Modelagem Ontológica no Apoio à Modelagem Conceitual

Maria Lúcia Bento Villela[1], Alcione de Paiva Oliveira[2], José Luís Braga[2]
[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)
Avenida Antônio Carlos 6627 - Pampulha - 31270-901 – Belo Horizonte – MG
[2]Departamento de Informática - Universidade Federal de Viçosa (UFV)
Avenida P. H. Rolfs s/n - Campus UFV - 36570-000 - Viçosa – MG
e-mail: [mlbv, alcione, zeluis]@dpi.ufv.br

Resumo

A modelagem conceitual consiste numa atividade importante no projeto de sistemas, pois é a partir dela que se obtém estruturação dos conceitos abstraídos de um domínio do mundo real, permitindo sua incorporação em um sistema de informação. Porém, para que a modelagem conceitual possa ser uma descrição adequada da realidade do domínio do problema, ela deve apresentar informações precisas e claras, que podem ser obtidas a partir de uma análise mais detalhada das propriedades dos objetos de um domínio. Tal análise, denominada análise ontológica, pode ser utilizada como fonte de conhecimento para modeladores conceituais, possibilitando minimizar a ocorrência de erros semânticos em seus modelos. Este artigo realiza uma revisão de literatura sobre o uso de ontologia em modelagem conceitual de sistemas de informação e apresenta uma proposta preliminar, denominada VERONTO, do uso de meta-propriedades ontológicas para validação de modelos conceituais expressos por meio de diagramas de classe, visando torná-los mais manuteníveis.

Abstract

Conceptual modeling is a core activity in systems analysis and design. The elicitation of concepts from the real world problem environment and their representation in a form suitable to be effectively used in information systems is the key to success in systems design and implementation. Although there are many theories and methods proposed to achieve good results in this activity, there still is a lack of formal and easy to use methods that could enhance the semantical adherence between a model and its real world counterpart. Ontological analysis is a promising technique to decrease this gap. It provides tools for thinking about knowledge and data structures and their mappings to models that will fit better to application needs. This paper presents a formal technique so called VERONTO, that provides the necessary tools to carry on ontological analysis based on ontological properties on a meta-level as related to the models themselves. The resulting model is represented as a class diagram Experimental results show that the use of VERONTO in existing application models eventually enhance them.

1. Introdução

A modelagem conceitual consiste numa atividade importante no projeto de sistemas, pois é a partir dela que se obtém a estruturação dos conceitos abstraídos de um domínio do mundo real, permitindo sua incorporação em um sistema de informação.
O objetivo da modelagem conceitual é construir uma representação de alta qualidade de um fenômeno selecionado em algum domínio. Os modelos conceituais resultantes devem facilitar projeto, implementação, operação e manutenção de sistemas de informação. Como as atividades de modelagem conceitual ocorrem normalmente durante as fases iniciais do processo de desenvolvimento de sistemas, erros e omissões de modelagem que não forem detectados inicialmente podem ter repercussões custosas. Dessa forma, o aprimoramento das metodologias que guiem a construção da modelagem conceitual é de suma importância.

Para que a modelagem conceitual possa ser uma descrição adequada da realidade do domínio do problema, ela deve apresentar informações precisas e claras, não permitindo a ocorrência de ambigüidades sobre os aspectos que devem ser modelados.
Porém, nem sempre os profissionais responsáveis pela tarefa de modelagem possuem um conhecimento claro sobre os objetos do domínio e termos que o denotam, o que pode levar a modelos conceituais que apresentam falhas sob o aspecto semântico. Uma análise mais detalhada das propriedades dos objetos de um domínio, denominada de análise ontológica, pode ser utilizada como fonte de conhecimento para modeladores conceituais, possibilitando minimizar esses erros.
A ontologia é um ramo da filosofia que estuda a organização e a natureza do mundo [3]. Na computação e, particularmente na sub-área de representação do conhecimento, o termo ontologia tem sido utilizado para denotar o registro das descrições dos conceitos e suas relações abstraídas de um domínio. Moreira [14] apresenta um estudo detalhado sobre as diferenças ente o significado do termo na computação e na filosofia.
Existem, na literatura, alguns trabalhos que relacionam ontologias e modelagem conceitual, seguindo basicamente duas linhas distintas. A primeira linha utiliza modelos ontológicos como ferramentas de validação de elementos de modelagem conceitual, no sentido de fornecer uma definição precisa dos mesmos e, conseqüentemente, obter modelos conceituais com uma semântica clara e bem definida [11, 12, 16, 21]. A segunda linha utiliza ontologias específicas de domínio como uma ferramenta de apoio à tarefa de modelagem conceitual, no sentido de fornecer conhecimento do mundo real sobre o domínio a ser modelado [17, 19]. No entanto, faltam propostas que mostrem como adequar a modelagem ontológica às linguagens e processos de especificação de software.
O objetivo deste trabalho é apresentar uma revisão de literatura sobre o uso de ontologia em modelagem conceitual de sistemas de informação, bem como apresentar uma proposta preliminar do uso das meta-propriedades ontológicas apresentadas em [5, 6] para validação de modelos conceituais expressos por meio de diagramas de classe. A contribuição principal deste trabalho é apresentar um mapeamento das categorias geradas pela combinação das meta-propriedades nos elementos do diagrama de classes da UML, permitindo, deste modo, a aplicação das restrições para relacionamentos hierárquicos, propostas por [5, 6]. Espera-se que o uso destas restrições permita a construção de modelos conceituais mais manuteníveis e flexíveis. A aplicação destas meta-propriedades exige uma análise ontológica dos elementos do domínio. Esta análise ontológica, em conjunto com as etapas do desenvolvimento, dá origem a uma nova metodologia para construção de sistemas, cuja etapa de verificação do modelo conceitual em função das restrições ontológicas é denominada neste trabalho como VERONTO (VERficação ONTOlógica).
Na próxima seção, é discutido o termo Ontologia, são apresentados também as meta-propriedades introduzidas em [5, 6] para construção de ontologias e os tipos de propriedades propostos em [22]. Na seção 3, é introduzida uma proposta de uso da meta-propriedades para validação de modelos conceituais expressos na forma de diagramas de classes e, na seção 4, são apresentadas as conclusões do presente trabalho.

2. Ontologias, Meta-propriedades e Tipos de Propriedades

Modelagem conceitual consiste na identificação, análise e descrição dos elementos e restrições de um domínio do mundo real, normalmente utilizando uma linguagem de modelagem, a fim de que tais elementos sejam incorporados em um sistema de informação. Já modelagem ontológica pode ser definida como a criação de uma ontologia específica de domínio, através da captura das entidades relevantes do domínio e incorporação das mesmas

em um conjunto de categorias que revelam sua natureza, por meio de uma linguagem de especificação de ontologia.
Como pode-se perceber, existe uma forte semelhança entre esses dois tipos de modelagem, uma vez que ambas objetivam capturar e modelar elementos do mundo real. Porém, enquanto na modelagem conceitual busca-se estabelecer as relações entre os conceitos abstraídos de um domínio, na modelagem ontológica o objetivo é identificar os objetos e entender sua natureza por meio da descrição de suas propriedades. Segundo Poli [15], a primeira possui um caráter epistemológico, em oposição ao caráter ontológico da segunda. Dessa forma, pode-se deduzir que as duas modelagens são possivelmente complementares, ou seja, a modelagem ontológica pode constituir-se numa base para modelagem conceitual, no sentido de prover ao projetista, de forma clara e sem ambigüidades, o conhecimento necessário sobre o domínio a ser modelado.

2.1. Ontologia

A palavra Ontologia, definida originalmente na Filosofia como o ramo da metafísica que estuda "os tipos de coisas que existem" no mundo, tem sido utilizada com diferentes sentidos na área de Inteligência Artificial e Engenharia do Conhecimento, o que tem provocado uma certa descaracterização do termo.
Guarino e Giaretta [2], com o objetivo de fornecer um esclarecimento a respeito do significado do termo "Ontologia", apresentam algumas possíveis interpretações para o mesmo, sendo visto tanto como entidade semântica conceitual, quanto como um objeto sintático específico. No entanto, os autores concentram-se numa análise minuciosa da definição de Ontologia fornecida por Gruber [1], como sendo "uma especificação explícita de uma conceitualização". Esta definição tem sido a mais utilizada pela comunidade da área de Inteligência Artificial e pode ser classificada como uma interpretação sintática para Ontologia [2], uma vez que trata a ontologia como um produto concreto sintático e manipulável.

2.2. Ontologia de Propriedades

2.2.1. Noções Básicas. Neste item, são descritas informalmente as mais importantes noções filosóficas - *identidade, essência, unidade* e *dependência* - que se constituem bases para as meta-propriedades apresentadas por Guarino e Welty [5, 6].
A noção de identidade, aqui considerada, está relacionada à maneira como nós em geral interagimos e, em particular, reconhecemos entidades individuais no mundo ao nosso redor [5]. Em [5, 8, 22] são realizadas comparações entre a noção de identidade e noções análogas de condição de membro de classe, ID globalmente único e chave primária, em modelagem conceitual de bases de conhecimento, sistemas orientados a objetos e banco de dados, respectivamente; porém nenhuma delas captura completamente a noção de identidade aqui apresentada.
De acordo com [5, 8, 9, 22], o primeiro passo para compreender melhor a noção de identidade requer considerar as distinções entre identidade e unidade. Identidade está relacionada ao problema de distinguir uma instância específica de uma certa classe de outras instâncias da mesma classe por meio de uma propriedade característica, que é única para cada instância. Unidade, por outro lado, está relacionada ao problema de distinguir as partes de uma instância do resto do mundo, por meio de uma relação de unificação que une suas partes, e nada mais. Por exemplo, a questão "Este é o meu cachorro?" é um problema de identidade, enquanto a questão "Esta coleira é do meu cachorro?" é um problema de unidade.
Quando se leva em consideração o tempo, deve-se admitir que um indivíduo possa permanecer o mesmo, enquanto exibe diferentes propriedades em tempos diferentes. Mas

quais propriedades podem mudar e quais não podem? E como é possível re-identificar uma instância de uma propriedade após algum tempo? A primeira questão leva à noção de propriedade essencial, que não muda no decorrer do tempo, e é base da definição de rigidez, discutida abaixo, enquanto a última está relacionada à distinção entre identidade sincrônica e diacrônica, discutida detalhadamente em [7, 9].
A noção de dependência [8, 22] pode envolver diferentes relações, tais como aquelas existentes entre pessoas e seus pais. Pode-se distinguir entre propriedade extrínseca e intrínseca, conforme elas dependam ou não de outros objetos além de suas próprias instâncias. Uma propriedade intrínseca é tipicamente algo inerente ao indivíduo, não dependente de outros indivíduos, como ter um coração ou uma impressão digital. Propriedades extrínsecas não são inerentes e possuem uma natureza relacional, como "ser o pai de Fred".

2.2.2. Meta-propriedades. As meta-propriedades, apresentadas por Guarino e Welty [5, 6] representam o comportamento de uma propriedade em relação às noções de *identidade, unidade, essência* e *dependência*, vistas anteriormente. Tais meta-propriedades visam facilitar a compreensão correta da natureza das propriedades pertencentes a um domínio.

2.2.2.1 Rigidez. A meta-propriedade Rigidez é originada a partir da noção de Essência, que está relacionada ao fato uma propriedade se aplicar a um elemento do domínio (também chamado de instância da referida propriedade), enquanto tal elemento existir.
Uma propriedade pode ser classificada, com relação à meta-propriedade Rigidez, conforme mostrado em [5, 8], da seguinte forma:

- Propriedade Rígida (denotada pelo símbolo $+R$) → propriedade que é essencial para todas as suas instâncias, ou seja, uma propriedade ϕ tal que: $\forall x \phi(x) \rightarrow \phi(x)$, onde o operador indica "em todos os possíveis mundos".
- Propriedade Não-Rígida (denotada pelo símbolo $-R$) → propriedade que não é essencial para alguma de suas instâncias, ou seja, $\exists x \phi(x) \wedge \neg \phi(x)$.
- Propriedade Anti-Rígida (denotada pelo símbolo $\sim R$) → propriedade que não é essencial para todas as suas instâncias, ou seja, $\forall x \phi(x) \rightarrow \neg \phi(x)$
- Propriedade Semi-Rígida (denotada pelo símbolo $\neg R$) → propriedade não-rígida, mas não anti-rígida.

Pode-se deduzir que uma propriedade anti-rígida é também uma propriedade não-rígida, porém o primeiro conceito é mais forte que o último, uma vez que restringe todas as instâncias de uma propriedade, enquanto o último restringe no mínimo uma instância.
A meta-propriedade Rigidez não é herdada ao longo da hierarquia de propriedades; o que permite que uma propriedade rígida subjugue uma propriedade não-rígida.
Para melhor esclarecimento, considere, por exemplo, as propriedades FUNCIONÁRIO e PESSOA. Obviamente, todas as instâncias de FUNCIONÁRIO são também instâncias de PESSOA, porém uma instância da primeira propriedade poderá deixar de ser um funcionário, uma vez que poderá ficar desempregado ou se tornar um empresário ou trabalhador autônomo, mas jamais deixará de ser uma instância de pessoa. Com isso, conclui-se que a propriedade FUNCIONÁRIO é anti-rígida (~R), pois todas as suas instâncias não as serão necessariamente durante toda a sua existência, e a propriedade PESSOA é rígida (+R), uma vez que todas suas instâncias assim a serão por toda a sua existência.

2.2.2.2 Dependência. A meta-propriedade Dependência Externa é originada a partir da noção filosófica de Dependência e é definida da seguinte forma [5, 6, 8, 22]:
"Uma propriedade ϕ é externamente dependente, denotada pelo símbolo +D (-D, caso contrário), de uma propriedade ψ se, para todas as suas instâncias x, necessariamente deve existir alguma instância de ψ, que não seja parte nem constituinte de x".

As relações de parte e constituinte são discutidas com maiores detalhes em [6].
Por exemplo, a propriedade CLIENTE é externamente dependente da propriedade EMPRESA, pois somente poderá existir uma instância da propriedade CLIENTE se existir uma instância da propriedade EMPRESA, da qual a primeira adquira produtos ou serviços prestados. Já a propriedade PESSOA não é externamente dependente de uma instância de CORAÇÃO ou CORPO, pois uma instância de PESSOA possui um coração como parte e é constituída de um corpo.

2.2.2.3 Identidade. Para entendimento das meta-propriedades baseadas na noção de Identidade, primeiro deve-se definir o conceito de Condição de Identidade (IC) para uma propriedade ϕ, que consiste numa relação ρ, satisfazendo a seguinte fórmula:

$$\phi(x) \wedge \phi(y) \rightarrow (\rho(x,y) \leftrightarrow x=y)$$

Dessa forma, uma propriedade "executa uma IC", denotada pelo símbolo +I (-I, caso contrário), se existirem, em tempos distintos, instâncias que, ou: 1) se satisfazem a mesma IC, então elas são iguais (IC suficiente); ou, 2) se são iguais, então satisfazem à mesma IC (IC necessária).
Além disso, uma propriedade ϕ "fornece uma IC", denotada pelo símbolo +O (-O, caso contrário), apenas se ela for rígida (+R), executar uma IC (+I), e esta não for executada por nenhuma propriedade que a subjuga. Isto significa que, se ϕ herda ICs diferentes (mas compatíveis) de múltiplas propriedades, ela ainda permanece fornecendo um IC.
IC's são herdados ao longo da hierarquia de propriedades. Assim, uma propriedade não-rígida pode executar uma IC, se e somente se esta for herdada de uma propriedade rígida que a subjuga.
Por exemplo, a propriedade CLIENTE, que é não-rígida, pode apenas executar suas ICs (+I), herdando-as de propriedades rígidas que a subjugam, como a propriedade PESSOA (+O). Isto se dá devido ao fato de uma mesma pessoa poder ser cliente em diferentes tempos de diferentes empresas. Assim, uma IC fornecida por cliente, como, por exemplo, ter o mesmo código, pode ser apenas local; enquanto que uma IC fornecida pela propriedade PESSOA, como ter o mesmo número de RG ou a mesma impressão digital, terá validade global. Já a propriedade VERDE não executa IC (-I), pois não existe IC necessária nem suficiente que identificará ou re-identificará coisas verdes, pelo simples fato de serem verdes.
Uma propriedade que executa uma IC (+I) é chamada de "sortal" (ou ordenável) [18]. Para reconhecer que uma propriedade é um "sortal", não é necessário saber qual IC ela executa: a distinção entre "sortal" e não "sortal" é normalmente suficiente para estabelecer ordem em taxonomias [5, 6].
Em [7, 9] são fornecidas discussões mais detalhadas sobre o conceito de identidade.

2.2.2.4 Unidade. Para entendimento das meta-propriedades baseadas na noção de Unidade, primeiro deve-se definir o que significa para uma certa propriedade possuir uma Condição de Unidade (UC), isto é, constituir-se um todo:

- Seja ω uma relação de equivalência. Em um dado tempo t, um objeto x é um todo contingente sob ω se cada parte de x estiver ligada por ω a todas as suas outras partes e a nada mais.
- Uma noção mais forte de todo pode ser definida por assumir que uma UC deve sustentar para um objeto por toda a sua existência: "Seja ω uma relação de equivalência. Um objeto x é um todo intrínseco sob ω se, a qualquer tempo em que x exista, ele é um todo contingente sob ω".

Dessa forma, uma propriedade "executa uma UC", denotada pelo símbolo $+U$, se e somente se existe uma relação de equivalência ω tal que todas as suas instâncias são todos intrínsecos sob ω.
Guarino e Welty [6, 8, 22] salientam que, como ocorre com a meta-propriedade Rigidez, pode ser útil em alguns casos distinguir entre propriedades que não executam uma UC comum para todas as suas instâncias, daquelas propriedades cujas todas as instâncias não sejam todos intrínsecos. Um exemplo do primeiro caso seria a propriedade AGENTE LEGAL, cuja todas as instâncias constituem-se todos intrínsecos (algumas pessoas, algumas empresas), porém não existe uma única relação ω para todas elas (desde que pessoas e empresas podem ter diferentes UCs). MONTE DE MATÉRIA pode ser um exemplo do segundo caso, desde que nenhuma de suas instâncias constitui-se todo intrínseco.
Uma propriedade possui "anti-unidade", denotada pelo símbolo ~U, se cada uma de suas instâncias não constituem todos intrínsecos.
Uma propriedade "não possui unidade", denotada pelo símbolo –U, se ela não executa uma UC comum para todas as suas instâncias, ou se suas instâncias não se constituem inteiros intrínsecos (~U).
Em [7] são discutidos mais detalhadamente os conceitos ligados a Unidade. Ainda nesse artigo, os autores fazem a seguinte definição: "Uma propriedade que executa uma IC (+I) e também executa uma UC (+U) é chamada de contável".

2.2.3. Restrições Taxonômicas. Guarino e Welty [5, 6, 8, 22] mostram como as meta-propriedades vistas anteriormente impõem algumas restrições naturais na estrutura taxonômica da ontologia, o que, por sua vez, facilita sua integração, comparação e entendimento. Abaixo, sejam ϕ e ψ duas propriedades arbitrárias. A notação ϕ^M é utilizada para indicar que a propriedade ϕ possui a meta-propriedade M.

2.2.3.1 Restrições sobre Rigidez:

- Uma propriedade anti-rígida não pode subjugar uma propriedade rígida, ou seja, $\neg(\psi^{+R} \rightarrow \phi^{\sim R})$.

2.2.3.2 Restrições sobre Identidade:

- Uma propriedade que executa uma IC não pode subjugar uma propriedade que não executa uma IC, ou seja, $\neg(\psi^{-I} \rightarrow \phi^{+I})$;
- Propriedades com IC's incompatíveis são disjuntas.

Para melhor esclarecer esta última restrição, Guarino [4] mostra um exemplo considerando as duas propriedades CONJUNTO e CONJUNTO-ORDENADO. À primeira vista, pode parecer que a última é subjugada pela primeira; porém, ao analisar suas ICs, chega-se à conclusão que as duas propriedades possuem ICs incompatíveis, pois "ter os mesmos membros" é uma IC suficiente para CONJUNTO, mas não para CONJUNTO-ORDENADO. Neste mesmo artigo, o autor propõe que o princípio proposto por Lowe [13], transcrito a seguir, seja adotado como o princípio básico para ontologias bem-fundamentadas:
"Nenhum indivíduo pode instanciar, ao mesmo tempo, duas propriedades, se estas possuem diferentes condições de identidade (ICs) com elas associadas".

2.2.3.3 Restrições sobre Dependência:

- Uma propriedade dependente não pode subjugar uma propriedade não dependente, ou seja, $\neg(\psi^{-D} \rightarrow \phi^{+D})$.

2.2.3.4 Restrições sobre Unidade:

- Uma propriedade que executa uma UC não pode subjugar uma propriedade que não possui unidade, ou seja, $\neg(\psi^{-U} \rightarrow \phi^{+U})$;
- Uma propriedade que possui anti-unidade não pode subjugar uma propriedade que executa uma UC, ou seja, $\neg(\psi^{+U} \rightarrow \phi^{\sim U})$;
- Propriedades com UC's incompatíveis são disjuntas.

Em [9] são examinados alguns exemplos de relacionamentos hierárquicos que, apesar de parecem corretos à primeira vista, tornam-se problemáticos quando as meta-propriedades relacionadas a Identidade e a Unidade são consideradas, por não respeitarem algumas das restrições acima expostas.

2.2.4. Suposições considerando Identidade. Guarino e Welty [5, 6, 8, 22] colocam as seguintes suposições, levando em consideração a noção de *Identidade*, adaptadas de [13]:

- Individualização Sortal: cada elemento do domínio deve instanciar alguma propriedade executando uma IC ("Nenhuma entidade sem identidade").
- Expansibilidade Sortal: se duas entidades (instâncias de diferentes propriedades) são a mesma, elas devem ser instâncias de uma única propriedade executando uma condição para suas identidades ("Cada entidade deve instanciar, no mínimo, uma propriedade rígida").

2.3. Tipos de Propriedades

Em [5] é proposta a combinação das meta-propriedades em oito tipos de propriedades, gerando o que é denominado "Ontologia Formal de Propriedades". A Tabela 1 consiste em uma adaptação da Tabela 1 mostrada em [5], uma vez que mostra os sete tipos mais importantes de propriedades, juntamente com a combinação das metas-propriedades que dão origem aos mesmos. Segundo Welty e Guarino [22], os tipos de propriedades facilitam a tarefa do modelador atribuir significado aos elementos do domínio uma vez que incluem uma descrição intuitiva e independente de domínio de como cada propriedade deve ser usada no domínio. Como os tipos de propriedades são baseados nas meta-propriedades, estes também restringem como os elementos do domínio podem ser relacionados hierarquicamente.
A seguir será apresentada uma breve descrição, adaptada de [5], de cada um dos tipos de propriedades da Tabela 1.

Tabela 1. Propriedades formadas a partir da combinação das meta-propriedades (Adaptada de [5]).

+O	+I	+R	+D	Tipo	SORTAL
			-D		
-O	+I	+R	+D	Quase-Tipo	
			-D		
-O	+I	~R	+D	Papel Material	
-O	+I	~R	-D	Sortal com Fases	
-O	-I	+R	+D	Categoria	NÃO SORTAL
			-D		
-O	-I	~R	+D	Papel Formal	
-O	-I	~R	-D	Atribuição	
		¬R	+D		
			-D		

Tipos consistem em propriedades rígidas (+R) e que fornecem sua própria IC (+O). Como eles são os únicos que fornecem IC, de acordo com o Princípio da Individualização Sortal, visto na seção 2.2.4, cada elemento do domínio modelado deve instanciar pelo menos um Tipo, representando este as propriedades mais importantes de um domínio. *Quase-Tipos*

consistem em propriedade rígidas (+R), que apenas executam e não fornecem IC (+I, -O). Quase-Tipos podem ser utilizados para agrupar entidades do domínio baseado em combinações de propriedades úteis que não afetam identidade. *Papéis Materiais* consistem em propriedades anti-rígidas (~R) e dependentes (+D), mas que executam IC (+I), herdada de algum Tipo que a subjuga. Papéis consistem em propriedades que expressam a função desempenhada por uma entidade do domínio em relacionamento específico entre duas ou mais entidades. *Sortais com Fases* consistem em propriedades que executam IC (+I), são anti-rígidas (~R) e independentes (-D). Eles não fornecem uma IC global (-O), uma vez que fornecem apenas uma IC local, que corresponde a uma certa fase temporal de suas instâncias. Entidades deste tipo possuem ICs que mudam no decorrer do tempo e em fases discretas.
Categorias consistem em propriedades rígidas (+R), que, porém, não fornecem nem executam uma IC (-O, -I). Dessa forma, elas não podem ser subjugadas por nenhuma propriedade sortal, o que as leva a estarem presentes normalmente nos níveis mais altos em uma taxonomia, sendo utilizadas para fins classificatórios. *Papéis Formais* consistem em propriedades anti-rígidas (~R), dependentes (+D) e que não executam IC (-I), representando papéis mais genéricos que aparecem nos níveis mais altos da hierarquia de papéis. *Atribuições* consistem em propriedades que não fornecem e nem executam IC (-O, -I) e são anti-rígidas (~R) e não dependentes (-D) ou semi-rígidas (¬R). Elas representam os atributos ou qualidades das entidades de um domínio, como cor, forma, tamanho, etc.

3. VERONTO: Uma Proposta de Utilização das Meta-propriedades e Regras baseadas em Estudos Ontológicos Formais no Apoio à Modelagem Conceitual

Pode-se perceber, pelo exposto anteriormente, que o estudo ontológico do domínio pode ser utilizado como ferramenta de apoio ao projetista em modelagem conceitual de sistemas, uma vez que revela conhecimentos sobre a natureza dos objetos que povoam o domínio a ser modelado. O estudo ontológico pode ser feito em uma etapa anterior ou posterior à criação do diagrama de classes do domínio, para ser usado então em sua construção ou validação, respectivamente.
Em [5, 6, 7, 8, 9, 22], foi verificado que as meta-propriedades baseadas nas noções filosóficas de *identidade*, *rigidez*, *unidade* e *dependência*, impõem uma série de restrições nos relacionamentos de um objeto, principalmente no que diz respeito aos relacionamentos taxonômicos. Tais restrições também podem ser úteis no estabelecimento de relacionamentos correspondentes a generalizações/especializações, na modelagem conceitual.
Foi constatada na literatura a ausência de trabalhos que fazem uso das meta-propriedades em questão, bem como as restrições por elas impostas, como base para uma análise detalhada dos elementos em modelos conceituais, expressos na forma de diagrama de classes. Assim, propõe-se uma técnica, denominada VERONTO, para utilização de tais meta-propriedades na validação de modelos conceituais especificados por meio de diagramas de classe de domínio em UML. VERONTO é, basicamente, uma proposta de mapeamento dos tipos de propriedades, vistos anteriormente na Tabela 1, nos elementos do diagrama de classes da UML. Este mapeamento torna possível a aplicação das restrições impostas pelas meta-propriedades às associações entre as classes do diagrama. Com isto, espera-se a obtenção de modelos conceituais mais consistentes e mais fáceis de serem compreendidos e compartilhados, podendo levar à implementação de sistemas mais manuteníveis.
A maior parte da proposta baseia-se nas meta-propriedades; no entanto, estas são aplicáveis, diretamente, apenas aos relacionamentos hierárquicos. Por isso, buscou-se complementar a proposta com as regras de restrições a relacionamentos, preconizadas nos estudos ontológicos formais de Wand et al. [21]. No entanto, neste artigo apresenta-se, apenas, o mapeamento dos elementos da ontologia proposta em [7, 22] nos elementos do diagrama de classes da UML.

3.1. Mapeamento dos Tipos de Propriedades nos elementos do Diagrama de Classes da UML

Um estudo teórico das meta-propriedades propostas [10], bem como das regras que elas impõem aos relacionamentos hierárquicos entre os elementos do domínio, foi realizado. Foi definida então uma forma de associá-las aos elementos de modelagem conceitual, expressa na forma de diagramas de classes em UML, por meio dos tipos de propriedades, conforme detalhado na Tabela 2 abaixo.

Tabela 2. Associação das meta-propriedades e regras que estas impõem aos relacionamentos hierárquicos aos construtores de modelagem conceitual.

Meta-Propriedades	Classificação da Propriedade [22]	Construtor UML	Restrições sobre Relacionamentos Hierárquicos
+O +I +R	**Tipo**	**Classe concreta** Ex.: Pessoa, Gato	➢ Superclasse de classes que representem propriedades sortais (pois fornece uma IC para suas subclasses). ➢ Pode ser subclasse de classes correspondentes a Categorias, outros Tipos ou Quase-Tipos (únicos que correspondem a propriedades rígidas).
-O +I +R	**Quase-Tipo**	**Classe concreta** Ex.: Animais Invertebrados, Animais Herbívoros	➢ Deve ser subclasse, no mínimo, de uma classe correspondente a um Tipo, com IC/UC compatível (para herdar a IC). ➢ Pode ser subclasse de classes referentes a Categorias (corresponde a propriedade rígida) e Mixins (não recomendado). ➢ Pode ser superclasse de classes que representem propriedades sortais (pois transmite sua IC para suas subclasses).
-O +I ~R +D	**Papel Material**	**Classe concreta** Ex.: Estudante, Casado, Comida	➢ Deve ser subclasse de uma classe correspondente a, no mínimo, um Tipo, com IC/UC compatível (para herdar a IC). ➢ Pode ser subclasse de qualquer tipo de classe – recomenda-se classes referentes a propriedades rígidas (para fornecer a IC). ➢ Pode ser superclasse de classes correspondentes a papéis materiais (pois é a única que, além de ser anti-rígida, executa uma IC e é externamente dependente). ➢ Corresponde a um papel da superclasse em uma associação.
-O +I ~R -D	**Sortal com Fase**	**Classe concreta** Ex.: Lagarta, Borboleta	➢ Deve ser subclasse de uma classe correspondente a um Tipo, com IC/UC compatível (para herdar IC). ➢ Pode ser subclasse de qualquer classe correspondente a uma propriedade independente (recomenda-se que também seja rígida). ➢ Pode ser superclasse de quaisquer classes não rígidas (recomenda-se apenas classes correspondentes a sortal com fases e papéis materiais). ➢ Corresponde a uma fase de sua superclasse, representada por uma classe correspondente a um Tipo ou Quase-Tipo, que subjuga apenas classes correspondentes a sortal com fases,

Meta-Propriedades	Classificação da Propriedade [22]	Construtor UML	Restrições sobre Relacionamentos Hierárquicos
			que representam as possíveis fases pelas quais passa durante sua existência.
-O -I +R	**Categoria**	**Classe Abstrata** Ex.: Entidade Concreta, Entidade Abstrata	➢ Pode ser superclasse de classes de qualquer espécie. ➢ Normalmente, correspondem a classes de mais alto nível, dentro de uma hierarquia. ➢ Pode ser subclasse somente de classes também correspondentes a Categorias (pois não executa IC e é rígida). ➢ Classe utilizada para fins classificatórios.
-O -I ~R +D	**Papel Formal**	**Classe Abstrata** Ex.: Paciente, Instrumento	➢ Superclasse de classes que representam papéis materiais. ➢ Pode ser uma subclasse de classes correspondentes a Categorias e Papéis Formais (não executa IC). ➢ Corresponde a um papel de uma classe em uma associação. ➢ Classe utilizada para organizar a hierarquia de classes correspondentes a papéis.
-O -I ~R -D -O -I ¬R	**Atribuição**	**Atributo** Ex.: Cor, Forma	➢ Não fornece e não executa IC (-O, -I), também é anti-rígida e não dependente (~R, -D), ou semi-rígida (¬R). ➢ Corresponde a um Atributo de uma classe em UML.

Nas situações anteriores, se a classe executa uma UC (+U - inteiro) ela não pode ser subclasse de uma classe que possui anti-unidade (~U) (ou seja, cujas instâncias não são inteiros intrínsecos). E se a classe não possui unidade (-U) (não executa uma UC comum para todas as suas instâncias ou estas não são inteiros intrínsecos), ela não pode ser subclasse de uma classe que executa uma UC (+U – inteiro). Na Tabela 3 é realizada a associação das meta-propriedades relacionadas a identidade e unidade, bem como das restrições que elas impõem a relacionamentos "Parte-Todo", a construtores UML.

Tabela 3. Associação das meta-propriedades relacionadas a identidade e unidade e restrições que elas impõem a relacionamentos "Parte-todo" a construtores de modelagem conceitual.

Meta-Propriedades	Denominação Propriedade [7]	Construtor UML	Restrições sobre Relacionamentos "Parte-Todo"
+I +U	Indivíduo	**Classe concreta**	➢ Pode ser uma classe parte de uma classe correspondente a uma todo (+U) – Agregação compartilhada ➢ Pode ser uma classe todo em um relacionamento "parte-todo" cujas partes sejam também todos (+U) ou apenas partes (-U).
+I -U	Identificável	**Classe concreta**	➢ Pode ser apenas parte em um relacionamento "parte-todo" – Composição.
-I +U	Inteiro	**Não é uma classe concreta**	

3.2. Exemplo do uso do mapeamento

Para ilustrar como o mapeamento dos tipos de propriedades pode influenciar a confecção de diagramas de classe, é mostrado, a seguir, um pequeno exemplo, utilizando um diagrama de

classe desenvolvido por estudantes de graduação de um curso de Ciência da Computação. Primeiramente é apresentado o diagrama inicial, em seguida o diagrama é analisado de acordo com os tipos de propriedades. Finalmente, é apresentado o diagrama modificado, obedecendo às restrições impostas pelas meta-propriedades. A Figura 1 mostra o diagrama de classes para um sistema de controle de eventos, proposta em [20]. O gerenciamento de eventos envolve o controle de inscrições e presenças de participantes, emissão de certificados, programação de atividades nos eventos, controle de patrocinadores, funcionários e serviços terceirizados, controle de equipamentos utilizados nas atividades, controle de stands presentes em eventos e emissão de credenciais. O resultado da análise ontológica sobre a especificação de requisitos, efetuada em [20], bem como do diagrama de classes de domínio, é descrito a seguir.

A classe *Usuário* armazena informações sobre os usuários que utilizam o Sistema de Gerenciamento de Eventos. Ela foi classificada como: Anti-Rígida (~R) – todo Usuário não será necessariamente um Usuário durante toda a sua existência; Não Fornece Identidade (-O) – desde que uma mesma pessoa pode ser um Usuário diferentes vezes em diferentes sistemas, uma condição de identidade (IC) fornecida por Usuário pode ser apenas local, dentro de um determinado sistema; Executa Identidade (+I) – executa uma IC relacionada a pessoa de um modo geral; e Não Dependente (-D) – não depende de nenhuma outra classe. A partir desta análise, pode-se concluir que a classe *Usuário* representa, segundo Welty e Guarino [22], um *Papel Material*.

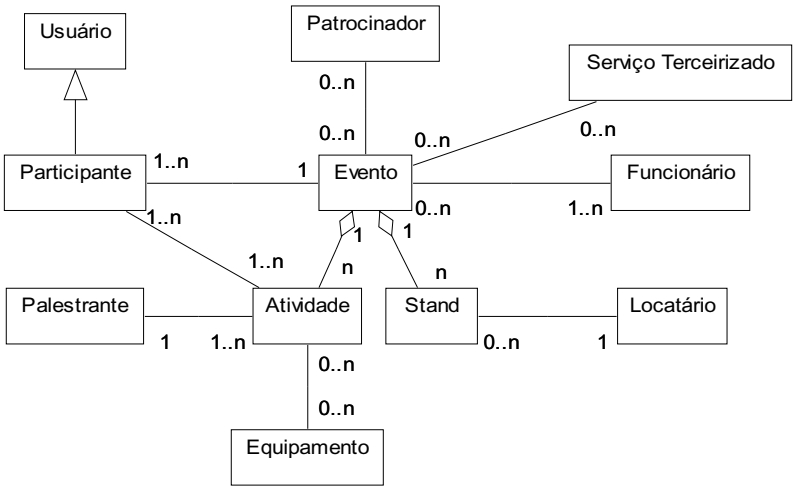

Figura 1. Diagrama de Classes de um Domínio de Gerenciamento de Eventos. Fonte: [20]

A classe *Participante* armazena informações pessoais sobre os participantes de um evento. Ela foi classificada como: Anti-Rígida (~R) – todo Participante não será necessariamente um Participante durante toda a sua existência; Não Fornece Identidade (-O) – desde que uma mesma pessoa pode ser um Participante diferentes vezes em diferentes eventos, uma condição de identidade (IC) fornecida por Participante pode ser apenas local, dentro de um determinado evento; Executa Identidade (+I) – executa uma IC relacionada a pessoa de um modo geral; e Dependente (+D) – é externamente dependente de *Evento*, pois alguém somente poderá ser um Participante se o for de um determinado evento. A partir desta análise, pode-se concluir que a classe *Participante* representa, segundo Welty e Guarino [22], um *Papel Material*.

A classe *Patrocinador* armazena informações especificas sobre os patrocinadores de eventos. Ela foi classificada como: Anti-Rígida (~R) – todo Patrocinador não será necessariamente um Patrocinador durante toda a sua existência; Não Fornece Identidade (-O) – desde que uma mesma empresa pode ser um Patrocinador diferentes vezes em diferentes eventos, uma condição de identidade (IC) fornecida por Patrocinador pode ser apenas local, dentro de um

determinado evento; Executa Identidade (+I) – executa uma IC relacionada a pessoa (jurídica) de um modo geral; e Dependente (+D) – é externamente dependente de *Evento*, pois alguém somente poderá ser um Patrocinador se o for de um determinado evento. A partir desta análise, pode-se concluir que a classe *Patrocinador* representa, segundo Welty e Guarino [22], um *Papel Material*.

A classe *Palestrante* armazena informações sobre as pessoas que apresentam atividades nos eventos. Ela foi classificada como: Anti-Rígida (~R) – todo Palestrante não será necessariamente um Palestrante durante toda a sua existência; Não Fornece Identidade (-O) – desde que uma mesma pessoa pode ser um Palestrante diferentes vezes em diferentes eventos, uma condição de identidade (IC) fornecida por Palestrante pode ser apenas local, dentro de um determinado evento; Executa Identidade (+I) – executa uma IC relacionada a pessoa de um modo geral; e Dependente (+D) – é externamente dependente de *Evento*, pois alguém somente poderá ser um Palestrante se o for em um determinado evento. A partir desta análise, pode-se concluir que a classe *Palestrante* representa, segundo Welty e Guarino [22], um *Papel Material*.

A classe *Funcionário* armazena informações pessoais sobre os funcionários da Empresa Organizadora de Eventos. Ela foi classificada como: Anti-Rígida (~R) – todo Funcionário não será necessariamente um Funcionário durante toda a sua existência; Não Fornece Identidade (-O) – desde que uma mesma pessoa pode ser um Funcionário diferentes vezes em diferentes empresas, uma condição de identidade (IC) fornecida por Funcionário pode ser apenas local, dentro de uma determinada empresa; Executa Identidade (+I) – executa uma IC relacionada a pessoa de um modo geral; e Não Dependente (-D) não depende de nenhuma outra classe. A partir desta análise, pode-se concluir que a classe *Funcionário* representa, segundo Welty e Guarino [22], um *Papel Material*.

A classe *Locatário* armazena informações pessoais sobre empresas que alugam stands nos eventos. Ela foi classificada como: Anti-Rígida (~R) – todo Locatário não será necessariamente um Locatário durante toda a sua existência; Não Fornece Identidade (-O) – desde que uma mesma empresa pode ser um Locatário diferentes vezes em diferentes eventos, uma condição de identidade (IC) fornecida por Locatário pode ser apenas local, dentro de um determinado evento; Executa Identidade (+I) – executa uma IC relacionada a pessoa (jurídica) de um modo geral; e Dependente (+D) – é externamente dependente de *Evento*, pois alguém somente poderá ser um Locatário de stand, se existir um evento do qual este faça parte. A partir desta análise, pode-se concluir que a classe *Locatário* representa, segundo Welty e Guarino [22], um *Papel Material*.

A classe *Serviço Terceirizado* armazena informações sobre empresas (segurança, buffet, limpeza) que prestam serviços em eventos. Ela foi classificada como: Anti-Rígida (~R) – todo Serviço Terceirizado não será necessariamente um Serviço Terceirizado durante toda a sua existência; Não Fornece Identidade (-O) – desde que um mesmo serviço terceirizado pode sê-lo diferentes vezes em diferentes eventos, uma condição de identidade (IC) fornecida por Serviço Terceirizado pode ser apenas local, dentro de um determinado evento; Executa Identidade (+I) – executa uma IC relacionada a pessoa (jurídica) de um modo geral; e Dependente (+D) – é externamente dependente de *Evento*, pois uma empresa somente poderá constituir-se em um Serviço Terceirizado dentro de um determinado evento. A partir desta análise, pode-se concluir que a classe *Serviço Terceirizado* representa, segundo Welty e Guarino [22], um *Papel Material*.

A classe *Equipamento* armazena informações referentes a equipamentos (retroprojetor, datashow etc.) utilizados nos eventos. Ela foi classificada como: Rígida (+R) – todo Equipamento assim o será necessariamente durante toda a sua existência; Fornece Identidade (+O) – cada Equipamento pode ser identificado globalmente, por meio de uma característica que lhe é própria; Executa Identidade (+I) – uma vez que fornece uma IC, obviamente

também a executa; e Não Dependente (-D) – não depende de nenhuma outra classe. A partir desta análise, pode-se concluir que a classe *Equipamento* representa, segundo Welty e Guarino [22], um *Tipo*.
A classe *Evento* armazena informações referentes a eventos promovidos pela empresa. Ela foi classificada como: Rígida (+R) – todo Evento assim o será necessariamente durante toda a sua existência; Fornece Identidade (+O) – cada Evento pode ser identificado globalmente, por meio de uma característica que lhe é própria; Executa Identidade (+I) – uma vez que fornece uma IC, obviamente também a executa; e Não Dependente (-D) – não depende de nenhuma outra classe. A partir desta análise, pode-se concluir que a classe *Evento* representa, segundo Welty e Guarino [22], um *Tipo*.
A classe *Stand* armazena informações referentes a stands contidos nos eventos. Ela foi classificada como: Rígida (+R) – todo Stand assim o será necessariamente durante toda a sua existência; Fornece Identidade (+O) – cada Stand pode ser identificado globalmente, por meio de uma característica que lhe é própria; Executa Identidade (+I) – uma vez que fornece uma IC, obviamente também a executa; e Não Dependente (-D) – não depende de nenhuma outra classe. A partir desta análise, pode-se concluir que a classe *Stand* representa, segundo Welty e Guarino [22], um *Tipo*.
A classe *Atividade* armazena informações referentes a atividades (palestras, minicursos, mesa redonda etc.) contidas nos eventos. Ela foi classificada como: Rígida (+R) – toda Atividade assim o será necessariamente durante toda a sua existência; Fornece Identidade (+O) – cada Atividade pode ser identificada globalmente, por meio de uma característica que lhe é própria; Executa Identidade (+I) – uma vez que fornece uma IC, obviamente também a executa; e Não Dependente (-D) – não depende de nenhuma outra classe. A partir desta análise, pode-se concluir que a classe *Atividade* representa, segundo Welty e Guarino [22], um *Tipo*.

3.2.1 Adaptação às restrições impostas pelas Meta-propriedades. Constatou-se que as classes *Usuário, Participante, Funcionário, Patrocinador, Palestrante, Locatário* e *Serviço Terceirizado* correspondem a *Papeis Materiais* [22]. Portanto, a fim de respeitar o *Princípio da Expansibilidade Sortal* [13], exposta no item 3.1.4, as classes acima devem ser subclasses de classe rígida, para que possa herdar as ICs que executam. Dessa forma, propõe-se a inclusão de três novas classes, constituindo-se em uma hierarquia: uma classe abstrata *Pessoa*, que é rígida e não executa identidade, para dividir as pessoas do domínio em dois tipos – *Pessoa Física* e *Pessoa Jurídica*. Estas duas últimas são classes rígidas e que fornecem ICs, sendo que a primeira constitui-se superclasse das classes *Usuário* e *Palestrante* e a segunda superclasse das classes *Patrocinador, Locatário* e *Serviço Terceirizado*.
A classe *Funcionário* passa a ser subclasse da classe *Usuário* (assim como a classe *Participante*), para herdar a IC que executa, proveniente da classe *Pessoa Física*. Tal relacionamento hierárquico é válido, uma vez que funcionários que trabalham em eventos são cadastrados como usuários do sistema de gerenciamento de eventos.
A Figura 2 mostra o novo diagrama de classes, com elementos, que representam as classes, devidamente classificados de acordo com Welty e Guarino [22] e com as modificações acima descritas, obtidas a partir da aplicação da técnica VERONTO.

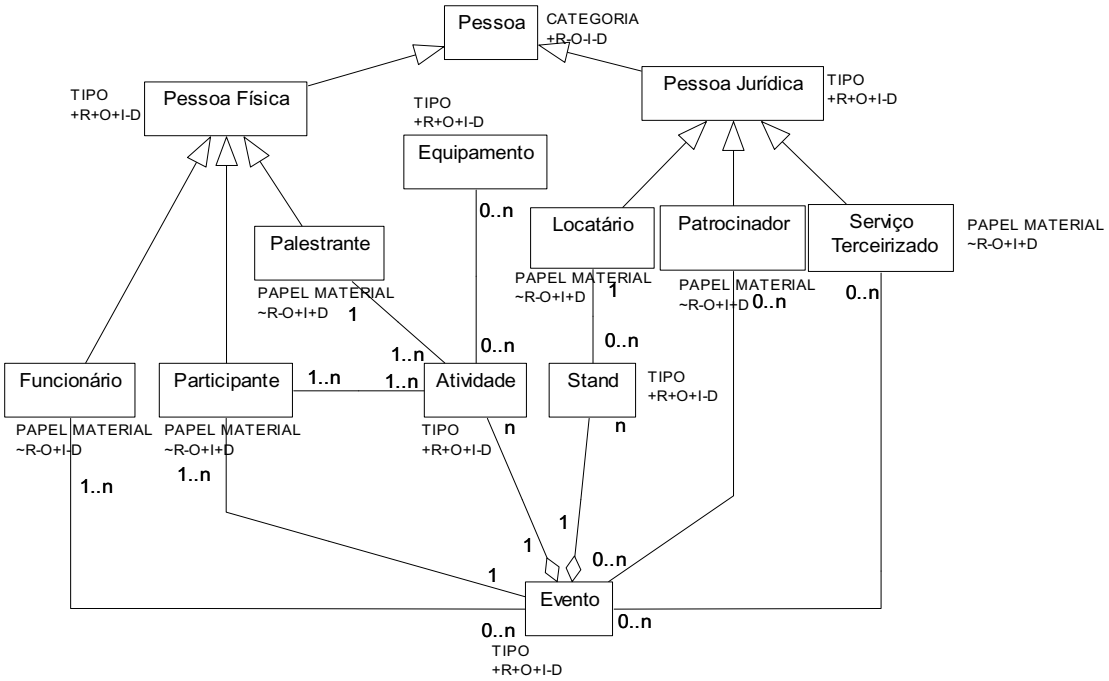

Figura 2. Diagrama de Classes Validado do Domínio de Gerenciamento de Eventos.

4. Conclusões

O presente trabalho apresentou uma revisão do uso da análise ontológica, discutindo mais detalhadamente a apresentada por Guarino e Welty. Foi introduzida também, uma forma de aplicar esta análise na modelagem conceitual orientada a objetos, por meio de um mapeamento dos tipos de propriedades, vistos na Tabela 1, em elementos do diagrama de classes da UML, permitindo, deste modo, a aplicação das restrições para relacionamentos hierárquicos, vistas na seção 2.2.3. A técnica proposta por este artigo, denominada VERONTO, proporciona a validação de um modelo conceitual especificado em UML, principalmente suas relações de hierarquia, por meio de uma análise dos elementos do domínio.

Foi apresentado um exemplo da aplicação da técnica em um diagrama de classes previamente modelado. A análise ontológica sugeriu a criação de novas classes que agrupariam as informações comuns a algumas classes já modeladas, funcionando como superclasses destas últimas. Deste modo, o modelo resultante representa melhor a realidade, diminuindo risco de inconsistências, por meio da eliminação de informações redundantes.

Espera-se que a VERONTO possa contribuir para a obtenção de informações precisas sobre o domínio a ser modelado, que por sua vez conduz a modelos conceituais mais claros e consistentes, levando a implementações de sistemas mais manuteníveis.

O próximo passo da pesquisa é usar a VERONTO em um modelo conceitual, desenvolvido profissionalmente, com o intuito de verificar se as alterações sugeridas pela técnica realmente conduzem a uma melhor modelagem.

Referências Bibliográficas

1. Gruber, T. R. 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5 (1993) 199-220.

2. Guarino, N.; Giaretta, P. 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In: N. Mars (ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing 1995. IOS Press, Amsterdam: 25-32.
3. Guarino, N. 1995. Formal Ontology, Conceptual Analysis and Knowledge Representation. Int. J. Hum.-Comput. Stud. 42, 6 (June 1995), 625-640.
4. Guarino, N. 1999. The Role of Identity Conditions in Ontology Design. In Proceedings of IJCAI-99 workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. Stockholm, Sweden, August 2, 1999.
5. Guarino, N.; Welty, C. 2000a. A formal ontology of properties. In R. Dieng, Ed., Proceedings of 12th Int. Conf. On Knowledge Engineering and Knowledge Management, Springer Verlag, 2000.
6. Guarino, N.; Welty, C. 2000b. Towards a methodology for ontology based model engineering. In Proceedings of the ECOOP-2000 Workshop on Model Engineering.
7. Guarino, N.; Welty, C. 2000c. Identity, Unity, and Individuality: Towards a Formal Toolkit for Ontological Analysis. In Proceedings of the ECAI-2000: The European Conference on Artificial Intelligence. IOS Press, Amsterdan. August, 2000.
8. Guarino, N.; Welty, C. 2000d. Ontological Analysis of Taxonomic Relationships. In Laender, A.; Storey, V. (eds.), Proceedings of ER-2000: The International Conference on Conceptual Modeling, vol. 1920, October, 2000. Springer-Verlag LNCS.
9. Guarino, N.; Welty, C. 2001. Identity and Subsumption. LADSEB-CNR Internal Report 01/2001.
10. Guarino, N.; Welty, C. 2002. Evaluating Ontological Decisions with ONTOCLEAN. Communications of the ACM. February, 2002, vol.5, n.2:61-65.
11. Guizzardi, G; Herre, H.; Wagner, G. 2002a. On the General Ontological Foundations of Conceptual Modeling. In Proceedings of 21th International Conference on Conceptual Modeling (ER 2002). Springer-Verlag, Berlin, Lecture Notes in Computer Science.
12. Guizzardi, G; Herre, H.; Wagner, G. 2002b. Towards Ontological Foundations for UML Conceptual Models. In Proceedings of 21th International Conference on Conceptual Modeling (ER 2002). Springer-Verlag, Berlin, Lecture Notes in Computer Science.
13. Lowe, E. J. 1989. Kinds of Being. A Study of Individuation, Identity and the Logic of Sortal Terms. Basil Blackwell, Oxford *apud* Guarino, N.; Welty, C. 2000a. A formal ontology of properties. In R. Dieng, Ed., Proceedings of 12th Int. Conf. On Knowledge Engineering and Knowledge Management, Springer Verlag, 2000.
14. Moreira, A. Tesauros e Ontologias: Estudo de Definições Presentes na Literatura das Áreas das Ciências da Computação e da Informação, Utilizando-se o Método Analítico-Sintético. Dissertação de Mestrado. Escola de Ciência da Informação, da Universidade Federal de Minas Gerais, 2003.
15. Poli, R. Framing ontology. 2001. Available from World Wide Web: http://www.formalontology.it/Framing first.htm.
16. Raban, R.; Garner, B. 2001. Ontological Engineering for Conceptual Modeling. In Proceedings of the ONTO-2001 Workshop on Ontologies. Viena, Austria, September 18, 2001.
17. Storey, V. C.; Dey, D.; Ullrich, H.; Sundaresan, S. 1998. An ontology-based expert system for database design. Data & Knowledge Engineering, 28(1): 31-46.
18. Strawson, P. F. 1959. Individuals. An Essay in Descriptive Metaphysics. Routledge, London and New York *apud* Guarino, N.; Welty, C. 2000a. A formal ontology of properties. In R. Dieng, Ed., Proceedings of 12th Int. Conf. On Knowledge Engineering and Knowledge Management, Springer Verlag, 2000.
19. Sugumaran, V.; Storey, V. C. 2002. Ontologies for conceptual modeling: their creation, use, and management. Data & Knowledge Engineering, 42: 251-271.

20. Toledo, A. B.; Gabrielli, B. V.; De Morais, R. F.; Amorim, V. G. Especificação dos Requisitos do Software Wind 1.0. Trabalho da Disciplina INF – Departamento de Informática, UFV, Viçosa, 2003.
21. Wand, Y; Storey, V. C.; Weber, R. 1999. An ontological analysis of the relationship construct in modeling conceptual. ACM Transactions on Database Systems, 24(4): 494-528, December 1999.
22. Welty, C.; Guarino, N. 2001. Suporting Ontological Analysis of Taxonomic Relationships. Data & Knowledge Engineering 39 (2001) 51-74.