BRUNA CAROLINA DE MELO CATOSSI

PROCEDIMENTO DE VALIDAÇÃO DE DIAGRAMA DE CLASSES DE DOMÍNIO BASEADO EM ANÁLISE ONTOLÓGICA PARA RELACIONAMENTOS DE AGREGAÇÃO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2010

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T

C366p
2010

Catossi, Bruna Carolina de Melo, 1985-
Procedimento de validação de diagrama de classes de domínio baseado em análise ontológica para relacionamentos de agregação / Bruna Carolina de Melo Catossi. – Viçosa, MG, 2010.
x, 101f. : il. ; 29cm.

Inclui apêndices.
Orientador: Alcione de Paiva Oliveira.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 98-101.

1. Engenharia de software. 2. Inteligência artificial. 3. Software - Desenvolvimento. 4. Projeto de sistemas. 5. Análise de sistemas. 6. Computação. I. Universidade Federal de Viçosa. II. Título.

CDD 22.ed. 005.1

Aos meus pais, João Carlos e Rita,
por serem tão especiais e sempre
presentes em todos os momentos
da minha vida.

“Meu reino não é deste mundo”.
(João, 18:33)

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por essa oportunidade por acreditar que Ele está sempre presente em tudo que nos cerca, proporcionando as energias renovadoras e o amparo necessário para dar-nos força e coragem para seguir em frente.

Agradeço à minha família, meus pais e minha irmã por estarem sempre dispostos a ajudar, esclarecer e apoiar toda vez que momentos difíceis surgem e também por poderem compartilhar comigo as alegrias das conquistas alcançadas.

Ao meu orientador Alcione de Paiva Oliveira, pela paciência, compreensão e disposição em ajudar sempre que preciso.

Aos professores Mauro Nacif Rocha e José Luís Braga, pelas conversas esclarecedoras sobre que rumo tomar.

A todos os outros professores do DPI que fizeram parte da minha formação acadêmica, pelos ensinamentos repassados que me permitiram estar concluindo mais esta etapa em minha vida e por enxergarem pessoas por trás de alunos.

A todos os meus colegas de pós-graduação, pelas horas de descontração, pelo compartilhamento de conhecimentos, dúvidas e aflições e pelas experiências trocadas durante o curso.

BIOGRAFIA

Nascida no dia 11 de junho de 1985, em Ribeirão Preto – SP, aos 3 anos mudou-se com os pais para uma cidade do interior de Minas Gerais, Viçosa, à qual se encontra atualmente.

Obteve o título de Bacharel em Ciência da Computação pela Universidade Federal de Viçosa (UFV) no ano de 2007. Em 2010 concluiu o mestrado também em Ciência da Computação pela UFV. No momento, ministra aulas nos cursos tecnológicos da UniViçosa na área de redes. Também possui interesse em engenharia de software, pesquisa operacional, banco de dados e sistemas de informação.

CONTEÚDO

LISTA DE TABELAS

LISTA DE FIGURAS

RESUMO

CATOSSI, Bruna Carolina de Melo, M.Sc., Universidade Federal de Viçosa, dezembro de 2010. **Procedimento de validação de diagrama de classes de domínio baseado em análise ontológica para relacionamentos de agregação**. Orientador: Alcione de Paiva Oliveira. Co-orientadores: José Luís Braga e Jugurta Lisboa Filho.

A dificuldade dos desenvolvedores de software para construir modelos conceituais fiéis à realidade é antiga. Existem algumas técnicas de análise ontológica para ajudar o modelador durante o processo de criação do diagrama de classes. No entanto, elas acabam não sendo práticas e não refletem os seus reais benefícios em suas aplicações, pois envolvem muitos conceitos filosóficos, o que as tornam complexas para modeladores comuns. Por esse motivo, procedimentos capazes de simplificar o entendimento desses conceitos e que se aproximam da realidade prática dos desenvolvedores tem surgido, como o PrOntoCon, que será discutido neste trabalho. O objetivo principal do PrOntoCon é guiar o modelador durante o processo de validação de um diagrama de classes UML para qualquer domínio, focando, especialmente, os relacionamentos de agregação/composição e de associação simples, visto que são os tipos de relacionamentos que geram mais dúvidas e controvérsias durante a modelagem. Assim, esse procedimento dá o suporte necessário para a correta identificação dessas relações, promovendo um estudo mais aprofundado sobre as restrições do domínio em questão. Portanto, o PrOntoCon combina o poder de modelagem da UML com a teoria da análise ontológica sobre relacionamentos parte-todo e de associação para criar um procedimento capaz de conceber modelos conceituais mais claros e confiáveis e que possam gerar sistemas mais robustos e manuteníveis.

ABSTRACT

CATOSSI, Bruna Carolina de Melo, M.Sc., Universidade Federal de Viçosa, December 2010. **Domain class diagram validation procedure based on ontological analysis for part-whole relations**. Advisor: Alcione de Paiva Oliveira. Co-advisors: José Luís Braga and Jugurta Lisboa Filho.

The difficulty software developers encounter to construct conceptual models that are closest to the real world is not new. There are some ontological analysis techniques that aim to help the modeler during the conception of the class diagram. However, those techniques are not practical and they do not reflect the real benefits of their applications, since they are involved with many philosophical aspects that are too complicated for common modelers to comprehend. For this reason, procedures that are capable of simplifying the understanding of those notions by bringing them closer to the reality of the developers have been emerged, like PrOntoCon that will be discussed in this research. The main goal of PrOntoCon is guide the modeler through the validation procedure of UML class diagrams of any domain, especially focusing on part-whole and association relations, due the fact they are the most controversial kinds of relationships along modeling. So, this procedure gives a necessary support to the correct identification of those relations, promoting a deeper study about the domain restrictions in question. Therefore, PrOntoCon combines the power of UML with ontological analysis theories of part-whole relations in order to give rise to a procedure that can create more clearly and faithful conceptual models that will improve the quality of systems generated in their totality.

1. Introdução

Todas as fases de desenvolvimento de um sistema de software são importantes. Porém, as etapas iniciais são as de maior importância, pois é nesse momento que a maior parte dos requisitos são extraídos para o correto entendimento do domínio e para a implementação adequada da solução do problema. Um entendimento equivocado de parte do problema certamente resultará em um produto final que não atingirá o real objetivo para o qual foi criado. E essa é a principal dificuldade dos desenvolvedores de software hoje em dia. A modelagem conceitual faz parte dessas iterações iniciais e tem por objetivo elaborar uma representação de qualidade do domínio desejado. Um modelo conceitual incorreto, ou que não represente adequadamente o domínio que está sendo modelado pode levar a problemas de projeto, implementação, operação e manutenção dos sistemas.

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 ambiguidades sobre os aspectos que devem ser modelados. Porém, nem sempre os profissionais responsáveis pela tarefa de modelagem possuem um conhecimento claro sobre a natureza dos objetos e conceitos pertencentes ao domínio, até porque o entendimento da natureza dos objetos do domínio ocorre no nível conceitual e depende da visão dos usuários do sistema. Esta falta de entendimento pode levar a modelos conceituais distorcidos sob o aspecto semântico, como duplicação de informação e hierarquias equivocadas.

A UML (*Unified Modeling Language*) (OMG, 2010) tornou-se uma linguagem de modelagem orientada a objetos padrão, sendo amplamente utilizada na construção de modelos conceituais em métodos de desenvolvimento de software orientados a objetos, visando estabelecer uma ligação explícita entre elementos conceituais e elementos executáveis. O diagrama de classes é um recurso da UML, utilizado para a tarefa de

modelagem conceitual, que consiste em uma representação gráfica de um conjunto de elementos de um domínio e que descreve a visão estática de um sistema, em termos de classes e relacionamentos entre estas, sendo sempre válida em qualquer ponto do desenvolvimento do sistema (FOWLER, 2005).

Uma das formas de auxiliar o processo de modelagem conceitual é fazer uso de uma análise mais detalhada das propriedades dos objetos pertencentes a um domínio. Esse processo chama-se análise ontológica (GUARINO e WELTY, 2000b). Modeladores conceituais podem fazer uso desse recurso e assim minimizar erros de modelagem. A ontologia é um ramo da filosofia que estuda a organização e a natureza do mundo. Na computação o termo ontologia tem sido utilizado para denotar o registro das descrições dos conceitos e suas relações abstraídas de um domínio (GUIZZARDI, 2005).

Guarino e Welty (2000a, 2000b) definiram quatro meta-propriedades ontológicas: rigidez, identidade, unidade e dependência. Essas meta-propriedades impõem uma série de restrições sobre os relacionamentos que podem existir entre os conceitos do domínio. O uso correto dessas meta-propriedades e restrições ontológicas na classificação e hierarquização dos conceitos de um domínio torna possível validar modelos conceituais expressos por diagramas de classes.

Através de um agrupamento das meta-propriedades, Guarino e Welty (2000a, 2000b) propuseram oito tipos de propriedades. Villela (2004) apresentou um mapeamento desses tipos de propriedades sobre os elementos do diagrama de classes da UML. Esse mapeamento, denominado Técnica de Verificação Ontológica (VERONTO), pode ser utilizado como uma etapa adicional no processo de desenvolvimento de software de forma a verificar o modelo conceitual a partir de uma análise ontológica dos elementos do domínio. A aplicação da técnica VERONTO se mostrou satisfatória, uma vez que modelos validados tiveram alterações significativas para o contexto e ampliou-se a semântica embutida nos modelos conceituais dos sistemas de informação analisados.

Guizzardi et al. (2004), apresentam um Perfil OntoUML para modelagem conceitual e representações ontológicas. Este trabalho foi

altamente influenciado pelas meta-propriedades definidas por Guarino e Welty (2000a, 2000b). Nesse mesmo trabalho, Guizzardi et al. (2004) propõem também um Padrão de Projeto, baseado no Perfil OntoUML, para resolver problemas relacionados à modelagem de “papéis”, que é um problema recorrente na literatura.

A partir de um estudo comparativo feito entre a VERONTO e o Perfil OntoUML foi possível perceber que continham muitas características semelhantes e complementares, já que ambas são apoiadas nas meta-propriedades de Guarino e Welty (2000 a, b), o que serviu de base para a estruturação da técnica chamada OntoCon. A necessidade de criação de uma nova técnica surgiu para facilitar o processo de validação de diagramas de classes UML tendo como objetivo principal as relações de generalização/especialização, visto que uma técnica única poderia eliminar as redundâncias do uso das duas em separado, acrescentar melhorias e, ainda, facilitar a aplicação prática da técnica com a criação de um procedimento para o uso da mesma, chamado PrOntoCon. Esse procedimento guia o modelador através de uma sequência de etapas com o intuito de validar os diagramas de classes UML existentes, verificando seus relacionamentos e papéis e detectando suas possíveis falhas.

1.1. O problema e sua importância

Um modelo pode ser definido como a descrição de algo, utilizando-se normalmente uma linguagem visual ou matemática, o que significa que a maior parte da informação contida nos modelos é expressa através de símbolos e conexões gráficas (ELMASRI e NAVATHE, 2002). A utilização de modelos é útil em qualquer engenharia, pois a construção de um produto ocorre normalmente baseada em modelos previamente construídos, que descrevem a aparência e o comportamento do mesmo. Para a engenharia de software não seria diferente.

Eriksson e Penker, citados em Villela (2004), afirmam que a obtenção de modelos conceituais é uma tarefa altamente criativa, pois não existe uma solução final ou resposta correta que possa ser checada ao final do trabalho. A construção de um modelo se faz por meio de um trabalho interativo, com

seus projetistas alcançando um conhecimento mais aprofundado do domínio do problema e buscando assegurar que o modelo em questão atinja os objetivos e requisitos do sistema e de seus usuários.

Bons modelos devem capturar a essência do domínio do problema, além de serem consistentes, íntegros e de fácil entendimento e manutenção. Modelos também devem ser capazes de serem integrados, ou seja, deve ser possível unir, sem introduzir inconsistências, um conjunto de modelos que apresentam os mesmos propósitos e representem informações sobre o mesmo domínio (PAULA FILHO, 2009).

Dessa forma, é possível perceber que processos de modelagem não são tão simples quanto parecem. Ainda existe uma deficiência de suporte metodológico para auxiliar os usuários de linguagens de modelagem na decisão de como modelar os elementos de um dado domínio. Na maioria das vezes, quando um modelo de diagrama de classes UML é construído, seus elementos são extraídos e relacionados de forma *ad hoc*, sem o total entendimento do domínio do problema.

Apesar de já existirem algumas técnicas de auxílio na análise e construção de diagramas de classe, como a VERONTO (VILLELA, 2004), ONTOCON (TAVARES et al., 2008) e o Perfil OntoUML (GUIZZARDI *et al*. 2004), o que se percebe é que o uso dessas técnicas não é algo trivial. Conseguir extrair os elementos de um domínio e analisá-los conforme suas características ontológicas não é uma tarefa fácil, pois os conceitos e formalizações ontológicas existentes nessas técnicas exigem um grande estudo de conceitos filosóficos que envolvem um grande formalismo. Por esse motivo, certamente a aplicação das técnicas apresentadas têm como grande barreira a dificuldade de uso. Para tentar solucionar esse problema é necessária a criação de um procedimento de uso das técnicas que, com base nas prescrições de modelagem ontológica, guie implicitamente o modelador: (a) na análise dos elementos do domínio que irão representar as classes do diagrama de classes e (b) na correta identificação dos devidos relacionamentos entre essas classes.

Embora o desenvolvimento deste procedimento tenha tido início na pesquisa de Tavares et al. (2008, 2009), com a criação do procedimento PrOntoCon, o foco do trabalho foi o relacionamento de

generalização/especialização, deixando de fora alguns aspectos que fazem parte da modelagem, como os relacionamentos parte-todo e de associações simples. Portanto, o presente trabalho tem por objetivo estender o PrOntoCon de modo que o procedimento guie o desenvolvedor na análise de relacionamentos de agregação/composição e de associação para que estes reflitam adequadamente os conceitos do domínio que está sendo validado e que são expressos em diagramas de classes UML.

1.2. Hipótese

A hipótese de trabalho desta pesquisa é que a análise ontológica dos relacionamentos parte-todo e de associações simples de modelos conceituais expressos em diagramas de classes geram modelos que são mais fáceis de serem mantidos e mais escaláveis.

1.3. Objetivos do Trabalho

O objetivo geral do projeto é a extensão de um procedimento de análise de conceitos (PrOntoCon) que permita guiar o modelador na validação das classes e dos relacionamentos necessários a um modelo de classes do domínio de uma aplicação, através do uso de técnicas de modelagem ontológica. Além de aplicar o procedimento para analisar diagramas de classes construídos sem o uso de modelagem ontológica, ou seja, o modelador terá a flexibilidade para escolher entre utilizar o procedimento desde o início da modelagem do diagrama de classes de domínio ou aplicá-lo, posteriormente, a um modelo que já havia sido desenvolvido.

Especificamente, pretende-se adicionar à Técnica OntoCon – Técnica de Validação de Diagrama de Classes de Domínio Baseado em Modelagem Ontológica – as restrições sobre relacionamentos “parte-todo” (agregação e composição) e relacionamentos de associação em geral e estender o procedimento PrOntoCon – Procedimento de Análise para Validação de Diagrama de Classes de Domínio Baseado em Modelagem Ontológica – utilizando-se o SPEM (*Software and Systems Process Engineering Meta-Model*), de forma a definir formas que permitam uma fácil aplicação das

regras e restrições de relacionamentos “parte-todo” e de associação impostas pela técnica OntoCon.

Dessa forma, espera-se como contribuição do presente projeto de pesquisa a elaboração de um procedimento de validação de diagramas de classes UML fácil de usar e que seja capaz de auxiliar no desenvolvimento de softwares mais robustos e manuteníveis.

2. Referencial Teórico

2.1. Modelagem Conceitual Utilizando Diagramas de Classes UML

Na fase de análise, os requisitos até então levantados passam por detalhamentos, estruturações e validações que permitem a construção de um modelo conceitual do sistema. Essa modelagem conceitual, realizada ainda na fase de análise, é necessária para que os conceitos relevantes do domínio do problema sejam devidamente modelados e assim usados como base para as fases seguintes do processo de desenvolvimento.

O modelo conceitual é um artefato do domínio do problema e não da solução do problema. Um modelo conceitual representa as informações que o sistema vai gerenciar. Qualquer modelagem equivocada irá provocar erros durante o processo de desenvolvimento do sistema, gerando um produto inadequado para o domínio no qual será utilizado.

Diagramas são utilizados para especificar como o analista quer representar o produto a ser desenvolvido. Para representar os conceitos que são extraídos na modelagem conceitual utiliza-se o diagrama de classes da UML. Segundo Elmasri e Navathe (2002), o diagrama de classes é uma representação gráfica dos conceitos abstratos de um domínio e exibe uma visão estática de um sistema mostrando suas classes e os relacionamentos existentes entre elas.

O diagrama de classes que irá representar o modelo conceitual do domínio do problema, denominado de diagrama de classes de domínio, deve conter somente os detalhes necessários para servir de base para o desenho do sistema. Portanto, deve-se evitar a inclusão de detalhes que estejam relacionados ao domínio da implementação (PAULA FILHO, 2009).

2.2. Tipos de Relacionamentos

2.2.1. Associação Simples

Em UML, uma associação é definida como um relacionamento que descreve um conjunto de ligações, onde cada ligação é definida como uma conexão semântica entre os objetos, que são instâncias das classes participantes da associação (MEDEIROS, 2004).

A associação no diagrama de classes é representada por uma linha que conecta duas ou mais classes e que pode possuir um nome perto da linha que a representa. Todos os nomes descrevendo as classes e seus relacionamentos devem estar de acordo com o domínio do problema.

Normalmente também são definidos a *multiplicidade* e os *papéis* das classes participantes do relacionamento de associação. A multiplicidade de participantes indica quantos objetos de uma classe podem estar relacionados com cada objeto da outra classe. Os papéis são denominações que exprimem em que qualidade um objeto de uma das classes do relacionamento se relaciona com um objeto da outra classe.

A Figura 1 mostra uma associação que indica que um objeto da classe *Empresa* participa do relacionamento *Fornece,* no papel de *Fornecedor,* uma quantidade de *Produto*. A multiplicidade no relacionamento indica que cada empresa pode fornecer zero ou mais produtos e que cada produto deve ser fornecido por, no mínimo, uma ou mais empresas.

Figura 1: Associação *Fornece* com suas multiplicidades (0..n para *Empresa* e 1..n para *Produto*) e o papel da classe *Empresa* (*+Fornecedor*) definidos. Extraído de Villela (2004).

2.2.2. Agregação

Um relacionamento de agregação é um tipo especial de associação que indica que o relacionamento entre as classes é do tipo “parte-todo”, refletindo construção física ou posse lógica. Uma agregação normalmente descreve diferentes níveis de abstração e utiliza as palavras-chave “consiste

de", "contém", "é parte de" na descrição do relacionamento entre as classes envolvidas (MEDEIROS, 2004).

No diagrama de classes da UML, um relacionamento de agregação é representado por uma linha conectando as classes relacionadas, como em qualquer associação, porém adicionada de um losango (ou diamante) ligado ao final da linha, do lado da classe que representa o "todo" no relacionamento.

Existem dois tipos de relacionamentos de agregação, dependendo do compartilhamento ou não, pela classe que representa o "todo", de alguma de suas partes com outro "todo".

2.2.3. Agregação Compartilhada

Uma agregação compartilhada é aquela em que as partes podem estar ligadas a mais de um "todo". Isto é demonstrado pela multiplicidade maior que um (1) do lado "todo". A Figura 2 mostra um exemplo de agregação compartilhada, onde um funcionário, que representa a classe "parte", pode ser membro de várias equipes.

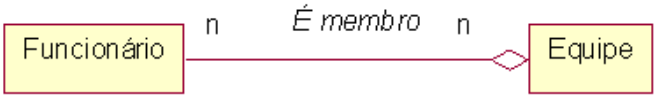


Figura 2: Exemplo de agregação compartilhada, envolvendo a classe "parte" *Funcionário* e a classe "todo" *Equipe*. Extraído de Villela (2004).

Outra característica deste tipo de relacionamento é a independência da existência dos objetos que instanciam tais classes, ou seja, as classes podem "viver" de forma independente, formando uma dependência genérica entre elas, em que o objeto "parte" pode existir sem o objeto que representa o "todo" (MEDEIROS, 2004).

2.2.4. Composição

O relacionamento de composição consiste em um tipo mais forte de relacionamento "todo-parte". Neste caso, um objeto da classe "parte" não possui existência própria, ou seja, se o "todo" for destruído, suas partes serão destruídas juntamente. A multiplicidade no lado "todo" do

relacionamento deve ser um (1), mas a multiplicidade no lado “parte” pode ser qualquer intervalo.

Uma vez que o relacionamento de composição constitui-se num tipo de agregação, no diagrama de classes da UML uma composição é representada da mesma forma que uma agregação comum, porém o losango (ou diamante) ligado ao lado “todo” do relacionamento é preenchido.

A Figura 3 mostra um exemplo de composição entre a classe “todo” *Caixa de Mensagem* e as classes “partes” *Botão Ok, Botão Cancelar* e *Informação.* Note que, para que objetos das classes *Informação, Botão Ok* e *Botão Cancelar* existam, eles devem fazer parte de um objeto da classe *Caixa de Mensagem.*

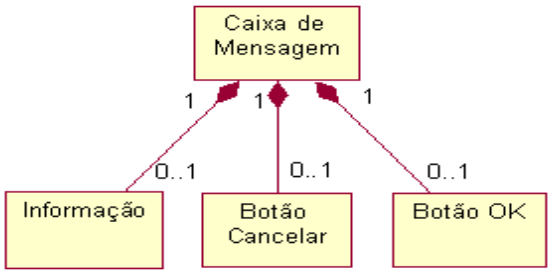


Figura 3: Exemplo de composição, envolvendo as classes “partes” *Informação, Botão Cancelar* e *Botão OK* e a classe “todo” *Caixa de Mensagem*. Extraído de Villela (2004).

2.2.5. UML 2.0 e as Estruturas Compostas

Segundo Fowler (2005), a UML 2.0 representa a maior mudança já ocorrida na UML, visto que modificações profundas no metamodelo aconteceram. As alterações mais visíveis foram a introdução de novos diagramas. Os diagramas de objetos e os diagramas de pacotes foram oficializados nessa nova versão. A UML 2.0 mudou o nome dos diagramas de colaboração para diagramas de comunicação, além de adicionar novos tipos de diagramas: diagramas de visão geral da interação, diagramas de temporização e diagramas de estruturas compostas.

Para melhor compreensão do uso de todos os possíveis diagramas da UML 2.0, eles são divididos em diagramas de estrutura e diagramas de comportamento. Os diagramas de estrutura são formados por diagrama de

pacotes, diagrama de objeto, diagrama de classes, diagrama de desenvolvimento, diagrama de componente e diagrama de estrutura composta. Os diagramas de comportamento são formados por diagrama de atividades, diagrama de interações, diagrama de casos de uso e diagrama de estados. Além disso, o diagrama de interação possui ramificações que são os diagramas de tempo, diagramas de comunicação, diagramas de colaboração e diagramas de seqüência.

Fowler (2005) afirma que um dos novos recursos mais significativos da UML 2.0 é a capacidade de decompor hierarquicamente uma classe em uma estrutura interna, pois isso permite que objetos complexos possam ser divididos em partes. Bock (2004) ainda complementa que as estruturas compostas incorporadas nessa nova versão, a UML 2.0, é um grande avanço sobre a representação anterior para composições (o losango, ou diamante, preenchido). Elas permitem que conexões entre partes que se apresentam no mesmo nível de decomposição sejam feitas, além das associações parte-todo comuns. Outra característica das estruturas compostas é que elas são capazes de promover uma modelagem estrutural mais detalhada, incluindo a troca de mensagens, o que facilita a sua aplicação dentro dos diagramas de componentes, através do conceito de portas. No entanto, esta última característica foge ao escopo do presente trabalho, sendo considerado somente o uso de estruturas compostas para a modelagem conceitual dentro dos diagramas de classes.

A Figura 4 mostra um exemplo simples modelado nas versões anteriores à UML 2.0, onde um carro possui motor, rodas dianteiras e rodas traseiras; e um barco possui motor e propulsor(es). Também seria interessante para o domínio que se especificasse que o motor do carro aciona as rodas dianteiras dele, assim como o motor do barco aciona o propulsor. Porém, com o uso na notação antiga para composição, a solução para se eliminar a ambiguidade de que o motor de um barco pode acionar as rodas dianteiras do carro e vice-versa torna-se muito complexa, como discutido em Bock (2004). Por essa razão, o uso de estruturas compostas conseguem solucionar a questão de forma muito mais simples sendo apresentada na Figura 5.

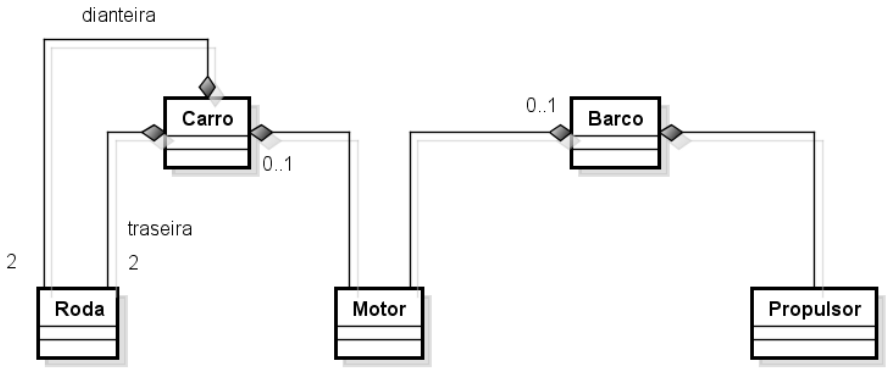


Figura 4: Composição feita utilizando-se UML 1.x (extraída e adaptada de BOCK, 2004).

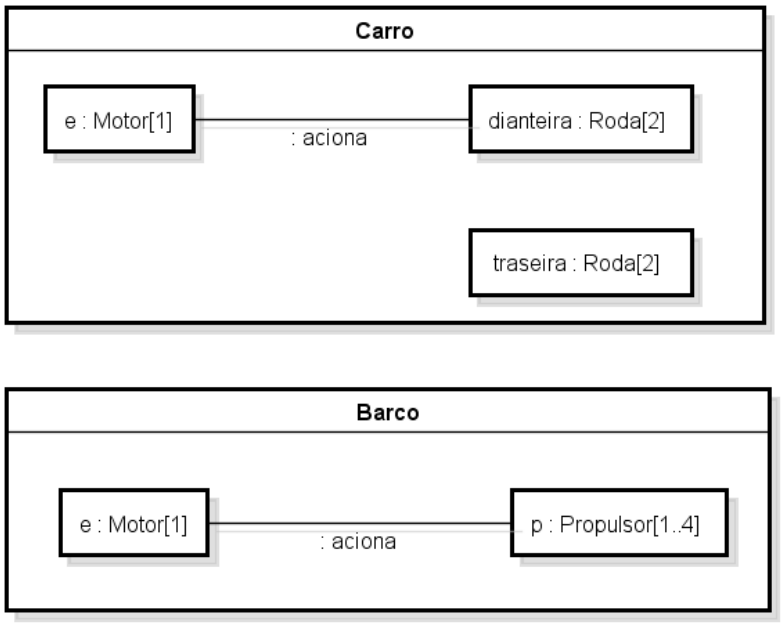


Figura 5: Composição feita utilizando-se UML 2.0 (extraída e adaptada de BOCK, 2004).

2.3. Análise Ontológica

Ao desenvolver uma modelagem conceitual de um determinado fenômeno de um domínio obtém-se os objetos relevantes para a compreensão não só isoladas desses objetos, como principalmente o entendimento dos relacionamentos entre os mesmos. Uma análise mais detalhada das

propriedades desses objetos é chamada de análise ontológica. Segundo Guizzardi (2005), modelagem ontológica pode ser definida como sendo a criação de uma ontologia específica de domínio. Esta ontologia é construída através da captura de entidades importantes do domínio e da incorporação dessas entidades em um conjunto de categorias que revelam sua natureza.

Ontologia tem sua definição original na filosofia como sendo a ciência que estuda “os tipos de coisas que existem no mundo”. Guarino, citado em Tavares (2008), diz que uma ontologia define um vocabulário específico de representação para capturar conceitos, relações e propriedades em algum domínio da realidade e um conjunto de axiomas restringindo o significado pretendido para esse vocabulário, ou seja, sua interpretação. No entanto, o termo ontologia tem sido utilizado com diferentes sentidos na área de Inteligência Artificial e Engenharia do Conhecimento, o que tem provocado certa descaracterização do mesmo (GUIZZARDI, 2005).

É importante destacar duas caracterizações dadas ao termo ontologia: ontologia de domínio e ontologia de nível topo. A primeira é um vocabulário de representação para um domínio ou assunto. Esse tipo de ontologia é utilizado como base de conhecimento do domínio dando suporte à tarefa de modelagem conceitual para sistemas do referido domínio. Uma classificação superior seria a ontologia de nível topo que abrange termos gerais que formam o fundamento para a representação de conhecimento em **todos** os domínios (VILLELA, 2004).

Ontologias de nível topo podem ser utilizadas com objetivo de classificar os conceitos de uma ontologia de domínio permitindo que seus elementos sejam mais bem entendidos, bem como os relacionamentos entre os mesmos (VILLELA, 2004). Pode-se perceber então que é possível utilizar ontologia de nível topo para auxiliar no processo de modelagem conceitual. Este trabalho utiliza o termo ontologia referindo-se à ontologia de nível topo. Ontologias de domínio, à medida que se tornam mais difundidas, poderão ser incorporadas ao processo, mas isto deverá ser avaliado em pesquisas futuras.

2.3.1. Ontologia Formal de Propriedades

Guarino e Welty (2000a, b, c) especificaram uma ontologia formal baseada em quatro meta-propriedades básicas: identidade, rigidez, dependência e unidade. A combinação dessas meta-propriedades formam restrições naturais aplicadas aos conceitos da ontologia específica sendo utilizada, facilitando a compreensão, comparação e agrupamento desses conceitos.

De forma resumida, essas meta-propriedades podem ser definidas da seguinte maneira:

- Se é dito que uma classe possui identidade (+I), então isso significa que todas as instâncias dessa classe possuem uma característica comum e de valor único para cada uma delas (e.g., a classe Pessoa possui identidade, pois a impressão digital é uma característica comum à todas as instâncias de pessoa e é única para cada uma delas).
- Quando uma propriedade for analisada sob o ponto de vista da rigidez, ela poderá ser classificada em rígida, não-rígida e anti-rígida. Assim, uma propriedade rígida (+R) é uma propriedade que é essencial para todas as suas instâncias, isto é, uma instância de uma classe continuará instanciando-a durante toda a sua existência (e.g., classe Pessoa). Uma propriedade não-rígida (-R) é uma propriedade que não é essencial para alguma de suas instâncias, é apenas a negação lógica da rigidez (e.g., classe Estudante). E uma propriedade é dita anti-rígida (~R) se não for essencial para todas as suas instâncias (e.g., classe Professor).
- Se é dito que uma classe possui dependência externa (+D), então isso significa que para todas as suas instâncias deve haver uma relação com outra classe que não seja parte constituinte dela (e.g., classe Cliente é externamente dependente da classe Loja).
- A propriedade unidade (+U) está relacionada ao tratamento de problemas de distinção das partes de uma instância de uma classe do resto do mundo, através de uma relação de unificação que une suas partes.

Realizando um estudo minucioso e sistemático sobre as possíveis combinações dessas quatro meta-propriedades para formar diferentes tipos de propriedades, surgiu o que Guarino e Welty (2000a) denominaram de Ontologia Formal de Propriedades. Um detalhamento sobre as discussões em torno dessas meta-propriedades pode ser encontrado em Guarino e Welty (2000 a, b, c).

Segundo Tavares (2008), a taxonomia dessas propriedades definidas auxiliam a tarefa do modelador em estabelecer o significado de cada elemento do domínio, visto que as propriedades são baseadas nas meta-propriedades e estas possuem restrições hierárquicas em relação a esses elementos, o que pode ser utilizado para guiar o processo de criação e validação de modelos conceituais.

2.3.2. Mereologia e Meronímia

Historicamente, a formalização de uma teoria sobre partes constituindo um todo teve início antes mesmo dos filósofos Platão e Aristóteles. Por ser um tema complexo e carregado de muitas controvérsias, este ainda é um assunto bastante discutido no meio científico e filosófico. O conceito de estruturas parte-todo vem fazendo parte de pesquisas tanto sob o aspecto da ontologia formal quanto cognitivo e linguístico (WAND et al., 1999; GUIZZARDI, 2005 e 2009; KEET, 2006; VARZI, 2009). Mereologia e meronímia são os nomes dados para esses dois ramos do estudo sobre as relações parte-todo, respectivamente. Mereologia é a investigação da ontologia formal sobre os relacionamentos parte-todo, ou seja, as relações da parte com o todo e as relações da parte com outra parte dentro de um todo (VARZI, 2009).

A mereologia, por ser uma teoria formal, tenta especificar os princípios gerais que regem o comportamento das estruturas parte-todo, bem como desvendar a natureza desses relacionamentos. Portanto, a mereologia assume que tanto as partes como o todo devem ser concretos ao mesmo tempo durante suas existências, como também assume que eles podem ser abstratos ao mesmo tempo durante suas existências.

Os princípios comuns que regem qualquer base sobre a teoria das relações parte-todo fazem parte da chamada *Ground Mereology*, ou

Mereologia Básica, composta de três restrições principais: reflexão, antisimetria e transitividade. Em linguagem natural, elas podem ser definidas da seguinte maneira (VARZI, 2009):

1. Tudo é parte de si mesmo.
2. Duas coisas distintas não podem ser parte uma da outra.
3. Qualquer parte de qualquer parte de uma coisa é por si só parte dessa coisa.

Formalmente, as restrições podem ser expressas como a seguir, respectivamente, utilizando-se a lógica de primeira ordem onde o predicado binário P indica uma relação parte-todo (VARZI, 2009):

1. Pxx
2. (Pxy & Pyx) → x = y
3. (Pxy & Pyz) → Pxz

O desenvolvimento de teorias formais promovem um maior entendimento sobre a noção de parte, além de poder formalizar esses conceitos por meio de axiomas, auxiliando a divulgação das ideias. No entanto, ainda há muita controvérsia sobre algumas propriedades envolvidas nas relações parte-todo, visto que elas acabam sendo incoerentes com as teorias conceituais e cognitivas aplicadas no mundo real (GUIZZARDI, 2009). A transitividade dos relacionamentos parte-todo, em particular, tem provocado muita discussão sobre a sua imposição para se caracterizar esses tipos de estruturas (e.g. uma parte de uma célula humana não poderia ser considerada parte do órgão). Comumente, essas indagações acontecem devido às ambiguidades sobre o conceito de partes, além dos casos em que objetos de natureza completamente diferentes (e.g. mão e porta, gato e casa) poderiam ser considerados como relacionamentos parte-todo. Porém, a mereologia se divide em várias ramificações para lidar com os diversos tipos de situações que as relações parte-todo trazem à tona (VARZI, 2009).

Assim, os três princípios básicos da *Ground Mereology* (reflexão, antisimetria e transitividade) podem ser combinados formando outros axiomas sobre os relacionamentos parte-todo, como descrito em Varzi

(2009), que, por sua vez, podem ser combinados e adicionados à *Ground Mereology* formando, portanto, suas ramificações de pesquisa.

O problema da transitividade nas teorias mereológicas é tratado em detalhes por Guizzardi (2005 e 2009), visto que já são encontrados vários contra-exemplos e discussões na literatura sobre os casos onde a transitividade não é alcançada através da cognição ou linguística. Para ilustrar essa situação, os seguintes exemplos são dados: a mão é parte da pessoa que por sua vez é parte de um grupo de pesquisa, mas a mão não é parte do grupo de pesquisa; Rio de Janeiro é parte do Brasil que por sua vez é parte das Nações Unidas, porém, nesse caso, Rio de Janeiro não é parte das Nações Unidas (GUIZZARDI, 2005 e 2009).

Além disso, Guizzardi (2009) afirma que relações parte-todo entre conjuntos funcionais não são transitivos e nem intransitivos, e sim não-transitivos, ou seja, transitivos em certas ocasiões e intransitivos em outras. Dessa forma, ele propõe uma teoria formal e uma tipologia das relações parte-todo entre conjuntos funcionais baseadas em uma análise linguística e cognitiva (meronímica) para a questão da transitividade das relações parte-todo incluída pela ontologia formal (mereologia). Como resultado dessa pesquisa, Guizzardi (2005) obteve quatro tipos ontológicos distintos para relacionamentos parte-todo, sendo eles: (i) *subquantity-quantify* (e.g., álcool-vinho) - partes são modeladas como uma quantidade de matéria que são unidas por uma relação topológica; (ii) *member-collective* (e.g., árvore-floresta) - a entidade representando o todo é modelada com partes que desempenham o mesmo papel com relação ao todo; (iii) *subcollective-collective* (e.g., porção sul da floresta-floresta) - segue o mesmo raciocínio do item anterior; (iv) *component-functional complex* (e.g., motor-carro, coração-sistema circulatório) - a entidade representando o todo é modelada de modo que todas as partes desempenham um papel diferente com relação ao todo. Alguns termos foram deixados em inglês para evitar a perda do significado original ocasionada por uma tradução equivocada.

Seguindo a mesma linha de raciocínio, Keet (2006) desenvolve sua pesquisa com base em estudos mereológicos e meronímicos, sendo estes últimos fundamentados no trabalho de Guizzardi (2005). Assim, levando em consideração os aspectos mais relevantes dessas abordagens para a

modelagem conceitual de relacionamentos parte-todo, Keet (2006) obteve como resultado de seu trabalho a formulação de uma taxonomia para relações parte-todo, além do desenvolvimento de um procedimento de decisão com o objetivo de facilitar a tarefa de construção de modelos conceituais aplicável a qualquer domínio. Os conceitos envolvidos sobre os tipos constituintes dessa taxonomia são mais detalhados em Artale e Keet (2008). Maiores esclarecimentos sobre essas teorias também serão apresentados ao longo do texto, visto que os trabalhos de Keet (2006), Artale e Keet (2008), Artale et al. (2008) e Guizzardi (2005) compõem a base teórica principal para o desenvolvimento do presente trabalho.

2.4. Uso da Análise Ontológica na Modelagem Conceitual de Sistema

Enquanto a modelagem conceitual concentra-se em relacionar de forma adequada os conceitos extraídos do domínio, a modelagem ontológica objetiva identificar os objetos do domínio e entender sua natureza por meio da descrição de suas propriedades (VILLELA, 2004). Analisando o objetivo dessas duas modelagens pode-se entender que elas podem ser usadas de forma complementar. A base para a modelagem conceitual pode ser extraída de uma modelagem ontológica.

A ontologia formal de propriedades definida por Guarino e Welty (2000a) auxilia no processo de entendimento de cada conceito do domínio bem como a estruturação hierárquica destes. Uma vez que os conceitos a serem modelados são mais bem entendidos e existem regras que podem ser checadas para validar relacionamentos, tem-se uma ferramenta de grande valia na construção e validação de modelos conceituais.

Alguns pesquisadores têm se preocupado em desenvolver técnicas capazes de auxiliar os modeladores a utilizarem a modelagem ontológica durante a modelagem conceitual. A seguir, será apresentada a técnica OntoCon (TAVARES, 2008), que é uma proposta de combinação das técnicas VERONTO (VILLELA, 2004) e do Perfil OntoUML para modelos conceituais UML (GUIZZARDI, 2004).

2.4.1. OntoCon

A OntoCon surgiu da combinação de características da VERONTO (VILLELA, 2004) e do Perfil OntoUML (GUIZZARDI, 2005), também sendo fundamentada nas meta-propriedades rigidez, identidade, unidade e dependência externa de Guarino e Welty (2000 a, b). A partir de um estudo comparativo feito entre a técnica VERONTO e o Perfil OntoUML foi possível perceber que continham muitas características semelhantes e complementares. A necessidade de criação de uma nova técnica surgiu para facilitar o processo de validação de diagramas de classes UML, visto que uma técnica única poderia eliminar as redundâncias do uso das duas em separado, acrescentar melhorias e, ainda, facilitar a criação de um procedimento para o uso da mesma, ou seja, a aplicação prática da técnica. A Tabela 1 a seguir resume os principais problemas encontrados nas técnicas VERONTO e Perfil OntoUML e as respectivas melhorias agregadas pela OntoCon.

A Tabela 2 mostra a combinação das meta-propriedades e seus respectivos estereótipos definidos pela OntoCon, bem como as restrições hierárquicas impostas a eles. Já a Tabela 3 apresenta as classificações das propriedades correspondentes aos estereótipos, juntamente com os possíveis subtipos e supertipos de cada uma e o construtor UML permitido. Para um estudo mais aprofundado sobre as meta-propriedades, sugere-se a leitura de Guarino e Welty (2000b).

A OntoCon ainda possui outras notações, além da +R (-R e ~R) mencionada, como: a notação +I (-I) indica que um elemento possui (ou não) uma condição de identidade; a notação +O (-O) determina se um elemento provê (ou não) uma condição de identidade; e a notação +D (-D) é usada para indicar se um elemento tem (ou não) uma dependência externa.

Através da aplicação da técnica OntoCon três pontos chaves podem ser validados em um diagrama de classes: (i) relacionamentos de generalização/especialização; (ii) relacionamentos de classes envolvidas na modelagem de papéis; e (iii) definição adequada de construtores UML (classe concreta, classe abstrata e interface) (TAVARES, 2008).

Tabela 1: Problemas detectados nas técnicas VERONTO e Perfil OntoUML e as melhorias acrescentadas pela OntoCon (extraída e adaptada de TAVARES, 2008).

Problemas com as técnicas abordadas	**Melhorias agregadas com a OntoCon**
Na VERONTO e no Perfil OntoUML, o mapeamento das propriedades categoria (<<category>>) e papel formal (<<rolemixin>>) é muito restritivo uma vez que se refere somente à classe abstrata.	Na OntoCon, os estereótipos relativos às propriedades categoria e papel formal podem ser mapeados tanto para interface quanto para classe abstrata.
Na VERONTO, o mapeamento das propriedades *tipo, quase-tipo, papel material* e *sortal de fase* torna-se muito restritivo uma vez que tais propriedades são mapeadas somente para classe concreta. O Perfil OntoUML não destaca nenhum mapeamento para os estereótipos <<kind>>, <<subkind>>, <<role>> e <<phase>>.	Na OntoCon, o mapeamento dos estereótipos relativos a essas mesmas propriedades é mais abrangente: um possível mapeamento tanto para interface e classe abstrata quanto para classe concreta é permitido.
As restrições de relacionamento entre classes e subclasses no que diz respeito à meta-propriedade dependência externa não são ressaltadas de forma clara na VERONTO.	Na OntoCon, todas as relações de supertipos e subtipos são tratadas respeitando a restrição de relacionamento da meta-propriedade dependência externa.
O Perfil OntoUML não trata nenhum dos subtipos possíveis de cada estereótipo e a VERONTO não trata de forma explícita todos os subtipos para cada tipo de propriedade.	A OntoCon trata todos os subtipos possíveis a cada estereótipo, o que proporciona uma validação mais fácil e completa de um diagrama de classe.
Tanto a VERONTO quanto o Perfil OntoUML não citam de forma explícita todos os possíveis supertipos para cada uma das propriedades e ou estereótipos.	A OntoCon cita explicitamente todos os possíveis supertipos para cada um dos estereótipos existentes, proporcionando uma maior segurança no uso da técnica.

Tabela 2: Estereótipos e restrições hierárquicas da OntoCon. Extraída e adaptada de Tavares (2009).

Estereótipos	Meta-propriedades	Estereótipos permitidos como Supertipos
<<type>>	+O +I +R -D	<<category>> <<type>> <<quasi-type>>
<<quasi-type>>	-O +I +R -D	<<type>> <<quasi-type>> <<category>>
<<material role>>	-O +I ~R +D	<<type>> <<quasi-type>> <<phased sortal>> <<material role>> <<formal role>> <<category>>
<<phased sortal>>	-O +I ~R -D	<<type>> <<quasi-type>> << phased sortal>> << category>>
<<formal role>>	-O -I ~R +D	<<category>> <<formal role>>

Contudo, a técnica OntoCon não contemplou todos os recursos existentes na técnica VERONTO e no Perfil OntoUML como, por exemplo, a verificação de relacionamentos parte-todo e de associação. Foi priorizado o tratamento dos relacionamentos de herança e a modelagem de papéis de forma mais ampla, para que fosse possível verificar uma diversidade de situações maior (TAVARES, 2008).

2.4.2. PrOntoCon

Como já discutido anteriormente, o uso da modelagem ontológica como ferramenta de validação de diagramas de classes UML exige do modelador um entendimento de conceitos filosóficos que, na maioria das vezes, não são inerentes à sua formação, ocasionando, assim, a dificuldade da aplicação de qualquer técnica ontológica voltada para esse fim. Com a OntoCon não poderia ser diferente. Mesmo sendo uma técnica construída para facilitar a vida do modelador, ela ainda está em um nível conceitual e filosófico que o desenvolvedor de software comum não consegue assimilar de forma rápida e satisfatória para sua correta aplicação durante a modelagem do domínio. Por essa razão, a implementação de um procedimento que sistematizasse as regras da OntoCon para facilitar sua aplicação se fez necessária. Assim, o procedimento intitulado PrOntoCon - Procedimento para uso da Técnica OntoCon - tem como objetivo principal facilitar a validação de diagramas de classes UML minimizando as

dificuldades advindas da Análise Ontológica devido aos seus conceitos e interpretações filosóficas.

Tabela 3: Propriedades com suas respectivas restrições hierárquicas e construtores UML permitidos. Extraída e adaptada de Tavares (2008).

Propriedade	**Estereótipo**	**Restrições Hierárquicas**	**Construtor UML**
Tipo	<<tipo>>	▪ Supertipos possíveis: <<categoria>>, <<tipo>>, <<quasi-tipo>>. ▪ Subtipos possíveis: <<tipo>>, <<quasi-tipo>>, <<papel material>>, <<fase>>. ▪ Todo objeto tem que ser instância de um *Tipo* em modelagem conceitual.	Classe abstrata ou classe concreta.
Quase-Tipo	<<quasi-tipo>>	▪ Supertipos possíveis: <<categoria>>, <<tipo>>, <<quasi-tipo>>. ▪ Subtipos possíveis: <<tipo>>, <<quasi-tipo>>, <<papel material>>, <<fase>>. ▪ Tem que ser subclasse de uma classe correspondente a <<tipo>> para herdar a identidade.	Classe abstrata ou classe concreta.
Papel Material	<<papel material>>	▪ Supertipos possíveis: <<categoria>>, <<tipo>>, <<quasi-tipo>>, <<fase>>, <<papel material>>, <<papel formal>>. ▪ Subtipos possíveis: <<papel material>>. ▪ Tem que ser subclasse de, no mínimo, uma classe correspondente a <<tipo>> para herdar a identidade.	Classe abstrata ou classe concreta.
Sortal com Fase	<<fase>>	▪ Supertipos possíveis: <<categoria>>, <<tipo>>, <<quasi-tipo>>, <<fase>>. ▪ Subtipos possíveis: <<papel material>>, <<fase>>. ▪ Tem que ser subclasse de uma classe correspondente a <<tipo>> para herdar a identidade.	Classe abstrata ou classe concreta.
Categoria	<<categoria>>	▪ Supertipos possíveis: <<categoria>>. ▪ Subtipos possíveis: todos. ▪ Normalmente, correspondem a classes de mais alto nível dentro de uma hierarquia.	Interface ou classe abstrata.
Papel Formal	<<papel formal>>	▪ Supertipos possíveis: <<categoria>>, <<papel formal>>. ▪ Subtipos possíveis: <<papel material>>, <<papel formal>>. ▪ Corresponde a um papel formal de uma classe em uma associação. ▪ Classe utilizada para organizar a hierarquia de classes correspondentes a papéis.	Interface ou classe abstrata.

Recapitulando, é possível dizer que através da aplicação da técnica OntoCon, três pontos chaves podem ser validados em um diagrama de classes: (i) relacionamentos de generalização/especialização; (ii) relacionamentos de classes envolvidas na modelagem de papéis; e (iii) definição adequada de construtores UML (TAVARES, 2008).

Portanto, a estruturação das etapas do procedimento PrOntoCon foram baseadas nesses três pontos, gerando quatro fases assim definidas: (i) identificação de estereótipos; (ii) verificação hierárquica; (iii) aplicação do padrão de projeto; e (iv) verificação de construtores UML. Para a formalização dessas fases, o SPEM (*Software Process Engineering Metamodel*) (OMG, 2008) foi adotado para a obtenção dos diagramas de atividade de cada uma das etapas que irão guiar o modelador durante a validação dos diagramas de classes de domínio.

O SPEM é um meta-modelo de engenharia de processo e também um framework conceitual, que pode prover os conceitos necessários para modelagem, documentação, apresentação, gerenciamento e intercâmbio de processos e métodos de desenvolvimento.

SPEM 2.0 é usado para definir processos de desenvolvimento de software e sistemas e seus componentes. O objetivo é acomodar um grande número de métodos e processos de desenvolvimento de diferentes estilos, culturas envolvidas, níveis de formalismo, modelos de ciclo de vida e comunidades. SPEM 2.0 define a habilidade para o desenvolvedor escolher o comportamento genérico do caminho de modelagem que melhor se encaixe em suas necessidades. Em outras palavras, SPEM 2.0 foca no provimento de estruturas de informação adicionais necessárias em processos modelados com UML 2.0 para descrever um processo de desenvolvimento existente (OMG, 2008).

A Figura 6 mostra os diagramas em SPEM das fases 1 (a) e 2 (b), respectivamente. A fase 1 do PrOntoCon tem como objetivo guiar o modelador para identificar qual o estereótipo OntoCon correspondente às classes existentes. Para esse fim, o modelador irá percorrer uma árvore de decisão contendo perguntas para ajudá-lo nessa tarefa, assim como vários exemplos e contra-exemplos para reduzir as dúvidas e os mal-entendidos que a análise do domínio pode gerar. Porém, nessa fase do PrOntoCon não

é possível distinguir entre os estereótipos <<type>> e <<quasi-type>>, podendo gerar classes identificadas como <<type>>/<<quasi-type>>, deixando para a segunda fase a tarefa de se chegar ao estereótipo específico. A fase seguinte, chamada *Verificação Hierárquica*, visa fazer a distinção entre classes <<type>>/<<quasi-type>> e verificar se há, ou se deveria existir, relacionamentos de generalização/especialização entre elas obedecendo as permissões de super/subclasses definidas na OntoCon. Dependendo do erro de modelagem encontrado durante o processo poderá redirecionar o modelador para a terceira fase ou conduzi-lo a refazer a classificação dos estereótipos para encontrar o possível erro.

A Figura 7 mostra os diagramas em SPEM das fases 3 (a) e 4 (b), respectivamente. A terceira fase do PrOntoCon aborda dois problemas: o primeiro deles diz respeito à classes <<type>> sendo subclasses de classes <<material role>>; e o segundo à uma classe <<material role>> sendo subclasse de mais de uma classe <<type>>. A quarta e última fase do procedimento é responsável pelo mapeamento dos construtores UML a serem utilizados pelo modelador para finalizar a construção do modelo conceitual do domínio de acordo com os estereótipos em que as classes foram classificadas. Dessa forma, classes estereotipadas como <<type>>, <<quasi-type>>, <<material role>> ou <<phased sortal>> devem ser mapeadas como classes abstratas ou concretas; enquanto classes estereotipadas como <<category>> ou <<formal role>> tornam-se interfaces ou classes abstratas.

Como exemplo de aplicação do procedimento PrOntoCon é apresentada a Figura 8, em que há um diagrama de classes UML parcialmente estereotipado em (a) (é possível verificar a presença de classes ainda indefinidas como <<type>>/<<quasi-type>>); e o resultado final do processo ilustrado em (b). Uma descrição mais detalhada sobre o procedimento e suas aplicações pode ser encontrada em Tavares (2008 e 2009).

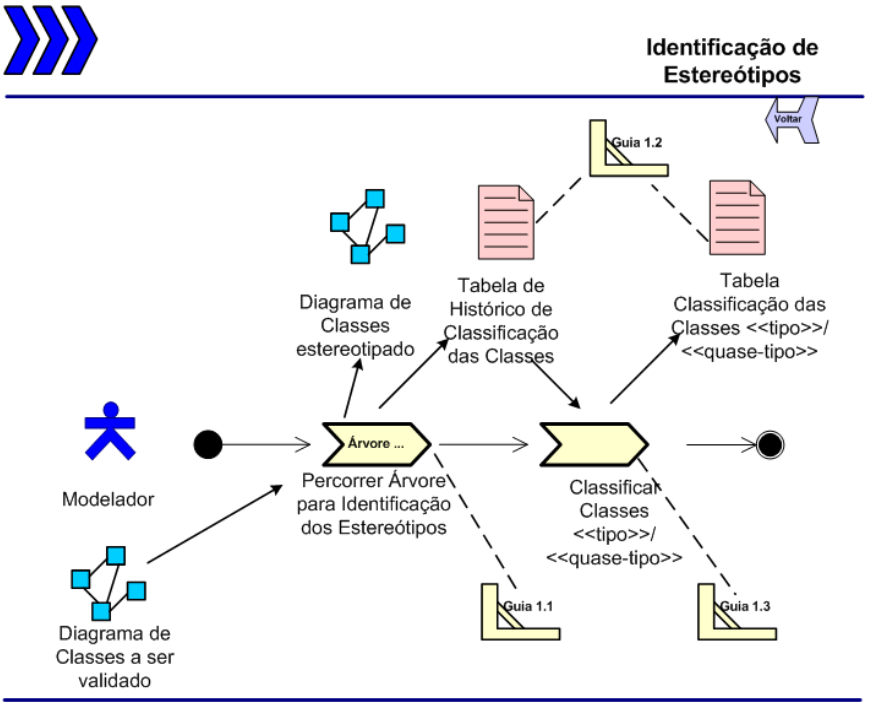


Diagrama de Atividade Fase 1: Identificação de Estereótipos

(a)

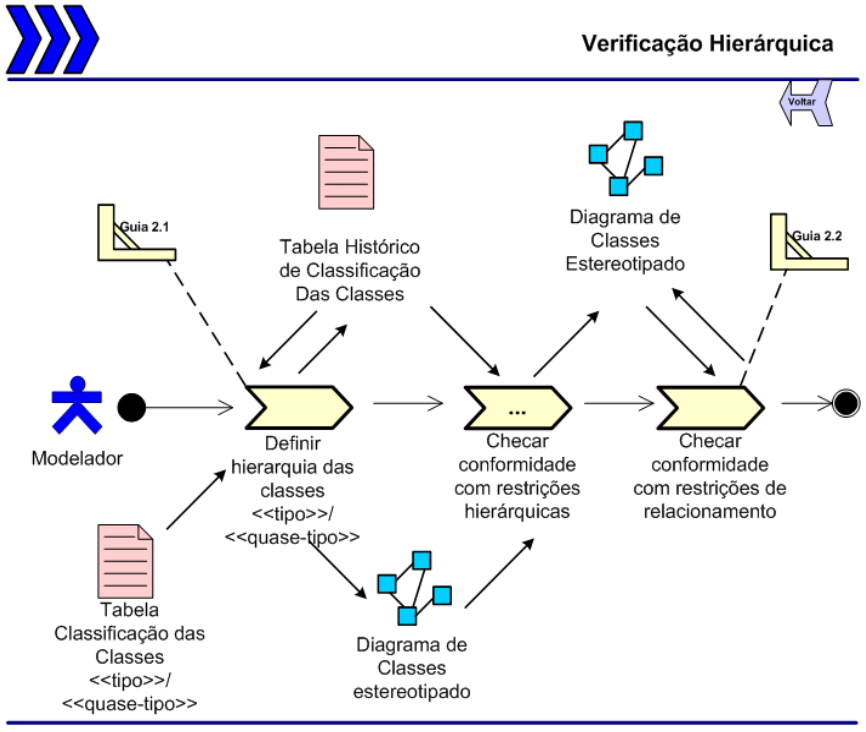
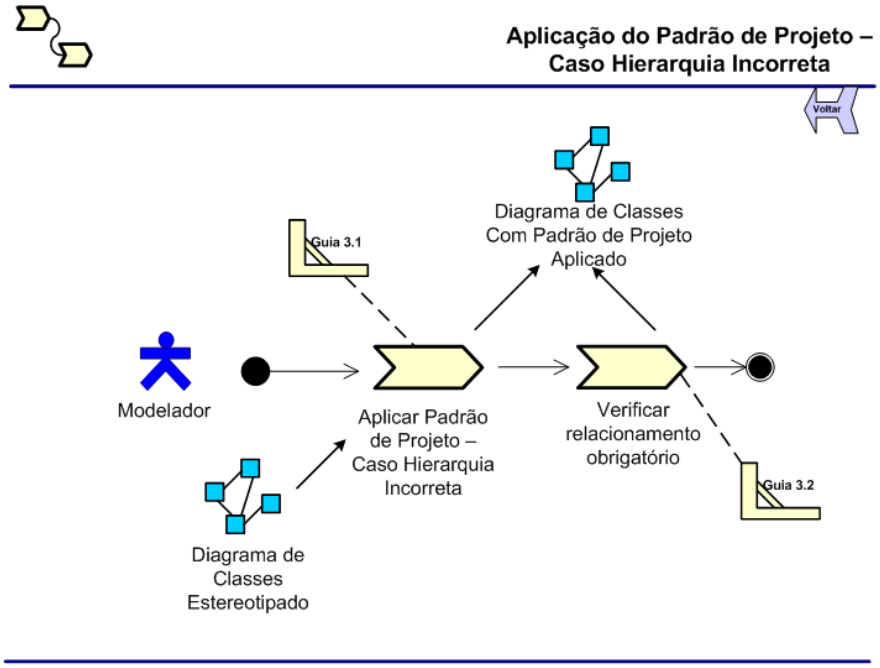


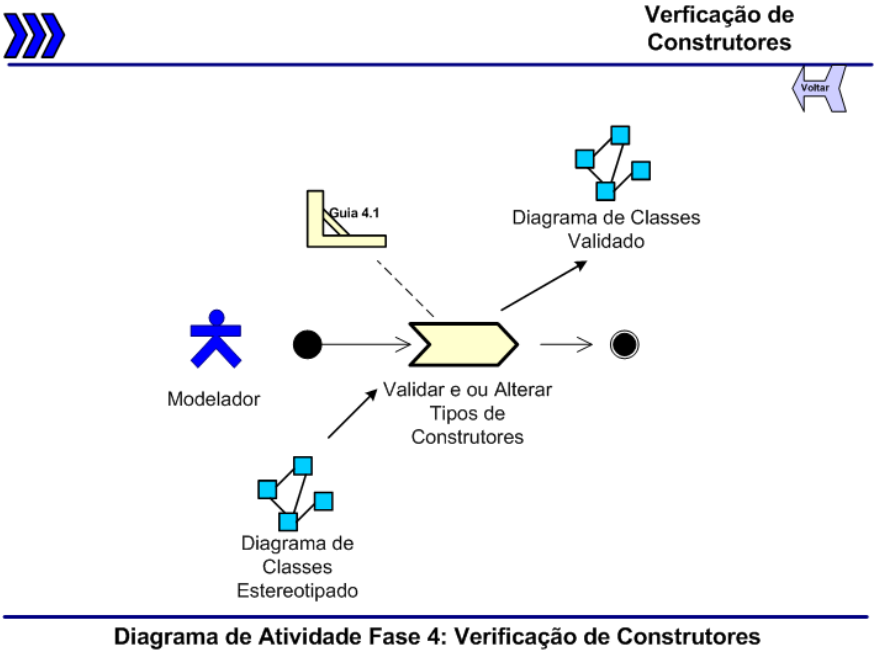
Diagrama de Atividade Fase 2: Verificação Hierárquica

(b)

Figura 6: (a) Diagrama da Fase 1: Identificação de Estereótipos. (b) Diagrama da Fase 2: Verificação Hierárquica. Extraído de Tavares (2008).



(a)



(b)

Figura 7: (a) Diagrama da Fase 3: Aplicação do Padrão de Projeto. (b) Diagrama da Fase 4: Verificação de Construtores UML. Extraído de Tavares (2009).

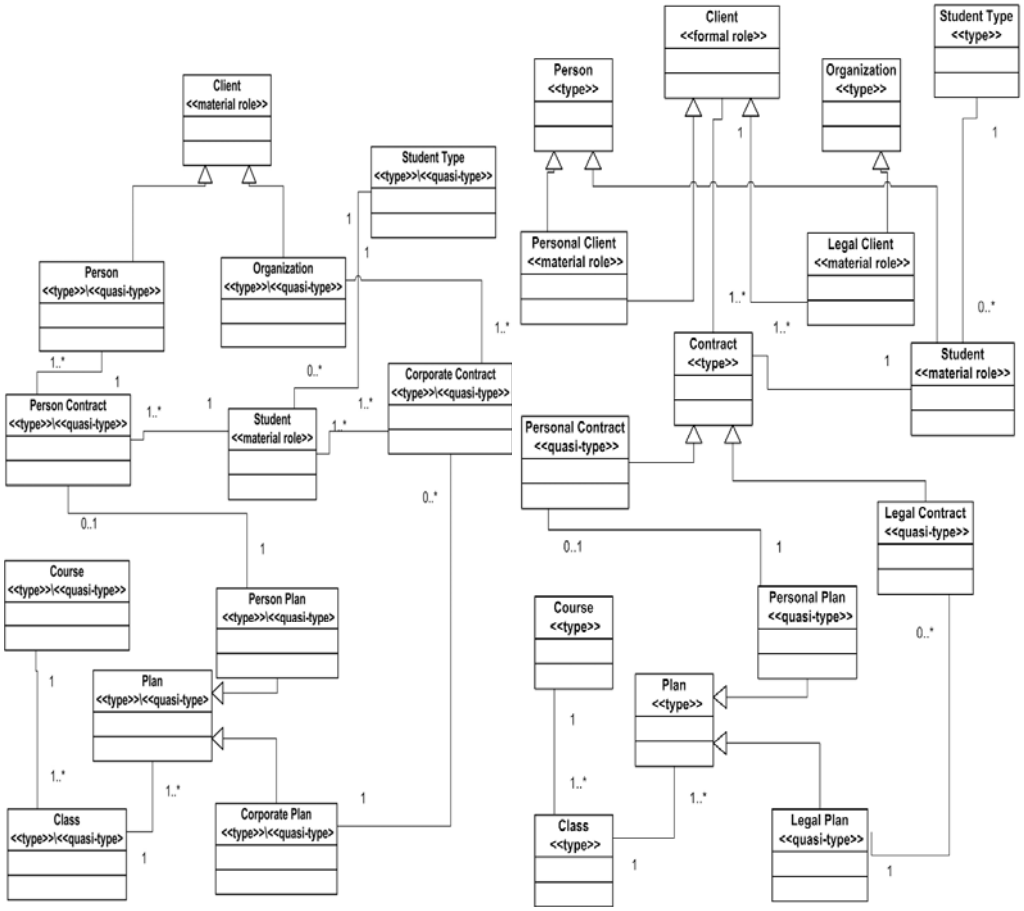


Figura 8: (a) Diagrama de classe UML parcialmente estereotipado pelo PrOntoCon. (b) Diagrama de classe UML ao final da aplicação do procedimento. Extraído de Tavares (2009).

3. OntoCon Estendida

Um modelo conceitual incorreto, ou que não represente adequadamente o domínio que está sendo modelado, pode levar a problemas de projeto, implementação, operação e manutenção dos sistemas, o que representa um dos principais desafios dos desenvolvedores de software hoje em dia. Dentro desse contexto, a UML tornou-se a linguagem de modelagem orientada a objetos padrão, sendo amplamente utilizada na construção dos modelos de domínio a fim de estabelecer uma ligação direta entre os elementos conceituais e os executáveis. Porém, a UML sozinha não é capaz de garantir a isenção de erros conceituais do modelo criado. Para esse fim, técnicas baseadas em análise ontológica surgiram para tentar facilitar o trabalho do modelador para validar seus diagramas de classes quando estes são expressos pela UML. Entre elas, podemos citar a VERONTO (VILLELA, 2004) e o Perfil OntoUML (GUIZZARDI, 2005), além da anteriormente comentada OntoCon (TAVARES, 2008) que surgiu da combinação destas duas últimas.

Portanto, é possível perceber que o uso de modelagem ontológica para validar diagramas de classes UML tem tido uma maior atenção dentro do meio acadêmico. Esse fato se deve à comprovada melhoria que tais técnicas agregam ao modelo de domínio sendo especificado, gerando modelos conceituais mais claros e confiáveis e que possam gerar sistemas mais robustos e manuteníveis, já que as aplicações desenvolvidas no mundo de hoje estão cada vez mais complexas e em constante expansão.

O objetivo principal da OntoCon foi a estruturação bem organizada de uma técnica que fosse capaz de permitir a validação de diagramas de classes UML, priorizando os relacionamentos de generalização e especialização, a fim de abranger uma variedade maior de situações. Assim, a técnica OntoCon definida por Tavares (2008) não contemplou a verificação de relacionamentos parte-todo e de associações simples, não menos importantes para uma modelagem conceitual mais fidedigna à realidade.

Agregação é um tipo de relacionamento que possui uma importância considerável na modelagem conceitual e é suportado por quase todas as linguagens de modelagem. Porém, os conceitos que envolvem a definição da parte e do todo são baseados na intuição. Muitos livros simplesmente sugerem evitar o uso dos relacionamentos de agregação ou explicam superficialmente a utilização destes, o que não é uma ajuda de grande valia quando esta informação é necessária para se modelar corretamente um diagrama conceitual. Mesmo entre as publicações de artigos, percebe-se que este ainda é um assunto tratado de forma limitada, tendo um número de pesquisadores e produções reduzido quando comparado a outras áreas dentro da Ciência da Computação. Assim, muitos modeladores acabam "inventando" suas próprias extensões da UML para resolver esses tipos de relacionamento, o que pode ocasionar problemas difíceis de serem tratados no futuro.

Por essa razão, viu-se a necessidade de extensão conceitual da OntoCon e, consequentemente do PrOntoCon, para abrigar os relacionamentos parte-todo e de associação simples, tornando o uso desse procedimento mais abrangente, capaz de satisfazer a maior parte das necessidades dos modeladores de software utilizando apenas uma única técnica.

É importante destacar que não houve modificações na classificação de propriedades, estereótipos, restrições hierárquicas e construtores UML correspondentes contidos na técnica OntoCon desenvolvida anteriormente por Tavares (2008). O presente trabalho visa somente acrescentar o suporte metodológico para a avaliação dos relacionamentos de agregação/composição e de associação. Tavares (2008) sugere o uso do conceito de *Relators* do perfil OntoUML (GUIZZARDI, 2005) para possibilitar a validação dos relacionamentos de associações, além de outros recursos fornecidos tanto pelo perfil OntoUML quanto pela VERONTO (VILLELA, 2004) para os relacionamentos parte-todo.

3.1. Justificativa da Linha de Pesquisa

Guizzardi et al. (2004b) introduzem o conceito de *relators* no contexto de *momentos*. *Momentos* são definidos como indivíduos que podem existir somente em outros indivíduos (o exemplo utilizado é de que uma carga elétrica pode somente existir em algum condutor de eletricidade), ou seja, *momentos* são existencialmente dependentes de outros indivíduos. A categoria de momentos inclui: (i) *qualities*: momentos que são dependentes de um único indivíduo (peso ou cor); (ii) *relators*: também chamados de momentos relacionais, são existencialmente dependentes de vários indivíduos (beijo, aperto de mão) e também de objetos sociais (uma conexão de vôo ou uma compra). Alguns termos não serão traduzidos para evitar a perda do significado original de suas obras que essa transformação pode ocasionar.

A partir daí, Guizzardi et al. (2004b) afirmam que relações são entidades que "colam" outras entidades (mais especificamente, momentos), podendo ser divididas em duas categorias: material e formal. São classificadas como relações formais aquelas que formam a superestrutura matemática do framework discutido em Guizzardi et al. (2004b) (relações de instanciação, *quale* de um *quality*, etc.) e também aquelas relações de domínio que apresentam características similares (relações de comparação como maior que, mais velho que, etc.). Por outro lado, relações materiais têm uma estrutura material intrínseca (e.g. beijos, brigas, reuniões). Assim, os relacionamentos de uma relação material são mediados por indivíduos chamados *relators*, que por sua vez possuem a capacidade de conectar entidades; por exemplo, uma conexão de vôo é um relator que conecta aeroportos, assim como uma matrícula é um relator que liga um estudante a uma instituição educacional.

Simplificadamente, um *relator* pode ser definido como sendo um indivíduo *r* existencialmente dependente de forma parcial de dois ou mais indivíduos *x* e *y*, distintos de *r* e entre eles; *r* é dito relacionar (*relate*) *x* a *y*. Além disso, um *relator* universal seria um universo onde suas instâncias são *relators*. Para entender o conceito de *relator* de uma forma mais completa, é necessário compreender outros conceitos envolvidos na taxonomia utilizada

por Guizzardi et al. (2004b) como, por exemplo, *quale* e *quality*. A Figura 9 mostra os relacionamentos existentes nessa taxonomia.

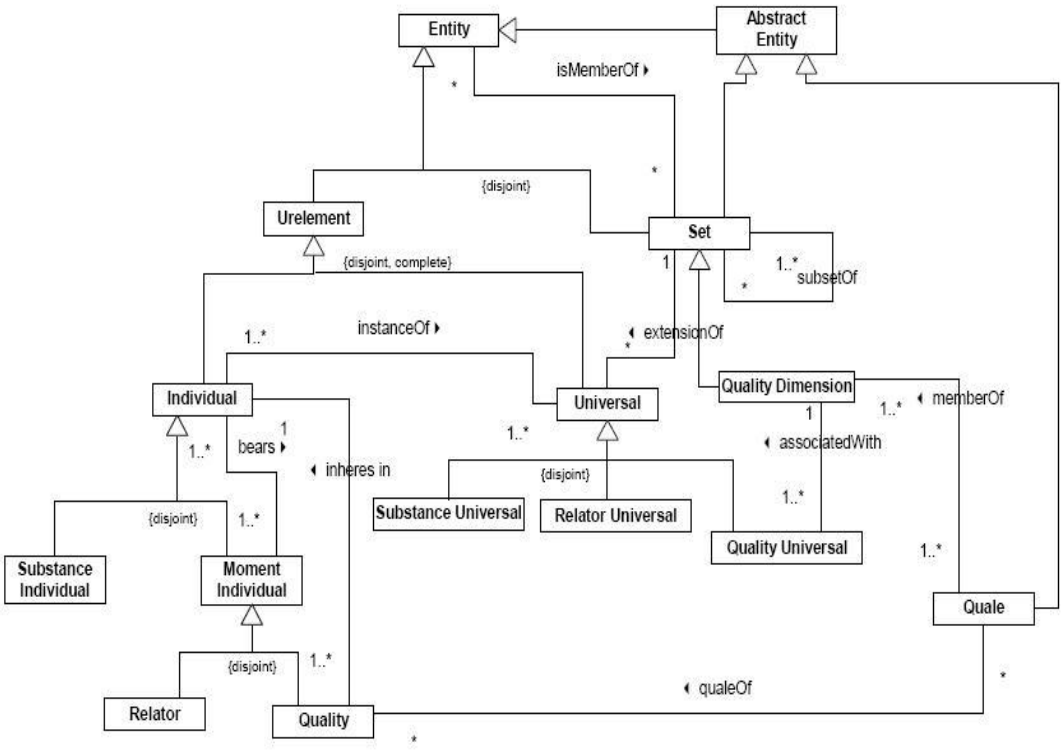


Figura 9: Taxonomia de um fragmento da Ontologia Formal Geral (ou GFO*, General Formal Ontology,* em inglês). Extraída de Guizzardi et al. 2004b.

Como é possível perceber, essas definições envolvem muitos conceitos filosóficos e uma capacidade de abstração e de entendimento muito grande, devido à sua complexidade. Maiores informações e detalhamentos sobre esses e outros conceitos encontram-se em Guizzardi (2005 e 2004b).

Além disso, Guizzardi (2007) propõe seis definições para incluir todos os possíveis tipos de relacionamentos parte-todo, sendo elas: dependência existencial, parte essencial, dependência genérica, parte obrigatória, parte inseparável e todo obrigatório. A seguir encontram-se suas respectivas definições:

- Dependência existencial: Um indivíduo x é existencialmente dependente de um outro indivíduo y (representado por ed(x,y)) se, e somente se, como um caso de necessidade, sempre que x existir deve existir y. Formalmente, $ED(x,y) =_{def} (\varepsilon(x) \rightarrow \varepsilon(y))$.

- Parte essencial: Um indivíduo x é uma parte essencial de um outro indivíduo y se, e somente se, y for existencialmente dependente de x e x for, necessariamente, uma parte de y. Formalmente, EP(x,y) =def ed(y,x) ^ (x ≤ y), ou ainda, EP(x,y) $=_{def}$ (ε(y) → (x ≤ y)). Em outras palavras, se a parte essencial for removida, então o todo deixa de existir, ou seja, perde sua identidade (e.g., o cérebro é uma parte essencial de pessoa e chassi é uma parte essencial de carro).
- Dependência genérica: Um indivíduo y é genericamente dependente de um tipo T se, e somente se, sempre que y existir é necessário que uma instância de T exista. Formalmente, GD(y,T) $=_{def}$ (ε(y) → ∃T,x ε(x)).
- Parte obrigatória: Um indivíduo x é parte obrigatória de outro indivíduo y se, e somente se, y for genericamente dependente de um tipo T que x instancia e y tem, necessariamente, uma instância de T como parte. Formalmente, MP(T,y) $=_{def}$ (ε(y) → (∃T,x)(x < y)). Por exemplo, coração é uma parte obrigatória de pessoa, assim como motor é uma parte obrigatória de carro.
- Parte inseparável: Um indivíduo x é uma parte inseparável de outro indivíduo y se, e somente se, x for existencialmente dependente de y e x for, necessariamente, parte de y. Formalmente, IP(x,y) $=_{def}$ (ε(x) → (x ≤ y)). No entanto, se a parte possuir existência sem estar conectada ao todo, então Guizzardi (2007) classifica esses casos como sendo *parte essencial com todo opcional* (e.g. o chassi não precisa estar relacionado ao carro para existir). Caso contrário, ou seja, se a parte for existencialmente dependente do todo, então ela é uma *parte inseparável*.
- Todo obrigatório: Um indivíduo y é um todo obrigatório para outro indivíduo x se, e somente se, x for genericamente dependente de um tipo T que y instancia e x for, necessariamente, parte do indivíduo instanciando T. Formalmente, MW(T,x) $=_{def}$ (ε(x) → (∃T,y)(x < y))). Em outras palavras, a parte não pode existir sem um todo, mesmo que esse todo seja um indivíduo diferente ao longo do tempo (e.g., devido aos transplantes de órgãos, dentre eles o coração, é possível dizer

que o coração pode existir antes do seu portador, ou seja, antes de ser transplantado, assim como sobreviver à sua morte, indicando a sua doação para outro portador, mas nunca sem a ligação com um indivíduo).

Tendo essas definições em vista, para poder distinguir todos esses casos a proposta de Guizzardi (2007) é estender a notação UML para agregação a fim de diferenciar partes essenciais de obrigatórias e partes inseparáveis de partes com todos obrigatórios, criando meta-atributos booleanos **essencial** e **inseparável**, respectivamente. Cabe nesse momento salientar que para que os conceitos de *relators* fossem utilizados também seria necessária a extensão na notação UML atual. Assim, é possível verificar que Guizzardi (2004b, 2005 e 2007) voltou suas pesquisas para a modificação do metamodelo da UML existente, a fim de acomodar suas novas definições para facilitar a modelagem desses conceitos no diagrama de classes.

E por esse motivo, o presente trabalho optou por não seguir essa linha de raciocínio, visto que as ideias ainda se baseiam em uma notação UML que não existe, e que também não é possível se dizer ao certo quando virá a existir. O foco deste trabalho é conduzir a tarefa do modelador de softwares de forma a produzir diagramas de classes mais compreensíveis utilizando-se as ferramentas disponíveis atualmente e, nesse caso, temos a UML 2.0. A tentativa de simplificação de conceitos filosóficos para modeladores comuns já não é uma atividade trivial; associada, ainda, ao fato de se introduzir conceitos que utilizam notações até agora não praticáveis na UML e tendo que adaptá-los à realidade atual da ferramenta tornou-se algo muito complexo, embora as definições sobre parte-todo descritas em Guizzardi (2007) sejam muito boas.

As restrições sobre os relacionamentos "parte-todo" encontradas na técnica VERONTO (VILLELA, 2004), juntamente com suas respectivas meta-propriedades e denominações, foram baseadas no trabalho de Guarino e Welty (2000c). Um esquema dessas restrições pode ser visto na Tabela 4. Já a adição de regras para agregações e associações na

VERONTO foi baseada no trabalho de Wand et al. (1999), podendo ser assim descritas:

- Regras para Relacionamentos Opcionais
    1. Associações opcionais (cardinalidade mínima 0) devem ser evitadas.
    2. A aquisição ou perda de uma associação deve ser modelada como uma mudança de classe.
    3. A capacidade de instâncias de uma classe participarem de uma associação, sem perderem propriedades, deve ser modelada como uma subclassificação (especialização).
- Regras para Agregação
    1. Cada classe componente deve ser associada com a classe composta através de um relacionamento de agregação.
    2. As propriedades emergentes da classe composta (“todo”) devem ser modeladas como atributos e associações.

Além disso, Wand et al. (1999) afirmam que um composto possui suas próprias propriedades e **deve** possuir no mínimo uma propriedade emergente, que pode ser modelada como um atributo (intrínseca) ou como um relacionamento (mútua). Define-se como propriedade emergente aquela que não é herdada de qualquer um dos componentes de um composto. Para ilustrar a diferença entre essas duas propriedades, Wand et al. (1999) utilizaram como exemplo o computador. O computador é um composto, pois é constituído de várias partes, como processador, memória principal, etc. A memória seria uma propriedade herdada porque é uma propriedade do componente memória. Por outro lado, o poder de processamento de um computador seria uma propriedade emergente porque não pode ser atribuída a qualquer componente isolado, visto que o poder de processamento é medido tendo como base o sistema como um todo. Wand et al. (1999) não oferecem maiores elucidações sobre propriedades emergentes, além de não distinguirem quando elas devem ser modeladas como atributos ou relacionamentos.

Tabela 4: Restrições sobre os relacionamentos parte-todo baseadas nas meta-propriedades unidade e identidade. Extraída e adaptada de Villela (2004).

Meta-Propriedades	Denominação das Propriedades	Construtor UML VERONTO	Restrições sobre Relacionamentos Parte-Todo
+I +U	Indivíduo	Classe concreta	▪ Pode ser uma classe parte de uma classe correspondente a um 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.
-I +U	Inteiro	Classe abstrata ou atributo	

Outras restrições estruturais para o "todo" incluem:

- fornecer cardinalidade mínima, ou seja, o número mínimo de partes em um dado relacionamento, que deve ser pelo menos um, mas pode ser maior;
- fornecer cardinalidade máxima para o número de partes em um dado relacionamento;
- se várias partes do mesmo tipo são permitidas no mesmo relacionamento "parte-todo", então elas representam coisas intercambiáveis.

Restrições envolvendo a "parte" definem que coisas que são “parte de” outras coisas podem desempenhar diferentes papéis, o que leva a pensar que subclasses dessas coisas devem ser formadas para acomodar esses papéis. Wand et al. (1999) dizem que se uma coisa não for sempre uma parte de outra coisa, então duas subclasses devem ser criadas, uma para representar a coisa independente (o todo) e outra o componente (a parte). Outro ponto relevante da pesquisa de Wand et al. (1999) é que somente foi considerado o aspecto mereológico dos relacionamentos parte-

todo. Porém, essa abordagem não garante que todos os tipos de relacionamentos parte-todo possam ser capturados. Além do mais, as restrições de cardinalidade mínima serem pelo menos um não se enquadram nas modelagens dos sistemas atuais, onde "todos" podem existir sem "partes" durante a sua existência e vice-versa, como já discutido anteriormente através de Guizzardi (2007).

Outro ponto negativo do método BWW (proposto por Bunge-Wand-Weber) é descrito em Guizzardi (2004b). O método BWW descrito em Wand et al. (1999) afirma que universais cujas instâncias são propriedades (momentos) não devem ser modelados como classes no modelo conceitual do domínio. Porém, em Guizzardi (2004b) é realizada uma interpretação guiada por formalidades para mostrar o que uma associação deve evidenciar. Dessa forma, foi possível demonstrar que representar momentos relacionais como classes não só está ontologicamente correto, mas também traz benefícios para uma abordagem prática sobre essa questão.

Além disso, Guarino e Guizzardi (2006) questionam a escolha de Wand e Weber pela ontologia de Bunge, visto que ela possui uma visão limitada de mundo e a justificativa apresentada por Wand e Weber está muito baseada na intuição e expressa de forma *ad hoc* ao longo da narrativa, comumente encontrando expressões do tipo “Eu usei a teoria de Bunge porque ela parece ser mais usável...”, “No meu ponto de vista, é a melhor teoria formulada e a mais completa...”, etc. O uso dessas expressões evidencia que a escolha da teoria de Bunge foi baseada nas próprias experiências de Wand e Weber, ao invés de ter sido sistematicamente testada e comparada com outras alternativas, o que seria a base para um contexto científico apropriado para a discussão do assunto.

Assim, por não prover um suporte metodológico satisfatório para a extensão do PrOntoCon no que diz respeito à relacionamentos de agregação/composição e associações, o trabalho de Wand et al. (1999) também não se encaixou no propósito do nosso trabalho sendo, portanto, não levado em consideração nessa pesquisa.

No entanto, os trabalhos de Keet (2006), Keet e Artale (2008) e Artale et al. (2008) trouxeram a discussão necessária para que a identificação dos

relacionamentos de agregação/composição pudesse ser feita de maneira transparente e relativamente simples.

3.2. Metodologia do presente trabalho

Keet (2006) parte do princípio de que relações parte-todo possuem tanto conceitos mereológicos quanto meronímicos, formando, assim, uma taxonomia baseada nessas duas características incluindo os tipos de relacionamentos mais relevantes para a modelagem conceitual dos relacionamentos parte-todo. A Figura 10 mostra essa taxonomia. A formalização desses tipos e seus detalhamentos estão presentes em Keet e Artale (2008).

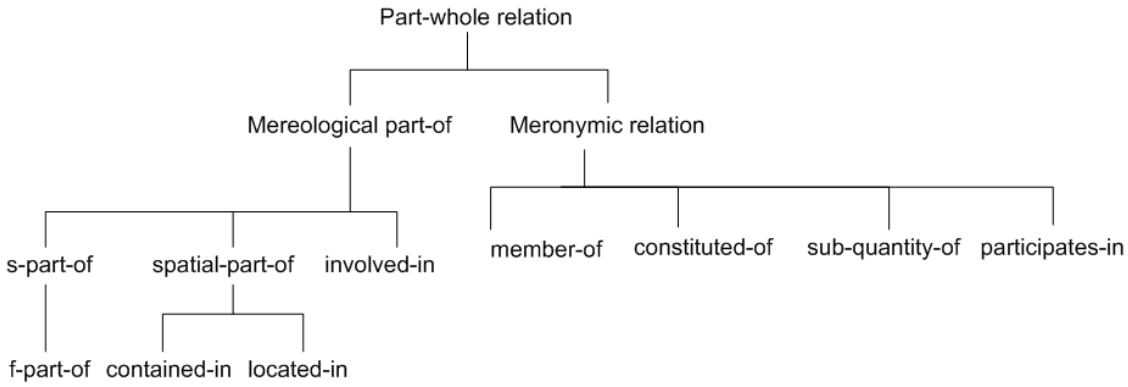


Figura 10: Taxonomia baseada em conceitos mereológicos e meronímicos das relações parte-todo, onde s-part-of = structural part-of e f-part-of = functional part-of. Extraída de Keet (2006).

Porém, o trabalho de Keet (2006) é voltado para a modelagem de relacionamentos parte-todo para a linguagem ORM. Dessa forma, o diagrama de decisão montado por ela prioriza a identificação de relacionamentos parte-todo para essa linguagem específica. A Figura 11 mostra esse diagrama de decisão. É possível perceber que até a pergunta que identifica uma relação *participates-in*, com exceção de *involved-in*, são perguntas baseadas na meronímia e daí para frente na mereologia.

Assim, fazendo-se um estudo minucioso em cima desses trabalhos - as formalizações em Keet e Artale (2008), o diagrama de decisão contido em Keet (2006) e as definições dos possíveis relacionamentos parte-todo

em Artale et al. (2008) - e voltado para o uso da UML 2.0, verificou-se que o diagrama de decisão da Figura 11 poderia ser condensado para adaptar-se ao objetivo da nossa atividade. Primeiramente, todos os conceitos da taxonomia tiveram que ser entendidos para que eles pudessem ser classificados como agregação ou composição, incluindo nesta tarefa os conceitos existentes em Guizzardi (2005) para as definições de relações parte-todo. É importante destacar que foi justificado anteriormente o não seguimento da linha de pesquisa de Guizzardi (2005) no que se refere ao uso específico dos *relators* e da notação para agregação/composição ainda inexistente para UML sugerida por ele, não significando o completo abandono de suas ideias e definições sobre essa área de estudo que envolve muito mais conceitos, visto que o próprio trabalho de Keet e Artale (2008) baseiam-se em várias definições presentes em Guizzardi (2005), sendo referenciadas sempre que necessário.

Artale et al. (2008) afirmam que se o todo for considerado rígido (rever discussão na Seção 2.3), então a distinção entre parte obrigatória e parte essencial, relacionadas à dependência genérica e específica, respectivamente, é suficiente para capturar todos os tipos úteis de comportamento de relações parte-todo. Entretanto, se o todo for não rígido, então é preciso acrescentar a ideia de partes imutáveis, geralmente sendo chamadas de partes essenciais condicionais. Isso significa que partes imutáveis também devem possuir uma dependência específica com o todo, porém, essa relação não tem a obrigatoriedade de acontecer durante toda a existência deles, mas somente enquanto o todo for uma instância daquele tipo. Um exemplo comum desse tipo, recorrente na literatura, é dado entre as mãos de um boxeador, ou seja, o boxeador deve ter as mesmas duas mãos enquanto ele exercer esse papel durante a sua existência.

Dessa forma, esses conceitos podem ser assim definidos, como feito em Artale et al. (2008):

1. *Dependência Genérica - Parte Obrigatória*: O todo deve ter uma parte em cada instante de sua existência. Assim, a presença da parte é obrigatória, mas pode ser substituída ao longo do tempo (e.g., coração-humano).

2. *Dependência Específica Incondicional - Parte Essencial*: A parte é obrigatória, mas não pode ser substituída sem destruir o todo (e.g., cérebro-humano).
3. *Dependência Específica Condicional - Parte Imutável* (também chamada de *parte essencial condicional*): A parte é obrigatória e não pode ser substituída, mas somente enquanto o todo pertencer à classe que o descreve (e.g., mão-boxeador).

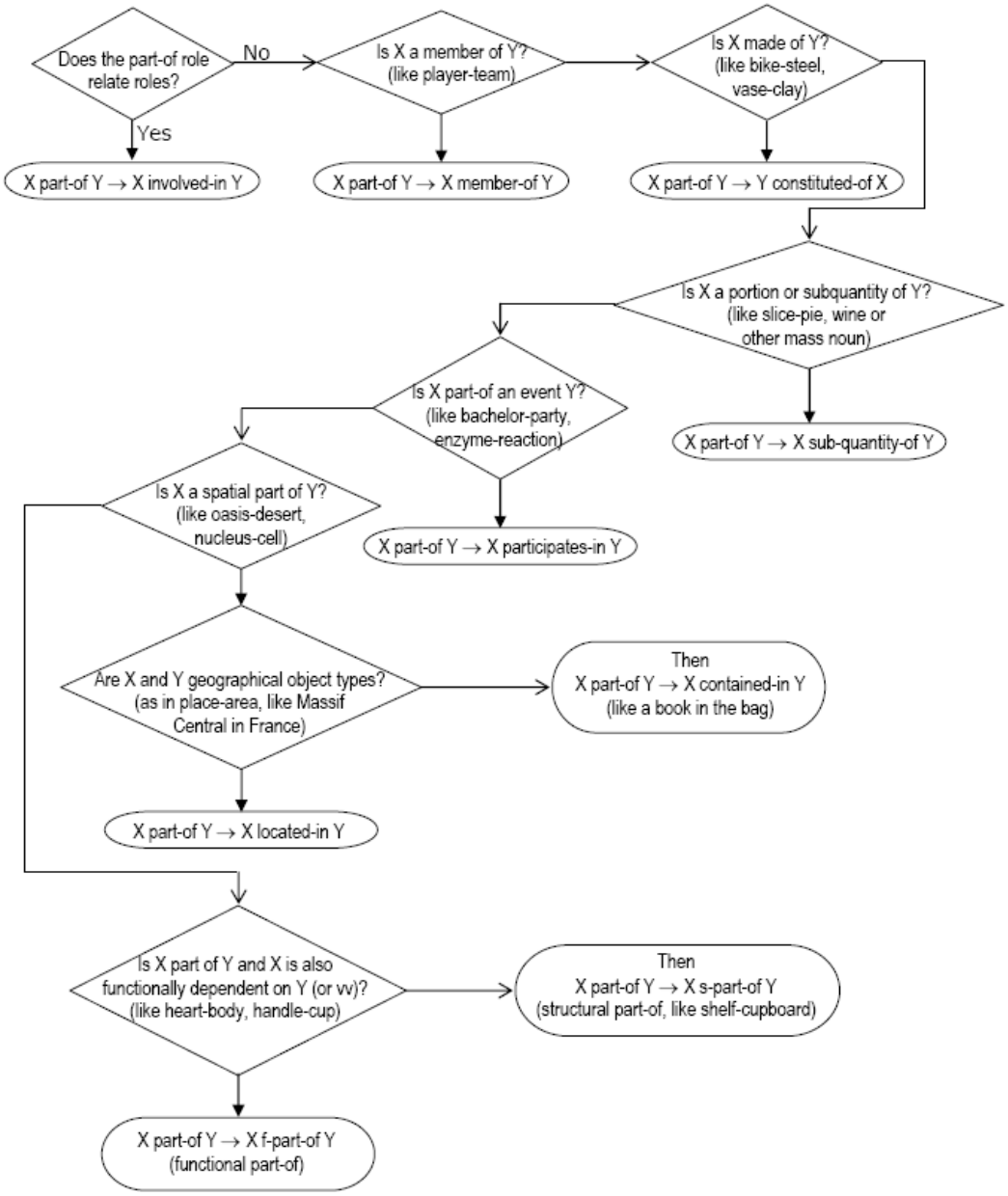


Figura 11: Diagrama de decisão para identificar a relação parte-todo apropriada. Extraída de Keet (2006).

Dessa maneira, todas essas três importantes considerações sobre os relacionamentos parte-todo foram utilizadas para se identificar a que tipo pertenciam todos os conceitos contidos na taxonomia de Keet (2006) (Figura 10). Assim, o estudo feito sobre as relações *member-of, constituted-of, sub-quantity-of, participates-in, involved-in, contained-in, located-in, s-part-of* e *f-part-of* e suas classificações podem ser resumidas como se segue:

- ***member-of***: Há dependência genérica – as partes podem ser mudadas ao longo do tempo sem perder o todo, porém, nesse caso, se não existir parte, o todo deixa de existir. Além disso, o todo é existencialmente dependente da parte e vice-versa. Está relacionado com objetos sociais ou seus papéis. Ex: jogador-time. Portanto, é um relacionamento parte-todo obrigatório com losango preenchido.
- ***constituted-of***: Está relacionado com objetos materiais. A parte não pode ser substituída sem alterar o todo, mas a questão é se o todo poderá mudar de classe durante a sua vida ou não (se é rígido ou de outro tipo, e.g., um vaso quebrado continua sendo uma instância de vaso ou muda de classe?). Essa questão sobre rigidez entre relacionamentos parte-todo é melhor detalhada em Varzi (2009). Exs: vaso-vidro, bicicleta-aço. Dependendo do caso, pode ser classificado como um relacionamento parte-todo essencial ou imutável.
- ***sub-quantity-of***: Está relacionado à quantidade-massa ou porção-objeto, e as partes são do mesmo tipo do todo. Exs: copo de vinho, pedaço de bolo. É um relacionamento parte-todo essencial.
- ***participates-in***: Relaciona “endurants” com “perdurants”. "Endurants" mapeiam grosseiramente para tipos de entidades e "perdurants" para processos ou relações/associações reificadas em modelos conceituais. Exs: pessoa-festa, enzima-reação. Assim, é um relacionamento parte-todo obrigatório e com losango preenchido.
- ***located-in***: Está relacionado com a ideia de lugares no espaço geográfico (2D). Exs: Paris-França, oásis-deserto. Portanto, é um relacionamento parte-todo obrigatório e com losango vazio.
- ***s-part-of***: Não pode haver dependência funcional e tanto a parte como o todo devem ser objetos físicos ao mesmo tempo ou objetos

não físicos ao mesmo tempo. Exs: asa-xícara, poltrona-avião. Assim, é um relacionamento parte-todo obrigatório e com losango vazio.

- ***f-part-of***: Está relacionado com a idéia de dependência funcional. Exs: coração-humano, cérebro-humano, programa-funções. Portanto, é um relacionamento parte-todo obrigatório (com losango vazio ou preenchido) ou essencial.
- ***involved-in***: Relaciona processos ou relações/associações reificadas (dois "perdurants"). Ex: mastigação-deglutição. Porém, a questão que surge é em qual tipo de relacionamento parte-todo ele pode se encaixar. Isso vai depender do tipo de modelagem. No caso de uma modelagem que captura cada instância de mastigação, vai existir uma dependência existencial entre a instância de mastigação e a instância do processo de deglutição. Já no caso onde se modela os tipos de processos, o processo de mastigação pode estar envolvido em outros tipos de processos como, por exemplo, mascar chiclete. Este é um tipo de meta-modelagem e possui um nível diferente do anterior. Neste caso o conceito é reificado.
- ***contained-in***: nem sempre é caracterizado como um tipo particular de parte-todo. Nesse caso, não será levado em consideração.

Esquematicamente, as classificações podem ser vistas na Tabela 5. É possível perceber que as relações *contained-in* e *involved-in* não se encontram na tabela. Isso se deve ao fato de que elas podem ocorrer raramente em um domínio para ser modelado conceitualmente, devido à complexidade da última e visto que a primeira nem sempre é caracterizada como um tipo particular de parte-todo, ambas podendo trazer muita dificuldade para que perguntas elucidativas para o modelador fossem elaboradas e sem agregar benefícios relevantes para essa identificação. E a relação *constituted-of* foi considerada essencial para fins práticos.

Dessa forma, as perguntas que irão dar o suporte metodológico para o procedimento PrOntoCon no que se refere aos relacionamentos parte-todo puderam ser elaboradas. A Tabela 6 mostra a sequência de perguntas e seus resultados, bem como alguns exemplos para ilustrar cada situação.

Para uma melhor visualização desta sequência, um diagrama de decisão foi montado, sendo apresentado na Figura 12.

Tabela 5: Esquema da classificação em relacionamentos essenciais ou obrigatórios das relações da taxonomia de Keet (2006).

Essencial	**Obrigatório**
sub_quantity_of	member_of (preenchido)
f_part_of	participates_in (preenchido)
constituted_of	located_in (vazio)
	s_part_of (vazio)
	f_part_of (vazio ou preenchido)

Entretanto, ainda falta o suporte metodológico para os relacionamentos de associações simples. O livro de Larman (2004) oferece um apoio muito bom para desenvolvedores de software que buscam utilizar UML e padrões em seus projetos, sendo até mesmo recomendado por Fowler (2005). Larman (2004) explica de forma detalhada as associações mais comuns que podem ser encontradas durante a modelagem de um domínio, fornecendo uma lista (como se fosse um *checklist*), separada por categorias de associações, para que o modelador consiga distinguir de forma adequada aquelas associações de classes que são pertinentes daquelas que são irrelevantes, podendo, assim, ser omitidas do modelo. Contudo, tendo como base os estudos realizados até o momento sobre relacionamentos parte-todo, constatou-se que várias categorias que estavam sendo sugeridas para serem modeladas como associações encaixavam-se, na verdade, em estruturas de agregação/composição. Esse acontecimento serviu para reafirmar a ideia de que a identificação dos relacionamentos parte-todo deve preceder a identificação das associações.

Dessa forma, as categorias de possíveis associações que se enquadraram em estruturas parte-todo (o diagrama de decisão definido na Figura 12 foi aplicado nessa tarefa) foram desconsideradas para a finalidade de identificação de associações, restando apenas as mostradas na Tabela 7 a seguir.

Tabela 6: Perguntas para ajudar a identificação de relacionamentos parte-todo em um diagrama de classes UML.

Perguntas	Sim	Não
Parte 1 – Obrigatório 1.1 Há dependência genérica entre a parte-todo, ou seja, o todo deve ter uma parte que pode ser substituída ao longo do tempo ou se o todo for destruído as partes também deixarão de existir? (e.g. funções-programa, poltrona-avião, subcategoria-categoria)	vá para a pergunta 1.2	vá para a Parte 2 – Essencial, pergunta 2.1
1.2 Porém, se as partes deixarem de existir, o todo continua existindo? (e.g. asa-xícara, se a asa quebrar, a xícara continua existindo; entretanto, em jogador-time, se não houver jogadores, o time deixa de existir)	Então é um relacionamento parte-todo obrigatório com o losango vazio. Encaixam-se nesse perfil relacionamentos estruturais (asa-xícara, poltrona-avião), funcionais (funções-programa, mãos-humano) e espaciais (cidade-país, oásis-deserto). Fim	Então é um relacionamento parte-todo obrigatório com o losango preenchido. Encaixam-se nesse perfil relacionamentos de agrupamento (jogador-time), de participação (pessoa-festa, enzima-reação) e funcionais (coração-humano). Fim
Parte 2 – Essencial 2.1 Há dependência específica entre a parte-todo, ou seja, as partes não podem ser substituídas sem destruir ou modificar o todo? (e.g. cérebro-humano, vaso-vidro, pedaço de bolo)	Então é um relacionamento parte-todo essencial com o losango preenchido. Encaixam-se nesse perfil relacionamentos funcionais (cérebro-humano), de constituição (vaso-vidro, bicicleta-aço) e sub-quantificados (taça de vinho, pedaço de bolo). Fim	Rever se o conceito de parte-todo realmente se encaixa nesse tipo de relacionamento. É provável que seja somente uma associação simples.

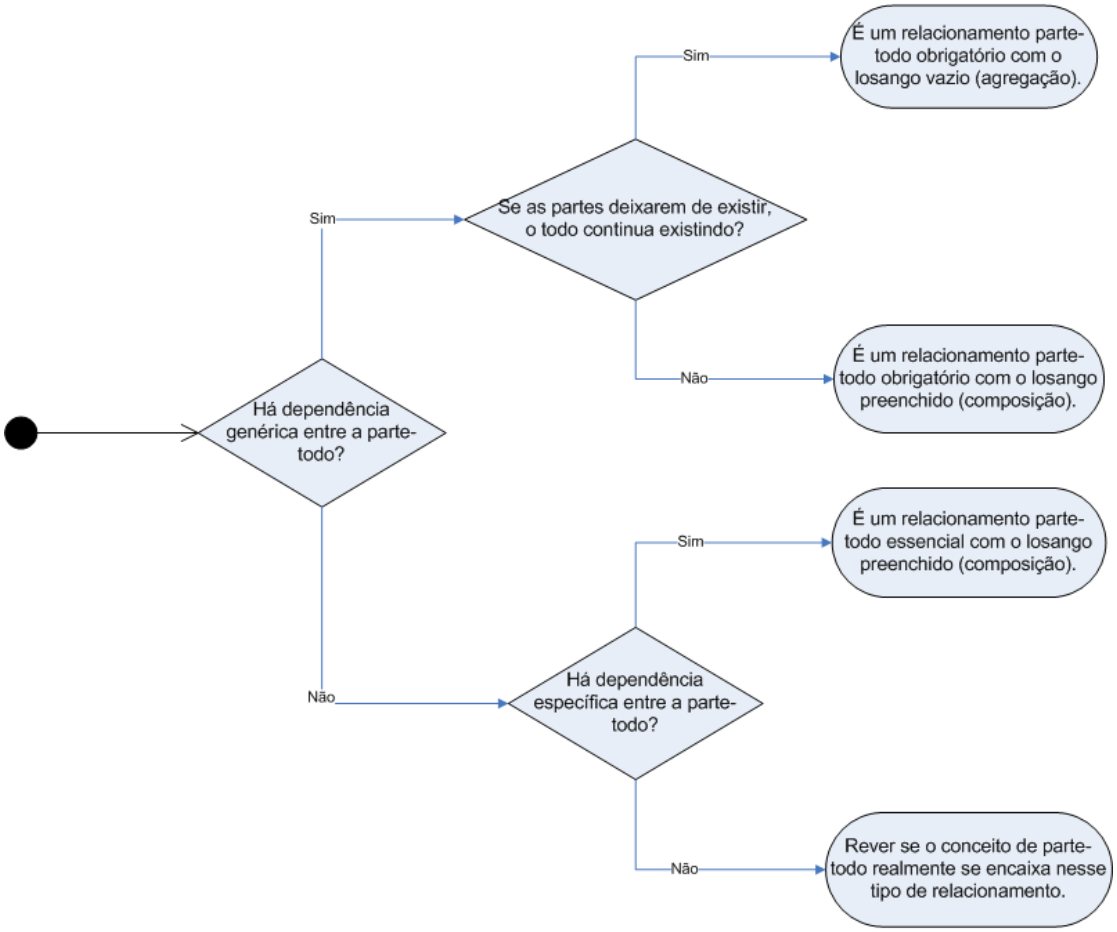


Figura 12: Diagrama de decisão baseado nas perguntas da Tabela 5.

Tabela 7**:** Definição das categorias pertinentes à identificação de associações.

Categoria	**Exemplos**
A está fisicamente contido em B.	Registro-Loja, Item-Prateleira, Passageiro-Aeronave
A usa ou gerencia B.	Caixa-Registro, Piloto-Aeronave
A se comunica com B.	Cliente-Caixa, AgenteDeReservas-Passageiro
A está relacionado a uma transação B.	Cliente-Pagamento, Passageiro-Bilhete
A é uma transação relacionada com uma outra transação B.	Pagamento-Venda, Reserva-Cancelamento
A é adjacente a B.	Produto-Produto, Cidade-Cidade

Nesse ponto, o suporte metodológico desejado para a extensão do PrOntoCon tanto para relacionamentos de agregação/composição quanto de associações foi conseguido. Porém, é importante destacar uma adição feita em relação ao trabalho de Keet (2006) e Keet e Artale (2008), que foi a consideração do uso de estruturas compostas para a modelagem de domínios em que "partes" podem interagir com outras "partes". Esse fato promove a atualidade e a conformidade dos estudos realizados com a UML 2.0 existente, de forma a ajudar a disseminação e um maior entendimento sobre o uso dessas estruturas. Essa adição e outras considerações serão feitas durante a descrição do PrOntoCon Estendido que será feita a seguir.

4. PrOntoCon Estendido

Como já discutido anteriormente, a criação de um procedimento prático baseado em conceitos de modelagem ontológica tem como objetivo facilitar a condução da tarefa do desenvolvedor de software durante a modelagem do domínio do problema em questão. Dessa forma, o procedimento intitulado PrOntoCon - Procedimento para uso da Técnica OntoCon - visa facilitar a validação de diagramas de classes UML minimizando as dificuldades advindas da Análise Ontológica devido aos seus conceitos e interpretações filosóficas.

O PrOntoCon foi primeiramente desenvolvido por Tavares (2008) e contemplou somente a validação de relacionamentos de generalização/especialização. Nesse primeiro estudo, o procedimento ficou sendo constituído por quatro fases necessárias para que os diagramas de classes fossem analisados do ponto de vista de seus domínios pelo modelador, sendo sugeridas mudanças ou adições nos relacionamentos pertinentes à hierarquia. As fases do PrOntoCon desenvolvido por Tavares (2008) foram brevemente descritas na Seção 2.4.2. Assim, uma nova etapa deve ser adicionada ao PrOntoCon a fim de validar os relacionamentos de agregação/composição e de associações, que é o estudo do presente trabalho. Essa nova etapa virá depois das quatro fases definidas no PrOntoCon original e será constituída por apenas uma fase, dividida em duas atividades distintas: (i) verificação de agregação/composição; e (ii) verificação de associações. Toda vez que nos referirmos ao procedimento PrOntoCon como "PrOntoCon Estendido" significa que estaremos levando em consideração somente esta última fase, correspondente aos relacionamentos parte-todo e de associações.

Assim como o PrOntoCon, o PrOntoCon Estendido foi modelado utilizando-se o SPEM (OMG, 2008) juntamente com a ferramenta Microsoft Visio 2003, a fim de se manter a consistência do método já existente. A

Figura 13 apresenta como a nova fase é vista pelo modelador. As seções seguintes irão detalhar a nova fase com suas respectivas atividades.

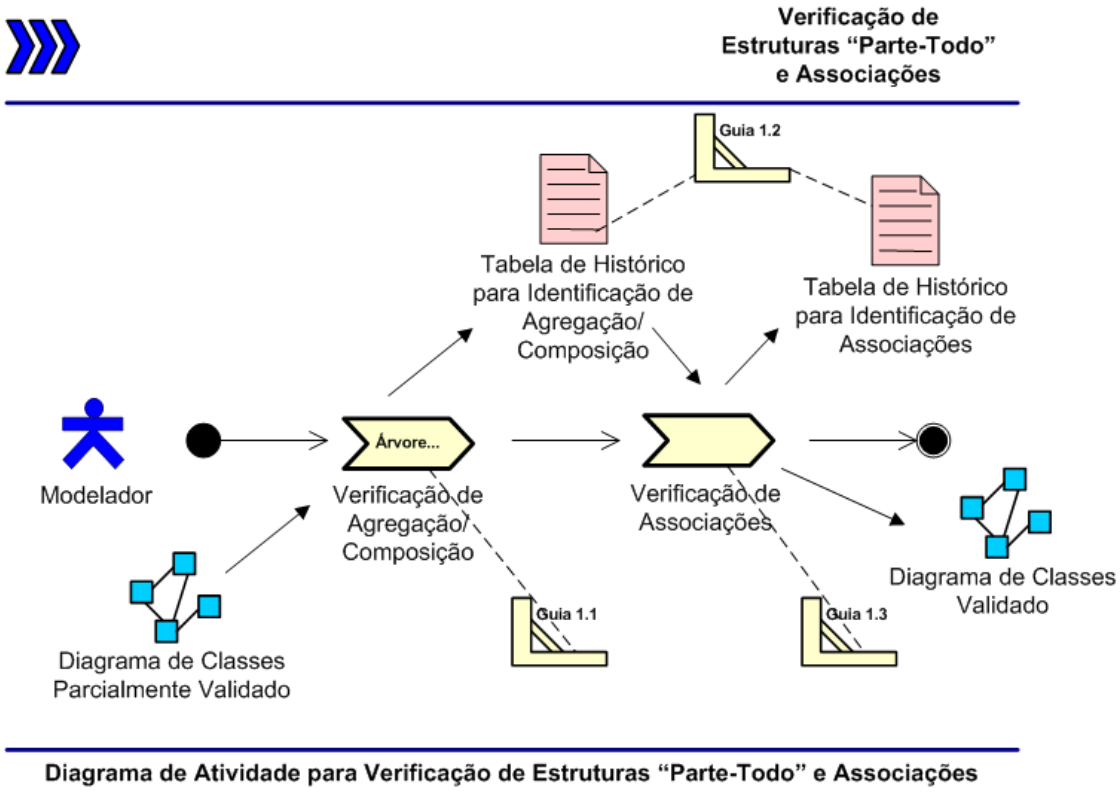


Figura 13: Diagrama de Atividade da nova fase: verificação de estruturas parte-todo e associações.

4.1. Primeira Atividade: Verificação de Agregação/Composição

No momento em que o modelador tem acesso a essa fase do procedimento, indicando que a atividade de verificação de agregação/composição deve ser realizada antes da atividade de verificação de associações, ele recebe as primeiras orientações. O Guia 1.1, como mostrado na Figura 13, esclarece o modelador sobre o que é o diagrama de classes parcialmente validado e como ele deverá proceder enquanto estiver fazendo a validação dos relacionamentos parte-todo. De modo geral, o diagrama de classes parcialmente validado refere-se ao diagrama final obtido através da aplicação do PrOntoCon, em que somente os relacionamentos de hierarquia foram validados. Por essa razão, estes relacionamentos não deverão ser

analisados novamente nesta atividade atual, apenas os que não envolvem generalização/especialização e, além disso, todos deverão passar pela verificação de relacionamentos parte-todo, mesmo aqueles que se encontram modelados como associações. O objetivo desta primeira atividade não é somente verificar se os relacionamentos de agregação/composição estão corretos, mas também identificar se estruturas parte-todo encaixariam-se naquela situação, onde, em um primeiro momento, o modelador não teria se atentado para o fato.

Para realmente se ter início à atividade, o modelador deve clicar em "Árvore...", como mostrado pela Figura 13, para ter acesso à um diagrama de decisão contendo as perguntas e exemplos necessários para que ele identifique se aquela relação sendo analisada irá se enquadrar em algum caso de dependência, caracterizando, assim, um relacionamento de agregação ou composição. A Figura 14 apresenta o diagrama de decisão encontrado pelo modelador ao executar essa ação.

Nesse instante, o modelador deverá responder a uma pergunta inicial que poderá guiá-lo para um ou para outro ramo da árvore de decisão, levando-o para a identificação do tipo de relacionamento entre as classes em análise. Cada uma das perguntas que compõem esse diagrama de decisão foram elaboradas de forma a facilitar o entendimento do modelador sobre o que é preciso identificar na possível relação das classes sendo analisadas sem, necessariamente, se aprofundar nos conceitos filosóficos envolvidos. Entretanto, muitas vezes somente um exemplo não é o suficiente para que o modelador consiga comparar o seu problema em questão com a pergunta sendo feita, podendo, assim, gerar análises superficiais. Por esse motivo, o modelador conta com um suporte para análise, contendo outros exemplos e contra exemplos de vários domínios, para que a abstração para o seu domínio específico possa ser feita de forma mais clara e simples. É importante frisar que todos os exemplos e contra exemplos contidos no PrOntoCon Estendido foram retirados da literatura e dos estudos de casos feitos para essa pesquisa.

A primeira questão da atividade tenta identificar se há dependência genérica entre as partes, ou seja, aquela em que tanto o "todo" como a "parte" podem ter existências independentes. A Tabela 8 ilustra o suporte

para análise dessa pergunta contido no PrOntoCon Estendido quando o modelador a seleciona. Se a resposta for afirmativa, então é perguntado se há a existência do todo sem as partes. Dependendo da resposta, o tipo de relacionamento pode ser uma agregação (losango vazio) ou uma composição (losango preenchido). Por outro lado, se não houver dependência genérica entre a parte e o todo, a questão sobre dependência específica é argumentada, ou seja, aquela em que as partes não podem ser substituídas sem modificar ou destruir o todo. Se, nesse caso, a resposta for afirmativa, então o relacionamento é uma composição. Caso contrário, o modelador é guiado para a próxima atividade para verificar associações.

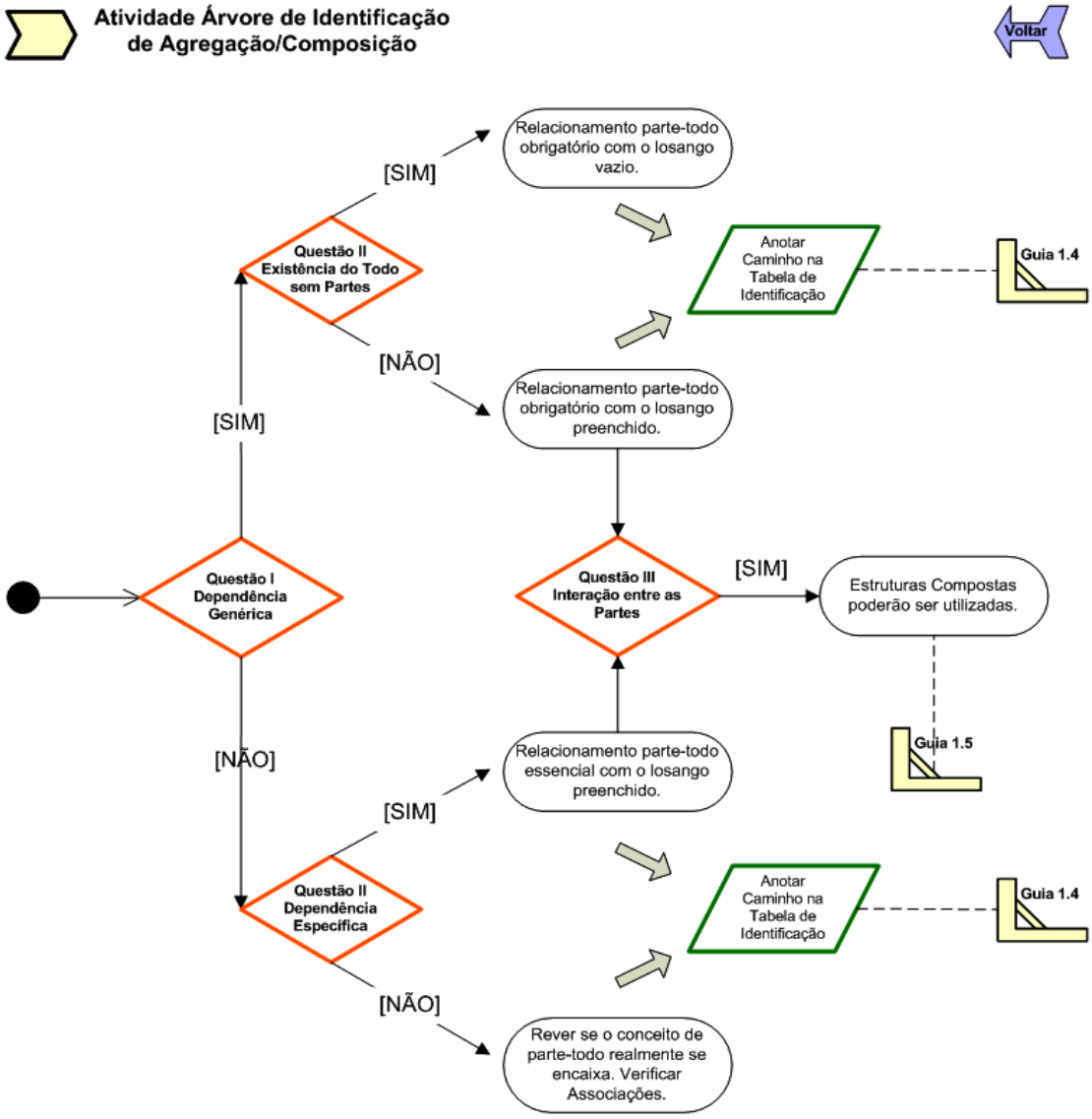


Figura 14: Árvore de identificação de Agregação/Composição.

Tabela 8: Suporte para análise relacionado à Questão I - Dependência Genérica.

Questão I – Dependência Genérica
Pergunta:
Há dependência genérica entre a parte-todo, ou seja, o todo deve ter uma parte que pode ser substituída ao longo do tempo? Ou ainda, se o todo for destruído as partes também deixarão de existir? Exemplo: na estrutura parte-todo FUNÇÕES-PROGRAMA podemos dizer que um PROGRAMA é composto por FUNÇÕES que podem ser substituídas ao longo do tempo; assim como em POLTRONA-AVIÃO podemos verificar que um AVIÃO possui POLTRONAS substituíveis e que deixarão de existir caso o avião sofra um acidente grave.
Suporte para análise da pergunta:
i. Outros exemplos: • (SUB)CATEGORIA-CATEGORIA: uma categoria (e.g. de produtos) pode ter subcategorias, porém, se uma categoria for excluída do registro, todas as subcategorias que se relacionavam com ela não têm mais o porquê de existir. • FUNCIONÁRIO-EMPRESA: uma empresa deve possuir funcionários que podem ser substituídos ao longo do tempo por alguma razão. • CORAÇÃO-HUMANO: devido aos transplantes, um humano pode ter o órgão coração substituído por outro. • ITEMCARRINHO-PEDIDO: para um pedido de compras ser efetivado ele deve possuir itens, e os itens de um pedido podem ser substituídos ao longo do tempo até o fechamento da compra. **ii. Contra exemplos:** • CÉREBRO-HUMANO: o cérebro não pode ser substituído sem ao menos causar algum dano ao humano. • VASO-VIDRO: um vaso é constituído de vidro, porém, se ele vier a quebrar, suas partes não poderão ser substituídas sem causar alteração no vaso original. • CUPOMDESCONTO-PEDIDO: o uso de um cupom de desconto é opcional, ou seja, o cliente utiliza-o se quiser e/ou se possuir. Assim, mesmo que o cliente possua um cupom de desconto, mas opta por não usá-lo no fechamento do pedido, podemos verificar que tanto o cupom quanto o pedido têm vidas independentes, o que não caracteriza uma dependência genérica. **iii.** A pergunta "se o todo for destruído as partes também deixarão de existir?" é uma pergunta auxiliar para guiar o modelador na identificação de uma dependência genérica. Não significa que as duas perguntas mencionadas acima devam ser afirmativas para passar para a próxima pergunta da árvore, basta que uma delas seja verdadeira.

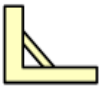
Então, quando o modelador já tiver identificado o tipo de relacionamento das classes de domínio sendo analisadas, esse será o momento de verificar se o uso de estruturas compostas é possível para os relacionamentos classificados como composição. Assim, uma pergunta adicional foi acrescentada àqueles relacionamentos que já poderiam ser modelados como composição utilizando-se a notação usual da UML, questionando se as partes interagem entre si. Se a resposta for negativa, nenhuma alteração deve ser feita no modelo; e, mesmo que a resposta seja afirmativa, ou seja, o uso de estruturas compostas é válido para facilitar a modelagem, a decisão final será sempre do modelador. O PrOntoCon Estendido apenas sugere o uso de estruturas compostas para esses casos, não significando que o modelo estará "mais correto" ou "mais errado" sem o uso delas; caberá ao modelador acatar ou não a sugestão, dependendo da sua familiaridade com a notação UML 2.0 ou com seu estilo de modelagem. O Guia 1.5 (mostrado na Figura 14) disponibiliza um exemplo ilustrado sobre o uso de estruturas compostas.

Também foi disponibilizado para o modelador um formulário modelo para que ele anote as decisões tomadas em cada pergunta e quais são as classes envolvidas, para facilitar a modelagem posteriormente. O Guia 1.2 (mostrado na Figura 13) possui o formulário que deve ser utilizado nessa tarefa e o Guia 1.4 (mostrado na Figura 14) mostra um exemplo desse formulário preenchido para guiar corretamente o modelador sobre sua função. As imagens de todos os elementos presentes no PrOntoCon Estendido estão disponíveis nos Apêndices A e B.

4.2. Segunda Atividade: Verificação de Associações

Quando as relações sendo analisadas pelo modelador não se enquadram em nenhum tipo de dependência entre a parte e o todo (chegando à folha da árvore que solicita a revisão do conceito parte-todo, podendo ser verificado na Figura 14), o PrOntoCon Estendido leva o modelador a realizar a segunda atividade dessa nova fase, que é a verificação de associações para essas relações em que a estrutura parte-todo não se encaixou.

Para se ter início a essa atividade, o modelador deve clicar no Guia 1.3 (como mostrado na Figura 13) para ter acesso às possíveis categorias de associações mais comuns. Cada uma delas possui exemplos elucidativos para ajudar o modelador nessa identificação. A Tabela 6, apresentada na Seção 3.2, ilustra quais são essas categorias e seus respectivos exemplos. No entanto, o PrOntoCon Estendido acrescentou suportes a essas questões, pelo mesmo motivo já comentado na seção anterior. Assim, a Figura 15 mostra o que o modelador encontrará ao iniciar esta segunda atividade.



Guia 1.3 – Atividade Verificação de Associações

- Verifique se os pares de classes que não se encaixaram como Agregação/Composição irão se encaixar como Associação (utilize a Tabela de Histórico para Identificação de Agregação/Composição como referência).
- Para esta tarefa, identifique se cada um dos pares de classes se enquadram em uma das categorias abaixo e anote na Tabela de Histórico para Identificação de Associações. Cada categoria possui exemplos extraídos dos domínios de Lojas e Reservas de Passagens Aéreas. Se somente os exemplos não forem suficientes para elucidar a identificação, clique no suporte correspondente para obter um detalhamento maior sobre a questão.

Categoria	Exemplos	Suporte
A está fisicamente contido em B.	Registro-Loja, Item-Prateleira, Passageiro-Aeronave	1
A usa ou gerencia B.	Caixa-Registro, Piloto-Aeronave	2
A se comunica com B.	Cliente-Caixa, AgenteDeReservas-Passageiro	3
A está relacionado a uma transação B.	Cliente-Pagamento, Passageiro-Bilhete	4
A é uma transação relacionada com uma outra transação B.	Pagamento-Venda, Reserva-Cancelamento	5
A é adjacente a B.	Produto-Produto, Cidade-Cidade	6

Obs.: Se ainda sobrarem classes sem relacionamentos, é provável que as etapas para verificação de Agregação/Composição e Associações tenham que ser refeitas. Se ainda assim o possível erro não tiver sido encontrado, então o modelador deverá refazer o processo inteiro, desde a verificação de Generalização/Especialização.

Figura 15: Atividade para verificação de associações.

O suporte feito para esta segunda atividade tentou seguir o padrão do suporte realizado para a primeira atividade da fase, disponibilizando mais exemplos com explicações pertinentes para ajudar o modelador na comparação do seu problema de domínio em questão com o que está sendo pedido para se identificar nas perguntas. A Tabela 9 mostra o modelo de um dos suportes construídos para esta atividade.

Tabela 9: Suporte referente à primeira categoria para identificação de associações (quando A está fisicamente contido em B).

Suporte 1
Pergunta:
A classe A está fisicamente contida na classe B? Exemplo: entre ITEM e PRATELEIRA podemos dizer que itens estão fisicamente contidos em alguma prateleira da loja. Portanto, encaixam-se nessa categoria e formam uma associação simples. Não são classificadas como agregação/composição porque não possuem dependência genérica (pode haver uma prateleira vazia e a existência de itens e prateleira são independentes) e nem dependência específica (a substituição dos itens não modifica a prateleira).
Suporte para análise da pergunta:
i. Outros exemplos: • REGISTRO-LOJA: o registro em papel de uma venda pode estar fisicamente contido em algum arquivo da loja, formando, assim, uma associação. Não são classificadas como agregação/composição porque não possuem dependência genérica (os registros não precisam necessariamente ser guardados na loja física e o fechamento da loja não implica na destruição dos registros) e nem dependência específica (a substituição dos registros não modifica ou destrói a loja). • PASSAGEIRO-AERONAVE: o passageiro está inserido naquela aeronave para aquele determinado vôo, caracterizando uma associação. Não são classificadas como agregação/composição porque não possuem dependência genérica (a aeronave pode estar vazia quando não há vôos programados e aeronave e passageiro têm existências independentes) e nem dependência específica (a substituição dos passageiros não implica na modificação da aeronave).

Se ao final da verificação dessas categorias ainda houver classes sem relacionamentos no modelo, o PrOntoCon Estendido alerta o modelador de que esta fase deve ser refeita. E, se mesmo assim o possível

erro não for encontrado, então será preciso refazer as fases anteriores do PrOntoCon (aquelas que dizem respeito à validação de relacionamentos de generalização/especialização). No entanto, se a validação de todas as classes e relacionamentos é feita de forma satisfatória, ou seja, sem nenhuma "sobra", o diagrama de classes final é considerado validado tanto para relacionamentos de generalização/especialização quanto para os de agregação/composição e de associações.

5. Estudo de Casos

Nesta seção serão abordadas duas soluções definidas para problemas de domínios diferentes já modelados conceitualmente através de diagramas de classes UML. Esta seção tem como objetivo demonstrar a aplicação do PrOntoCon Estendido a cada um desses diagramas para que as melhorias proporcionadas pelo uso do procedimento possam ser discutidas durante a apresentação desses exemplos.

5.1. Domínio de Compras Online

O primeiro exemplo de domínio apresentado para a aplicação do PrOntoCon Estendido é um domínio de comércio eletrônico, muito familiar para aquelas pessoas que já realizaram alguma compra pela Internet. Além da praticidade de se fazer compras parceladas em várias vezes e ainda receber tudo em casa sem nenhum esforço, o preço muitas vezes menor das lojas online do que as físicas tem atraído cada vez mais consumidores para essa nova realidade. Porém, tanta praticidade oferecida para facilitar a vida dos usuários vem acompanhada de muita responsabilidade também. Isso acontece porque compras envolvem números de cartões de crédito, endereços de entrega, milhares de pedidos realizados por dia, outros cancelados e várias outras atividades contidas nessa tarefa que, se contiverem erros durante as suas execuções, podem trazer um transtorno muito grande para o comprador e para a empresa. Assim, esse domínio requer uma atenção especial durante sua modelagem para evitar problemas relacionados a inconsistência de dados ou até mesmo a perdas dos mesmos.

O exemplo utilizado foi retirado da revista MundoJ (PERÍLIO, 2010) e é um modelo simplificado do domínio de comércio eletrônico, tendo sido construído por um bacharel em ciência da computação e atual mestrando do

ITA, possuindo mais de cinco anos de experiência em desenvolvimento de software. A seguir uma breve descrição do domínio apresentada no artigo é feita: "... uma categoria pode ter nenhuma ou muitas subcategorias. As categorias são compostas de produtos, que podem ter muitos produtos relacionados. Um item de carrinho de compras tem um produto. [...] Um pedido possui um ou muitos itens de carrinho de compras, pode possuir até um cupom de desconto, um endereço de entrega, dados de pagamento, status referente à condição atual do pedido e sempre pertence a um cliente." (PERÍLIO, 2010). A Figura 16 apresenta essa situação.

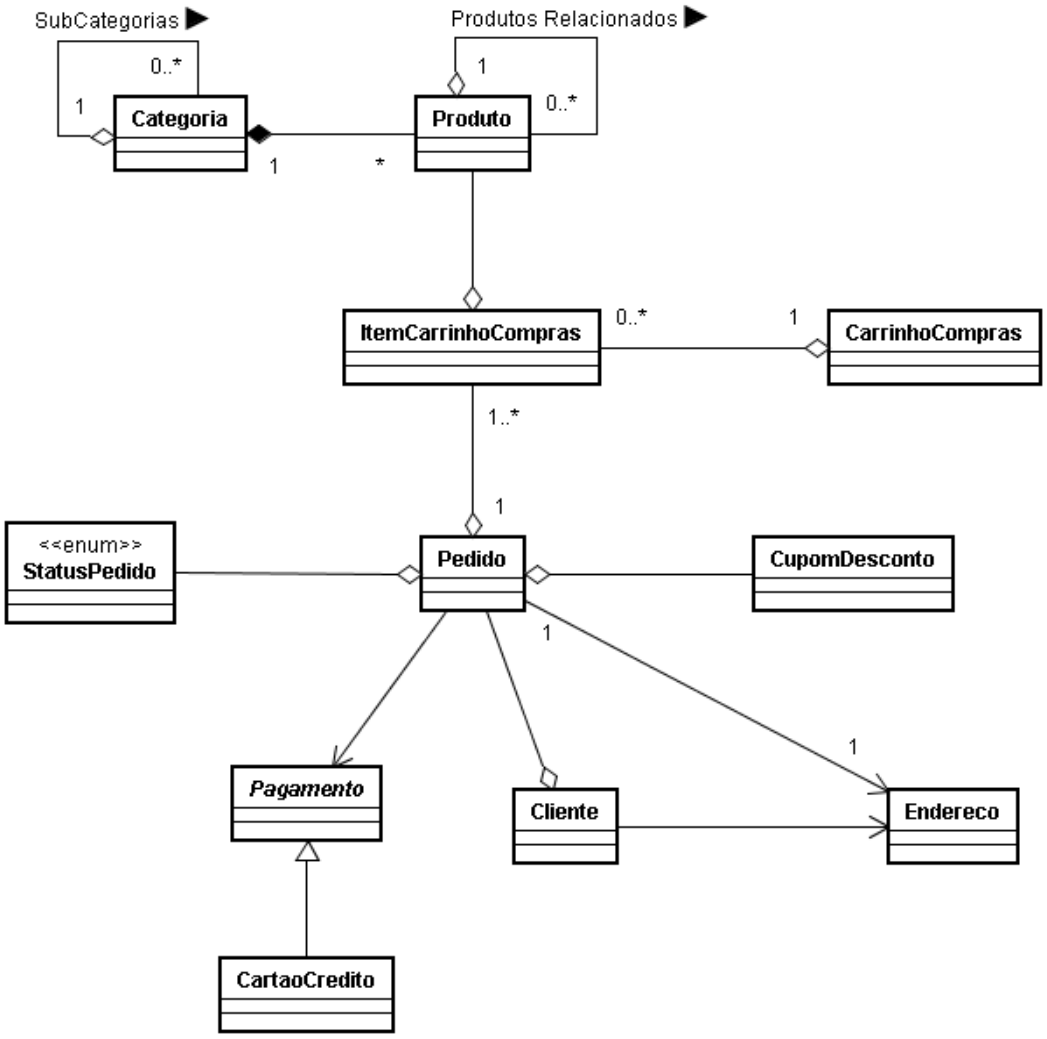


Figura 16: Representação original simplificada de um domínio de comércio eletrônico. Extraída e adaptada de Perílio (2010).

Baseado nesse diagrama de classes, a aplicação do procedimento PrOntoCon juntamente com o PrOntoCon Estendido teve início. Primeiramente, com a aplicação do PrOntoCon desenvolvido por Tavares (2008), identificou-se que Cliente deveria possuir uma superclasse (para classes estereotipadas como <<papel material>> deve existir uma superclasse, imediata ou não, classificada como <<tipo>>) e que a hierarquia entre Pagamento e CartaoCredito estava correta, incluindo a associação feita com Pedido (Pagamento foi estereotipado como <<tipo>> e CartaoCredito como <<quasi-tipo>>). Visto que a superclasse necessária à classe Cliente não existe no modelo representado na Figura 16, então foi necessária a sua criação, sendo a classe Pessoa. Com essas modificações realizadas, a continuação do processo, o PrOntoCon Estendido, pôde ter andamento.

No entanto, durante a aplicação do PrOntoCon Estendido, percebeu-se que para o caso desse domínio em discussão caberiam respostas diferentes para certas questões analisadas, o que gerou como resultado final duas propostas de modelagem distintas. Não há como se afirmar que uma é melhor que a outra, apenas são consequências de possíveis domínios diferentes para um mesmo problema, isto é, o domínio de comércio eletrônico para uma pequena empresa será diferente do de uma multinacional, assim como um cliente A pode pedir restrições que um cliente B não pediria para seu negócio. O que é importante destacar aqui é que o resultado final da aplicação do PrOntoCon Estendido **sempre** vai depender da resposta que o modelador fornecer para o seu domínio específico. Não é correto afirmar que todos os domínios de comércio eletrônico existentes possuem exatamente as mesmas restrições, pois elas podem variar de um caso para outro e isso vai influenciar o resultado do modelo final. Desse modo, as duas possíveis situações serão apresentadas de forma separada para não causar confusão sobre cada abordagem.

5.1.1. Caso 1

5.1.1.1. Aplicação da primeira atividade do PrOntoCon Estendido

Relembrando, a primeira atividade do PrOntoCon Estendido é a verificação de agregação/composição que é constituída por uma árvore de decisão para a identificação de estruturas parte-todo. Assim, cada par de classes foi analisado segundo o questionamento contido nesse diagrama. Entenda-se como pares de classes todos aqueles que já possuem algum relacionamento no diagrama original, acrescido de pares de classes sem relacionamentos e que estejam **dentro do contexto do domínio**, para que não só aconteça a verificação das relações existentes, mas também o acréscimo de alguma relação importante que possa ter passado despercebida. Detalhes de cada análise realizada são mostrados a seguir.

Classe Cliente como todo e classe Pedido como parte:

A primeira pergunta tenta identificar se há uma dependência genérica entre essas classes, ou seja, se o todo deve ter uma parte que pode ser substituída ao longo do tempo. A resposta dada foi **não** para a Questão I - Dependência Genérica pois, se for levado em consideração que o cadastro do cliente deve ficar armazenado, então o cliente continua sendo cliente mesmo sem fazer um pedido. Mesmo a resposta para a pergunta auxiliar dessa questão também foi **não** (se as partes deixam de existir quando o todo é destruído), visto que mesmo que um cliente não esteja executando uma ordem de compra (um pedido) num dado instante, seus pedidos anteriores podem continuar armazenados como uma forma de histórico para visualização do cliente.

Essa resposta negativa conduz para a Questão II - Dependência Específica, a qual também é respondido **não**, pois a substituição dos pedidos não modifica ou destrói o cliente. O resultado desta atividade foi rever se o conceito parte-todo se aplica, mas antes passando para a próxima atividade para verificar associações.

Classe Pedido como todo e classe ItemCarrinhoCompras como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, visto que os itens de um pedido podem ser substituídos. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **não**, já que não tem sentido existir um pedido sem itens. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango preenchido.

Classe CarrinhoCompras como todo e classe ItemCarrinhoCompras como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, se for considerado que um carrinho de compras deve ter itens que podem ser substituídos ao longo do tempo. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **sim**, pois um carrinho de compras pode existir mesmo sem itens, ou seja, seria um carrinho vazio. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango vazio.

Classe ItemCarrinhoCompras como todo e classe Produto como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, pois itens de carrinhos de compras são produtos que podem ser substituídos. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **não**, visto que se os produtos deixarem de existir, não há mais sentido em se referir a itens de carrinho de compras. Deve haver uma ligação explícita entre estes dois conceitos neste domínio. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango preenchido.

Classe Categoria como todo e classe Produto como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, se pensarmos que não tem sentido em se falar de uma categoria que não contenha pelo menos um produto relacionado, ou seja, sempre deve haver um produto em uma categoria. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **não**, senão

estaríamos caindo em contradição com o exposto anteriormente. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango preenchido.

Classe Categoria como todo e classe (Sub)Categoria como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, tendo em mente que se uma categoria for excluída, as suas subcategorias também deverão ser excluídas. Esta resposta leva para a Questão II - Existência do Todo sem Partes, a qual foi respondido **sim**, pois uma categoria não tem necessidade de possuir subcategorias para existir. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango vazio.

Classe Pedido como todo e classe CupomDesconto como parte:

A resposta para a Questão I - Dependência Genérica foi **não**, pois um pedido não tem a obrigatoriedade de possuir um cupom de desconto para sua finalização. Mesmo a resposta para a pergunta auxiliar desta questão foi **não**, visto que um cupom de desconto existe mesmo sem haver um pedido, já que o cliente opta por utilizá-lo ou não na sua compra.

Essa resposta negativa conduz para a Questão II - Dependência Específica, a qual também é respondido **não**, pois cupons de desconto não interferem na instância de pedido, somente reduzem o valor da compra (um atributo). O resultado desta atividade foi rever se o conceito parte-todo se aplica, mas antes passando para a próxima atividade para verificar associações.

Classe Pedido como todo e classe StatusPedido como parte:

A resposta para a Questão I - Dependência Genérica foi **não**, levando para a Questão II - Dependência Específica, a qual também é respondido **não**. Nesse caso, status de um pedido é somente um momento específico de um pedido em um dado instante, não cabendo o conceito parte-todo para essa situação. O resultado desta atividade foi rever se o conceito parte-todo se aplica, mas antes passando para a próxima atividade para verificar associações.

Classe Produto como todo e classe Produto como parte:

Embora a descrição do domínio não tenha esclarecido muito bem sobre a questão de produtos possuírem muitos produtos relacionados (MUNDOJ, 2010), será levado em consideração a ideia de que durante a compra online outros produtos podem ser sugeridos para aquele produto sendo pedido. Assim, a resposta para a Questão I - Dependência Genérica foi **não**, visto que produtos não necessitam de outros produtos para existirem, além do fato de que a exclusão de um produto não acarreta, necessariamente, a exclusão de outros produtos. Isso significa que produtos possuem vidas independentes.

Essa resposta negativa conduz para a Questão II - Dependência Específica, a qual também é respondido **não**, pois produtos podem ser substituídos sem afetar outros produtos, ou seja, uma sugestão de produtos interligados pode ser modificada posteriormente deixando de relacioná-los e mesmo assim as instâncias de produtos não serão afetadas. O resultado desta atividade foi rever se o conceito parte-todo se aplica, mas antes passando para a próxima atividade para verificar associações.

Classe Cliente como todo e classe Endereço como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, já que um cliente pode ter mais de um endereço de entrega cadastrado e pode substituí-los a qualquer momento. Esta resposta leva para a Questão II - Existência do Todo sem Partes, a qual foi respondido **não**, pois não há sentido em um cliente fazer um pedido sem um destino para sua entrega. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango preenchido.

Classe Pedido como todo e classe Endereço como parte:

Este relacionamento é importante quando se tem mais de um endereço de entrega cadastrado, escolhendo-o na hora do fechamento do pedido. Assim, a resposta para a Questão I - Dependência Genérica foi **não**, já que após o endereço ter sido escolhido, não é possível modificá-lo para a entrega do pedido. Esta resposta leva para a Questão II - Dependência Específica, a qual foi respondido **não**, pois não há sentido em destruição e

modificação do pedido por causa da mudança de endereço. O resultado desta atividade foi rever se o conceito parte-todo se aplica, mas antes passando para a próxima atividade para verificar associações.

Para essas análises feitas, o uso de estruturas compostas para os relacionamentos parte-todo obrigatório com losango preenchido não foi identificado, visto que não verificou-se nenhum caso em que partes de instâncias de classes diferentes interagissem entre si para um mesmo todo.

5.1.1.2. Aplicação da segunda atividade do PrOntoCon Estendido

Relembrando, a segunda atividade do PrOntoCon Estendido é a verificação de associações, em que possíveis casos de associações devem enquadrar-se em alguma categoria apresentada nesta atividade. Assim, esta atividade será realizada para aqueles pares de classes que não se encaixaram no conceito de estruturas parte-todo identificadas na primeira atividade do PrOntoCon Estendido. Detalhes da análise são apresentados a seguir.

Classe Cliente e classe Pedido:

Analisando sequencialmente as categorias, identificou-se que as classes Cliente e Pedido encaixaram-se no caso "A está relacionado a uma transação B", ou seja, um cliente está relacionado com um pedido que é uma transação. Assim, o relacionamento entre essas classes será uma associação.

Classe Pedido e classe CupomDesconto:

Novamente, analisando sequencialmente as categorias, identificou-se que as classes Pedido e CupomDesconto enquadraram-se no caso "A usa ou gerencia B", visto que um cupom de desconto será apenas usado pelo pedido para diminuir o valor da compra. Dessa forma, relacionamento entre essas classes também será uma associação.

Classe Pedido e classe StatusPedido:

Analisando sequencialmente as categorias, identificou-se que as classes Pedido e StatusPedido enquadraram-se no caso "A usa ou gerencia B", já que o pedido irá gerenciar o status do pedido para informar como está o andamento do mesmo. Assim, o relacionamento entre essas classes será uma associação.

Classe Produto e classe Produto:

Analisando sequencialmente as categorias, identificou-se que as classes Produto e Produto encaixaram-se no caso "A é adjacente a B", pois são classes de um mesmo tipo, ou seja, adjacentes conceitualmente. Portanto, relacionamento entre essas classes será uma associação.

Classe Pedido e classe Endereço:

Analisando sequencialmente as categorias, identificou-se que as classes Pedido e Endereço encaixaram-se no caso "A está relacionado a uma transação B", isto é, um endereço está relacionado com um pedido que é uma transação. Assim, o relacionamento entre essas classes será uma associação.

Nesse ponto, em que todos os pares de classes foram analisados tanto para relacionamentos de agregação/composição e de associação e não há nenhuma classe sem um relacionamento, a aplicação do PrOntoCon Estendido chega ao fim, fornecendo o diagrama de classes expresso na Figura 17.

A Figura 18 é apresentada para facilitar a visualização das diferenças de modelagem entre a representação original e a fornecida pelo PrOntoCon Estendido. As modificações ocorreram nos relacionamentos entre as classes Produto, ItemCarrinhoCompras, Pedido, StatusPedido, CupomDesconto, Cliente e Endereço.

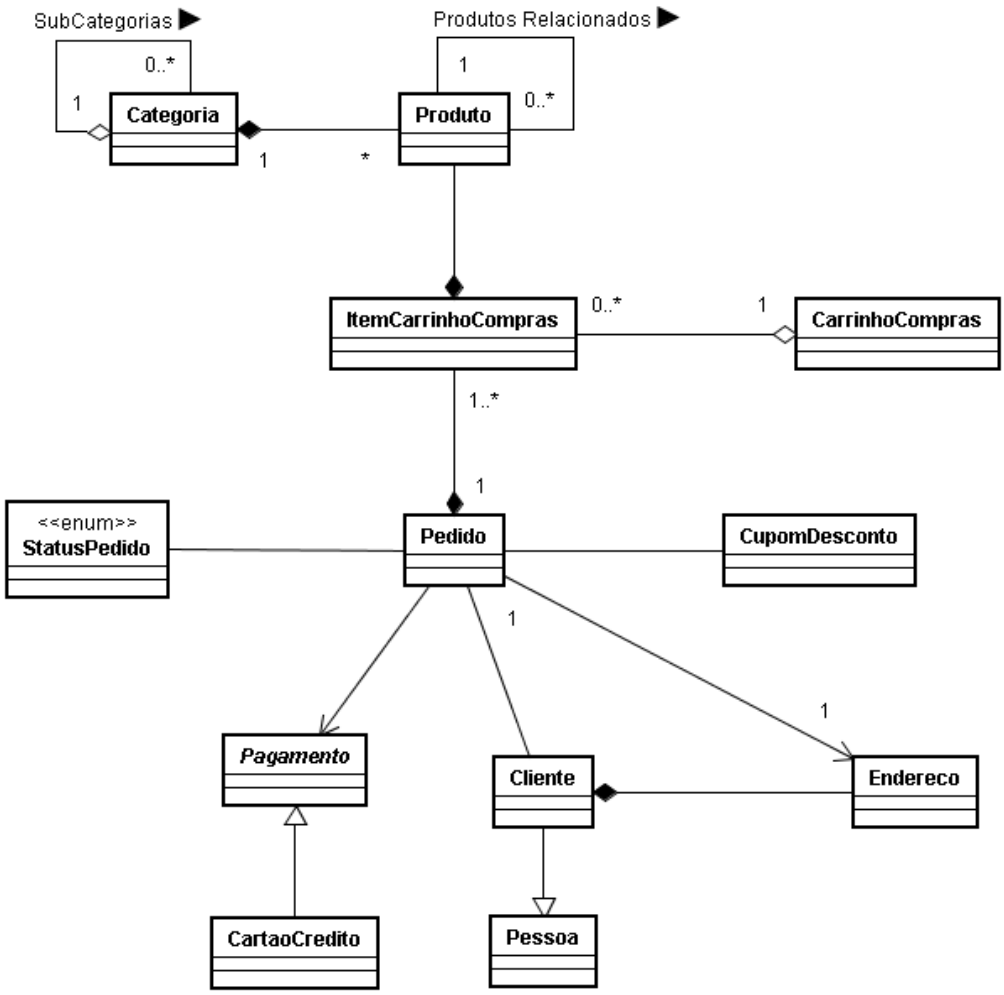


Figura 17: Diagrama de classes após a aplicação do PrOntoCon Estendido para a primeira análise do domínio (Caso 1).

As melhorias que podem ser evidenciadas com essa nova modelagem se referem à:

- Uma aproximação maior com a realidade do domínio estudado, já que conceitos desse domínio puderam ser explorados mais a fundo e, assim, compreendidos de forma mais clara;
- Indicar onde ocorrem as dependências de criação-destruição da parte em relação ao todo (e vice-versa), o que terá impacto, em termos de integridade referencial e caminhos de exclusão em cascata, na ligação entre os elementos de software (classes) e os elementos de banco de dados que representam o todo e as partes. A composição

identificada entre os relacionamentos das classes Pedido, ItemCarrinhoCompras e Produto evidencia, justamente, as dependências de criação-destruição entre elas. Isso significa que se um produto for excluído, o seu respectivo item de carrinho de compras também deverá ser. O raciocínio para pedido e itens é semelhante, pois se um pedido for destruído, então os itens de carrinho de compras alocados para aquele pedido deverão ser liberados.

- Esclarecer as restrições presentes no domínio com relação à dependência (representada por estruturas parte-todo) ou independência (representada por associações) da existência de uma instância de uma classe em relação a uma outra instância de outra classe, ou seja, se o tempo de vida delas estão vinculados ou não. Assim, é possível verificar neste exemplo que a alteração de relacionamentos de agregação entre as classes Produto, Cliente, CupomDesconto e StatusPedido para associação simples evidenciam a independência de suas existências, isto é, o tempo de vida das instâncias dessas classes não estão vinculados (e.g., um cupom de desconto pode ser criado antes do pedido, bem como ter validade após este ter sido concluído, caso o cupom não tenha sido utilizado. O pensamento para os outros casos é análogo). Também se enquadra nesse caso o relacionamento entre Cliente e Endereço, visto que não há sentido em haver endereços que não estejam vinculados a algum cliente.
- Facilitar a adição de novos papéis no futuro, além do Cliente, com a criação da classe Pessoa.

Assim, é possível perceber que tendo-se esses benefícios alcançados, os sistemas desse domínio se tornarão mais manuteníveis (com a integridade referencial melhorada), escaláveis (visto que os conceitos desse domínio foram mais claramente compreendidos) e robustos, o que proporcionará a facilidade de expansão do domínio bem como sua integração com outros domínios.

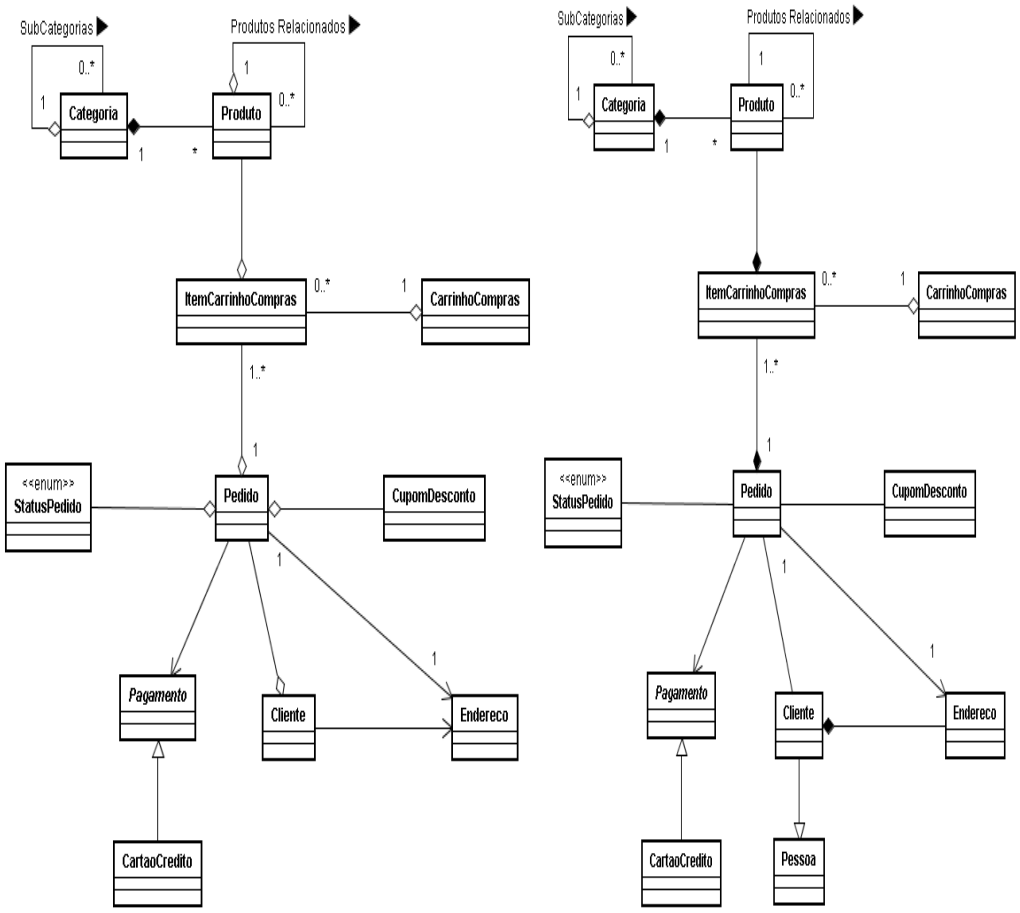


Figura 18: Comparação entre os diagramas de classes original (à esq.) e o sugerido pelo PrOntoCon Estendido (à dir.).

5.1.2. Caso 2

5.1.2.1. Aplicação da primeira atividade do PrOntoCon Estendido

Relembrando, a primeira atividade do PrOntoCon Estendido é a verificação de agregação/composição que é constituída por uma árvore de decisão para a identificação de estruturas parte-todo. Assim, cada par de classes foi analisado segundo o questionamento contido nesse diagrama. Detalhes de cada análise realizada são mostrados a seguir. Porém, só serão apresentados as relações entre os pares de classes que obtiveram respostas diferentes do Caso 1, já que o Caso 2 tem como objetivo mostrar uma segunda possibilidade de restrições do domínio. Dessa forma, deve-se

considerar a análise das relações restantes como idêntica ao explicitado na Seção 5.1.1.1.

Classe Cliente como todo e classe Pedido como parte:

A primeira pergunta tenta identificar se há uma dependência genérica entre essas classes, ou seja, se o todo deve ter uma parte que pode ser substituída ao longo do tempo. A resposta dada foi **sim** para a Questão I - Dependência Genérica, se for levado em consideração que cliente só é cliente se estiver executando um ou mais pedidos.

Essa resposta afirmativa conduz para a Questão II - Existência do Todo sem Partes, a qual também é respondido **sim**, pois um cliente pode ter seu cadastro armazenado mesmo sem estar fazendo pedidos em um dado instante. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango vazio.

Classe CarrinhoCompras como todo e classe ItemCarrinhoCompras como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, se for considerado que não há sentido em um carrinho de compras sem itens, ou seja, sempre deve haver pelo menos um item em um carrinho de compras. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **não**, senão estaríamos caindo em contradição com o exposto anteriormente, ou seja, nesse caso não cabe a ideia de um carrinho vazio. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango preenchido.

Classe Categoria como todo e classe Produto como parte:

A resposta para a Questão I - Dependência Genérica foi **não**, se pensarmos que uma categoria pode existir mesmo sem possuir algum produto relacionado a ela, ou seja, a relação entre categoria e produto não é uma obrigatoriedade. Mesmo a resposta para a pergunta auxiliar dessa questão também foi **não** (se as partes deixam de existir quando o todo é destruído), visto que se uma dada instância de categoria (uma classificação)

for excluída, os produtos que se relacionavam a ela não podem ser destruídos junto.

Esta resposta levou para a Questão II - Dependência Específica, a qual também foi respondido **não**, pois produtos podem mudar de categoria sem alterar as características desta última. O resultado desta atividade foi rever se o conceito parte-todo se aplica, mas antes passando para a próxima atividade para verificar associações.

Para essas análises feitas, o uso de estruturas compostas para os relacionamentos parte-todo obrigatório com losango preenchido não foi identificado, visto que não verificou-se nenhum caso em que partes de instâncias de classes diferentes interagissem entre si para um mesmo todo.

5.1.2.2. Aplicação da segunda atividade do PrOntoCon Estendido

Relembrando, a segunda atividade do PrOntoCon Estendido é a verificação de associações, em que possíveis casos de associações devem enquadrar-se em alguma categoria apresentada nesta atividade. Assim, esta atividade será realizada para aqueles pares de classes que não se encaixaram no conceito de estruturas parte-todo identificadas na primeira atividade do PrOntoCon Estendido. Detalhes da análise são apresentados a seguir. Porém, só serão apresentados as relações entre os pares de classes que obtiveram respostas diferentes do Caso 1, já que o Caso 2 tem como objetivo mostrar uma segunda possibilidade de restrições do domínio. Dessa forma, deve-se considerar a análise das relações restantes como idêntica ao explicitado na Seção 5.1.1.2, com exceção das classes Cliente e Pedido, visto que nesse segundo caso elas se enquadraram na análise de estruturas parte-todo.

Classe Categoria e classe Produto:

Analisando sequencialmente as categorias, identificou-se que as classes Categoria e Produto enquadraram-se no caso "A usa ou gerencia B", já que a categoria irá gerenciar a classificação dos produtos para

agrupar aqueles que tiverem características similares. Assim, o relacionamento entre essas classes será uma associação.

Nesse ponto, em que todos os pares de classes foram analisados tanto para relacionamentos de agregação/composição e de associação e não há nenhuma classe sem um relacionamento, a aplicação do PrOntoCon Estendido chega ao fim, fornecendo o diagrama de classes expresso na Figura 19.

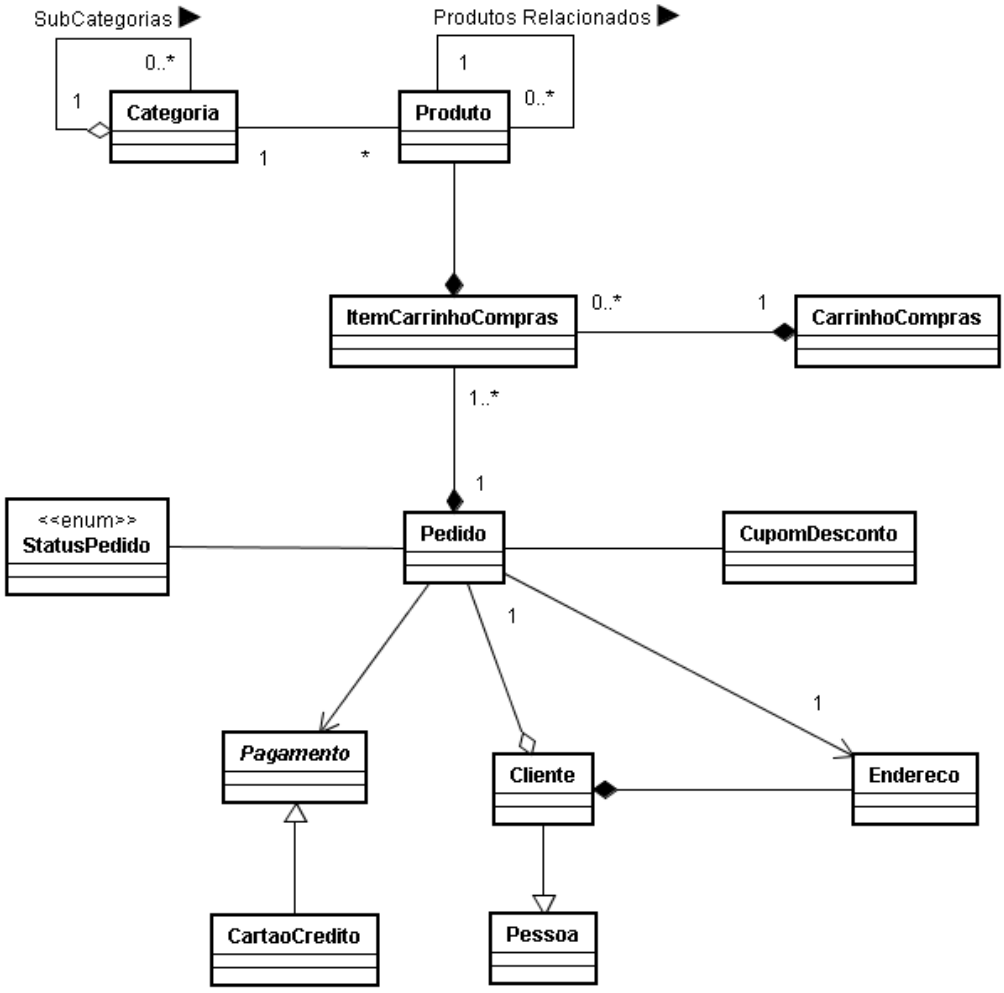


Figura 19: Diagrama de classes após a aplicação do PrOntoCon Estendido para a segunda análise do domínio (Caso 2).

A Figura 20 é apresentada para facilitar a visualização das diferenças de modelagem entre a representação original e a fornecida pelo PrOntoCon Estendido. As modificações ocorreram nos relacionamentos entre as classes

Produto, ItemCarrinhoCompras, Pedido, StatusPedido, CupomDesconto e Endereço. As melhorias que podem ser evidenciadas com essa nova modelagem, assim como os benefícios alcançados que irão refletir nos sistemas, são as mesmas discutidas ao final da Seção 5.1.1.2. As únicas diferenças dizem respeito à identificação de uma composição entre as classes CarrinhoCompras e ItemCarrinhoCompras, o que sugere uma dependência de criação-destruição entre elas; e à identificação de uma associação entre Produto e Categoria, o que indica que o tempo de vida dessas classes não estão vinculados.

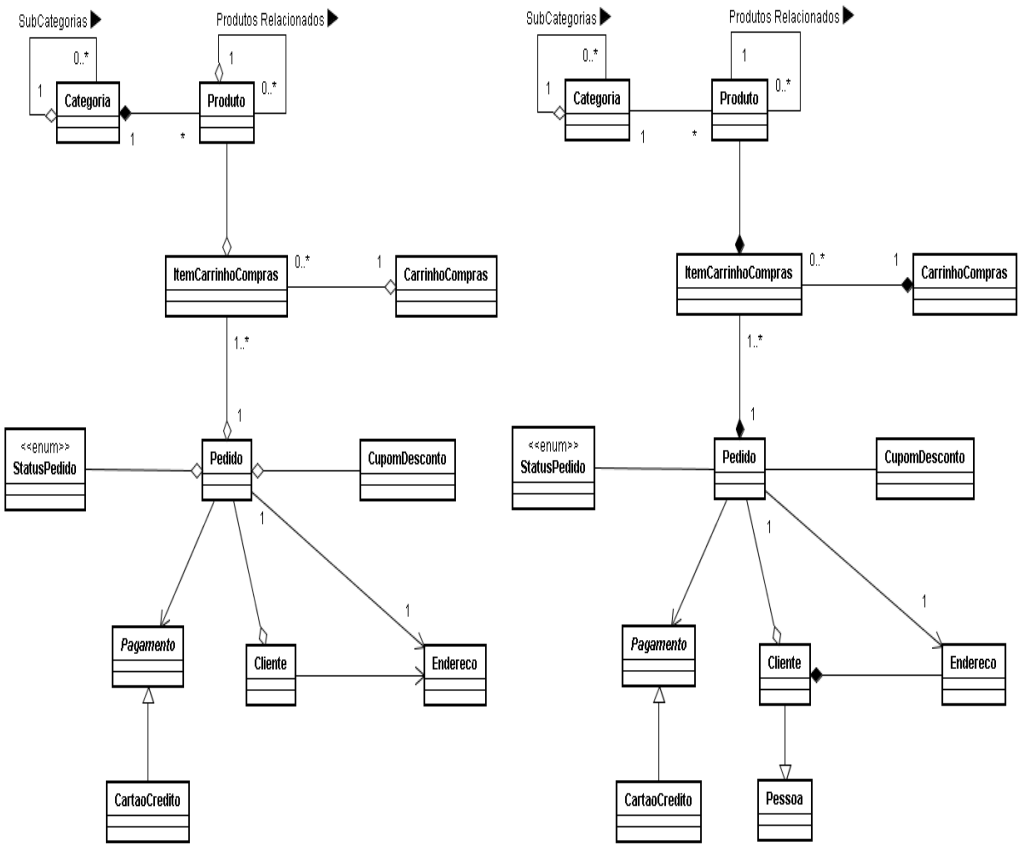


Figura 20: Comparação entre os diagramas de classes original (à esq.) e o sugerido pelo PrOntoCon Estendido (à dir.).

Entretanto, não há como se fazer uma comparação com respeito à corretude entre os diagramas fornecidos pelo PrOntoCon Estendido para o Caso 1 e para o Caso 2 (Figura 21), pois ambos podem ser considerados corretos do ponto de vista das restrições dos domínios que abordaram. Dessa forma, o resultado final do PrOntoCon Estendido **sempre** irá

depender das respostas fornecidas pelo modelador durante a análise do domínio em questão. O papel do PrOntoCon Estendido é guiar o modelador para facilitar essa tarefa, a fim de que os modelos gerados possam ser mais claros e fidedignos de acordo com o domínio do problema especificado.

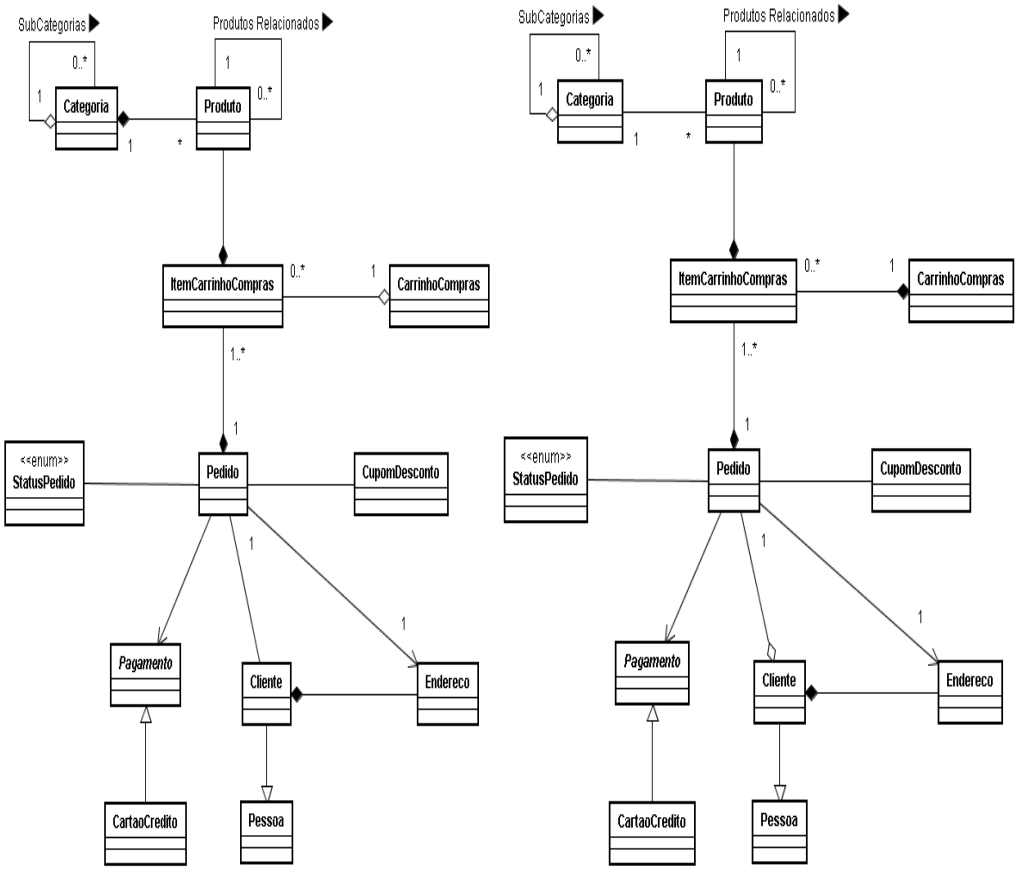


Figura 21: Comparação entre os diagramas de classes para o Caso 1 (à esq.) e para o Caso 2 (à dir.).

5.2. Domínio de Oficina Mecânica

O segundo exemplo de domínio apresentado para a aplicação do PrOntoCon Estendido é um domínio de controle de ordens de serviços em uma oficina mecânica. O desenvolvimento de softwares nos últimos anos só tem aumentado, incluindo os sistemas de controles de compras e serviços para supermercados, lojas, farmácias, restaurantes e muitos outros locais que, até pouco tempo, nem sonhavam com essa tecnologia. Atualmente, esses sistemas fazem parte da nossa realidade e a tendência é que, em um futuro próximo, todos os estabelecimentos comerciais informatizem seus serviços. É inegável o fato de que sistemas devidamente elaborados e construídos para satisfazer os requisitos de um determinado domínio podem trazer muitos benefícios em relação a controle de estoques, folha de pagamento, registro de serviços, cadastro de funcionários, etc. Assim, é natural que oficinas mecânicas também participem dessa inclusão tecnológica.

O exemplo utilizado foi retirado da revista SQL Magazine (SQL MAGAZINE, 2007), tendo sido construído por um doutorando em Engenharia de Sistemas e Computação. A seguir uma breve descrição do domínio apresentada no artigo é feita: "... um hipotético sistema para controle de ordem de serviços de veículos em uma oficina mecânica. Para isso, devem ser feitos os cadastros de clientes e de seus veículos. Sendo assim, um cliente poderá ter vários veículos cadastrados e todo veículo deverá pertencer a um cliente. Quando um veículo dá entrada na oficina, é aberta uma ordem de serviço sob a responsabilidade de um funcionário, contendo os serviços que serão executados no veículo do cliente. Desta forma, uma ordem de serviço é de responsabilidade de um funcionário, que pode abrir essas ordens de serviço. Além disso, uma ordem de serviço é de um único veículo e possui esses serviços. Serviços, assim como veículos, podem estar associados a essas ordens de serviço." (ARAÚJO, 2007). A Figura 22 apresenta essa situação.

Baseado nesse diagrama de classes, a aplicação do procedimento PrOntoCon juntamente com o PrOntoCon Estendido teve início. Primeiramente, com a aplicação do PrOntoCon, identificou-se que a hierarquia entre as classes Pessoa, Funcionário e Cliente já estava correta. Dessa forma, a continuação do procedimento com a aplicação do PrOntoCon Estendido pôde ter andamento. A descrição desta análise será apresentada nas próximas seções.

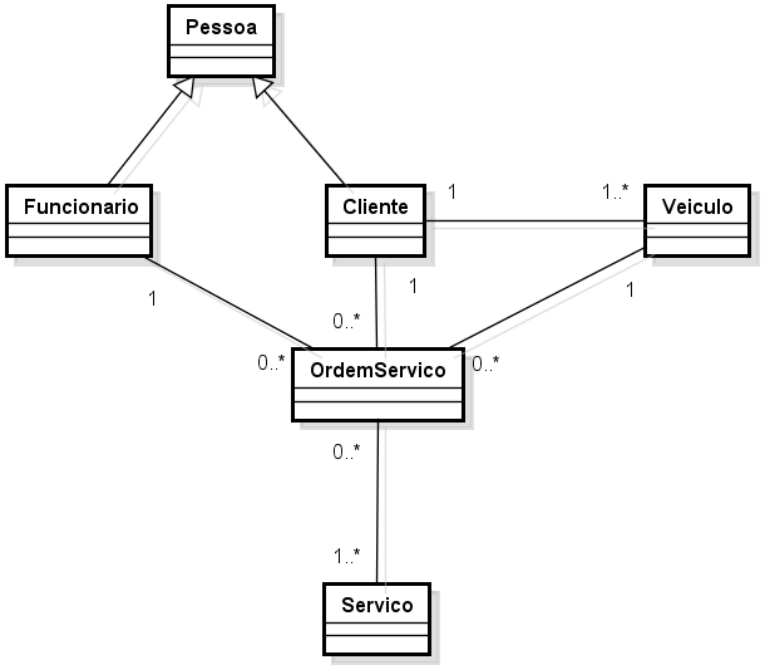


Figura 22: Representação original de um domínio de controle de ordens de serviços em uma oficina mecânica. Extraída e adaptada de Araújo (2007).

5.2.1. Aplicação da primeira atividade do PrOntoCon Estendido

Relembrando, a primeira atividade do PrOntoCon Estendido é a verificação de agregação/composição que é constituída por uma árvore de decisão para a identificação de estruturas parte-todo. Assim, cada par de classes foi analisado segundo o questionamento contido nesse diagrama. Detalhes de cada análise realizada são mostrados a seguir.

Classe Cliente como todo e classe Veiculo como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, pois uma das restrições do domínio é que todo veículo deve pertencer a um

cliente, ou seja, para ser cliente da oficina é necessário possuir pelo menos um carro cadastrado. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **não**, senão estaríamos caindo em contradição com o exposto anteriormente, ou seja, nesse caso não cabe a ideia de um cliente sem um carro. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango preenchido.

Classe Veiculo como todo e classe OrdemServico como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, se for levado em consideração que se o cadastro de um dado veículo for excluído, então todas as ordens de serviços relacionadas àquele veículo também deverão ser, pois não haveria sentido em se referenciar uma ordem de serviço sem a identificação do respectivo veículo. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **sim**, já que um veículo pode ficar sem uma ordem de serviço por um tempo indeterminado. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango vazio.

Classe Cliente como todo e classe OrdemServico como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, se for levado em consideração que se o cadastro de um cliente for excluído, então todas as ordens de serviços relacionadas àquele cliente também deverão ser, pois não haveria sentido em se referenciar uma ordem de serviço sem a identificação do respectivo cliente e, consequentemente, do seu veículo. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **sim**, já que um cliente pode ficar sem levar o seu veículo à oficina por um tempo indeterminado. O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango vazio.

Classe Funcionario como todo e classe OrdemServico como parte:

A resposta para a Questão I - Dependência Genérica foi **não**, se pensarmos que um funcionário pode existir mesmo sem possuir alguma

ordem de serviço relacionada a ele, ou seja, a relação entre funcionário e ordem de serviço não é uma obrigatoriedade. Mesmo a resposta para a pergunta auxiliar dessa questão também foi **não** (se as partes deixam de existir quando o todo é destruído), visto que se um funcionário for demitido enquanto sua ordem de serviço ainda estiver "em aberto", esta ordem de serviço não poderá ser excluída, apenas o funcionário responsável por ela deverá ser alterado.

Esta resposta levou para a Questão II - Dependência Específica, a qual também foi respondido **não**, pois a alteração das ordens de serviços não interferem nas características do funcionário. O resultado desta atividade foi rever se o conceito parte-todo se aplica, mas antes passando para a próxima atividade para verificar associações.

Classe OrdemServico como todo e classe Servico como parte:

A resposta para a Questão I - Dependência Genérica foi **sim**, pois uma ordem de serviço deve possuir um ou mais serviços relacionados à ela, descrevendo o que será feito no veículo do cliente. Esta resposta levou para a Questão II - Existência do Todo sem Partes, a qual foi respondido **não**, senão estaríamos caindo em contradição com o exposto anteriormente, ou seja, nesse caso não cabe a ideia de uma ordem de serviço sem a sua descrição (os serviços relacionados). O resultado desta atividade foi a construção de um relacionamento parte-todo obrigatório com o losango preenchido.

Para essas análises feitas, o uso de estruturas compostas para os relacionamentos parte-todo obrigatório com losango preenchido não foi identificado, visto que não verificou-se nenhum caso em que partes de instâncias de classes diferentes interagissem entre si para um mesmo todo.

5.2.2. Aplicação da segunda atividade do PrOntoCon Estendido

Relembrando, a segunda atividade do PrOntoCon Estendido é a verificação de associações, em que possíveis casos de associações devem enquadrar-se em alguma categoria apresentada nesta atividade. Assim, esta atividade será realizada para aqueles pares de classes que não se encaixaram no

conceito de estruturas parte-todo identificadas na primeira atividade do PrOntoCon Estendido. Detalhes da análise são apresentados a seguir.

Classe Funcionario e classe OrdemServico:

Analisando sequencialmente as categorias, identificou-se que as classes Funcionario e OrdemServico enquadraram-se no caso "A usa ou gerencia B", já que um funcionário gerencia uma ordem de serviço, ou seja, uma ordem de serviço é aberta por um determinado funcionário, de acordo com a descrição do domínio do problema. Assim, o relacionamento entre essas classes será uma associação.

Nesse ponto, em que todos os pares de classes foram analisados tanto para relacionamentos de agregação/composição e de associação e não há nenhuma classe sem um relacionamento, a aplicação do PrOntoCon Estendido chega ao fim, fornecendo o diagrama de classes expresso na Figura 23.

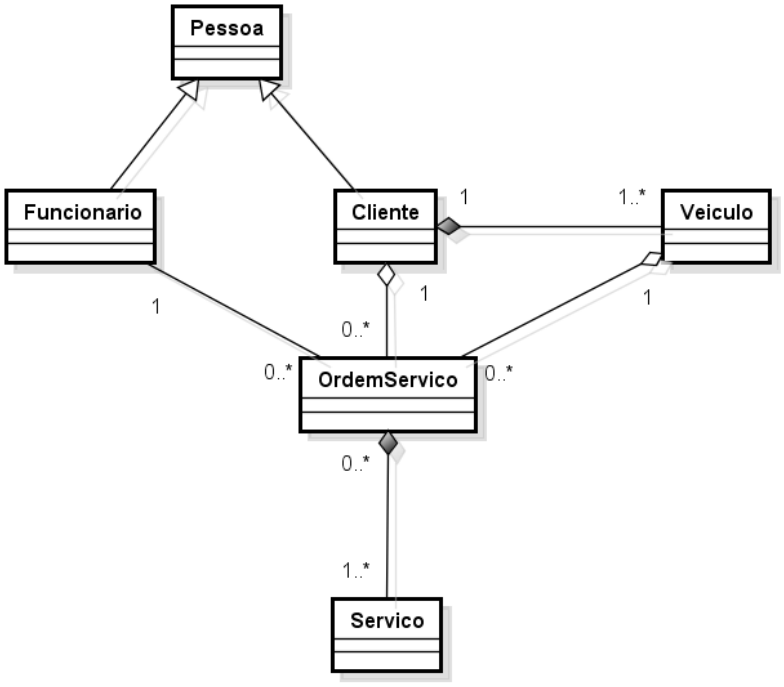


Figura 23: Diagrama de classes após a aplicação do PrOntoCon Estendido.

A Figura 24 é apresentada para facilitar a visualização das diferenças de modelagem entre a representação original e a fornecida pelo PrOntoCon

Estendido. As modificações ocorreram nos relacionamentos entre as classes Cliente, Veiculo, OrdemServico e Servico. As melhorias que podem ser evidenciadas com essa nova modelagem se referem à:

- Uma aproximação maior com a realidade do domínio estudado, visto que os conceitos desse domínio puderam ser compreendidos mais claramente;
- Indicar onde ocorrem as dependências de criação-destruição da parte em relação ao todo (e vice-versa), ou seja, melhorando a integridade referencial. A identificação de composição entre os relacionamentos das classes Cliente e Veiculo e OrdemServiço e Serviço evidencia uma dependência de criação-destruição entre elas, ou seja, se um cliente for excluído do cadastro, todos os veículos vinculados a ele também deverão ser; assim como um serviço não pode existir se não houver uma ordem de serviço especificada para ele.
- Esclarecer as restrições presentes no domínio com relação à vinculação do tempo de vida das instâncias das classes. A alteração dos relacionamentos de associação entre as classes Cliente, OrdemServiço e Veiculo para agregação mostra que as instâncias dessas classes possuem existências independentes, entretanto, em algum momento de suas vidas elas deverão se vincular para que o registro de uma ordem de serviço contenha todas as informações necessárias. Essa alteração também indica cautela quando for preciso fazer exclusões em cascata para que não se perca a integridade referencial.

Dessa forma, é possível perceber que ao se alcançar esses benefícios, os sistemas desse domínio poderão se tornar mais manuteníveis, escaláveis e robustos, já que questões importantes como a integridade referencial e o esclarecimento de restrições inerentes a esse domínio foram tratadas de forma menos intuitiva, pois contaram com a ajuda de um procedimento formal, o PrOntoCon.

Assim, com o fim da apresentação do estudo de casos, torna-se evidente o suporte formal que o PrOntoCon Estendido proveu durante a verificação (e, posteriormente, a validação) dos relacionamentos dos diagramas de classes dos domínios aqui mostrados. O PrOntoCon Estendido possibilitou um estudo mais aprofundado sobre o significado de cada conceito do domínio, obrigando o modelador a pensar realmente sobre as razões e justificativas que o levaram a modelar de uma forma e não de outra, pois ele precisa dessas respostas para poder percorrer a árvore de decisão do procedimento. A vantagem que o modelador obtém após a aplicação do PrOntoCon Estendido é o fato de poder descobrir restrições que ainda se encontravam obscuras; entender melhor o porquê de uma agregação ou composição, ao invés de uma associação (e vice-versa), representar de forma mais conveniente aquele tipo de relação que ele pretende armazenar; usar de forma mais consciente as estruturas parte-todo, entendendo de forma mais clara o que significa ter tempos de vida de instâncias de classes vinculados ou não. Além de poder ter o contato com a sugestão do uso de estruturas compostas, um importante acréscimo feito à UML 2.

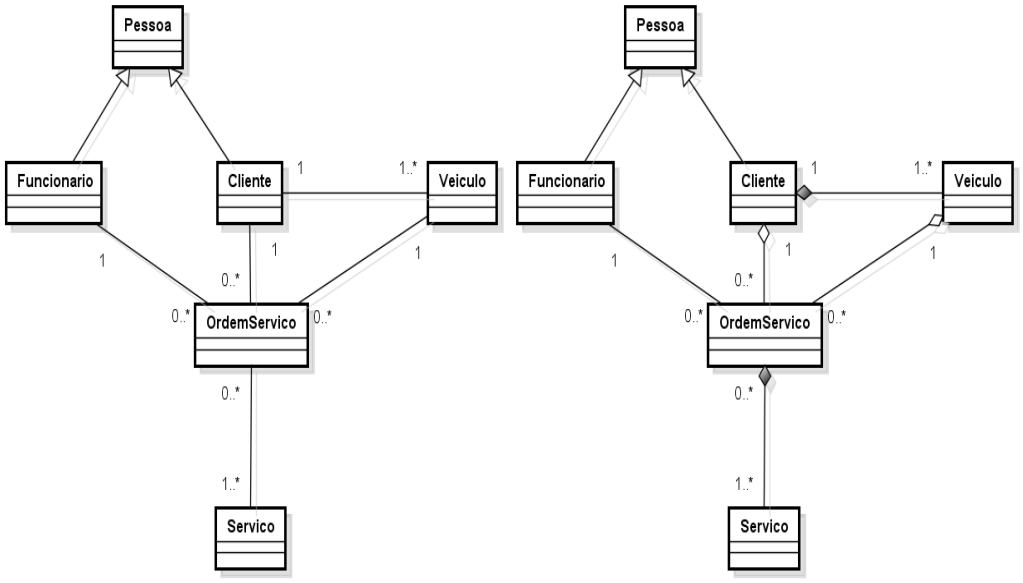


Figura 24: Comparação entre os diagramas de classes original (à esq.) e o sugerido pelo PrOntoCon Estendido (à dir.).

6. Conclusões

Ainda hoje a dificuldade dos desenvolvedores de software para construir um modelo conceitual fiel à realidade é visível. Muitas vezes os modeladores não conseguem expressar as ideias contidas em suas mentes de forma clara e correta, culminando em vários problemas durante as etapas do desenvolvimento do software e até mesmo após a sua entrega, produzindo gastos extras para corrigi-los e que, muitas vezes, não estavam planejados. Uma situação incômoda e indesejada, tanto para os colaboradores da organização quanto para os clientes, podendo causar o abalo da confiança destes últimos nos primeiros. Assim, a criação de métodos e procedimentos que tenham como objetivo unir a prática com a teoria tendem a diminuir cada vez mais esses inconvenientes.

No caso específico deste trabalho, ligado às etapas inicias do desenvolvimento de software, a complexidade das análises ontológicas de domínio foi quebrada para que qualquer modelador de diagramas de classes UML pudesse obter os benefícios de se usar um procedimento formal para a definição dos relacionamentos de agregação e composição e de associação simples, que costumam gerar muitas dúvidas durante a modelagem. Assim, este trabalho teve como resultado a criação do PrOntoCon Estendido, feito em SPEM, que de uma forma muito interativa consegue guiar o modelador durante a identificação de relacionamentos parte-todo e de associação. Como resultado final da aplicação do procedimento, o modelador terá um diagrama de classes UML validado, ou seja, um modelo mais coerente com as restrições do domínio do problema e que esteja apoiado no formalismo das análises ontológicas.

Como foi possível perceber durante o desenvolvimento do presente trabalho, as relações entre partes e todos costumam gerar muitas controvérsias, sendo, assim, investigadas sob diferentes aspectos por vários pesquisadores atualmente. Além do fato de que o uso da análise ontológica na modelagem conceitual de um domínio, com o intuito de ajudar a

esclarecer o uso de todos os possíveis tipos de relacionamentos de uma forma mais correta, vem ganhando muitos adeptos e várias contribuições que comprovam a eficácia do método. Artale e Keet (2008) afirmam que essas contribuições, mais especificamente as ligadas aos relacionamentos parte-todo, resultam, além de avanços significativos para a área, em uma melhor identificação dos problemas inerentes à modelagem conceitual.

Complementando os benefícios trazidos por essa abordagem para os modelos conceituais, Keet e Artale (2008) asseguram que uma análise conceitual mais precisa e detalhada e que leva em conta as noções ontológicas sobre as relações parte-todo irá aumentar a qualidade do modelo conceitual elaborado, além de reduzir erros durante o desenvolvimento e, assim, economizar recursos durante a fase de teste, resultando em um melhor produto. Larman (2004) também concorda que a representação dos relacionamentos de agregação e composição esclarecem as restrições existentes no domínio referentes à dependência ou independência da parte em relação ao todo (e vice-versa), ou seja, se possuem tempos de vida distintos. Esse fato terá impacto sobre as fases sucessoras do desenvolvimento do projeto se não for devidamente explicitado, visto que as classes conceituais do domínio serão mapeadas para elementos de banco de dados que irão representar o todo e as partes e as dependências incorretas de criação-destruição desses elementos podem causar problemas em relação a integridade referencial e a caminhos de exclusão em cascata. Todas essas vantagens acabam refletindo diretamente sobre a confiabilidade, escalabilidade e manutenção dos sistemas, o que os tornam mais robustos.

Através dos estudos de casos, foi possível perceber a melhoria dos modelos fornecidos pelo uso do PrOntoCon Estendido com relação a todos esses benefícios supracitados sobre a representação dos relacionamentos de agregação e composição na modelagem conceitual de domínios, visto que o desenvolvimento do PrOntoCon Estendido teve como objetivo principal o uso da análise ontológica para a criação de um procedimento prático. Em adição, o procedimento também foi contemplado com a análise dos relacionamentos de associações, além de fornecer a possibilidade do uso de estruturas compostas na modelagem, um importante acréscimo feito

à UML 2.0, o que garante a atualidade do trabalho realizado. Dessa maneira, chega-se a conclusão de que a hipótese do trabalho desta pesquisa foi plenamente satisfeita.

Portanto, espera-se que o uso efetivo do PrOntoCon gere diagramas de classes mais claros e confiáveis, evitando-se, assim, os vários transtornos causados por uma análise de domínio mal interpretada.

6.1. Trabalhos Futuros

Uma ferramenta para automatização de partes do PrOntoCon foi desenvolvida por Oliveira e Biasutti, citados em Tavares (2008), com o objetivo de conduzir o modelador de forma mais simples e prática durante o processo de modelagem. Dessa forma, assim como o procedimento PrOntoCon foi ampliado para abranger os relacionamentos de agregação/composição e de associações, gerando o PrOntoCon Estendido, essa ferramenta desenvolvida também deve ser ampliada para acomodar as adições realizadas.

Comprovar a eficácia da utilização tanto do PrOntoCon quanto do PrOntoCon Estendido por meios estatísticos. Para executar tal tarefa, seria necessário o estabelecimento de métricas e critérios objetivos de avaliação de uma comunidade de desenvolvedores de software, com o intuito de capturar todos os aspectos a serem analisados sobre os resultados obtidos dos diagramas produzidos com o uso do procedimento e sem o uso deste.

Apêndice A – PrOntoCon Estendido: Primeira Atividade

O procedimento PrOntoCon foi construído utilizando-se a ferramenta Microsoft Visio 2003, que permitiu gerar uma sequência de páginas *web* representando-se, assim, a sequência dos passos a serem seguidos durante o procedimento. Duas observações são importantes para o entendimento da ligação entre as páginas: (i) cada guia existente nessas páginas possui um link e (ii) cada nó da árvore de decisão ou cada suporte da tabela de associações é um link que direciona para sua página relacionada.

Neste anexo é apresentado cada uma das páginas referentes à primeira atividade do PrOntoCon Estendido: verificação de agregação/composição, além da página inicial do procedimento (Figura A.1).

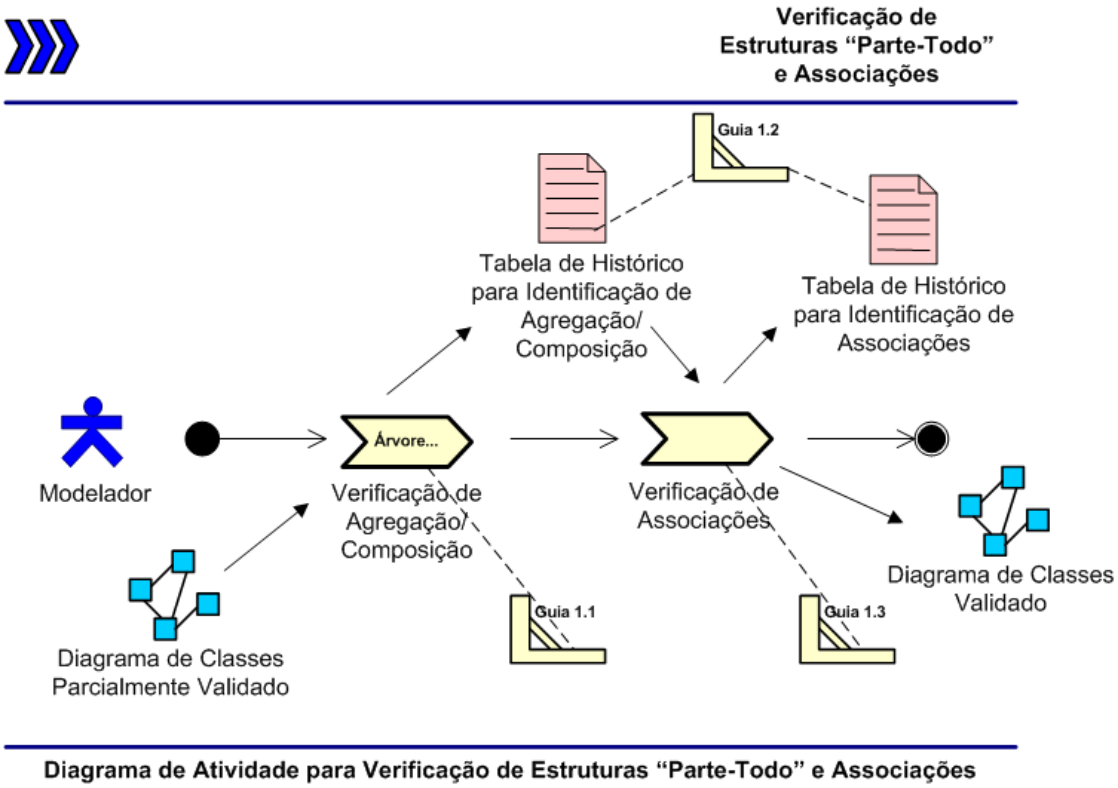


Figura A. 1: Diagrama de Atividades do procedimento PrOntoCon Estendido (tela inicial da nova fase).

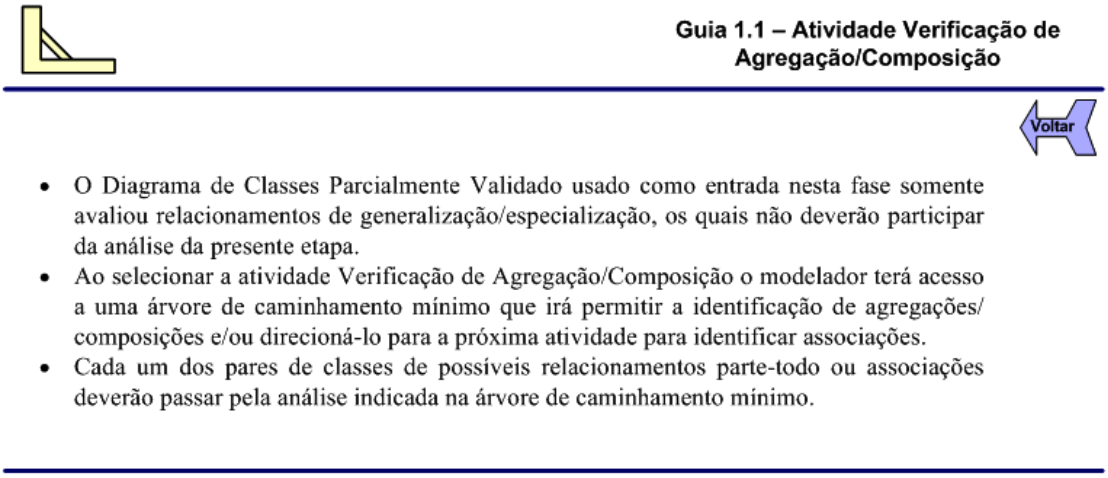
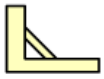


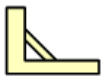
Figura A. 2: Guia 1.1 – Atividade Verificação de Agregação/Composição.



Guia 1.2 – Tabela de Histórico para Identificação de Agregação/Composição

Voltar

Classe Todo	Classe Parte	Há dependência genérica? Sim/Não.	Há existência do todo sem partes? Sim/Não.	Há dependência específica? Sim/Não.	Representação UML correspondente. Losango vazio/preenchido.	Há possibilidade de estruturas compostas? Sim/Não.



Guia 1.2 – Tabela de Histórico para Identificação de Associações

Classe A	Classe B	Categoria	Sim/Não
		A está fisicamente contido em B.	
		A usa ou gerencia B.	
		A se comunica com B.	
		A está relacionado a uma transação B.	
		A é uma transação relacionada com uma outra transação B.	
		A é adjacente a B.	

Figura A. 3: Guia 1.2 – Tabelas de Histórico para Identificação de Agregação/Composição e de Associações.

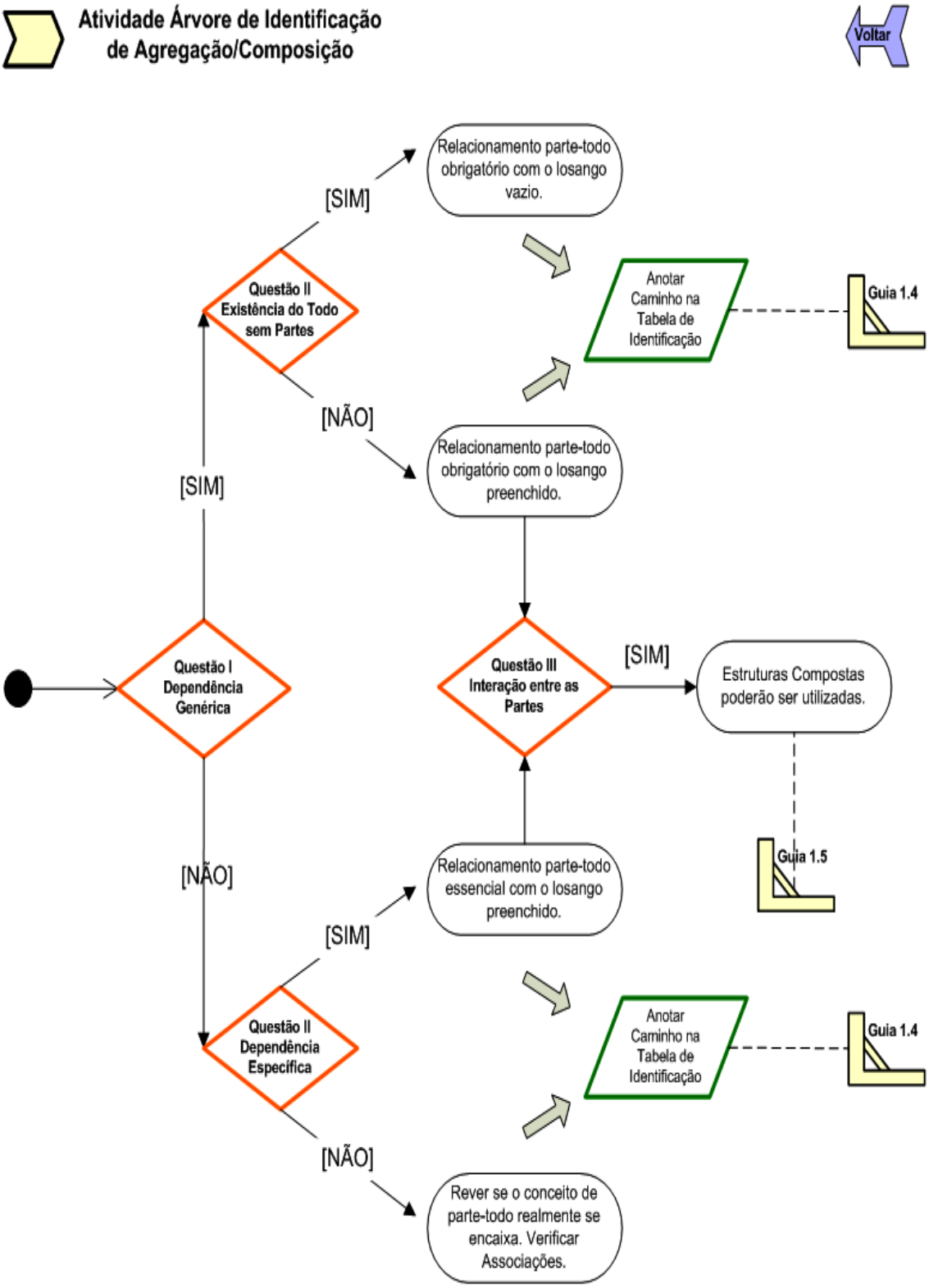


Figura A. 4: Atividade Árvore de Identificação de Agregação/Composição.

Questão I – Dependência Genérica

Pergunta:

Há dependência genérica entre a parte-todo, ou seja, o todo deve ter uma parte que pode ser substituída ao longo do tempo? Ou ainda, se o todo for destruído as partes também deixarão de existir? Exemplo: na estrutura parte-todo FUNÇÕES-PROGRAMA podemos dizer que um PROGRAMA é composto por FUNÇÕES que podem ser substituídas ao longo do tempo; assim como em POLTRONA-AVIÃO podemos verificar que um AVIÃO possui POLTRONAS substituíveis e que deixarão de existir caso o avião sofra um acidente grave.

Suporte para análise da pergunta:

i. **Outros exemplos:**
   - (SUB)CATEGORIA-CATEGORIA: uma categoria (e.g. de produtos) pode ter subcategorias, porém, se uma categoria for excluída do registro, todas as subcategorias que se relacionavam com ela não têm mais o porquê de existir.
   - FUNCIONÁRIO-EMPRESA: uma empresa deve possuir funcionários que podem ser substituídos ao longo do tempo por alguma razão.
   - CORAÇÃO-HUMANO: devido aos transplantes, um humano pode ter o órgão coração substituído por outro.
   - ITEMCARRINHO-PEDIDO: para um pedido de compras ser efetivado ele deve possuir itens, e os itens de um pedido podem ser substituídos ao longo do tempo até o fechamento da compra.

ii. **Contra exemplos:**
   - CÉREBRO-HUMANO: o cérebro não pode ser substituído sem ao menos causar algum dano ao humano.
   - VASO-VIDRO: um vaso é constituído de vidro, porém, se ele vier a quebrar, suas partes não poderão ser substituídas sem causar alteração no vaso original.
   - CUPOMDESCONTO-PEDIDO: o uso de um cupom de desconto é opcional, ou seja, o cliente utiliza-o se quiser e/ou se possuir. Assim, mesmo que o cliente possua um cupom de desconto, mas opta por não usá-lo no fechamento do pedido, podemos verificar que tanto o cupom quanto o pedido têm vidas independentes, o que não caracteriza uma dependência genérica.

iii. A pergunta "se o todo for destruído as partes também deixarão de existir?" é uma pergunta auxiliar para guiar o modelador na identificação de uma dependência genérica. Não significa que as duas perguntas mencionadas acima devam ser afirmativas para passar para a próxima pergunta da árvore, basta que uma delas seja verdadeira.

Figura A. 5: Questão I – Dependência Genérica.

Questão II – Existência do Todo sem Partes

Pergunta:

Porém, se as partes deixarem de existir, o todo continua existindo? Exemplo: na estrutura parte-todo ASA-XÍCARA podemos dizer que se a asa da xícara quebrar, a xícara continua existindo.

Suporte para análise da pergunta:

i. **Outros exemplos:**

- POLTRONA-AVIÃO: mesmo que as poltronas de um avião sejam retiradas ele continua existindo.
- FUNÇÕES-PROGRAMA: podemos dizer que um programa existe mesmo sem funções.
- MÃOS-HUMANO: um humano pode sobreviver apenas com uma ou sem nenhuma mão.
- CIDADE-PAÍS: mesmo que todas as cidades de um país sejam extintas, o conceito de país ou a sua delimitação continua existindo.
- (SUB)CATEGORIA-CATEGORIA: uma categoria não tem necessidade de possuir subcategorias para existir.

Obs.: Encaixam-se nesse perfil relacionamentos estruturais (asa-xícara, poltrona-avião), funcionais (funções-programa, mãos-humano) e espaciais (cidade-país, oásis-deserto).

ii. **Contra exemplos:**

- JOGADOR-TIME: se não houver jogadores, o time deixa de existir.
- PESSOA-FESTA: não há sentido em uma festa sem pessoas.
- CORAÇÃO-HUMANO: um humano não sobrevive sem um coração.
- PRODUTO-ITEMCARRINHO: podemos dizer que itens de um carrinho de compras estão associados a produtos (no sentido de que são compostos por produtos), porém, se os produtos deixarem de existir, não há mais sentido em se falar de itens de carrinho de compras sem esses produtos correspondentes.

Obs.: Encaixam-se nesse perfil relacionamentos de agrupamento (jogador-time), de participação (pessoa-festa, enzima-reação) e funcionais (coração-humano).

Figura A. 6: Questão II – Existência do Todo sem Partes.

Questão II – Dependência Específica

Pergunta:

Há dependência específica entre a parte-todo, ou seja, as partes não podem ser substituídas sem destruir ou modificar o todo? Exemplo: na estrutura parte-todo CÉREBRO-HUMANO podemos dizer que o cérebro não pode ser substituído sem que o humano morra ou fique visivelmente debilitado.

Suporte para análise da pergunta:

i. **Outros exemplos:**

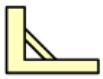
- VASO-VIDRO: um vaso é constituído de vidro, porém, se ele vier a quebrar, suas partes não poderão ser substituídas sem causar alteração no vaso original.
- PEDAÇO DE BOLO: um pedaço de bolo é uma parte de um bolo inteiro. Assim, a substituição dessas partes por pedaços de outros bolos ou simplesmente a inversão de ordem desses pedaços não poderá ser feita sem a modificação visível do bolo inteiro original.

Obs.: Encaixam-se nesse perfil relacionamentos funcionais (cérebro-humano), de constituição (vaso-vidro, bicicleta-aço) e sub-quantificados (taça de vinho, pedaço de bolo).

ii. **Contra exemplos:**

- FUNCIONÁRIO-EMPRESA: uma empresa deve possuir funcionários que podem ser substituídos ao longo do tempo por alguma razão.
- CORAÇÃO-HUMANO: devido aos transplantes, um humano pode ter o órgão coração substituído por outro.

Figura A. 7: Questão II – Dependência Específica.



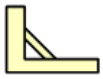
Questão III - Interação entre as Partes

Voltar

Questão III – Interação entre as Partes

Pergunta:
Há interação entre as partes, ou seja, é possível que uma parte interfira ou controle o funcionamento de outra? Exemplo: um CARRO possui como partes RODAS e MOTOR, sendo que o MOTOR irá acionar as RODAS.
Suporte para análise da pergunta:
i. Outros exemplos: • MOTOR-PROPULSOR-BARCO: um barco tem como partes constituintes um motor e propulsor (es), sendo que o motor será responsável por acionar o funcionamento do(s) propulsor (es). • FUNCIONÁRIO-EQUIPAMENTO-EMPRESA: uma empresa precisa de funcionários e equipamentos para funcionar, onde um equipamento será operado por funcionários, seja ele uma pessoa específica ou um grupo de pessoas. **ii. Contra exemplos:** • VOLANTE-BANCO-CARRO: um carro possui um volante e bancos dianteiros e traseiros, porém, estes últimos não interferem de forma alguma no funcionamento do volante e vice-versa.

Figura A. 8: Questão III – Interação entre as Partes. Pergunta utilizada para descobrir se o uso de estruturas compostas da UML 2.0 se encaixa.



Guia 1.4 – Instruções para Anotações de Identificação na Tabela

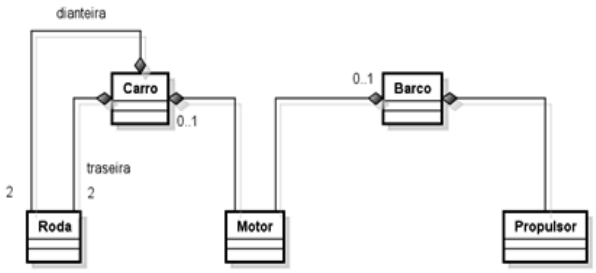
Voltar

- Anotar na Tabela de Histórico para Identificação de Agregação/Composição as decisões tomadas durante o caminhamento da árvore e as informações relevantes, como mostrado no exemplo abaixo.

Classe Todo	Classe Parte	Há dependência genérica? Sim/Não.	Há existência do todo sem partes? Sim/Não.	Há dependência específica? Sim/Não.	Representação UML correspondente. Losango vazio/preenchido.	Há possibilidade de estruturas compostas? Sim/Não.
Pedido	ItemCarrinho	Sim	Não	-	Losango preenchido	Não
Pedido	CupomDesconto	Não	-	Não	Verificar associação	-

Figura A. 9: Guia 1.4 – Instruções para Anotações na Tabela de Histórico.

- Neste exemplo, um carro possui motor, 2 rodas traseiras e 2 rodas dianteiras; assim como o barco possui motor e propulsor(es). Contudo, não é possível expressar corretamente o relacionamento entre motor e rodas dianteiras, significando que somente o motor de um carro (e não de um barco) as aciona. O pensamento é análogo para o caso do barco.



- Por esta razão, modelar a situação acima utilizando estruturas compostas da UML 2 facilita a compreensão do modelo do domínio, além de eliminar as ambiguidades comentadas anteriormente.

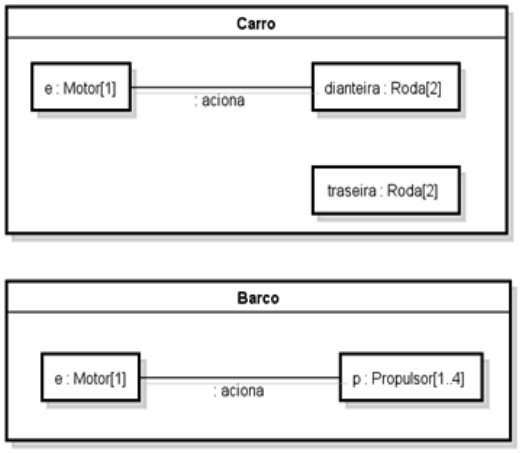


Figura A. 10: Guia 1.5 – Aplicação de Estrutura Composta.

Apêndice B – PrOntoCon Estendido: Segunda Atividade

Neste anexo é apresentado cada uma das páginas referentes à segunda atividade do PrOntoCon Estendido: verificação de associações, além da página inicial do procedimento (Figura B.1).

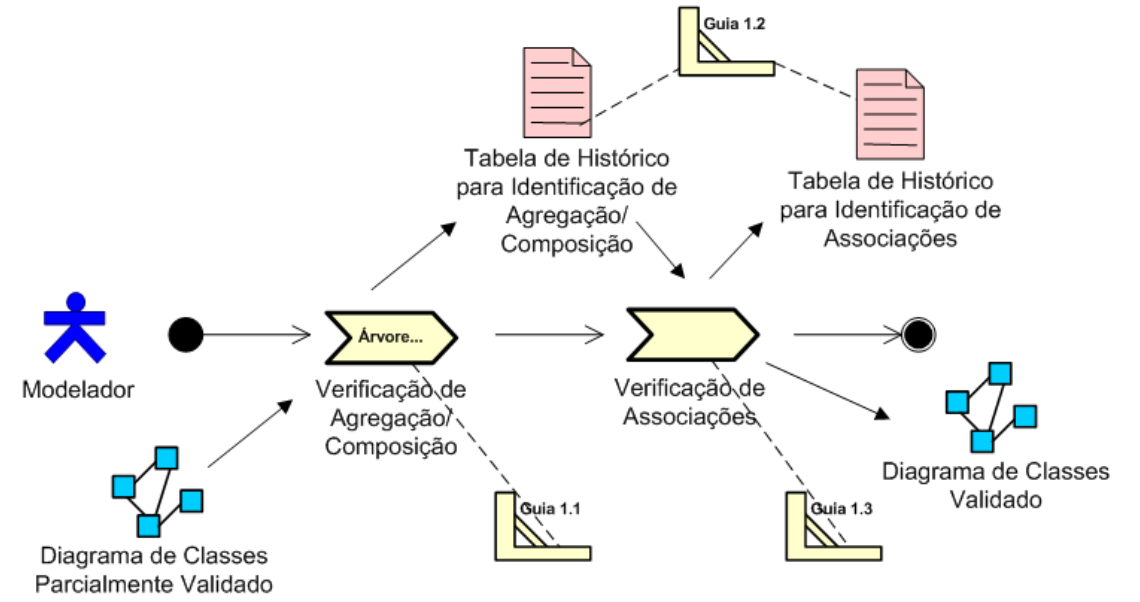
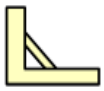


Figura B. 1: Diagrama de Atividades do procedimento PrOntoCon Estendido (tela inicial da nova fase).



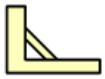
Guia 1.3 – Atividade Verificação de Associações

- Verifique se os pares de classes que não se encaixaram como Agregação/Composição irão se encaixar como Associação (utilize a Tabela de Histórico para Identificação de Agregação/Composição como referência).
- Para esta tarefa, identifique se cada um dos pares de classes se enquadram em uma das categorias abaixo e anote na Tabela de Histórico para Identificação de Associações. Cada categoria possui exemplos extraídos dos domínios de Lojas e Reservas de Passagens Aéreas. Se somente os exemplos não forem suficientes para elucidar a identificação, clique no suporte correspondente para obter um detalhamento maior sobre a questão.

Categoria	Exemplos	Suporte
A está fisicamente contido em B.	Registro-Loja, Item-Prateleira, Passageiro-Aeronave	1
A usa ou gerencia B.	Caixa-Registro, Piloto-Aeronave	2
A se comunica com B.	Cliente-Caixa, AgenteDeReservas-Passageiro	3
A está relacionado a uma transação B.	Cliente-Pagamento, Passageiro-Bilhete	4
A é uma transação relacionada com uma outra transação B.	Pagamento-Venda, Reserva-Cancelamento	5
A é adjacente a B.	Produto-Produto, Cidade-Cidade	6

Obs.: Se ainda sobrarem classes sem relacionamentos, é provável que as etapas para verificação de Agregação/Composição e Associações tenham que ser refeitas. Se ainda assim o possível erro não tiver sido encontrado, então o modelador deverá refazer o processo inteiro, desde a verificação de Generalização/Especialização.

Figura B. 2: Guia 1.3 – Atividade Verificação de Associações.

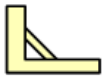


Suporte 1

Suporte 1

Pergunta:
A classe A está fisicamente contida na classe B? Exemplo: entre ITEM e PRATELEIRA podemos dizer que itens estão fisicamente contidos em alguma prateleira da loja. Portanto, encaixam-se nessa categoria e formam uma associação simples. Não são classificadas como agregação/composição porque não possuem dependência genérica (pode haver uma prateleira vazia e a existência de itens e prateleira são independentes) e nem dependência específica (a substituição dos itens não modifica a prateleira).
Suporte para análise da pergunta:
i. Outros exemplos: • REGISTRO-LOJA: o registro em papel de uma venda pode estar fisicamente contido em algum arquivo da loja, formando, assim, uma associação. Não são classificadas como agregação/composição porque não possuem dependência genérica (os registros não precisam necessariamente ser guardados na loja física e o fechamento da loja não implica na destruição dos registros) e nem dependência específica (a substituição dos registros não modifica ou destrói a loja). • PASSAGEIRO-AERONAVE: o passageiro está inserido naquela aeronave para aquele determinado vôo, caracterizando uma associação. Não são classificadas como agregação/composição porque não possuem dependência genérica (a aeronave pode estar vazia quando não há vôos programados e aeronave e passageiro têm existências independentes) e nem dependência específica (a substituição dos passageiros não implica na modificação da aeronave).

Figura B. 3: Suporte 1 – para categoria “A está fisicamente contido em B”.

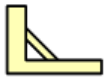


Suporte 2

Suporte 2

Pergunta:
A classe A usa ou gerencia a classe B? Exemplo: entre PILOTO e AERONAVE podemos dizer que o piloto gerencia a aeronave durante o vôo. Portanto, encaixam-se nessa categoria e formam uma associação simples. Não são classificadas como agregação/composição porque não possuem dependência genérica (se a aeronave estiver em manutenção não necessitará de piloto, além de possuírem existências independentes) e nem dependência específica (a substituição dos pilotos não modifica a aeronave).
Suporte para análise da pergunta:
i. Outros exemplos: • CAIXA-REGISTRO: o funcionário exercendo o papel de caixa usa o sistema de registro de vendas da loja, formando, assim, uma associação. Não são classificadas como agregação/composição porque não possuem dependência genérica (o caixa e os registros têm existências independentes) e nem dependência específica (a substituição dos registros não modifica ou destrói o caixa). **Obs.:** Note a dificuldade nesses casos em se definir a classe que fará o papel do todo e a que será a parte.

Figura B. 4: Suporte 2 – para categoria “A usa ou gerencia B”.



Suporte 3

Voltar

Suporte 3

Pergunta:

A classe A se comunica com a classe B? Exemplo: entre CLIENTE e CAIXA podemos dizer que um cliente se comunica com o caixa (um funcionário específico) durante a compra. Portanto, encaixam-se nessa categoria e formam uma associação simples. Não são classificadas como agregação/composição porque não possuem dependência genérica (cliente e caixas possuem existências independentes) e nem dependência específica (a substituição dos clientes não modifica o caixa).

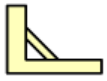
Suporte para análise da pergunta:

i. **Outros exemplos:**

- AGENTEDERESERVAS-PASSAGEIRO: o agente de reservas se comunica com o passageiro para efetuar uma reserva em um vôo, formando, assim, uma associação. Não são classificadas como agregação/composição porque não possuem dependência genérica (o agente e o passageiro têm existências independentes) e nem dependência específica (a substituição dos passageiros não modifica ou destrói o agente de reservas).

Obs.: Note a dificuldade nesses casos em se definir a classe que fará o papel do todo e a que será a parte.

Figura B. 5: Suporte 3 – para categoria “A se comunica B”.



Suporte 4

Voltar

Suporte 4

Pergunta:

A classe A está relacionada a uma transação com a classe B? Exemplo: entre CLIENTE e PAGAMENTO podemos dizer que o cliente efetua o pagamento de uma compra. Portanto, encaixam-se nessa categoria e formam uma associação simples. Não são classificadas como agregação/composição porque não possuem dependência genérica (um cliente pode não fazer o pagamento de uma dívida) e nem dependência específica (não cabe a ideia de substituição de pagamentos e nem a implicação de modificar o cliente).

Suporte para análise da pergunta:

i. **Outros exemplos:**

- PASSAGEIRO-BILHETE: o passageiro compra um bilhete para o vôo escolhido, formando, assim, uma associação. Não são classificadas como agregação/composição porque não possuem dependência genérica (passageiros e bilhetes têm existências independentes) e nem dependência específica (não cabe a ideia de substituição de bilhetes e nem a implicação de modificar o passageiro).

Obs.: Note a dificuldade nesses casos em se definir a classe que fará o papel do todo e a que será a parte.

Figura B. 6: Suporte 4 – para categoria “A está relacionado a uma transação B”.

Suporte 5

Pergunta:
A classe A é uma transação relacionada com uma outra transação da classe B? Exemplo: entre PAGAMENTO e VENDA podemos dizer que todo pagamento está relacionado a uma venda (seja de produtos ou serviços). Portanto, encaixam-se nessa categoria e formam uma associação simples. Não são classificadas como agregação/composição porque não possuem dependência genérica (possuem um tempo de vida curto, mas que deve ser capturado, não cabendo a ideia de substituição de pagamentos ou de vendas ao longo do tempo) e nem dependência específica (também não cabe a ideia de substituição de pagamentos e nem a implicação de destruição da venda).
Suporte para análise da pergunta:
i. **Outros exemplos:** • RESERVA-CANCELAMENTO: uma reserva pode ser cancelada, formando, assim, uma associação. Não são classificadas como agregação/composição porque não possuem dependência genérica (possuem um tempo de vida curto, mas que deve ser capturado, não cabendo a ideia de substituição de reservas ou de cancelamentos ao longo do tempo) e nem dependência específica (também não cabe a ideia de substituição de reservas e nem a implicação de destruição do cancelamento). **Obs.:** Note a dificuldade nesses casos em se definir a classe que fará o papel do todo e a que será a parte.

Figura B. 7: Suporte 5 – para categoria “A é uma transação relacionada com uma outra transação B”.

Suporte 6

Pergunta:

A classe A é adjacente a uma classe B? Exemplo: em um relacionamento entre PRODUTO e PRODUTO podemos dizer que é uma relação entre classes do mesmo tipo, ou seja, adjacentes. Portanto, encaixam-se nessa categoria e formam uma associação simples. Podemos querer fazer esse tipo de associação para sugerir outros produtos quando aquele produto específico estiver sendo comprado, por exemplo. Não são classificadas como agregação/composição porque não possuem dependência genérica (produtos não dependem de outros produtos para existirem) e nem dependência específica (pelo mesmo motivo anterior).

Suporte para análise da pergunta:

i. **Outros exemplos:**
   - CIDADE-CIDADE: podemos, por exemplo, querer fazer esse tipo de associação para sugerir roteiros de turismo ou simplesmente montar um caminho de cidades próximas. Não são classificadas como agregação/composição porque não possuem dependência genérica (cidades não dependem de outras cidades para existirem) e nem dependência específica (pelo mesmo motivo anterior).

ii. **Contra exemplos:**
   - CATEGORIA-CATEGORIA: uma categoria (e.g. de produtos) pode ter subcategorias, porém, se uma categoria for excluída do registro, todas as subcategorias que se relacionavam com ela não têm mais o porquê de existir, o que caracteriza uma dependência genérica.

Obs.: Note que as classes que fazem parte desse tipo de associação devem ser do mesmo tipo, ou seja, adjacentes conceitualmente.

Figura B. 8: Suporte 6 – para categoria “A é adjacente a B”.

Referências Bibliográficas

ARAÚJO, M. A. Modelagem de dados com a Visual Paradigm: do modelo de classes à criação do banco de dados. SQL MAGAZINE. Rio de Janeiro: DevMedia Group, edição 42, 2007.

ARTALE, A. e KEET, C.M. Essential and mandatory part-whole relations in conceptual data models. 21st International Workshop on Description Logics (DL'08). 13-16 May 2008, Dresden, Germany. CEUR-WS Vol-353.

ARTALE, A., GUARINO, N., e KEET, C.M. Formalising temporal constraints on part-whole relations. 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08). Gerhard Brewka, Jerome Lang (Eds.) AAAI Press, pp 673-683. Sydney, Australia, September 16-19, 2008.

BOCK, Conrad. “UML 2 Composition Model”, Journal of Object Technology, Volume 3, no. 10 (November 2004), pp. 47-73**.**

ELMASRI, R; NAVATHE, S. B. Sistemas de banco de dados – fundamentos e aplicações. 3.ed. Rio de Janeiro: LTC, 2002.

FOWLER, Martin. UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos. 3.ed. Porto Alegre: Bookman, 2005.

GUARINO, N.; GUIZZARDI, G. In the defense of ontological foundations for conceptual modeling. Scandinavian Journal of Information Systems, 2006, 18(1): 115-126.

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.

GUARINO, N.; WELTY, C. 2000b. *Towards a methodology for ontology based model engineering*. In Proceedings of the ECOOP-2000 Workshop on Model Engineering.

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.

GUIZZARDI, G.; WAGNER, G.; GUARINO, N.; SINDEREN, M. 2004a. "An Ontologically Well-Founded Profile for UML Conceptual Models". In Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE, 2004), Latvia, LNCS 3084, Springer-Verlag.

GUIZZARDI, G.; WAGNER, G.; HERRE, H. 2004b. "On the Foundations of UML as an Ontology Representation Language". 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW), Northamptonshire, UK, 2004.

GUIZZARDI, G., Ontological Foundations for Structural Conceptual Models. PhD thesis, University of Twente, The Netherlands, 2005.

GUIZZARDI, G. 2007. "Modal Aspects of Object Types and Part-Whole Relations and the *de re*/*de dicto* distinction". 19th International Conference on Advanced Information Systems Engineering (CAISE'07), Trondheim, 2007, Lecture Notes in Computer Science 4495, Springer-Verlag.

GUIZZARDI, G. The Problem of Transitivity of Part-Whole Relations in Conceptual Modeling Revisited, 21st Intl. Conf. on Advanced Information Systems Engineering (CAISE'09), The Netherlands, 2009.

KEET, C.M. Part-whole relations in Object-Role Models. 2nd International Workshop on Object-Role Modelling (ORM'06), Montpellier, France, Nov 2-3, 2006. In: OTM Workshops 2005. Meersman, R., Tari, Z., Herrero., P. et al. (Eds.), Lecture Notes in Computer Science LNCS 4278. Berlin: Springer-Verlag, 2006. pp1116-1127.

KEET, C. M. e ARTALE, A. 2008. Representing and reasoning over a taxonomy of part-whole relations. Appl. Ontol. 3, 1-2 (Jan. 2008), 91-110.

LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado. 2.ed. Porto Alegre: Bookman, 2004.

MEDEIROS, Ernani Sales de. *Desenvolvendo software com UML 2.0: definitivo*. São Paulo, Pearson Makron Books, 2004.

OMG. Software and systems process engineering meta-model specification. Technical report. Object Management Group – OMG. 2008.

OMG. Unified Modeling Language: Superstructure. v. 2.3, formal/2010-05-05. Object Management Group – OMG. 2010. Disponível em: http://www.omg.org/spec/UML/2.3/Superstructure. Último acesso em: 20 de nov. 2010

PAULA FILHO, W. de P. *Engenharia de Software: Fundamentos, Métodos e Padrões*. Rio de Janeiro, LTC Editora, 2009.

PERÍLIO, R. Domain Model: Uma forma mais eficiente de construir aplicações Enterprise. MUNDOJ. Curitiba: Editora Mundo, número 42, ano VIII, 2010.

TAVARES, D. B. Procedimento de análise para validação de diagrama de classes de domínio baseado em análise ontológica. Master's thesis,

Universidade Federal de Viçosa, UFV – Viçosa – MG – Brasil. Dissertação de Mestrado. 2008.

TAVARES, D. B. ; OLIVEIRA, A. P. ; BRAGA, J. L. ; LISBOA FILHO, J. . Validação de Diagrama de Classes por meio da Técnica OntoCon. In: CLEI 2008 / XXXIV Conferencia Latinoamericana de Informática, 2008, Santa Fe. CLEI 2008 / XXXIV Conferencia Latinoamericana de Informática, 2008. p. 330-339.

TAVARES, D. B. ; OLIVEIRA, A. P. ; BRAGA, J.L. ; LISBOA FILHO, J. . Analysis procedure for validation of the domain class diagrams based on ontological analysis. In *Proceedings of the ER 2009 Workshops (Comol, Ethecom, Fp-Uml, Most-onisw, Qois, Rigim, Secogis) on Advances in Conceptual Modeling - Challenging Perspectives* (Gramado, Brazil, November 09 - 12, 2009). C. A. Heuser and G. Pernul, Eds. Lecture Notes In Computer Science. Springer-Verlag, Berlin, Heidelberg, 159-168.

VARZI, Achille. Mereology. Stanford Encyclopedia of Philosophy, revisado em Maio 2009. Disponível em: http://plato.stanford.edu/entries/mereology/. Último acesso em: 20 de nov. 2010.

VILLELA, M. L. B. *Validação de Diagramas de Classe por meio de Propriedades Ontológicas*. Belo Horizonte: Universidade Federal de Minas Gerais, 2004. (Dissertação, Mestrado em Ciência da Computação).

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.