UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

VISUALIZAÇÃO DE ONTOLOGIAS

VISUALIZAÇÃO COMPUTACIONAL

JOÃO PAULO ORLANDO
KLEBERSON JUNIO DO AMARAL SERIQUE

SÃO CARLOS – SP

2010

1 Introdução

A Web Semântica é composta por enormes coleções de conhecimento, como as ontologias. Elas são representadas através de árvores cujos nós são dobráveis e é constituída por classes e instâncias. A convenção de ontologias, em geral representadas na forma de gráficos, para árvores é construída por herança múltipla. A visualização da informação dos dados dessas ontologias torna-se uma pesquisa interessante, devido ao fato de que a visualização de uma ontologia pode conter coleções de conhecimentos com milhões de conjuntos de dados.

As ontologias são muito interessantes em varias áreas, a prova disso é que muitas estão sendo desenvolvidas para fornecer uma variedade de serviços de conhecimento. Contudo, há uma necessidade crescente de ferramentas para visualizar graficamente e interativamente tais estruturas, melhorando seu entendimento por parte dos usuários e desenvolvedores de ontologias. Este trabalho apresenta as principais técnicas de visualização da informação utilizadas para criação e investigação de conhecimento em ontologias, levantando as principais características dos algorítimos utilizados e das vantagens e desvantagem especificas de cada. Citamos também a ferramenta Protégé e as suas principais extensões visuais de auxilio a criação de ontologias. Essas visualizações ajudam na compreensão das estruturas das ontologias, na analise de suas conexões de rede, na observação de mudanças, etc.

O trabalho visa mostrar o estado da arte das ferramentas existentes e das técnicas de visualização da informação. O trabalho foi divido em uma revisão literária sobre ontologias (apresentada no capitulo 2), das principais técnicas de visualização da informação aplicadas em ontologias (apresentada no capitulo 3) e a ferramenta Protégé com suas extensões de visualização de ontologias (apresentada no capitulo 4).

2 Ontologias

O termo ontologia é emprestado da filosofia (Gruber, 1993) e empregado na ciência da informação e em inteligência artificial, de modo geral, como uma forma de representação do conhecimento (Smith e Welty, 2001). São diversas as definições, tipos, propostas de aplicações e metodologias de construção encontradas na literatura para o termo (Almeida e Bax, 2003).

Gruber (1993) define ontologia como uma especificação explícita da conceituação de um domínio, visando uma classificação padronizada e formal por meio de um vocabulário controlado que representa semanticamente e exatamente os termos de um domínio específico e seus relacionamentos.

Uma minuciosa análise das possíveis interpretações e definições do termo “ontologia” é realizada por Guarino e Ginaretta (1995) que a definem sob a perspectiva de uma explicação parcial de uma conceituação ou como uma estrutura semântica intencional que encapsula regras para representar uma parte da realidade.

Simplificando as definições de Gruber (1993); e Guarino e Ginarreta (1995), Uschold e Grüninger (1996) relatam que o termo ontologia é usado para referenciar o entendimento compartilhado de algum domínio de interesse, que pode ser usado como um *framework* para resolver problemas de interoperabilidade, de reuso e de especificação de dados. Em um trabalho posterior, Uschold (1998) acrescenta que o termo ontologia pode ser empregado como um meio para estruturar uma base de conhecimento, como parte de uma base de conhecimento ou como um mecanismo de interlíngua, integrando repositórios de dados.

Discussões e estudos aprofundados sobre a definição e conceituação do termo ontologia podem ser encontrados em Chandrasekaran, Josephson e Benjamins (1999); Smith e Welty (2001); Almeida e Bax (2003) e Corazzon (2010). Mesmo sem um consenso sobre sua definição, ontologias compartilham características comuns e impulsionam o desenvolvimento de diversos trabalhos referentes a metodologias, ferramentas, linguagens e aplicações.

2.1 Características, linguagens e tipos de ontologias

Uma ontologia é especificada por meio de componentes básicos que são as classes, relações, axiomas e instâncias, além de ser expressa por meio de uma linguagem de construção (Almeida e Bax, 2003).

As classes, o foco da maioria das ontologias, são utilizadas para descrever os conceitos de um domínio, possibilitando a organização das classes em um sistema lógico e hierárquico contendo subclasses que representam conceitos mais específicos (Noy e McGuinness, 2001). As relações representam o tipo de interação entre os conceitos de um domínio e as propriedades presentes nas classes e indivíduos. Os termos *slot*, *role*, propriedade e até mesmo atributo podem ser empregados como sinônimo para as relações (Noy e McGuinness, 2001; Horridge et al., 2004).

As relações podem ter características próprias como ser transitiva, simétrica, e ter uma cardinalidade definida, entretanto essas características não são abordadas por todas as linguagens. Os axiomas são utilizados para modelar regras assumidas como verdadeiras no domínio em questão, de modo que seja possível associar o relacionamento entre os indivíduos, além de fornecer características descritivas e lógicas para os conceitos.

Para Uschold e Grüninger (1996) os axiomas são especificados para definir a semântica e significado dos termos (classes e propriedades) e sugere que a fase de definição dos axiomas (especificação da ontologia) é a mais difícil na construção de ontologias. Por fim, os indivíduos, ou instâncias das classes, são utilizados para representar elementos específicos, ou seja, os próprios dados, que juntamente com a definição de uma ontologia constituem a base de conhecimento (Noy e McGuinness, 2001).

Os indivíduos representam objetos do domínio de interesse (Horridge et al., 2004). Os componentes básicos de uma ontologia são definidos por meio de uma linguagem de representação. Horridge et al. (2004) apontam que diferentes linguagens para ontologias proporcionam diferentes facilidades.

3 Tipos de visualizações de Ontologias

Ontologias são extremamente grandes e complexas, tornando assim difícil a navegação e avaliação das mesmas. A visualização pode ajudar a usuários a analisar ontologia através de um panorama geral e identificar conceitos principais. Os principais tipos de apresentação de dados de ontologias são nas formas de Grafo 3D, Árvore, Árvore Radial, TreeMaps e Grid.

3.1 Grafo 3D

Neste tipo de representação é visualizado um conceito por vez, no mais alto nível de detalhação. Os conceitos filho e pai mostram portanto sua raiz central e suas relações semânticas, inclusive as heranças. As relações semânticas são representadas por fechas que terminam com um pequeno cone, caso seja uma representação de saída, o cone será vermelho, caso seja de entrada o cone será verde. E relações de *DataType* devem terminar com um cone azul.

Relações diretas são desenhadas próxima do conceito, com uma cor mais opaca, enquanto que as relações herdadas estão localizadas mais longe do centro, representados por uma cor mais transparente.

Este tipo de representação pode ser muito útil durante as verificações de consistência, pois facilita encontrar regiões de conceitos ou relações inconsistentes, por exemplo, sempre que um conceito herdado de um ancestral que contrasta logicamente com outras próprias características. Todas essas características podem serem vistas na Figura 1:

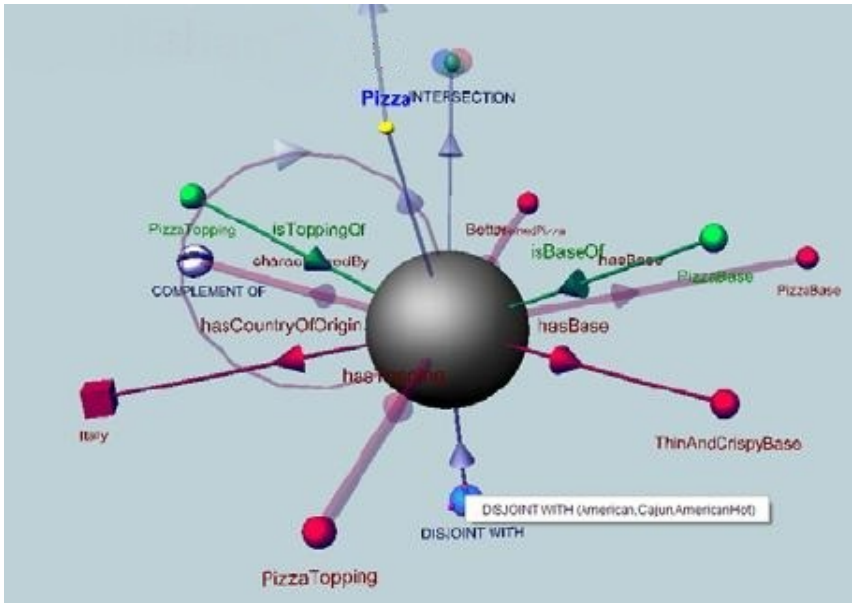

Figura 1: Visualização de um conceito em 3D

A visualização da Figura 2 mostra uma subárvore a partir de um termo da ontologia, ela exibe a hierarquia de seus termos, bem como outras ligações semânticas entre as classes representadas.

Como as evidências provam que usar muitos elementos na tela ao mesmo tempo, dificultam a atenção do usuário, esta visualização apresenta apenas os três níveis expandidos por vez. Enquanto o usuário navega na árvore, o sistema realiza automaticamente as operações de expansão e colapso, a fim de manter uma razoável complexidade da visualização.

Elementos recolhidos são brancos e seu tamanho é proporcional ao número de elementos presentes em sua sub-árvore. Conceitos localizada no mesmo nível de profundidade dentro da árvore tem a mesma cor, a fim de identificar facilmente grupos de irmãos.

A relação *is_a* é exibida com uma cor neutra (cinza) e sem rótulo, enquanto outras relações semânticas envolvendo dois conceitos são exibidos em vermelho, acompanhada do nome da relação. Se um elemento da árvore é relacionado a um nó que não se encontram na visualização, uma pequena esfera é adicionado para que o nó na proximidade do elemento de dado, de modo que encerra no final da

seta. Essa características podem ser vistas na Figura 2:

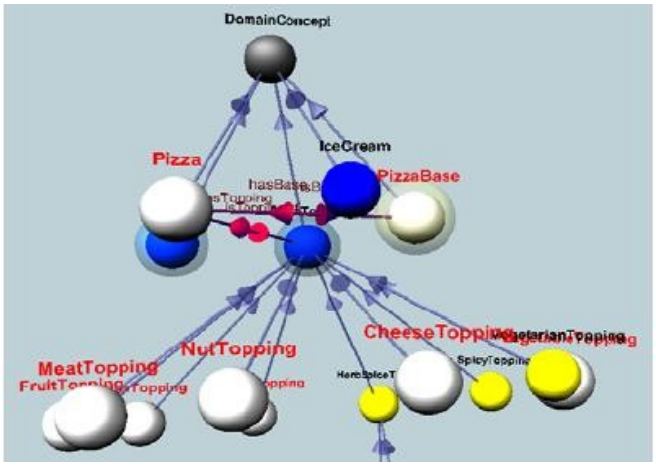

Figura 2: Visualização de 3 níveis de termos em 3D

3.2 Árvore

Historicamente, as árvores estão entre as primeiras classes estudadas em Desenho de Grafos, em função da sua popularidade e da simplicidade de sua estrutura (Nascimento e Ferreira 2005). As árvores podem ser classificadas de acordo com a natureza de dados que representam. Árvores podem expressar dados hierárquicos (como sistemas de arquivos, web sites, classificação categórica, similaridade, etc.), dados de processos ramificados (como genealogia e linhagens) e de dados de processos de decisão (como índices ou árvores de pesquisa, árvores de decisão e entre outros).

Técnicas para desenhar árvores podem ser dividas em duas: Árvores com Raiz e Árvores Livres. Árvores com Raiz representam dados hierárquicos com a presença de um nó raiz, já as Árvores Livres não possuem árvores enraizadas. No caso das ontologias prevalece a técnica de Árvores com Raiz pois as ontologias representam hierarquias de classes, atributos e indivíduos interligados por seus relacionamentos. Uma ontologia pode ser definida como um Árvore de dados representando um conjunto de conceitos em um determinado domínio e os relacionamentos entre eles. Na figura 3 é mostrado a ferramenta Jambalaya mostrando uma visualização baseada em árvores.

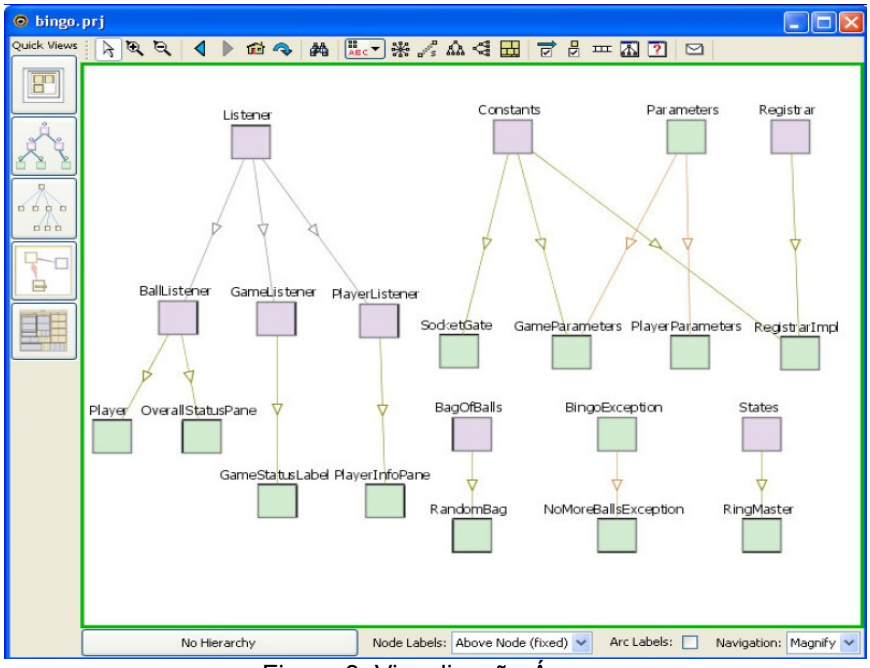

Figura 3: Visualização Árvore

3.3 Árvore Radial

O layout de Árvore Radial é um algoritmo que desenha um gráfico (Eklund et al., 2002). Este algorítimo começa com um nó raiz (posicionado no centro do gráfico) e chama recursivamente cada nó filho. Cada nó filho é posicionado fatias de espaço iguais e afastado do pai. O algoritmo cria uma fatia de espaço igual para cada nó filho. Este espaço é utilizado para alocar os nós filhos. O vértice raiz inicia com uma fatia de espaço de 360 graus para desenhar os nós filhos. Se um vértice raiz possui n nós filhos, cada filho recebe uma fatia igual à 360/n graus, na qual irá desenhar seus filhos. Os nós filhos são colocados a uma distância fixa radial de seus pais. Este algorítimo chama recursivamente cada nó filho e assim sucessivamente até atingir as folhas.

A desvantagem desse algorítimo é que se um determinado vértice possuir vários filhos, todos os nós filhos deles irão dividir o mesmo espaço, e para um outro determinado vértice irmão deste vértice que possui menos filhos irá poder alocar em uma fatia igual, porém com muito mais espaço para mostrá-los. Em outras

abordagens de layout deste algorítimo utiliza fatias de espaços flexíveis, nas quais os nós com grande número de filhos e ancestrais são alocados em um percentual maior de espaço em relação aos nós com menos ancestrais, que irão pegar um menor espaço em sua fatia.

Os algoritmos de layout de Árvore Radial com percentual ponderado comparam cada número de folhas de cada subárvore, resultantes dos nós filhos, e da mesma forma do layout radial é utilizado, porém em vez de atribuir o mesmo tamanho de fatias iguais para cada nó filho, o algoritmo substitui por um espaço ponderado de acordo com um cálculo de folhas resultantes das subárvores analisadas. A vantagem dessa abordagem de algoritmo é que ele pode aproveitar melhor os espaços gerados para cada nó filho, permitindo assim que mais nós sejam melhor visualizados no plano de coordenadas do gráfico.

A vantagem de utilizar um tamanho fixo é que ele produz disposição simétrica de árvores bem equilibradas, ou seja, árvores com aproximadamente o mesmo número de ancestrais para cada irmão, veja Figura 4 abaixo.

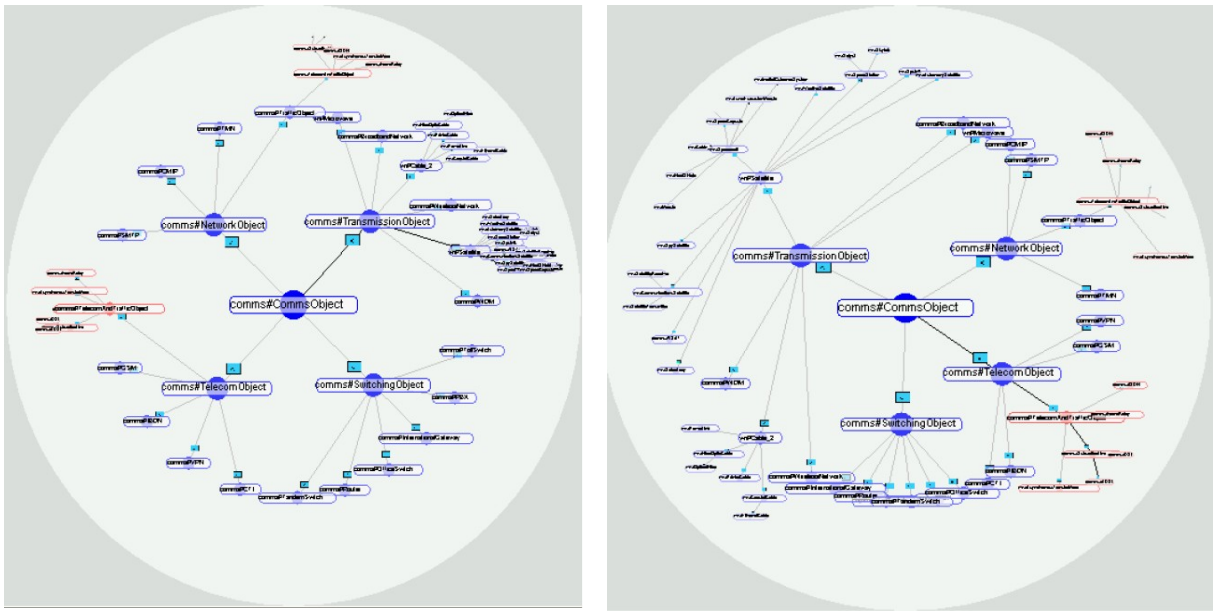

Figura 4: Uma visão hiperbólica utilizando o esquema de base radial (esquerdo) e layout radial ponderada (direito) – Adaptado de Eklund et al. (2002).

3.4 Treemaps

O *Treemaps* é uma técnica de visualização que permite exibir grandes conjuntos de informações hierárquicas através de métodos de preenchimento de espaço (Shneiderman, 1992). Isto ocorre pela divisão da área de exibição em uma sequência aninhada de retângulos correspondentes a atributos de um conjunto de dados.

O tamanho e a cor do retângulo podem ser utilizados para complementar a informação apresentada. As propriedades do *Treemaps* são enumeradas a seguir:

1. Existe somente sobreposição entre retângulos antecessores e predecessores.
2. A área do retângulo é proporcional ao seu tamanho.
3. O tamanho do retângulo é maior ou igual à soma do tamanho dos seus filhos.

A visualização provida pelo *Treemaps* permite uma mineração visual da informação, tornando mais fácil à descoberta de conhecimento a respeito dessa informação. As principais vantagens dessa técnica são mostradas a seguir:

- Utilização eficiente da tela de visualização.
- Preservação do contexto geral da informação.
- Navegação rápida entre nós.
- A estrutura do espaço de informação fica implícita no mapa exibido.
- Muito útil na exibição de variáveis quantitativas dos dados.

Percebe-se que nessa técnica, os galhos e a raiz não aparecem. É por esse motivo que as variações dessa técnica adicionam rótulos junto ao contorno dos retângulos para identificar os nodos intermediários (Brown e Gittens, 1997).

Essa técnica é útil quando as folhas e os valores dos dados são mais importantes do que a sua topologia na árvore. Porém, quando for importante visualizar a estrutura da hierarquia, a utilização dessa técnica não é aconselhável.

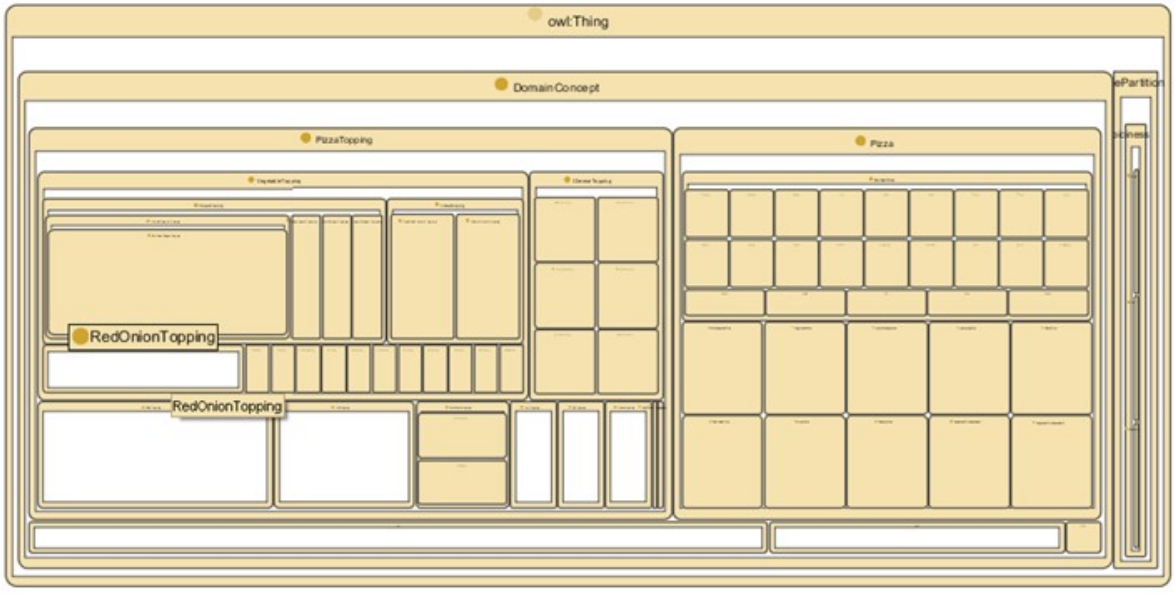

Figura 5: Visualização *Treemaps* - Completo

Como pode ser visto na figura 5 é uma visualização *Treemaps*, nela é possível identificar todos as características supracitadas desse tipo de visualização, nesse exemplo está sendo visualizado uma ontologia de pizza. Tudo pertence a uma única classe chamada *owl:Thing*, dentro dessa classe existem outras duas classes *DomainConcept* e *ValuePartition*, elas tem um tamanho proporcional a quantidade de itens que possuem, nesse caso a segunda possui poucas classe.

Como uma ontologia pode ser muito grande, é possível diminuir o número de níveis visualizados, na Figura 6 é apenas visualizados 3 níveis, mas mesmo assim é mantido a proporcionalidade dos itens.

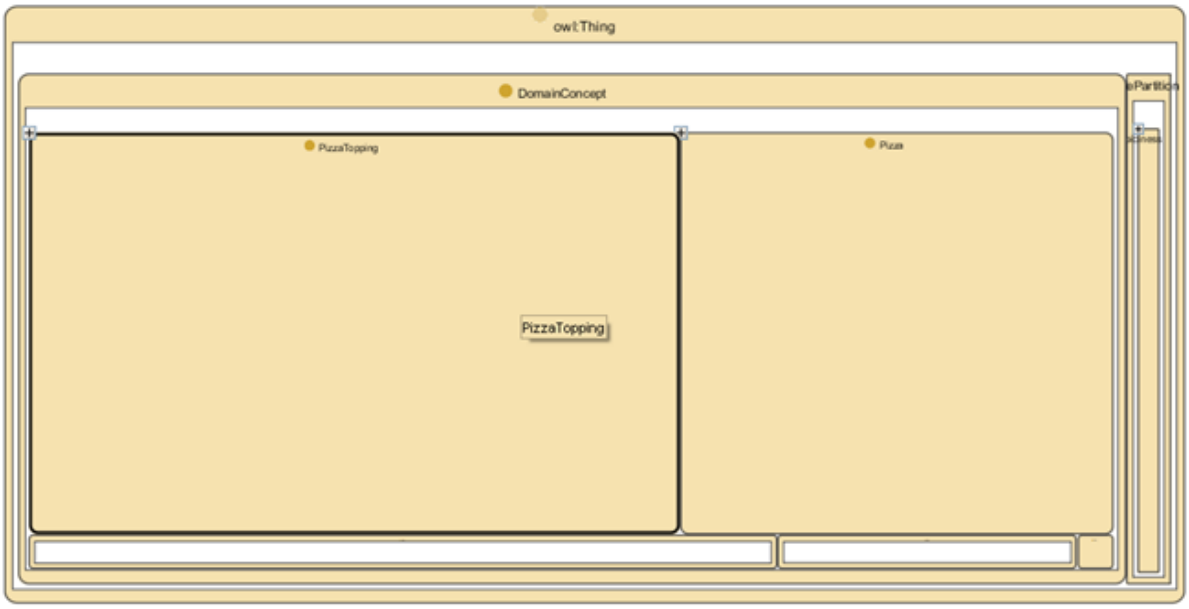

Figura 6: Visualização *Treemaps* - Parcial

Outra forma de navegação é escolher uma parte da ontologia, e visualizar só essa parte, assim é possível de navegar em toda ontologia, aproximando em parte que seja necessário, um exemplo pode ser visto na Figura 7:

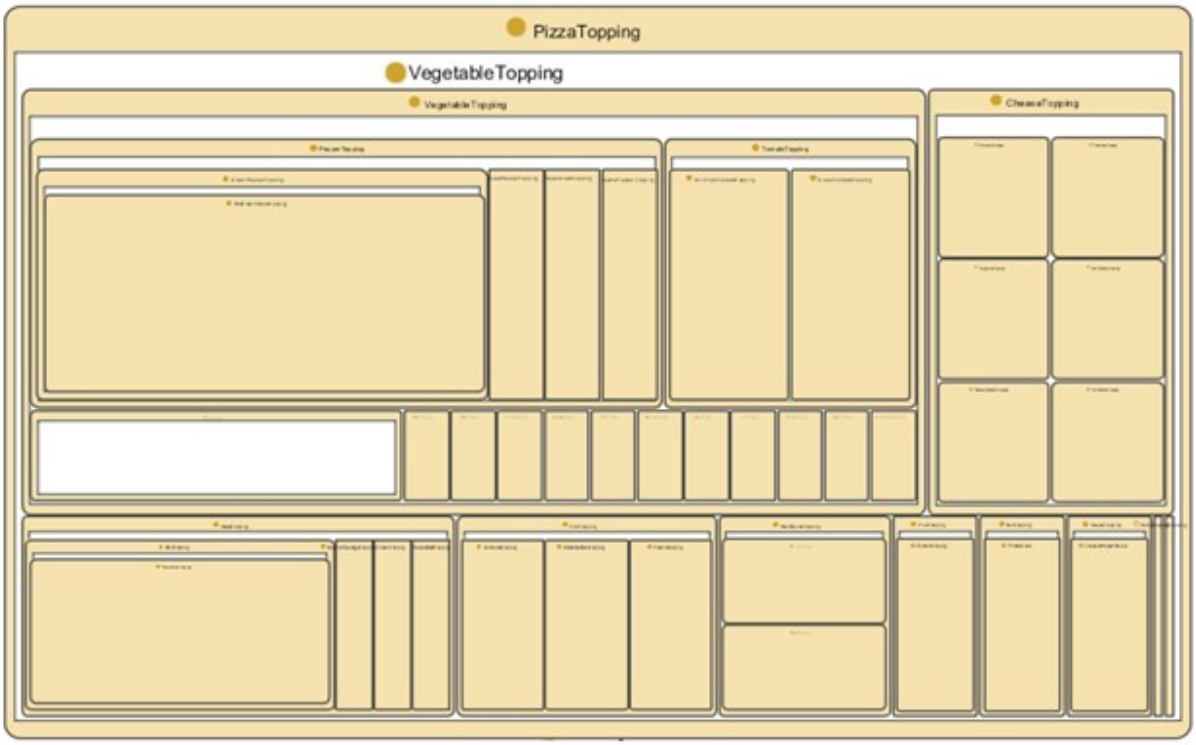

Figura 7: Visualização *Treemaps* - Parte

3.5 *Grid*

Grid é uma visualização semelhante ao *Treemaps*, as classes e as instâncias são representadas por pequenos retângulos. Em comparação ao *Treemaps*, esse tipo de visualização contêm todas as ligações entre as classes. Ligações de diferentes tipos podem ser distinguidos com tonalidades de cores diferentes. Arestas (arcos) são usados para mostrar relações entre conceitos e casos, como por exemplo, a relação *is-a* entre classes na hierarquia de conceitos.

Além de usar os arcos para mostrar as relações como as ligações entre as classes e instâncias, o usuário pode mostrar um tipo de relação de contenção. Por exemplo, ao invés de desenhar arcos entre os nós para mostrar a relação *is-a*, podemos "aninhar" subclasses dentro de suas superclasses. Além disso, isto poderia também ser vantajoso ao utilizar outros tipos de slot definidos pelo usuário para nodos aninhados. Por exemplo, em uma ontologia de anatomia, o usuário definiu o relacionado *part-of* pode ser uma relação mais adequada para os nós do que a relação *is-a*. Um exemplo de visualização *Grid* pode ser visto na Figura 8.

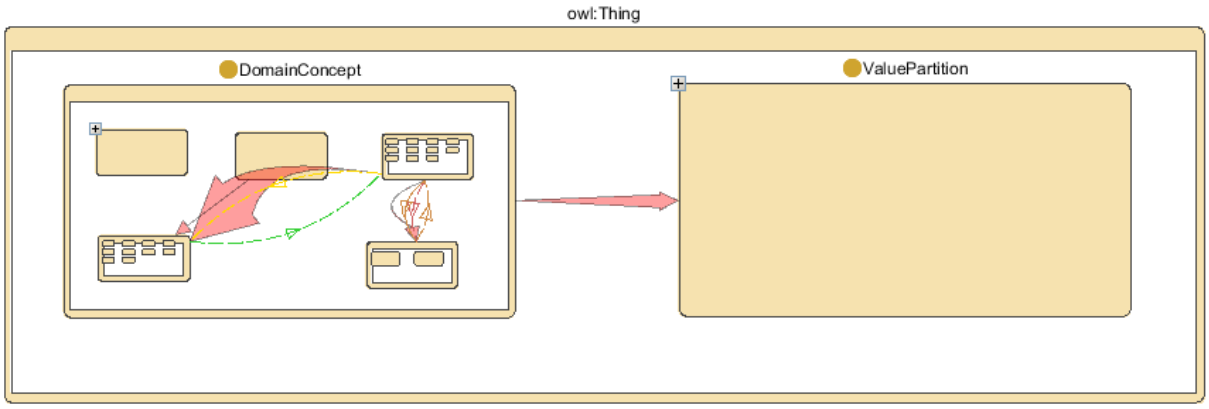

Figura 8: Visualização *Grid*

Ainda é possível trabalhar com menos níveis recolhendo ou expandindo classes. Na Figura 9 é possível ver a ligações entre as classes. A ligação em cor verde que esta selecionada, é uma ligação da classe *PizzaTopping* para *Pizza*. O nome da ligação é *isToppingOf*. Com ela possível dar uma logica as ligações.

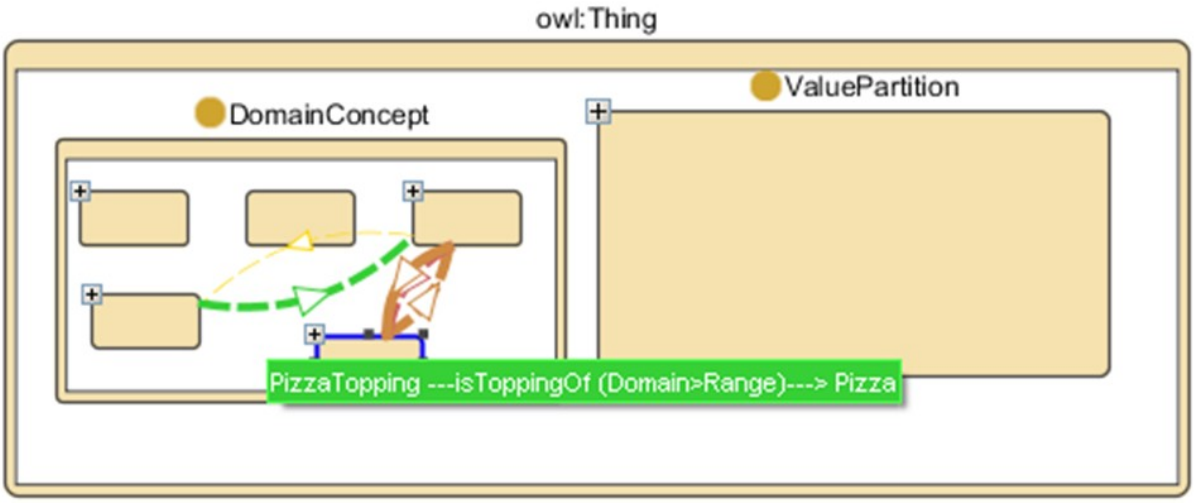

Figura 9: Visualização Grid - Ligação

4 Visualizações no Protégé

Protégé é o mais popular editor de ontologias e é desenvolvido pela Universidade de Stanford. As ontologias podem ser editadas e vistas usando essa ferramenta e também fornece uma Java API para acessar as ontologias. Ele possui uma arquitetura plug-in e permite a extensão da ferramenta com novas funcionalidades e capacidades.

Atualmente o Protégé possui duas versões em operação, a 3 e a 4. Alguns plug-in rodam apenas em uma das versões. Dentre os plug-in de visualização de ontologias mais conhecidos do Protégé podemos citar o Jambalaya, Ontoviz Tab, OWL Viz Tab e o TGViz Tab.

4.1 Jambalaya

É um plug-in criado para o Protégé utilizado para visualizar ontologias em diversos formatos (Storey et al., 2001). O Jambalaya é baseado no ShriMP (Simple Hierarchial Multi-Perpective), que é uma técnica de visualização da informação para gerir as estruturas de sub-relacionamentos de redes complexas da informação com

o zoons animados. Ele também oferece visões permutáveis dos gráficos para uma navegação interativa da ontologia, onde o usuário pode seguir relações por saltos entre os conceitos relacionados.

Atualmente este plug-in oferece visão permutáveis dos gráficos para uma navegação interativa da ontologia. Suporta seis tipos de layouts: Grid, Spring, Radial, Árvore, Sugiyama e TreeMap. Na figura abaixo apresentado o plug-in.

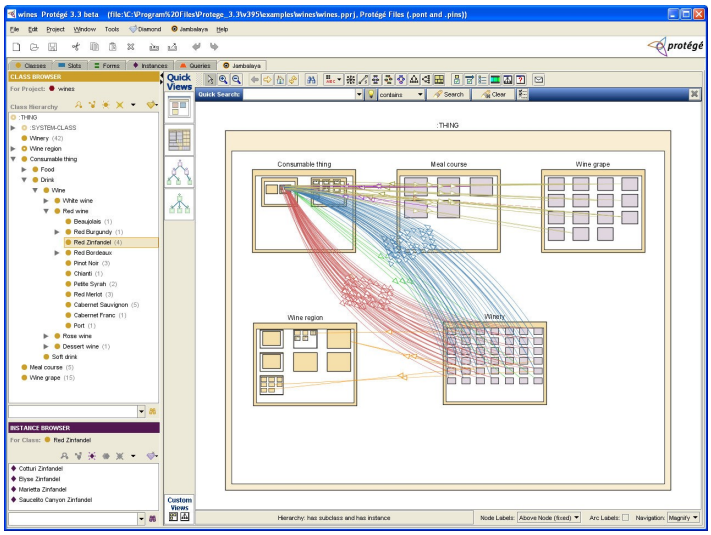

Figura 10: *Tela do Protégé com o plug-in Tab Jambalaya.*

4.2 OntoViz

É uma Tab plug-in do Protégé para visualização de ontologias (Fuit et al. 2004). Ela possibilita a visualização de classes e instâncias. Há principio, um esquema de árvore clássica pode ser elaborado, tanto para a hierarquia de classes e para as instâncias, depois disso, uma visualização pode ser vista por uma visualização mesclada que permite zoom gradual do geral para conceitos pais específicos.

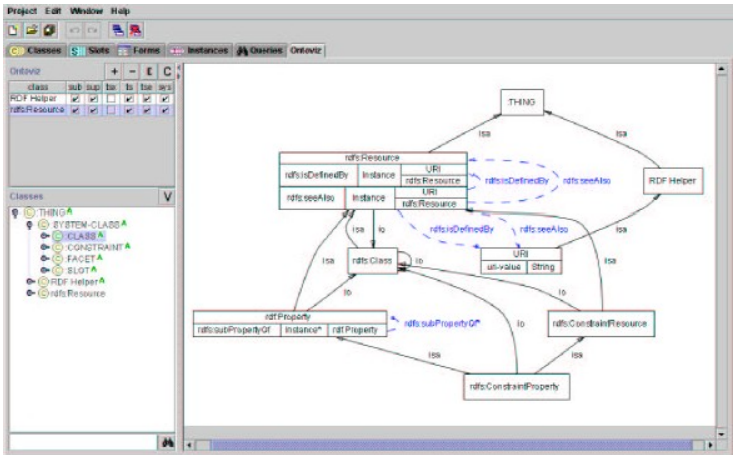

Figura 11: *OntoViz Tab do Protégé.*

4.3 TGVizTab

TGVizTab (TouchGraph Visualisation Tab) é um plugin para Protégé que permite a visualização de ontologias utilizando a biblioteca TouchGraph. TGViz Tab é um genérico, dinâmico e personalizável TouchGraph para ontologias no ambiente Protégé, ele também permite que os gráficos sejam salvos em XML e visualizados por outros aplicativos TouchGraph. A Figura 12 mostra a interface do usuário TGVizTab.

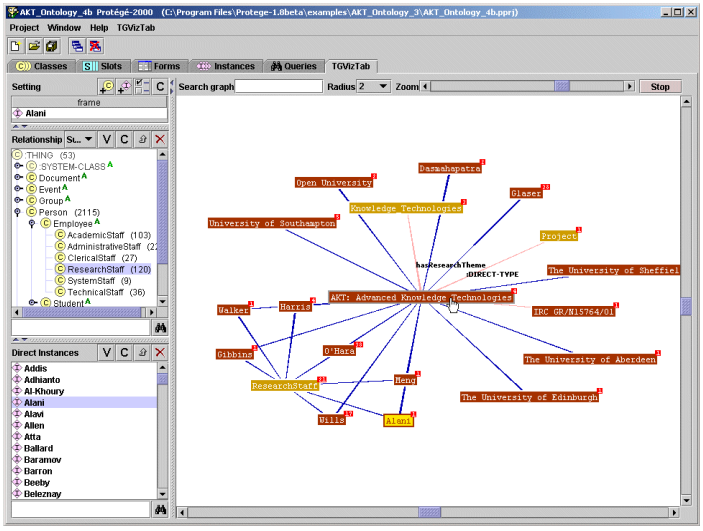

Figura 12: TGVizTab *Tab do Protégé.*

5 Conclusão

A visualização computacional surgiu para facilitar o entendimento de conjuntos de dados de alta complexidade. Ela está sendo muito importante para diversas áreas. Na área de ontologias, já existem excelentes ferramentas, mas ainda é possível evoluir e mostrar outros tipos de visualizações, sem ser a visualização em árvore ou derivados. Um exemplo de acréscimo, poderia ser feito uma análise da estruturação de diversas ontologias e não somente de uma.

6 Referências

Almeida, M. B. e Bax, M. P. 2003. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. Ci. Inf., Brasília, v. 32, n. 3, 7-20. DOI= 10.1590/S0100-19652003000300002.

Berners-Lee, T., Hendler, J. e Lassila, O. 2001. The Semantic Web. Scientific American, 34–43, http://www.scientificamerican.com/article.cfm?id=the-semantic-web, acesso em Out. 2009.

Brown, J., Gittens, S. 2010. Treemap. Disponível em: <http://www.otal.umd.edu/Olive/Class/Trees/>. Acesso em: set. 2010.

Chandrasekaran, B., Josephson, J. R., e Benjamins, V. R. 1999. What Are Ontologies, and Why Do We Need Them?. IEEE Intelligent Systems 14, 1, 20-26. DOI=http://dx.doi.org/10.1109/5254.747902.

Corazzon, R. 2010. Theory and History of Ontology. A Resource Guide for Philosophers. http://www.formalontology.it/, acesso em Jan. 2010.

Eklund, P.; Roberts, N.; Green, S.; 2002. OntoRama: Browsing RDF ontologies using a hyperbolic-style browser. Proceedings. First International Symposium on Cyber Worlds. ISBN: 0-7695-1862-1. 405-411.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L. e Patel-Schneider, P. F. 2001. OIL: An Ontology Infrastructure for the Semantic Web. IEEE Intelligent Systems, 16(2), 38-45. DOI=http://dx.doi.org/10.1109/5254.920598.

Ferreira, C. B. R. e Nascimento, H. A. D., 2005 Visualização de Informações – Uma abordagem Prática. Anais do XXV Congresso da Sociedade Brasileira de Computação. São Leopoldo, RS, Brasil, pp. 1262-1312.

Fluit, C.; Sabou, M. e Harmelen, F. Van. 2004 Supporting user tasks through visualisation of light-weight ontologies. S. Staab and R. Studer, (eds.) Handbook on Ontologies, pp. 415–434. Springer-Verlag, Berlin.

Golbreich, C., Horridge, M., Horrocks, I., Motik, B. e Shearer, R. 2007. OBO and OWL: leveraging semantic web technologies for the life sciences. In Proceedings 6th International Semantic Web Conference (ISWC 2007).

Gruber, T. R. 1993. A translation approach to portable ontology specifications. Knowl. Acquis. 5(2), 199-220. DOI= http://dx.doi.org/10.1006/knac.1993.1008.

Guarino, N. e Giaretta, P. 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing.

Horridge, M., Knublauch, H., Rector, A., Stevens, R. e Wroe, C. 2004. A practical guide to building OWL ontologies using the protégé-OWL plugin and CO-ODE tools edition 1.0. Technical report, University Of Manchester. http://www.co-ode.org/resources/, acesso em Out. 2009.

McGuinness, D. L. e Harmelen F. (Editors). 2004. OWL Web Ontology Language Overview. W3C. http://www.w3.org/TR/2004/REC-owl-features-20040210/, acesso em Jun. 2009.

Noy, N. F. e McGuinness, D. L. 2001. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05.

Ouellet, R. e Ogbuji, U. 2002. Introduction to DAML: Part I. http://www.xml.com/pub/a/2002/01/30/daml1.html, acesso em Jan. 2010.

Shneiderman, B. 1992. Tree Visualization with TreeMaps: 2D Space-Filling Approach. ACM Transactions on Graphics, v. 11, n. 1(January), 92-99.

Smith, B. e Welty, C. 2001. Ontology: Towards a New Synthesis. *In*: ACM International Conference on Formal Ontology and Information Systems – FOIS'01, 3-9.

Storey, M.A.; Musen, M.; Silva, J.; Best, C.; Ernst,N.; Fergerson, R. e Noy, N.. 2001 Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protege. Workshop on Interactive Tools for Knowledge Capture,

Victoria, B.C. Canada.

Uschold, M. 1998. Knowledge level modelling: concepts and terminology. The Knowledge Engineering Review, 13(1), 5-29.

Uschold, M. e Grüninger, M. 1996. Ontologies: principles, methods and applications. The Knowledge Engineering Review, 11(2), 93-155.