palavras-chaves como em engenhos de buscas, mas a *qualquer* fato que diga respeito às páginas, como estrutura, regiões, conceitos contidos nelas. Abaixo duas mensagens em KQML em que o agente CFP primeiro informa um caso com alguns

conceitos e depois pede links de páginas (variável ?l) já classificadas como Artigos-de-Conferência, que estejam antes do “abstract” e conttenham algum dos conceitos do caso.

```
(inform :sender cfp :receiver ppr :language JessTab :ontology Science
:content ([ppr_00008] of Case (Description "cfp suggestions")
(Concepts [Call-for-participation] [annual]
[conference][Call-for-papers] [workshop])))

(ask-all :sender cfp :receiver ppr :language JessTab :ontology Science
:content (object (is-a Anchor) (Link-Text ?l))
(Result (Page-Status CLASSIFIED) (Class "Conference-Paper"))
(object (is-a Web-Page) (Contents ?co))
(test (and (if-occur ?l (begin-until "abstract" ?co))
(if-occur (slot-get [ppr_00008] Concepts) ?l))))
```

Todos os testes conectando os agentes diretamente a engines de busca da Web tiveram taxa de acerto no reconhecimento (e.g. se uma página é um evento) e classificação (que tipo de evento) em mais de 80%.

7. A Web Semântica

A principal aplicação e benefício das ontologias consiste em prover semântica à Internet. A visão de uma Internet semântica tem sido apregoada por ninguém menos que um dos idealizadores da Internet, Tim Bernes-Lee. A revista PC Week [PC Week 2000], apresentou uma retrospectiva da Internet, que finaliza com este possível advento. Segundo a reportagem, a primeira geração da rede, a Internet, permitia apenas a troca de dados entre *máquinas* distintas. A segunda geração, a *World Wide Web* (teia de alcance mundial), ou simplesmente Web, provocou uma revolução por disponibilizar uma vasta gama de aplicativos e informação para as *pessoas*, tornando possível, ainda, o comércio eletrônico entre clientes e empresas (no jargão de negócios, *business-to-clients* ou *b2c*). Porém, como as páginas só possuem informação léxica, mesmo os agentes e/ou robôs dotados de inferência encontram um ambiente hostil para a realização de suas tarefas, porque tanto o conteúdo das páginas como o relacionamento entre elas é difícil de ser compreendido por entidades de software, por encontrarem-se, em sua maioria, em linguagem natural. As pessoas, em consequência, sofrem com essa falta de definições semânticas: por vezes os usuários possuem dados parciais sobre a informação que procuram e não podem utilizá-los para localizar esta informação.

A próxima geração da rede está sendo batizada de *Web semântica*. Sua maior motivação é transformar os dados e aplicativos em elementos úteis, legíveis e compreensíveis para o *software*, ou, mais exatamente, para os *agentes inteligentes*, de forma a facilitar-lhes a comunicação dinâmica, a cooperação e o comércio eletrônico entre empresas (*business-to-business* ou *b2b*).

7.1. As camadas da Web Semântica propostas pelo W3C

O Consórcio da Web ou W3C (*World-Wide Web Consortium*) tem se inclinado a padronizar novas linguagens para definição de páginas e seus respectivos padrões, baseadas em conhecimento estruturado em ontologias. Em 2000, o W3C apresentou uma proposta [Koivunen & Miller 2001] definindo várias novas camadas para a Web, e sugerindo linguagens e padrões para as camadas, delineadas na figura 15.

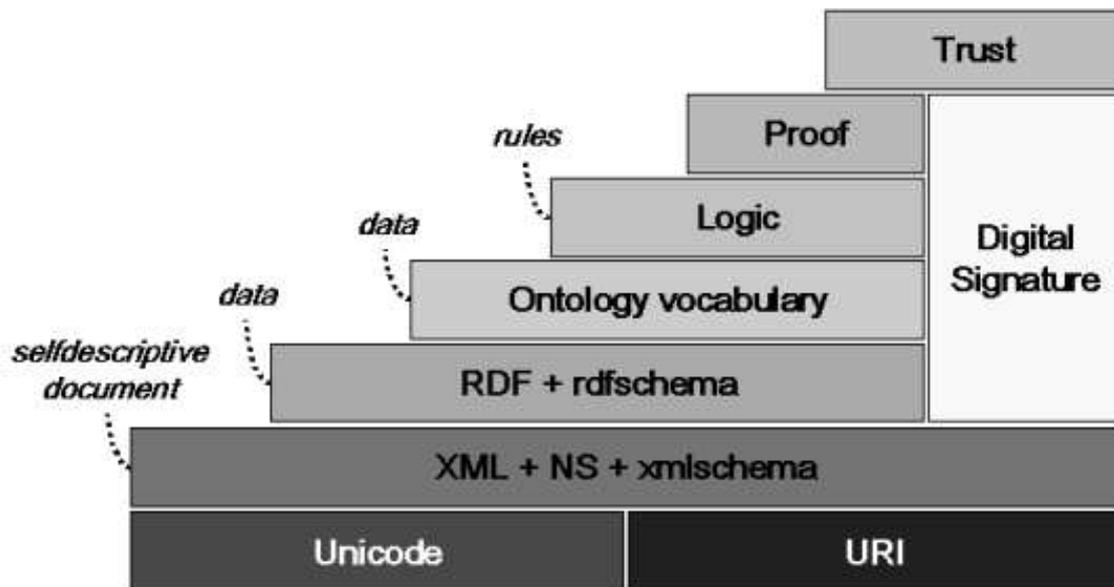


Figura 15. As camadas da Web semântica [Koivunen & Miller 2001].

7.2. A camada Unicode

A primeira camada garante o uso padronizado dos mesmos conjuntos de caracteres (Unicode) e uma forma unívoca para a identificação e localização de páginas (URI – *Uniform Resource Indicator*, indicador uniforme de recursos).

7.3. A camada XML

Vejamos um trecho de código de uma página que discorre sobre um livro, codificado na atual linguagem de anotação de páginas HTML, (adaptado de [van der Vilt 2000]):

```
<html>
  <body>
    <h2>Being a Dog Is a Full-Time Job</h2>
    <p> by Charles M. Schulz</p>
    <p>ISBN: 0836217462</p>
  </body>
</html>
```

Como se pode perceber, HTML possui severas limitações: não existem recursos na linguagem para anotação semântica – os comandos `<h2>` e `<p>` são apenas de editoração, para aumentar o tamanho da fonte e pular linha. Estes recursos trariam significado aos dados da página. Para descrever algum tipo de semântica das páginas, a linguagem HTML provê apenas *tags* opcionais como título, descrição, sumário e palavras-chave. Devido a essa deficiência, HTML foi abstraída para XML (*eXtensible Markup Language*), que, na verdade é uma *meta-linguagem* de editoração, pois permite a representação de outras linguagens de forma padronizada. Vejamos alguns recursos de XML, através do mesmo exemplo:

```
<library>
  <book>
    <title>Being a Dog Is a Full-Time Job</title>
    <author>Charles M. Schulz</author>
    <isbn>0836217462</isbn>
  </book>
```

</library>

No exemplo, os dados estão descritos por *elementos*, o que facilita seu tratamento, se o software que trata a página conhece este formato. Este formato é especificado por *definições de tipos de documentos* (DTDs), ou por esquemas XML (XML Schemas ou XMLS). Um DTD define o aninhamento léxico de um documento, as classes e os atributos destas, valores default para estes (opcional) e a ordem de aparecimento dos dados das instâncias das classes definidas pelo DTD. Abaixo, temos o DTD da biblioteca (*library*) que definiu o livro mencionado como parte da biblioteca:

```
<!DOCTYPE library [  
  <!ATTLIST book id ID #IMPLIED>  
  <!ATTLIST author id ID #IMPLIED>  
  <!ATTLIST ISBN id ID #IMPLIED>]>
```

DTDs definem a estrutura e sintaxe de um documento, ajudando a validar se ele está em conformidade com uma estrutura.. Esquemas XML têm a mesma função, mas são um pouco mais ricos. Pode-se definir tipo e formato exato dos atributos, número exato de instâncias de um aninhamento e há mecanismos de inclusão e derivação que proporcionam o seu reuso. Estas capacidades ajudam a reduzir a distância entre DTDs e ontologias, pois podem ser introduzidos meios para restringir estruturas de documentos e conteúdos, herança para elementos e atributos, tipos primitivos e outras características úteis [Fensel et al 2001]. A idéia é que esta camada descreva a estrutura do documento, deixando para as que estão acima dela a definição de seu conteúdo.

7.4. A camada RDF

RDF (*Resource Description Framework*, ou modelo de descrição de recursos) adiciona mais semântica a um documento, com a vantagem de não precisar referir-se à sua estrutura [Fensel 2001]. RDF descreve “recursos” da Web, que devem ter um identificador na Web, seja ele parte específica de um documento ou dados como lugares, pessoas, etc [Koivunen & Miller 2001].

Para expressar algo sobre os recursos, o modelo de dados de RDF equivale em termos formais às redes semânticas. Os recursos são descritos como trios de objetos-atributos-valores, semelhantes ao sujeito-verbo-objeto das redes semânticas. Os objetos são recursos e os valores são recursos ou strings. Trios descritos em RDF podem ser representados como grafos diretos rotulados [Klein 2001]. Segue um exemplo de grafo ligando duas páginas da Web:

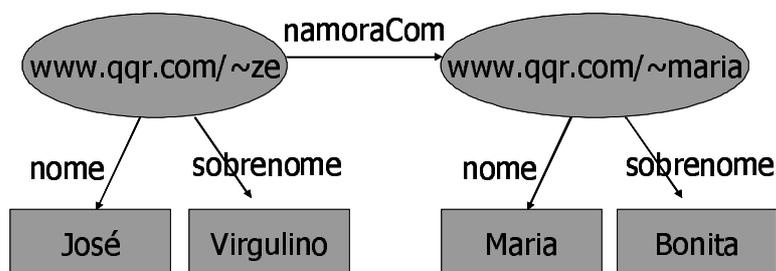


Figura 16. Grafo direto rotulado de um trio RDF.

O grafo acima indica a anotação de que o dono da primeira página namora com a dona da segunda. Ele tem o seguinte código RDF:

```

<rdf:Description about=http://www.qqr.com/~ze>
  <nome>Jose</nome>
  <sobrenome>Virgulino</sobrenome>
  <namoraCom>
    <rdf:Description about=http://www.qqr.com/~maria>
      <nome>Maria</nome>
      <sobrenome>Bonita</sobrenome>
    </rdf:Description>
  </namoraCom>
</rdf:Description>

```

A vantagem de RDF como linguagem de descrição de recursos sobre DTDs e esquemas XML reside na liberdade de ignorar as imposições da estrutura do documento, referindo-se apenas a dados sobre o conteúdo. Para uma padronização de uso de RDF, foram criados os esquemas RDF (RDF-Schemes ou RDFS), que fornecem tipos básicos para a criação de esquemas voltados à aplicações específicas. As primitivas a serem usadas para modelar novos esquemas RDF incluem classe, subclasse (herança), propriedade, sub-propriedade (para construir hierarquias de propriedades), instância e restrição. A classe Pessoa e o atributo nome do exemplo acima são assim definidos:

```

<rdf:Description ID="Pessoa">
  <rdf:type resource="http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Class">
  <rdfs:subClassOf rdf.resource="http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Resource">
</rdf:Description>

<rdf:Description ID="nome">
  <rdf:type resource="http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Property">
  <rdfs:domain rdf.resource="#Pessoa">
  <rdfs:range rdf.resource="http://www.w3c.org/TR/xmlschema-2#string">
</rdf:Description>

```

O código diz que Pessoa é do tipo classe e herda as características de um recurso. Observe que, pelo código, tanto classe como recurso estão definidos por URIs que apontam para o sítio da W3C. Já o atributo nome é uma propriedade cujo domínio são elementos da classe Pessoa e cuja imagem é uma string. Também propriedade e string estão definidos URIs que apontam para no sítio da W3C. Como se vê, especificar código RDF pode ser um pouco complicado à primeira vista. Existem ferramentas para edição e *parsing* sobre RDF, que têm a finalidade de auxiliar o usuário .

Para esta camada, existem outras linguagens candidatas, como a precursora SHOE (*Simple HTML Extensions*, ou simples extensões de HTML) [Luke et al 96] e a XOL (*Ontology eXchange Language*, ou linguagem de trocas de ontologias) [Karp et al 99], baseada no OKBC. Porém RDF/RDFS tornaram-se padrões de fato, apesar da expressividade de representação deste par ainda deixar a desejar quanto à modelagem de ontologias.

7.5. A camada de ontologias

Esta é camada mais importante e pesquisada da Web semântica. Ela é responsável por oferecer a expressividade necessária à representação de ontologias. Isso é feito

aproveitando a extensibilidade de RDFS para definir restrições complexas e outras construções que implementam características de *frames* e lógica de descrições.

Já existe uma linguagem padrão, a OWL (*Web Ontology Language*, ou linguagem de ontologias para a Web) [Smith et al 2003], que, na verdade, deriva de um consenso entre duas propostas, a européia OIL (*Ontology Inference Layer*, camada de inferência para ontologias, ou ainda, *Ontology Interchange Language*, linguagem de intercâmbio em ontologias) e a DAML (*DARPA Agent Markup Language*, linguagem de anotação para agentes do Departamento de Defesa dos Estados Unidos). OIL foi a primeira destas linguagens, e teve como principal requisito a facilidade de adoção por parte dos desenvolvedores, servindo principalmente à comunidade ligada à Web semântica [Horrocks et al 2000]. Para essa finalidade, seu projeto incorporou as seguintes características:

- Permitir definições baseadas em *frames* implementando as primitivas definidas pelo OKBC, mas também definições em lógica de descrições. As definições de ontologias são geradas sobre XML e RDF.

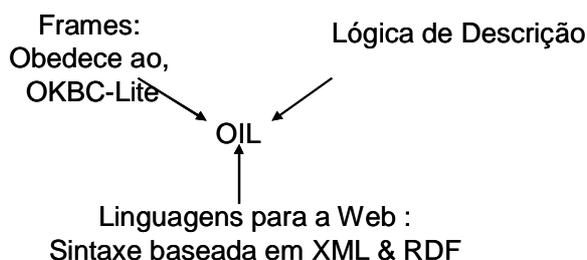


Figura 17. Origens da linguagem OIL [Fensel 2001].

- Modularidade: OIL era dividida em camadas de acordo com a complexidade que se deseja de seu uso, garantindo o uso mais simples pelos usuários. A divisão básica era composta por uma camada de núcleo (*core OIL*), que corresponde a RDFS - porém sem permitir reificação (transformar relações em objetos de uma linguagem), como em RDFS -, e outras camadas para permitir definições compatíveis com *frames* e lógica de descrições [Fensel et al 2001].
- Inferência: foi sacrificado um pouco da expressividade da Ontolingua, tendo, em contrapartida, um motor de inferência consistente, completo e eficiente, capaz de manipular tanto *frames* quanto lógica de descrições.

A linguagem DAML, um investimento de milhões de dólares do DARPA estendeu OIL sem, no entanto, prover um motor de inferência. Mais adiante, as duas linguagens foram unidas, criando a linguagem DAML+OIL [McGuinness et al 2002], que posteriormente seria acrescida de requisitos de internacionalização (Unicode) e de apresentação e documentação (rótulos para axiomas, nomes locais únicos, etc), originando a linguagem OWL. OWL seguiu a receita modular de OIL, dividindo suas classes em três sublinguagens, de acordo com sua expressividade [Smith et al 2003]:

- *OWL Lite* abrange a expressividade de *frames* e lógica de descrições, com algumas restrições. Por exemplo, a cardinalidade máxima ou mínima assume apenas os valores 0 ou 1. Apesar disso, a linguagem é dotada de riqueza

semântica, sendo, por isto, ideal para usuários iniciantes e desenvolvedores que preferem *frames* a lógica de descrições.

Atributos (aqui chamados de propriedades) podem ter transitividade, simetria, atributos inversos, propriedades funcionais (se $P(x,y) \wedge P(y,x) \Rightarrow x=y$) e funcionais inversas (se $P(x,y) \wedge P(z,x) \Rightarrow x=z$) e papéis.

- *OWL DL* garante completude, decidibilidade e toda a expressividade da lógica de descrições, almejando satisfazer engenheiros de conhecimento familiarizados com esta tecnologia. A expressividade torna-se ainda maior do que em *OWL Lite*: classes podem ser construídas por união, interseção e complemento, pela enumeração de instâncias e podem ter disjunções. Tipos são mantidos cuidadosamente separados (por exemplo, uma classe não pode ser instância e propriedade ao mesmo tempo).
- *OWL Full* fornece a expressividade de *OWL* e a liberdade de usar *RDF*, inclusive permitindo novas metaclasses, já que elas são subclasses definidas em *RDFS*. Fazendo este uso mais complexo, não há garantia de computabilidade. Aqui, não cabem as restrições de separação de tipos da versão anterior e é possível manipular e modificar metaclasses.

As linguagens menos expressivas (*OWL Lite* e *DL*) estão contidas dentro das mais expressivas (*OWL DL* e *Full*), de maneira que uma ontologia definida numa linguagem menos expressiva é aceita por uma linguagem mais expressiva; a recíproca, naturalmente, não é verdadeira.

7.5.1. Localizando os tipos básicos de OWL: os *Headers*

Para definir uma ontologia em *OWL*, é preciso em primeiro lugar, dizer onde estão, na Web, as classes primitivas das ontologias para que se possa definir novas classes como subclasse destas. Também é necessário determinar um namespace para a nova ontologia. Isto é codificado da seguinte forma num trecho da ontologia chamado de *Headers*, como no exemplo [Chen et al 2003]:

```
<rdf:RDF
  xmlns="file:/G:/myclasses#"
  xmlns:eyeglass="file:/G:/Glasses#"
  xmlns:owl="http://www.w3.org/2003/02/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#" >
```

As classes a serem definidas estarão localizadas no *namespace* da primeira definição. A segunda definição serve para que ontologias externas possam referenciar a ontologia sendo definida. As restantes localizam as definições primitivas de *OWL*, *RDF*, *RDFS* e *XMLSchema*.

7.5.2. Classes e atributos em OWL

Como foi dito, classes podem ser construídas de várias formas: por herança, união, interseção, complemento, pela enumeração de instâncias ou por restrições de propriedades. Por exemplo [Costello & Jacobs 2003], a classe “rio fluvial” é subclasse de

“rio”, e o atributo “desemboca” (*emptiesInto*) tem como imagem instâncias da classe “águas” (*BodyOfWater*):

```
<owl:Class rdf:ID="Flueve">
  <rdfs:subClassOf rdf:resource="#River"/>
</owl:Class>

<rdf:Property rdf:ID="emptiesInto">
  <rdfs:domain rdf:resource="#River"/>
  <rdfs:range rdf:resource="#BodyOfWater"/>
</rdf:Property>
```

Note-se as referências às superclasses primitivas `owl:Class` e `rdf:ID`, que conseguem ser localizadas graças aos Headers. Outro fato digno de nota, é que, em lógica de descrições, a definição dos atributos não têm de estar junto com a classe. Como dispomos da expressividade de lógica de descrições, podemos definir a classe “rio fluvial” como a subclasse de “rio” que desemboca (atributo *emptiesInto*) em mares, ou seja, a classe é definida com o auxílio de uma restrição:

```
<owl:Class rdf:ID="Flueve">
  <rdfs:subClassOf rdf:resource="#River"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#emptiesInto"/>
      <owl:allValuesFrom rdf:resource="#Sea"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

7.6. As camadas de lógica, prova e confiança

As camadas mais altas da proposta de Web semântica ainda não tomaram corpo. A camada lógica permite a especificação de regras que atuam sobre instâncias e recursos, enquanto a camada de prova as executa, e a de confiança avalia se a prova está correta ou não [Koivunen & Miller 2001]. Para que estas camadas entrem em operação, as camadas inferiores devem estar bem sedimentadas, o que ainda está acontecendo. Além do mais, sob o ponto de vista ontológico, não é interessante antecipar o uso de ontologias com regras, pois isto pode restringir a sua aplicabilidade. Porém regras podem ter utilidade para restringir atributos e exprimir axiomas.

A camada de lógica já dispõe de protótipos de linguagens, como DAML-L e RuleML [Boley et al 2001]. Vejamos um exemplo de uma regra em RuleML (*Rule Markup Language*, em português, linguagem de anotação de regras) que infere simetria numa cooperação entre duas pessoas:

```
<imp>
  <_head>
    <atom>
      <rel>cooperatesWith</rel>
      <var>person2</var>
      <var>person1</var>
    </atom>
  </_head>
  <_body>
    <atom>
      <rel>cooperatesWith</rel>
```

```
        <var>person1</var>
        <var>person2</var>
    </atom>
</body>
</imp>
```

7.7. ITTalks: uma aplicação da Web semântica

O ITTalks [Cost et al 2002] desenha um cenário de aplicações da Web semântica. O sistema consiste de um sítio da Web que dá acesso a uma base de dados de palestras passadas ou futuras, pessoas (professores, cientistas, convidados, etc) e lugares (salas, instituições). Cada instância destas classes foi anotada semanticamente através da linguagem DAML, de acordo com ontologias distintas – porém relacionadas - sobre tópicos de pesquisa, perfis pessoais, calendários e conversação, além de uma ontologia da ACM (*Association for Computer Machinery*, uma editora de pesquisa dos EUA). A anotação de tópicos de pesquisa tanto em páginas quanto nos perfis de usuários é executada por um classificador baseado em aprendizado automático, que consulta os resumos das palestras e as páginas pessoais e currículos dos usuários. Os usuários se cadastram especificando que tópicos são de seu interesse, e agentes do sistema notificam-nos quando alguma palestra correspondente a seu perfil de tópicos é incluída na base de dados. Como as páginas estão semanticamente anotadas, será fácil, no futuro, preparar os agentes do sistema para mediar o acesso à base de dados.

7.8. Avaliação e perspectivas da Web semântica

A Web semântica surge como tábua de salvação para a diversidade da Web. Contudo, organizar toda a Internet ontologicamente esbarra em problemas de várias naturezas:

- O usuário comum, que navega na rede e publica páginas, mesmo com interfaces gráficas teria dificuldade de formular consultas que envolvessem regras de lógica e ontologias, e também de lidar com as complexidades em especificar ontologias ou instanciá-las em suas próprias páginas, ainda mais em padrões complexos, como os baseados em lógica de descrições, e que se sobrepõem em várias camadas, como o trio OWL-RDF-XML;
- As próprias páginas possuem, às vezes, conteúdo ambíguo e vago, e isso pode fazer parte da própria informação, seja por seu conteúdo (como, por exemplo, em poesia), ou pela localidade do vocabulário empregado, mantendo o problema da linguagem natural;
- Dificilmente um padrão ontológico para a codificação de páginas, no tocante às ontologias a referenciar, será adotado pela “rede da liberdade” num espaço curto de tempo;

Provavelmente, no futuro, muitas das páginas da Internet, especialmente as voltadas para comércio eletrônico e as de comunidades específicas – como o domínio científico - se farão disponíveis através de formalismos lógicos, permitindo a agentes cognitivos tratarem as informações com maior precisão, sem tropeçar tanto em sintaxe e polissemia, como os engenhos de busca atuais. Sob o prisma da evolução da área de ontologias, vários fatos podem ser assinalados.

Por um lado, trata-se de uma área em pleno amadurecimento, com ferramentas cada vez mais sofisticadas sendo desenvolvidas, tentando atender a um mercado que percebe seus benefícios, que tendem a, gradativamente, melhorar a Web como um todo. Floresce uma nova geração de motores de inferência capazes de lidar com a abrangência das novas linguagens de anotação – o Euler para OWL, e o FaCT para OIL [Horrocks et al 2000] –, o que não invalida os outros motores, como Jess, Prolog, Algernon [Hewett 2003] e F-Logic, que podem ser aplicados usando os formatos da Web semântica,.

Por outro lado, talvez este seja apenas o primeiro passo da Web semântica, que se vê tentada a incorrer nos mesmos erros cometidos na Ontolingua: padrões complexos, excessivamente expressivos – por estarem ligados à lógica de descrições - sacrificam a legibilidade e clareza que as ontologias devem almejar, a ponto de tornar-se difícil definir páginas para a Web semântica sem o auxílio de ferramentas gráficas.

8. Tópicos Abertos de Pesquisa em Ontologias

O crescimento de interesse, e conseqüente pesquisa sobre ontologias é patente, por tratar-se de uma área relativamente nova e promissora em informática. Por isso, muitos tópicos têm requerido pesquisa ativa.

8.1. A Web semântica

Este é, com certeza, o tópico mais debatido de toda a área. Linguagens ligadas a ontologias têm sido discutidas, com suas limitações, complexidades, facilidades, expressividade e aplicabilidade em sistemas para a Web e comércio eletrônico, principalmente.

8.2. Concepção de ontologias

Apesar dos princípios de construção enumerados nesta seção, ainda não existem ferramentas de metodologia que guiem uma elaboração completamente padronizada, como em outros tipos de software. Em consequência disso, não há métodos sedimentados de validação, verificação, desenvolvimento [Gómez-Pérez 99], e mesmo documentação de ontologias, que muito facilitariam seu reuso e aplicação. Engenharia e metodologias têm sido estudadas e propostas, visando preencher este vácuo.

Em problemas relativos à concepção epistemológica de ontologias, ou seja, a forma e abordagem de construção de ontologias, uma questão parece ser crucial, e que soa como um paradoxo: os princípios de construção apresentam uma contradição, pois sugerem que uma ontologia deve ser projetada visando sua aplicação, e, ao mesmo tempo, ser extensível, representando as entidades dos domínios com forte engajamento ontológico. Na verdade, esse paradoxo representa a tensão entre sua aplicabilidade funcional (que deve restringi-las) e extensibilidade (que deve expandi-las, incluindo o máximo de conhecimento acerca do domínio). Outra faceta deste problema diz respeito à abordagem para projetar ontologias: devem seguir um prisma epistemológico, modelando um domínio de acordo com suas entidades (visão real) [Guarino 97], ou sob o prisma dos serviços que as entidades do domínio disponibilizam (visão funcional) [Visser & Cui 98]?

Ainda sobre concepção, a definição de relações parte-todo requer cuidados [Staab & Maedche 2000], pois existem vários tipos de partições que devem gerar

inferências distintas, como componente-objeto (ramo/árvore), membro-coleção (árvore/floresta), porção-massa (fatia/bolo), matéria-objeto (alumínio/avião), característica-atividade (pagar/comprar), lugar-área (cidade/estado), fase-processo (adolescente/crescimento) [Artale et al 95]. Ainda não existe uma solução padrão para modelagens parte-todo.

8.3. Aplicações de aprendizado para ontologias

Técnicas de aprendizado automático podem se tornar o mais forte aliado em tudo o que diz respeito a ontologias. Elas podem ser usadas na construção de ontologias a partir de texto, na anotação de páginas baseadas em ontologias, e na extração e classificação de informação.

A pesquisa em construção de ontologias foi inaugurada pelo sistema IMPS [Crow & Shadbolt 98], um sistema multiagente que processa textos pré-selecionados. Existem, no sistema, dois tipos de agentes: Agentes de Extração de Conhecimento e Agentes de Construção de Ontologias. O sistema emprega ainda dois conjuntos de arquivos de código, um para extração de conhecimento e outro que fornece informação suplementar sobre as tarefas a serem desempenhadas, além da base de dados e do dicionário semântico da língua inglesa WordNet. O sistema não empregava aprendizado, mas já existe um protótipo, o Text-to-Onto, que processa fontes de conhecimento, como dicionários, e, baseando-se em palavras-chave, tenta achar através de aprendizado de regras de associação, relações entre os conceitos, ajudando a formar uma ontologia.

O mesmo grupo desenvolveu uma ferramenta para agrupamento conceitual de textos baseada em aprendizado [Hotho et al 2001] - em que se descobrem que conceitos ajudam a distinguir conjuntos de textos -, além de outra para anotação semi-automática de páginas a partir de ontologias - que pode ser muito útil para a Web semântica.

Também é sabido que as técnicas de aprendizado de máquina prestam valioso serviço às tarefas de categorização de informação e extração de informação sobre textos, promovendo flexibilidade à representação em relação a variedades lingüísticas dos documentos, sendo, inclusive, empregada em sistemas de Processamento de linguagem Natural (PLN).

8.4. Ontologias para a comunidade de orientação a objetos: UML

Dado o interesse despertado pela noção de agentes, a comunidade de desenvolvimentno de sistemas orientados a objetos têm despertado para o uso de ontologias através da linguagem diagramática de modelagem unificada UML (*Unified Modeling Language*) [Eriksson & Penker 98]. A linguagem foi adotada como padrão em modelagem pela OMG (*Object Management Group*) [OMG 2003], e é largamente utilizada no desenvolvimento de sistemas orientados a objeto.

UML não pode ser considerada um formalismo de representação, devido à ausência de declaratividade, de um motor de inferência e de uma semântica formal - sua semântica é definida por um metamodelo - e também por ser empregada mais para modelagem estrutural e comportamental - que favorecem à implementação de objetos - do que conceitual, como as ontologias. Ela, entretanto, possui construtos abstratos o suficiente para permitirem a representação de ontologias, como classes e atributos. Restrições são definidas na linguagem auxiliar OCL (*Object Constraint Language*, ou

linguagem de restrição de objetos), que tem a expressividade da lógica de 1ª. ordem [Evans & Kent 99]. Relacionamentos são chamados em UML de *associações*, e herança de *generalização*. Existe apenas um tipo de relação meronímica (parte-todo), chamada de *agregação*, o que pode dificultar a modelagem de relações meronímicas complexas das ontologias (vide subseção 8.2.).

Existem propostas para aproximar o universo UML de ontologias. A definição de uma semântica formal, bem como de regras de transformação dedutivas para provar que um diagrama é conseqüência de outro foi proposta em [Evans 98], e pode redundar numa forma de inferência para UML. Outra proposta nesse sentido visa acoplar UML à Web semântica através de tradutores entre UML e a linguagem DAML [Lockheed 2000]. A tradução DAML-UML é mais complexa, porque as palavras-chaves de DAML podem ter várias traduções possíveis em UML, e qualquer delas perderá em clareza.

Também já foram criadas aplicações de UML para problemas atacados tipicamente com ontologias, o que corrobora a proximidade entre as áreas. Por exemplo, ela foi usada pela alfândega e receita federal holandesas na concepção de um sistema de gestão de conhecimento para processos legislativos, que apressa a implementação de mudanças em leis [Engers & Glassée 2001].

8.5. Outros tópicos

Um tópico recente de pesquisa gerou uma nova área: a *integração inteligente de informação*. Ela visa atacar a diversidade de visões de ontologias sobre um mesmo domínio, que o abordam sobre perspectivas distintas. Têm sido propostas soluções de mapeamento e integração semântica através de contextos comuns para a solução deste problema [Wache & Stuckenschmidt 2001].

Outro tópico de pesquisa relevante versa sobre a integração entre ontologias e bancos de dados: qual a melhor forma de armazenamento em bancos de dados de um grande volume de dados representados em *frames* de ontologias? Leve-se em conta que atributos de *frames*, por poderem assumir múltiplos valores - ao invés de um único como num campo de uma tabela relacional -, não podem ser diretamente colocados em bancos de dados relacionais de forma normalizada [Bertino et al 2001]. Diversos tipos de solução, acoplando as tecnologias de bancos de dados e dedução, cabem neste contexto: modelagens para armazenamento em bancos de dados relacionais, extensões da linguagem de manipulação de dados SQL, bancos de dados dedutivos, extensões de bancos de dados orientados a objetos e *data warehousing*.

9. Conclusões

Face às dificuldades de seleção e recuperação de informações em sistemas abertos, como a Internet, para a tomada de decisões, as ontologias constituem a técnica adequada e imprescindível para as necessidades de comunicação e para um melhor reuso automático de informação e conhecimento, para a atual conjuntura da computação.

As ontologias já começam a desempenhar o papel de conhecimento estruturado disponível para reuso em larga escala por sistemas e programas, um acontecimento sem paralelo na história da ciência da computação, e devem tornar-se o pilar de sustentação para aplicações como comércio eletrônico, *workflow* e gestão de conhecimento. A própria Internet pode ser redesenhada em função de seus potenciais benefícios,

tornando-se uma Web semântica, em que as informações podem ser compreendidas dentro de seu contexto.

As ontologias também têm servido, em certas áreas, para unificar o conhecimento e para formar consensos sobre certos conceitos, causando uma integração de grupos de pesquisas, e sendo utilizadas, inclusive, com propósitos educativos. O seu emprego como elemento estruturador de informações pode representar para a história da informática o que, para a história humana, representou a criação de enciclopédias e bibliotecas: o armazenamento de conhecimento, já que o conhecimento pode, agora, trafegar entre computadores e sistemas que podem lançar mão deles, manipulá-los e aplicá-los no cumprimento de suas funções.

Agradecimentos

Agradeço a Guilherme Bittencourt, Alexandre Vasconcelos e Geber Ramalho, pelas revisões de versões anteriores de partes deste tutorial.

Bibliografia

- [Ait-Kaci & Podelski 93] Ait-Kaci, H., Podelski, A.; 1993 *Towards a Meaning of LIFE*. Journal of Logic Programming, 16(3&4). USA.
- [Ambite & Knoblock 97] Ambite, J.; Knoblock, C.: 1997. *Agents for Information Gathering*. In Software Agents. Bradshaw, J. Ed., MIT Press, Pittsburgh, PA, USA
- [Artale et al 96] Artale, A., Franconi, E., Guarino, N., Pazzi, L. 1996. *Part-Whole Relations in Object-Centered Systems: An Overview*. <http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/Parts.ps>
- [Austin 62] Austin, J.; 1962. *How to do things with words*. Clarendon Press, Oxford, UK.
- [Baalen & Fikes 93] Van Baalen, J., Fikes, R.; 1993. *The Role of Reversible Grammars in Translating Between Representation Languages*. Disponível na Internet em: <http://ksl-web.stanford.edu/knowledge-sharing/papers/index.html#reversible-grammars>
- [Baeza-Yates & Ribeiro-Neto 99] Baeza-Yates, R., Ribeiro-Neto, B.; *Modern Information Retrieval*. Addison Wesley. New York 1999.
- [Benjamins et al 98] Benjamins, R.; Fensel, D.; Pérez, A. 1998. *Knowledge Management through Ontologies*. Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management, Basel, Suíça.
- [Bertino et al 2001] Bertino, E., Catania, B., Zarri, G. 2001. *Intelligent Database Systems*. Addison Wesley, Inglaterra.
- [Bittencourt 2001] Bittencourt, Guilherme; 2001. *Inteligência Artificial : Ferramentas e Teorias*. Editora da UFSC. Florianópolis, SC.
- [Boley et al 2001] Boley, H., Tabet, S., Wagner, G. *Design Rationale of RuleML: A Markup Language for Semantic Web Rules*. <http://www.mynisus.com/members/stabet/ruleml/ratoruleml2.0.xml>
- [Brachman & Schmolze 85] Brachman, R., Schmolze, J. 1985. *An Overview of the KL-ONE Knowledge Representation System*. Cognitive Science, 9(2):171-216, USA.

- [Chandrasekaram 99] Chandrasekaran, B. 1999. *What Are Ontologies, and Why Do We Need Them?*. IEEE Intelligent Systems, Janeiro 7 Fevereiro 99, EUA.
- [Chandrasekaram & Josephson 97] Chandrasekaram, B. and Josephson, J. R.; 1997. *The Ontology of Tasks and Methods*. In Proceedings of th Symposium on Ontological Engineering, 1997, Stanford, CA, USA.
- [Chaudri et al 98] Chaudri, V., Farquhar, A., Fikes, R., Karp, P., Rice, J. 1998. *OKBC: A Programmatic Foundation for Knowledge Base Interoperability*. Proceedings of AAAI-98, Madison, WI, EUA.
- [Chen et al 2003] Chen, C., Wang, X., Lu, Y., Zheng, Z. 2003. *Web Ontology Language-OWL*. www.cs.concordia.ca/~faculty/haarslev/teaching/semweb/OWL.ppt
- [Clark 99] Clark, D. 1999. *Mad cows, metathesauri, and meaning*. IEEE Intelligent Systems. Jan/Fev 99, USA.
- [Corcho & Gómez-Pérez 2000] Corcho, O., Gómez-Pérez, A. 2000. *Evaluating knowledge representation and reasoning capabilities of ontology specification languages*. Proceedings of ECAI'00 Workshop on Application of Ontologies and Problem Solving Methods.
- [Cost et al 2002] Cost, R., Finin, T., Joshi, A., Peng, Y., Nicholas, C., Soboroff, I., Chen, H., Kagal, L., Perich, F., Zou, Y., Tolia, S. 2002. *ITalks: a case study in the semantic Web and DAML+OIL*. IEEE Intelligent Systems. Jan/Feb. pp 40-47. EUA.
- [Costello & Jacobs 2003] Costello, R., Jacobs, . 2003. *OWL Web Ontology Language*. <http://www.xfront.com/owl/>
- [Crow and Shadbolt 99] Crow, L.; Shadbolt, N. 1999. *IMPS – Internet Agents for Knowledge Engineering*. <http://ksi.ucalgary.ca/KAW98S/Shadbolt>.
- [Cutovsky et al 93] Cutovsky, M., engelmores, R., Fikes, R., Genesereth, M., Gruber, T., Mark, W., Tenenbaum, J. 1993. *PACT: an experiment in integrating concurrent engineering systems*. <http://www.ksl.stanford.edu/knowledge-sharing/papers/#pact>
- [Domingues 98] Domingues, J. 1998. *Tadzebao and WebOnto: Discussing, Browsing and Editing Ontologies on the Web*. Proc. Of the 11th Workshop on Knowledge Acquisition, Modelling and Management.
- [Duineveld et al 99] Duineveld, A., Stoter, R., Weiden, M., Kenepa, B., Benjamins, V. 1999. *WonderTools: a comparative study of ontological engineering tools*. www.swi.psy.uva.nl/wondetools
- [Engers & Glassée 2001] Engers, T., Glassée, E. 2001. *Facilitating the legislation process using a shared sonceptual model*. IEEE Intelligent System, Jan/Fev, pp 50-58. EUA.
- [EON 2002] 2002. *Workshop of Evaluation of ONtology tools*. <http://km.aifb.uni-karlsruhe.de/eon2002>
- [Eriksson & Penker 98] Eriksson, H., Penker, M. 1998. *UML toolkit*. New York. Wiley Computer Publishing. EUA.
- [Eriksson 2000] JessTab plugin for Protégé. <http://www.ida.liu.se/~her/JessTab/>

- [Evans 98] Evans, A.. 1998. *Reasoning with UML diagrams*. Proceedings of the Workshop on Industrial Strength Formal Methods (WIFT'98). IEEE Press.
- [Evans & Kent 99]. Evans, A., Kent, S. 1999. *Meta-modelling semantics of UML*. www.cs.york.ac.uk/puml/papers/pumluml99.pdf
- [Farquhar 96] Farquhar, A., Fikes, R., Rice, J.; 1996. *The Ontolingua Server: a Tool for Collaborative Construction*, Computer Science Department, Stanford University.
- [Farquhar 97] Farquhar, A. 1997. *Ontolingua tutorial*.
<http://www.ksl.stanford.edu/software/ontolingua/tutorial.pdf>
- [Feigenbaum et al 71] Feigenbaum, E., Buchanan, B., Lederberg, J. 1971. *On generality and problem solving: a case study using the Dendral program*. Em Meltzer, B., Michie, M. (eds.). Machine Intelligence – vol. 6. Edinburgh Univ. Press, Escócia.
- [Fensel 2001] Fensel, D. 2001. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag. Berlim.
- [Fensel et al 2001] Fensel, D., Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P. 2001. *OIL: an ontology infrastructure for the semantic Web*. IEEE Intelligent Systems, Mar/Apr, pp. 38-45. EUA.
- [Fikes 98] Fikes, Richard; 1998. *Reusable Ontologies: A Key Enabler for Electronic Commerce*. <http://www.ksl.stanford.edu>.
- [Finin et al 94] Finin, T.; Fritzson, R.; McKay, D.; and McEntire, Robin; 1994. KQML as an agent communication language. Proceedings of the International Conference on Information and Knowledge Management. ACM Press, NY.
- [Freitas 2001] Freitas, F. 2001. *Ontology of Science*.
http://protege.stanford.edu/plugins/ontologyOfScience/ontology_of_science.htm
- [Freitas & Bittencourt 2002] Freitas, F.; Bittencourt, G. 2002. *Comunicação entre Agentes em Ambientes Distribuídos Abertos: o Modelo “peer-to-peer”*. Revista Eletrônica de Iniciação Científica (REIC). Ano II No. II Vol. II. Sociedade Brasileira de Computação (SBC). Brasil.
- [Freitas & Bittencourt 2003] Freitas, F.; Bittencourt, G. 2003. *An Ontology-based Architecture for Cooperative Information Agents*. A ser publicado nos Proceedings of the Internacional Joint Conference on Artificial Intelligence – IJCAI'2003. Acapulco, México.
- [Friedmann-Hill 2000] Friedmann-Hill, E. 2000. *Jess, The Java Expert System Shell*.
<http://herzberg.ca.sandia.gov/Jess>
- [Gennari et al 2003] Gennari, J., Musen, M., Ferguson, R., Grosso, W., Crubézy, M., Eriksson, H., Noy, N., Tu, S. 2003. *The evolution of Protégé': an environment for knowledge-based systems development*.
http://smi.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0943.html
- [Genesereth & Fikes 92] Genesereth, M. R., Fikes, R. E.; 1992. *Knowledge Interchange Format, Version 3.0 Reference Manual*, Logic-92-1, Computer Science Department, Stanford University, EUA.

- [Gómez-Pérez 94] Gómez-Pérez, A. 1994. *From Knowledge Based Systems to Knowledge Sharing Technology: Evaluation and Assessment*. <http://>
- [Gómez-Pérez 99] Gómez-Pérez, A. 1999. Tutorial on Ontological Engineering. Internacional Joint Conference on Artificial Intelligence – IJCAI'1999. Estocolmo, Suécia. <http://www.ontology.org/main/papers/madrid-tutorials.html>
- [Gruber 92] Gruber, Thomas .R.; 1992. *Ontolingua: A Mechanism to Support Portable Ontologies*. Technical Report KSL-91-66. Stanford University, Knowledge Systems Laboratory, EUA.
- [Gruber 95] Gruber, Thomas R.; 1995. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human and Computer Studies*, 43(5/6): 907-928.
- [Guarino 97] Guarino, N. 1997. *Some Organizing Principles for a Unified Top-Level Ontology*. Proceedings of the American Association of Artificial Intelligence Conference (AAAI-97) in the Spring Symposium on Ontological Engineering, EUA.
- [Hayes-Roth et al 78] Hayes-Roth, F., Waterman, D. A. and Lenat, D. B.; 1978. Principles of Pattern-Directed Inference Systems. In *Pattern-Directed Inference Systems*, Academic Press, New York, NY, USA.
- [Hewett 2003] Hewett, M. 2003. *Algernon in Java*. <http://smi.stanford.edu/people/hewett/research/ai/algernon/>
- [Horrocks et al 2000] Horrocks, I., Fensel, D., Broekstra, J., Decker, S., Erdmann, M., Goble, C., Harmelen, F., Klein, M., Staab, S., Studer, R., Motta, E. 2000. *The Ontology Inference Layer OIL*. <http://www.ontoknowledge.org/oil/TR/oil.long.html>
- [Hotho et al 2001] Hotho, A., Staab, S., Maedche, A. 2001. *Ontology-based Text Clustering*. <http://www.cs.cmu.edu/~mccallum/textbeyond/papers/hotho.pdf>
- [Huhns & Singh 97] Huhn, Michael N. and Singh, Munindar P.; 1997. *Agents and Multiagent Systems: Themes, Approaches, and Challenges*. In Huhns, Michael, Singh, Munindar, Eds. 1997. Readings in Agents. Morgan Kaufman Publishers, USA.
- [Huhns & Singh 97a] Huhn, Michael N. and Singh, Munindar P.; 1997. *Ontologies for Agents*. IEEE Internet Computing 1(6):81-83. Na coluna “Agents on the Web”. EUA.
- [KAON 2003] *KAON - the Karlsruhe ONtology and semantic web tool suite*. 2003. <http://kaon.semantic-web.org>
- [Karp et al 99] Karp, P., Chaudri, V., Thomere, J. 1999. *XOL: an XML-based ontology exchange language*. Version 0.4. www.ai.sri.com/~pkarp/xol
- [Klein 2001] Klein, M. 2001. *XML, RDF and relatives*. IEEE Intelligent Systems Mar/Apr, pp 26-28. EUA.
- [Kifer et al 95] Kifer, M., Lausen, G., Wu, J. 1995. *Logical Foundations of Object-Oriented and Frame-Based Languages*. Journal of the ACM. 42:741-843. EUA.
- [Koivunen & Miller 2001] Koivunen, M., Miller, E. 2001. *W3C Semantic Web Activity*. <http://www.w3.org/2001/12/semweb-fin/w3csw>

- [Lenat & Guha 90] Lenat, D. B. and R. V. Guha. 1990. *Building Large Knowledge Based Systems*. Reading, Massachusetts: Addison Wesley,
- [Lockheed 2000] Lockheed. 2000. *UBOT details*.
http://ubot.lockheedmartin.com/ubot/details/uml_to_daml.html
- [Luke et al 96] Luke, Sean; Spector, Lee; Rager, David. 1996. *Ontology-Based Knowledge Discovery on the World-Wide Web*. Proceedings of the Workshop on Internet-based Information Systems, AAAI-96 (Portland, Oregon). USA.
- [Maes 97] Maes, P.1997. *Pattie Maes on Software Agents: Humanizing the Global Computer*. Entrevista na IEEE Internet Computing, julho-agosto de 1997.
- [McCarthy 58] McCarthy, John; 1958. *Programs with Common Sense*. In Proceedings of the Symposium on Mechanisation of Thought Processes , 1:77-84, London. Her Majesty's Stationery Office, UK.
- [McGregor 90] McGregor, R.; 1990. *LOOM User's Manual*, ISI/WP-22.USC Information Sciences Institute. University of Southern California, EUA.
- [McGuinness et al 2000] McGuinness, D., Fikes, R., Rice, J., Wilder, J.. *The Chimaera Ontology Environment*. Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000). Austin, Texas. July 30 - August 3, 2000.
- [McGuinness et al 2002] McGuinness, D., Fikes, R., Hendler, J., Stein, L. 2002. *DAML+OIL: an ontology language for the semantic Web*. IEEE Intelligent Systems, Sep/Oct, pp. 72-80. EUA.
- [Miller 95] Miller, G. 1995. *WordNet: A Lexical Database for English*. Communications of the ACM. 38(11):39-41. EUA.
- [Minsky 75] Minsky, M.; 1975. A Framework for Representing Knowledge. In *The Psychology of Computer Vision*, p.211-281, McGraw-Hill, New York. USA.
- [Minsky 86] Minsky, M.; 1986. *Society of Minds*. Simon & Schuster, Inc. EUA.
- [Maedche & Staab 2001] Maedche, A., Staab, S. 2001. *Ontology learning for the semantic Web*. IEEE Intelligent Systems Mar/Apr pp. 72-79. EUA.
- [Motta 99] Motta, e. 1999. *Components of knowledge modelling*. IOS Press. Amsterdam, Holanda.
- [Neches 91] Neches, R., Fikes, R., Finin, T., Gruber, T.R., Patil, R., Senator, T., Swartout, W.; 1991. *Enabling Technology for Knowledge Sharing*. In AI Magazine, 12(3): 36-56.
- [Newell 82] Newell, A.; 1982. The Knowledge Level. *Artificial Intelligence* 18(1):87-127.
- [Noy 97] Noy, N. 1997. *Knowledge Representation for Intelligent Information Retrieval in Experimental Sciences*. Tese de Doutorado, Faculdade de Ciência da Computação, Universidade Nordestina de Boston, Massachusetts, EUA.
- [Noy et al 2000] Noy, N., Ferguson, R., Musen., M. 2000. *The knowledge model of Protege-2000: Combining interoperability and flexibility*. 2th International

- Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France .
- [Oates et al 94] Oates, T.; Prasad, M.; Lesser, V. 1994. *Cooperative Information Gathering: A Distributed Problem Solving Approach*. Computer Science Technical Report 94-66- version 2. University of Massachusetts, Amherst, USA.
- [OilEd 2003] 2003. OilEd. <http://img.cs.man.ac.uk/oil/>
- [Ontoprise 2003] Ontoprise produkts. 2003. www.ontoprise.de/produkts
- [ODBC 98] 1998. Open DataBase Connectivity., <http://www.obdc.com/index.html>
- [OMG 2003] 2003. Object Management Group. www.omg.org
- [PC Week 2000] PC Week. 2000. *DAML could take search to a new level*. February 7,2000. USA.
- [Riley 99] Riley, G. 1999. *CLIPS: A Tool for Building Expert Systems*. <http://www.ghg.net/clips/CLIPS.html>
- [Robinson 65] Robinson, J. 1965. *A machine-oriented logic based on the resolution principle*. Journal of the ACM, 12(1):23-41, EUA.
- [Rumelhart et al 86] Rumelhart, D. McClelland, J. and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press. Cambridge, MA, EUA.
- [Schank 91] Schank, Roger C.; 1991. *Where's the AI?*. In AI Magazine.12:38-49. USA.
- [Schreiber et al 94] Schreiber, A., Wielinga, B., de Hoog, R., Akkermans, H., van de Velde, W. 1994. *CommonKADS: A Comprehensive Methodology for KBS Development*. IEEE Expert, Dezembro,28-37, EUA.
- [Smith et al 2003] Smith, M., Welty, C., McGuinness, D. 2003. *Web Ontology Language (OWL) Guide Version 1.0*. <http://www.w3.org/TR/2003/WD-owl-guide-20030210/>
- [Sowa 1999]. Sowa, J. 1999. *Knowledge representation: logical, philosophical and computational foundations*. Brooks Cole Pub. Co. Pacific Grove, EUA.
- [Staab & Maedche 2000] Staab,S., Maedche, A. 2000. *Axioms are Objects, too – Ontology Engineering beyond the Modeling of Concepts and Relations*. Proceedings of ECAI 2000, 14th European Conference on Artificial Intelligence , Workshop on Applications of Ontologies and Problem Solving Methods.
- [Studer et al 98] Studer, R., Benjamins, R., Fensel, D. 1998. *Knowledge Engineering: Principles and Methods*. Data and Knowledge Engineering. 25(1998):161-197.
- [Uschold & Gruninger 99] Uschold, M., Gruninger, M. 1999. *A Framework for Understanding and Classifying Ontology Applications*. [http:// sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/11-uschold.pdf](http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/11-uschold.pdf)
- [van der Vilt 2000] van der Vilt, E. 2000. *XML Linking Technologies*. <http://www.xml.com/pub/a/2000/10/04/linking>

- [Valente & Breuker 96] Valente, A., Breuker, J. 1996. *Towards Principled Core Ontologies*. In B.R. Gaines and M. Mussen, editors, Proceedings of the KAW-96, Banff, Canada.
- [Valente et al 99] Valente, A., Russ, T., MacGregor, R., Swartout, W. 1999. *Building and (Re)Using an Ontology of Air Campaign Planning*, IEEE Expert, January/February. EUA.
- [Visser & Cui 98] Visser, P.; Cui, Z. 199. *On Accepting Heterogeneous Ontologies in Distributed Architectures*. Proceedings of the ECAI'98 workshop on Applications of Ontologies and Problem-solving methods, Brighton, UK.
- [Wache & Stuckenschmidt 2001] Wache, H. Stuckenschmidt, H. 2001. *Practical Context Transformation for Information System Interoperability*. In Proceedings of the 3rd International Conference on Modeling and Using Context (CONTEXT'01), Lecture Notes in AI, Springer Verlag. Berlin. Alemanha.
- [Warren 83] Warren, D.; 1983. *The runtime environment for a prolog compiler using a copy algorithm*. Technical Report 83/052, Suny and Stone Brook, New York. USA.
- [WebODE 2003] WebODE. 2003.
<http://webode.dia.fi.upm.es/webode/login.html>
- [Wielinga et al 92] Wielinga, B. J.; Schreiber, A T. and Breuker, J. A.; 1992. KADS A modelling approach to knowledge engineering. *Knowledge Acquisition*. 4(1):5-53.
- [Winograd 75] Winograd, T.; 1975. Frame Representations and the declarative/procedimental controversy. In *Representation and Understanding Studies in Cognitive Science*, 185-210, Academic Press, New York, NY. USA.
- [Winston & Horn 81] Winston, Patrick H. and Horn, Berthold K.; 1981. *LISP*. Addison-Wesley Publishing Company.
- [Woods 75] Woods, W. A.; 1975. What's in a link: Foundations for semantic networks. In *Representation and Understanding Studies in Cognitive Science*, Academic Press, New York, NY. USA.