UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
MESTRADO EM INFORMÁTICA

ALINE FREITAS MARTINS

CONSTRUÇÃO DE ONTOLOGIAS DE TAREFA E SUA REUTILIZAÇÃO NA ENGENHARIA DE REQUISITOS

VITÓRIA, JUNHO 2009

ALINE FREITAS MARTINS

CONSTRUÇÃO DE ONTOLOGIAS DE TAREFA E SUA REUTILIZAÇÃO NA ENGENHARIA DE REQUISITOS

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

VITÓRIA, JUNHO 2009

ALINE FREITAS MARTINS

CONSTRUÇÃO DE ONTOLOGIAS DE TAREFA E SUA REUTILIZAÇÃO NA ENGENHARIA DE REQUISITOS

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 24 de junho de 2009.

COMISSÃO EXAMINADORA

_____
Prof. Ricardo de Almeida Falbo, D.Sc.
Universidade Federal do Espírito Santo (UFES)
Orientador

_____
Profª. Fernanda Araújo Baião, D.Sc.
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

_____
Prof. Giancarlo Guizzardi, PhD.
Universidade Federal do Espírito Santo (UFES)

VITÓRIA, JUNHO 2009

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Martins, Aline Freitas, 1984-

M386c Construção de ontologias de tarefa e sua reutilização na engenharia de requisitos / Aline Freitas Martins. – 2009.

178 f. : il.

Orientador: Ricardo de Almeida Falbo.

Dissertação (mestrado) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Software - Desenvolvimento. 2. Engenharia de software. 3. Software - Reutilização. 4. Ontologias (Recuperação da informação). I. Falbo, Ricardo de Almeida. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

DEDICATÓRIA

Dedico esta dissertação a meus pais, Lindalva e Vanderley.

AGRADECIMENTOS

Agradeço a **Deus**, pela vida e pela força.
A meus **pais** pelo exemplo, pela oportunidade e pelo apoio.
A minha **irmã** pela companhia e pelo carinho.
Ao **Felipe** pelo amor e pela paciência.
Aos **amigos** pelo companheirismo e por me animarem sempre.
Ao **Ricardo Falbo** pela orientação perfeita.
Aos **professores** que colaboraram com seus conhecimentos.
A **todos do departamento** que estiveram dispostos ajudar.

À **FAPES (Fundação de apoio à Ciência e Tecnologia do Espírito Santo),** pelo apoio financeiro, viabilizado por meio de uma bolsa de mestrado**.**

RESUMO

Atualmente, reconhece-se que a reutilização oferece uma importante oportunidade para alcançar melhorias no desenvolvimento de software. Maiores benefícios, no entanto, são obtidos pela reutilização em níveis mais altos de abstração, sobretudo pelo reúso de conhecimento. Em relação à reutilização de conhecimento, dois grandes tipos de conhecimento devem ser considerados: conhecimento de domínio e de tarefa. Para desenvolver o conhecimento para a reutilização, necessita-se de modelos para capturar ambos e ontologias podem ser utilizadas para este fim. Ontologias de domínio descrevem o vocabulário relacionado a um domínio genérico, enquanto ontologias de tarefa descrevem o vocabulário relacionado a uma tarefa genérica. Ontologias de domínio têm sido amplamente utilizadas nas mais diversas áreas da Ciência da Computação, entretanto o mesmo não ocorre com ontologias de tarefa. Existem poucos trabalhos apresentando ontologias de tarefa e não há uniformidade na representação das mesmas.

O conhecimento de tarefa envolve dois aspectos principais: a decomposição em subtarefas e os papéis de conhecimento que as entidades do domínio irão exercer na realização da tarefa. Este trabalho propõe o uso de perfis UML baseados na Ontologia de Fundamentação Unificada (*Unified Foundational Ontology* – UFO) para representar ontologias de tarefa: OntoUML (com base em diagramas de classes), para a modelagem dos papéis de conhecimento envolvidos e suas propriedades e relações, e E-OntoUML (com base em diagramas de atividades), para capturar a decomposição de tarefa e a participação dos papéis de conhecimento nas mesmas. O primeiro tem sido bastante utilizado para representar ontologias de domínio, enquanto o segundo é um novo perfil, proposto neste trabalho. Discute-se, também, como ontologias de tarefa podem ser combinadas com ontologias de domínio, a fim de descrever o conhecimento relativo a uma classe de aplicações.

Por fim, uma vez que o principal objetivo para a captura do conhecimento é permitir o seu reuso e compartilhamento, propõe-se uma abordagem para reutilização de ontologias de tarefa no processo de Engenharia de Requisitos.

Palavras-chave: Ontologias, Conhecimento de Tarefa, Ontologias de Tarefa, Reutilização e Engenharia de Requisitos.

ABSTRACT

Nowadays, it is acknowledged that reuse offers an important opportunity to achieve improvements in software development. Greater benefits, however, are achieved by reusing knowledge. Concerning knowledge reuse, two major kinds of knowledge should be considered: domain and task knowledge. For developing knowledge for reuse, models are needed to capture both, and ontologies can be used for this purpose. Domain ontologies describe the vocabulary related to a generic domain, while task ontologies describe the vocabulary related to a generic task. Domain ontologies have been extensively used in several areas in Computer Science, however, the same does not occur with task ontologies. There are few works presenting task ontologies, and there is no uniformity in representing them.

Task knowledge involves two different facets: task decomposition and knowledge roles involved in the fulfillment of the subtasks. This work proposes the use of UFO (Unified Foundational Ontology) based UML profiles for representing task knowledge: OntoUML (that concerns class diagrams) modeling the knowledge roles involved and their properties and relations, and E-OntoUML (that concerns activity diagrams) capturing task decomposition and how knowledge roles act in their fulfillment. OntoUML is currently used to represent several domain ontologies. E-OntoUML is a new profile that is proposed here. This work also discusses how task ontologies can be combined with domain ontologies in order to describe the knowledge involved in a class of applications.

Finally, since the main goal for capturing knowledge is to allow its reuse and sharing, an approach is proposed for reusing task ontologies in the Requirements Engineering process.

Keywords: Ontologies, Task Knowledge, Task Ontologies, Requirement Engineering, Reuse.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1. INTRODUÇÃO

Este capítulo apresenta as motivações que serviram de base para a elaboração desta dissertação e os objetivos que foram buscados. Também aborda a maneira como a pesquisa foi conduzida e qual a organização deste trabalho.

1.1. MOTIVAÇÃO

Atualmente, é consenso que a reutilização oferece uma importante oportunidade para atingir melhorias no desenvolvimento de software. O reúso de software pode ocorrer em diversos níveis, desde código até conhecimento. Contudo, maiores benefícios podem ser atingidos ao se reutilizar artefatos de mais alto nível de abstração, isto é, ao se reusar conhecimento (WANG; CHAN, 2001).

Visando à reutilização, dois principais tipos de conhecimento podem ser considerados: conhecimento de domínio e de tarefa (GUARINO, 1998). Para o desenvolvimento com reúso, é necessário capturar ambos os tipos de conhecimento e ontologias podem ser utilizadas para este propósito. Uma ontologia define um vocabulário para descrever uma certa realidade (GUARINO, 1998) e pode ser usada para organizar uma porção de conhecimento capturado, com o objetivo de facilitar o acesso, o entendimento e permitir o reúso.

De acordo com Guarino (1998), existem quatro tipos principais de ontologias: ontologias de topo, de domínio, de tarefa e de aplicação. Ontologias de topo ou de fundamentação descrevem conceitos e relações gerais, aplicáveis aos diversos domínios e tarefas. Ontologias de domínio e de tarefa descrevem, respectivamente, conceitos e relações de um domínio e de uma tarefa genéricos. Já ontologias de aplicação descrevem os conceitos adotados em uma aplicação específica.

Ontologias de domínio vêm sendo muito estudadas e utilizadas como artefatos reutilizáveis (GUIZZARDI, 2005), (GÓMEZ-PÉREZ et al., 2004), (NARDI, 2006) e (FALBO, 2004), já existindo diversos métodos bem estruturados para o desenvolvimento das mesmas. Porém, em se tratando de ontologias de tarefa, pouco se tem estudado. Não existe uma representação padrão para ontologias de tarefa, metodologias difundidas para sua elaboração e técnicas para integração com outros tipos de ontologias.

Ontologias de tarefa são importantes, pois capturam o conhecimento sob uma perspectiva comportamental e são um dos pilares para a geração de ontologias de aplicação, que podem ser obtidas pela sua integração com ontologias de domínio. Esses três tipos de

ontologias podem, ainda, estar fundamentados em ontologia de fundamentação, o que permite adicionar maior semântica a elas, tornando-as mais consistentes.

Ontologias de domínio e de tarefa capturam conhecimento sob diferentes perspectivas e podem ser utilizadas como artefatos reutilizáveis durante a modelagem de um sistema. O conhecimento modelado nessas ontologias inclui informações estruturais e comportamentais a respeito dos domínios nos quais o sistema vai atuar e das tarefas que o sistema ou seus usuários vão realizar. Assim, nas fases iniciais do desenvolvimento de software, quando se definem os requisitos do sistema, essas ontologias podem ser importantes artefatos reutilizáveis.

A Engenharia de Requisitos (ER) é o ramo da Engenharia de Software que contempla das atividades do processo de software responsáveis por tratar os requisitos (funcionais e não-funcionais) de um sistema software a ser construído. Segundo Kotonya e Sommerville (1998), a Engenharia de Requisitos é o processo sistemático de levantamento, entendimento, análise, documentação e gerência de requisitos. Uma das principais atividades do processo de ER é a modelagem. Aspectos estruturais e comportamentais são observados e estes são modelados e documentados para, assim, registrar e tornar explícito o conhecimento adquirido. Os modelos agregam, portanto, informações que são fundamentais para dar prosseguimento no desenvolvimento de um sistema.

A atividade de modelagem é custosa, exigindo muito tempo e mão de obra qualificada, além de ser uma atividade de intensa captura de conhecimento. Muitos esforços podem ser poupados quando inseridas abordagens de reúso nessa etapa e há diversas iniciativas nesse sentido, tais como Engenharia de Domínio e Padrões de Análise. Além disso, ontologias de domínio têm sido utilizadas como base para entendimento e desenvolvimento de modelos estruturais de sistemas (NARDI, 2006).

Porém o mesmo não ocorre com ontologias de tarefa. O conhecimento que elas capturam é pouco utilizado, o que é decorrência, sobretudo, da ausência de padrões para a sua especificação e de abordagens para a integração com ontologias de domínio e para a sua reutilização. Diante do exposto, este trabalho procurou investigar formas de representar ontologias de tarefa e como reutilizá-las na ER.

1.2. OBJETIVOS

O objetivo geral deste trabalho é propor uma organização e representação do conhecimento de tarefa em ontologias de tarefa, assim como definir uma estratégia de sua

integração com ontologias de domínio para dar origem a ontologias de classe de aplicação. Esses três tipos de ontologias (de domínio, de tarefa e de classe de aplicação) são genéricos e reutilizáveis e, portanto, são definidas formas de reutilizá-los no processo de desenvolvimento de software, mais especificamente no processo de Engenharia de Requisitos (ER).

Ontologias de tarefa visam capturar o conhecimento envolvido em uma tarefa genérica, o que inclui tanto conhecimento estrutural, isto é, os papéis que conceitos do domínio vão exercer na execução da tarefa, quanto conhecimento comportamental, que se refere ao fluxo de controle de sub-tarefas e o comportamento que instâncias dos papéis terão nas sub-tarefas identificadas. Uma vez que o conhecimento de tarefa envolve tanto uma perspectiva estrutural quanto uma comportamental, definiu-se como objetivo específico deste trabalho definir uma representação que seja capaz de modelar essas duas perspectivas. Além disso, é importante que ontologias de tarefa e domínio estejam fundamentadas em ontologias de fundamentação. Uma ontologia de fundamentação serve como base para o estabelecimento de consenso e negociação entre humanos, pois provê semântica de mundo real para seus elementos de modelo (GUIZZARDI, 2005). Levando isso em consideração, é um objetivo específico deste trabalho que a representação proposta seja fundamentada em uma ontologia de fundamentação para aumentar a consistência e qualidade dos modelos gerados. Para tal, foi utilizada a Ontologia de Fundamentação Unificada (*Unified Foundational Ontology* – UFO) (GUIZZARDI, 2005) (GUIZZARDI et al., 2008a) (GUIZZARDI et al., 2008b).

Aplicações atuam em um determinado domínio realizando uma tarefa ou mais tarefas específicas. Assim, é importante integrar esses tipos de conhecimento, sendo também um objetivo específico deste trabalho propor diretrizes para a integração de ontologias de tarefa com ontologias de domínio.

Por fim, no contexto deste trabalho, deseja-se reutilizar o conhecimento capturado por ontologias de domínio e de tarefa no processo de Engenharia de Requisitos. Assim, é também um objetivo específico deste trabalho propor diretrizes para apoiar a reutilização desse conhecimento na Engenharia de Requisitos.

1.3. HISTÓRICO DO DESENVOLVIMENTO DO TRABALHO

Para atingir os objetivos listados na seção anterior, foram realizadas as atividades descritas a seguir.

Inicialmente, foi feito um levantamento bibliográfico sobre os temas relacionados à ER com foco em reutilização. Durante esse estudo, notou-se, em especial, que o reúso de modelos

e ontologias de domínio já está bem difundido, sendo os mesmos utilizados para a geração e compreensão de modelos estruturais de sistemas. Porém o mesmo não acontece com ontologias de tarefas. Essa conclusão foi o primeiro passo na direção da definição do tema tratado neste trabalho.

Definido o escopo de interesse, foi feito um levantamento bibliográfico sobre os principais temas relacionados ao reúso de conhecimento de tarefa na ER, com ênfase em ontologias de tarefa. Durante esses estudos, notou-se a falta de padronização para modelagem e representação de ontologias de tarefa, o que foi considerado um dos principais obstáculos à sua reutilização. Assim, antes de propor diretrizes para a reutilização de conhecimento de tarefa na ER, foi necessário estudar formas para representá-lo. A partir do estudo realizado, foi proposta uma abordagem de organização e representação de ontologias de tarefa usando diagramas da Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML) (OMG, 2007), por ser um padrão já bastante difundido para modelagem de ontologias de domínio. Essa abordagem considera duas perspectivas principais para a captura do conhecimento de tarefa: uma visão estrutural, que trata dos papéis de conhecimento envolvidos na tarefa e seus relacionamentos, e outra comportamental, que trata da decomposição das tarefas e de seu fluxo de controle. Utilizam-se diagramas de classe para representar a visão estrutural da tarefa e diagramas de atividades para representar a visão comportamental. Para avaliar a abordagem proposta e identificar pontos positivos e negativos, desenvolveu-se uma ontologia de tarefa de Locação como estudo de caso. Considerou-se, ainda, a sua integração com ontologias de domínio para a construção de ontologias de classes de aplicação. Essa primeira parte do trabalho foi publicada no 3º Workshop de Ontologias e suas Aplicações (*3rd Workshop on Ontologies and Their Applications* – WONTO'2008) sob o título "*Models for Representing Task Ontologies*" (MARTINS; FALBO, 2008).

Tendo em mente que idealmente ontologias de tarefa (e de domínio) devem ser construídas com base em ontologias de fundamentação, passou-se a estudar a Ontologia de Fundamentação Unificada (*Unified Foundational Ontology* - UFO) (GUIZZARDI, 2005) (GUIZZARDI et al., 2008a) (GUIZZARDI et al., 2008b). Seguindo a linha de trabalho de GUIZZARDI (2005), que procurou enriquecer a semântica de diagramas de classes da UML com as distinções ontológicas de UFO, passou-se a investigar como enriquecer as notações de diagramas de atividades da UML com a semântica da UFO. Para tal, estudou-se a parte do meta-modelo da UML que trata dos diagramas de atividades e se propôs um perfil UML ontologicamente bem fundamentado para a representação de ontologias de tarefa, denominado E-OntoUML. Neste momento, identificou-se correspondências semânticas entre os elementos

do meta-modelo da UML e os conceitos da UFO e pontos em que a UFO poderia complementar a semântica da UML.

Definido o perfil E-OntoUML, iniciaram-se os estudos para elaboração de uma proposta de reutilização de ontologias de tarefa na Engenharia de Requisitos. Estudou-se a correspondência entre os modelos tipicamente utilizados na especificação de requisitos de software e os modelos propostos para a representação de ontologias de tarefa, de modo a definir uma abordagem de reúso.

Por fim, após definidos os principais aspectos da abordagem, foi iniciada a elaboração do texto desta dissertação.

1.4. ORGANIZAÇÃO DO TRABALHO

Esta dissertação está organizada em seis capítulos. Além deste capítulo, que apresenta a Introdução, há mais cinco capítulos com o seguinte conteúdo:

- *Capítulo 2 – Engenharia de Requisitos e Reutilização*: apresenta uma revisão da literatura sobre Engenharia de Requisitos com enfoque na reutilização ao longo desse processo.
- *Capítulo 3 – Ontologias*: apresenta um referencial teórico a respeito de ontologias, incluindo definições, tipos e aplicações. É dado destaque às ontologias de tarefa e de fundamentação, pois estas são a base para a proposta apresentada neste trabalho.
- *Capítulo 4 - Ontologias de Tarefa*: *Construção e Integração com Ontologias de Domínio*: apresenta a abordagem de organização de ontologias de tarefa e o perfil de modelagem E-OntoUML. Trata, ainda, da integração de ontologias de tarefa com ontologias de domínio.
- *Capítulo 5 – Reutilização de Conhecimento de Tarefa na Engenharia de Requisitos*: apresenta uma abordagem de reutilização de ontologias de tarefa no desenvolvimento de modelos comportamentais e estruturais durante a atividade de Modelagem Conceitual da Engenharia de Requisitos.
- *Capítulo 6 - Considerações Finais*: apresenta as conclusões do trabalho, as contribuições, dificuldades e propostas de trabalhos futuros.

CAPÍTULO 2. ENGENHARIA DE REQUISITOS E REUTILIZAÇÃO

Este capítulo apresenta os principais conceitos a respeito da Engenharia de Requisitos e de abordagens de Reutilização nesse processo. Enfatiza-se a importância da modelagem e do reúso de conhecimento de tarefa e de domínio na geração de modelos estruturais e comportamentais de sistemas.

2.1. INTRODUÇÃO

O processo de desenvolvimento de software tem início pela identificação dos requisitos, ou seja, dos serviços e das funções que o sistema deverá prover, bem como de suas restrições e propriedades importantes. A medida de sucesso de um software é dada em grande parte pelo grau que ele atende a esses requisitos (ROBERTSON; ROBERTSON, 1999). Para tanto, é importante não só um bom levantamento dos requisitos, mas também o acompanhamento das atividades relacionadas a estes durante todo o ciclo de vida do software.

Ao conjunto das atividades que tratam os requisitos de um sistema dá-se o nome de Engenharia de Requisitos (ER). O processo de ER envolve as atividades de identificação, análise, documentação, manutenção e gerência dos requisitos do sistema. Ele produz especificações e modelos que representam os requisitos do sistema e que direcionam o processo de desenvolvimento. É importante, portanto, que esses modelos sejam abrangentes e consistentes, representando aspectos estruturais e comportamentais do sistema.

A modelagem aparece como o núcleo do processo de ER. Ela é uma atividade cara, porque pessoas experientes despendem tempo elaborando modelos e os analisando em busca de conflitos e ambiguidades. Muitos esforços são gastos com a elaboração de modelos e uma forma de minimizar esses esforços é inserir abordagens de reúso nesse processo. Atualmente, a reutilização tem sido apontada como um fator importante para o aumento da qualidade e da produtividade no desenvolvimento de software (GIMENES; HUZITA, 2005). No processo de ER pode-se buscar reutilizar conhecimento tanto estrutural quanto comportamental.

Este capítulo aborda a Engenharia de Requisitos, com foco na reutilização em seu processo, e está organizado da seguinte forma: a Seção 2.2 apresenta uma visão geral da ER, a Seção 2.3 discute com um pouco mais de detalhes a modelagem conceitual, a Seção 2.4 apresenta abordagens de reutilização de conhecimento na ER e, por fim, a Seção 2.5 apresenta as considerações finais do capítulo.

2.2. ENGENHARIA DE REQUISITOS

Requisitos definem o que o software deverá fazer, quais são suas restrições e características. Eles devem refletir as necessidades e expectativas do usuário. Esse fato liga os requisitos à qualidade do produto final. O IEEE (*The Institute of Electrical and Electronics Engineers*) (IEEE,1998) define qualidade de software como o grau com que um sistema, componente ou processo atende aos requisitos especificados e às expectativas ou necessidades de clientes ou usuários. Portanto, é importante o bom gerenciamento do processo de Engenharia de Requisitos para garantir que, ao final do processo de desenvolvimento, os requisitos levantados tenham sido atendidos.

Identificar bem os requisitos é um passo essencial para se conseguir a satisfação do cliente e, por conseguinte, entregar um produto sólido e de qualidade (WIEGERS, 2003). Os requisitos guiam várias etapas do desenvolvimento e, por isso, é essencial um bom entendimento sobre eles, sua classificação e documentação.

Segundo Sommerville (2003) e Kotonya e Sommerville (1998), os requisitos podem ser:

- *Funcionais:* descrevem a funcionalidade ou os serviços que se espera que o sistema forneça.
- *Não funcionais*: se referem a restrições sobre os serviços ou funções oferecidos pelo sistema, podendo estar relacionados a propriedades como confiabilidade, eficiência, usabilidade etc. Surgem das necessidades dos usuários em razão de restrições de orçamento, políticas organizacionais, necessidade de interoperação com outros sistemas, hardware, legislação, fatores externos etc.

A Engenharia de Requisitos (ER) é o ramo da Engenharia de Software que envolve as atividades relativas a desenvolver, documentar e dar manutenção ao conjunto de requisitos de um sistema (KOTONYA; SOMMERVILLE, 1998). Pode ainda ser descrita como um processo, ou seja, um conjunto organizado de atividades, métodos, técnicas, práticas e transformações que ajudam a derivar, validar e manter os requisitos gerados.

O processo de engenharia de requisitos envolve criatividade, interação de diferentes pessoas, conhecimento e experiência para transformar informações diversas (sobre a organização, sobre leis, sobre o sistema a ser construído etc.) em modelos e documentos que direcionem o desenvolvimento de software, como ilustra a Figura 2.1 (KOTONYA; SOMMERVILLE, 1998). A ER é fundamental, pois possibilita estimativas de custo e tempo

mais precisas e permite que mudanças em requisitos levantados inicialmente sejam melhor gerenciadas.

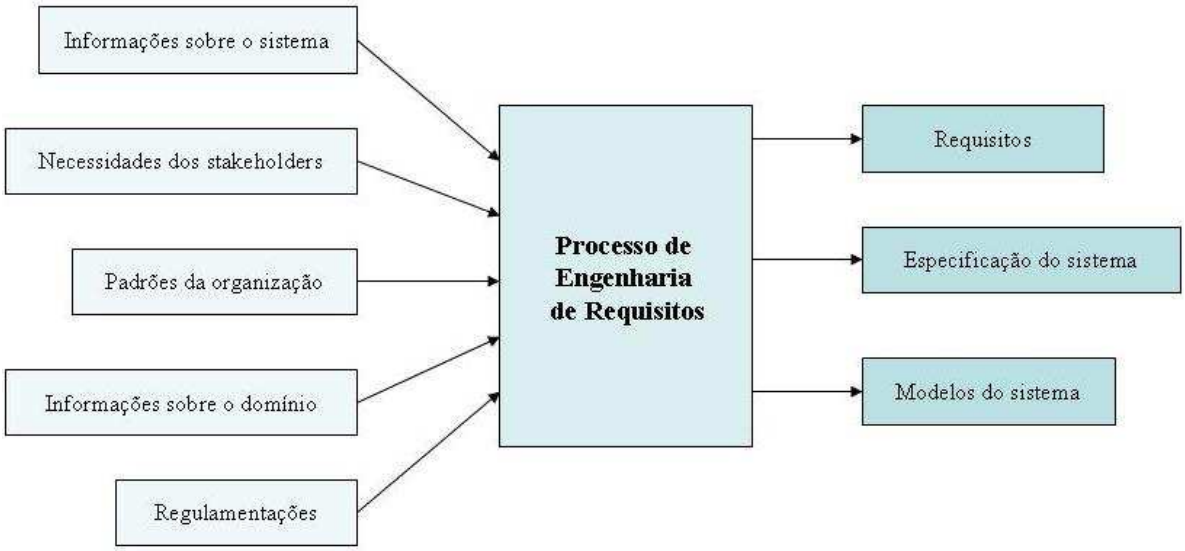


Figura 2.1 - Entradas e saídas do processo de engenharia de requisitos (KOTONYA; SOMMERVILLE, 1998)

Kotonya e Sommerville (1998) propõem que no processo de ER sejam realizadas as atividades mostradas na Figura 2.2, a saber: levantamento, análise e negociação, documentação, verificação e validação e gerência de requisitos.

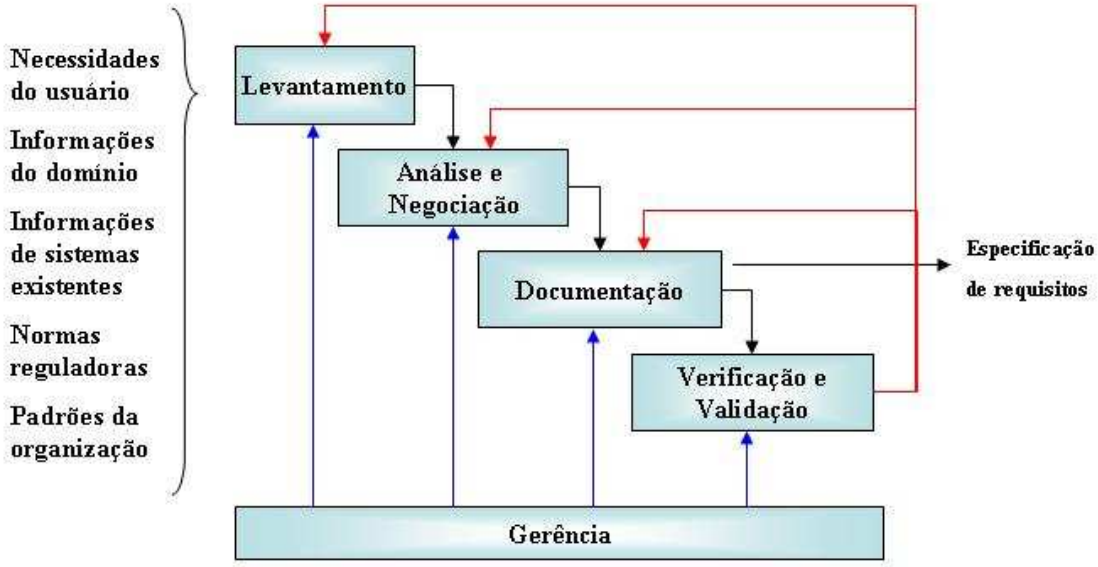


Figura 2.2 - Atividades do Processo de Engenharia de Requisitos.

O levantamento de requisitos corresponde à fase inicial do processo de Engenharia de Requisitos e envolve as atividades de descoberta dos requisitos. Costuma ser chamado também de processo de aquisição de requisitos ou de descoberta de requisitos.

Para levantar quais são os requisitos de um sistema, devem-se obter informações dos interessados (*stakeholders*), consultar documentos, obter conhecimento acerca do domínio e estudar o negócio da organização. A atividade de levantamento de requisitos é dominada por fatores humanos, sociais e organizacionais e envolve pessoas com diferentes conhecimentos e objetivos, o que a torna complexa. Christel e Kang (1992 apud PRESSMAN, 2006) e Kotonya e Sommerville (1998) citam alguns problemas que tornam o levantamento de requisitos uma tarefa difícil. Dentre eles, destacam-se:

- Os clientes/usuários têm pouca compreensão das capacidades e limitações de seu ambiente computacional, não têm pleno entendimento do domínio do problema, têm dificuldade de comunicar as necessidades ao engenheiro de sistemas e, muitas vezes, omitem informação por acreditar que ela é “óbvia”.
- É difícil compreender e coletar informações quando existem muitos termos desconhecidos, manuais técnicos etc.
- Pessoas que entendem o problema a ser resolvido podem ser muito ocupadas e não ter muito tempo para, juntamente como analista, levantar os requisitos e entender o sistema.
- Os interessados (*stakeholders*) não sabem muito o que querem do sistema e não conhecem muitos termos.

Análise e negociação são atividades que envolvem a modelagem, a descoberta de problemas com os requisitos do sistema e a obtenção da concordância entre os interessados. São atividades caras, porque pessoas experientes despendem tempo lendo documentos cuidadosamente e pensando sobre a implicação das afirmações desses documentos (KOTONYA; SOMMERVILLE, 1998).

A Análise de Requisitos envolve o estudo dos produtos do levantamento de requisitos para identificar os problemas e conflitos na definição dos requisitos (KOTONYA; SOMMERVILLE, 1998), com o objetivo de estabelecer um conjunto de requisitos completo e consistente. Essa etapa envolve a criação de modelos descrevendo requisitos num nível de abstração mais alto. Modelos podem indicar requisitos incorretos, inconsistentes, supérfluos e ausentes.

Resumidamente, pode-se dizer que a análise atende a dois propósitos: (i) prover um caminho para clientes e desenvolvedores concordarem com o que o sistema irá fazer e (ii) a especificação provê um guia para o projeto (*design*) do sistema (PFLEEGER, 1998).

Descobertos os conflitos e definidas as propriedades do conjunto de requisitos do sistema, é necessário discutir essas conclusões com o cliente. Essa atividade é chamada de negociação e envolve a discussão dos conflitos e prioridades entre requisitos com os interessados.

Os requisitos capturados nas etapas anteriores são descritos e modelados em documentos. Documentação é, portanto, uma atividade de registro e oficialização dos resultados da engenharia de requisitos, que tem como produto principal o Documento de Requisitos de Software ou Especificação de Requisitos de Software (ERS) (SOMMERVILLE, 2003).

Nas atividades de Verificação e Validação de Requisitos examina-se a especificação para assegurar que: (i) todos os requisitos do sistema tenham sido declarados de modo não-ambíguo, (ii) as inconsistências, omissões e erros tenham sido detectados e corrigidos, (iii) os requisitos estão em conformidade com as características de qualidade e (iv) realmente satisfazem a necessidades dos usuários (PRESSMAN, 2006) (KOTONYA; SOMMERVILLE, 1998) (WIEGERS, 2003).

A Gerência de Requisitos é definida como o conjunto de atividades que ajudam a equipe de projeto a identificar, controlar e rastrear requisitos e gerenciar mudanças de requisitos em qualquer época, à medida que o projeto prossegue (KOTONYA; SOMMERVILLE, 1998) (PRESSMAN, 2006). Envolve o gerenciamento de relacionamentos entre requisitos e de dependências entre documentos de requisitos e outros documentos do processo de desenvolvimento (KOTONYA; SOMMERVILLE, 1998).

Note que a modelagem aparece como o núcleo do processo de Engenharia de Requisitos. Modelos estão em todas as etapas, pois são objetos de comunicação e formalização das informações e possibilitam raciocinar sobre seu conteúdo. Modelos proveem a base para a documentação, gerência e evolução. Isso é corroborado pelo fato de a maioria das pesquisas na ER serem voltadas para técnicas de modelagem e especificação (VAN LAMSWEERDE, 2000). A seção seguinte apresenta maiores detalhes sobre a modelagem de requisitos, enfatizando os modelos criados e a linguagem comumente utilizada para apoiar essa atividade.

2.3. MODELAGEM CONCEITUAL

A modelagem conceitual é uma parte importante da Engenharia de Requisitos. O levantamento do conhecimento geral necessário para um sistema é uma atividade necessária. Sistemas não podem ser projetados por programadores sem um levantamento inicial do conhecimento que ele precisa tratar. Segundo Fowler (1997), identificar e analisar requisitos envolve tentar entender o problema e listar requisitos e casos de uso. Além disso, envolve analisar mais a fundo os requisitos e criar um modelo mental do que é o problema, com o objetivo de simplificá-lo e melhor entendê-lo. À criação desse modelo mental dá-se o nome de modelagem conceitual (FOWLER, 1997).

A escolha de quais modelos usar para representar um sistema afeta a flexibilidade e reusabilidade do sistema resultante. Logo, nessa escolha, deve-se considerar também a manutenibilidade e possibilidade de expandir o sistema no futuro.

Para construir um software para um propósito específico, tem-se que desenvolver um modelo conceitual que é apropriado às suas necessidades (FOWLER, 1997). Para que o sistema execute funções, é preciso ter algum conhecimento sobre o domínio e sobre as funções que o mesmo terá que executar. A esse conhecimento dá-se o nome de esquema conceitual. O principal propósito da modelagem conceitual é, portanto, identificar o esquema conceitual de um sistema (OLIVÉ, 2007).

Esquemas conceituais são escritos em linguagens ditas linguagens de modelagem conceitual e a Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML) vem sendo utilizada com esse propósito (OLIVÉ, 2007). Ela é capaz de representar o conhecimento sobre o domínio, incluindo seus conceitos e relacionamentos, e sobre as funções que serão realizadas por um sistema.

2.3.1. A Linguagem de Modelagem Unificada

A Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML) (OMG, 2007) é uma linguagem para especificação que tem como objetivo prover para arquitetos de sistemas, engenheiros de software e desenvolvedores, ferramentas para análise, projeto e implementação de sistemas e modelagem de processos (OMG, 2007). Ela sintetiza os principais métodos existentes, sendo considerada uma das linguagens mais expressivas para a modelagem de sistemas. Por meio de seus diagramas, é possível representar sistemas sob diversas perspectivas.

A UML 2.0 foi formalmente adotada em 2005 e possui treze tipos de diagramas, cada uma com um foco específico. Esses diagramas são classificados em dois tipos principais: estruturais e comportamentais (ERICKSON, 2008). Os diagramas estruturais da UML 2.0 são (OMG, 2007): diagramas de classes, diagramas de estruturas compostas, diagramas de componentes, diagramas de objetos e diagramas de pacotes. Os diagramas comportamentais dessa versão da UML são (OMG, 2007): diagramas de atividades, diagramas de casos de uso, diagramas de máquinas de estados e diagramas de interação, os quais podem ser diagramas de sequência, diagramas de comunicação, diagramas de visão geral de interação (*interaction overview diagrams*) e diagramas de regulação de tempo (*timing diagrams*)

Levando-se em conta o estado da prática, os modelos mais comumente usados são os diagramas de casos de uso e suas descrições, os diagramas de classes e os diagramas de máquinas de estados, de atividade e de sequência. Particularmente para este trabalho, os diagramas de maior relevância são os diagramas de classes, de casos de uso, de atividades e de máquinas de estados, que são brevemente comentados nas subseções seguintes.

Uma das maiores críticas à UML é em relação à sua semântica parcamente definida. Em outras palavras, os símbolos da UML podem ser interpretados de várias formas, deixando muita subjetividade na interpretação (ERICKSON, 2008).

Para amenizar esse problema, a UML possui mecanismos de extensão que possibilitam adicionar semântica aos seus modelos, permitindo modificar os elementos da linguagem para suprir necessidades de modelagem. Extensões da linguagem podem ser feitas por meio de especializações do metamodelo da UML para adicionar a nova semântica aos elementos de modelo. Um conjunto coerente de tais extensões, definido de acordo com um propósito ou domínio específico, constitui um perfil UML (OMG, 2007). Um perfil pode ser criado para: (i) adicionar semântica que não é determinada ou que não exista no metamodelo, (ii) dar uma diferente notação para símbolos já existentes e (iii) adicionar restrições que restringem o modo como é usado o metamodelo (OMG, 2007).

2.3.1.1. Diagramas de Classes

Diagramas de classe têm como objetivo representar a estrutura estática de um sistema, isto é, as classes e os relacionamentos existentes entre elas. O diagrama de classes é considerado estático, pois a estrutura descrita é sempre válida em qualquer ponto no ciclo de vida do sistema e ele é a base para a construção de outros modelos (BOOCH et al., 2005) (PRESSMAN, 2006).

2.3.1.2. Diagramas e Descrições de Casos de Uso

Diagramas de casos de uso procuram prover uma descrição clara e não ambígua de como o usuário e o sistema interagem para produzir uma base sólida para a evolução do sistema e validações futuras (PRESSMAN, 2006). Casos de uso são usados para descrever os requisitos funcionais de um sistema visíveis externamente. Eles descrevem as funções que o usuário quer que o sistema proveja (SCHNEIDER, WINTERS, 1998). No que diz respeito à sua classificação, essas funções podem ser essenciais (ou primárias), que descrevem regras de negócio fundamentais para atender aos objetivos dos atores, e custodiais (ou secundárias), que tratam casos de uso de manutenção de informações, como cadastros, e que são indiretamente necessários para a realização das regras de negócio (POMPILHO, 1995).

Conceitos chave associados a casos de uso são atores e sujeito. O sujeito é o sistema sob consideração ao qual o caso de uso se aplica (OMG, 2007). Um ator, por sua vez, segundo a UML, especifica um papel exercido por um usuário ou outro sistema que interage com o sujeito. Atores sempre modelam entidades externas ao sistema, que podem ser usuários humanos, hardware externo ou outros sujeitos (sistemas) (OMG, 2007).

Casos de uso podem ser associados a outros casos de uso. Os dois principais tipos de associações entre casos de uso são a extensão e a inclusão. Um relacionamento de extensão especifica que o comportamento de um caso de uso pode ser estendido pelo comportamento de outro, usualmente suplementar. Ele indica como e quando um comportamento definido em um caso de uso (que estende) pode ser inserido em outro caso de uso (estendido). A extensão ocorre em um ou mais pontos de extensão específicos definidos no caso de uso estendido. Contudo, o caso de uso estendido é definido independentemente do outro e é significativo independentemente dele (OMG, 2007).

Um relacionamento de inclusão define que um caso de uso contém o comportamento definido em outro caso de uso. Ele é usado quando existem partes comuns de um comportamento em dois ou mais comportamentos. Essa parte comum é extraída para um caso de uso separado, a ser incluído por todos os casos de uso que têm essa parte comum. Uma vez que o uso básico do relacionamento de inclusão é visando ao reúso de partes comuns, o que é deixado no caso de uso base geralmente não é completo em si, mas dependente das partes incluídas para ser significante (OMG, 2007).

A realização de um caso de uso segue um fluxo de eventos, que é uma lista de sentenças declarativas dos passos da execução do mesmo. Existem fluxos normais, que ocorrem com maior frequência, e fluxos alternativos, que ocorrem em condições especiais (SCHNEIDER,

WINTERS, 1998). Ambos podem ser melhor detalhados em diagramas de atividades (OMG, 2007).

Com respeito às restrições para a execução de um caso de uso, aplicam-se os conceitos de pré e pós-condições, que correspondem aos estados do sistema no início do caso de uso e no fim de sua execução, respectivamente (SCHNEIDER, WINTERS, 1998). Outro aspecto que pode ainda ser identificado na documentação dos casos de uso são as classes relacionadas aos mesmos, estabelecendo uma ligação rastreável entre os modelos de casos de uso e as classes identificadas no modelo estrutural do sistema.

2.3.1.3. Diagramas de Atividades

Diagramas de atividades são usados para representar processos, sendo utilizados tanto para modelar processos de negócio quanto para representar a realização de um caso de uso. Eles foram adicionados à UML relativamente tarde e segundo Störrle e Hausmann (2005) apresentavam uma integração deficiente, falta de expressividade e semântica inadequada. Na UML 2.0, novos conceitos e notações foram introduzidos, baseados em Redes de Petri, alterando a semântica do diagrama de atividades, que até a UML 1.5 era uma máquina de estados (OMG, 2007).

A modelagem de atividades enfatiza a sequência e as condições que regem os comportamentos, definindo seu fluxo de controle e o fluxo dos objetos (OMG, 2007). Para representar esses fluxos, a UML provê elementos de modelos que representam ações, objetos, estados de objetos, fluxos de ações e objetos, condições, eventos, início e fim de fluxo, exceções e agrupamento de ações por determinada característica.

2.3.1.4. Diagramas Máquinas de Estados

Diagramas de Máquinas de Estados, por sua vez, mostram as sequências de estados pelos quais um objeto pode passar ao longo de sua vida, em resposta a estímulos recebidos, juntamente com suas respostas e ações (BOOCH et al., 2005). É tipicamente um complemento de uma classe e relaciona os possíveis estados que os objetos da classe podem ter e quais eventos podem causar uma transição de um estado para outro.

2.4. REUTILIZAÇÃO DE CONHECIMENTO NA ENGENHARIA DE REQUISITOS

Quando se abordam problemas similares, tende-se a utilizar soluções semelhantes. Neste sentido, soluções vão sendo desenhadas para uma determinada classe de problemas até o ponto de serem padronizadas e documentadas para posterior busca e utilização (PRIETO-DÍAZ, 1993). A reutilização no contexto do desenvolvimento de software tem como objetivos melhorar o cumprimento de prazos, diminuir custos e obter produtos de maior qualidade (GIMENES; HUZITA, 2005), uma vez que artefatos de software podem ser reutilizados com o intuito de diminuir o tempo de construção, investindo, assim, esforço na adaptação e na reutilização de itens já construídos.

Uma vez que esses itens tenham sido desenvolvidos para reúso, o esforço de adaptação e reutilização tende a ser minimizado. Analisando o processo de ER, é possível notar que a reutilização pode ser útil, sobretudo, no reúso de requisitos de sistemas similares e de modelos (com destaque para os modelos conceituais). Contudo, na prática, na grande maioria das vezes, requisitos e modelos são construídos a partir do zero. Ou seja, requisitos iniciais são levantados junto aos interessados, modelos são construídos levando-se em conta esses requisitos iniciais, que são posteriormente refinados e novamente modelados, até se atingir um acordo com o cliente sobre o que o sistema deve prover. Entretanto, essa abordagem tem se mostrado insuficiente (FALBO et al. 2007).

Requisitos se referem a domínios e tarefas específicos. Requisitos em um domínio similar e para uma tarefa similar podem ser reutilizados (VAN LAMSWEERDE, 2000). Por isso existem diversas abordagens que aderem à análise de domínio e tarefa como fases anteriores (ou paralelas) à Engenharia de Requisitos (NARDI, 2006) (FALBO et al. 2007) (KAIYA; SAEKI, 2006) (LU et al., 2003), (CHUANG; FANG, 2007) (CHANDRASEKARAN, 1990), (MONTABERT et al. , 2005) e (LIU; FANG, 2006). Essas atividades geram e analisam modelos genéricos elaborados para reúso de conhecimento de domínio e de tarefa.

2.4.1. Reutilização de Conhecimento de Domínio

A maioria das abordagens de reutilização de conhecimento e modelos trata do reúso de conhecimento de domínio, ou seja, apoia a elaboração de modelos estruturais de um sistema, compreendendo suas classes e relacionamentos. Dentre essas abordagens destacam-se a

Análise de Domínio, o uso de Padrões de Análise e o reúso de ontologias de domínio, esta última discutida no Capítulo 3.

2.4.1.1. Análise de Domínio

A Análise de Domínio é um processo no qual elementos relevantes de um domínio são identificados e disponibilizados para serem utilizados no desenvolvimento de sistemas para esse domínio. A Análise de Domínio tem por objetivo explicitar e formalizar aspectos de domínio para auxiliar os desenvolvedores na resolução de questões relacionadas a um domínio específico (ARANGO, 1994), o que envolve extrair e empacotar informações reusáveis (VALERIO et al., 1997). Três conceitos básicos são importantes (ARANGO, 1994): (i) domínio do problema, (ii) modelo do domínio e (iii) análise e modelagem do domínio.

O domínio do problema consiste de um conjunto de itens de informação inter-relacionados, presentes em um certo contexto do mundo real (ARANGO, 1994). Já um modelo do domínio é um conjunto formal de termos, relação entre termos, regras de composição de termos, regras para raciocínio usando esses termos e regras para mapeamento de itens do domínio do problema para expressões no modelo. Define entidades, operações, eventos e relações que abstraem similaridades e regularidades em um determinado domínio, formando uma arquitetura de componentes comuns às aplicações analisadas. Serve como fonte unificada de referência para discussões sobre o domínio, auxiliando de forma direta a comunicação (ARANGO, 1994).

A análise e a modelagem do domínio compreendem um conjunto de atividades, cujo propósito é reduzir a complexidade da percepção humana sobre um determinado domínio, impondo organização aos dados adquiridos por meio de experimentos, levantamento junto a especialistas e engenharia reversa de sistemas existentes (ARANGO, 1994).

A análise de domínio pode ser vista como uma versão em meta-nível da análise requisitos, podendo ser considerada como uma atividade anterior ao desenvolvimento de software (NARDI, 2006) ou integrada a ele (VALERIO et al., 1997).

2.4.1.2. Padrões de Análise

Na engenharia de software, padrões são usados para descrever soluções de sucesso para problemas de software comuns. O uso pelos engenheiros de software de soluções já conhecidas e testadas ajuda a aumentar o nível de abstração, a produtividade e a qualidade dos projetos nas várias fases do desenvolvimento de software. Padrões de software favorecem o

reúso, através da definição e da representação explícita de um problema e da solução adotada em determinado contexto (COTA, 2004).

Padrões de análise são modelos reusáveis resultantes de atividades de análise orientada a objetos aplicadas a problemas comuns (FOWLER, 1997). Eles contêm conhecimento de domínio e experiência que podem ser usados no desenvolvimento de novos sistemas (DEVEDZIC, 2002). Esses padrões descrevem modelos que se repetem na análise de requisitos. Eles não refletem a implementação do software, mas definem a estrutura conceitual do problema (FOWLER, 1997) (DEVEDZIC, 2002)

O vocabulário oferecido por padrões ajuda a tornar claro o pensamento, além de aumentar o nível de abstração, possibilitando a discussão entre especialistas e novatos, sendo uma unidade transferível de conhecimento especializado. Essas soluções já conhecidas e testadas são mais facilmente instanciadas ou especializadas para compor a solução completa para um projeto, aumentando a produtividade e a qualidade (COTA ET AL., 1999).

Fowler (1997) aponta que padrões são frequentemente descobertos e não inventados. Apresentam, portanto, uma aproximação genérica para resolver um problema, mas eles têm que ser trabalhados e adaptados para casos específicos. Em outras palavras, eles proveem conhecimento sobre soluções bem sucedidas a problemas recorrentes no desenvolvimento de software.

Nardi (2006) propõe uma abordagem de gerência de conhecimento no apoio à Engenharia de Requisitos que fez uso de padrões de análise como elementos de reúso para análise de domínio e geração de modelos de domínio.

2.4.2. Conhecimento de Tarefa na ER

Muitos trabalhos vêm mostrando interesse em utilizar modelos de tarefas para complementar a ER tradicional. Isso porque modelos de tarefas são capazes de capturar informações não facilmente coletadas por outras técnicas de levantamento de requisitos.

No contexto do reúso de conhecimento de tarefa, uma das mais conhecidas abordagens é o CommonKADS (BREUKER; VAN DE VELDE, 1994). Além dessa abordagem, outras linhas de pesquisa vêm sendo consideradas para o reúso de modelos que capturam tarefas, como a análise de tarefa (LIU; FANG, 2006) (CHUANG; FANG, 2007) e o uso de modelos de processos de negócios (ESTRADA et al., 2002) (CARDOSO et al., 2008). Outros trabalhos ainda consideram heurísticas diretas para mapear modelos de tarefas existentes para modelos comportamentais utilizados no desenvolvimento de software. Algumas dessas abordagens são brevemente apresentadas nas subseções que seguem. Há, ainda, alguns

trabalhos que tratam do reúso de ontologias de tarefa na ER, os quais são discutidos no Capítulo 3.

2.4.2.1. CommonKADS

CommonKADS é uma metodologia que apoia o desenvolvimento de sistemas baseados em conhecimento em suas diversas fases e aspectos. CommonKADS inclui uma biblioteca de tarefas genéricas que define vários modelos para captura do conhecimento: modelos de organização, de tarefa, de agentes, de comunicação, de experiência e de projeto (BREUKER; VAN DE VELDE, 1994). Os quatro primeiros capturam o contexto da atividade de solução do problema. O modelo de projeto descreve a execução computacional da tarefa. Finalmente, o modelo de experiência é o modelo central da metodologia CommonKADS. Ele descreve o conhecimento e o raciocínio envolvidos na realização de uma tarefa. Esse modelo divide o conhecimento da aplicação em três níveis: nível do domínio (conhecimento do domínio relevante para a execução da tarefa), nível de inferência (estabelece como o conhecimento de domínio é usado nos passos de raciocínio) e nível de tarefa (trata da decomposição em subtarefas e a sua ordem de execução) (BREUKER; VAN DE VELDE, 1994). A metodologia especifica um processo pelo qual sistemas baseados em conhecimento são desenvolvidos e provê suporte à reutilização por permitir o reúso de modelos definidos anteriormente (SCHREIBER et al., 1994).

2.4.2.2. Análise de Tarefa

Muitos consideram uma fase anterior ou integrada à ER que identifica e trata de aspectos específicos de tarefa, como (LU et al., 2003),(CHUANG; FANG, 2007) (CHANDRASEKARAN, 1990), (MONTABERT et al. , 2005) e (LIU; FANG, 2006). Isso é importante, pois desenvolvedores normalmente se confrontam com novas tarefas que eles devem entender e modelar. A natureza da tarefa precisa ser analisada a fundo, muitas vezes em um nível de granularidade fino, contendo as diversas informações necessárias (MIZOGUCHI et al., 1995a).

A análise de tarefa é uma atividade de modelagem. O analista identifica o problema, bem como as entradas e saídas do processo de solução do problema. O foco inicial da análise de tarefa é concentrar em conceitos mais importantes sobre os quais o modelo de tarefa precisa ser construído (LIU; FANG, 2006). O resultado da análise de tarefa pode ser uma especificação formal dos problemas, mas frequentemente ela é uma descrição informal inicial de um problema (MIZOGUCHI et al., 1995a).

Ikeda e outros. (1998) apontaram que a análise de tarefa é feita de acordo com dois passos principais: (1) identificação grosseira e (2) análise detalhada da tarefa. Com base em fontes de conhecimento, a identificação grosseira da estrutura da tarefa é um problema de classificação, enquanto a análise detalhada da tarefa se refere à interação com especialistas do domínio e a articulação de como executar essa tarefa (CHUANG; FANG, 2007).

O propósito da análise da tarefa é decompor tarefas da vida real em um número de tarefas genéricas e associar estas a métodos de solução de problemas apropriados. Juntos, métodos e tarefas formam modelos de tarefa (VAN HEIJST et. al, 1997).

A análise de tarefa é uma atividade contínua no sentido de que desenvolvedores devem ser preparados para revisar e estender seus modelos da tarefa à medida que eles têm mais percepções sobre o problema (MIZOGUCHI et al., 1995a).

2.4.2.3. Reutilização de Modelos de Processos de Negócio

A modelagem de processos de negócio é um instrumento poderoso na análise da organização, servindo de base para sua estruturação e permitindo a visualização de melhorias efetivas. Esta representação do modo de funcionamento de uma organização pode ser constituída por vários modelos, capturando diversas informações relevantes. Cada um dos modelos é formado por um ou vários diagramas, que visam mostrar uma parte específica da estrutura da organização (modelos estruturais) ou situação do negócio (modelos comportamentais) (ERIKSSON, PENKER, 2000).

É essencial entender os processos e objetivos de negócio das organizações para construir os sistemas capazes de atendê-los da maneira apropriada. Em um cenário em que a organização desenvolve seu modelo de negócio como uma forma de perceber possíveis melhorias de seus serviços, é possível, e interessante, considerar as informações nele presentes para auxiliar o levantamento de requisitos de seus sistemas (KNIGHT, 2004).

Martins (2001) define uma metodologia de levantamento de requisitos baseada na Teoria da Atividade (KAPTELININ, 1996). O foco do trabalho é no uso de diagramas de atividades como unidades de especificação de requisitos de sistemas. O trabalho mostra que os diagramas de atividades são capazes de descrever um cenário com um conjunto mais rico de informações do que os diagramas de casos de uso.

Em (ESTRADA et al., 2002), faz-se uso do *framework i** (YU; MYLOPOULOS, 1993) de modelagem para modelar processos de negócio e, a partir desses modelos, geram-se modelos de casos de uso.

Dando continuidade aos dois trabalhos citados anteriormente, Cruz e outros (2004) descrevem diretrizes de mapeamento que transformam diagramas de atividades (usados como em (MARTINS, 2001)) em modelos *i**. Constata-se que modelos de atividades da Teoria da Atividade, além de servirem para guiar o processo de geração dos modelos organizacionais da abordagem *i**, podem ser usados como documento complementar de requisitos para um melhor entendimento do contexto.

Cardoso e outros (2008) falam da experiência com a utilização de modelos de processos de negócio na Engenharia de Requisitos. A modelagem de processos complementa as práticas convencionais de Engenharia de Requisitos, auxiliando o cliente a adquirir maturidade acerca da complexidade do seu próprio negócio e revelando o grau de adequação dos requisitos levantados aos processos da organização. Segundo eles, seguindo a abordagem baseada em modelos de processos, é possível obter um conjunto de requisitos mais completo, correto, rastreável e que reflete consistentemente a visão dos vários interessados (*stakeholders*).

2.4.2.4. Modelos de Tarefa e Diagramas de Casos de Uso

Seguindo a idéia do reúso de modelos de tarefas genéricas, alguns trabalhos definem heurísticas para mapear diretamente elementos de modelos de tarefas para descrições e diagramas de casos de uso, que são a ferramenta mais comumente utilizada para capturar comportamento nas primeiras fases da Engenharia de Requisitos.

Considerando uma representação própria para o conhecimento de tarefa, chamada Descrição de Solução de Problemas (DSP), ZLOT et al. (2002) consideraram, dentre outros, que:

- A descrição de um caso de uso pode ser definida utilizando uma descrição do problema em linguagem natural (nível verbal da DSP).
- O nome do caso de uso deve estar relacionado com a função do sistema que está sendo especificada, devendo ser baseada no nome da própria tarefa.
- Um fluxo de eventos de um caso de uso pode ser obtido a partir da ordem em que as subtarefas são chamadas durante o fluxo de controle da tarefa..

Já em (CRUZ, 2004), definem-se heurísticas para a derivação de casos de uso a partir de modelos de processos de negócios que utilizam diagramas de atividades em sua notação. Algumas dessas heurísticas são:

- As raias (partições na UML 2.0 (OMG, 2007)) caracterizam os responsáveis que efetivamente executam as ações nelas contidas. Esses responsáveis devem ser identificados como atores em potencial.

- Atividades devem ser representadas na forma de casos de uso distintos, i.e., cada atividade deve dar origem a um caso de uso.
- Objetos interagindo com atividades devem dar origem a novos casos de uso que permitam a consulta às suas informações.
- Os casos de uso originados de atividades concorrentes devem ser representados isoladamente.
- Os atores identificados devem ser relacionados a todos os casos de uso principais originados de atividades de sua raia.

Diferentemente das abordagens anteriormente apresentadas, Lu e outros (1998) realizaram um estudo explorando a semântica comum de modelos de tarefa e modelos comportamentais de sistemas. Inicialmente foi proposta uma abordagem para construção automática de modelos de tarefa a partir de diagramas de casos de uso e de sequência. Em 2003, os mesmos autores e mais um pesquisador (LU et al., 2003) apresentaram o estudo inverso, considerando uma abordagem de geração de modelos comportamentais de sistemas a partir de modelos de tarefas. Por possuírem uma base semântica comum, modelos de tarefa foram considerados como base para a geração de modelos comportamentais do sistema, permitindo a integração de modelos de tarefas no processo orientado a objetos. LU et al. (2003) sugeriram, dentre outros, o seguinte mapeamento entre elementos de modelos de tarefas e casos de uso:

- Tarefas compostas são diretamente mapeadas em casos de uso. Atributos da tarefa – precondição, saídas, restrições, ator e comentários – podem ser usados para derivar a documentação.
- Os atores estão implícitos na definição das tarefas, logo o modelo de tarefas provê informações para derivar atores.
- Quando há uma ligação com condições entre duas tarefas, pode-se derivar um relacionamento de extensão. A direção é inversa à da ligação das tarefas.

2.5. CONSIDERAÇÕES FINAIS

Este capítulo apresentou os alguns conceitos relativos à Engenharia de Requisitos (ER), com destaque para o processo de ER, suas atividades principais e seus problemas. Foram brevemente discutidas, também, a atividade de modelagem e a Linguagem de Modelagem Unificada (UML), que é uma linguagem de modelagem bem difundida, compreensível e

muito utilizada na ER. Esse é um fato relevante para fundamentar a escolha dessa linguagem como base da abordagem proposta, conforme discutido no Capítulo 4.

Por fim, discutiu-se a reutilização na ER, com destaque para o reúso de conhecimento de tarefa. Diante das abordagens de reutilização de conhecimento de tarefa citadas, notou-se que trabalhos estão ressaltando a crescente importância de modelar tarefas como parte dos esforços da ER. Nessa linha, este trabalho propõe uma abordagem para reutilizar conhecimento existente sobre tarefas com o objetivo de poupar esforços e aumentar a qualidade, conforme discutido no Capítulo 5.

CAPÍTULO 3. ONTOLOGIAS

O objetivo deste capítulo é apresentar os principais conceitos relacionados a ontologias, seus tipos, métodos e linguagens para sua construção e aplicações. Dá-se maior foco a ontologias de tarefa, pois é o objeto central de estudo deste trabalho, e também a ontologias de fundamentação, que fornecem uma base para a construção tanto de ontologias de domínio quanto de tarefa.

3.1. INTRODUÇÃO

Ontologia é um conceito já há muito tempo utilizado pela Filosofia, definido como o estudo de tipos de coisas que existem (CHANDRASEKARAN et al., 1999). Posteriormente, a Inteligência Artificial também passou a adotá-lo, com uma interpretação um pouco diferente, mais voltada para a modelagem de conhecimento. Passou-se a considerar ontologia também como um artefato, constituído de um vocabulário de termos organizados em uma taxonomia, suas definições e um conjunto de axiomas formais usados para criar novas relações e para restringir as suas interpretações (GUIZZARDI, 2005) (GUARINO, 1998).

Segundo Guarino (1998), uma ontologia define um vocabulário específico usado para descrever uma certa realidade e um conjunto de decisões explícitas, de forma a fixar de forma rigorosa o significado pretendido para o vocabulário. Ela captura os conceitos e relações em determinado domínio e um conjunto de axiomas, que restringem a sua interpretação.

Ontologias envolvem a descrição de conceitos em um domínio específico de conhecimento, com suas propriedades e restrições. Servem, portanto, como meio para facilitar a comunicação, integração, busca, armazenamento e representação do conhecimento (O'LEARY, 1998) e, por isso, sua utilização é importante em diferentes áreas que fazem uso maciço de conhecimento.

Este capítulo aborda o referencial teórico sobre ontologias utilizado como base para o desenvolvimento deste trabalho e está organizado da seguinte forma: a Seção 3.2 fornece definições da literatura para o termo ontologia, a Seção 3.3 apresenta os tipos de ontologias e a classificação que se optou seguir, bem como a descrição de seus tipos. Na Seção 3.4 é apresentada uma discussão sobre conceituação e linguagens de representação de ontologias. Na Seção 3.5 são apresentados usos para ontologias e, por fim, na Seção 3.6 são feitas as considerações finais do capítulo.

3.2. DEFINIÇÕES

Uma definição muito citada para o termo ontologia é a sugerida por Gruber (1993): “uma ontologia é uma especificação formal e explícita de uma conceituação compartilhada”. “Conceituação” se refere a um modelo abstrato de uma realidade que identifica seus conceitos relevantes. “Explícita” significa que os conceitos usados e as restrições do seu uso são definidos explicitamente. “Formal” é referente ao fato de ser passível de entendimento por máquinas. “Compartilhada” reflete que uma ontologia captura o conhecimento consensual aceito por uma comunidade (DING, 2001).

Guarino (1997) discute a definição de Gruber (1993) sob a luz de outras presentes na literatura, a saber (GRUBER, 1995), (WIELINGA; SCHREIBER 1993), (ALBERTS, 1993), (VAN HEIJST et al., 1997), (SCHREIBER et al., 1995) e (GUARINO; GIARETTA, 1995), e propõe uma definição mais satisfatória segundo seu ponto de vista: “Uma ontologia é uma descrição parcial e explícita de uma conceituação”. Para ele, portanto, o grau de especificação de uma conceituação depende do propósito desejado para uma ontologia.

O termo ontologia é, às vezes, usado para referenciar o corpo do conhecimento que descreve algum domínio de conhecimento de senso comum. Uma ontologia provê um vocabulário de representação, dado por um conjunto de termos com os quais se descreve os fatos relativos a esse domínio (CHANDRASEKARAN et al., 1999). Ela define, com diferentes níveis de formalidade, o significado dos termos e as relações entre eles (GÓMEZ-PÉREZ, BENJAMINS, 1999). É vista, portanto, como uma descrição de conceitos em um domínio específico de conhecimento, suas propriedades, que descrevem características, seus atributos e restrições, bem como seus possíveis relacionamentos (NOY; MCGUINNESS, 2001) (CHANDRASEKARAN et al., 1999).

3.3. CLASSIFICAÇÃO DE ONTOLOGIAS

Existem diversas classificações para ontologias. Segundo Uschold (1996), ontologias podem ser classificadas em três dimensões. Segundo o *grau de formalidade*, que depende da forma como é descrita a ontologia, considerando desde a linguagem natural até uso de uma semântica formal, teoremas e provas de propriedades, ontologias podem ser classificadas em: altamente informal, semi-informal, semiformal, rigorosamente formal. De acordo com o *propósito*, uma ontologia pode ser desenvolvida para comunicação, interoperabilidade ou apoio à engenharia de sistemas. Finalmente, considerando a *natureza do assunto* que a ontologia está tratando, tem-se: (i) ontologias de domínio, que expressam conceituações para

domínios específicos, (ii) ontologias de tarefa, método e resolução de problemas, quando o assunto é a solução de um problema ou execução de uma tarefa, e (iii) ontologias de representação ou metaontologias, quando trata de uma linguagem de representação de conhecimento (DING, 2001).

Van Heijst e outros (1997) classificam ontologias em duas dimensões. A primeira dimensão trata da *estrutura da conceituação*, podendo ser: (i) ontologia terminológica, como um dicionário que define os termos a serem usados para representar conhecimento, (ii) ontologia de informação, que especifica a estrutura de registros de banco de dados, e (iii) ontologia de modelagem de conhecimento, que especifica a conceituação do conhecimento. A segunda dimensão trata da *natureza da conceituação* e classifica as ontologias como: (i) ontologias de representação, as quais explicam as conceituações que fundamentam formalismos de representação de conhecimento, (ii) ontologias de domínio, que expressam conceituações que são específicas para um domínio particular, (iii) ontologias genéricas, que são similares às de domínio, porém seus conceitos são genéricos para vários domínios, e (iv) ontologias de aplicação, que contêm as definições que são necessárias para modelar o conhecimento requerido por uma particular aplicação.

Já Guarino (1997) considera duas dimensões para classificação de ontologias. A primeira considera o *nível de detalhes* usado para caracterizar a conceituação, diferenciando ontologias como: (i) ontologia de documentação (ou *on-line*), quando for simples como um dicionário, (ii) ontologia divisível (ou *off-line*), quando utiliza teorias mais sofisticadas para descrever seus termos. A segunda dimensão trata do *nível de dependência e da natureza* da conceituação, classificando ontologias como genérica, de representação, de domínio ou de aplicação. Em (GUARINO, 1998), o mesmo autor sugere uma classificação quanto à *generalidade*, na qual, como mostra a Figura 3.1, ontologias podem ser classificadas em: (i) ontologias de fundamentação ou de topo, que descrevem conceitos muito gerais, como espaço, tempo, problema, objeto, evento, ação etc., (ii) ontologias de domínio, que descrevem o vocabulário relacionado a um domínio genérico como, por exemplo, medicina, direito etc., (iii) ontologias de tarefa, que descrevem o vocabulário relacionado a uma tarefa genérica, como, por exemplo, diagnose, venda etc. e (iv) ontologias de aplicação, que descrevem conceitos dependentes de um domínio e uma tarefa particulares, os quais são, frequentemente, especializações de ontologias relacionadas. Essa classificação é considerada como base para este trabalho e, por isso, seus tipos são melhor detalhados nas subseções seguintes.

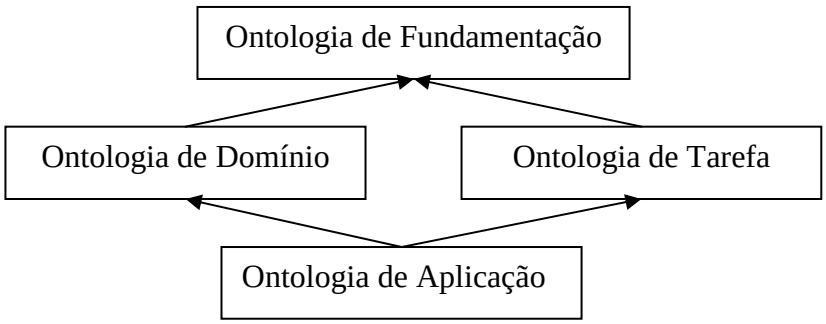


Figura 3.1: Classificação de ontologias proposta por Guarino (1998).

3.3.1. Ontologias de Fundamentação

No nível mais geral de abstração, a preocupação é com as categorias que se aplicam às diversas áreas de conhecimento. Chama-se este nível de descrição de Ontologia Geral, Ontologia de Altonível ou Ontologia de Fundamentação (HERRE et al., 2006). Essas ontologias são sistemas de categorias filosoficamente bem fundamentados e independentes de domínio (GUIZZARDI et al., 2008b).

Ontologias de fundamentação servem como base para o estabelecimento de consenso e negociação entre humanos. Elas têm sido utilizadas com sucesso para melhorar a qualidade de linguagens de modelagem e modelos conceituais. (GUIZZARDI, 2005).

Algumas ontologias de fundamentação existentes são: DOLCE (*Descriptive Ontology for Linguistic and Cognitive Engineering*) (BOTTAZZI; FERRARIO, 2006), GFO (*General Formal Ontology*) (HERRE et al., 2006), SUMO (*Suggested Upper Merged Ontology*) (NILES; PEASE, 2001), UFO (*Unified Foundational Ontology*) (GUIZZARDI, 2005) e Cyc (LENAT; GUHA, 1990)

Neste trabalho, utiliza-se a ontologia de fundamentação UFO, assim suas partes e principais conceitos são descritos a seguir.

3.3.1.1. Ontologia de Fundamentação Unificada

A Ontologia de Fundamentação Unificada (*Unified Foundational Ontology* – UFO) tem sido desenvolvida baseada em um número de teorias das áreas de Ontologias Formais, Lógica Filosófica, Filosofia da Linguagem, Linguística e Psicologia Cognitiva (GUIZZARDI et al. 2008b). A UFO tenta suprir as limitações na habilidade de capturar os conceitos básicos de Linguagens de Modelagem Conceitual e de outras ontologias de fundamentação, como a GFO e a DOLCE. A proposta da UFO é justamente unificar essas ontologias, aproveitando suas características positivas e sanando as limitações detectadas (GUIZZARDI; WAGNER, 2005).

A UFO tem sido aplicada com sucesso para avaliar, (re) projetar e integrar os modelos de linguagens de modelagem conceitual, bem como para prover semântica de mundo real para seus elementos de modelo (GUIZZARDI et al. 2008b) (GUIZZARDI, WAGNER, 2005). Por agregar conceitos relativos a eventos, objetos, agentes e recursos, ela pode ser utilizada no contexto de tarefas para prover semântica aos modelos que capturam o conhecimento dinâmico de uma tarefa (GUIZZARDI; WAGNER, 2005).

A UFO foi proposta inicialmente em (GUIZZARDI, 2004) e é dividida em três partes complementares: a UFO-A é uma ontologia de indivíduos duradouros (*endurants*) e é o cerne da UFO; a UFO-B é uma ontologia de eventos (*perdurants*); por fim, a UFO-C é uma ontologia de entidades sociais, construída sobre as partes A e B da UFO. As descrições apresentadas na sequência são uma síntese das distinções feitas em (GUIZZARDI, 2005), (GUIZZARDI; WAGNER, 2005), (GUIZZARDI et al. 2008a) e (GUIZZARDI et al. 2008b). Neste texto os conceitos da UFO são apresentados na língua portuguesa, seguidos dos correspondentes termos na língua inglesa entre parênteses. Nos diagramas que apresentam os modelos da UFO, contudo, optou-se por apresentar os termos em inglês, de modo a facilitar uma ligação com as principais referências à UFO, quase todas elas escritas em inglês.

a) UFO-A

Todos os elementos da UFO especializam o conceito fundamental da UFO-A denominado ***Entidade*** (*Entity*). A distinção principal da UFO é entre as categorias de ***Universais*** (*Universal*) e ***Indivíduos*** (*Particular*). A primeira se refere a tipos de entidades, ou seja, padrões de características que podem ser percebidos em diferentes ***indivíduos***. Estes, por sua vez, são entidades que existem na realidade e possuem uma identidade única (GUIZZARDI et al., 2008b). Cada ***indivíduo*** é, portanto, instância de algum ***universal***, como se pode ver na Figura 3.2.

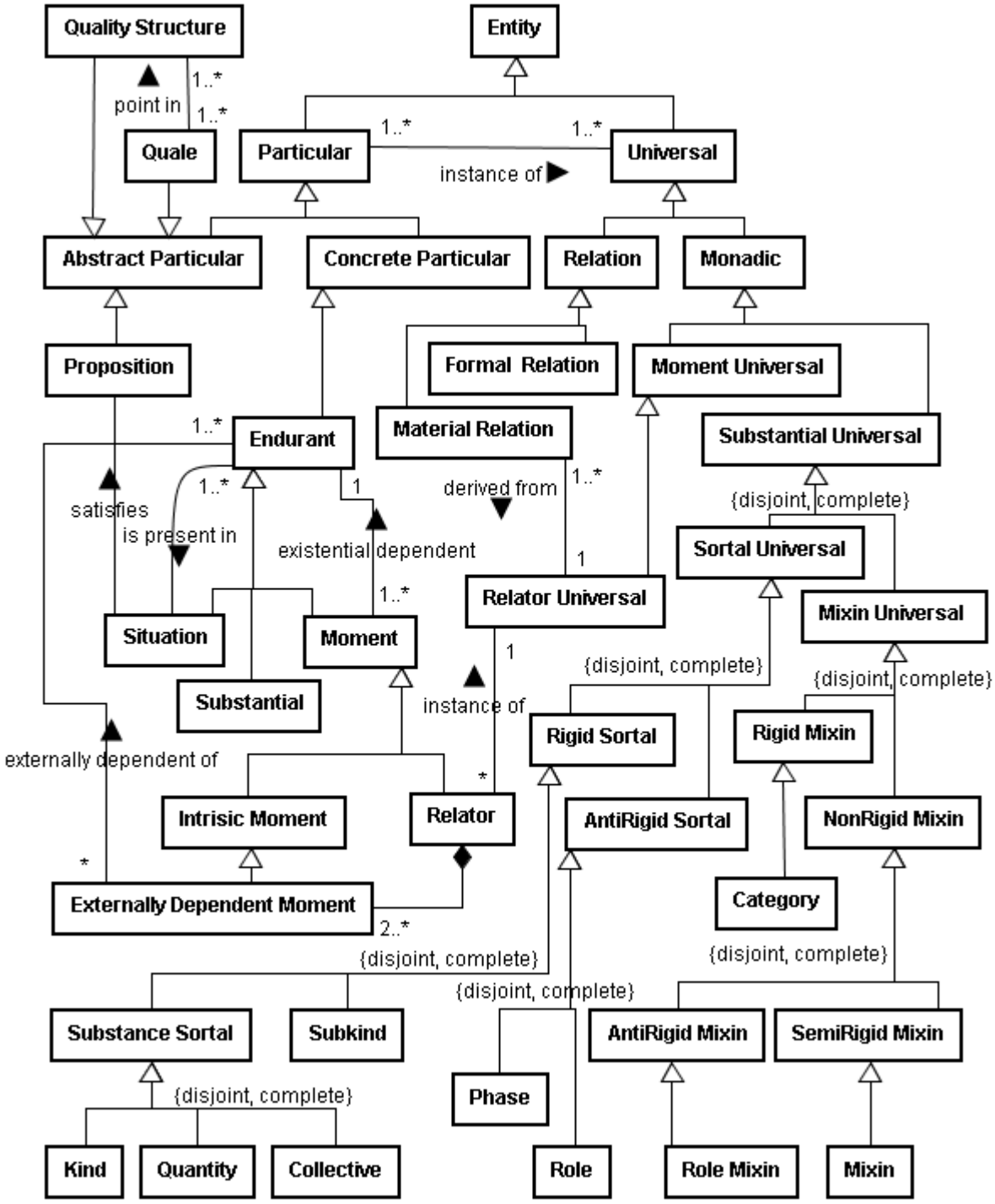


Figura 3.2: UFO-A

No que se refere a ***indivíduos*** (*Particular*), existem ***indivíduos concretos*** (*Concrete Particular*) e ***abstratos*** (*Abstract Particular*). ***Indivíduos duradouros*** (*Endurant*) são tipos de ***indivíduos concretos*** que podem ser categorizados em: ***substanciais*** (*Substantial*), ***modos*** (definidos como *Mode* em (GUIZZARDI et al., 2008b) e *Moment* em (GUIZZARDI et al., 2008a)) e ***situações*** (*Situation*).

Os ***substanciais*** são indivíduos existencialmente independentes. Exemplos incluem indivíduos duradouros do senso comum, tais como uma pessoa, um cachorro, uma casa, Tom Jobim e Os Beatles.

Os ***modos***, em contraste, denotam a instanciação de uma propriedade. Um ***modo*** é um indivíduo que só pode existir em outro indivíduo e é dito ser inerente a esse indivíduo, ou ainda, existencialmente dependente. Exemplos típicos de modos são uma cor, uma carga elétrica, um sintoma etc. A dependência existencial também pode ser usada para diferenciar ***modos intrínsecos*** e ***relacionais***. ***Modos intrínsecos*** (*Intrinsic Moment*) são dependentes de um único indivíduo, como uma cor, uma dor de cabeça e uma temperatura. ***Modos relacionais*** (*Relator*), por sua vez, dependem de vários indivíduos e têm o poder de conectá-los, tal como um emprego, um tratamento médico e um casamento.

Situações (*Situation*) são entidades complexas, constituídas possivelmente por vários indivíduos duradouros (incluindo outras situações), sendo tratadas como um sinônimo para o que é chamado na literatura de "estado de coisas" (*state of affairs*), ou seja, uma porção da realidade que pode ser compreendida como um todo, por exemplo "João está gripado e com febre". Tomando por base a noção de situação, define-se a relação "estar presente em" (*is present in*) entre indivíduos duradouros e as situações que eles constituem. No exemplo anterior, pode-se dizer que o substancial João e seus modos febre e gripe estão presentes na situação "João está gripado e com febre" (GUIZZARDI et al., 2008b).

Os ***indivíduos abstratos*** podem ser ***estruturas de qualidade*** (*Quality Structure*), um ***ponto na estrutura de qualidade*** (*Quale*) ou uma ***proposição*** (*Proposition*) (GUIZZARDI, et al., 2008b). Uma ***estrutura de qualidade*** pode ser entendida como uma estrutura de medição (ou um espaço de valores) em que qualidades individuais podem tomar seus valores. Ou seja, uma qualidade está associada a uma ***estrutura de qualidade***. Por exemplo, a qualidade 'peso' está associada a um espaço de valores que é uma estrutura linear isomórfica ao eixo positivo dos números reais (GUIZZARDI, 2005). Um ***ponto na estrutura da qualidade*** (*Quale*) é uma percepção ou concepção de uma propriedade intrínseca.

Indivíduos (*Particular*) são instâncias de ***Universais*** (*Universal*), que são especializados em ***Universal Unário*** (*Monadic*) e ***Relação*** (*Relation*). O primeiro aplica-se a

um indivíduo e o segundo, a dois ou mais indivíduos. Há categorias de ***universais de substância*** (*Substantial Universal*) e ***universais de modo*** (*Moment Universal*), ambos especializações de ***Universal Unário***.

Universais de substância podem ser ***Universais Sortais*** (*Sortal Universal*) ou ***Universais Mistos*** (*Mixin Universal*). ***Universal Sortal*** é o tipo de universal de substância que provê um princípio de identidade para suas instâncias, permitindo julgar se dois particulares são o mesmo. ***Universal Misto***, por sua vez, é o tipo de universal de substância que cobre conceitos com diferentes princípios de identidade (GUIZZARDI, 2005).

Universais de substância podem ser rígidos, não rígidos e antirrígidos. Um universal de substância rígido é aquele que necessariamente aplica-se a todas as suas instâncias, i.e., em todos os mundos possíveis. Por exemplo, uma pessoa é sempre uma pessoa em qualquer configuração possível de mundo. Um universal de substância não rígido é aquele não se aplica necessariamente a pelo menos uma de suas instâncias. Por fim, um universal de substância antirrígido é aquele que necessariamente não se aplica a todas as suas instâncias. Por exemplo, estudante é antirrígido, pois necessariamente não se aplica a todas as pessoas. ***Espécie*** (*Kind*), ***Subespécie*** (*Subkind*), ***Coletivo*** (*Collective*) e ***Quantidade*** (*Quantity*) são sortais rígidos. ***Fase*** (*Phase*) e ***Papel*** (*Role*) são sortais antirrígidos.

Espécie (*Kind*) é um sortal rígido que provê o princípio de identidade às suas instâncias. ***Espécies*** podem ser especializadas em outros subtipos rígidos, ditos ***subespécies*** (*Subkind*) que herdam seus princípios de identidades providos. Por exemplo, Pessoa é uma ***espécie*** e se pode definir Homem e Mulher como suas ***subespécies***. Um grupo de pessoas, por sua vez, é uma instância do tipo ***Coletivo*** (*Collective*). Esse tipo, assim como ***Espécie***, provê princípio de identidade a suas instâncias, porém, são conjuntos de objetos que exercem o mesmo papel. Por fim, ***quantidades*** (*Quantity*), ao contrário dos demais tipos de sortais rígidos, carecem de princípios de individualização e contagem.

Pode-se dizer que um indivíduo que é instância de uma ***Espécie, Subespécie***, ***Coletivo*** ou ***Quantidade*** é necessariamente instância deste (em todos os mundos possíveis). Já o tipo ***Fase*** é um sortal instanciado em determinado mundo ou período de tempo, mas não necessariamente em todos. Por exemplo, Criança, Adolescente e Adulto são ***fases*** da ***Espécie*** Pessoa. Também ***Papel*** é um tipo de ***Sortal*** instanciado eventualmente, mais precisamente na participação em um evento ou numa determinada relação. Por exemplo, Mãe é um papel para a subespécie Mulher mediante a existência da relação de maternidade com uma instância do papel Filho da espécie Pessoa. (GUIZZARDI, 2005).

Categoria (*Category*) é um tipo de ***Universal Misto*** (*Mixin Universal*) que classifica as entidades que pertencem a espécies diferentes, mas que compartilham uma propriedade comum essencial (ou seja, uma propriedade que eles não podem deixar de ter). Por exemplo, uma categoria Entidade Racional pode ser especializada pelas espécies Pessoa e Agente Artificial (GUIZZARDI, 2005).

Retomando aos ***Universais*** têm-se as ***Relações,*** que são entidades que aglutinam outras entidades. A UFO, baseada na Filosofia, considera dois tipos de relações: ***Materiais*** (*Material Relation*) ou ***formais*** (*Formal Relation*). ***Relações formais*** acontecem entre duas ou mais entidades diretamente, sem nenhum outro indivíduo intermediando. Relações materiais, por outro lado, são aquelas relações entre indivíduos intermediadas por ***modos relacionais*** (*Relator*), que são indivíduos com o poder de conectar (mediar) outros indivíduos. Por exemplo, um tratamento médico conecta um paciente a uma unidade médica, uma matrícula conecta um estudante a uma instituição de ensino, o casamento de João e Maria, que conecta essas duas pessoas etc.

No exemplo do casamento de João e Maria, há um modo relacional do tipo *casamento* que media João e Maria. Além disso, João adquire várias propriedades em virtude de estar casado com Maria, bem como responsabilidades legais no contexto dessa relação. Essas novas propriedades adquiridas são instanciadas em modos intrínsecos de João e, por conseguinte, são existencialmente dependentes dele. Entretanto, esses modos também dependem da existência de Maria. Esse tipo de modo é chamado de ***modo externamente dependente*** (*Externally Dependent Moment*), i.e, modos intrínsecos que são inerentes a um único indivíduo, mas que são existencialmente dependentes de outros (possivelmente vários) indivíduos. O modo relacional Casamento, neste caso, é a soma de todos os modos externamente dependentes que João e Maria adquirem em virtude de estarem casados um com o outro (GUIZZARDI et al., 2008a).

b) UFO-B

A UFO-B, como mostra a Figura 3.3, diferencia explicitamente ***Eventos*** (*Perdurant*) e ***Indivíduos duradouros*** (*Endurant*) em termos de suas respectivas relações com o tempo. Diz-se que ***indivíduos duradouros*** estão inteiramente presentes em qualquer instante do tempo em que estiverem presentes, isto é, se em uma circunstância *c1*, um indivíduo duradouro *I* possui a propriedade *p1* e em uma circunstância *c2* esse mesmo indivíduo possui a propriedade *p2* (possivelmente incompatível com *p1*), ele continua sendo o mesmo indivíduo *I* em ambas as circunstâncias.

Eventos (*Perdurant* ou *Event*) são indivíduos compostos de partes temporais. Eles se estendem no tempo acumulando partes temporais. São exemplos de eventos: uma conversa, uma partida de futebol, a execução de uma sinfonia e um processo de negócio. Em qualquer momento em que um evento está presente, apenas algumas de suas partes temporais estarão presentes. Como uma consequência, eventos não podem sofrer mudanças no tempo no sentido genuíno, uma vez que nenhuma de suas partes temporais mantém sua identidade ao longo do tempo (GUIZZARDI et al., 2008b).

Eventos transformam uma ***situação*** (*Situation*) para outra na realidade, ou seja, eles podem alterar o estado de coisas da realidade de um estado (***pré-estado*** (*pre-state*)) para outro (***pós-estado*** (*pos-state*)), conforme os relacionamentos da Figura 3.3. ***Eventos*** são entidades ontologicamente dependentes no sentido de, para existirem, dependerem existencialmente de seus participantes. Um evento pode ser composto da ***participação*** (*Participation*) individual de cada um de seus participantes (***substantiais***). Cada uma dessas ***participações*** é por si própria um evento que pode ser ***complexo*** (*Complex Event*) ou ***atômico*** (*Atomic Event*), mas que existencialmente depende de um único ***substancial***. Em UFO-B, ser atômico e ser instantâneo são noções ortogonais, i.e., participações atômicas podem se estender no tempo, bem como eventos instantâneos podem ser compostos de múltiplas participações (instantâneas).

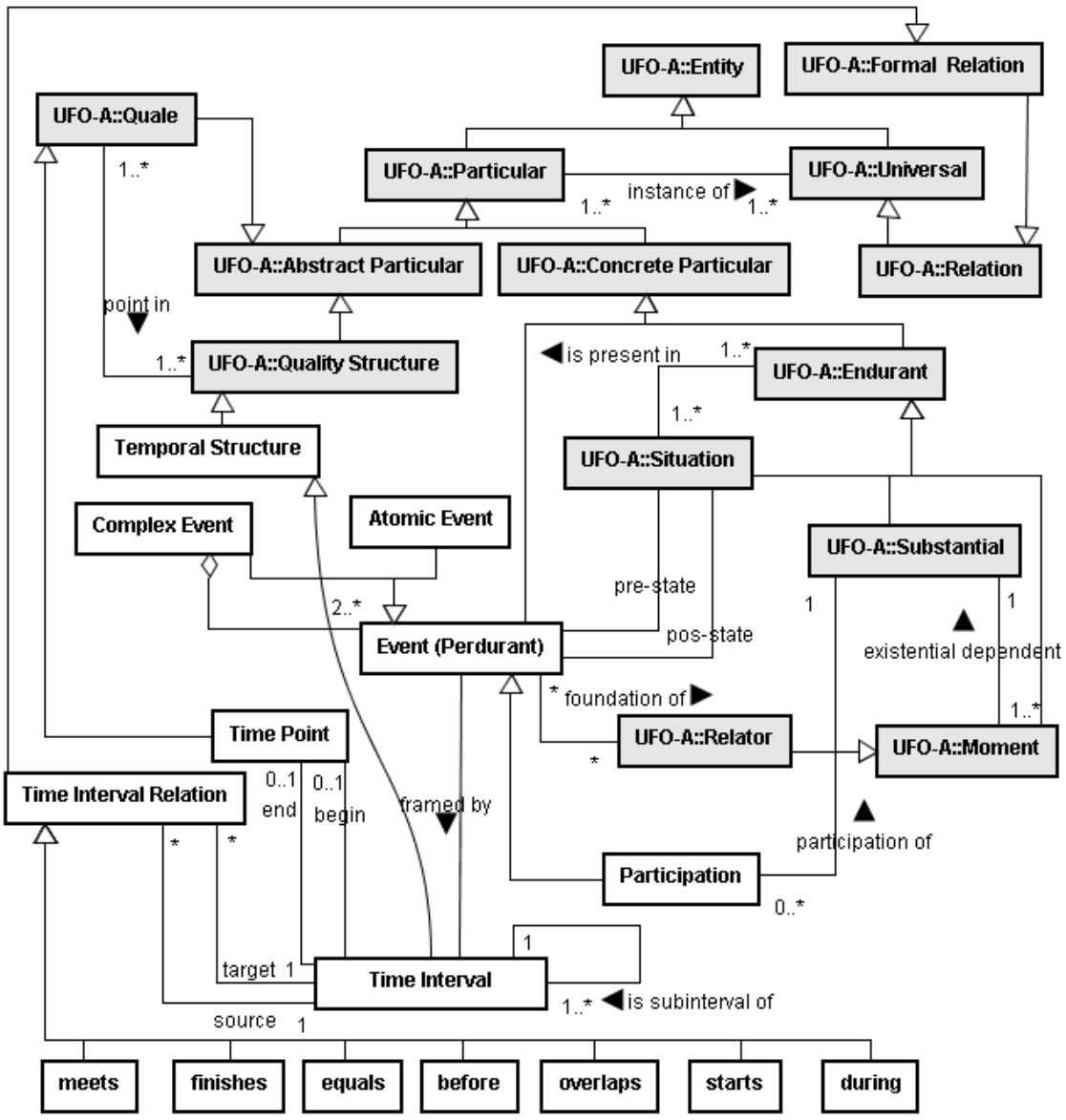


Figura 3.3: UFO-B

Análogo ao que foi discutido para ***indivíduos duradouros*** (*Endurant*) na UFO-A, os modos temporais de eventos têm seus valores (***qualia*** (*Quale*)) obtidos pela sua projeção em uma ***estrutura de qualidade*** (*Quality Structure*), que é aqui representada pela ***estrutura temporal*** (*Temporal Structure*), mais especificamente um ***intervalo temporal*** (*Time Interval*). Assume-se, ainda, que o espaço conceitual de tempo é uma estrutura composta de ***intervalos temporais*** que, por sua vez, são compostos de ***instantes*** (*Time Point*). Dois dentre esses instantes merecem destaque, pois definem os limites do intervalo: os instantes de ***início*** (*begin*) e ***fim*** (*end*). ***Intervalos temporais*** possuem relações chamadas ***relações temporais*** (*Time Interval Relation*).

Instantes podem ser representados como números reais e intervalos temporais como conjuntos de números reais. Entretanto, outras estruturas temporais, tais como tempo linear, ramificado, paralelo e circular, também podem ser usadas (GUIZZARDI et al., 2008). No contexto deste trabalho, consideram-se as ditas *relações entre intervalos de Allen* (ALLEN, 1983), a partir das quais as correspondentes relações entre eventos podem ser derivadas (GUIZZARDI et al., 2008) a partir da comparação entre um intervalo ***origem*** (*source*) a outro ***destino*** (*target*). A Figura 3.4 apresenta todas as possíveis ***relações temporais*** nessa estrutura, que são sete: (i) ***precede*** (*before*), quando a origem inicia e termina antes do destino, ou seja, um intervalo ocorre antes do outro, (ii) ***encontra*** (*meets*), quando o instante de fim da origem é igual ao de início do intervalo destino, logo um intervalo inicia imediatamente após o outro, (iii) ***sobrepõe*** (*overlaps*), quando o início da origem é antes do início do destino e o fim da origem é depois do início e antes do fim do destino, dessa forma há sobreposição da parte final do intervalo de origem com a inicial do outro, (iv) ***inicia*** (*starts*), quando ambos iniciam no mesmo instante, (v) ***durante*** (*during*), quando o início da origem é posterior ao início do destino e o fim é anterior ao fim do destino, dessa forma, um intervalo inteiro sobrepõe parte do outro, (vi) ***termina*** (*finishes*), quando o instante de fim de ambos são iguais, e (vii) ***equivale*** (*equals*), quando o início e o fim da origem são os mesmos do destino, ou seja, os dois eventos possuem intervalos de tempo iguais.

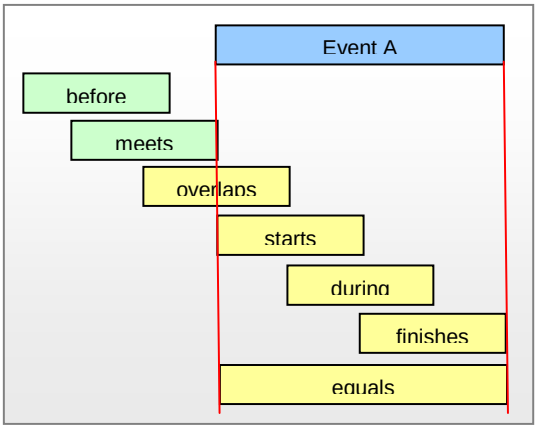


Figura 3.4: Relações de Allen (ALLEN, 1983)

c) UFO-C

UFO-C fundamenta-se em UFO-A e UFO-B para sistematizar conceitos sociais, tais como plano, ação, objetivo, agente, intencionalidade, comprometimento e compromisso (GUIZZARDI et al. 2008b). Ela é uma ontologia de entidades sociais (tanto indivíduos duradouros quanto eventos) e está apresentada na Figura 3.5.

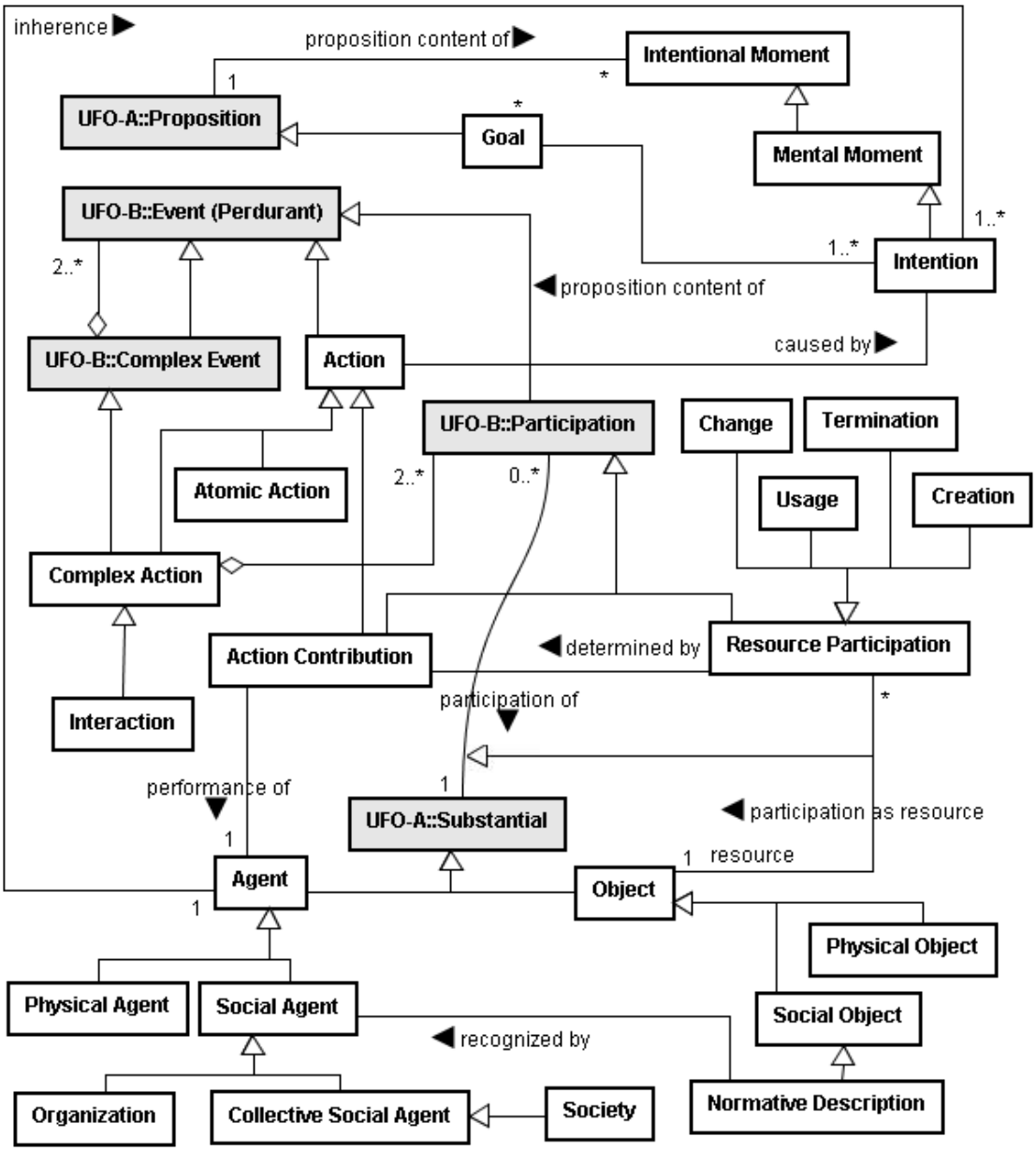


Figura 3.5: UFO-C

Começa-se pela distinção entre substanciais agentivos e não agentivos, respectivamente denominados ***agentes*** (*Agent*) e ***objetos*** (*Object*). Agentes podem ser ***físicos*** (*Physical Agent*),

por exemplo, uma pessoa, ou ***sociais*** (*Social Agent*), tal como uma organização (*Organization*) ou uma sociedade (*Society*).

Assim como agentes, objetos podem ser ***físicos*** (*Physical Object*), como um livro, um carro, uma árvore, ou ***sociais*** (*Social Object*), p.ex., dinheiro, linguagem e normas. Uma ***descrição normativa*** (*Normative Description*) é um tipo de objeto social que define uma ou mais regras/normas ***reconhecidas por*** (*recognized by*), pelo menos, um ***agente social***. São exemplos de ***descrições normativas*** a Constituição Brasileira e o regimento do Mestrado em Informática da UFES (GUIZZARDI et al. 2008b).

Agentes (*Agent*) são substanciais que podem possuir tipos especiais de ***modos*** (*Moment*) chamados de ***modos intencionais*** (*Intentional Moments*). Todo ***modo intencional*** tem uma e somente uma proposição (*Proposition*) como seu conteúdo proposicional (*proposition content of*).

Modos intencionais (*Intentional Moment*) podem ser ***modos sociais*** (*Social Moment*) ou ***modos mentais*** (*Mental Moment*). ***Intenções*** (*Intention*) são modos mentais representando estados de coisas desejados e que o agente se compromete a perseguir e por isso executam ações. O conteúdo proposicional (*propositional content of*) de uma ***intenção*** é um ***objetivo*** (*Goal*). Uma ***situação*** na realidade pode ***satisfazer*** (*satisfies*) a ***proposição*** que representa o ***conteúdo proposicional de*** (*propositional content of*) um ***modo intencional***.

Ações (*Action*) são eventos intencionais, ou seja, têm o propósito específico de satisfazer (o conteúdo proposicional de) alguma ***intenção*** (*Intention*). Assim como eventos, ações podem ser ***atômicas*** (*Atomic Action*) ou ***complexas*** (*Complex Action*), sendo uma ação complexa composta de duas ou mais ***participações*** (*Participation*). Participações intencionais de agentes são denominadas ***contribuições de ação*** (*Action Contributions*). Uma ação complexa composta de contribuições de ações de diferentes agentes é denominada uma ***interação*** (*Interaction*), tal como um diálogo entre dois agentes.

Objetos (*Object*) são substanciais inanimados que podem participar em ações de diferentes maneiras. São considerados quatro tipos de ***participação de recurso*** (*Resource Participation*): ***criação*** (*Creation*), ***término*** (*Termination*), ***alteração*** (*Change*) e ***uso*** (*Usage*). Considere ***o*** um objeto, ***a*** uma ação e ***s1*** e ***s2*** duas situações correspondendo, respectivamente, ao pré e ao pós-estado da ação ***a***. Tem-se, então:

- Criação: uma participação de recurso de ***o*** em ***a*** é uma criação se: (***o*** não está presente em ***s1***) E (***o*** está presente em ***s2***) E (existe pelo menos uma contribuição de ação ***ac*** que é parte de ***a*** e que ***s2*** satisfaz o conteúdo proposicional de ***ac***).

- Término: uma participação de recurso de ***o*** em **a** é um término se: (***o*** está presente em ***s1***) E (***o*** não está presente em ***s2***) E (existe pelo menos uma contribuição de ação ***ac*** que é parte de ***a*** e que ***s2*** satisfaz o conteúdo proposicional de ***ac***).
- Alteração: uma participação de recurso de ***o*** em **a** é uma alteração se: (i) existe pelo menos um modo ***m*** tal que (***m*** é inerente a ***o*** em ***s1*** E ***m*** não é inerente a ***o*** em ***s2***) OU (***m*** não é inerente a ***o*** em ***s1*** E ***m*** é inerente a ***o*** em ***s2***); E (ii) existe pelo menos uma contribuição de ação ***ac*** que é parte de ***a*** e que ***s2*** satisfaz o conteúdo proposicional de ***ac***.
- Uso: uma participação de recurso que não é de nenhum dos tipos anteriores é uma participação de uso.

3.3.2. Ontologias de Domínio

Uma ontologia de domínio define um vocabulário comum para compartilhar informações de um domínio específico. Ela é uma descrição de conceitos em um domínio de discurso, suas propriedades e restrições (NOY, N. F., MCGUINNESS, 2001). Os objetivos principais de se desenvolver uma ontologia de domínio são: compartilhar informação, reusar elementos do domínio, tornar suposições do domínio explícitas, separar conhecimentos de domínio de conhecimento operacional e analisar o conhecimento do domínio.

Ontologias de domínio têm sido investigadas intensivamente pelas comunidades da Inteligência Artificial e Engenharia do Domínio, motivados pela necessidade de reforçar o reúso de software em um nível mais elevado de abstração. Uma quantidade grande de ontologias de domínio tem sido desenvolvida para domínios como medicina, direito, engenharia, modelagem organizacional e química (GUIZZARDI, 2005). Também diversos métodos para a engenharia de ontologias, focalizando principalmente em ontologias do domínio, foram propostos (GÓMEZ-PÉREZ et al., 2004). Para representar ontologias do domínio, no geral, algum modelo estrutural é usado, como os diagramas de classes da UML (GUIZZARDI, 2005) (CRANEFIELD; PURVIS, 1999) (FALBO, 2004).

Para o desenvolvimento de ontologias de domínio, têm-se diversos métodos e representações. Por exemplo, SABiO (*Systematic Approach for Building Ontologies*) (FALBO, 2004) estabelece atividades para a construção de ontologias e orientações de como proceder na sua realização. Esse método propõe um processo cujas atividades são: (i) identificação do propósito e usos esperados, na qual se desenvolvem questões de competência que a ontologia deve ser capaz de responder; (ii) captura da conceituação com base nas questões de competências, identificando e organizando conceitos e relações relevantes; (iii)

formalização da ontologia, estabelecendo formalismos para representar as categorias de conhecimento e restrições da ontologia explicitamente em uma linguagem formal; (iv) integração com ontologias existentes, visando aproveitar conceituações existentes; (v) avaliação da ontologia para verificar se esta satisfaz aos requisitos estabelecidos na especificação; e (vi) documentação da ontologia. Para apoiar a captura, SABiO advoga o uso de modelos estruturais e propõe um perfil UML simples para representar ontologias (MIAN; FALBO, 2003).

Em sua tese, Guizzardi (2005) propõe um perfil UML ontologicamente correto, chamada posteriormente de OntoUML. Esse perfil é uma extensão da UML 2.0, ontologicamente bem fundamentada e possui um metamodelo isomórfico à UFO-A, parcialmente mostrado na Figura 3.6. OntoUML permite produzir ontologias de qualidade e usá-las como um modelo conceitual de alta expressividade. Diversas ontologias já foram modeladas utilizando esse perfil e benefícios foram obtidos do uso dessa abordagem, conforme relatado em (GONÇALVES et al., 2007), (GUIZZARDI et al., 2008a) e (GUIZZARDI et al. 2008b), (FALBO; NARDI, 2008) e (OLIVEIRA et al., 2007). Nesses trabalhos, a UFO foi utilizada para avaliar, reprojetar e dar semântica de mundo real para ontologias de domínio, corrigindo problemas conceituais nessas ontologias, tornando-as mais fiéis ao domínio representado e tornando seus comprometimentos ontológicos explícitos.

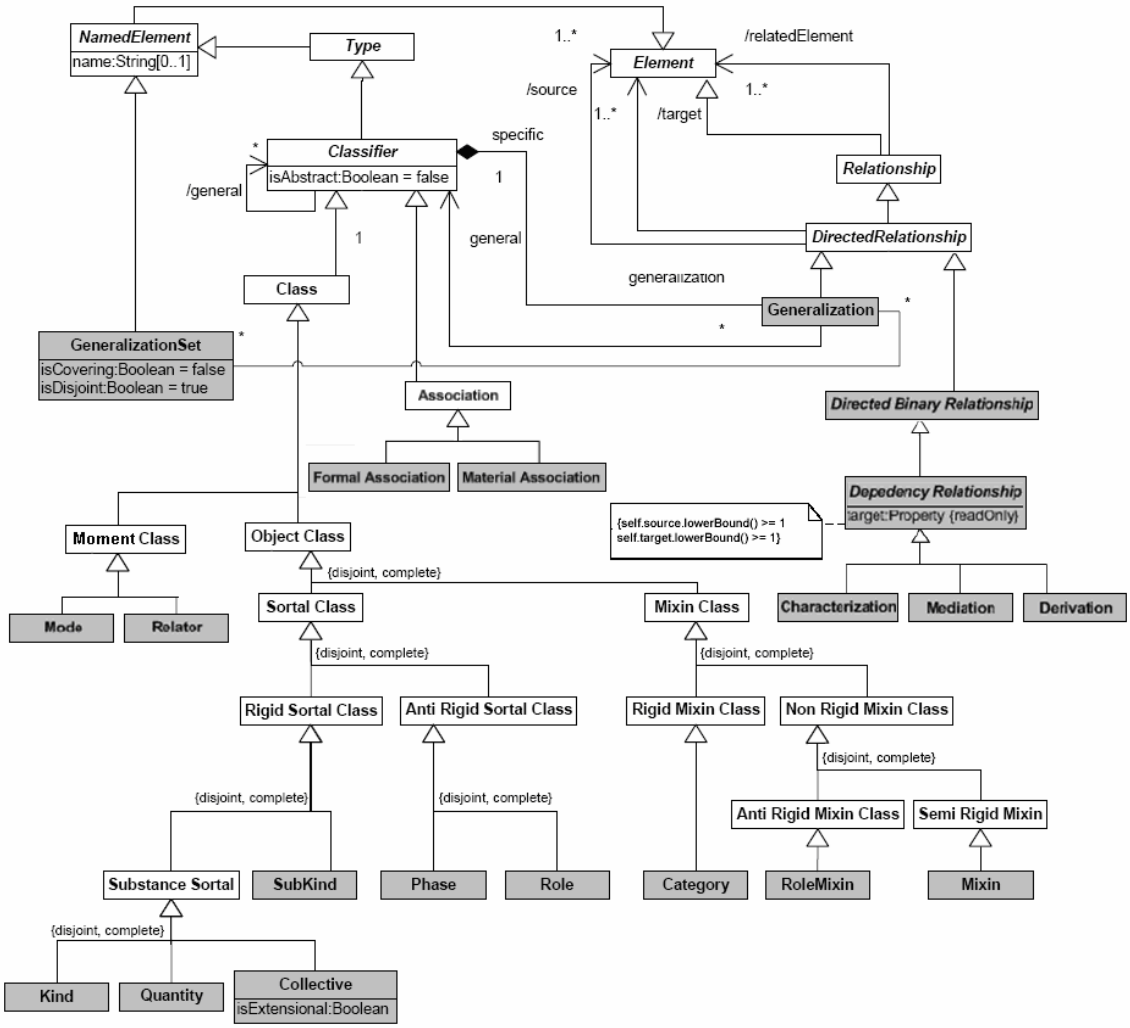


Figura 3.6: Metamodelo do perfil OntoUML (GUIZZARDI, 2005)

Os elementos de OntoUML seguem as definições da UFO-A e seus significados são resumidos na Tabela 3.1 (GUIZZARDI, 2005).

Tabela 3.1 – Conceitos da linguagem de modelagem OntoUML

Concept	Description
<<kind>>	Representa um sortal de substância cujas instâncias são complexos funcionais. Exemplos incluem espécies da natureza (tais como pessoa, árvore etc.) e artefatos (cadeira, carro etc.), bem como elementos sociais (p.ex.: organização, linguagem etc).
<<quantity>>	Representa um sortal de substância cujas instâncias são quantidades (p. ex.: água, areia, sal).
<<collective>>	Representa um sortal de substância cujas instâncias são coletivos (p.ex., floresta).
<<subkind>>	Uma especialização de espécie (*kind*) que herda dessa entidade o princípio de identidade
<<phase>>	Representa um sortal instanciado em determinado mundo ou período de tempo, mas não necessariamente em todos (p.ex.: criança, adolescente, adulto).
<<role>>	Representa um sortal instanciado eventualmente, mais precisamente na participação em um evento ou numa determinada relação. Define um papel, ou seja, algo que pode ser assumido em um "mundo", mas necessariamente não em todos os mundos (um universal antirrígido e relacionalmente dependente). Por exemplo, o papel Estudante desempenhado por instâncias da espécie Pessoa.
<<mixin>>	Representa propriedades que são essenciais para algumas de suas instâncias e acidentais para outras (semi-rigidez).
<<category>>	Representa um tipo de universal misto (*mixin universal*) que classifica as entidades que pertencem a espécies (*kinds*) diferentes, mas que compartilham uma propriedade comum essencial (ou seja, uma propriedade que suas instâncias não podem perder).
<<rolemixin>>	Representa um não sortal, antirrígido e relacionalmente dependente, i.e., um universal dispersivo que agrega propriedades que são comuns a diferentes papéis (*roles*).
<<mode>>	Representa um universal de momento intrínseco e, portanto, toda instância é existencialmente dependente de exatamente uma entidade. Exemplos típicos de modos são cor, carga elétrica, sintoma etc.
<<relator>>	Representa um universal de momento relacional sendo toda instância existencialmente dependente de pelo menos duas entidades. Exemplos incluem casamento, locação etc.
Relations	**Description**
<<characterization>>	É uma relação formal entre um universal de modo e o universal duradouro (*endurant*) que ele caracteriza.
<<mediation>>	Representa um tipo de relação formal que acontece entre um universal de modo relacional (*relator*) e os universais duradouros (*endurants*) que ele media.
<<derivation>>	Representa a relação formal de derivação que acontece entre uma relação material e o universal de modo relacional (*relator*) do qual essa relação material é derivada.
<<material>>	Representa uma relação material, i.e., um universal relacional que é induzido por um universal de modo relacional (*relator*).
<<formal>>	Representa uma relação formal (que ocorre diretamente entre indivíduos dos universais relacionados).

3.3.3. Ontologias de Tarefa

Uma ontologia de tarefa provê um vocabulário de termos usados para resolver problemas associados com uma tarefa, que pode ou não ser realizada em um mesmo domínio. Em outras palavras, compreende um conjunto de primitivas de representação da estrutura da tarefa, de forma independente de domínio (MIZOGUCHI et al., 1995a). O conhecimento de tarefa é associado à descrição da decomposição de uma tarefa em subtarefas, do controle de fluxo ao longo dessas subtarefas e dos papéis de conhecimento para o conhecimento do domínio que é usado ou produzido pelas subtarefas (ZONG-YONG et al., 2007) (BREUKER; VAN DE VELDE, 1994).

No contexto de tarefas, um termo muito utilizado é o de método de solução de problema (*Problem Solving Method* - PSM). Segundo Fensel e outros (2003), um PSM define o processo de raciocínio em um sistema baseado em conhecimento independentemente de termos do domínio. Um PSM representa uma abstração da sequência de passos comum a uma determinada classe de problemas. Zlot (2002) faz uma diferenciação entre ontologias de tarefa e PSMs. Segundo ele, uma ontologia de tarefa se limita a permitir simular o processo de solução do problema conceitualmente, não mostrando como o problema pode ser resolvido. Já um PSM descreve a solução do problema especificando as inferências necessárias para que o objetivo da tarefa seja alcançado. Já Ikeda e outros (1998) consideram uma ontologia de tarefa como a ontologia capaz de capturar o conhecimento de solução de um problema independentemente de domínio, ou seja, o conhecimento capturado por PSMs (CHANDRASEKARAN et al., 1998). Eles reforçam que uma ontologia de tarefa especifica não somente o esqueleto de um processo de solução do problema, mas também o contexto onde os conceitos do domínio são usados (IKEDA et al., 1998).

Considerando essa última definição e a classificação de ontologias proposta por Guarino (1998), nota-se que as ontologias de tarefa capturam o conhecimento genérico a respeito de uma tarefa, devendo, portanto, englobar todo tipo de conhecimento relativo à execução de uma tarefa (decomposição das tarefas, ordem de execução, objetivos e papéis de conhecimento). Logo, neste trabalho considera-se o termo ontologia de tarefa como um meio de capturar conhecimento diversificado relativo a uma tarefa.

Em contraste às ontologias do domínio, não há métodos amplamente aceitos para representar ontologias de tarefa. A notação usada para capturar o conhecimento de tarefa varia de um trabalho para outro. Por exemplo, Mizoguchi e outros (1995a) usam quatro tipos de conceitos para descrever uma ontologia da tarefa: (i*) substantivos genéricos,* que representam os objetos que refletem os papéis que aparecem no processo de solução do problema, (ii)

verbos genéricos, que representam atividades necessárias para atingir a solução, (iii) *adjetivos genéricos*, que indicam a modificação dos objetos, e (iv) outros conceitos específicos da tarefa.

Em (IKEDA et al., 1998), os autores incluem restrições e advérbios genéricos como parte do vocabulário genérico usado para descrever tarefas e usam uma rede de processos genérica como um modelo gráfico representar o processo de resolução de problema em termos de primitivas da ontologia.

As Figura 3.7 e Figura 3.8 apresentam as ontologias da tarefa de agendamento (*scheduling*) propostas nesses trabalhos. Em ambas, usam-se substantivos genéricos para representar os papéis dos objetos presentes na tarefa de agendamento, tais como agenda (*Schedule*), dados (*data/information*), representação da agenda (*Schedule representation*), recurso de agendamento (*RSC- schedule resource)* e beneficiário do agendamento (*RCP - schedule recipient*), e verbos genéricos para descrever tarefas, tais como marcar (*assign*), classificar (*classify*) e combinar (*combine*).

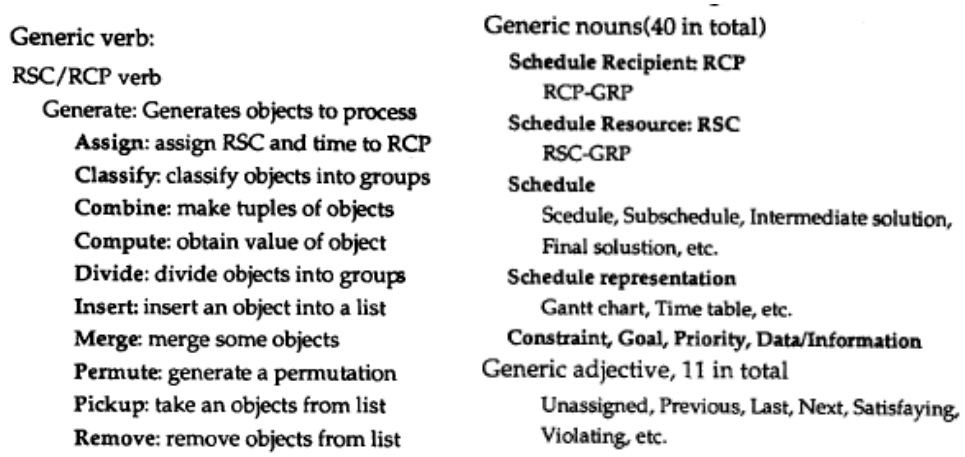


Figura 3.7: Representação da Tarefa de Agendamento segundo (MIZOGUCHI et al., 1995a).

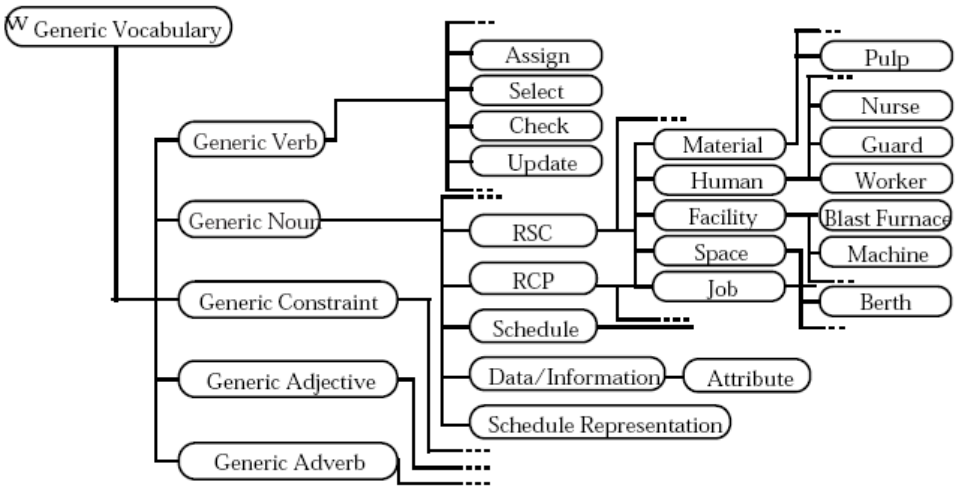


Figura 3.8: Representação da Tarefa de Agendamento segundo (IKEDA et al., 1998).

Na linha desses trabalhos, Wang e Chang (2001) usam diagramas da UML para representar ontologias. A ontologia representada em (WANG; CHANG, 2001) é dependente de domínio, mas trata também de aspectos de tarefa. No estudo de caso apresentado, eles tratam do cálculo da média da área contaminada por petróleo. Eles utilizam diagramas de atividades para representar a visão de atividade e a visão operacional da tarefa (Figura 3.9) e diagramas de classes para representar a visão estrutural (Figura 3.10). A visão de atividade permite representar a ordem de execução das atividades. Na parte à esquerda da Figura 3.9, primeiro mede-se a área contaminada e depois se calcula o volume de água contaminada. Já a visão operacional é restrita a representar os raciocínios envolvidos nessas tarefas. Na parte à direita da Figura 3.9, usam-se expressões OCL para representar as condições que definem se a área contaminada é pequena (*small*), média (*medium*) ou grande (*large*). Na visão estrutural são apresentados os conceitos, suas relações e propriedades. Na Figura 3.10 o conceito Média (*Media*) é apresentado e contém as propriedades necessárias para as operações descritas, como o tamanho (*site size*), a área (*area*) e o volume (*volume*) da área contaminada.

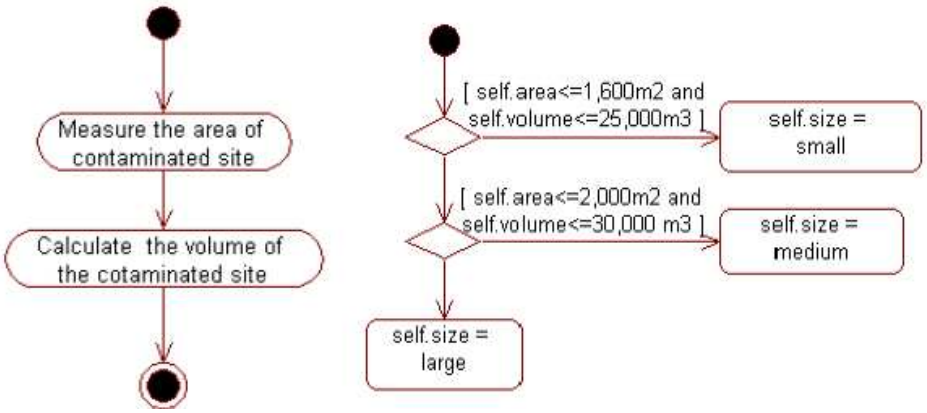


Figura 3.9: Visão de atividade e visão operacional (WANG; CHANG, 2001).

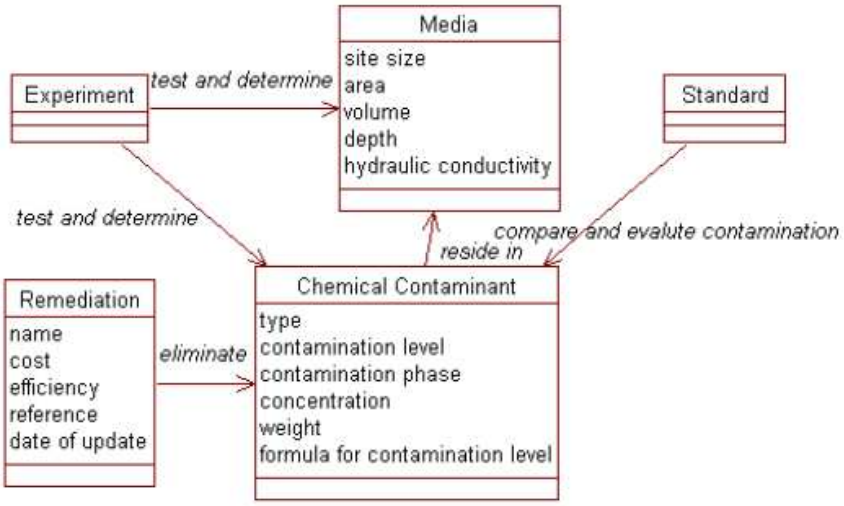


Figura 3.10: Visão estrutural (WANG; CHANG, 2001).

Rajpathak e outros propõem ontologias para as tarefas de agendamento (*scheduling*) (RAJPATHAK et al., 2001) e de planejamento (*planning*) (RAJPATHAK; MOTTA, 2004). Eles não descrevem fluxos das atividades, mas sim os conceitos, as relações e as propriedades envolvidos nessas tarefas, como mostrado nas Figura 3.11 e Figura 3.12 que tratam da tarefa de agendamento. A primeira indica as classes da ontologia de tarefa de agendamento, simbolizadas por retângulos. Por meio de setas diferenciadas são indicadas as subclasses, partes e metaclasses. Por exemplo, Tarefa (*Job*), Recurso (*Resource*) e Restrição (Constraint) são classes; um Tipo de Recurso (*Resource type*) é uma meta classe de Recurso (*Resource*); Condições Severas (*Hard constraint*) e Flexíveis (*Soft constraint*) são subtipos de Restrição (*Constraint*); e uma Atividade (*Activity*) é parte de uma Tarefa (*Job*). Os retângulos de cantos arredondados na Figura 3.12 indicam as relações entre as classes apresentadas e as setas representam argumentações entre elas. Por exemplo, um Horário (*Schedule*) é maximamente aceitável (*maximally admissible*) se satisfaz (*schedule satisfies*) as Restrições (*Hard* e *Soft Constraints*).

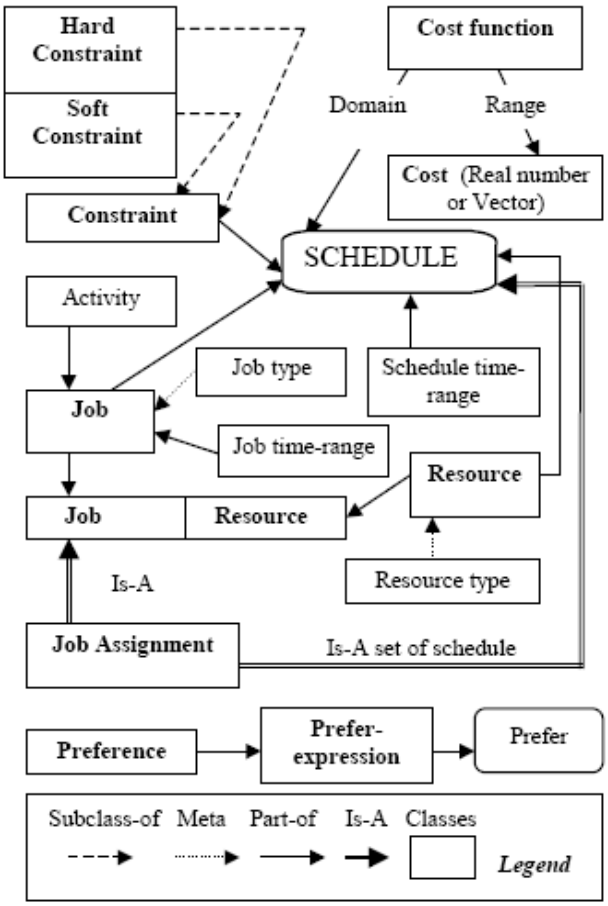


Figura 3.11: Conceitos da tarefa de agendamento e seus relacionamentos segundo a representação de (RAJPATHAK et al., 2001).

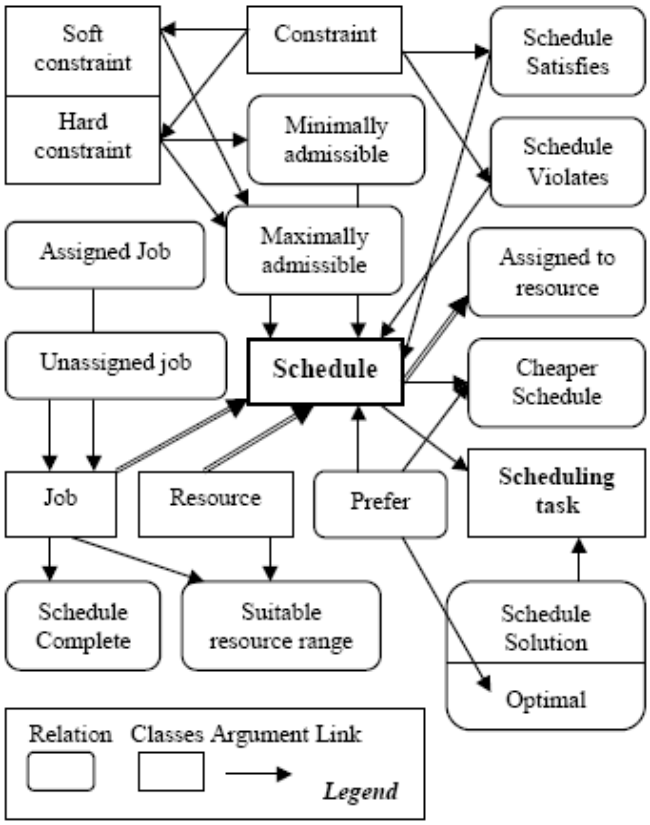


Figura 3.12: Atividades, conceitos envolvidos, entradas e saídas (RAJPATHAK et al., 2001).

Conciliando as abordagens discutidas acima, Zlot e outros (2002) representam os conceitos da tarefa (papéis do conhecimento a serem substituídos pelos conceitos do domínio) usando um modelo conceitual estrutural e o fluxo de controle nas tarefas por meio de pseudoalgoritmos escritos em linguagem natural estruturada e em uma notação gráfica simples para representar a decomposição da tarefa. Na Figura 3.13 é apresentada a representação em pseudoalgoritmo da tarefa de configuração e na Figura 3.14 é apresentada a tarefa de configuração em notação gráfica, enfatizando a sua composição. É possível observar em ambas as figuras que a tarefa de configuração (*Configuration*) é composta de quatro subtarefas: selecionar (*select*), propor (*propose*), verificar (*verify*) e revisar (*revise*). No pseudoalgoritmo (Figura 3.13) é possível visualizar o controle de fluxo dessas tarefas e na Figura 3.14 associa-se a essas tarefas quais os métodos necessários para resolvê-las.

```
While exists a parameter without computed
      value or a constraint that it had been
      violated do
Select parameter
       Propose a value for  the parameter
       For each constraint do
              Verify constraint
       End-For-Each
       For each violated constraint do
                     Revise constraint
        End-For-Each
End-While
```

Figura 3.13: Representação em pseudoalgoritmo para a tarefa de configuração (ZLOT et al., 2002).

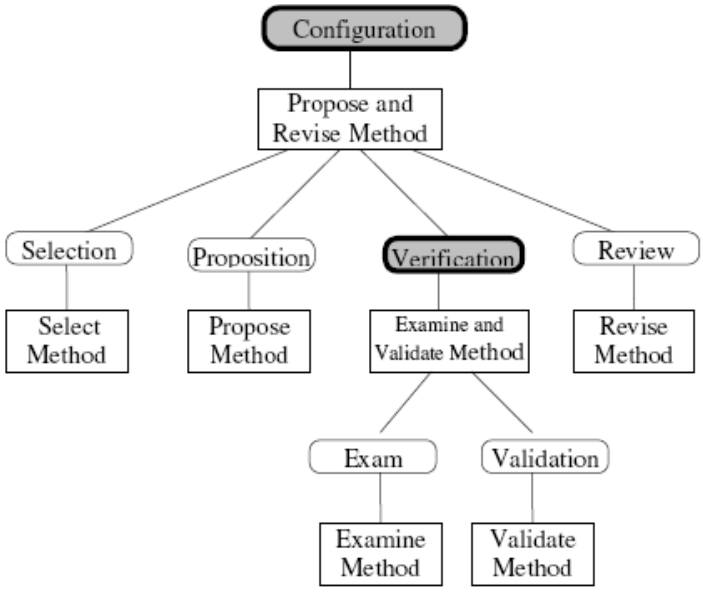


Figura 3.14: Representação gráfica da tarefa de configuração (ZLOT et al., 2002).

Em (TRAN; TSUJI, 2007) é apresentada a linguagem OWL-T, que pode ser usada para descrever e especificar formal e semanticamente uma tarefa. Essa pode ser, então, transformada em processo de negócio executável por sistemas. Na Figura 3.15 é apresentada uma implementação em OWL-T para a tarefa de reserva de passagens aéreas (*bookflight*). É definido que essa tarefa é uma tarefa simples (*simpleTask*), que possui uma descrição (*description*), pré (*hasPrecondition*) e pós-condições (*hasPostcondition*), entradas (*hasInput*) e saídas (*hasOutput*), efeitos (*hasEffect*), que são eventos que ocorrem após a execução da tarefa, e preferências (*hasPreference*), que são propriedades desejadas de uma solução. Deve-se realçar que o exemplo apresentado também é dependente de domínio, ainda que genérico para o domínio e não específico de aplicação.

```
<simpleTask rdf:ID = "bookFlight">
  <relatesToTOnto rdf:resource = "#TranTOnto"/>
  <refersToDOnto rdf:resource = "#FlightDOnto"/>
  <name>BookFlight</name>
   <description>This task is for booking an
ordinary flight ticket…</description>
  …
  <hasInput rdf:resource = "#CustomerInfo"/>
  <hasPrecondition>
    <Expression>(Price <= 500)</Expression>
  </hasPrecondition>
  <hasPreference>
    <Expression>(Flight in morning)</Expression>
  </hasPreference>
  <hasOutput rdf:resource = "#TConfirmation"/>
  <hasPostcondition>
     <Expression>(all account information of user
must be deleted)</Expression>
  </hasPostcondition>
  <hasEffect>
     <Expression>(ticket is delivered at user's
address)</Expression>
  </hasEffect>
</simpleTask>
```

Figura 3.15: Tarefa de reserva de passagens aéreas (TRAN; TSUJI, 2007).

Ontologias de tarefa têm sido desenvolvidas com o objetivo, dentre outros, do reúso de conhecimento de tarefa. Entretanto, o seu reúso não tem alcançado a mesma intensidade que ontologias de domínio. Além das ontologias de tarefa, há outros trabalhos que enfocam o reúso de conhecimento de tarefa, a maioria deles propostos pela comunidade de Inteligência Artificial, tal como CommonKADS (BREUKER; VAN DE VELDE, 1994), apresentado no Capítulo 2.

Existe, ainda, a abordagem UPML (*Unified Problem-Solving Method Development Language*) (FENSEL et al., 2003), que foi desenvolvida juntamente com o projeto IBROW (MOTTA; LU, 2000) e tem como objetivo permitir o reúso semiautomático de métodos de solução de problemas. Ela é uma abordagem para modelar genericamente um PSM,

independentemente de domínio. Ela descreve os diferentes componentes de software de um Sistema Baseado em Conhecimento (SBC), integrando duas importantes linhas de pesquisa em Engenharia do Conhecimento: reúso de componentes e ontologias. O *framework* UPML apoia a modelagem de SBCs a partir de componentes reusáveis, adaptadores, diretrizes de desenvolvimento, linguagem de descrição e ferramentas (FENSEL et al., 2003). Para essa linguagem foi construído um *plugin* para o Protégé[1], chamado PSM Librarian (CRUBÉZY; MUSEN, 2003), que permite procurar bibliotecas de PSMs de forma uniforme e adaptá-los ao conhecimento de domínio. Ele inclui um interpretador que processa as relações mapeadas para reformular o conhecimento de domínio. A Figura 3.16 mostra uma tela da utilização desse *plugin* no Protegé, onde se tem a implementação do método Propor e Revisar (*Propose & Revise*) citado em (BREUKER; VAN DE VELDE, 1994). A notação gráfica apresenta: (i) as subtarefas (*Subtasks*), a saber: selecionar (*select*), propor (*propose*), revisar (*revise*) e verificar (*verify*); (ii) as entradas (*Input roles*) e saídas (*Output roles*), dentre elas restrições (*constraints*), consertos (*fixes*) e parâmetros (*parameters*). Essas informações aparecem também no painel central, assim como pré-condições (*Preconditions*) e pós-condições (*Postconditions*).

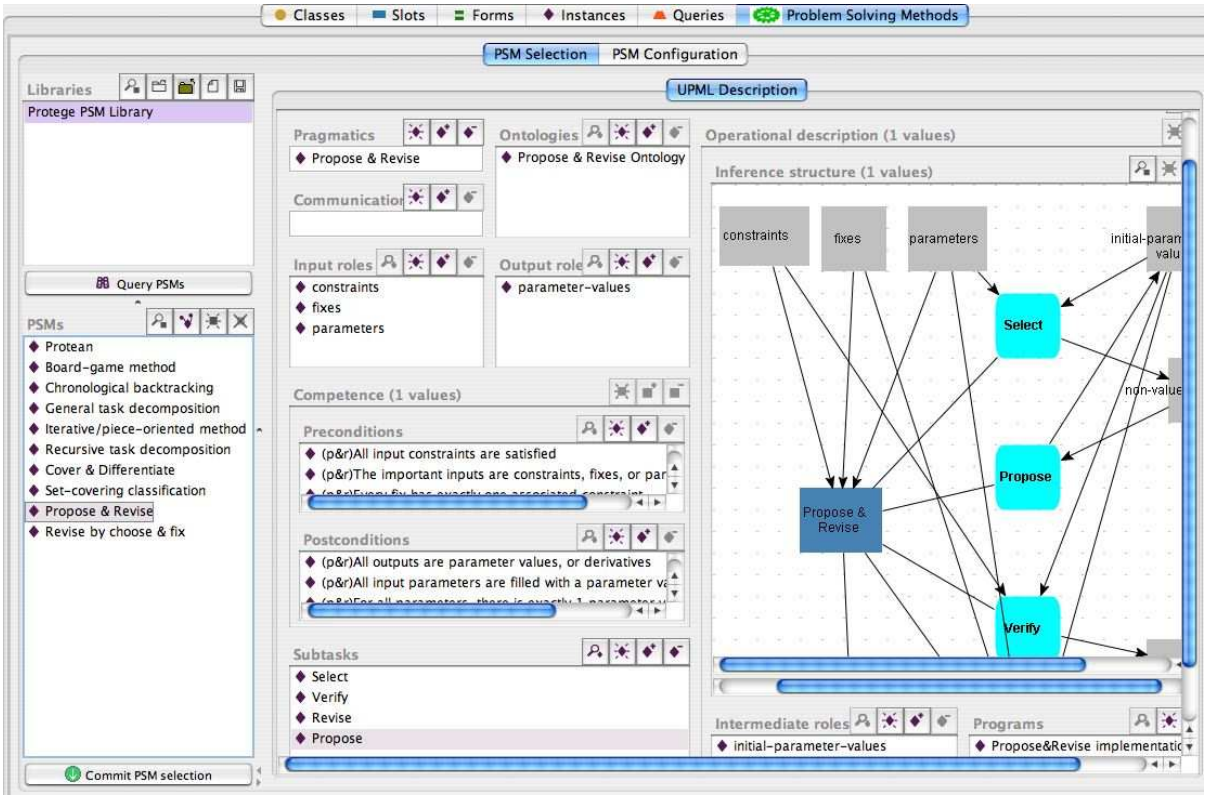


Figura 3.16: Imagem do uso do plugin PSM tab no Protégé.

[1] http://protege.stanford.edu/

Cada uma das abordagens citadas tem um procedimento próprio de modelagem, com notação e divisão do conhecimento específicos. Essa falta de padronização e a complexidade das notações utilizadas podem ser apontadas como motivos que expliquem a baixa utilização de ontologias de tarefa, quando comparadas a ontologias de domínio.

3.3.4. Ontologias de Aplicação e de Classes de Aplicação

A noção de ontologias de aplicação foi introduzida por Gennari e outros (1994), como uma ontologia que cobre o conhecimento requerido por um método de solução de problema que usa a terminologia de uma ontologia de domínio. Para eles, ontologias de aplicação são criadas com o propósito de reduzir a distância entre ontologias de domínio e de tarefa, e para permitir ao especialista de domínio ter a mesma linguagem adotada na aplicação em mãos.

Segundo van Heijst e outros (1997), ontologias de aplicação contêm as definições que são necessárias para modelar o conhecimento requerido por uma aplicação particular. Elas são uma mistura de conceitos derivados de ontologias de domínio, genéricas e de tarefa.. Elas descrevem, no contexto de uma aplicação, conceitos dependentes de um domínio e de tarefas particulares, os quais são, frequentemente, especializações das ontologias relacionadas (GUARINO, 1998). Resumidamente, uma ontologia de aplicação provê conceitos e relacionamentos entre esses conceitos para uma dada aplicação (ZONG-YONG et al., 2007).

3.4. CONCEITUAÇÃO E LINGUAGENS DE REPRESENTAÇÃO DE ONTOLOGIAS

Guizzardi (2007) afirma que um dos principais fatores de sucesso no uso de uma linguagem de modelagem reside na habilidade de a própria linguagem prover aos usuários primitivas de modelagem que sejam capazes de expressar diretamente conceitos relevantes de um domínio, ou seja, de expressar a conceituação desse domínio. Os elementos que constituem essa conceituação são usados para falar sobre abstrações de certas situações do mundo real que podem ser chamadas de abstrações de domínio. Essas conceituações e as abstrações são entidades não materiais que existem no universo mental de um usuário ou de um conjunto de usuários de uma linguagem.

Uma linguagem é o meio pelo qual se representa em símbolos (conjunto de elementos que compõem um vocabulário) as conceituações de uma realidade (ou parte dela). O objetivo de uma linguagem é capturar, documentar, comunicar e analisar as abstrações dos usuários de

uma maneira concisa, completa e não ambígua, ou seja, representar os modelos mentais (as abstrações) em artefatos concretos e representativos de termos (GUIZZARDI, 2007).

O Triângulo de Ullmann (ULLMANN, 1972 apud GUIZZARDI, 2005), representado na Figura 3.17, mostra as relações entre linguagem, conceituação e a porção da realidade que essa conceituação abstrai.

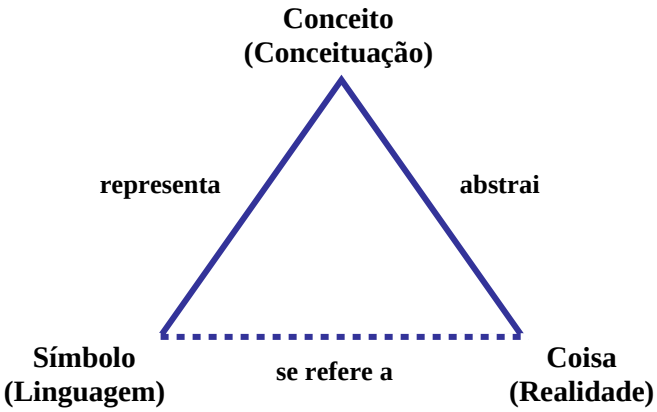


Figura 3.17: Triângulo de Ullmann.

A relação “representa” diz respeito à definição da semântica da linguagem em termos de entidades do mundo real. A relação pontilhada entre a linguagem e a realidade é sempre intermediada por certa conceituação. Essa relação é apresentada na Figura 3.18, onde são destacas as relações entre conceituação, abstração, linguagem de representação e especificação de modelo (GUIZZARDI, 2007).

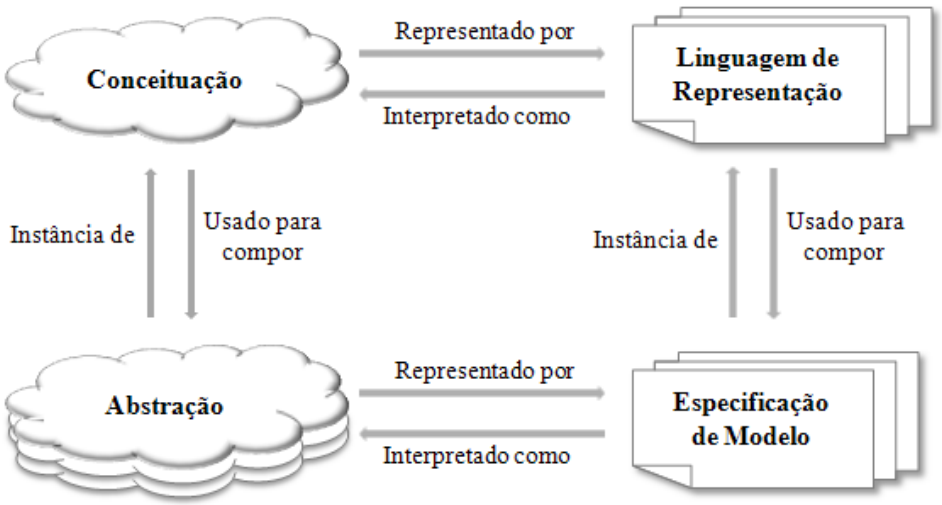


Figura 3.18: Relações entre Conceituação, Abstração, Linguagem de Representação e Especificação de Modelo (GUIZZARDI, 2007).

A representação de uma abstração de domínio em termos de uma linguagem é chamada de Especificação de Modelo (ou simplesmente Modelo) e a linguagem usada para a sua criação é chamada de Linguagem de Representação ou Linguagem de Modelagem. Essa linguagem representa uma conceituação de domínio usada para compor a abstração representada por uma especificação de modelo. Por exemplo, uma conceituação para o domínio de genealogia pode ser construída considerando conceitos como homem, mulher, pai e mãe, dentre outros. Usando esses conceitos, podem-se articular várias abstrações de domínio, por exemplo, que uma mulher chamada Maria é mãe de um homem chamado João (GUIZZARDI, 2005).

Pode-se dizer que, para um modelo representar fielmente uma abstração, as primitivas de modelagem da linguagem usadas para produzir esse modelo devem representar fielmente a conceituação usada para articular essa abstração. Quanto mais forte a correspondência entre uma abstração da realidade e seu modelo de representação, mais fácil é comunicar e raciocinar com esse modelo. A força dessa correspondência está diretamente ligada à qualidade da linguagem de representação usada e pode ser medida através de critérios como **adequação ao domínio** e **adequação à compreensão**. O primeiro mede a capacidade de uma linguagem de se adequar aos fenômenos de um domínio. O outro mede a adequação pragmática da linguagem, ou seja, refere-se ao quão fácil é para um usuário reconhecer o significado dos construtores da linguagem e entender, comunicar e raciocinar com as especificações produzidas (GUIZZARDI, 2007).

Uma conceituação de domínio delimita todas as possíveis abstrações que são admissíveis no domínio (GUARINO, 1998). Por exemplo, uma conceituação do domínio de genealogia não pode admitir uma abstração em que uma pessoa seja pai do seu próprio pai, pois tal estado do mundo não pode acontecer na realidade. Da mesma forma, a linguagem de representação delimita as possíveis especificações que podem ser construídas, isto é, toda especificação gramaticalmente válida da linguagem (GUIZZARDI, 2007). Retornando ao exemplo anterior, também uma linguagem de representação do domínio de genealogia não pode permitir que tal abstração seja representada. Sendo assim, pode-se dizer que uma linguagem ideal de representação é aquela fiel ao seu domínio, ou seja, que tem como especificações de modelo válidas (gramaticalmente corretas) apenas as que representam abstrações admissíveis pela conceituação daquele domínio (GUIZZARDI, 2007). Isso é possível quando o metamodelo da linguagem de representação é isomórfico à ontologia ideal de domínio, como mostra a Figura 3.19.

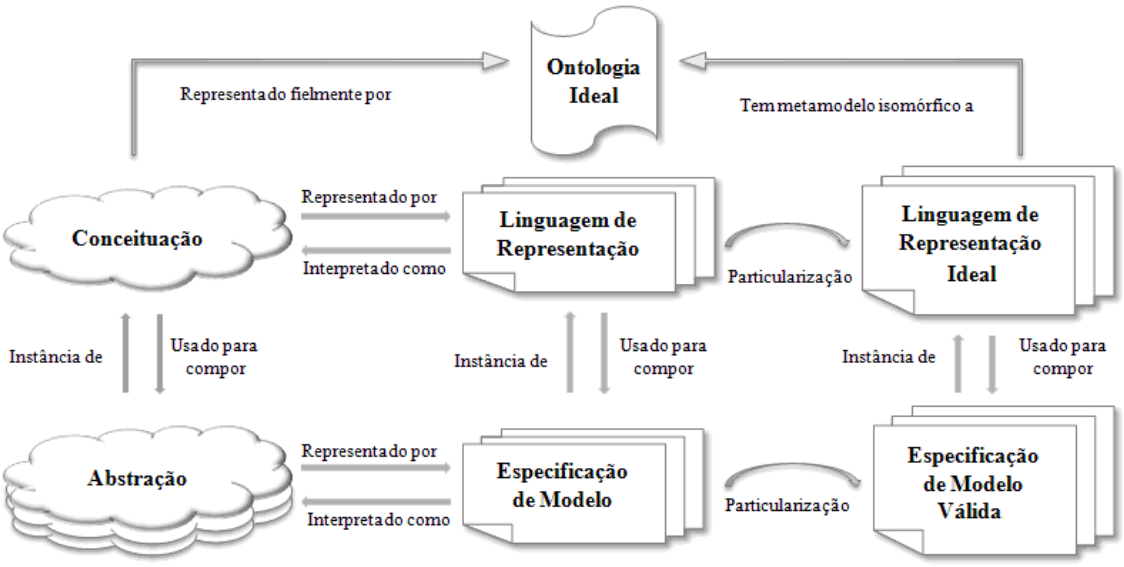


Figura 3.19: Aplicação de ontologias na criação e avaliação de Linguagens de Representação (ZAMBORLINI, 2008).

Uma Ontologia Ideal ou Ontologia de Referência é aquela que descreve todos e somente os estados de mundo admissíveis pela (meta)conceituação com a qual ela se compromete. Uma Ontologia de Referência de Domínio é, portanto, uma descrição explícita e formal da porção da realidade correspondente ao domínio em termos de um artefato concreto, por meio do qual a estrutura de conceituação do domínio deve também ser acessível. Seu objetivo é fazer a melhor descrição possível do domínio da realidade com respeito a um certo nível de granularidade e perspectiva (GUIZZARDI, 2007).

A Figura 3. 20 é análoga à Figura 3.18, porém num nível acima (metanível), em que se considera a existência da metaconceituação e de (Meta)Linguagens de Modelagem, usadas para compor modelos conceituais.

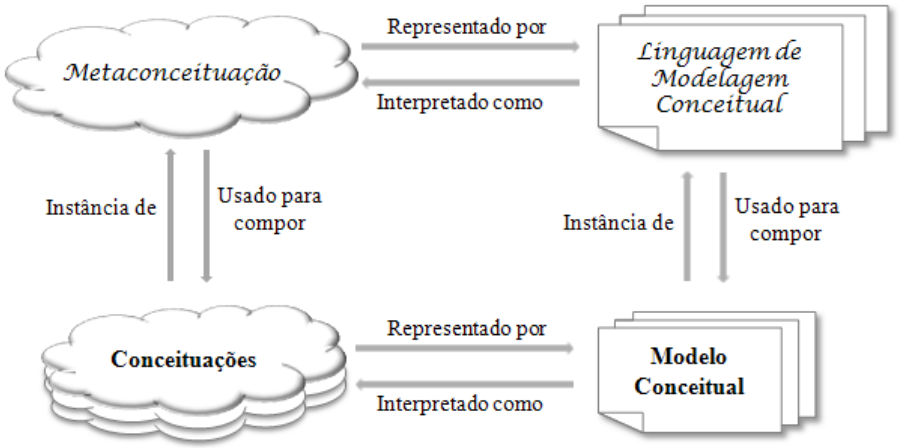


Figura 3. 20: Metaconceituação e (Meta)Linguagens usadas para compor Modelos Conceituais (GUIZZARDI, 2007) .

Uma metaconceituação define o conjunto de todas as conceituações de domínio que são fieis à realidade. Em outras palavras, ela é um conjunto de metaconceitos independentes de domínio que são usados para articular conceituações dos diversos domínios, por exemplo, conceitos como parte, todo, papel e evento. Esses conceitos devem ser devidamente capturados pelas Linguagens de Modelagem Conceitual para serem utilizados nos modelos conceituais. Estes são propostos para representar as conceituações de domínio e também são usados como metamodelos para construir Linguagens de Representação Específicas de Domínio (GUIZZARDI, 2007).

Um metamodelo de linguagem é a descrição da sintaxe abstrata da linguagem, que define um conjunto de construtores, selecionados com o propósito de realizar tarefas específicas, bem como um conjunto de regras bem formadas para combinar esses construtores a fim de criar modelos gramaticalmente válidos na linguagem (GUIZZARDI, 2007).

Uma Linguagem Ideal de Modelagem Conceitual é aquela que representa fielmente os conceitos da metaconceituação com a qual se compromete. Para isso, ela deve ser suficientemente expressiva para caracterizar formalmente as distinções ontológicas, sendo também uma Linguagem Ideal de Representação de Ontologias (GUIZZARDI, 2007). Utilizando-se essa linguagem, cujo metamodelo é isomórfico à Ontologia de Fundamentação Ideal, pode-se construir Ontologias de Domínio de Referência, que são um tipo específico de modelo conceitual, como mostra a Figura 3. 21.

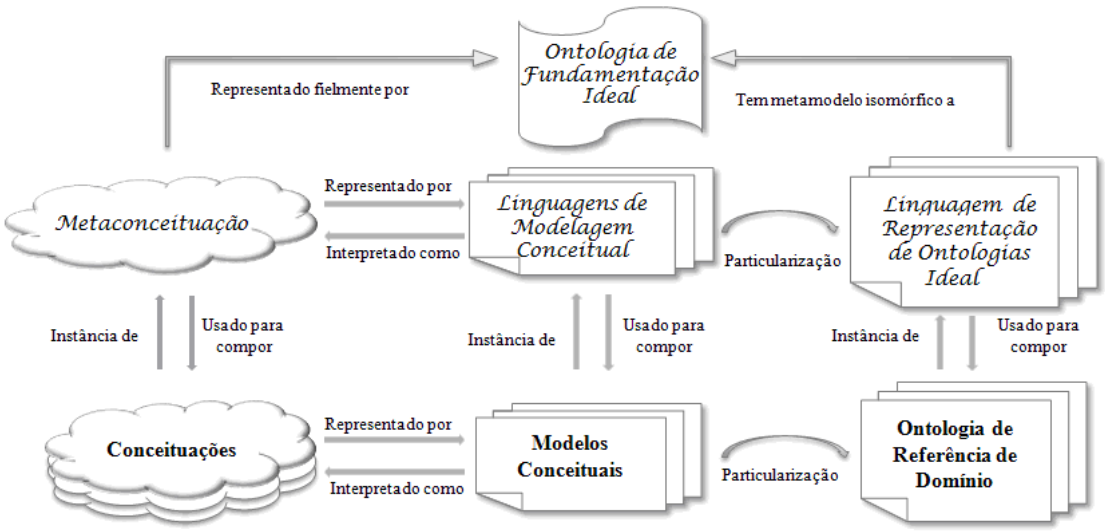


Figura 3. 21: Aplicação de Ontologias de Fundamentação na criação e avaliação de Modelos Conceituais/Ontologias de Referência (ZAMBORLINI, 2008).

Uma Ontologia de Fundamentação Ideal ou Ontologia de Fundamentação de Referência deve ser independente de domínio e deve prover os conceitos fundamentais sobre os quais ontologias de domínio devem ser construídas (GUIZZARDI; WAGNER, 2005).

Em resumo, uma ontologia de referência para um certo domínio pode ser vista como o seu modelo conceitual mais adequado, pois permite representar todos e apenas os estados de mundo que a sua conceituação admite. Além disso, essa ontologia precisa ser escrita numa linguagem ontologicamente bem fundamentada. Para isso, é necessário que o metamodelo dessa Linguagem de Representação de Ontologias seja isomórfico a uma Ontologia de Fundamentação que represente fielmente a metaconceituação usada para compor a conceituação do domínio (GUIZZARDI, 2007).

3.5. APLICAÇÕES DE ONTOLOGIAS

Ontologias esclarecem a estrutura do conhecimento (CHANDRASEKARAN et al., 1999), promovendo um entendimento compartilhado acerca de um domínio, tarefa ou aplicação. Elas podem ser utilizadas com os objetivos de: (i) compartilhar informação, (ii) reusar elementos, (iii) tornar suposições do domínio explícitas, (iv) separar conhecimentos de domínio de conhecimento operacional e (v) analisar o conhecimento do domínio (NOY; MCGUINNESS, 2001).

Uma vez que ontologias capturam um conhecimento específico, elas podem ser utilizadas no desenvolvimento de sistemas em diferentes momentos:

- Ontologias utilizadas em tempo de execução: permitem que agentes de software se comuniquem utilizando conceituações compartilhadas e formalizadas em ontologias, o que torna possível a interpretação das mensagens trocadas (GUARINO, 1998) (DRAGAN et al., 2006);
- Ontologias utilizadas em tempo de desenvolvimento: neste caso tem-se um conjunto de ontologias reusáveis à disposição, organizadas em uma biblioteca. Isso permite reduzir o custo da modelagem conceitual, possibilitando que o desenvolvedor pratique reúso de alto nível e compartilhe conhecimento usando um vocabulário comum em torno de uma plataforma de software heterogênea (GUARINO, 1998).
- Ontologias para a integração: podem ser utilizadas para integração de bases de dados ou de conhecimento, uma vez que proveem um esqueleto de conhecimento e uma infraestrutura independente de uma implementação

particular. Podem ser utilizadas na tradução de diferentes representações de bases de dados, fornecendo uma conceituação única a partir da qual outras conceituações se normalizam (USCHOLD, 1996).

- Apoio à Engenharia de Software: ontologias auxiliam na especificação, uma vez que promovem um entendimento compartilhado acerca de um domínio ou tarefa, e na reutilização sobretudo na engenharia de requisitos. Dentre trabalhos que abordam o reúso de ontologias no contexto da Engenharia de Software podem-se citar (GUIZZARDI, 2000), (NARDI, 2006), (ZLOT, 2002), (LU et al., 2003) e (ZONG-YONG et al., 2007).

Além desses trabalhos mais voltados para a engenharia de conhecimento e análise de domínio e tarefa, McGuinness (2003) dá ênfase à *Web* e considera a utilização de ontologias em áreas de ciência da informação, mecanismos de busca na *Web* baseados em ontologias, comércio eletrônico (*e-commerce*), dentre outros (MCGUINNESS, 2003) (DRAGAN et al., 2006).

Diante das diversas aplicações de ontologias, é importante enfatizar a importância e as vantagens desse uso (USCHOLD, 1996) (GUIZZARDI, 2007) (DRAGAN et al., 2006):

- Comunicação e colaboração entre pessoas: ontologias reduzem os conflitos conceituais e terminológicos dentro da organização, uma vez que proveem uma conceituação unificada para a mesma;
- Formalização: devido à natureza formal de uma ontologia, há a eliminação de contradições e inconsistências.
- Interoperabilidade entre sistemas: ontologias podem ser utilizadas na tradução de diferentes representações de bases de dados, fornecendo uma conceituação única a partir da qual outras conceituações se normalizam.
- Representação do conhecimento e reúso: ontologias formam um vocabulário de consenso e representam o conhecimento de forma explícita no seu mais alto nível de abstração, possuindo um elevado potencial de reúso.

3.5.1. Uso de Ontologias na Engenharia de Requisitos

A Engenharia de Requisitos (ER) exige muitos esforços na elaboração de modelos e uma forma de minimizar esses esforços é inserir abordagens de reúso nesse processo. Durante a ER, desenvolvem-se diversos diagramas. Dois dos principais diagramas utilizados já no levantamento de requisitos são os diagramas de classes, que proveem uma visão estrutural do

sistema, e os diagramas de casos de uso, que tratam de uma visão funcional, ou seja, uma visão comportamental que engloba atores e principais funcionalidades do sistema.

Nardi (2006), como citado no Capítulo 2, propõe uma abordagem que fez uso de ontologias, assim como padrões de análise e modelos de projetos similares anteriores, como elementos de reuso para análise de domínio para geração de modelos de domínio. Conforme essa abordagem, em (FALBO et al. 2007), um processo foi definido, como mostra a Figura 3.22, para direcionar o processo de ER quanto ao reuso de modelos estruturais, incluindo ontologias de domínio.

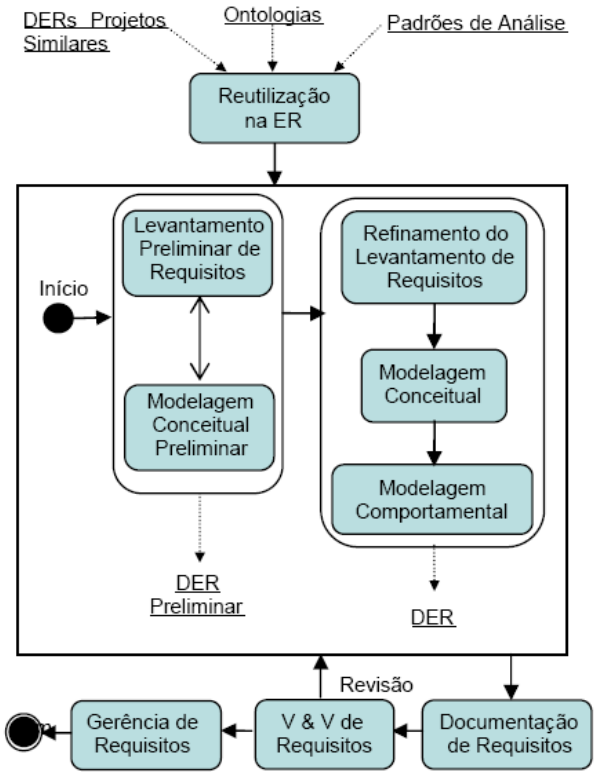


Figura 3.22: Processo de Reutilização de conhecimento na ER (FALBO et al. 2007).

Kaiya e Saeki (2006) apresentam um método de levantamento de requisitos chamado ORE (*Ontology-based Requirements Elicitation*). Eles sugerem que ontologias de domínio sejam utilizadas como conhecimento de domínio, pois contêm regras semânticas que dão significado para sentenças de requisitos. Dessa forma, ao usar regras de inferência é possível descobrir quais requisitos devem ser adicionados ou excluídos para garantir completude e consistência de uma versão de requisitos.

Ontologias de tarefa capturam aspectos comportamentais e estruturais que podem ser usados como base para o entendimento e levantamento de requisitos, bem como para a geração de modelos comportamentais. Poucos trabalhos tratam do reúso de conhecimento de

tarefa na Engenharia de Requisitos e estes servem como base para fortificar a abordagem apresentada no Capítulo 5 deste trabalho.

Zong-Yong e outros (2007) propõem uma estrutura, que usa múltiplas ontologias para o levantamento de requisitos, baseada na modelagem de conhecimento de KADS (BREUKER; VAN DE VELDE, 1994). A ideia básica é combinar ontologias de fundamentação, ontologia de domínio e ontologia de tarefa em um metamodelo para direcionar e padronizar o processo de levantamento de requisitos, como ilustra a Figura 3.23.

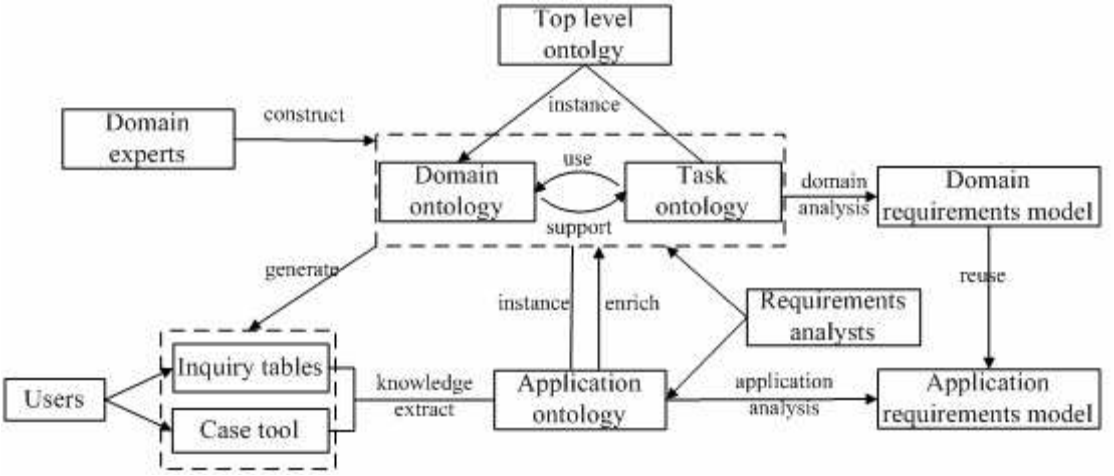


Figura 3.23: Estrutura de utilização de ontologias proposta por (ZONG-YONG et al., 2007)

Zlot e outros (2002), apesar de estarem focados na representação de ontologias de tarefa e sua combinação com métodos de solução de problemas, definiram uma relação de mapeamento entre os elementos da estrutura de descrição de tarefas proposta e especificações de caso de uso. Na estrutura de Descrição de Solução de Problemas (DSP) apresentada, o conhecimento de tarefa é dividido em três níveis, a saber: verbal, conceitual e formal. O nível verbal descreve todas as etapas para a resolução do problema em linguagem natural e a partir dele são derivadas descrições de casos de uso (Figura 3.24), conforme as heurísticas citadas no Capítulo 2. No nível conceitual, utiliza-se um algoritmo para representação do fluxo de controle das tarefas, que é utilizado para gerar o fluxo de eventos dos casos de uso (Figura 3.25). Além disso, são modelados os conceitos e relações envolvidos no problema, utilizando a linguagem LINGO (FALBO, 1998). Finalmente, o nível formal trata os conceitos e relações definidos e as inferência necessárias para solucionar a tarefa, utilizando a linguagem Prolog para esse fim; porém essa informação não é utilizada diretamente no processo de desenvolvimento.

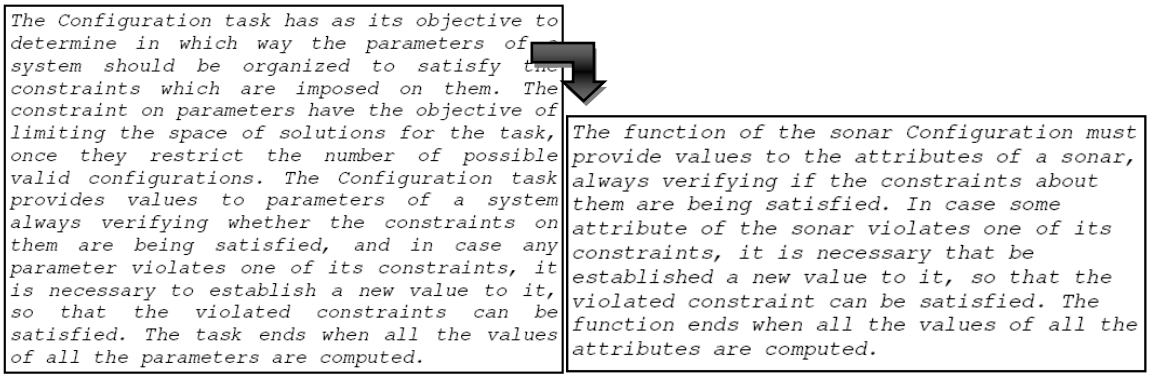


Figura 3.24: Derivação da descrição de casos de uso a partir do nível verbal (ZLOT et al., 2002).

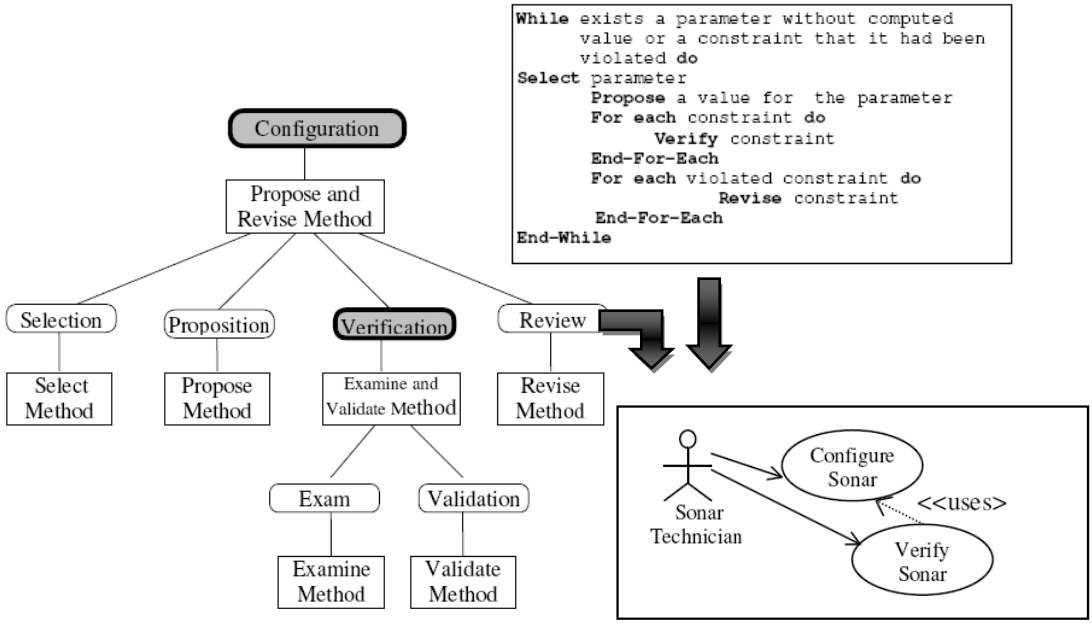


Figura 3.25: Derivação dos casos de uso a partir do nível conceitual (ZLOT et al., 2002).

A ER, em resumo, exige muitos esforços com levantamento de requisitos e com a elaboração de modelos que explicitem os requisitos dos sistemas. Uma forma de minimizar esses esforços é inserir abordagens de reúso nesse processo. O reúso permite que esforços investidos anteriormente sejam reutilizados ao se utilizar ontologias como base para capturar conhecimento ao invés de construir modelos a partir do zero. Além disso, com essa abordagem, é possível aumentar a qualidade e minimizar o risco de que erros sejam cometidos.

3.6. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas as definições para o termo ontologia, bem como algumas tipologias desenvolvidas para sua classificação. A classificação de Guarino (1998)

foi enfatizada, pois esta foi adotada e estendida na abordagem apresentada neste trabalho, com a introdução de ontologias de classes de aplicações.

Sobre as ontologias de tarefa, foram apresentados os trabalhos que tratam de sua definição e representação e foi destacada a falta de padronização que existe tanto para a organização do conhecimento de tarefa quanto no que se refere a linguagens utilizadas para sua representação. Diante dessas carências é proposto no Capítulo 4 um perfil UML para representação de conhecimento de tarefa e, portanto, de ontologias de tarefa.

Com relação a ontologias de fundamentação, foi destacada a Ontologia de Fundamentação Unificada (*Unified Foundational Ontology* – UFO), que foi apresentada com o intuito de deixar o leitor a par de sua conceituação, pois isso será necessário para o entendimento da linguagem de representação proposta no Capítulo 4.

Para finalizar, foram discutidas algumas aplicações de ontologias no contexto do desenvolvimento de software, com destaque para a Engenharia de Requisitos, destacando-se a importância e as vantagens dessa utilização. No Capítulo 5, é apresentada uma abordagem de reúso de conhecimento de tarefa na Engenharia de Requisitos, com o intuito de derivar modelos comportamentais de um sistema.

Foram apresentadas, portanto, informações importantes para fundamentar, justificar e apoiar as ideias apresentadas neste trabalho.

CAPÍTULO 4. ONTOLOGIAS DE TAREFA: CONSTRUÇÃO E INTEGRAÇÃO COM ONTOLOGIAS DE DOMÍNIO

Diante do exposto no capítulo anterior, nota-se a necessidade de uma abordagem de organização e modelagem do conhecimento de tarefa. Este capítulo apresenta E-OntoUML, um perfil UML para a modelagem de ontologias de tarefa fundamentado semanticamente na Ontologia de Fundamentação Unificada (*Unified Foundational Ontology* – UFO). São definidos os elementos a serem capturados utilizando ontologias de tarefa e como eles devem ser organizados, bem como quais os elementos de modelo que podem ser utilizados para representá-los. Para finalizar, é apresentada uma abordagem de integração com ontologias de domínio, destacando como a linguagem de representação e a organização de conhecimento propostas facilitam essa integração.

4.1. INTRODUÇÃO

Um fator chave para capturar o conhecimento é ter modelos para representá-lo. Um modelo é uma simplificação de algo que se pode visualizar, manipular e raciocinar sobre ele (MELLOR et al. 2004). O uso de modelos gráficos é fortemente reconhecido como essencial para o entendimento, a comunicação e o reúso. Logo, modelos gráficos para representação de ontologias de tarefa são fundamentais.

Em relação ao desenvolvimento de ontologias de tarefa, diferentemente do desenvolvimento de ontologias de domínio, existe uma carência de métodos amplamente aceitos para captura de seus elementos. Existem poucos trabalhos que tratam de ontologias de tarefa e não existe uniformidade para representá-las. Assim, é necessário definir uma abordagem sistemática para a criação de ontologias de tarefa. Neste capítulo procura-se dar um passo nesta direção, propondo modelos para organizar e representar o conhecimento de tarefa. É apresentado E-OntoUML um perfil de modelagem que permite representar aspectos comportamentais das ontologias de tarefa, sendo este baseado na Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML) (OMG, 2007) e na Ontologia de Fundamentação Unificada (*Unified Foundational Ontology* – UFO). A UML é um padrão mundialmente reconhecido, o que favorece a aplicação do perfil proposto. A UFO, por sua vez, é uma ontologia de fundamentação que provê semântica de mundo real a linguagens de representação que se comprometam com suas distinções ontológicas, melhorando, assim, sua qualidade e evitando ambiguidades.

Para apresentar essa abordagem, este capítulo foi organizado da seguinte maneira: na Seção 4.2 discutem-se quais os elementos tipicamente considerados em ontologias de tarefa e

quais modelos permitem representar o conhecimento que ela captura; na Seção 4.3 é apresentada a descrição da Tarefa de Locação, que é utilizada como exemplo para ilustrar as ideias apresentadas neste capítulo; na Seção 4.4 são apresentados a forma como o conhecimento de tarefa é organizado e o perfil E-OntoUML para a modelagem comportamental de ontologias de tarefa; na Seção 4.5 é feita uma avaliação desse perfil de modelagem; na Seção 4.6 é apresentada a abordagem de integração de ontologias de tarefa e domínio; por fim, na Seção 4.7 são apresentadas as conclusões do capítulo.

4.2. ELEMENTOS TIPICAMENTE CONSIDERADOS EM ONTOLOGIAS DE TAREFA

Para propor uma linguagem de representação, precisa-se saber ‘o quê’ se pretende representar, isto é, quais os elementos e relações que existem no metanível do objeto de representação (GUIZZARDI, 2005). Uma linguagem de representação de tarefa deve, portanto, prover ao usuário um conjunto de primitivas que permita expressar os conceitos envolvidos em uma tarefa, comprometendo-se com a correta conceituação dos mesmos.

Como primeiro passo para o desenvolvimento de uma linguagem para este fim, olhou-se para linguagens de representação de ontologias de tarefa já existentes para identificar quais os elementos que podem ou devem ser incluídos numa linguagem que seja usada para especificar as necessidades de abstração em alto nível de tarefas. Foram considerados nessa análise os seguintes trabalhos: (LU et al., 2003), (ZLOT, 2002), (VAN WELIE et al., 1998) (ZONG-YONG et al., 2007), (TRAN; TSUJI, 2007) e (FENSEL et al., 2003).

As Figuras 4.1, 4.2 e 4.3 apresentam os metamodelos encontrados em (TRAN; TSUJI 2007) (VAN WELIE et al., 1998) e (ZONG-YONG et al. 2007), respectivamente. Eles buscam estruturar o conhecimento de tarefa e usá-los como base para a definição de linguagens para representação de ontologias de tarefa.

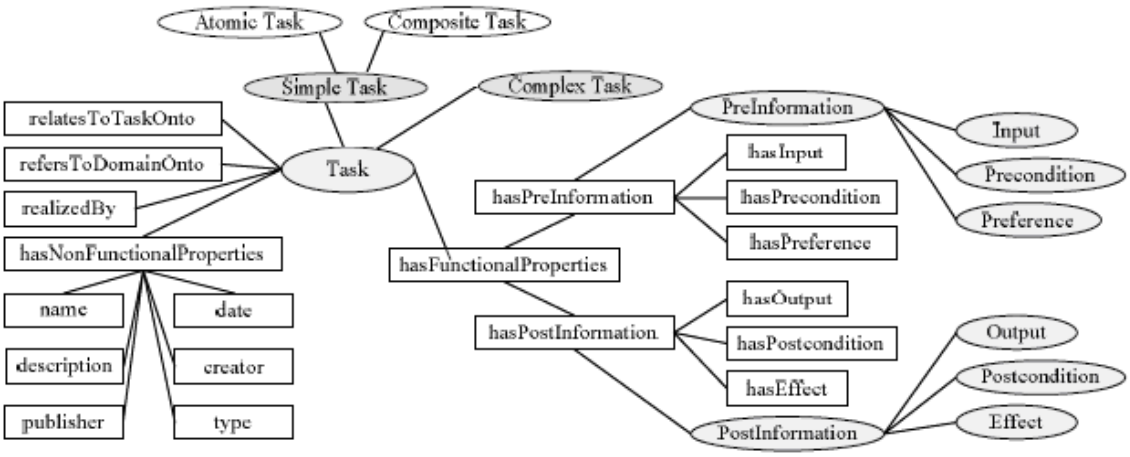


Figura 4.1: Estrutura do Conhecimento de tarefa proposta por (TRAN; TSUJI 2007)

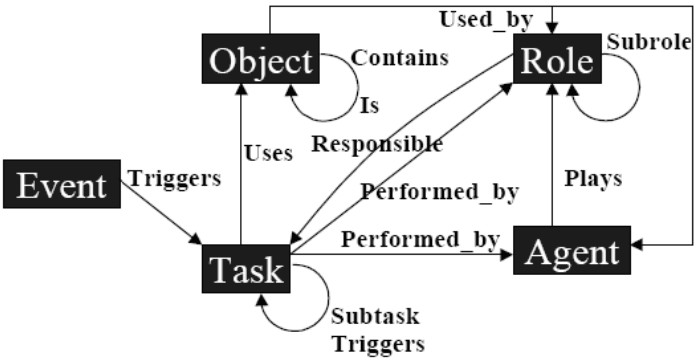


Figura 4.2: Metamodelo proposto por (VAN WELIE et al., 1998)

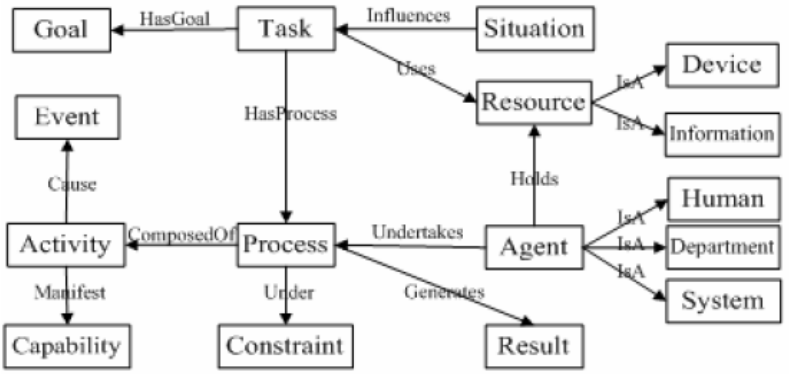


Figura 4.3 : Estrutura do conhecimento de tarefa apresentador por (ZONG-YONG et al. 2007).

Analisando os modelos das Figuras 4.1, 4.2 e 4.3, pode-se notar que os conceitos centrais na descrição de uma tarefa são a própria tarefa e sua decomposição em subtarefas (TRAN; TSUJI 2007), (LU et al. 2003), (ZLOT 2002), (VAN WELIE et al. 1998).

Segundo Tran e Tsuji (2007), que definem a linguagem OWL-T, uma tarefa pode ser simples (uma tarefa atômica ou composta) ou complexa (consiste de uma ou mais tarefas simples), como mostra a Figura 4.1. Uma tarefa simples, por sua vez, pode ser atômica (completada por uma única operação) ou composta (completada por uma composição de

várias operações). Os demais trabalhos não fazem distinções entre tipos de tarefas, ainda que van Welie e outros (1998) admitam a possibilidade de uma tarefa possuir subtarefas.

Ainda olhando para os aspectos comuns entre os modelos, percebe-se que para executar uma tarefa, é possível que haja restrições. Pré e pós-condições são exemplos dessas restrições, correspondendo, respectivamente, a condições que devem ser satisfeitas antes e depois da execução de uma tarefa ou subtarefa (LU et al. 2003) (TRAN; TSUJI 2007) (VAN WELIE et al., 1998). Em outras palavras, uma tarefa é influenciada por uma situação, que engloba pré e pós-condições, objetivos dos agentes que realizam a tarefa e o ambiente onde ela é realizada (ZONG-YONG et al., 2007) (VAN WELIE et al., 1998).

São conceitos importantes, também, o fluxo de controle da tarefa (LU et al., 2003) (ZLOT, 2002) (ZONG-YONG et al., 2007) (VAN WELIE et al., 1998), o que envolve a ordem em que elas são realizadas, como elas são sincronizadas e as decisões que influenciam esse fluxo (TRAN; TSUJI, 2007). Ainda existem trabalhos que diferenciam dois tipos especiais de tarefas: iterações, que correspondem à realização de uma mesma tarefa para cada elemento de um conjunto de objetos, e eventos, que acontecem sem o controle direto de um ator em determinado ponto no tempo (TRAN; TSUJI, 2007) (VAN WELIE et al., 1998).

No contexto de processos e tarefas, existem agentes que participam da execução da tarefa (LU et al., 2003) (VAN WELIE et al., 1998). Eles utilizam recursos, que podem ser dispositivos ou apenas informações (ZONG-YONG et al., 2007). Logo, ao definir o fluxo de uma tarefa, é necessário definir quais são recursos de entrada e saída e quais os seus papéis durante a execução da tarefa (VAN WELIE et al., 1998) (LU et al., 2003) (ZLOT, 2002) (TRAN;TSUJI, 2007). Também devem ser definidos os estados e as mudanças de estado dos objetos, bem como qual a sua colaboração e participação na execução da tarefa (VAN WELIE et al., 1998).

Pela breve análise feita acima, nota-se que ontologias de tarefa capturam o conhecimento a respeito do processo que é realizado para resolver um determinado problema e também deixam explícitos quais os objetivos e papéis que guiam esse processo. Ela agrega, portanto, elementos de modelagem de processos de negócio e modelagem de objetivos. Atualmente, para representar cada um desses elementos há padrões mundialmente reconhecidos.

Para a representação de modelos de processos de negócios, dentre as diversas linguagens de modelagem existentes, duas são mais amplamente utilizadas, a saber: UML (OMG, 2007), mais especificamente seu Diagrama de Atividades, e BPMN (*Business Process Modeling Notation*) (OMG, 2006), mais especificamente seu Diagrama de Processo de

Negócio. Já para modelagem de objetivos, a linguagem Tropos (BRESCIANI et al., 2004) é bastante difundida.

Como ontologias de tarefa englobam elementos dessas duas áreas de modelagem, essas linguagens e seus contextos de aplicação foram investigados para verificar a adequação de seu uso para a representação de ontologias de tarefa. Para isso foram enumerados os elementos envolvidos numa ontologia de tarefas e métodos de solução de problemas, segundo as referências anteriormente citadas, e foi montada a Tabela 4.1. Nessa tabela, podem-se verificar quais os elementos tipicamente relacionados ao conhecimento de tarefas e quais diagramas de modelagem de processos de negócios e de modelagem de objetivos são capazes de representá-los.

Tabela 4.2 – Elementos de ontologias de tarefa e diagramas para sua captura e representação.

Elementos	Diagrama de Atividades da UML	Diagrama de Processos de Negócio do BPMN	Tropos
Tarefas e Subtarefas	X	X	X
Fluxo de controle	X	X	-
Ordem	X	X	-
Sincronização	X	X	-
Condição / Decisão	X	X	X
Eventos	X	X	-
Pré-condição	X	X	-
Pós-condição	X	X	-
Objetivos	-	-	X
Objetos	X	X	X
Entradas / Saídas	X	X	X
Estados	X	X	-
Agentes	X	X	X

Algumas observações devem ser feitas sobre a Tabela 4.1, sobretudo em relação à forma como os elementos são representados e quais as restrições de sua representação. Em relação aos Diagramas de Atividades da UML, vale lembrar que os aspectos de sincronização são possíveis de representar utilizando-se padrões de *workflow* (WHITE, 2004), como ocorre com os Diagramas de Processo de Negócio da BPMN. Ainda, nos diagramas de atividades, usando anotações é possível representar objetivos, mas não se trata de um padrão da linguagem e sim uma adaptação possível, tratando objetivos como pré e pós-condições. A BPMN, por sua vez, também carece de um símbolo para pós-condições, permitindo apenas o uso de anotações para isso. Já Tropos carece de muitos dos elementos relativos a processos e alguns deles são representados de forma bem restrita, como é o caso da modelagem da

utilização de recursos, em que não é definido exatamente quando um recurso é entrada ou saída de uma tarefa.

Como se pode notar pela Tabela 4.1, no que tange à representação da decomposição de tarefas, os diagramas envolvidos na representação de processos são mais completos e permitem, ainda, adaptações para a representação de objetivos. Vale ressaltar que a definição dos objetivos é feita com um nível de detalhamento muito inferior do que em Tropos, que se propõe principalmente a representá-los. Neste trabalho, como o foco é o controle de fluxo das tarefas, optou-se, portanto, por escolher um dos tipos de diagramas de representação de processos.

Assim, foram considerados para representar a perspectiva comportamental de uma tarefa dois diagramas: os Diagramas de Atividades da UML e os Diagramas de Processos de Negócio da BPMN, ambos padrões amplamente aceitos. WHITE (2004) examinou esses dois tipos de diagramas considerando 21 padrões de *workflow* e concluiu que ambas as notações são adequadas para modelar a maioria dos padrões analisados. De acordo com White, eles proveem soluções similares, o que indica quão completas são as notações em sua apresentação. Eles inclusive compartilham de muitos símbolos comuns para o mesmo propósito. RUSSEL et al. (2006) também examinou a utilização de Diagramas de Atividades da UML 2.0 para a modelagem de processos de negócio. Suas conclusões indicam que diagramas de atividades podem ser utilizados para a modelagem de processos de negócio, mas eles não são adequados para todos os aspectos desse tipo de modelagem. Embora ofereçam suporte para as perspectivas de controle de fluxo e de dados, eles modelam aspectos organizacionais e relativos a recursos de forma bastante limitada. Ainda de acordo com RUSSEL et al. (2006), essas limitações são compartilhadas com a maioria dos outros formalismos de modelagem de processos de negócios e refletem a grande ênfase que tem sido dada ao controle de fluxo e à perspectiva de dados nas notações de modelagem de processos.

Como um dos propósitos deste trabalho é facilitar a integração de ontologias de tarefa com ontologias de domínio e pelo fato de Diagramas de Classes da UML serem utilizados para representar ontologias de domínio, decidiu-se usar os Diagramas de Atividades da UML para representar a parte comportamental das ontologias de tarefa (MARTINS; FALBO, 2008).

Antes de apresentar o perfil UML proposto para representar ontologias de tarefa, a próxima seção discute brevemente a tarefa de locação, que é utilizada posteriormente para ilustrar o uso do perfil proposto na modelagem de tarefas.

4.3. A TAREFA DE LOCAÇÃO

Para ilustrar a abordagem proposta neste capítulo, foi desenvolvida uma ontologia da tarefa de locação, que descreve, de forma genérica, independente de domínio e aplicação, as subtarefas, o controle de fluxo e os papéis que estão envolvidos na execução de uma tarefa dessa natureza.

Locação é uma tarefa que é recorrente em diversos domínios. Por exemplo, em uma biblioteca, usuários pegam livros emprestados, em locadoras de veículos clientes alugam carros, em imobiliárias locatários alugam imóveis e assim por diante. Em todos os casos, existem dois tipos principais de subtarefas: o empréstimo e a devolução de um item.

Independentemente do domínio no qual a tarefa ocorre, a locação inicia-se quando o locatário, dentre os itens disponibilizados para locação pelo locador, escolhe aquele que deseja locar. Com o objetivo de formalizar a locação, um contrato é estabelecido entre o locador e o locatário e o mesmo é registrado para manter o controle da locação. Finalmente, o item é entregue ao locatário.

Na devolução, o locatário entrega o item ao locador, que verifica se alguma condição do contrato foi violada. Neste caso, de acordo com as regras estabelecidas no contrato, uma penalidade é aplicada e o locatário deve pagá-la. Finalmente, a devolução é registrada e a locação concluída.

4.4. MODELAGEM DE ONTOLOGIAS DE TAREFA

Para se propor uma abordagem para a representação de ontologias de tarefa, é importante definir dois pontos essenciais (MARTINS, FALBO, 2008): (i) a organização do conhecimento e (ii) uma linguagem que seja capaz de capturar os elementos necessários, respeitando a sua semântica. Para ilustrar as ideias apresentadas nesta seção, a tarefa de Locação, descrita na seção anterior, é usada como exemplo.

4.4.1. Organização do Conhecimento de Tarefa

O conhecimento de tarefa possui duas facetas: (i) a decomposição da tarefa e (ii) os papéis de conhecimento envolvidos na realização da mesma. A decomposição da tarefa consiste em dividir a tarefa em subtarefas, determinando objetivos e descrevendo o fluxo de controle entre as subtarefas (ZONG-YONG et al., 2007). Já os papéis de conhecimento são as especificações dos conceitos e relacionamentos que aparecem na tarefa de interesse (IKEDA et al., 1998).

A decomposição da tarefa e o seu fluxo de controle compreendem a visão comportamental de uma tarefa. A solução de problemas engloba um processo e, portanto, pode ser descrita como uma sequência de atividades que mudam o estado de objetos e são executadas por agentes (OMG, 2006). Como consequência, para representar a decomposição de tarefas e o fluxo de controle, são necessários modelos que sejam capazes de representar atividades (subtarefas), agentes e objetos (entradas e saídas).

Papéis de conhecimento, por outro lado, dão uma visão estrutural da tarefa. Eles representam papéis que entidades do domínio desempenham quando a tarefa é executada (GUARINO, 1998). Os conceitos envolvidos em uma tarefa são tipicamente papéis de conhecimento que entidades do domínio (agentes e objetos) podem exercer na solução de um problema envolvendo uma tarefa em um dado domínio (ZLOT et al., 2002). De fato, papéis de conhecimento ligam o conhecimento de domínio ao controle de fluxo da tarefa. Eles são a interface entre o conhecimento de domínio e o raciocínio do problema. Ainda, eles têm a importante função de permitir que as tarefas sejam especificadas usando termos independentes de um domínio de aplicação particular, facilitando o reúso dessas tarefas genéricas (BRUAUX et al., 2007). Assim, papéis de conhecimento representam uma visão estrutural do conhecimento de tarefa e devem ser claramente descritos, bem como seus relacionamentos devem ser explicitamente identificados e modelados.

Esses tipos de conhecimento (fluxo de controle da tarefa e papéis de conhecimento), ainda que muito inter-relacionados, capturam diferentes visões de uma tarefa. De fato, eles representam diferentes dimensões de modelagem, enfatizando visões particulares de uma mesma porção da realidade. Logo, são necessários diferentes modelos para representá-los e, por conseguinte, para representar ontologias de tarefa, precisa-se de modelos comportamentais e estruturais para representar essas duas dimensões do conhecimento.

Conforme discutido na Seção 4.2, neste trabalho propõe-se o uso de diagramas da UML para representar ontologias de tarefa, sendo diagramas de atividades utilizados para capturar a decomposição de tarefas e como os papéis de conhecimento agem em sua execução, e diagramas de classes usados para modelar os papéis de conhecimento envolvidos e suas propriedades e relações (MARTINS, FALBO, 2008).

Porém, os elementos de modelo da UML não são semanticamente bem definidos, o que, muitas vezes, pode levar a usos com significados diferentes em diferentes modelos. Em outras palavras, não há compromisso com distinções ontológicas importantes como as feitas em uma ontologia de fundamentação.

Visando tratar esse problema em relação a diagramas de classes da UML, Guizzardi (2005) utiliza a ontologia de fundamentação UFO-A para definir o perfil UML de modelagem OntoUML, estando este perfil UML focado em aspectos estruturais e já vem sendo utilizado para representar ontologias de domínio (GONÇALVES et al., 2007) (GUIZZARDI et al., 2008a) (FALBO, NARDI, 2008). Esse perfil define estereótipos para elementos de modelo de diagramas de classes da UML e pode ser utilizado para representar os papéis de conhecimento de tarefa. Porém, para a parte comportamental, não foram encontrados perfis UML estabelecidos com base em uma ontologia de fundamentação. Assim, neste trabalho, seguindo uma linha análoga à do trabalho de Guizzardi (2005), utilizam-se os mecanismos de extensão da UML para adicionar a semântica da UFO a elementos de modelo de diagramas de atividades da UML. Dessa forma, busca-se estabelecer uma semântica clara para esses elementos de modelo, por se comprometerem ontologicamente com uma ontologia de fundamentação.

4.4.2. Modelagem de Papéis de Conhecimento envolvidos em Tarefas

Como previamente mencionado, uma vez que descrever o comportamento envolve referenciar os papéis de conhecimento, é necessário definir como representá-los. De maneira geral, qualquer modelo estrutural capaz de representar conceitos, relações, propriedades e certas restrições, tais como multiplicidades, pode ser usado para representar papéis de conhecimento.

Com o objetivo de manter os padrões já existentes e para facilitar a integração com ontologias de domínio, optou-se por modelos tipicamente usados para a construção de ontologias de domínio. Estes podem ser aplicados para a representação de papéis de conhecimento em ontologias de tarefa (MARTINS, FALBO, 2008).

Com respeito a modelos estruturais usados para representar ontologias de domínio, podem-se citar Diagramas de Classes da UML (CRANEFIELD; PURVIS, 1999) e extensões dos mesmos, tais como o perfil UML proposto no método SABiO (MIAN; FALBO, 2003) e OntoUML, o perfil UML fundamentado em UFO (GUIZZARDI, 2005). Como é de interesse deste trabalho considerar os quatro tipos de ontologias propostos por Guarino (1998), optou-se por utilizar a OntoUML, pois desta forma os modelos gerados já incluirão conceitos da ontologia de fundamentação UFO por esta ter sido considerada na definição dessa linguagem.

A Figura 4.4 mostra os principais conceitos e relações envolvidos em uma tarefa de locação usando o perfil OntoUML. Em uma **locação**, o **locador** empresta um **item** ao **locatário**. A **locação**, portanto, é um conceito que conecta **locador, locatário** e **item** e por

isso é tida como um ***modo relacional universal*** (*relator universal)* (GUIZZARDI et al., 2008a). Já **locador**, **item** e **locatário** são papéis que serão desempenhados por diferentes entidades de acordo com o domínio de realização da tarefa, logo são ***papéis mistos*** (*rolemixins*) segundo a UFO.

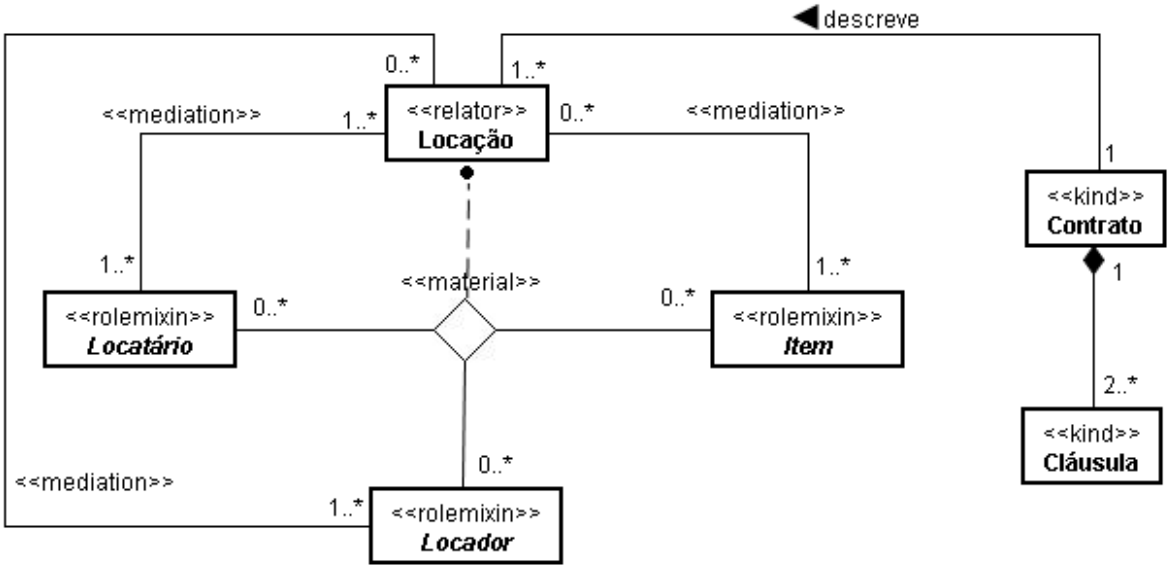


Figura 4.4: Papéis de Conhecimento da Ontologia de Tarefa de Locação

A **locação** é governada por um **contrato** que é composto de (*componentOf*) de **cláusulas**. **Contrato** e **cláusula** são artefatos com características próprias que os diferenciam uns dos outros e, portanto, possuem um princípio de identidade e são definidos como ***espécies*** (*kinds*). A relação de composição entre contrato e cláusula é definida na UFO como ***componente de*** (*componentOf*), que é uma relação que possui a metapropriedade de suplementação fraca (*weak supplementation*), que indica que o todo (contrato) tem de ser constituído por pelo menos duas partes disjuntas (cláusulas) (GUIZZARDI, 2005).

Vale ressaltar na Figura 4.4 que algumas das relações também são estereotipadas. Conforme discutido no Capítulo 3, a UFO estabelece uma hierarquia de relações. As relações que ocorrem diretamente entre ***universais de substância*** (*substantial universal*) são ditas ***formais*** (*formal relation*) e não são estereotipadas. As relações entre ***universais de substância*** (*substantial universal*) que são intermediadas por um ***modo relacional*** (*relator*) são chamadas ***materiais*** (*materials)*, por exemplo, entre **item, locador** e **locatário**. Essas relações definem a existência de um ***modo relacional*** (*relator*), nesse caso a **locação.** Já as relações entre ***universais de substância*** (*substantial universal*) e ***modos relacionais*** (*relators*) são definidas como ***mediações*** (*mediations*). ***Mediação*** é uma relação de dependência existencial entre um ***modo relacional*** *(relator)* e o ***universal de substância*** (*substantial universal*) que ele

intermedia. Por exemplo, as relações de mediação da Figura 4.4 definem que a **locação** depende do **item**, do **locador** e do **locatário** para existir.

No diagrama da Figura 4.4, ainda podem ser mostradas as fases das entidades, quando relevantes. A Figura 4.5 ilustra a representação das fases da entidade *Item* em OntoUML. Deve-se notar que esse diagrama não captura as transições de estados, sendo necessário recorrer ao modelo comportamental para capturar essa informação.

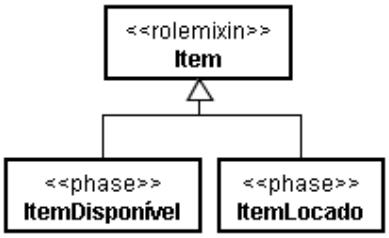


Figura 4.5: Fases de um Item de Locação

É importante enfatizar que os conceitos e relações mostrados na Figura 4.4 são genéricos, isto é, são independentes de domínio e aplicação. Em outras palavras, conceitos como locador, locatário e item correspondem a papéis que podem ser exercidos por entidades de diferentes domínios no qual a tarefa de locação pode ocorrer, mantendo suas relações e funções. Esse modelo descreve, portanto, apenas um dos tipos de conceitos citados por Mizoguchi e outros (1995b), chamados de substantivos genéricos (*generic nouns*) que refletem o papel de um objeto no processo de solução de um problema. Vale ressaltar, ainda, que esse modelo não é completo segundo a OntoUML, pois ele não leva em consideração restrições importantes da linguagem que só serão satisfeitas ao se integrar conceitos de domínio. Considerando os postulados da OntoUML, o **rolemixin** *Item*, mostrado na Figura 4.4, é abstrato e apenas agrega propriedades, mas para o modelo estar correto deveria haver em sua hierarquia um elemento com critério de identificação (*sortal*). Isso só ocorrerá quando este modelo for integrado a um domínio. Por exemplo, no domínio de livros haverá um livro que exercerá o papel de item de locação. Pode-se dizer, portanto, que se trata de um modelo OntoUML parcial.

4.4.3. Modelagem da Decomposição de Tarefa e Fluxo de Controle

O modelo mostrado na Figura 4.4 captura uma importante parte do conhecimento envolvido em uma tarefa de locação, que é seu aspecto estrutural. Entretanto, ele não captura aspectos comportamentais da tarefa. Quais são as subtarefas envolvidas na tarefa de locação?

Qual a sua ordem de precedência? Quando são usados os elementos mostrados na Figura 4.4? Todas essas questões permanecem sem resposta. Logo são necessários diagramas capazes de representar essas informações comportamentais que complementam o diagrama da Figura 4.4 e, conforme discutido na Seção 4.2, neste trabalho foram escolhidos os Diagramas de Atividade da UML.

Um diagrama de atividades é um modelo comportamental que permite capturar o fluxo de controle, isto é, a ordem de atividades (subtarefas) e suas condições de execução. Ele também permite representar a relação com papéis de conhecimento, mostrando como se comportam as instâncias desses papéis na execução de uma atividade. Ainda é possível modelar agentes que são responsáveis por executar as subtarefas e as entradas e saídas das mesmas, especificando os estados anteriores e posteriores de cada objeto que participou da subtarefa.

Porém, a UML é uma linguagem de especificação carente de uma semântica formal. Os significados dos seus elementos de modelo são dados por descrições textuais e modelos que descrevem sua sintaxe abstrata. Seus elementos nem sempre possuem a semântica correspondente aos conceitos do mundo real. Isso porque ela é uma linguagem que não se propõe a representar fielmente a realidade e sim é uma linguagem de especificação, voltada para a captura de elementos de modelagem para análise, projeto e implementação de sistemas, bem como de processos de negócios (OMG, 2007).

Pode-se dizer que uma linguagem de modelagem de tarefa deve prover ao usuário um conjunto de primitivas que permitam expressar diretamente os conceitos da tarefa, comprometendo-se com a correta conceituação dela. Entretanto, a UML não provê, por si só, essa conceituação esperada para os elementos de modelo usados para representar conceitos de uma tarefa. Mas então por que este trabalho propõe o uso de diagramas da UML? A UML possui mecanismos de extensão que possibilitam adicionar semântica aos seus modelos, permitindo modificar os elementos da linguagem para suprir necessidades de modelagem, conforme discutido no Capítulo 2.

Com base nessas informações, é proposto, neste trabalho, o perfil de modelagem de ontologias de tarefa E-OntoUML, que estende o metamodelo da UML, mais especificamente a parte que trata de elementos de um diagrama de atividades. Utilizou-se como base semântica a Ontologia de Fundamentação Unificada (UFO), apresentada no Capítulo 3. A UFO é uma ontologia que possui como principal objetivo prover uma semântica para elementos do mundo real, sempre se preocupando com termos genéricos que são instanciados em diferentes domínios, tarefas e aplicações. Logo, ela pode ser utilizada para prover semântica de mundo

real para os diagramas de atividades e para restringir possíveis interpretações de suas primitivas de modelagem.

Nas subseções seguintes, é apresentado o perfil E-OntoUML que recebeu este nome por ser focado em conceitos relativos a eventos e por ter objetivo análogo ao do perfil OntoUML, contudo com foco em eventos. De maneira análoga ao feito em (GUIZZARDI, 2005), utilizam-se os mecanismos de extensão da UML para adicionar as distinções semânticas feitas pelas UFOs B e C aos Diagramas de Atividade da UML. O perfil é apresentado em partes que seguem a organização mostrada na Figura 4.6.

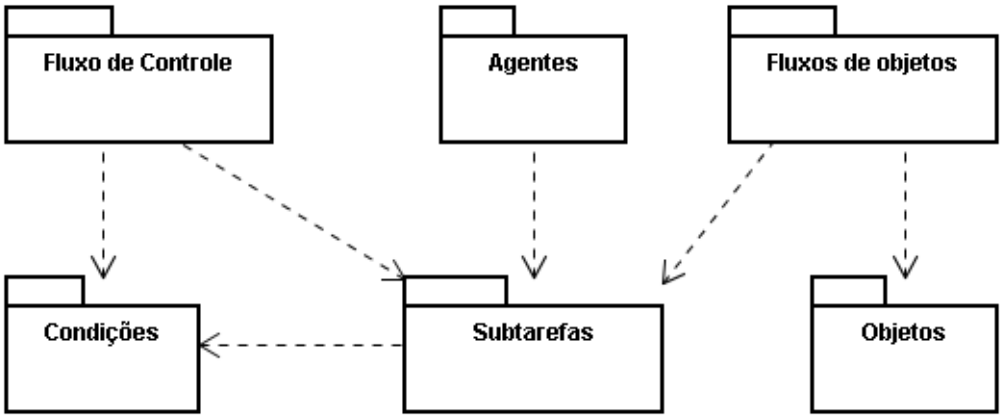


Figura 4.6: Organização dos elementos do perfil proposto

Inicialmente são apresentados os conceitos centrais que tratam da especificação de **subtarefas**, tais como eventos e atividades. A seguir, discute-se como modelar **condições** para a execução de uma tarefa. São apresentados também como representar **objetos** e quais os tipos e propriedades que são relevantes para a representação de uma tarefa. Posteriormente trata-se de como representar o **fluxo desses objetos** nas atividades, definindo qual a sua participação efetiva na tarefa. Na sequência, mostra-se como representar os **agentes** e seus papéis, e finalmente são representados os **fluxos de controle** entre as tarefas, definindo relações temporais e aspectos de sincronização.

Deve-se realçar que, para não se comprometer com a descrição e o significado dos elementos da UML, optou-se por sempre especializar esses elementos, considerando para a representação dos elementos de ontologias de tarefa apenas as classes estendidas e não as do próprio metamodelo da UML. Dessa forma, utiliza-se a notação provida pela UML e se compromete com a semântica da UFO. A estes elementos ainda foram adicionados relacionamentos, multiplicidades e especializações necessárias para tentar manter ao máximo

a aderência à UFO, porém fazendo as devidas adaptações para respeitar as restrições de modelagem especificadas pela UML.

Em todos os modelos apresentados nesta seção, conceitos da UFO são apresentados com o fundo cinza escuro, elementos do metamodelo da UML em cinza claro e os elementos resultantes, parte integrante do perfil E-OntoUML, têm o fundo branco. Além disso, no texto, conceitos da UFO aparecem destacados em negrito e itálico, elementos de modelo da UML são escritos em negrito e elementos do perfil proposto aparecem em negrito e sublinhados. Finalmente, em todas as subseções seguintes, uma vez apresentados os elementos de modelo de E-OntoUML, exemplos de sua aplicação no caso de estudo da tarefa de locação são apresentados.

4.4.3.1. Subtarefas

Uma tarefa pode ser definida como uma atividade executada para atingir um certo objetivo e a decomposição da tarefa em subtarefas é o ingrediente mais comum de modelos de tarefa (VAN WELIE et al., 1998).

As partes B e C da UFO propõem-se a definir semanticamente os conceitos relativos a eventos e entidades sociais, respectivamente. Definem, portanto, uma gama de elementos que são indispensáveis para a definição de qualquer conceito nesses domínios, como a definição de tarefas.

Dentre os conceitos da UFO, alguns são indispensáveis para definir tarefas, subtarefas e suas especializações. Subtarefas correspondem na UFO a ***ações*** (*actions*), pois são eventos que tem o propósito específico de satisfazer alguma intenção de um agente. Ações podem ser ***atômicas*** (*atomic action*) ou ***complexas*** (*complex action*), quando composta de duas ou mais ***participações*** (*participations*). Participações intencionais de agentes são denominadas ***contribuições de ação*** (*action contributions*) e uma ação complexa composta de contribuições de ações de diferentes agentes é denominada uma ***interação*** (*interaction*). Já uma participação de um objeto é chamada ***participação de recurso*** (*resource participation*).

A UML define ação como um elemento que é uma unidade fundamental para a execução de uma funcionalidade e atividade como uma sequência coordenada de ações (OMG, 2007). Para representar atividades e ações, que podem ser usadas para representar subtarefas, a UML provê os símbolos mostrados na Figura 4.7. O primeiro é usado para representar uma **ação** (*Action*), que representa um único passo em uma atividade, isto é, algo que não é mais decomposto. Uma ação é simples do ponto de vista da atividade que a contém, mas pode ser complexa em seu efeito (OMG, 2007). Já o segundo, corresponde a uma **Ação**

de Comportamento de Chamada (*CallBehaviorAction*) que invoca um comportamento diretamente. Quando trata da chamada de uma **atividade** (*Activity*), está-se indicando que a mesma ocorre naquele ponto do fluxo e que suas ações são modeladas em outro diagrama, sendo a atividade, portanto, de natureza complexa e composta de ações ou outras atividades.

Ação **Ação de Comportamento de Chamada**

Figura 4.7: Elementos de modelo que representam ações e atividades na UML.

Logo, considerando a UFO e o metamodelo da UML, foram definidos os elementos apresentados nas Figuras 4.8 e 4.9. Pode-se observar que alguns elementos são isomórficos à UFO (*AgentAction*, *AtomicAgentAction*, *Interaction*) e outros foram adicionados para adaptar os conceitos da UFO aos elementos de modelo de diagramas de atividades da UML com o objetivo de utilizar a sua notação (*CallActivity*, *AgentActivity*, *ObjectAgentAction*).

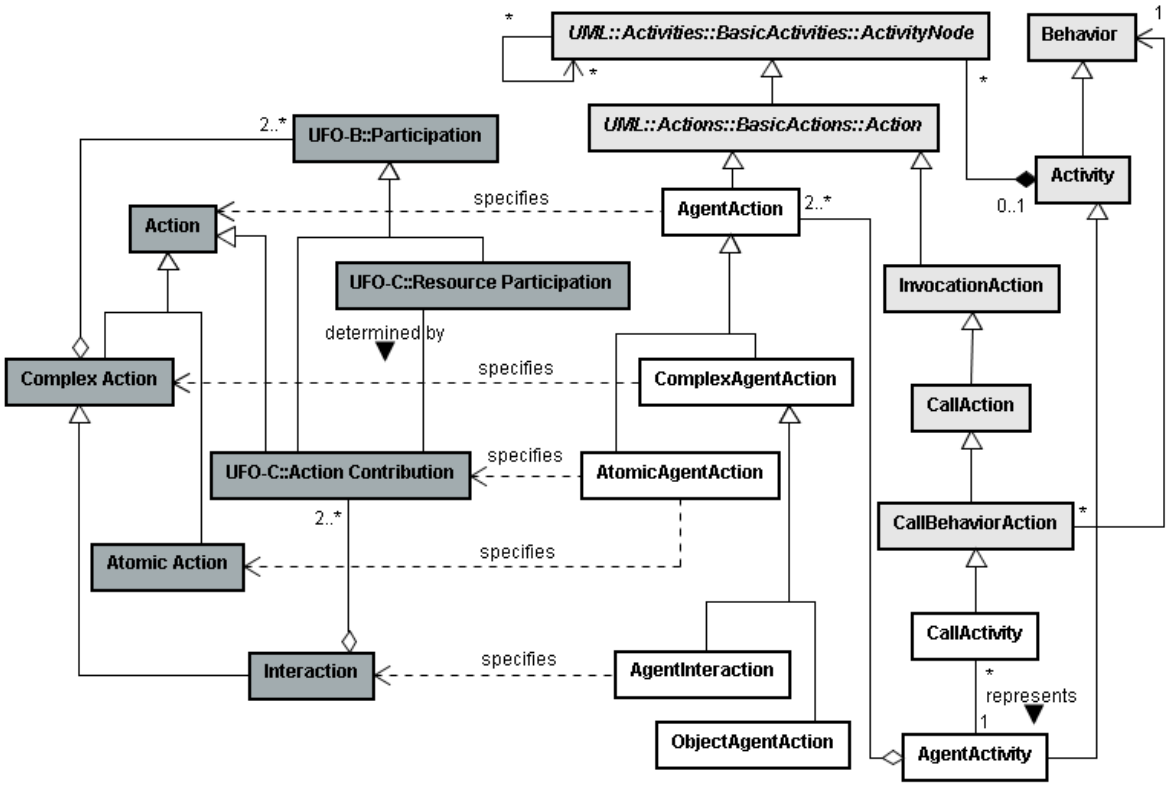


Figura 4.8: Elementos criados para representação de ações

<u>Ação de Agente</u> *(AgentAction)* é o elemento criado especializando as ações da UML para especificar ações da UFO, representando uma tarefa qualquer. Esse elemento adiciona ao elemento *Action* da UML a semântica de uma ***ação*** (*Action*) da UFO, ou seja, uma **<u>ação de</u>**

agente é representada pelo elemento de modelo **ação** (*Action*) da UML e é definida como um evento que muda o estado do mundo e que é guiada pela intenção de um agente.

Uma **ação de agente** *(AgentAction)* pode ter um efeito complexo ou atômico, dependendo do número de participações de agentes e de objetos envolvidos. Quando possui efeito complexo, ela é decomposta em subtarefas. Foram criadas, então, as especializações **Ação de Agente Atômica** *(AtomicAgentAction)* e **Ação de Agente Complexa** *(ComplexAgentAction)* para capturar as distinções da UFO em relação a ações.

Uma **Ação de Agente Atômica** *(AtomicAgentAction)* especifica uma tarefa que possui apenas uma participação de agente. Já uma **Ação de Agente Complexa** *(ComplexAgentAction)* inclui tarefas que são complexas em efeito ou que podem ser divisíveis, podendo ser de dois tipos principais: (i) **Interação** *(Interaction)* e (ii) **Ação de Agente com Objeto** *(ObjectAgentAction)*.

Uma **Interação** *(Interaction)* envolve a participação de mais de um agente, sendo considerada de efeito complexo por agregar participações de vários agentes, que na UFO são também ***ações***. Já uma **Ação de Agente com Objeto** *(ObjectAgentAction)* envolve a participação tanto de agentes quando de objetos. Logo, ela especifica uma ***ação complexa*** (*ComplexAction*) que é obrigatoriamente composta de pelo menos uma ***participação de recurso*** (*ResourceParticipation*) e de pelo menos uma ***contribuição de ação*** (*Action Contribution*). Vale observar que esses dois últimos elementos (interação e ação de agente com objeto), apesar de serem de natureza complexa, não envolvem a decomposição de uma tarefa em subtarefas em um diagrama de atividades usando o perfil E-OntoUML.

Uma **Atividade de Agente** (*AgentActivity*), por sua vez, é uma ação complexa não só no efeito, mas também pelo fato de ser divisível em subações, correspondendo ao que Tran e Tsuji (2007) chamam de tarefa complexa. Nota-se na Figura 4.9 que uma **atividade de agente** é composta de pelo menos duas ações, mais especificamente **ações de agente** *(AgentAction)*. Como citado anteriormente, uma **atividade** em um diagrama de atividades é graficamente representada pelo elemento de modelo **ação de comportamento de chamada** (*CallBehaviorAction*), logo foi necessário estendê-lo para representar uma **atividade de agente**, sendo criado o elemento **Atividade de Chamada** (*CallActivity*) que representa a chamada a uma **atividade de agente** *(AgentActivity)*.

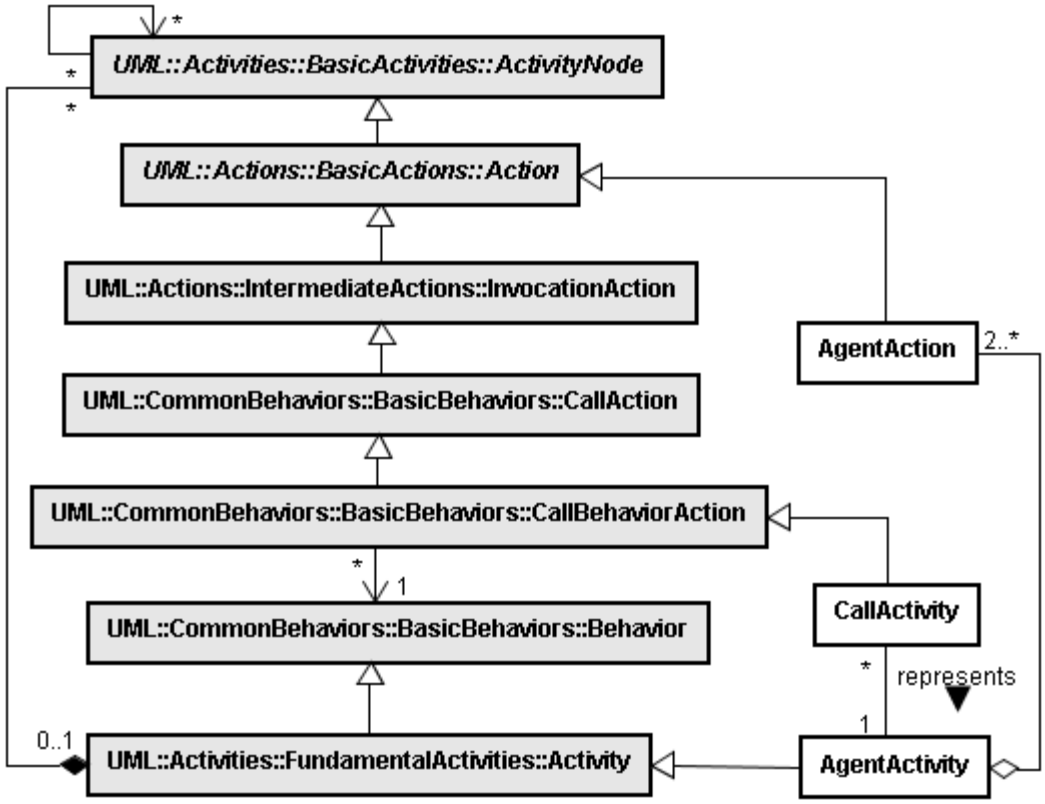


Figura 4.9: Representação de Atividades

As representações para **atividade de chamada** (*CallActivity*), **ação de agente atômica** (*AtomicAgentAction*), **interação** (*Interaction*) e **ação de agente com objeto** (*ObjectAgentAction*) são mostradas na Figura 4.10. Caso a tarefa seja de mais de um tipo, basta adicionar todos os estereótipos correspondentes, sendo que no caso de uma **ação de agente atômica** não faz sentido usar outros estereótipos. A **ação de agente com objeto** é o elemento mais comum e por isso, para evitar adição de muitos detalhes aos modelos, considerou-se este como valor *default*, tornando desnecessário o uso de um estereótipo quando a subtarefa não agregar outras classificações. Contudo, quando ela for também uma **interação** e/ou uma **atividade de chamada**, o estereótipo é necessário.

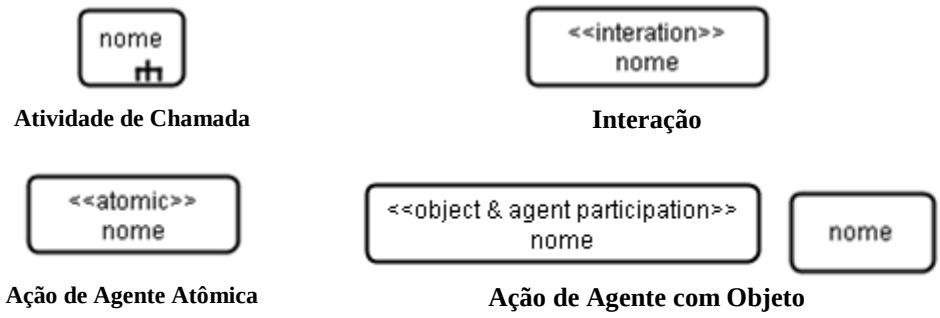


Figura 4.10: Representação da hierarquia de ações

Seja o exemplo da tarefa de Locação. Considerando a sua subtarefa "Emprestar item", mostrada na Figura 4.11, ela envolve os agentes locador e locatário, tratando-se, portanto, de uma **interação**. Como ela é também uma ação complexa decomposta em subtarefas, ela caracteriza-se como uma **atividade de chamada**, que representa uma ação complexa e decomponível. Por fim, ela é também uma **ação de agente com objeto** por possuir a participação de objetos como recursos, no caso, itens.

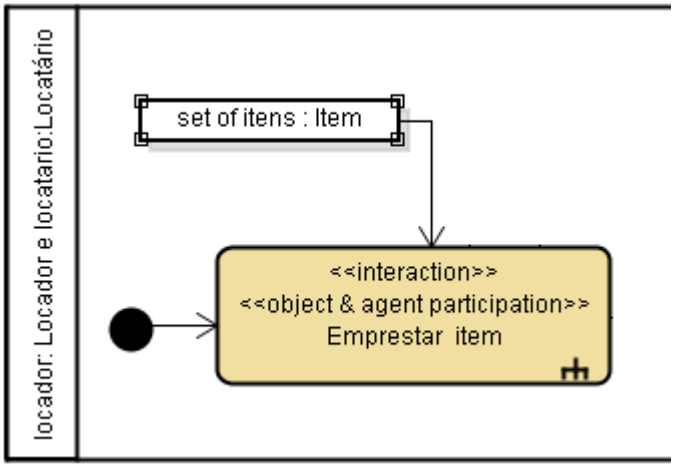


Figura 4.11: Representação para Interação, ação de agente com recurso e atividade de chamada.

4.4.3.2. Condições

Em UFO, condições podem ser vistas como ***proposições*** (*Proposition)*. Uma ***proposição*** é a parte de sentenças e cláusulas que é constante (LOSS et al., 2009). São "suposições" que podem ter um valor verdade (*true* ou *false*) (HERRE et al., 2006).

Nos diagramas de atividade, uma **condição de guarda** (*guard*) é uma condição avaliada em tempo real para determinar se o fluxo seguirá por um dado caminho. Ela é representada por uma sentença entre colchetes anexada a uma seta de **fluxo de controle** (*ControlFlow*) e

expressa uma situação que permite que o fluxo siga pelo caminho na qual ela está definida. A metaclasse que representa essa sentença no metamodelo da UML é a **especificação de valor** (*ValueSpecification).* Ela é usada para identificar valores em um modelo qualquer da UML. Vale lembrar que, de acordo com a UML, existem condições de guarda para todas as setas de fluxo de controle em um diagrama de atividades, mesmo aquelas em que uma condição de guarda não é explicitamente declarada, como mostra a Figura 4.12. Note que nessa figura a multiplicidade do seu relacionamento com **aresta de atividade** (*ActivityEdge)* é 1. Porém, como *default*, é adotado o valor verdadeiro (*True*) e, portanto, quando uma condição de guarda não for explicitamente modelada, assume-se que ela existe e seu valor é verdadeiro, indicando que se pode seguir o fluxo por aquele caminho. Assim, na maioria das vezes em que há fluxo sequencial, não são utilizadas condições de guarda explícitas. O uso de condições de guarda explícitas ocorre com frequência quando se trata de **nós de decisão** (*DecisionNode*).

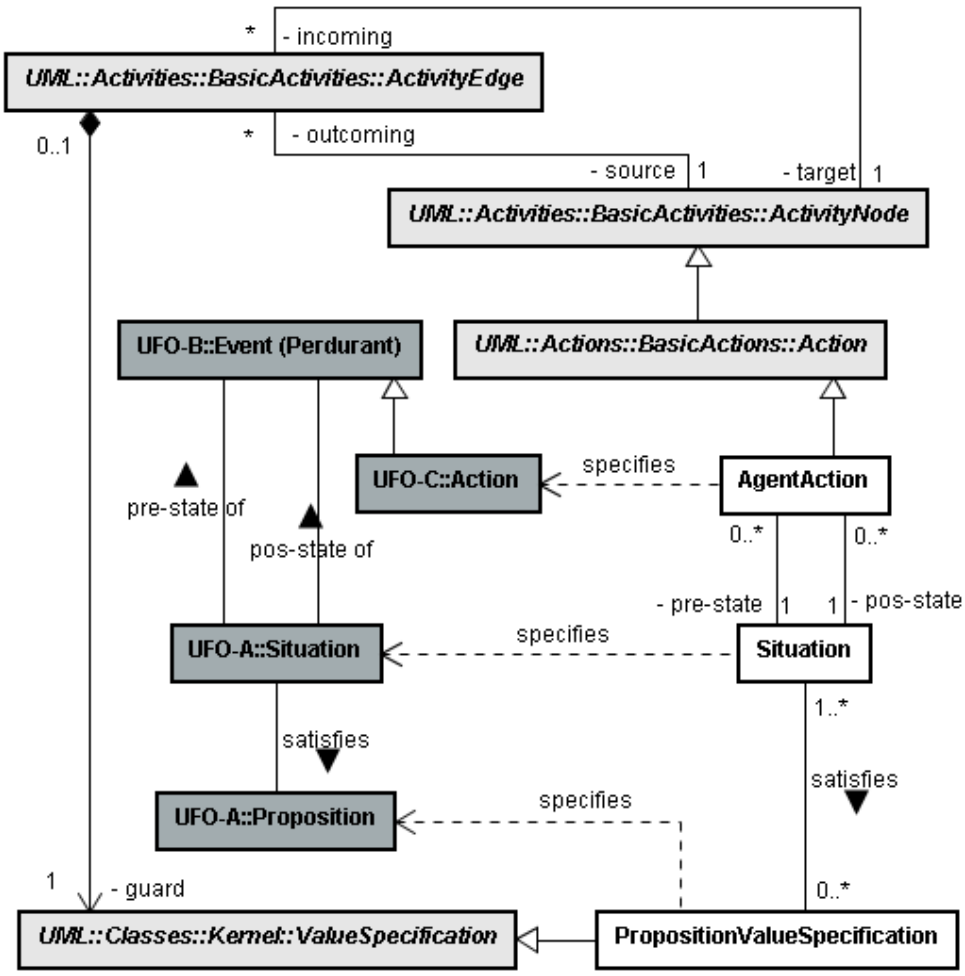


Figura 4.12: Situações

Na UFO, o conceito correspondente às condições de guarda é ***proposição*** (*Proposition*). Esse conceito é usado em E-OntoUML como base para restringir a interpretação do elemento de modelo **especificação de valor** (*ValueSpecification*) da UML, que é muito geral. Especializou-se, então, a metaclasse **especificação de valor** (*ValueSpecification*), gerando o elemento de modelo **especificação de valor de proposição** (*PropositionValueSpecification)*, que especifica o conceito ***proposição*** (*Proposition*) da UFO, como mostra a Figura 4.12. Esse elemento de modelo representa proposições que são satisfeitas pelo estado do mundo em um determinado ponto no fluxo de uma tarefa. A notação da UML para condições de guarda é usada em E-OntoUML sem alterações, pois não há outras distinções a serem feitas.

Outro ponto importante a ser observado é que, na UFO, uma situação (*Situation*) após uma ação (*pos-state*) pode satisfazer proposições (*Proposition*), representadas em E-OntoUML por condições de guarda, habilitando a realização da tarefa subsequente ao satisfazer a condição especificada. Para que essa relação seja válida também para o perfil E-OntoUML, criou-se um elemento de modelo similar em E-OntoUML chamado **situação** (*Situation*), que mapeia o conceito situação de UFO, com mostra a Figura 4.12. Esse elemento, contudo, não tem uma representação gráfica associada, tendo sido criado apenas para manter a consistência com a UFO. Como mostra a figura, uma ação de agente (*AgentAction*) possui uma situação (*Situation*) como pré-estado (*pre-state*) e uma situação como pós-estado (*pos-state*). Uma situação pode satisfazer várias especificações de valor de proposição (*PropositionValueSpecification*). Quando uma ação de agente leva a uma situação que é o pré-estado de outra ação de agente (e, portanto, satisfaz as correspondentes especificações de valor de proposição), então a segunda pode ser realizada.

Durante a tarefa de locação, por exemplo, o locatário deve escolher o(s) item(ns) que ele deseja locar dentre o conjunto de itens disponibilizados pelo locador. É nítido que duas situações podem ocorrer: (i) o locatário pode não encontrar nenhum item que lhe interesse (ii) ou escolher um item (ou alguns) para a locação. Dois fluxos são definidos para englobar essas possibilidades, como mostra a Figura 4.13. No primeiro caso, o fluxo da tarefa é interrompido e a locação não é efetivada. No segundo, um contrato é estabelecido para reger a locação. Note que a condição “algum item foi escolhido” é satisfeita pelo pós-estado (situação) da tarefa “Escolher item”, ao mesmo tempo em que é um pré-estado para a tarefa “Estabelecer contrato”.

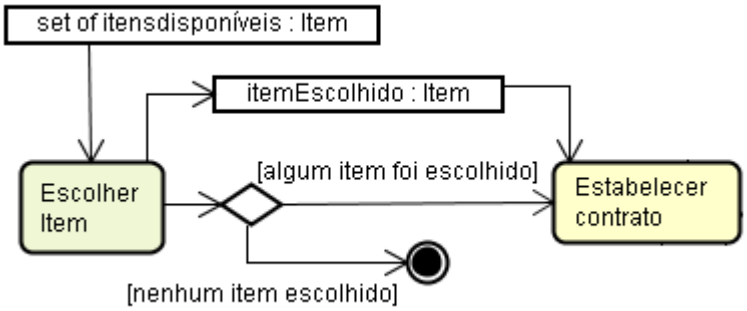


Figura 4.13: Uso de condições na Tarefa de Locação.

4.4.3.3. Objetos: Tipos e Estados

Durante a realização de uma tarefa, objetos são necessários e seus estados são alterados. Nesse sentido, a UFO define o conceito ***objeto*** (*Object),* para distinguir qualquer substancial que não seja um ***agente*** (*Agent*) (GUIZZARDI, 2006). Objetos (ou substanciais inanimados) (GUIZZARDI et al., 2008a), assim como agentes, podem ser ***objetos físicos*** (*physical object*) ou ***objetos sociais*** (*social object*) (GUIZZARDI et al., 2008a). Uma ***descrição normativa*** (*normative description*) é um tipo de objeto social que define uma ou mais regras/normas ***reconhecidas por*** (*recognized by*), pelo menos, um ***agente social*** (*social agent*). Objetos, como todo ***substancial*** (*substantial*), podem possuir estágios (estados) em sua história, o que a UFO trata com o conceito de ***fase*** (*Phase).*

Nos diagramas de atividades da UML, um **nó de objeto** (*ObjectNode)* é parte da definição do fluxo de objetos em uma atividade, podendo ser representado pelos símbolos mostrados na Figura 4.14, sendo o da direita uma sugestão da UML para a representação de um conjunto de objetos.

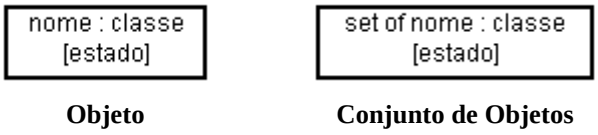


Figura 4.14: Representação de Objetos em Diagramas de Atividades da UML

Um **nó de objeto** indica uma instância de um particular **classificador** (*Classifier*), em um determinado estado. Um **classificador**, por sua vez, descreve elementos com características comuns, isto é, ele classifica instâncias. Essa relação entre **nó de objeto** e **classificador** ocorre, de fato, entre suas superclasses, respectivamente, **Elemento Tipado** (*TypedElement)* e **Tipo** (*Type*), como mostra a Figura 4.15. **Elemento Tipado** define um elemento que possui um **tipo** (*Type)* que serve como uma restrição de valores que o elemento

pode representar. No caso dos diagramas de atividade, o **tipo** é um **classificador**, mais especificamente uma **classe** (*Class)*, que descreve um conjunto de elementos que compartilham a mesma especificação de características, restrições e semântica (OMG, 2007).

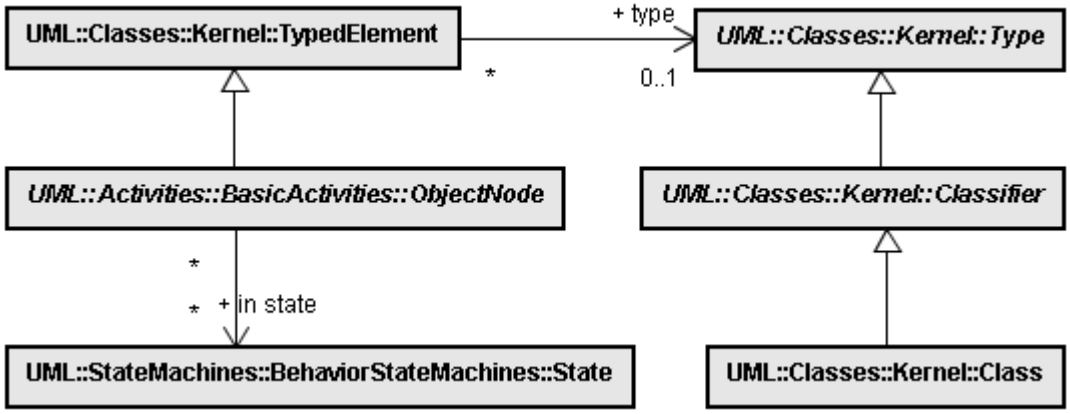


Figura 4.15: Relações definidas no metamodelo da UML para tratar de Objetos, seus tipos e sua representação.

O elemento **nó de objeto** da UML, portanto, representa uma visão instantânea (*snapshot*) de um ***objeto*** (*Object*) em determinado ponto de um processo, isto é, a situação de um ***objeto*** em um momento.

Na UFO-A existe o conceito de ***situação*** (*Situation*). Como citado anteriormente, esse conceito representa uma parcela da realidade. Situações são entidades complexas que são constituídas possivelmente por muitos ***indivíduos duradouros*** (*Endurants*), o que inclui os ***objetos***.

Em E-OntoUML criou-se o elemento **nó de situação de objeto** (*ObjectSituationNode*) para especificar o conceito de ***situação***, indicando um *snapshot* de um ***objeto*** em determinado momento da tarefa. Esse elemento representa uma instância de um ***objeto*** (*Object*) em um momento no tempo e em um estado específico. Ele trata do "estado das coisas" restrito ao "mundo" do objeto. Vale lembrar que um ***objeto*** (*Object*) é um ***indivíduo duradouro*** (*Endurant)* e, portanto, pode estar presente em uma ***situação*** (*Situation)*, como é expresso pelo relacionamento "***presente em***" (*present in*).

Seguindo a hierarquia dos ***objetos*** (*Object*) na UFO, foram criados ainda os seguintes elementos: (i) **nó de situação do objeto físico** (*PhysicalObjectSituationNode)*, que especifica a situação de um objeto físico, (ii) **nó de situação do objeto social** (*SocialObjectSituationNode*), que especifica a situação de um objeto social, e (iii) **nó de situação da descrição normativa** (*NormativeDescriptionSituationNode)*, que especifica a

situação de uma descrição normativa. Esses elementos mantêm a aderência com a hierarquia da UFO (Figura 4.16). Vale ressaltar que a hierarquia de objetos não é completa e, portanto, outras classificações podem ser adicionadas. Se isso ocorrer, para que o perfil mantenha o compromisso com a UFO, será necessário adicionar novos elementos de modelo à E-OntoUML.

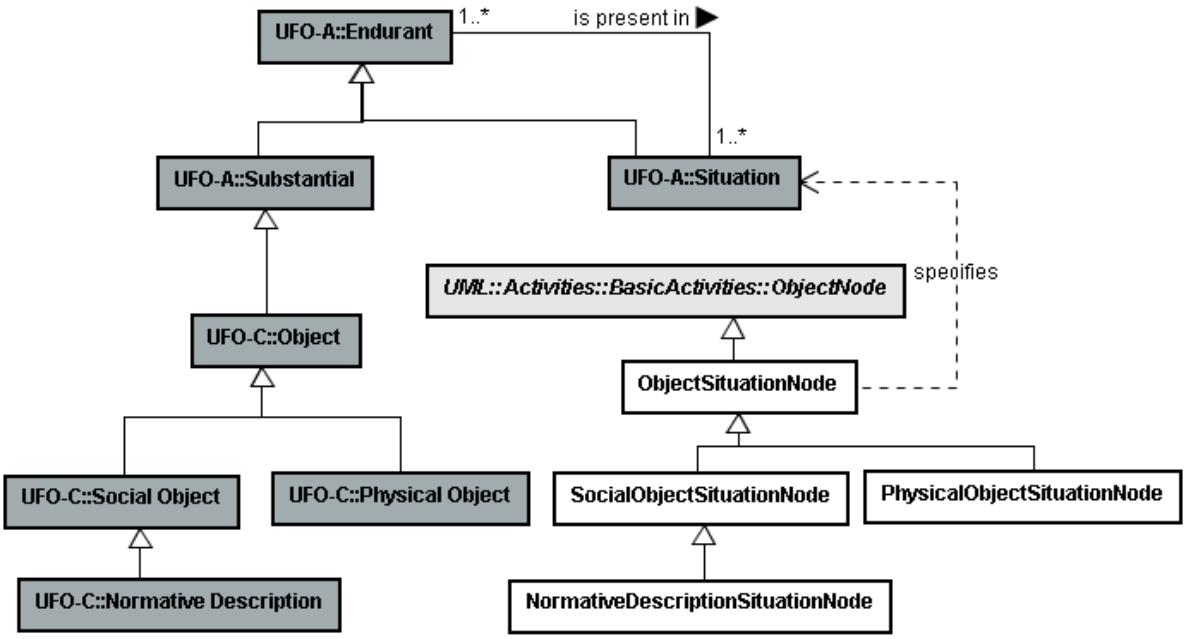


Figura 4.16: Elementos para representação de Situações de Objetos durante uma Tarefa.

Para representação dos elementos de modelo criados, utiliza-se a notação da UML para **nós de objetos**, complementada pelos estereótipos que indicarão qual o tipo do objeto cujo *snapshop* de sua instância está sendo representado, conforme notação mostrada na Figura 4.17.

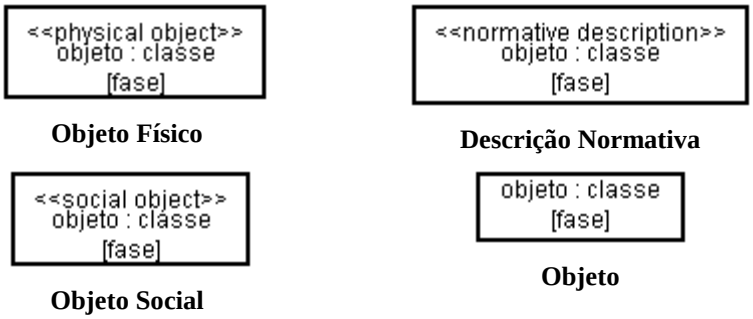


Figura 4.17: Representação de Situações de Objetos

Uma classe da UML pode ser vista como um ***universal*** *(Universal)* em UFO, pois ela define padrões de características que podem ser instanciados em um número de diferentes

indivíduos. Mais especificamente, no contexto da representação de classes de objetos, classes representam ***universais de substância*** (*Universal Substantials).*

No perfil OntoUML foi definida a hierarquia mostrada na Figura 3.6 (Capítulo 3) que, partindo de **classe** (*Class)* da UML, define metaclasses que são isomórficas à hierarquia dos ***universais de substância*** (*Universal Substantials*) da UFO. Conforme discutido na subseção 4.4.2, o perfil OntoUML é usado como base para a especificação do modelo de papéis de conhecimento de tarefas. Assim, para manter o alinhamento entre as duas perspectivas do conhecimento de tarefa (papéis de conhecimento e fluxo de controle das tarefas), os objetos representados no diagrama de atividades devem ser instâncias de conceitos presentes no modelo de papéis de conhecimento.

Ainda com respeito aos objetos é necessário definir a semântica para seus estados. Na UML o elemento de modelo usado para descrevê-los é chamado **estado** (*State*). Na UFO, estados de um objeto são capturados pelo conceito de ***fase*** (*Phase*). Assim, em E-OntoUML foi criado o elemento **fase do objeto** (*ObjectPhase)* que representa o estado de um objeto em um determinado momento da tarefa, como mostra a Figura 4.18.

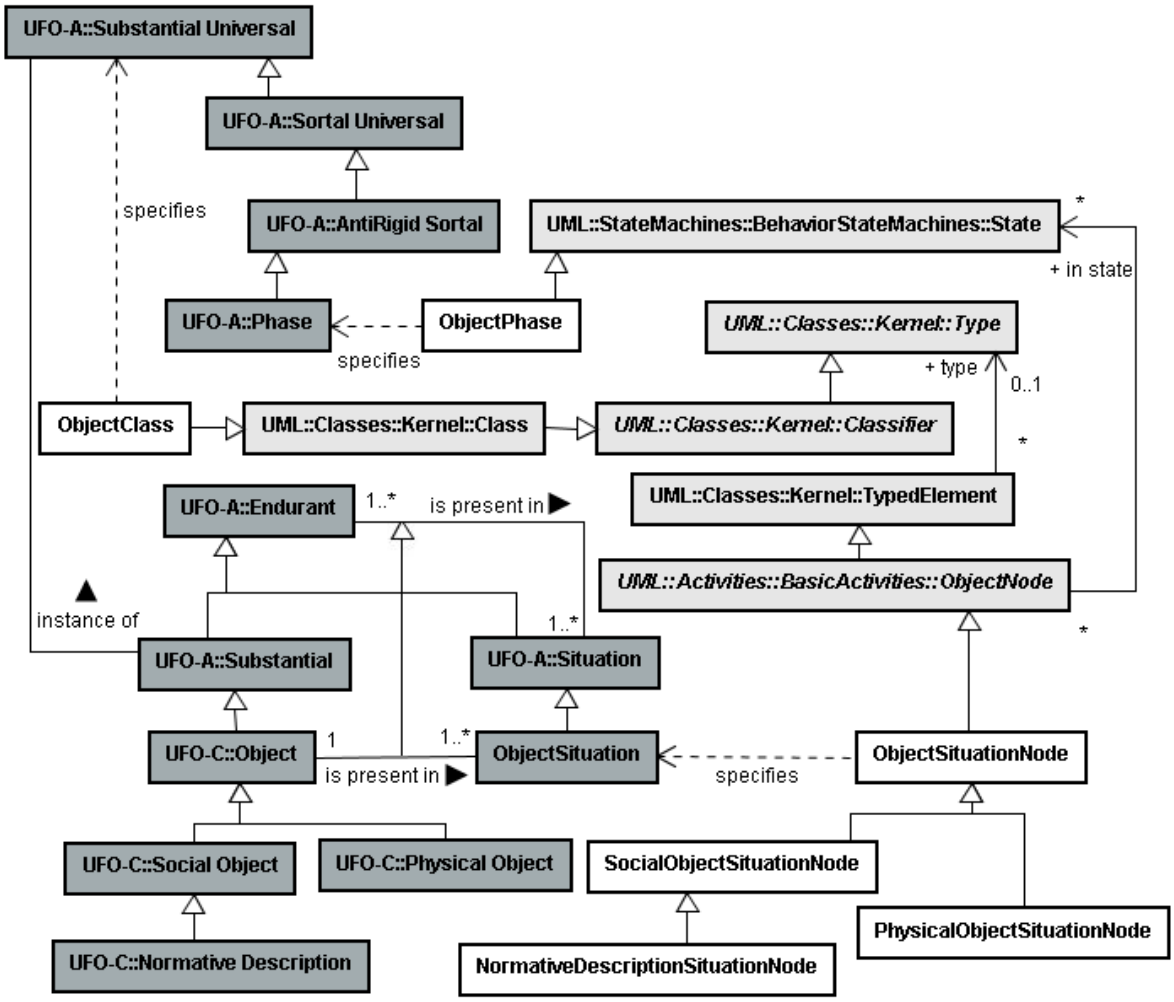


Figura 4.18: Objetos

Seja o exemplo da tarefa de locação, mostrado na Figura 4.19. Para emprestar um item e estabelecer um contrato para reger a locação, um conjunto de itens disponíveis é utilizado como insumo, um deles é selecionado e um contrato é estabelecido. Cada um desses objetos é representado, quando pertinente, por símbolos estereotipados com termos que indicam qual o conceito da UFO (B e C) eles instanciam. No exemplo, Contrato é uma descrição normativa. Os demais são apenas objetos.

Para cada instância representada, é definida a classe a que ela pertence, dentre aquelas definidas no modelo de papéis de conhecimento correspondente (Figura 4.4). Podem-se observar, ainda, os estados dos objetos. Por exemplo, a instância `contratoRegistrado` de `Contrato` está **vigente** após o empréstimo do item e a instância `itemLocado` de `Item` está **locado**, e, portanto, indisponível para outras locações.

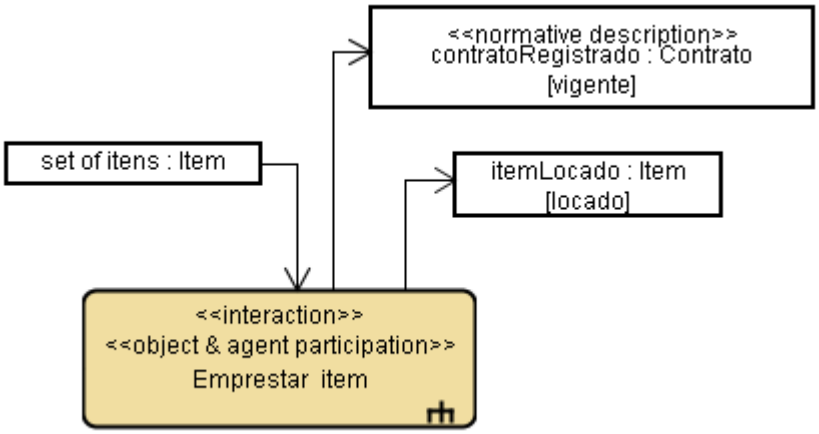


Figura 4.19: Representação de Objetos no perfil E-OntoUML.

4.4.3.4. Objetos: Entradas, Saídas e Tipos de Participações

A UFO estabelece que ***objetos*** (*objects*) podem participar de ações de diferentes maneiras. Para tratar as participações de objetos em ações é definido o conceito ***participação de recurso*** (*resource participation*). Uma ***participação de recurso*** pode ser de quatro tipos: ***criação*** (*creation*), ***término*** (*termination*), ***alteração*** (*change*) e ***uso*** (*usage*).

Os diagramas de atividade da UML permitem representar as entradas e saídas de uma tarefa, utilizando **arestas de atividade** (*ActivityEdge*). **Arestas de atividade** são conexões direcionadas (isto é, possuem uma origem e um destino) ao longo das quais símbolos (*tokens*) podem fluir. Graficamente, são representadas por setas que conectam dois **nós de atividade** (*ActivityNode*) (OMG, 2007). Para representar a participação de objetos em ações, deve-se conectar um **nó de objeto** à tarefa da qual ele participa, como ilustra a Figura 4.20.

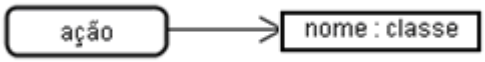


Figura 4.20: Representação da UML para Participações de Objetos.

Porém, utilizando apenas a notação da UML, não há como definir explicitamente qual o tipo da participação de um objeto em uma tarefa. A UFO, como discutido anteriormente, define quatro tipos de participações de objetos em tarefas (criação, término, uso e alteração) e esta diferenciação pode ser útil para complementar a semântica em diagramas de atividades, enriquecendo os modelos de tarefas.

Analisando os tipos de participação, pode-se observar que as setas que chegam e saem indicam um conjunto disjunto de participações. As setas que partem de um objeto e chegam a uma ação podem indicar o uso ou a destruição do objeto. Já as setas que partem de uma ação e chegam a um objeto indicam ou a criação ou a alteração desse objeto. Em ambos os casos, há a participação de objetos na ação e, portanto, trata-se de uma **ação de agente com objeto** (*ObjectAgentAction*).

Como mostra a Figura 4.21, para capturar essas distinções no perfil E-OntoUML, foram criadas duas classes: **aresta de entrada de objeto** (*IncomingObjectEdge*) e **aresta de saída de objeto** (*OutcomingObjectEdge*). A primeira tem como origem (*source*) uma **nó de situação de objeto** (*ObjectSituationNode*) e como destino (*target*) uma **ação de agente com objeto** (*ObjectAgentAction*), sendo especializada em **aresta de uso de objeto** (*UsageObjectEdge*), quando o objeto é apenas usado na ação, e **aresta de destruição de objeto** (*TerminationObjectEdge*), quando o objeto é consumido na ação e não mais existe após a sua realização. A segunda, **aresta de saída de objeto** (*OutcomingObjectEdge*), tem como origem (*source*) uma **ação de agente com objeto** (*ObjectAgentAction*) e como destino (*target*) uma **nó de situação de objeto** (*ObjectSituationNode*), sendo especializada em **aresta de criação de objeto** (*CreationObjectEdge*), quando o objeto é criado durante a realização da ação, e **aresta de alteração de objeto** (*ChangeObjectEdge*) quando o objeto é alterado durante a realização da ação.

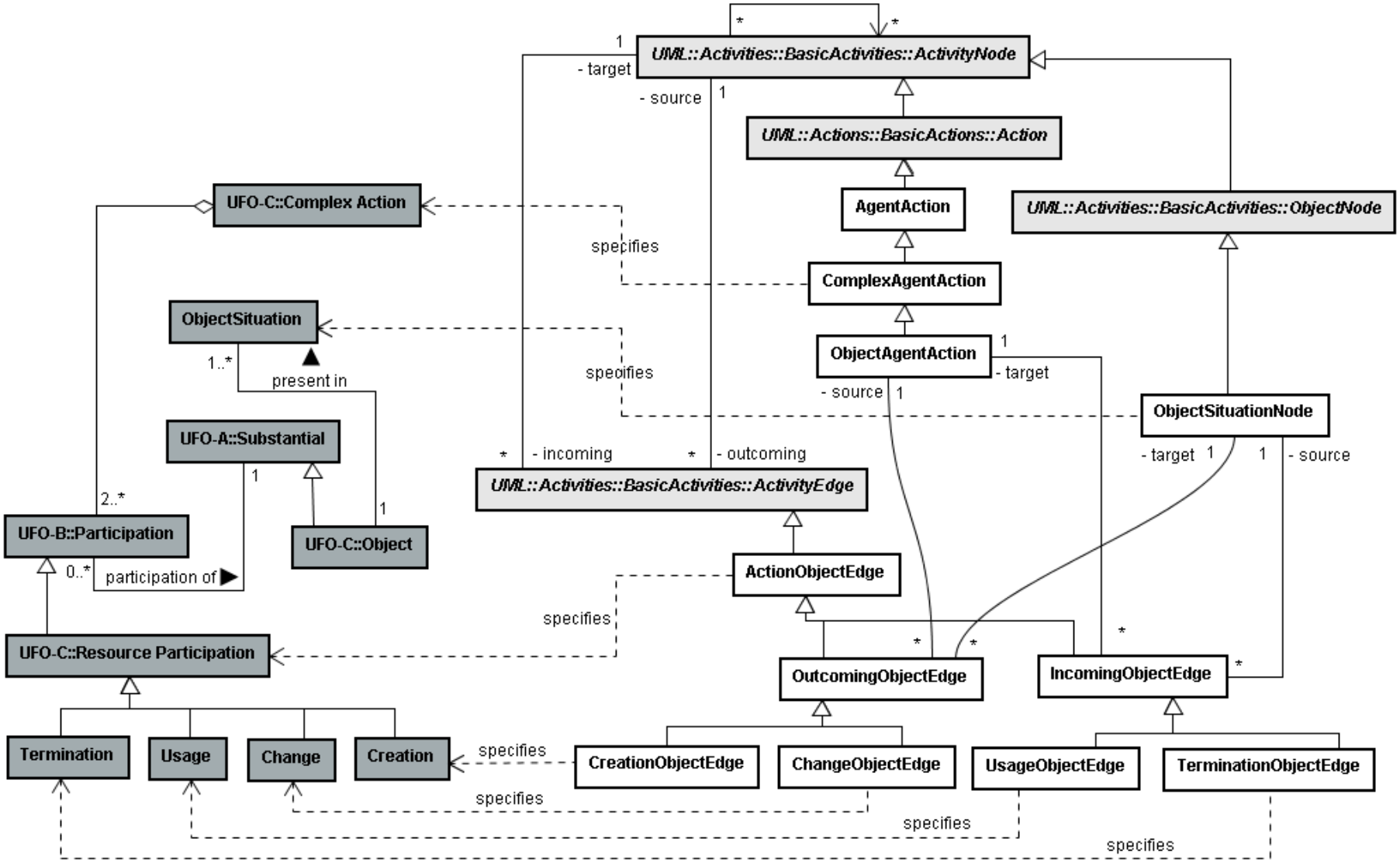


Figura 4.21: Tipos de Participações

Para representar esses elementos, utiliza-se a representação de **aresta de atividade** (*ActivityEdge*) acrescida de um estereótipo especificando o tipo da participação do objeto, como mostra a Figura 4.22. Porém, duas dessas participações são muito comuns e deixou-se facultativo o uso do estereótipo, a saber: ***Uso*** (*Usage*) e ***Alteração*** (*Change*). Logo, sempre que a seta estiver chegando a uma ação e ela não estiver estereotipada, diz-se que é um **uso**. Se a seta estiver partindo de uma ação e não estiver estereotipada, então trata-se de uma **alteração**. Vale ressaltar, ainda, que uma alteração só faz sentido juntamente com um uso do objeto.

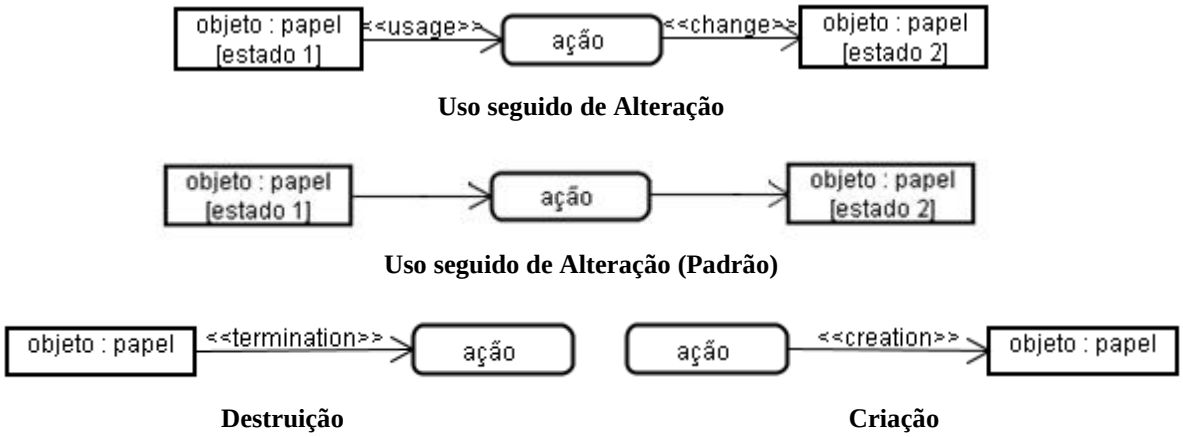


Figura 4.22: Representação das Participações de um Objeto

No exemplo da tarefa de locação, há diversas participações de recursos, como as apresentadas na Figura 4.23. Nessa figura, um item tem seu estado alterado para locado após ser emprestado e o contrato é criado para reger a locação. Ambos são usados na tarefa Devolver Item, onde terão novamente seus estados alterados.

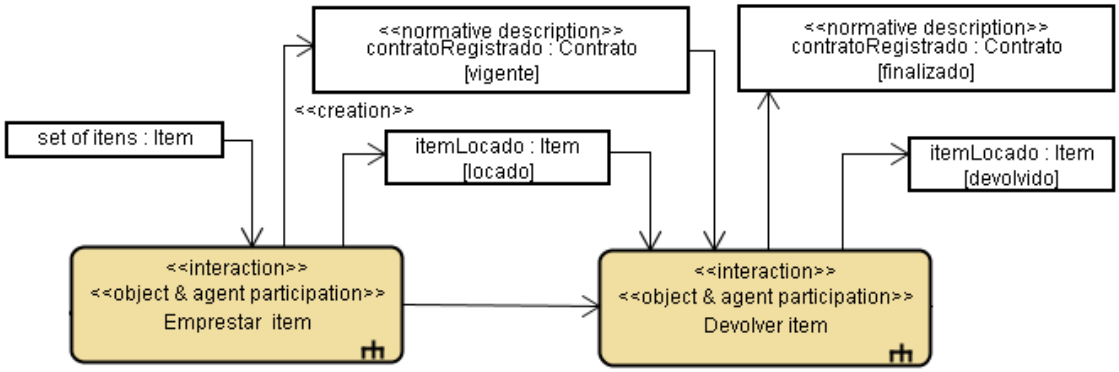


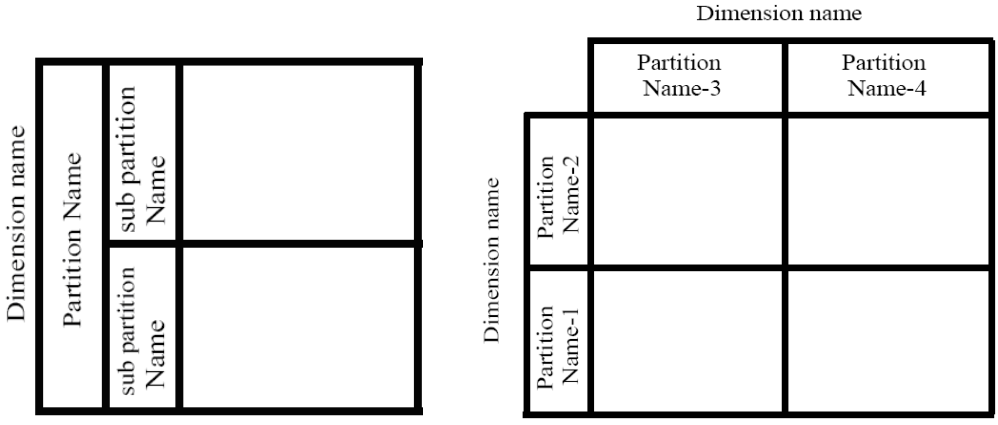
Figura 4.23: Exemplo da representação proposta para participações de recursos.

4.4.3.5. Agentes: Participações e Papéis.

Um elemento fundamental para a execução de uma tarefa é o agente que a executa. Para a UFO, ***agentes*** (*agents*) são substanciais capazes de possuir ***modos intencionais*** e que podem realizar ***ações***. A UFO classifica agentes em ***agentes físicos*** (*Physical Agent*) e ***sociais*** (*Social Agent*), sendo que agentes sociais, por sua vez, são divididos em ***organização*** (*Organization*) e ***agente social coletivo*** (*Collective Social Agent*), que pode ainda ser especializado como uma ***sociedade*** (*Society*).

Na UML, para representar os agentes que participam de uma ação, pode-se utilizar partições, definidas no metamodelo como **partição de atividade** (*ActivityPartition)*. Uma **partição de atividade** é agrupamento genérico de nós e arestas usado para identificar ações que têm alguma característica em comum (OMG, 2007). Essas ações são colocadas dentro de um retângulo, como mostra a Figura 4.24, com um título que ressalta essa característica, que, na maioria das vezes, trata dos atores envolvidos na ação.

Representação de Partições com uma dimensão



Representação de Partições com mais de uma dimensão

Figura 4.24: Formas possíveis de representação de partições na UML

Partições restringem e proveem uma visão sobre os comportamentos invocados por uma atividade de acordo com o tipo de elemento que a partição representa. Essa relação é expressa no metamodelo pela associação "**representa"** (*represents*) e tem multiplicidade 0..1 com **elemento** (*Element*) (Figura 4.25). Porém, **elemento** representa qualquer elemento que faça parte de um modelo UML e possui diversas especializações (OMG, 2007).

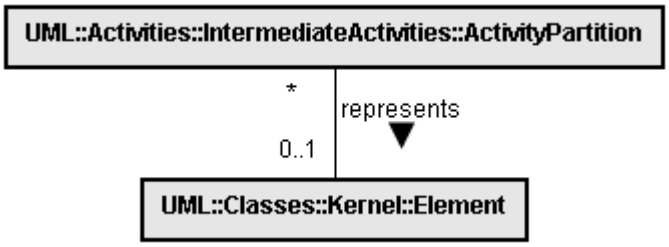


Figura 4.25: Relação com *Element*

No metamodelo é definido que ele pode representar um **classificador** (*Classifier*) ou uma instância. Quando representa um **classificador**, isso significa que os comportamentos invocados na partição são de responsabilidade de instâncias do classificador representado. Já quando representam uma instância, ele impõe as mesmas restrições que um **classificador**, mas restritas a uma particular instância do **classificador** (OMG, 2007). Este último é o caso necessário para este trabalho, em que se quer representar o comportamento de instâncias exemplo que exercem determinado papel de conhecimento na apresentação do fluxo de subtarefas.

Como se pode notar, a definição de **partição** é mais genérica do que é necessário para o objetivo de representar agentes em uma ontologia de tarefa. Logo é necessário especializar esse elemento de modelo, de modo a restringi-lo à representação de agentes e focando na representação de instâncias.

O relacionamento ***representa*** entre ***partições*** e ***elementos*** no meta-modelo da UML possui cardinalidade 0..1, ou seja, só é possível representar um agente em cada **partição**. Há a possibilidade de colocar partições uma dentro da outra para agrupar agentes ou ainda se utilizar várias dimensões, como ilustrado na Figura 4.24. Porém há casos em que uma tarefa pode ser realizada por vários agentes ou por um dentre vários agentes e nem sempre é possível representar isso usando esses recursos. Precisa-se, portanto, de uma notação que permita representar uma interação entre diversos atores. Para isso estendeu-se a classe **partição de atividade** (*ActivityPartition*), criando o elemento **partição de atividade de agente** (*AgentActivityPartition*), associado (associação **representa** (*represents*)) aos ***agentes*** (*Agent*) que participam de uma determinada ação, como mostra a Figura 4.26. Essa associação impõe que haja pelo menos um agente e, por outro lado, permite que mais de um agente participe de uma ação.

As distinções da UFO relativas a tipos de agentes podem ser relevantes para se fazer a classificação dos agentes nos diagramas. Assim, no perfil E-OntoUML, consideraram-se os tipos de agentes definidos na UFO, a saber: ***agente físico*** (*Physical Agent*), ***agente social***

(*Social Agent*), ***organização*** (*Organization*), ***agente social coletivo*** (*Collective Social Agent*) e ***sociedade*** (*Society*). Portanto, considerou-se essa hierarquia ao definir as partições do perfil e foram criadas partições estereotipadas para cada tipo de agente citado na UFO, como definido na Figura 4.26. Vale ressaltar que, se algum tipo de agente for adicionado na UFO, será necessário incluí-lo também ao metamodelo desse perfil.

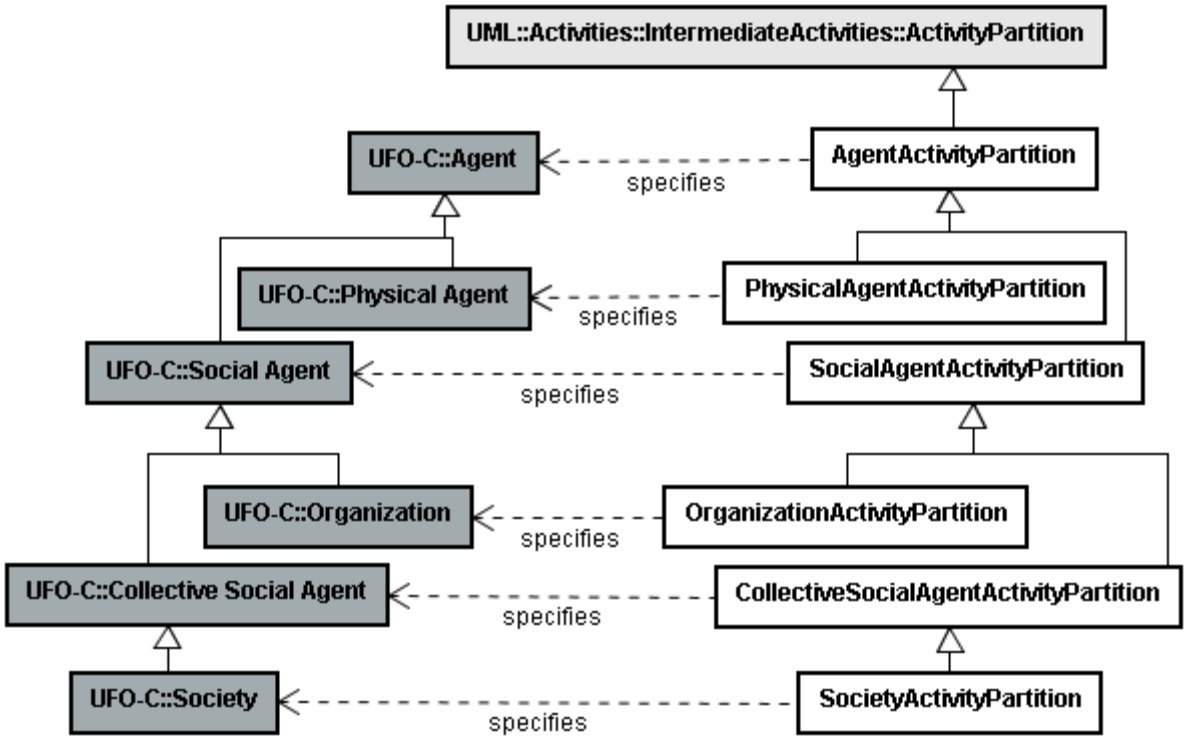


Figura 4.26: Elementos para representação de Agentes.

Uma **partição de atividade de agente** (*AgentActivityPartition)* representa ***agentes*** (*Agent),* ou seja, indivíduos capazes de agir, perceber e ter estados mentais (GUIZZARDI et al., 2008b). Considerou-se esse tipo de partição como *default* e, portanto, torna-se optativo o uso de estereótipos em sua representação (Figura 4.27).

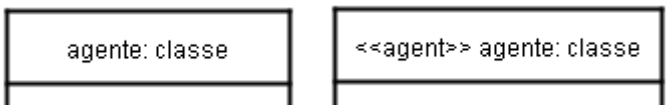


Figura 4.27: Representação proposta para Agentes.

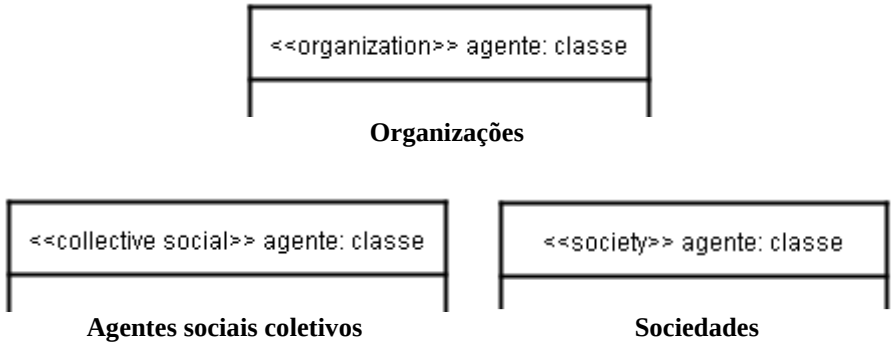
Buscando refletir a hierarquia de tipos de agentes proposta na UFO, foram criadas a **partição de atividades de agente físico** (*PhysicalAgentActivityPartition*) e a **partição de atividades de agente social** (*SocialAgentActivityPartition),* cujas representações são mostradas na Figura 4.28. A primeira representa um ***agente físico*** (*Physical Agent)* (uma pessoa, por exemplo) capaz de realizar ações, perceber eventos e que pode ter um estado

mental. A segunda representa um ***agente social*** (*Social Agent*), que se refere a agentes que necessitam de uma ***descrição normativa*** (*Normative Description)* para existir ou para se tornarem legais (ALMEIDA et al., 2009).

Representação para agentes físicos **Representação para agentes sociais**

Figura 4.28: Representação dos tipos de agentes

Para representar os tipos de ***agentes sociais*** (*Social Agents)*, foram criadas, ainda, a **partição de atividade de organização** (*OrganizationActivityPartition*) para representar uma ***organização*** (*Organization*) e **partição de atividade de agente social coletivo** (*CollectiveSocialAgentActivityPartition)* para representar um ***agente social coletivo*** (*Collective Social Agent*). Finalmente, uma **partição de atividade de sociedade** (*SocietyActivityPartition*) representa uma ***sociedade*** (*Society*), como mostra a Figura 4.29.



Organizações

Agentes sociais coletivos **Sociedades**

Figura 4.29: Hierarquia para representação de *Social Agents*.

Note que na representação proposta há um elemento não utilizado normalmente em diagramas de atividades convencionais, a "classe". A UML não faz uma ligação entre o nome da partição e outros elementos de modelo. O nome da partição é apenas uma descrição textual de algo que é comum às atividades da partição. Para o propósito de representação de ontologias de tarefa, contudo, é importante mostrar a relação entre a instância que está sendo representada e a classe a que ela pertence, que tem de ser um dos conceitos modelados no diagrama de papéis de conhecimento. Por isso acrescentou-se essa notação, seguindo o padrão utilizado para representar classes dos objetos.

Voltando ao exemplo da Tarefa de Locação, há dois agentes principais: o Locador e o Locatário. Ao se modelar a tarefa de locação, foram criadas partições para esses dois papéis

existentes na tarefa, deixando explícito o que cada papel executa. A Figura 4.30 apresenta a representação de duas instâncias chamadas `locador` e `locatario` que são relativas aos papéis Locador e Locatário, respectivamente. Vale lembrar que neste caso não é necessário o uso de estereótipos, pois não há necessidade de se comprometer com nenhum tipo de agente específico. Tratam-se apenas de agentes.

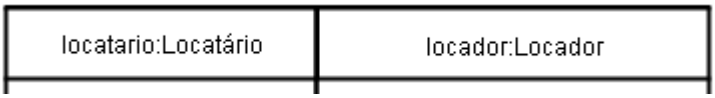


Figura 4.30: Agentes da tarefa de Locação

4.4.3.6. Fluxo de Controle: Ordem e Sincronização

A UFO, seguindo as definições de (ALLEN, 1993), determina sete relações temporais entre dois eventos, como mostra a Figura 3.4 do Capítulo 3. São elas: ***precede*** (*before*), ***encontra*** (*meets*), ***sobrepõe*** (*overlaps*), ***inicia*** (*starts*), ***durante*** (*during*), ***termina*** (*finishes*) e ***equivale*** (*equals*). As duas primeiras tratam de eventos sequenciais e as demais de eventos paralelos.

O fluxo de controle em diagramas de atividade da UML é representado por arestas (*edges*) denominadas **fluxos de controle** (*ControlFlows*) e **nós de controle** (*ControlNodes*). Um **fluxo de controle** (*ControlFlow*) é uma aresta que indica o início de um **nó de atividade** (*ActivityNode*) após a finalização do anterior, como mostra a Figura 4.31. Ele define a ordem de execução das ações que conecta, mas não trata a relação temporal entre elas.

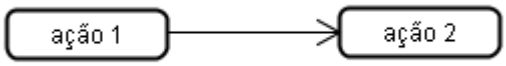


Figura 4.31: Representação do Fluxo de Controle (*Control Flow*).

Já um **nó de controle** (*ControlNode*) é um tipo de **nó de atividade** (*ActivityNode*) que coordena o fluxo em uma atividade. É um nó de atividade usado para coordenar o fluxo de controle entre outros nós (OMG, 2007), sendo especializado em seis tipos de nós de controle: **nó de decisão** (*DecisionNode*), **nó de fusão** (*MergeNode*), **nó de bifurcação** (*ForkNode*), **nó de junção** (*JoinNode*), nó inicial (*InitialNode*) e **nó final** (*FinalNode*). Nó final, por sua vez, pode ser um **nó final de atividade** (*ActivityFinalNode*) ou um **nó final de fluxo** (*FlowFinalNode*). A Figura 4.32 mostra as notações gráficas para os diversos tipos de nós de controle citados acima.

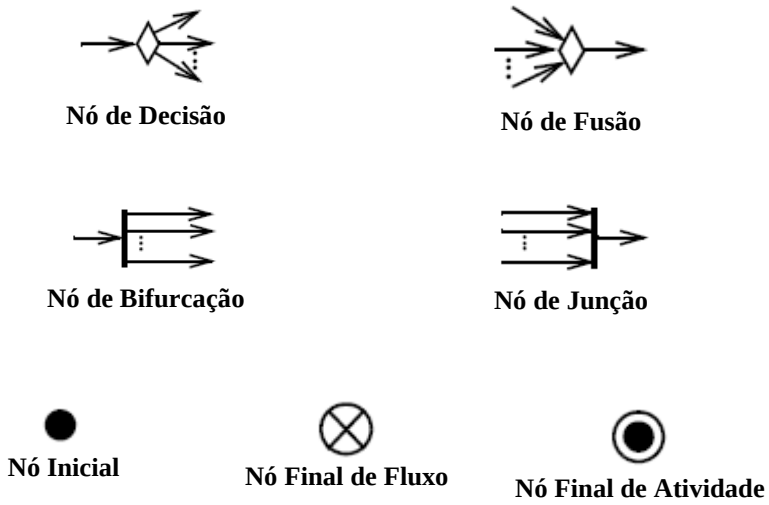


Figura 4.32: Nós de controle da UML

Nó de decisão (*DecisionNode*) é um nó de controle que seleciona um fluxo, dentre os fluxos que saem dele, a ser seguido na execução da tarefa. **Nó de fusão** (*MergeNode*) é um nó de controle que reúne múltiplos fluxos alternativos, mas não é usado para sincronizar fluxos concorrentes e sim para aceitar um dentre vários fluxos alternativos. **Nó de bifurcação** (*ForkNode*) é um nó de controle que divide um fluxo em múltiplos fluxos concorrentes. **Nó de junção** (*JoinNode*) é um nó de controle que sincroniza múltiplos fluxos. **Nó inicial** (*InitialNode*) é um nó de controle no qual o fluxo inicia quando a atividade é invocada. **Nó final de atividade** (*ActivityFinalNode)* é um nó final que finaliza todos os fluxos em uma atividade. Por fim, **nó final de fluxo** (*FlowFinalNode*) é um nó final que termina um fluxo em uma atividade. Considerando uma atividade como uma ação complexa, pode-se dizer que o último modela o fim de um subconjunto de ações de uma atividade.

Analisando os elementos de modelo da UML e as definições da UFO, percebeu-se que se pode complementar os diagramas de atividade da UML adicionando informações sobre as relações temporais entre subtarefas aos diagramas de atividades. Para tal, a classe **fluxo de controle** (*ControlFlow*) foi especializada para capturar a semântica das relações temporais nos modelos de tarefa. Assim, em E-OntoUML foram criados os elementos de modelo mostrados na Figura 4.33.

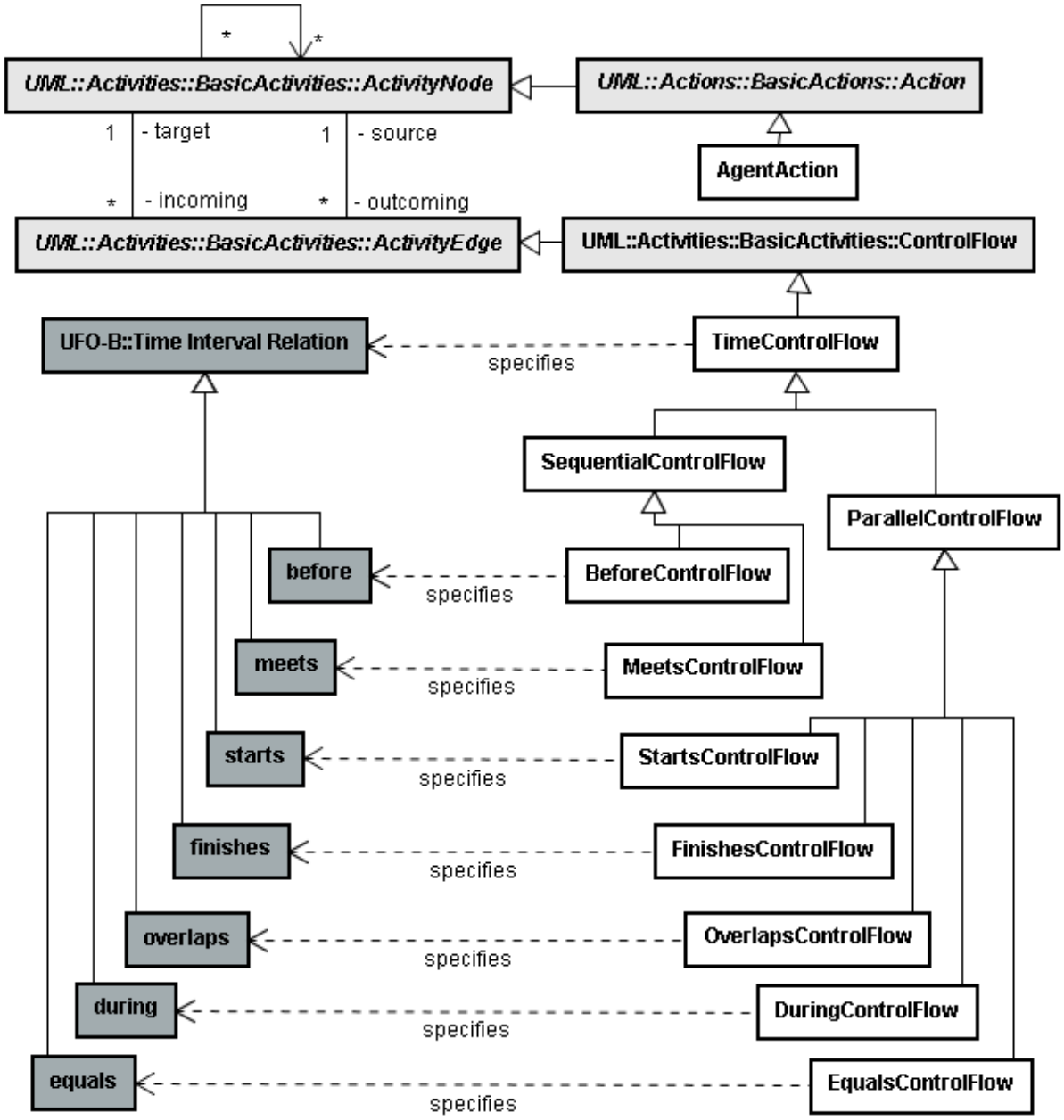


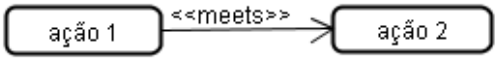
Figura 4.33: Especialização de ControlFlow para representação de relações temporais.

O elemento **fluxo de controle de tempo** (*TimeControlFlow)* trata de maneira geral uma ***relação temporal*** da UFO (*Time Interval Relation*). Esse elemento foi especializado em **fluxo de controle sequencial** (*SequencialControlFlow*) e **fluxo de controle paralelo** (*ParalellControlFlow*), que, apesar de não serem isomórficos à UFO, são necessários para melhor especificar as relações de tempo e sua relação com **nós de controle** (*ControlNode*).

Em relação ao **fluxo de controle sequencial** (*SequencialControlFlow*), foram especificadas duas subclasses: (i) **fluxo de controle de precedência** (*BeforeControlFlow)* e

(ii) **fluxo de controle de encontro** (*MeetsControlFlow)*, que definem respectivamente as relações ***precede*** (*before)* e ***encontra*** (*meets)* de ALLEN (2003). A representação desses elementos é feita pela adição de um estereótipo ao **fluxo de controle** (*ControlFlow*), indicando qual a relação temporal que está sendo considerada, como mostra a Figura 4.34. Considera-se a relação temporal ***precede*** (*before)* como *default* para sequências de subtarefas, pois é a mais comumente utilizada, sendo, portanto, facultativo o uso do estereótipo *<<before>>*.

Fluxo de controle de precedência



Fluxo de controle de encontro

Figura 4.34: Fluxo de Controle Sequencial.

Já para o **fluxo de controle paralelo** (*ParalellControlFlow*) foram consideradas as cinco demais relações temporais citadas, a saber: ***sobrepõe*** (*overlaps*), ***inicia*** (*starts*), ***durante*** *(during)*, ***termina*** *(finishes)* e ***equivale*** *(equal)*. Para cada uma delas foi criado um elemento para sua representação no perfil, a saber: **fluxo de controle de sobreposição** (*OverlapsControFlow*), **fluxo de controle de início** (*StartsControFlow*), **fluxo de controle durante** (*DuringControFlow*), **fluxo de controle de término** (*FinishesControFlow*) e **fluxo de controle de equivalência** (*EqualsControFlow*).

Assim como no caso dos tipos de fluxo de controle sequencial, a representação desses elementos é feita pela adição de um estereótipo ao **fluxo de controle** (*ControlFlow*) indicando qual a relação temporal que está sendo considerada, como mostra a Figura 4.35.

Note que as relações temporais definidas na UFO envolvem apenas dois eventos e, portanto, a mesma restrição vale para os elementos do perfil. Logo, as notações apresentadas nas Figura 4.354 e 4.35 são passíveis de uso somente para duas subtarefas. Caso seja necessário representar paralelismo de várias ações, devem-se utilizar os elementos originais dos diagramas de atividades da UML sem estereótipos que indiquem as relações temporais.

Ainda com respeito à notação apresentada na Figura 4.35, vale enfatizar que se deve inserir o estereótipo apenas em uma das setas que saem do **nó de bifurcação,** pois ela se aplica ao par de ações ocorrendo em paralelo.

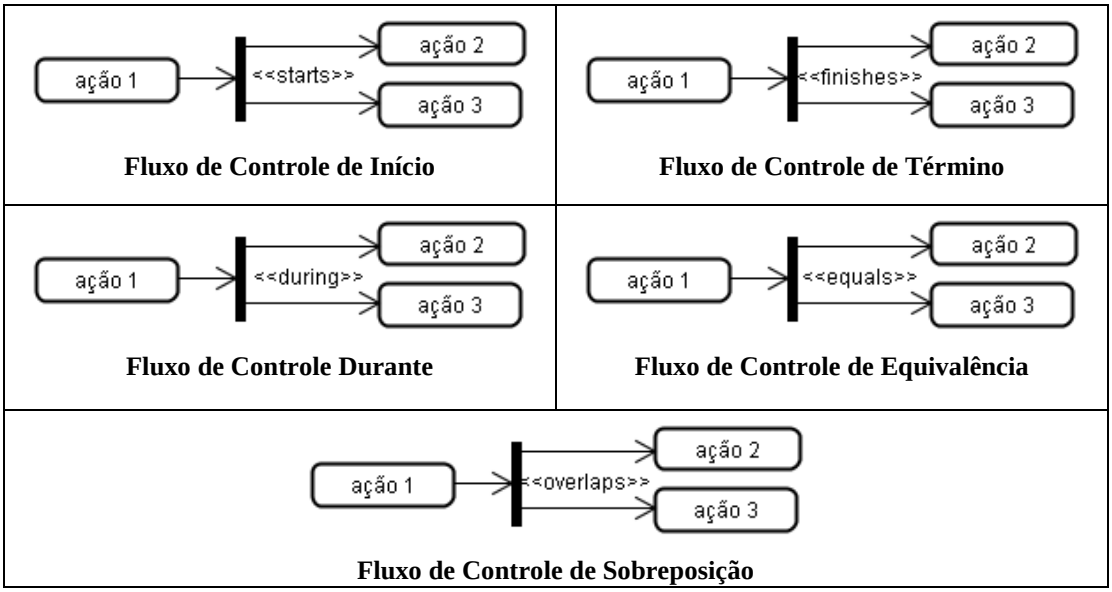


Figura 4.35: Fluxo de Controle Paralelo

Na tarefa de locação não se vê grandes exemplos da utilização desses diferentes fluxos de tarefas. As subtarefas seguem uma sequência, sem que haja a necessidade de paralelismo. Alguns poucos fluxos alternativos envolvem escolhas e não se exige que as atividades sejam realizadas imediatamente quando a outra termina, ou seja, todas as relações temporais são apenas ***precedências*** (*before*), como mostra o exemplo da Figura 4.36.

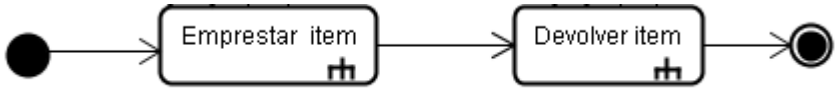


Figura 4.36: Fluxos de controle da tarefa de locação.

Para **fluxos de controle** (*ControlFlow*) que ligam duas ações diretamente podem-se utilizar os nós que especializam **fluxos de controle sequenciais** (*SequentialControlFlow)*. O mesmo se dá para nós que unem fluxos, como é o caso dos **nós de fusão** *(MergeNode)* e **de junção** (*JoinNode)*. Já os **nós de bifurcação** (*ForkNode*) e **de decisão** (*DecicionNode*) são considerados **fluxos de controle paralelo** (*ParallelControlFlow*s) e, portanto, podem definir a relação de paralelismo entre os diversos fluxos que partem desses nós. Os modelos das Figura 4.37 e Figura 4.38 mostram como esses elementos são tratados em E-OntoUML.

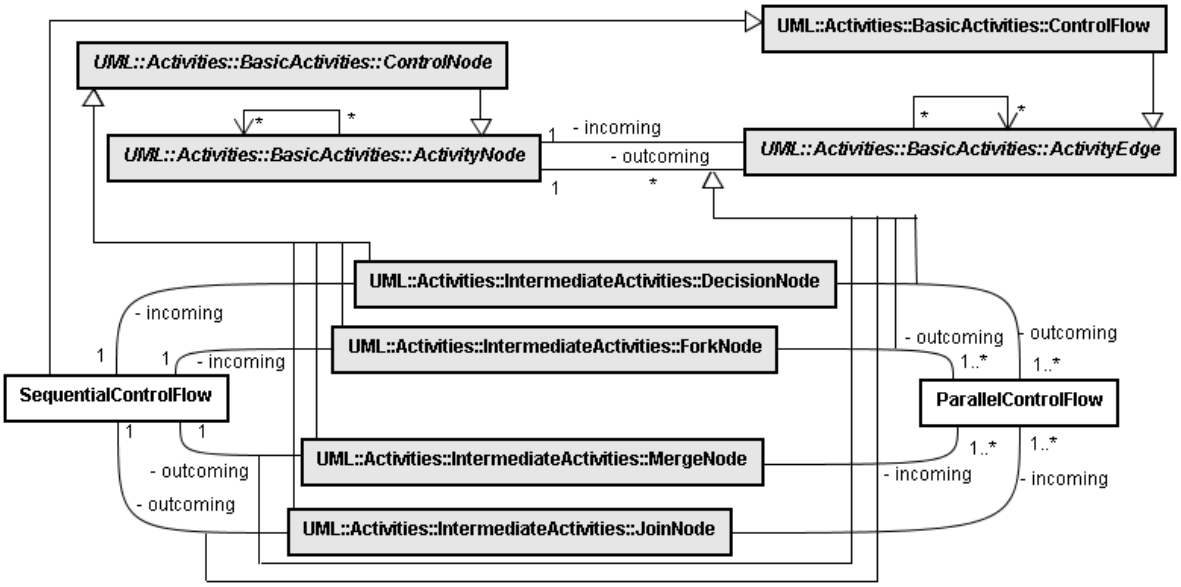


Figura 4.37: Especialização do relacionamento o*utcoming* de acordo com os novos conceitos adicionados.

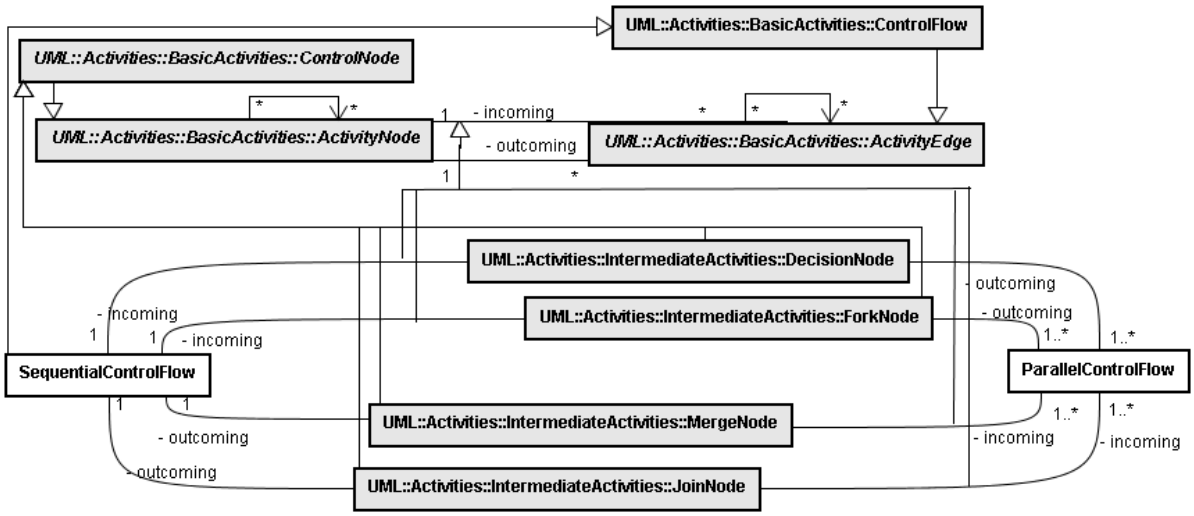


Figura 4.38: Especialização do relacionamento *incoming* de acordo com os novos conceitos adicionados.

Existem, ainda, três nós especiais, a saber, **nó inicial** (*InitialNode*), **nó final de atividade** (*ActivityFinalNode*) e **nó final de fluxo** (*FlowFinalNode*). Esses nós foram considerados meramente elementos de especificação que não possuem um significado atrelado a elementos da UFO que agregue nova semântica relevante ao modelo. Contudo, são importantes elementos de especificação, uma vez que definem, respectivamente, os pontos de início e término de um fluxo de atividades.

4.4.4. Estudo de caso: Decomposição e Fluxo de Controle da Tarefa de Locação

As Figura 4.39, Figura 4.40 e Figura 4.41 mostram a perspectiva comportamental da ontologia da tarefa de locação completa, mostrando seu fluxo de controle, entradas, saídas e

agentes que executam cada uma das subtarefas. A Figura 4.39 apresenta o diagrama de atividades completo da tarefa de locação, no qual duas principais subtarefas são identificadas: “Emprestar item” e “Devolver item”. Estas são refinadas em outros diagramas de atividades, mostrados nas Figura 4.40 e Figura 4.41, respectivamente. Elas são tarefas que envolvem a participação de diversos objetos e de dois agentes e por isso foram consideradas **interações** (estereótipo <<interaction>>) e **ações de agente com objeto** (estereótipo <<object & agent participation>>). As partições mostram os agentes que participam das ações. Note que não se usou estereótipo, pois se trata do tipo definido como default, <<agent>>, que é facultativo. Na partição são apresentadas as instâncias cujas ações estão sendo representadas e os papéis que elas exercem na realização da tarefa. Neste caso as instâncias são `locador` e `locatario`, que desempenham os papéis de `Locador` e `Locatário`, respectivamente.

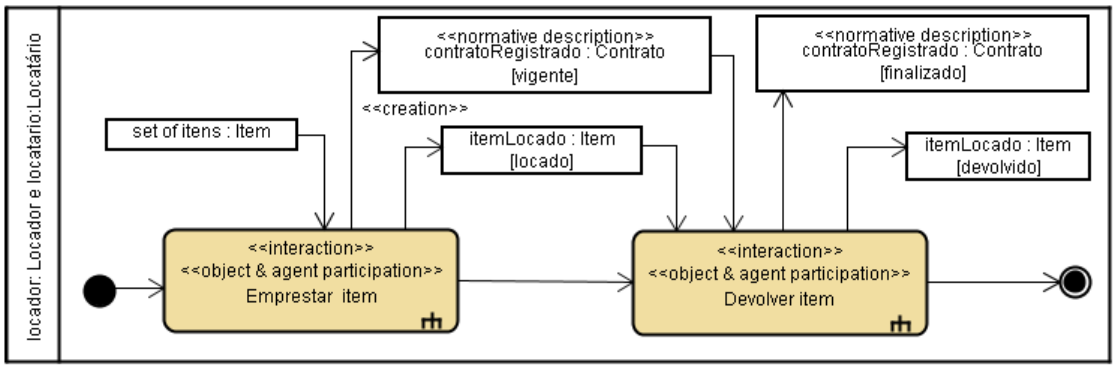


Figura 4.39: Decomposição de Tarefa de Locação

O fluxo de dados em um diagrama de atividades é mostrado por meio dos objetos. As correspondentes instâncias dos papéis de conhecimento identificados na perspectiva estrutural são conectados às tarefas em que participam e sua classificação também é definida seguindo as distinções da UFO. Itens podem ser objetos quaisquer e, por isso, nenhum estereótipo foi usado. Já contratos são descrições normativas e, portanto, foram diferenciados com o estereótipo <<normative description>>. Ainda foram definidas as participações desses objetos, sendo estereotipada apenas aquela que se refere à criação (<<creation>>) de um objeto. As demais, por se tratarem de participações que foram definidas como *default* (a saber, uso (*Usage*), quando a seta parte de um objeto e chega a uma ação, e alteração (*Change*), quando a seta parte de uma ação e chega a um objeto) são mostradas sem estereótipos.

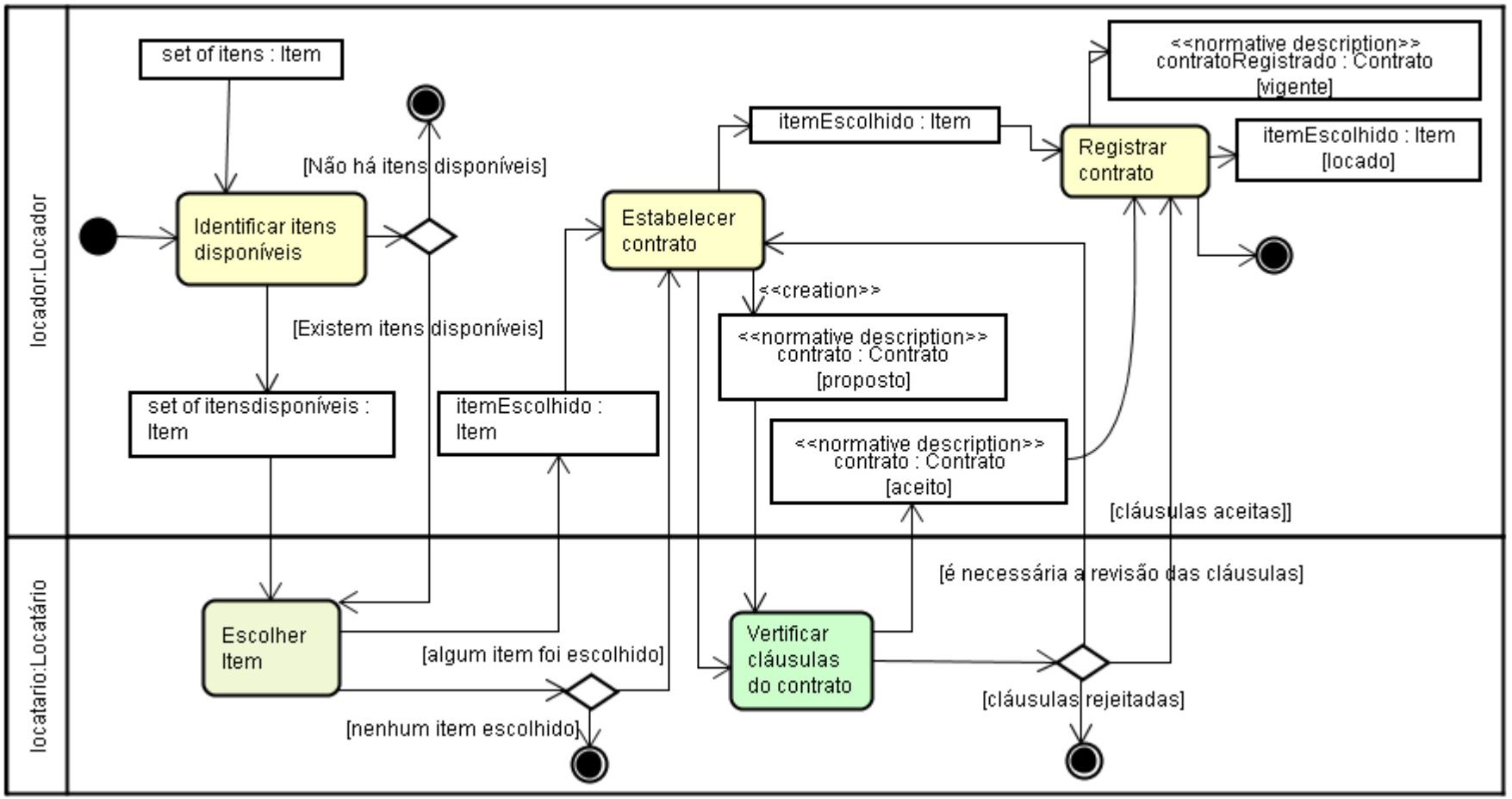


Figura 4.40: Decomposição da subtarefa “Emprestar Item”.

Como mostra a Figura 4.40, a subtarefa de empréstimo inicia com a identificação dos itens disponíveis por parte do locador. Se existirem itens disponíveis o locatário escolhe um item a ser locado. Escolhido o item, o próximo passo é estabelecer um contrato que irá reger a locação. O locatário deve verificar as cláusulas que são propostas no contrato. Se elas forem aceitas pelo locatário, o locador registra o contrato, que se torna vigente, e o item escolhido é considerado locado e, portanto, indisponível para outras locações.

Note que alguns elementos de modelagem não têm sua notação diferenciada, tais como as notações de condições de guarda e estados dos objetos, sendo as mesmas propostas pela UML. Mas vale lembrar que o perfil atribui a semântica da UFO a esses elementos, apesar de não alterar sua representação. Logo, quando são utilizadas as notações de estado e condições de guarda, está-se referindo, respectivamente, a fases (*Phases*) dos objetos e proposições (*Proposition*) que serão satisfeitas por pós-estados de ações.

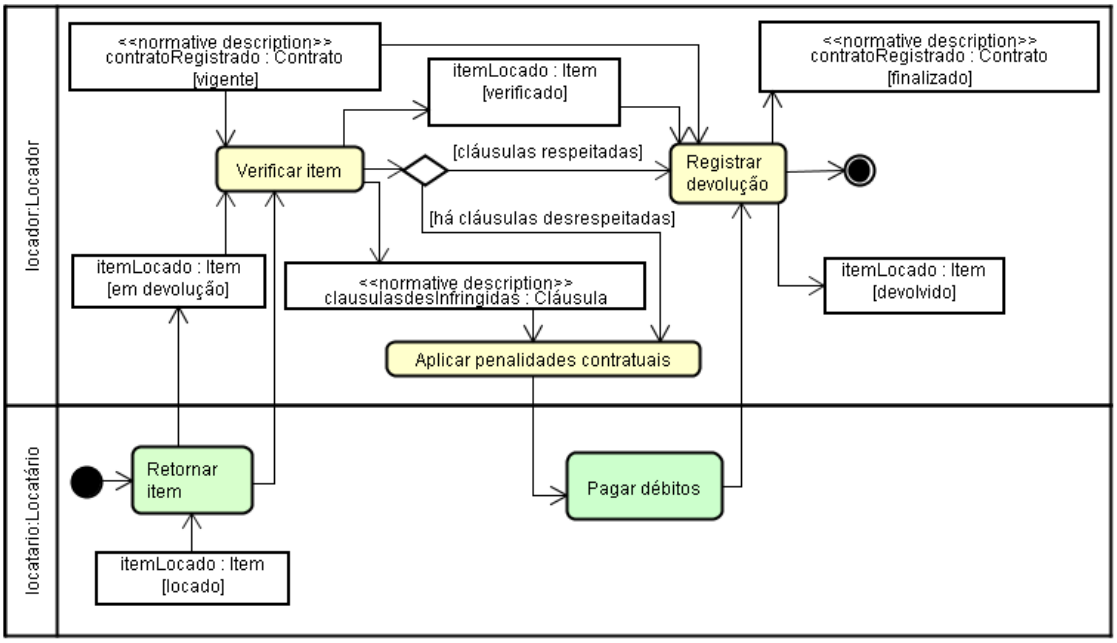


Figura 4.41: Decomposição da subtarefa “Devolver Item”.

No caso da tarefa “Devolver item” (Figura 4.41), a subtarefa inicia com o locatário entregando o item locado. O locador deve, então, verificá-lo de acordo com as cláusulas do contrato. Se alguma cláusula do contrato é infringida, a penalidade correspondente é aplicada. O locador deve pagar os débitos e depois disso a devolução é registrada. Enfim, o contrato é finalizado e o item é devolvido.

4.5. AVALIAÇÃO PRELIMINAR DO PERFIL PROPOSTO

Durante o desenvolvimento do perfil E-OntoUML, procurou-se considerar algumas propriedades que devem ser reforçadas para um mapeamento entre uma ontologia de fundamentação e o metamodelo de uma linguagem, a saber (GUIZZARDI, 2005): (i) Solidez (*Soundness*), quando cada primitiva da linguagem representa pelo menos um conceito da ontologia, (ii) Completude (*Completeness*), quando cada conceito presente na ontologia é representado por pelo menos uma primitiva da linguagem, (iii) Lucidez (*Lucidity*), quando cada primitiva da linguagem representa no máximo um conceito da ontologia, e (iv) Concisão ou Brevidade (*Laconicity*), todo conceito presente na ontologia é representado por no máximo uma primitiva da linguagem.

Considerando-se os diagramas de atividade da UML, ainda sem as extensões propostas pelo perfil E-OntoUML, e os analisando à luz das partes B e C da UFO, pode-se concluir que a UML:

- É sólida, pois todos os elementos adicionados ao perfil possuem correspondentes na ontologia. Alguns elementos originais dos diagramas de atividades que não apresentam correspondentes na ontologia são elementos de especificação que servem apenas para facilitar o entendimento do modelo gráfico gerado. Por exemplo, nó de decisão (*DecisionNode*), nó de fusão (*MergeNode*) e nó de bifurcação (*ForkNode*) são elementos de especificação da UML que não necessitam de um correspondente na UFO.
- Não é completa, uma vez que não é capaz de representar todos os conceitos da ontologia. Isso se dá porque esse não é um objetivo real da linguagem e também porque a ontologia se propõe a representar mais do que conceitos de definições de atividades e processos. Por exemplo, diagramas de atividades não permitem representar os tipos de participação de objeto e diferenciar tipos de objetos e agentes, assim como não possuem elementos para modelar intenções e compromissos, definidos pela UFO-C.
- Não é lúcida, pois existem elementos da linguagem que representam mais do que um conceito da ontologia. Isso ocorre principalmente com elementos que são especializados na ontologia e não têm sua hierarquia refletida na linguagem, tais como especializações de agentes, objetos e ações. A linguagem usa o mesmo símbolo para representar a superclasse e não diferencia suas especializações.

- Não é breve (ou concisa), pois apresenta elementos distintos da linguagem que representam um mesmo elemento da ontologia, tal como a notação de atividades, que representa uma ação complexa da ontologia, como mostra a Figura 4.42.

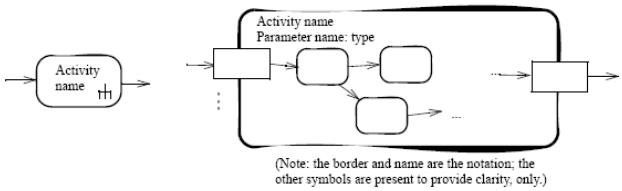


Figura 4.42: Elemento que possui dupla representação.

Considerando o perfil E-OntoUML e a UFO, ainda que não se atinja o isomorfismo, alguns aspectos são melhorados. Adiciona-se lucidez, por especializar os elementos que apresentam hierarquias, e concisão, por considerar apenas o uso de alguns elementos na linguagem que são especialmente necessários para a representação dos conceitos relacionados a tarefas.

A respeito da completude, pode-se dizer que o perfil é mais completo do que os diagramas de atividades da UML sem extensões, porém não é, de fato, completo em relação à UFO. Ele apresenta meios de representar, por exemplo, tipos de participação de objeto e permite diferenciar tipos de objetos e agentes, o que não é feito nos diagramas de atividades da UML. Contudo, a UFO, principalmente a UFO-C, trata de aspectos que vão além das necessidades de representação de tarefa, logo não é objetivo deste trabalho fazer um perfil completo nesse sentido.

Quando o assunto é solidez, a situação é similar aos diagramas de atividades da UML sem extensões, pois o perfil não estabelece, por exemplo, conceitos da UFO que representem nós de decisão (*DecisionNode*), de fusão (*MergeNode*), de junção (*JoinNode*) e de bifurcação (*ForkNode*). Isso ocorre, pois não há na UFO correspondentes semânticos para eles e os mesmos se mostraram importantes para representar modelos de tarefas.

4.6. INTEGRAÇÃO DE ONTOLOGIAS DE TAREFA COM ONTOLOGIAS DE DOMÍNIO E A DERIVAÇÃO DE ONTOLOGIAS DE CLASSES DE APLICAÇÃO

Pode-se ver que muitos domínios podem ser mapeados na tarefa de locação apresentada anteriormente, como livros, automóveis, DVDs e imóveis, dentre outros. Isto acontece porque a ontologia de tarefa foi desenvolvida de forma independente de domínio. As descrições são feitas usando termos genéricos que descrevem os papéis dos objetos e agentes que executam a tarefa. A separação do conhecimento de tarefa faz com que a integração da ontologia de tarefa de locação com ontologias de domínio seja facilitada. Adicionalmente, o uso da mesma notação para ontologias de domínio e o modelo de papéis de conhecimento de tarefa visa facilitar a melhor integração desses dois modelos.

A integração consiste, de fato, em identificar quais os papéis, dentre os elementos do modelo estrutural da ontologia de tarefa, que os elementos do domínio exercem na execução da tarefa. Em outras palavras, é suficiente definir a função que entidades do domínio irão exercer na execução da tarefa, posto que papéis já são mapeados no fluxo da tarefa.

Seja a ontologia do domínio de livros mostrada na Figura 4.43. Ela apresenta os conceitos que são senso comum a respeito de livros, como exemplar, editora, autor, país e obra. Ela apresenta os relacionamentos e multiplicidades que se comprometam com a realidade desse domínio, provendo a conceituação para o mesmo.

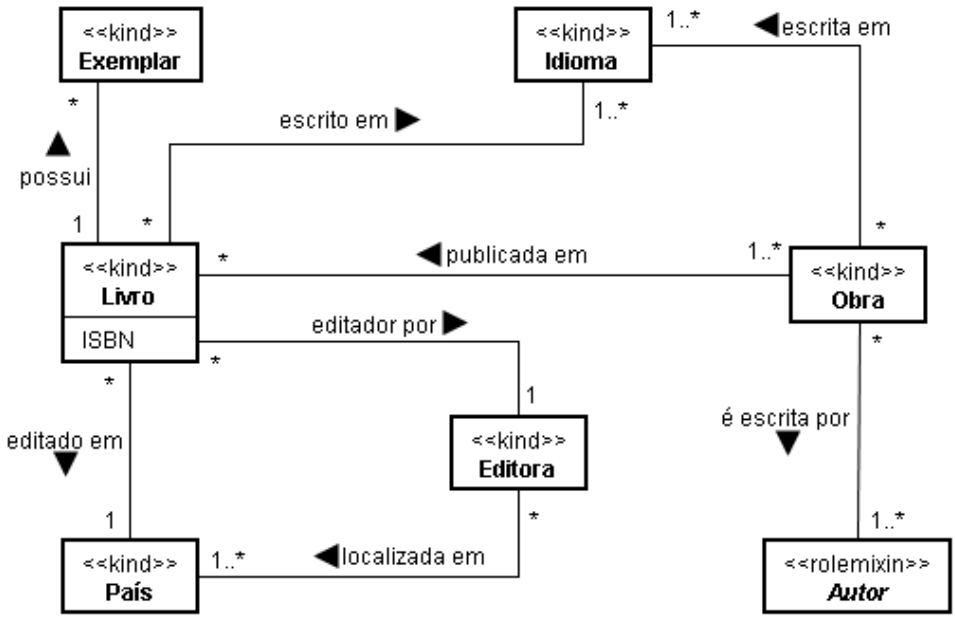


Figura 4.43: Ontologia do Domínio de Livros

A Figura 4.44 ilustra a integração da ontologia de tarefa de locação com a ontologia para o domínio de livros. Neste exemplo conceitos da ontologia do domínio de livros foram mapeados aos papéis de conhecimento da tarefa de locação. Dessa forma os conceitos do domínio de livros se integram não só à parte estrutural, mas também à parte comportamental, pois esta é fortemente relacionada por referência aos papéis de conhecimento.

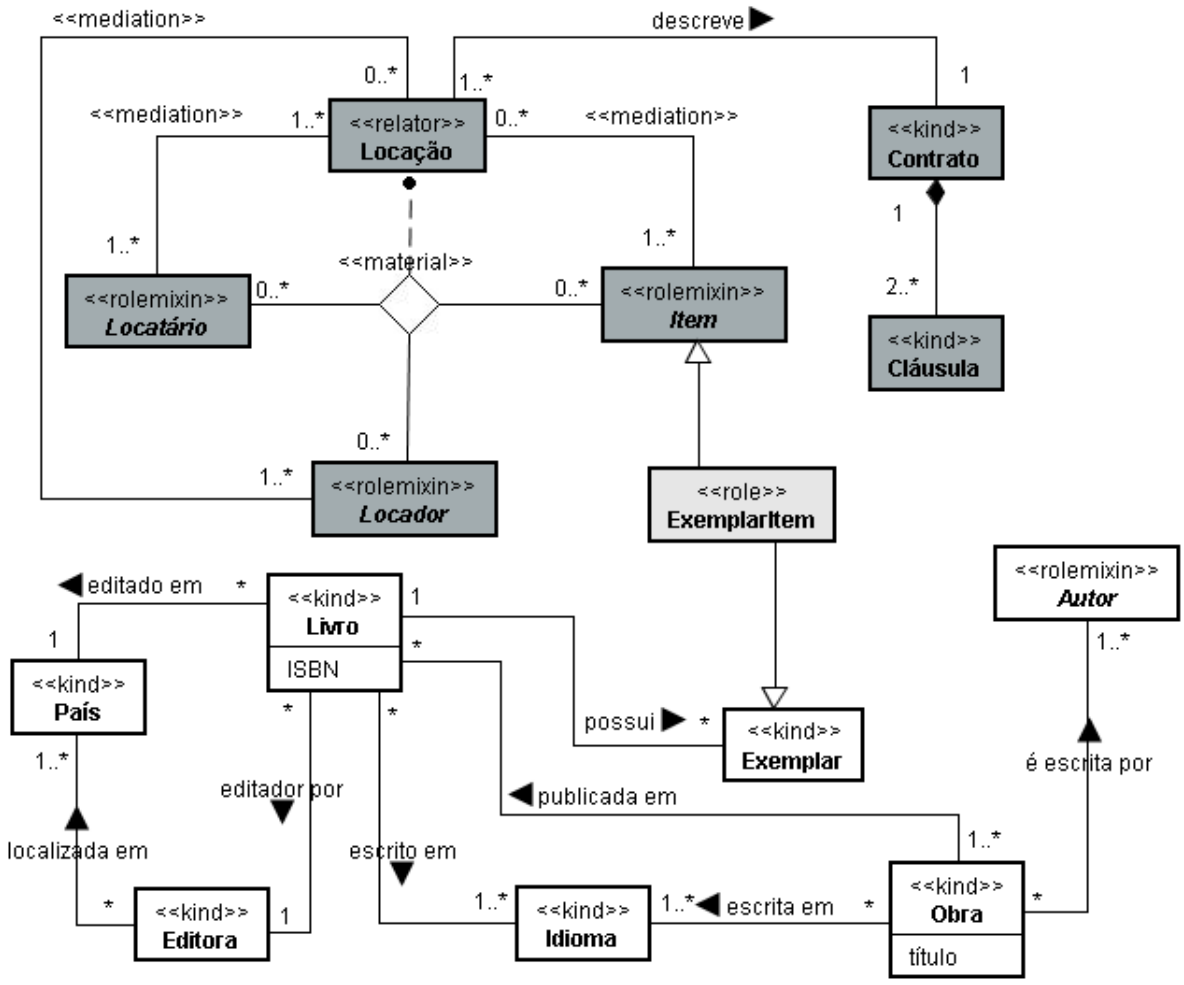


Figura 4.44: Integração da Ontologia de Tarefa de Locação com uma Ontologia de Livros

Um ponto importante a ser considerado no momento da integração são as restrições da linguagem de representação escolhida. Tem-se que ficar atento aos postulados impostos pela OntoUML quando se estiver fazendo o mapeamento entre papéis de conhecimento com ontologias de domínio. Resumidamente, esses postulados dizem que (GUIZZARDI, 2005):

- Um sortal de substância (*substance sortal)* – espécie *(kind)*, quantidade *(quantity)* e coletivo *(collective)* – não pode ter como um supertipo uma fase (*phase*), papel (*role*) ou papel misto (*roleMixin*).

- Um sortal de substância (*substance sortal*) (espécie (*kind*), quantidade (*quantity*) e coletivo (*collective*)) não pode ter como supertipo uma espécie (*kind*), subespécie (*subkind*), quantidade (*quantity*) ou coletivo (*collective*).
- Um subespécie (*subkind)* não pode ter como supertipo uma fase (*phase*), papel (*role*) ou papel misto (*roleMixin*).
- Uma fase (*phase*) não pode aparecer em um modelo como um supertipo de um universal rígido (*rigid universal)* – espécie (*kind*), subespécie (*subkind*), quantidade (*quantity*), coletivo (*collective*) ou categoria (*category*);
- Um papel (*role*) não pode aparecer em um modelo como um supertipo de um universal rígido (*rigid universal)* – espécie (*kind*), subespécie (*subkind*), quantidade (*quantity*), coletivo (*collective*) ou categoria (*category*);
- Uma mistura (*mixin*) não pode ter como supertipo de uma espécie (*kind*), quantidade (*quantity*), coletivo (*collective*), subespécie (*subkind*), fase (*phase*) ou papel (*role*).
- Uma categoria (*category)* não pode ter um papel misto (*rolemixin)* como um supertipo.
- Uma mistura (*mixin*) não pode ter um papel misto (*rolemixin)* como supertipo.

Além desses postulados, também é fundamental considerar o padrão ontológico para modelagem de papéis apresentado em (GUIZZARDI, 2005). Ele é importante, pois define regras para integração de ***espécies*** *(kids)* e ***papéis mistos*** (*roleMixin*) e ao modelar uma ontologia de tarefa busca-se representar papéis de conhecimento genéricos que em sua maioria serão ***papéis mistos*** a serem associados a ***espécies*** do domínio.

Note na Figura 4.44 que um **exemplar** exercerá a função de **item** durante uma locação, porém não é verdade que todo exemplar de livro no mundo é um item de locação e nem que todo item de locação é um exemplar de livro. Logo a especialização não pode ser direta em nenhuma das direções (Figura 4.45).

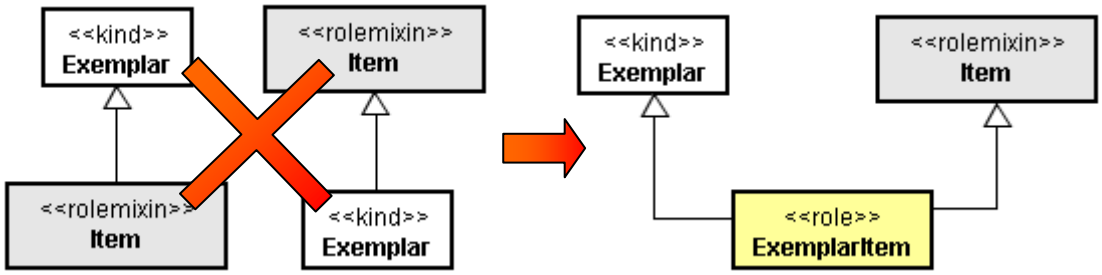


Figura 4.45: Padrão de projeto ontológico para modelagem de papéis

O padrão define que se deve criar um conceito que seja um ***papel*** (*role*) e este vai agregar os critérios de identidade da ***espécie*** (*kind*) e exercer o ***papel misto*** (*roleMixin*) definido. No exemplo, foi criado o conceito **ExemplarItem** que especializa tanto **Exemplar**

como **Item**. Dessa forma fica definido que existem Exemplares que são itens de locação e outros que não são, assim como nem todo item de locação é definido como um exemplar de livro, podendo ser outra coisa como um exemplar de DVD ou um carro.

Apesar dessas restrições, que são necessárias para manter o comprometimento ontológico com a UFO, a integração é feita de forma simples. Note que essa facilidade de integração é proveniente de dois pontos principais: (i) a forma como o conhecimento da tarefa é organizado em visões estrutural e comportamental e (ii) a utilização da mesma notação de ontologias de domínio para a representação dos papéis de conhecimento na perspectiva estrutural da ontologia de tarefa e a sua forte ligação com o modelo de fluxo de controle da perspectiva comportamental da ontologia de tarefa.

No que se refere ao modelo comportamental, deve-se observar a necessidade de considerar a especialização de agentes e objetos. Quando se define uma tarefa genérica, opta-se pelo menor nível de especialização para não restringir sua interpretação. Porém, quando se trata de uma ontologia de domínio, seus conceitos tendem a ser mais específicos. Por exemplo, um item de locação pode ser qualquer tipo de ***objeto***, porém quando se fala em exemplar de um livro está-se definindo um ***objeto físico***. Logo adaptações devem ser feitas no modelo comportamental, como mostra a Figura 4.46.

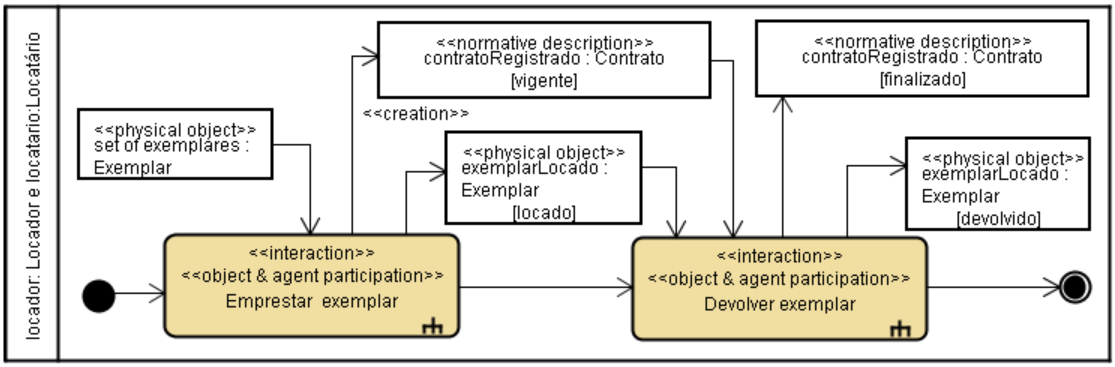


Figura 4.46: Modelo comportamental da ontologia de classe de aplicação de locação de livros.

Note também que alguns conceitos da ontologia de domínio podem não ser mapeados em papéis da ontologia de tarefa, assim como o contrário, ou seja, papéis da ontologia de tarefa podem não ter um correspondente no domínio especificado. Seja o exemplo da integração da ontologia de livros e com a ontologia de tarefa de locação apresentada. A ontologia de livros, mostrada na Figura 4.44 com os conceitos em branco, possui os seguintes conceitos: livro, exemplar, obra, autor, editora, país e idioma. Comparando com os papéis definidos no modelo estrutural da locação, nota-se que diversos conceitos ficam sem mapeamento. Podem-se adicionar novos conceitos para capturar as informações não

mapeadas ou, ainda, podem-se integrar outras ontologias de domínio, como uma ontologia de domínio de organização, na qual há conceitos como pessoa, documento e serviço, e ainda uma ontologia de domínio de biblioteca, que fornece conceitos como biblioteca, acervo, publicação, usuário, bibliotecário etc.

Considerando as várias definições de ontologias de aplicação apresentadas no Capítulo 3, pode-se perceber que elas são vistas como uma descrição de conceitos e relações (tanto de domínio quanto de tarefa) adotados em uma aplicação específica. Entretanto, há de se considerar que ontologias de domínio e de tarefa podem ser combinadas não apenas para descrever o conhecimento de uma aplicação, mas também de uma classe de aplicações. Assim, em (MARTINS, FALBO, 2008) se definiu o termo "ontologias de classes de aplicação" como um tipo de ontologia gerada a partir da integração de ontologias de tarefa e ontologias de domínio, obtendo, portanto, um vocabulário que descreve conceitos relativos a um conjunto de aplicações que atuam realizando uma determinada tarefa em um respectivo domínio. Como exemplo, temos a ontologia gerada pela integração da ontologia de tarefa de locação com a ontologia de livros. Ela é uma ontologia que identifica os conceitos essenciais para aplicações de locação de livros, porém ela não contém todos os conceitos típicos de aplicações de locação de livros. Assim, além dos conceitos, outros poderiam ser adicionados para se obter uma ontologia de locações de livros. Uma ontologia de aplicação, por outro lado, deveria englobar outros conceitos e restrições que são específicos para de uma aplicação de locação de livros, tal como o sistema de empréstimo da Biblioteca da UFES.

Portanto, diferentemente da definição dada para ontologias de aplicação, que abrangem uma única aplicação em particular, ontologias de classes de aplicações definem termos genéricos comuns a um conjunto de aplicações que realizam certas tarefas em um domínio particular. O conhecimento capturado por uma ontologia de classes de aplicações pode servir como base para derivar ontologias de aplicação, visto que contém conceitos mais gerais que podem ser especializados de acordo com uma aplicação específica, como ilustra a Figura 4.47.

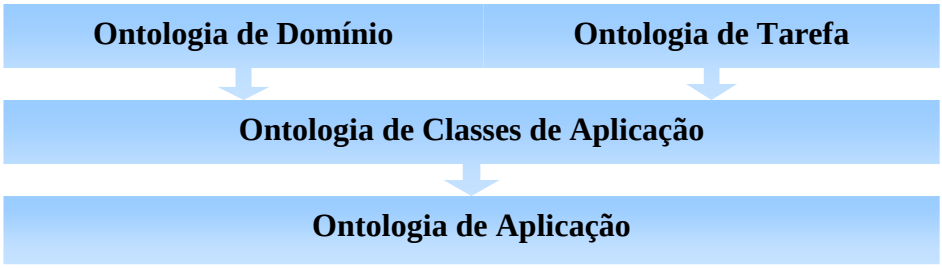


Figura 4.47: Tipologia de Ontologias utilizada neste trabalho

Logo, as ontologias geradas dessa integração são ontologias de classe de aplicação, que são diferentes das ontologias de aplicação citadas por Guarino (1998). As ontologias, aqui geradas, incluem termos, conceitos e modelos que são comuns a uma classe de aplicações e não a uma aplicação específica, como é definido o termo ‘ontologia de aplicação’ na literatura. Vale ressaltar a diferença entre uma ontologia de classes de aplicação e uma ontologia de aplicação: a última contém informações específicas de uma aplicação que não podem ser determinadas apenas pela integração de ontologias do domínio e tarefa, possuindo conceitos que não são gerais para todas as aplicações daquela classe.

4.7. CONCLUSÕES DO CAPÍTULO

Gradativamente ontologias de tarefa estão recebendo mais atenção. Entretanto, diferentemente de ontologias de domínio, que têm diversos métodos e notação para elas, o desenvolvimento de ontologias de tarefa carece de abordagens de representação e metodologias para sua elaboração.

Neste capítulo, deu-se um passo nesta direção, focando em modelos para a representação das ontologias de tarefa e seus relacionamentos. Por envolver dimensões comportamentais e estruturais, utilizou-se o perfil OntoUML, que é baseado em diagramas de classes, para representar a parte estrutural, e definiu-se o perfil E-OntoUML, que adiciona aos diagramas de atividade da UML as distinções semânticas da UFO. Discutiu-se também como essas duas perspectivas são complementares e finalmente como ontologias de domínio podem ser facilmente integradas a ontologias de tarefa para gerar ontologias de classes de aplicação.

Pode-se ressaltar que a notação baseada em modelos da UML adiciona facilidade de compreensão e reusabilidade dos modelos gerados, principalmente pelo fato de ser uma notação amplamente difundida no desenvolvimento de software. Portanto, apesar da UML não dispor de uma semântica formal, seus diagramas são muito utilizados para a modelagem de ontologias e, com adição das distinções ontológicas da UFO, a linguagem passa a ter um

conjunto de primitivas que permite expressar os conceitos de forma mais concisa, mais completa e menos ambígua.

Tendo o conhecimento de determinada tarefa capturado e representado usando esses perfis UML, pode-se reusar esse conhecimento. No próximo capítulo é apresentada uma abordagem para a utilização desse conhecimento na Engenharia de Requisitos e é mostrado como o perfil E-OntoUML apoia essa reutilização.

É importante enfatizar que apesar de o perfil proposto ter sido elaborado com a intenção de representar ontologias de tarefa, nota-se que ele tem potencial para representar de diferentes tipos de modelos de tarefa e comportamentais, como modelos de processos de negócios, por exemplo.

CAPÍTULO 5. REUTILIZAÇÃO DE CONHECIMENTO DE TAREFA NA ENGENHARIA DE REQUISITOS

No capítulo anterior foi apresentada uma abordagem para representação do conhecimento de tarefa que utiliza de um perfil UML, uma linguagem comumente utilizada na modelagem de sistemas. Tendo esse conhecimento capturado em modelos compreensíveis, é viável que ele seja reusado. O reúso pode trazer grandes benefícios se adotado, sobretudo, em atividades complexas, como a modelagem de requisitos do sistema. Este capítulo apresenta uma abordagem que utiliza os modelos gerados com o perfil E-OntoUML na Engenharia de Requisitos para a derivação de modelos comportamentais e estruturais de sistemas.

5.1. INTRODUÇÃO

Conforme discutido no Capítulo 2, o processo de Engenharia de Requisitos (ER) envolve as atividades relativas ao tratamento dos requisitos de um sistema, incluindo: levantamento, modelagem, negociação, documentação, validação e gerência de requisitos (KOTONIA; SOMMERVILE, 1998). Em um processo de ER, geralmente, requisitos são inicialmente levantados e documentados na forma de sentenças que descrevem as funções que o sistema deve prover e restrições. A seguir, modelos estruturais e comportamentais do sistema são desenvolvidos. Esses modelos servem como uma base comum a todas as atividades desse processo, guiando os esforços (FALBO et al., 2006).

Os dois diagramas da UML mais amplamente usados na modelagem de software são os diagramas de classes, que proveem uma visão estrutural do sistema, e os diagramas de casos de uso que tratam de uma visão comportamental que engloba atores e principais funcionalidades do sistema. Visando detalhar as atividades de um caso de uso, ainda são desenvolvidos diagramas de atividade e de sequência, e para especificar as mudanças de estados de objetos de uma determinada classe são elaborados diagramas de estados.

Muitos esforços são despendidos com a elaboração desses modelos e uma forma de minimizar esses esforços é inserir abordagens de reúso no processo da ER. Para reusar o conhecimento de domínio ou de tarefa, é necessário capturar e formalizar esses conhecimentos, o que pode ser feito por meio de ontologias.

Muitos trabalhos tratam do reúso de ontologias de domínio na ER, entretanto abordagens para reúso de ontologias e modelos de tarefa ainda são pouco utilizadas. Propõe-se neste capítulo uma abordagem que visa reutilizar o conhecimento de tarefa juntamente com o de domínio, contidos em ontologias de domínio, de tarefa e de classes de aplicação.

A abordagem de representação de conhecimento de tarefa apresentada no capítulo anterior usa perfis da UML para modelar o conhecimento estrutural e comportamental de tarefas. Como defendido por Wang e Xang (2001), o uso da UML permite a usuários e engenheiros de software melhor entender uma ontologia por usar uma linguagem comum para eles. Além disso, a fácil compreensão desses modelos permite que eles possam ser mais rapidamente validados e reusados no desenvolvimento de software. Seguindo essas ideias, define-se neste trabalho um conjunto de diretrizes para extrair informações de ontologias de tarefa, de classes de aplicação e de aplicação que utilizam o perfil E-OntoUML para a elaboração de modelos de aplicações.

A abordagem está centrada no uso de ontologias como conhecimento para o entendimento do comportamento e levantamento dos requisitos, bem como para a elaboração de modelos estruturais e comportamentais. No que se refere a modelos estruturais, são derivados diagramas de classe; em relação aos modelos comportamentais, são derivados diagramas e descrições de casos de uso, diagramas de atividades para os casos de uso e diagramas de estados.

Para ilustrar as ideias propostas nesta dissertação, considera-se como estudo de caso a derivação de modelos para uma aplicação de locação de livros na UFES. A UFES possui diversas bibliotecas, sendo uma central e as demais setoriais, onde alunos e servidores podem locar livros. Ainda podem ser locados outros tipos de publicações e recursos, mas estes não são considerados com o objetivo de simplificar o estudo de caso.

O restante deste capítulo está organizado da seguinte forma: a Seção 5.2 apresenta a abordagem para reutilização de conhecimento estrutural; a Seção 5.3 trata do reúso de conhecimento comportamental; por fim, a Seção 5.4 apresenta as conclusões e considerações finais do capítulo.

5.2. DERIVAÇÃO DE MODELOS ESTRUTURAIS A PARTIR DE ONTOLOGIAS DE TAREFA, DE DOMÍNIO E DE CLASSES DE APLICAÇÃO

Como citado no Capítulo 2, diversos trabalhos tratam do uso de ontologias de domínio para apoiar a elaboração de modelos de estruturais de uma aplicação a ser desenvolvida nesse domínio. Porém o conhecimento de tarefa não tem sido levado em conta nessas abordagens.

Papéis de conhecimento representam a visão estrutural de uma tarefa. Integrando a parte estrutural de ontologias de tarefa a ontologias de domínio, têm-se novas informações que

definem os papéis dos conceitos do domínio na realização de uma tarefa. Os modelos de papéis de conhecimento se assemelham a padrões de análise e, portanto, como proposto em (NARDI, 2006), podem ser reusados da mesma forma que ontologias de domínio. Pelo fato de utilizarem a mesma notação, a do perfil OntoUML, a integração com ontologias de domínio é facilitada, o que tende a aumentar a consistência do modelo resultante.

Conforme discutido no Capítulo 4, pela integração de ontologias de domínio e de tarefa, pode-se obter o modelo estrutural de uma ontologia de classe de aplicações. A Figura 5.1 mostra esse modelo para a ontologia de classe de aplicações de locação de livros. Vale ressaltar que, a esse modelo, ainda podem ser integradas outras ontologias de tarefa, como uma ontologia da tarefa de reserva ou compra, e outras ontologias de domínio, como ontologia do domínio de organizações ou instituições de ensino superior, de forma a abranger todas as tarefas que uma classe de aplicações deve tratar e todos os domínios de sua atuação.

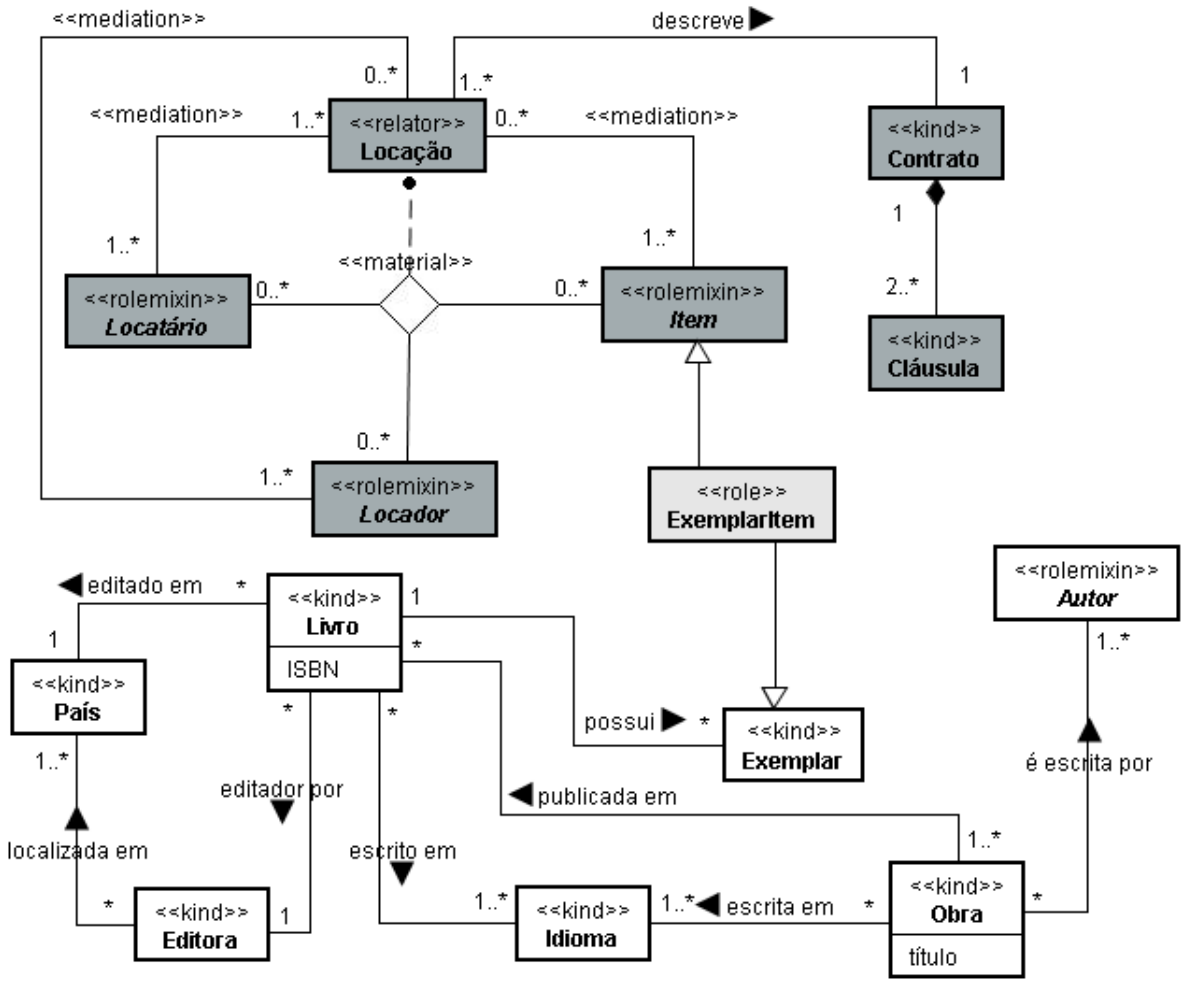


Figura 5.1: Modelo Estrutural de uma Ontologia da Classe de Aplicações de Locação de Livros, gerado a partir da integração das Ontologias de Tarefa de Locação e do Domínio de Livros.

A partir desse modelo estrutural da ontologia de classe de aplicação (Figura 5.1), podem ser derivados os modelos estruturais da ontologia de aplicação para a aplicação de locação de livros da Biblioteca da UFES, conforme as regras e diretrizes apresentadas nas próximas seções.

5.2.1. Derivação do Modelo Estrutural da Ontologia de Aplicação

A partir do modelo estrutural da ontologia de classe de aplicação apresentado na Figura 5.1, gerado pela integração de ontologias de domínio e tarefa, pode-se derivar a ontologia de aplicação para o sistema em questão. Basta adicionar a ela conceitos e relações que ainda sejam necessários para tratar os diversos aspectos dessa aplicação específica, bem como excluir aqueles que são desnecessários, por serem irrelevantes para a mesma. Ainda é possível trocar nomes dos conceitos para termos mais relevantes para o escopo do sistema.

Como mostra a Figura 5.2, para tratar conceitos que não estão representados nas ontologias de tarefa e de domínio, foram adicionados os conceitos de *Biblioteca*, *Acervo*, *Servidor* e *Aluno*. Além disso, foi alterado o termo *Locatário* para *Usuário*, pois este último é um termo mais relevante para o escopo definido. Por fim, foram adicionados os conceitos *BibliotecaLocadora*, *AlunoUsuário* e *ServidorUsuário* para compatibilidade com o padrão ontológico para modelagem de papéis, mencionado no Capítulo 4.

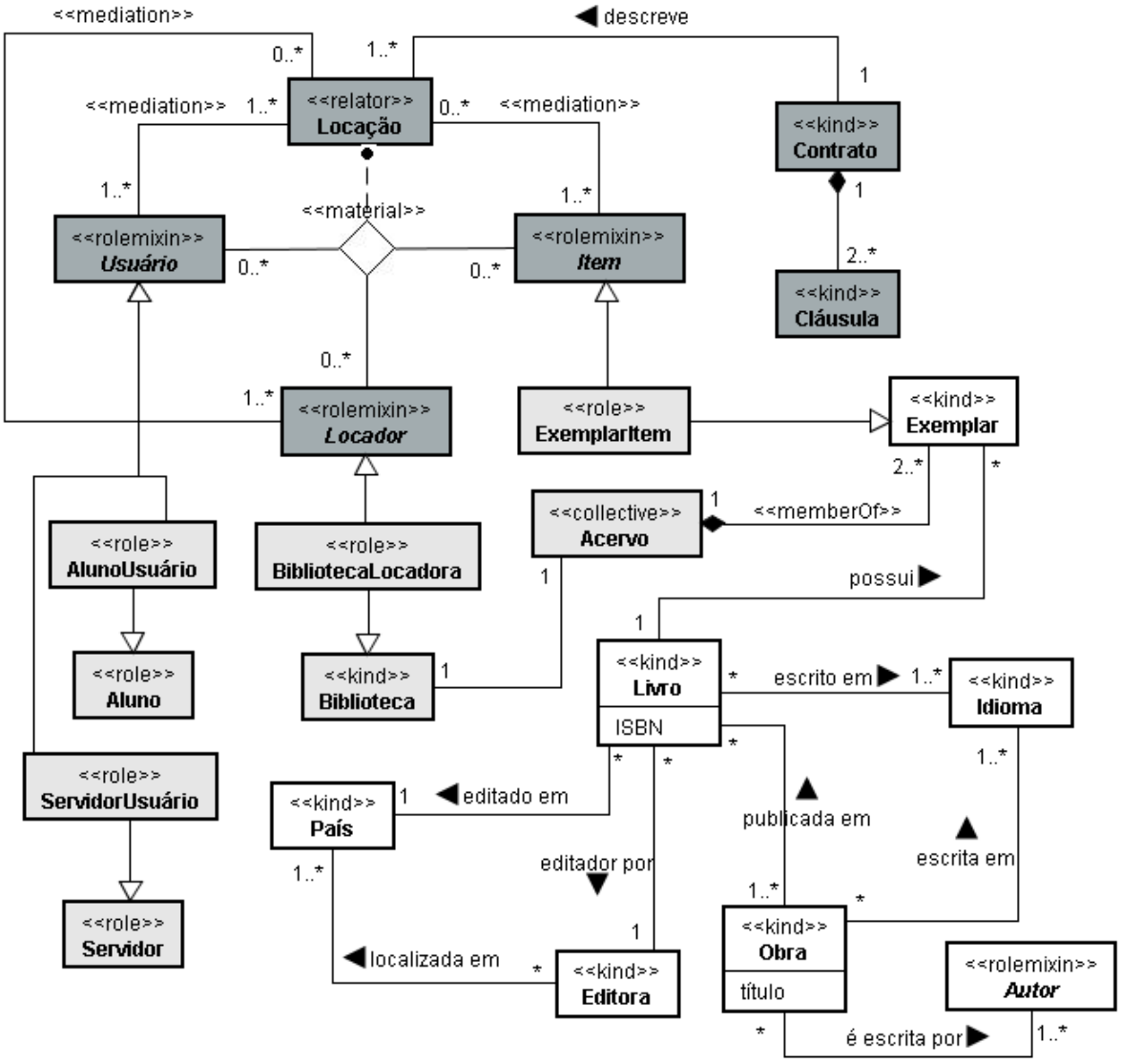


Figura 5.2: Modelo Estrutural da Ontologia de Aplicação de Locação de Livros da Biblioteca da UFES.

A Figura 5.3 complementa o diagrama da Figura 5.2 com aspectos necessários para a definição dos **papéis** (*role*) *Aluno* e *Servidor*. De acordo com a UFO, um **papel** (*role*) é um universal antirrígido e, portanto, aplica-se a um indivíduo apenas durante parte de sua existência, mais precisamente enquanto este estiver envolvido em uma relação que caracteriza o papel. No caso dos papéis *Aluno* e *Servidor*, trata-se de *Pessoas*, uma espécie (*kind*), que assumem esses papéis, respectivamente, quando se matriculam em um curso da Universidade ou quando são contratados por uma Unidade Organizacional da mesma.

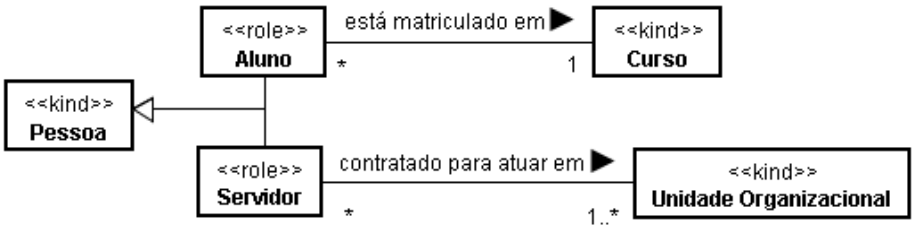


Figura 5.3: Definições complementares

5.2.2. Derivação do Modelo Estrutural da Aplicação

Definida a ontologia de aplicação, que possui sua parte estrutural modelada com o perfil OntoUML, podem-se derivar diagramas de classes considerando os mapeamentos propostos em (GUIZZARDI, 2000). Guizzardi define que, geralmente, os conceitos do domínio são mapeados para classes, relações para associações, propriedades para atributos e axiomas para métodos. Além disso, seguindo as diretrizes propostas em (GUIZZARDI, 2005) e (TAVARES, 2008) classes derivadas de papéis mistos (*rolemixins*) e categorias (*category*) devem ser classes abstratas, enquanto as demais podem ser tanto classes abstratas quanto concretas.

Considerando esses mapeamentos foi elaborado o diagrama de classes apresentado na Figura 5.4. Note que *Locador*, *Usuário* e *Item* tornam-se classes abstratas que são especializadas por elementos específicos do domínio. Além disso, a relação ternária material entre *Locador*, *Usuário* e *Item* foi retirada do diagrama, mantendo-se apenas as correspondentes relações de mediação. Esse diagrama é o ponto de partida para a elaboração do diagrama de classes da aplicação a ser usado no projeto do sistema, podendo sofrer alterações para incorporar detalhes específicos da aplicação em questão.

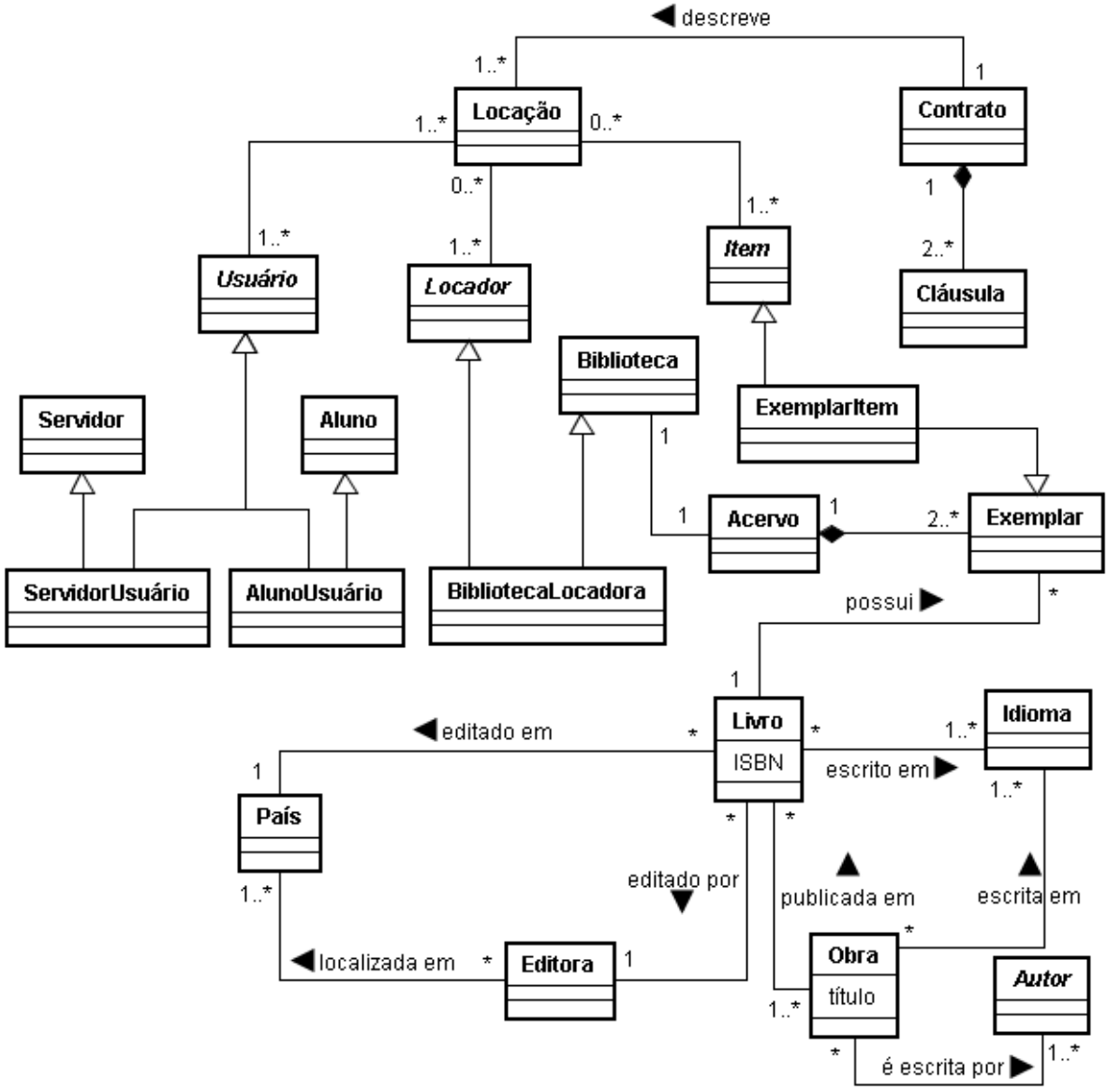


Figura 5.4: Diagrama de classes preliminar para a aplicação de locação de livro da UFES.

Em suma, foram realizadas três etapas principais no estudo de caso: (i) integração das ontologias de tarefa e de domínio, gerando uma ontologia de classe de aplicação, (ii) adaptação da ontologia de classe de aplicação à aplicação específica, criando uma ontologia de aplicação, e (iii) derivação do modelo conceitual da aplicação a partir da ontologia de aplicação. Essas etapas não são obrigatórias, já que as ontologias de domínio e os modelos estruturais das ontologias de tarefa e de classe de aplicação são reutilizáveis independentemente de estarem integrados ou não. A diferença é que, executando essas etapas, há uma adaptação gradativa de ontologias para modelos conceituais de sistema, pois a cada passo o *gap* semântico vai sendo diminuído. Além disso, usufrui-se ao máximo das regras de

validação que a linguagem OntoUML provê, criando modelos mais consistentes, menos ambíguos e com semântica mais próxima da realidade.

Considerando essas etapas, se a ontologia de aplicação desenvolvida for suficientemente completa, o modelo conceitual dela derivado tenderá a estar próximo do modelo conceitual a ser adotado, sem alterações significativas. Porém caso se reutilize diretamente as ontologias de domínio, tarefa ou de classe de aplicação, provavelmente maiores esforços serão necessários para adaptar o modelo derivado.

No exemplo de estudo apresentado, o modelo conceitual derivado diretamente da ontologia de aplicação (Figura 5.4) pode ser simplificado para melhor atender às especificidades da aplicação. Por não existir outra entidade que exercerá o papel de *Locador*, pode-se substituir diretamente *Locador* por *Biblioteca*, eliminando as classes *Locador* e *BibliotecaLocadora*, tornando o modelo mais simples, como mostra a Figura 5.5. Note que o mesmo não foi feito para *Exemplar*, *Item* e *ExemplarItem*, pois a UFES trabalha com a locação de outros tipos de itens. Dessa forma o sistema de locação de livros pode ser facilmente estendido, pois já está preparado para aceitar diferentes tipos de itens. Ainda foram excluídos os conceitos *Contrato* e *Cláusula*, pois em nível de aplicação, para o caso de uma biblioteca, essas informações se fundem na classe *Locação*.

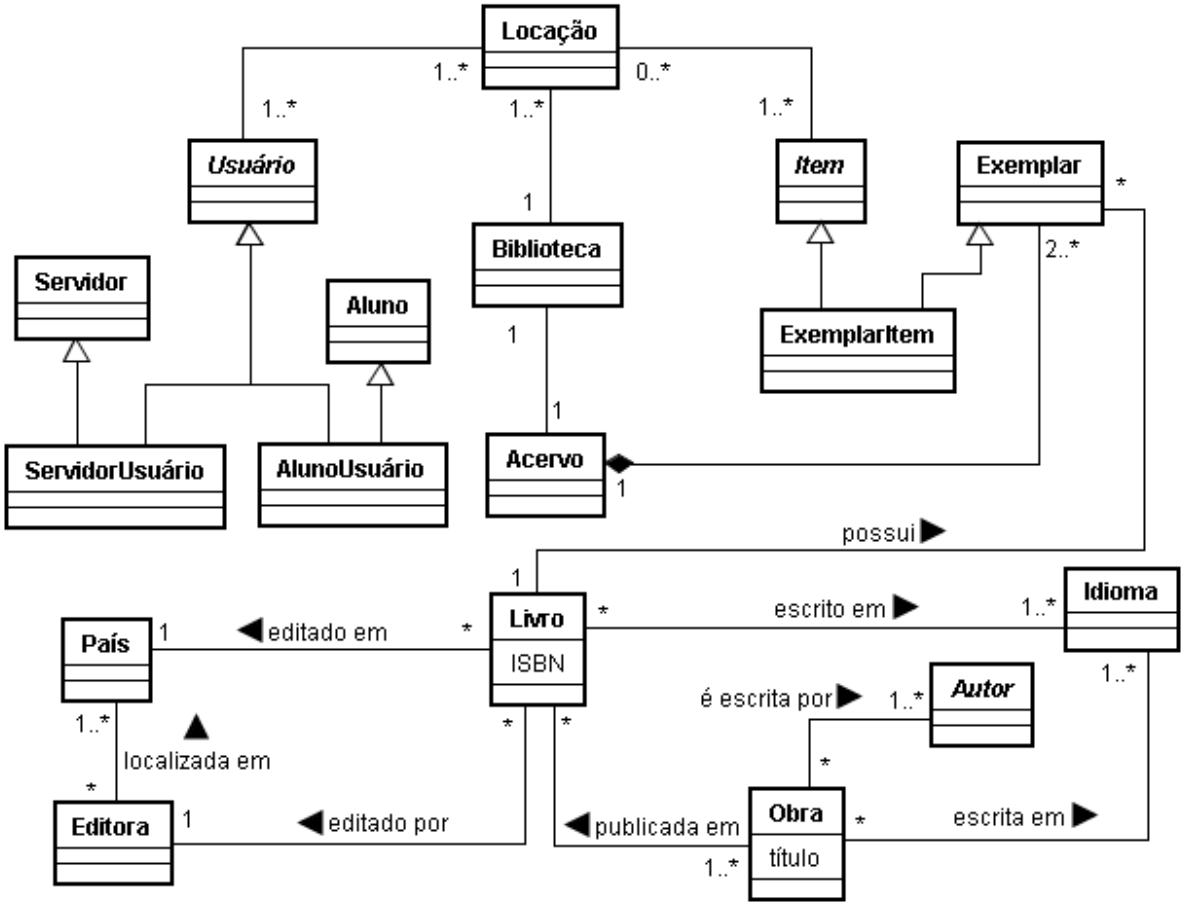


Figura 5.5: Diagrama de classes da aplicação de locação de livro da UFES.

5.3. DERIVAÇÃO DE MODELOS COMPORTAMENTAIS A PARTIR DE ONTOLOGIAS DE TAREFA E DE CLASSE DE APLICAÇÃO

Seguindo a mesma forma de apresentação utilizada da seção anterior, para mostrar a derivação de modelos comportamentais a partir de ontologias de classe de aplicações, inicialmente discute-se como chegar a ontologias de aplicação para depois, a partir delas, derivar os modelos comportamentais da aplicação, seguindo os mapeamentos sugeridos.

A Figura 5.6 mostra um dos modelos comportamentais resultantes da integração da Ontologia de Tarefa de Locação com a Ontologia de Domínio de Livros, discutida no Capítulo 4. Esse modelo é a base para derivação da parte comportamental da Ontologia de Aplicação da aplicação de locação de livros da Biblioteca da UFES, a qual é usada para derivar os modelos comportamentais para a aplicação, conforme discutido a seguir.

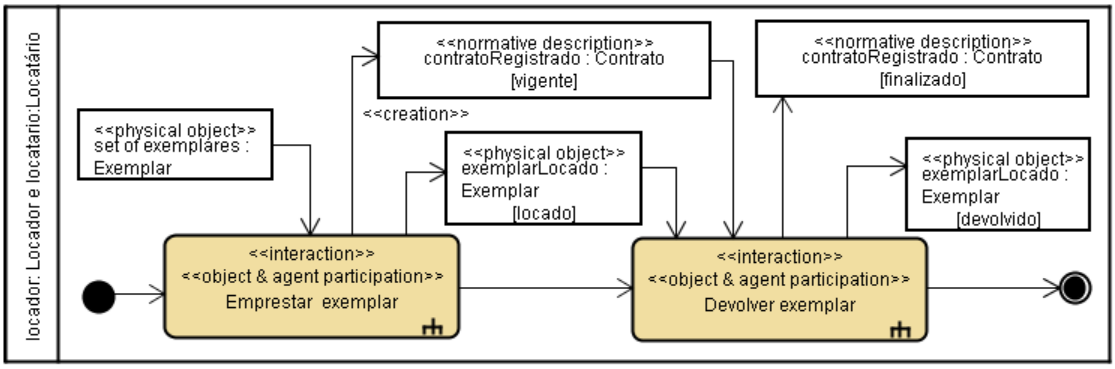


Figura 5.6: Modelo comportamental da Ontologia de Classe de Aplicação de Locação de Livros

5.3.1. Derivação do Modelo Comportamental da Ontologia de Aplicação

Para derivar o modelo comportamental de uma ontologia de aplicação, é necessário adaptar o modelo comportamental da ontologia de classe de aplicação (Figura 5.6) considerando os seguintes passos:

- Verificar os conceitos específicos de domínio e de aplicação que foram acrescidos na parte estrutural e substituir os correspondentes papéis de conhecimento, prestando atenção para especializações de agentes e objetos. No exemplo de estudo, *Locador* é substituído por *Biblioteca* que não é simplesmente um agente como *Locador*, mas sim uma organização (*organization*);
- Adaptar as tarefas para a aplicação específica, quando necessário. Por exemplo, na biblioteca da UFES não há como alterar as cláusulas de um contrato, o aluno pode apenas desistir da locação caso o prazo estipulado não seja de seu interesse. Logo, o fluxo deve ser alterado para representar esse fato.

As Figura 5.7 e Figura 5.8 mostram os modelos comportamentais da ontologia de aplicação resultantes da realização dos dois passos acima. Nessas figuras, os papéis de conhecimento foram substituídos por elementos específicos da aplicação definidos na parte estrutural (Figura 5.2). Além disso, vale destacar que houve alterações no modelo que trata da subtarefa “Emprestar exemplar”, mostrada na Figura 5.8, para tratar a especificidade da aplicação em questão, a saber, foi acrescida a subtarefa “Consultar Situação do Usuário”, pois é necessário verificar a situação do usuário antes de efetuar a locação,.

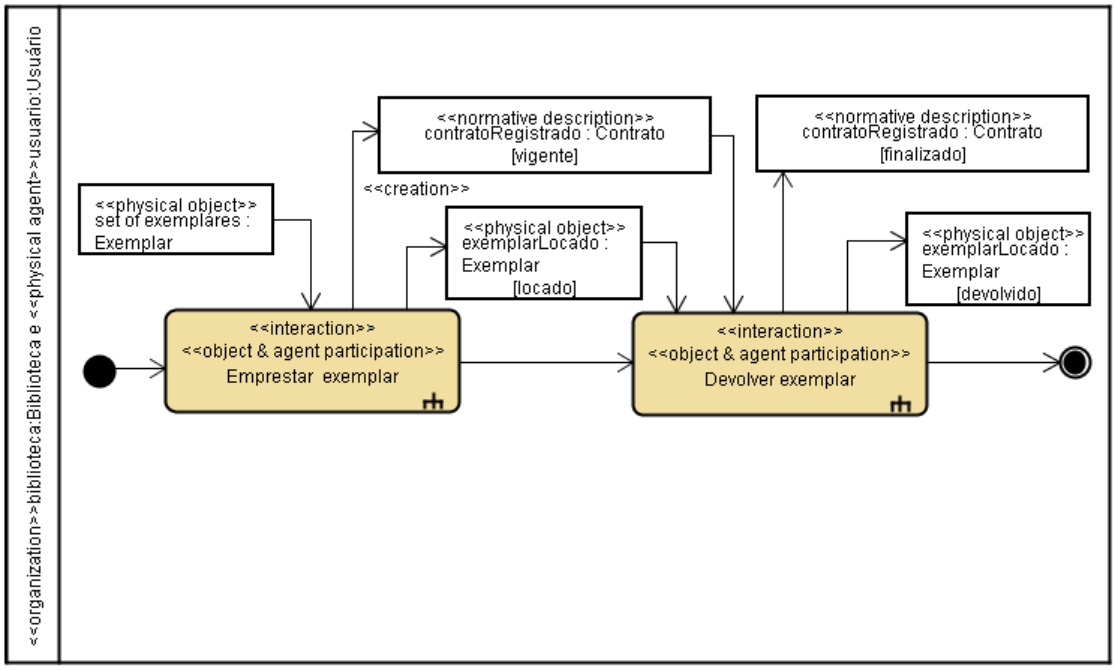


Figura 5.7: Modelo comportamental da Ontologia de Aplicação do Sistema de Locação de Livros da UFES

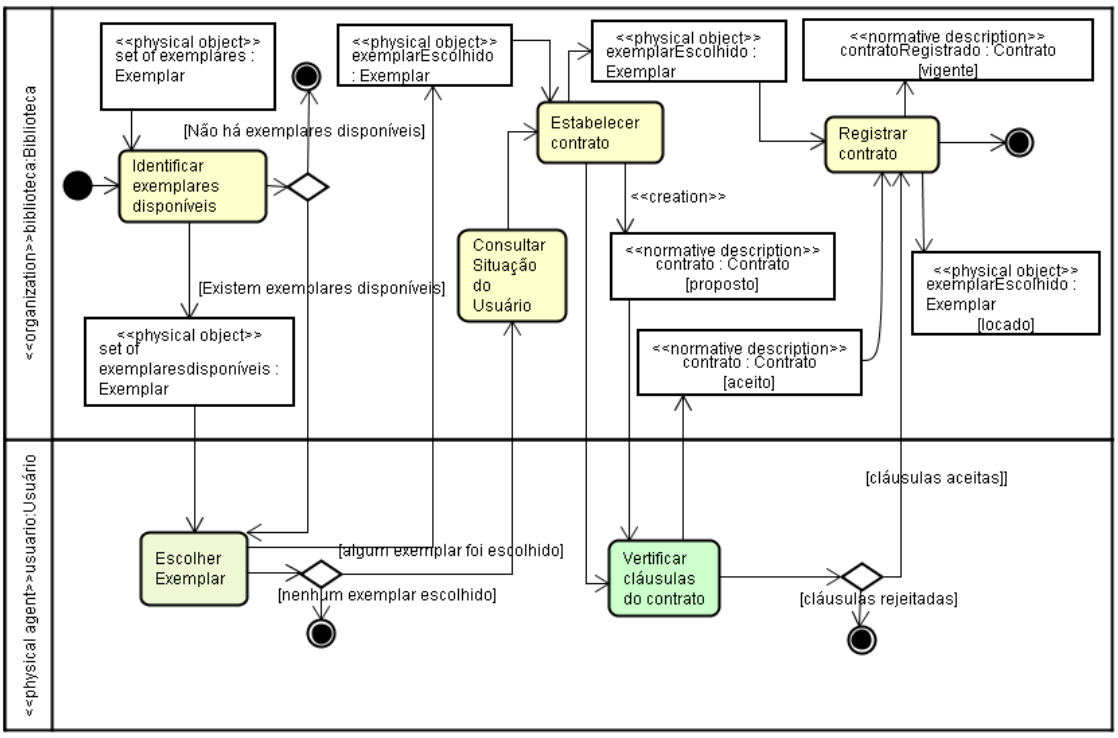


Figura 5.8: Diagrama de atividades da subtarefa "Emprestar Exemplar".

Nas Figura 5.7 e Figura 5.8 os agentes das ações passam a ser: *Usuário*, caracterizado agora como um agente físico (*physical agent*), tendo em vista que apenas pessoas físicas podem ser usuários da biblioteca da UFES, e *Biblioteca*, que é uma organização (*organization*). Eles desempenham, respectivamente, os papéis de Locatário e Locador durante a locação de livros na UFES. É importante ressaltar que os elementos do perfil E-

OntoUML permitem representar as especializações não somente dos agentes, mas também de objetos, como foi o caso de *Exemplar* discutido anteriormente.

Até este ponto foram apresentadas as alterações necessárias à parte comportamental da tarefa, quando integradas a ontologias de domínio, gerando ontologias de classe de aplicações (Capítulo 4), e quando derivadas ontologias de aplicações por inserir conceitos que são específicos da aplicação, como mostram as setas de linha contínua na Figura 5.9. Considera-se neste trabalho que as ontologias de domínio, de tarefa e de classe de aplicação são genéricas e, portanto, constituem um conjunto reutilizável de modelos, que podem ser reutilizadas diretamente no processo de Engenharia de Requisitos, como ilustram as setas de linha tracejada na Figura 5.9. Já as ontologias de aplicação são apenas um recurso de adaptação, para diminuir o *gap* entre as ontologias e os modelos conceituais das aplicações. Na sequência discute-se como esses modelos podem ser reutilizados no processo de Engenharia de Requisitos para a derivação do modelo comportamental da aplicação, considerando o último passo apresentado na Figura 5.9.

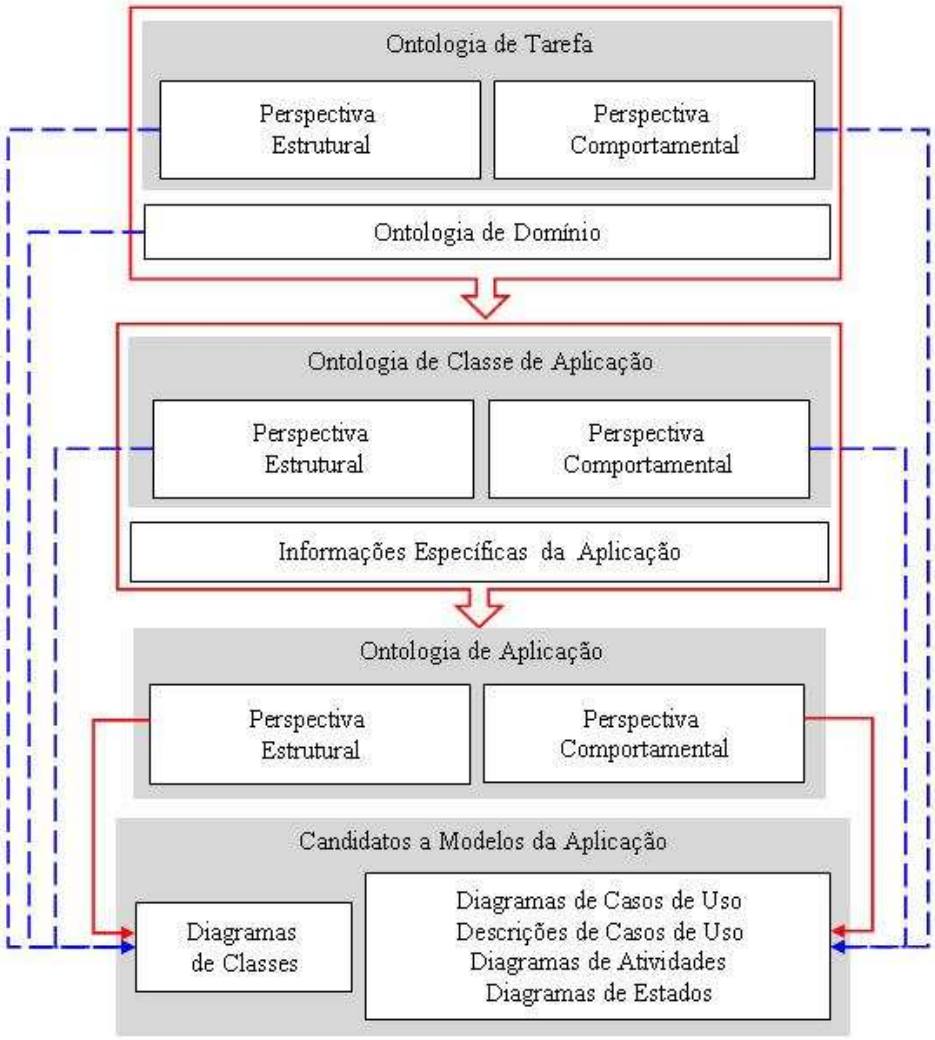


Figura 5.9: Visão Geral da Abordagem de Reutilização de Ontologias na Engenharia de Requisitos.

5.3.2. Derivação do Modelo Comportamental da Aplicação

O modelo comportamental de uma aplicação pode ser construído usando diferentes diagramas para mostrar diferentes perspectivas, dentre eles diagrama de casos de uso, diagramas de atividades, diagramas de estados, diagramas de sequência e diagramas de comunicação. Neste trabalho, explora-se a derivação de modelos de casos de uso a partir de ontologias de aplicação, ainda que algumas diretrizes também tratem de diagramas de atividades e de estados.

Considerando os elementos das ontologias de tarefa discutidos no Capítulo 4, podem-se identificar vários itens que tratam de aspectos similares aos considerados em um modelo de casos de uso. Ambos tratam de aspectos comportamentais: ontologias de tarefa capturam conceitos genéricos de uma tarefa e casos de uso definem as tarefas a serem realizadas por um sistema e como estas devem ser realizadas. Considerando o perfil E-OntoUML, foram

definidos mapeamentos entre os elementos do perfil e elementos de casos de uso. Para estabelecer uma base para a discussão, neste trabalho, considera-se que uma especificação de casos de uso tem os seguintes elementos:

- Mostrados em diagramas de casos de uso: nome do caso de uso, atores, associações de inclusão, associações de extensão e seus pontos de extensão,
- Mostrados em descrições de casos de uso: descrição do caso de uso, fluxos de eventos normais e alternativos, pré e pós condições e classes relacionadas.

5.3.3. Diagramas de Casos de Uso

A perspectiva comportamental das ontologias de tarefa e das ontologias geradas a partir delas (ontologias de classe de aplicação e de aplicação) trata da decomposição de uma tarefa em subtarefas até o nível de detalhamento (decomposição) desejado. Essas subtarefas capturam possíveis funcionalidades que uma aplicação deve realizar e são candidatas a casos de uso do sistema. Logo, dentre essas subtarefas, deve-se identificar quais que deverão ter o apoio computacional da aplicação e qual o nível de detalhamento que é importante para a modelagem do sistema, lembrando que não é relevante criar casos de uso para subtarefas não decompostas em outras (LU et al., 2003) (CRUZ, 2004), a não ser em casos de inclusão, conforme discutido mais adiante. Define-se, assim, o conjunto de tarefas que devem dar origem a casos de uso essenciais, identificando, para cada um deles, o nome, a partir do nome da tarefa correspondente (LU et al., 2003) (ZLOT et al., 2002). Considerando o exemplo da aplicação de locação de livros da Biblioteca da UFES, pode-se criar inicialmente os casos de uso “Emprestar Exemplar” e “Devolver Exemplar”, que derivam das tarefas principais de uma locação de livros, segundo a Figura 5.7.

As ontologias definem também quais os agentes que participarão das tarefas, os quais podem corresponder a atores que realizam os respectivos casos de uso (LU et al., 2003). Analisando os diagramas de atividades construídos usando o perfil E-OntoUML, vê-se que os agentes são representados em partições de atividades de agentes (*AgentActivityPartition*) e suas especializações, logo os elementos descritos nas partições são candidatos a atores (CRUZ, 2004). Há, ainda, uma consideração a ser feita quanto a agentes sociais, representados por partições de atividades de agentes sociais (*SocialAgentActivityPartition*). Uma vez que um ator em diagramas de casos de uso representa um papel exercido por um usuário ou outro sistema que interage com o sistema em questão, pode ser útil substituir o agente social por um agente físico que o represente e que interaja efetivamente com o sistema.

Ainda em relação a atores, quando se integram ontologias de domínio a ontologias de tarefa, identificam-se no domínio os conceitos que exercem os papéis definidos na tarefa. Pode-se definir como atores tanto os papéis de conhecimento definidos na ontologia de tarefa quanto os conceitos do domínio que os exercem. Se ambos forem mantidos, deve haver uma associação de especialização entre esses atores, como ilustra a Figura 5.10.

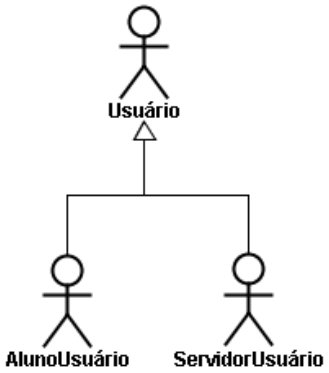


Figura 5.10: Especialização de atores

Tomando por base a ontologia da tarefa de locação, tem-se como atores o **locador** e o **locatário**, porém quando se integra a ontologia de domínio e adicionam-se novos elementos para criar a ontologia de aplicação (conforme apresentado na seção anterior), atribui-se aos papéis **aluno** e **servidor** ser um **locatário (usuário)** e à **biblioteca** ser um **locador**. No caso da Biblioteca, como ela é uma organização (*organization*), ou seja, um agente social, é mais indicado modelar um agente físico que a represente, tal como um bibliotecário, que será quem de fato vai interagir com o sistema, conforme mostra a Figura 5.11. O mesmo procedimento é válido para agentes definidos como agentes sociais (*social agent*), agentes sociais coletivos (c*ollective social agent*) e sociedades (*society*).

Outro aspecto importante a se considerar é que, nas tarefas que são candidatas a casos de uso, devem-se derivar atores apenas para os agentes que realizam pelo menos uma subtarefa que terá apoio computacional e que necessita da interação do agente com o sistema, ou seja, alguma tarefa que é parcialmente automatizada. No estudo de caso, “Emprestar Exemplar” representa uma interação entre um bibliotecário e um usuário, porém apenas o bibliotecário é responsável pelas tarefas que serão apoiadas pelo sistema. Portanto, apenas *Bibliotecário* será ator desse caso de uso, descartando-se o ator *Usuário*, como mostra a Figura 5.11.

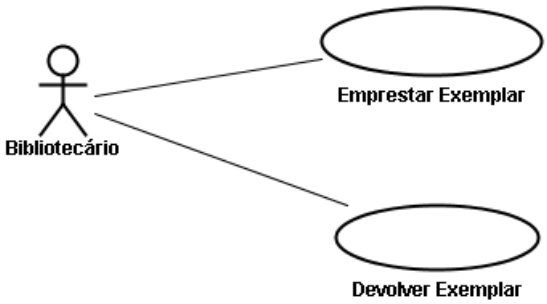


Figura 5.11: Casos de Uso Essenciais

Além de casos de uso essenciais derivados diretamente das tarefas mostradas na ontologia, também é possível derivar casos de uso custodiais, relativos a funcionalidades de manutenção das informações a serem manipuladas pelos casos de uso essenciais. Prováveis casos de uso incluem o cadastro dos atores envolvidos (CRUZ, 2004). Além disso, as participações de recurso indicam a necessidade de cadastros para os objetos correspondentes ou pelo menos funcionalidades de consulta (CRUZ, 2004). Os tipos de participações que ocorrem em uma tarefa definem os eventos de casos de uso de cadastro ou passos de casos de uso essenciais. Considerando os casos de uso essenciais, podemos dizer que: (i) o uso (*usage*) requer tipicamente uma funcionalidade de consulta de informações sobre o objeto, (ii) a criação (*creation*) sugere que haja um evento para criar objeto, (iii) uma alteração (*change*) indica um passo de atualização de informações do objeto e por fim, (iv) uma participação de término (*termination*) requer um passo para a exclusão do objeto. Todos esses eventos podem ser descritos no fluxo de um caso de uso essencial. Entretanto, quando eles forem úteis em outras funcionalidades (casos de uso), deve-se investigar a possibilidade da criação de um caso de uso que será incluído ou que estenderá outros casos de uso. Outro aspecto importante é que se pode derivar casos de uso custodiais para a manutenção das informações (criação, alteração, consulta e exclusão) para cada classe de objetos que participam de tarefas. Logo: (i) se há uma participação de criação, então potencialmente precisa-se de um caso de uso custodial com eventos para consultar, alterar e excluir; (ii) se há uma participação de uso, potencialmente há a necessidade de haver um caso de uso custodial para criar, alterar e excluir; (iii) se há uma participação de alteração, potencialmente há a necessidade de haver um caso de uso custodial para criar, excluir e consultar; e (iv) se há uma exclusão, pode haver a necessidade de um caso de uso custodial para criar, alterar e consultar o objeto.

Além de olhar para as participações com o objetivo de identificar casos de uso custodiais, é importante também verificar o modelo estrutural, pois certamente as informações nele contidas serão importantes para o apoio à realização das funcionalidades do sistema.

Logo, no exemplo, podemos identificar o cadastro de livros, usuários, exemplares, bibliotecas, editoras, entre outros. A relevância deve ser definida de acordo com informações identificadas no levantamento feito junto aos interessados. Na Figura 5.12 são apresentados alguns dos cadastros relevantes para o exemplo da Biblioteca.

Os atores responsáveis pelos casos de uso de cadastro devem ser identificados com um levantamento de requisitos complementar, pois nem sempre serão exatamente aqueles que executam as tarefas que incluem participações dos recursos. Além disso, os cadastros relativos a papéis de conhecimento podem ser especializados para os conceitos do domínio ou elementos da aplicação que os exerçam. No caso da locação, se for criado o caso de uso "Cadastrar Usuário", o mesmo deverá ser especializado em "Cadastrar Aluno Usuário" e "Cadastrar Servidor Usuário". Porém vale lembrar que essa especialização poderá ser retirada, deixando-se apenas os casos de uso mais especializados, caso não exista comportamento comum relevante entre eles. Em outras palavras, a especialização ocorre quando um caso de uso provê os mesmos fluxos de eventos principais, possivelmente alterados do caso de uso mais geral, além de poder prover novos fluxos (OMG, 2007). Na Figura 5.12 são mostrados os casos de uso custodiais para aplicação da Biblioteca de UFES. Nessa figura nota-se a necessidade de dois novos atores, a saber o Sistema de Recursos Humanos da Universidade e o Sistema Acadêmico da mesma. O primeiro provê informações de servidores para o cadastro dos mesmos como usuários da biblioteca; o segundo de alunos.

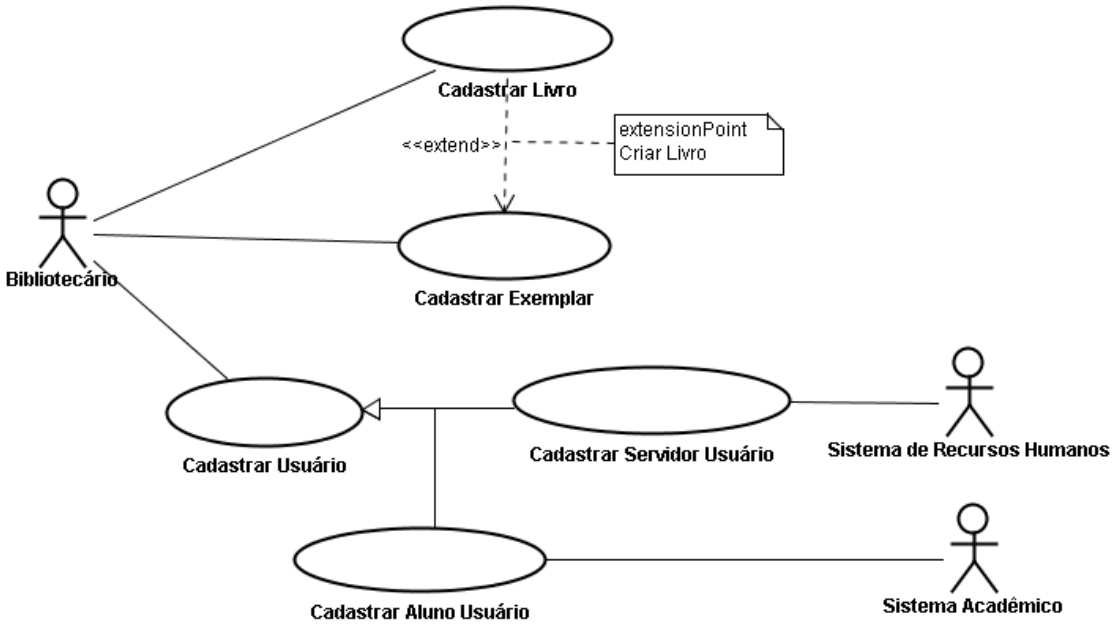


Figura 5.12: Casos de uso Custodiais

Com respeito a relacionamentos entre casos de uso, podem ser criados relacionamentos de inclusão quando uma subtarefa de uma tarefa que originou um caso de uso for compartilhada por outra tarefa que derivou outro caso de uso. Assim essa subtarefa também deve tornar-se um caso de uso por ser necessária para a realização de mais de um caso de uso. Deve-se frisar que, para essa situação ser modelada como uma inclusão, é necessário que o caso de uso de inclusão proveja resultados indispensáveis para o caso de uso que o inclui. Caso contrário, poder-se-á usar o relacionamento de extensão. Neste caso, o caso de uso de extensão deve ser independente do significado do caso de uso estendido e frequentemente é complementar à realização da tarefa (condicional ou não) (OMG, 2007).

Muitas vezes, pontos em que entidades são criadas, alteradas, consultadas ou excluídas podem se tratar de inclusões ou extensões dos cadastros definidos. Se o comportamento for parte essencial para mais de um caso de uso, então deve-se usar um relacionamento de inclusão. Se o comportamento em questão não for parte crucial da realização do caso de uso e/ou necessitar de uma condição, deve-se usar um relacionamento de extensão. Neste último caso a condição de guarda daquele fluxo define o ponto de extensão. Em outros casos o ponto de extensão será o evento que é requisitado pela extensão. No exemplo da Figura 5.13, os casos de uso “Emprestar Exemplar” e “Devolver Exemplar” são estendidos pelos casos de uso “Cadastrar Usuário” e “Cadastrar Exemplar”, pois pode ser necessário consultar informações do usuário e do exemplar ao realizar uma locação.

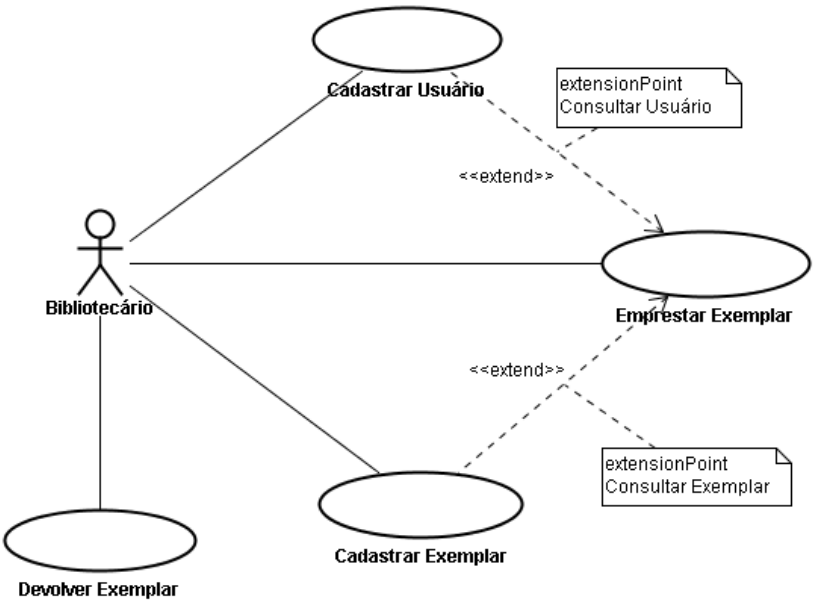


Figura 5.13: Casos de uso derivados da ontologia.

5.3.4. Descrições de Casos de Uso

Quando se analisam os modelos comportamentais elaborados usando o perfil E-OntoUML podem-se capturar informações acerca dos passos tanto do fluxo normal quanto de fluxos alternativos de um caso de uso, observando a decomposição de uma tarefa em subtarefas e sua ordem de execução. A partir daí, pode-se elaborar uma descrição dos fluxos de eventos para os casos de uso.

A Figura 5.14 apresenta um exemplo de especificação de caso de uso para o caso de uso “Emprestar Exemplar” com base nas informações provenientes da visão comportamental da ontologia de aplicação. Por fim, as classes relacionadas ao caso de uso são identificadas na tarefa como os objetos que participam da mesma, os quais devem ser instâncias de classes do modelo estrutural do sistema.

Caso de Uso: Emprestar Exemplar

Descrição: Esse caso de uso é responsável por realizar uma locação de um exemplar de livro da Biblioteca da UFES para um usuário.

Pré-condições: Haver um conjunto de exemplares para a locação e um deles ter sido escolhido pelo usuário.

Pós-condições: O exemplar do livro é locado e há um registro da locação que estabelece um contrato dessa locação.

Fluxo principal

1. O bibliotecário informa o usuário e o exemplar a ser locado.
2. Verifica-se se o usuário está em situação regular.
3. Se o usuário estiver em situação regular, uma nova locação é definida considerando a data corrente como data de locação e a data de devolução é calculada de acordo com prazo definido para cada tipo de usuário.
4. Se o usuário aceitar as condições da locação, a locação é registrada e o exemplar passa a estar registrado como locado, o que o torna indisponível para outras locações.
5. O exemplar é entregue ao usuário.

Fluxos alternativos

1. Se o usuário estiver em situação irregular no passo 2, a locação não é concretizada e uma mensagem de erro é exibida. O bibliotecário pode realizar o fluxo de eventos “Consultar Usuário” do caso de uso “Cadastrar Usuário” para consultar informações do usuário, incluindo os seus débitos.
2. Se no passo 4 o usuário não aceitar as condições de locação, a locação não é concretizada.
3. No passo 1, se desejado, o bibliotecário pode consultar informações do exemplar, realizando o fluxo de eventos “Consultar Exemplar” do caso de uso “Cadastrar Exemplar”.

Classes Relacionadas: Usuário, Exemplar, Locação, Item, ItemExemplar.

Figura 5.14: Exemplo de descrição de um caso de uso para Empréstimo de um Exemplar de Livro.

5.3.5. Diagramas de Atividades e de Estados

Diagramas e descrições de casos de uso tratam algumas das informações providas pelas ontologias de tarefa e ontologias delas derivadas (ontologias de classe de aplicação e ontologias de aplicação). Logo, é útil explorar outros modelos que também podem ser criados

para aproveitar ao máximo as informações contidas nessas ontologias, tais como diagramas de atividades e diagramas de estados

Quando uma tarefa derivar um caso de uso, diagramas de atividades podem ser elaborados com base nos diagramas de atividades desenvolvidos como parte do modelo comportamental das ontologias, bastando adaptar o modelo à aplicação em questão.

Retomando o exemplo da biblioteca da UFES, note que pela própria descrição é possível identificar pontos em que é necessária a adaptação. Considerando o caso de uso "Emprestar Exemplar", descrito na Figura 5.14, e o diagrama de atividades "Emprestar Exemplar" da ontologia de aplicação, apresentado na Figura 5.8, chegou-se ao modelo apresentado na Figura 5.15, no qual foram feitas adaptações necessárias para adequar o modelo ao sistema, a saber: (i) foram retirados os estereótipos específicos do perfil E-OntoUML, (ii) acrescentaram-se as informações dos usuários como entradas para o processo de locação, (iii) foram eliminadas tarefas não apoiadas pelo sistema em desenvolvimento, (v) como feito no modelo estrutural, considerou-se *Locação* como o registro que contrata o empréstimo, e (iv) a atividade de verificação das cláusulas da locação foi delegada ao bibliotecário, pois o usuário não interage diretamente com o sistema.

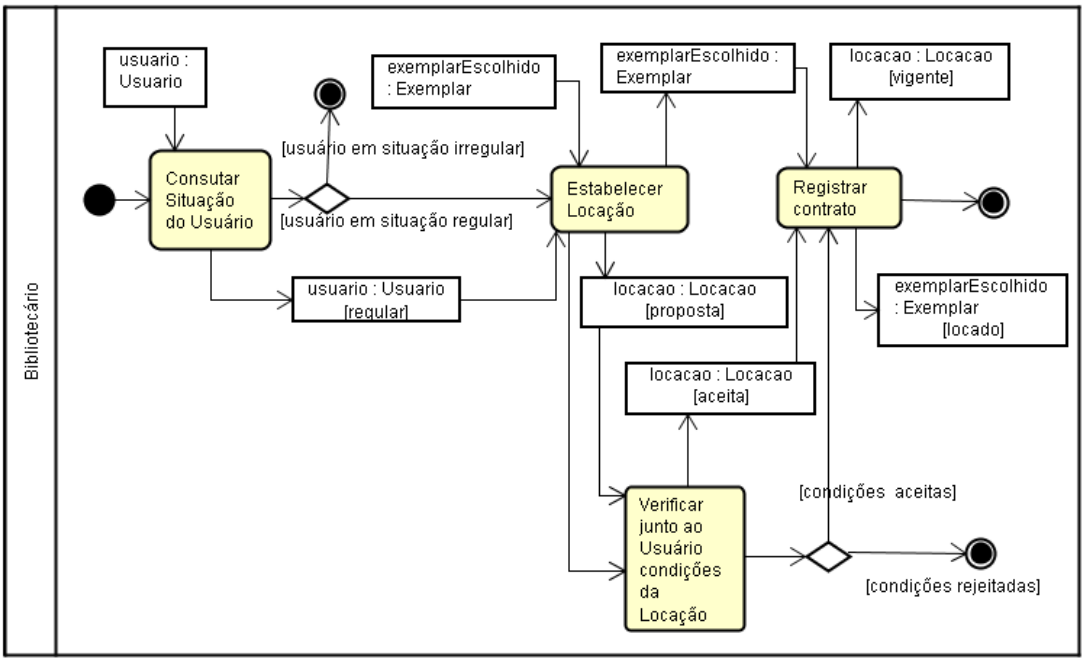


Figura 5.15: Diagrama de Atividades para o caso de uso "Emprestar Exemplar".

Voltando à Figura 5.7, existe ainda uma informação considerada importante na modelagem de sistemas, que são as fases ou estados dos objetos. Essas informações podem ser capturadas em um diagrama de estados, como mostra a Figura 5.16, que ilustra a representação das fases da entidade *Item* em um diagrama de estados. Esse diagrama captura

as transições de estados, obtidas a partir da Figura 5.7, observando a situação dos objetos. No exemplo, são apresentadas as Fases de um *Item,* que tem como estado inicial “disponível” que é alterado para “locado” sempre que é feito um empréstimo e retorna ao seu estado anterior quando é devolvido.

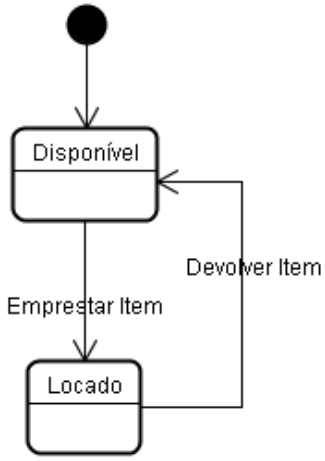


Figura 5.16: Fases de um Item.

5.4. CONCLUSÕES DO CAPÍTULO

A abordagem de reúso apresentada neste capítulo está baseada nos quatro tipos de ontologias definidos por Guarino (1998) e ontologias de classe de aplicação para a geração de modelos estruturais e comportamentais de sistema. Ontologias de domínio, de tarefa, de classe de aplicação e de aplicação, que utilizam uma notação baseada em uma ontologia de fundamentação, foram utilizadas como fonte para se chegar a modelos de uma aplicação real, incluindo diagramas de classes, de casos de uso, de atividades e de estados.

Como vantagem quando comparada a uma abordagem de Engenharia de Requisitos (ER) tradicional, pode-se citar a criação de modelos bem fundamentados com semântica mais próxima à realidade. Além disso, o reúso minimiza problemas recorrentes da ER, como: dificuldades de comunicação com o cliente, omissão de informações, compreensão dos termos desconhecidos e falta de acesso a especialistas.

CAPÍTULO 6. CONSIDERAÇÕES FINAIS

Este capítulo apresenta as conclusões do trabalho realizado, mostrando suas contribuições, pontos positivos e negativos, além das dificuldades encontradas no seu desenvolvimento. Descreve, ainda, possíveis trabalhos futuros que podem surgir baseados neste.

6.1. CONCLUSÕES

O reúso de conhecimento vem sendo apontado como meio para diminuir tempo e custos nas fases iniciais do desenvolvimento de software, bem como para aumentar a qualidade dos modelos gerados. Visando à reutilização, dois principais tipos de conhecimento podem ser considerados: conhecimento de domínio e conhecimento de tarefa (GUARINO, 1998). É necessário capturar ambos e ontologias podem ser utilizadas para organizar o conhecimento capturado, facilitando o acesso, o entendimento e permitindo o reúso.

De acordo com Guarino (1998), existem quatro tipos principais de ontologias: ontologias de fundamentação, de domínio, de tarefa e de aplicação. Ontologias de domínio vêm sendo muito estudadas e utilizadas como artefatos reutilizáveis (GUIZZARDI, 2005), (GÓMEZ-PÉREZ et al., 2004), (NARDI, 2006) e (FALBO, 2004), já existindo diversos métodos bem estruturados para o desenvolvimento das mesmas. O mesmo não ocorre com ontologias de tarefa, para as quais não existe uma representação padrão, metodologias difundidas para sua elaboração e técnicas para integração com outros tipos de ontologias.

O conhecimento modelado em ontologias de tarefa captura informações estruturais a respeito dos papéis que entidades de domínio vão exercer numa tarefa e seus relacionamentos, e comportamentais, que definem a decomposição e o fluxo de controle nas subtarefas. Essas ontologias podem ser importantes artefatos reutilizáveis no desenvolvimento de software, principalmente no processo de Engenharia de Requisitos, quando se definem os requisitos do sistema, incluindo as tarefas que o mesmo deve apoiar.

Este trabalho procurou avançar na área de Engenharia de Ontologias de Tarefas, propondo a separação dos aspectos estruturais e comportamentais de ontologias de tarefas em modelos separados, mas de forma conectada. Essa separação favorece a organização do conhecimento e facilita a integração de ontologias de tarefa com ontologias de domínio.

Para representar os papéis de conhecimento, usou-se o perfil OntoUML (GUIZZARDI, 2005), que é uma extensão de diagramas de classes tomando por base a semântica da UFO-A, ou seja, a parte relativa a objetos da Ontologia de Fundamentação Unificada (UFO)

(GUIZZARDI, 2005). Esse perfil já vem sendo utilizado para a modelagem de ontologias de domínio. Para modelar a decomposição de uma tarefa em subtarefas e o fluxo de controle nas subtarefas, desenvolveu-se um perfil UML, denominado E-OntoUML. Ele também é fundamentado na UFO, porém nas partes B e C, que tratam de eventos e entidades sociais, respectivamente. E-OntoUML é uma extensão do metamodelo da UML em sua parte referente a diagramas de atividades. Essa extensão visa adicionar aos elementos de modelos da UML distinções ontológicas importantes capturadas nos conceitos da UFO.

A representação proposta facilita a integração de ontologias de domínio e ontologias de tarefa para a derivação de ontologias de classes de aplicação. O uso da mesma notação para ontologias de domínio e para a parte estrutural do conhecimento de tarefa facilita a integração que deve respeitar os postulados da linguagem OntoUML. Feita a integração na parte estrutural, torna-se simples a adaptação da parte comportamental, bastando substituir os papéis de agentes e objetos por conceitos do domínio correspondentes e definindo a sua especialização segundo a hierarquia de agentes e objetos definida no perfil.

Por fim, visando aproveitar os benefícios providos pela representação de ontologias de tarefa proposta, foi desenvolvida uma abordagem para reutilizar esse conhecimento no desenvolvimento de software, apoiando atividades de modelagem do processo de Engenharia de Requisitos. As ontologias de domínio, de tarefa e de classe de aplicação, modeladas com os perfis UML propostos, agregam conhecimento para a derivação de modelos estruturais, como diagramas de classes, e modelos comportamentais, como diagramas de casos de uso, diagramas de atividades e diagramas de estados. A principal vantagem decorrente do uso desses perfis está no fato de eles utilizarem a notação da UML que é bem conhecida e muito utilizada no desenvolvimento ao mesmo tempo que capturam importantes distinções ontológicas providas por UFO.

Em suma, são contribuições deste trabalho:

- ***Organização do conhecimento de tarefa:*** Elaboração de uma abordagem de organização do conhecimento de tarefa por meio de ontologias de tarefa, considerando duas perspectivas capturadas por modelos diferentes, mas fortemente relacionadas: uma comportamental e outra estrutural.
- ***Representação de ontologias de tarefa***: Para a representação da parte estrutural foi utilizado o perfil OntoUML, baseado na UFO e utilizado para representação de ontologias de domínio. Já para a parte comportamental criou-se o perfil E-OntoUML.
- ***Perfil E-OntoUML:*** Foi elaborado do perfil E-OntoUML, um perfil UML fundamentado nas partes B e C da UFO. Esse perfil utiliza diagramas de atividades com elementos estereotipados para representar os principais conceitos envolvidos em uma tarefa.
- ***Integração de ontologias de tarefa com ontologias de domínio:*** Considerando que ontologias de domínio são também representadas utilizando-se um perfil UML, foi proposta uma abordagem para a integração de ontologias de domínio e de tarefa para dar origem a ontologias de classe de aplicação.
- ***Abordagem de reúso de conhecimento de tarefa:*** O uso da UML torna compreensíveis e altamente reutilizáveis os modelos de ontologias que utilizam OntoUML e E-OntoUML. Explorando essa característica, foi apresentada uma abordagem para o reúso de conhecimento de tarefa no processo de Engenharia de Requisitos, com diretrizes para a derivação de diagrama de classes, de atividades, de estados e de casos de uso, incluindo a elaboração de descrições de casos de uso.

As abordagens apresentadas neste trabalho, apesar de utilizarem como base para apresentação a UML, que é uma linguagem de modelagem muito difundida entre desenvolvedores de software, podem ter o seu entendimento um pouco dificultado, pois não é trivial entender a semântica que o perfil adiciona aos modelos da UML. Isso decorre do fato de os conceitos da UFO, na qual ele se fundamenta, serem, algumas vezes, de difícil compreensão. A UFO visa adicionar semântica de mundo real às linguagens que nela se fundamentam. Essa é uma vantagem importantíssima, mas à vista de desenvolvedores que trabalham com uma visão mais técnica do problema, os conceitos envolvidos podem gerar alguma dificuldade. Porém essa fundamentação é indispensável para a produção de modelos

mais consistentes, menos ambíguos e mais reutilizáveis. Para poder tirar conclusões mais bem embasadas, é importante a realização de um estudo experimental que avalie a utilidade do perfil e das abordagens propostas em situações práticas. A realização de tal estudo, contudo, transcende o tempo disponível para a realização deste trabalho e deve ser realizado futuramente.

Durante o desenvolvimento de E-OntoUML, a maior dificuldade foi o tamanho e a complexidade do metamodelo da UML e da UFO. A UML é bem documentada, mas a semântica dos elementos não é bem definida e muitos elementos de modelos são usados em diversos diagramas com significados diferentes. A UFO, por sua vez, é um projeto em andamento e que apresenta alguns conceitos ainda não muito bem definidos. Existem diversas publicações, mas ainda assim é difícil encontrar definições para todos os conceitos e não há publicações que se proponham a mostrá-la como um todo. Por causa da imaturidade das discussões sobre alguns conceitos da UFO, como situações, proposições e objetivos, optou-se por deixar de lado a representação de objetivos, neste primeiro momento.

Outro problema observado no desenvolvimento foi a forma como foi realizado o mapeamento entre os conceitos da UFO e os elementos de modelos da UML. Primeiro foi criada e publicada uma abordagem de uso de diagramas da UML, sem qualquer fundamentação ontológica, para a representar as duas visões do conhecimento de tarefa. Foi proposto o uso de diagramas de classe para representar a visão estrutural e diagramas de atividades para a visão comportamental. A partir daí buscou-se a fundamentação ontológica para esses modelos. Dessa forma optou-se por verificar para cada elemento do meta-modelo da UML qual o conceito da UFO que ele simbolizava. Porém, acredita-se que a linha de pensamento adotada deveria ter sido a contrária: ter partido da UFO, que é a base semântica, e identificado dentre os elementos dos diagramas de atividades aqueles que fossem capazes de representar os conceitos da UFO. Ainda assim, não é certo que se chegaria aos mesmos resultados, nem que o desenvolvimento da abordagem seria simplificado.

Uma observação importante a ser feita a respeito de E-OntoUML é que esse perfil tem potencial para ser utilizado como ferramenta ontologicamente fundamentada para uma modelagem de processos de negócio (*Business Process Modeling* – BPM). É aparente que ele precisa atingir um maior grau de maturidade e tratar alguns aspectos mais específicos de modelagem de processos, mas os principais conceitos necessários para representar processos já estão definidos.

Outro aspecto importante a se considerar é que não há na literatura explicações a respeito de ligações entre os conceitos da UFO-A e conceitos das UFOs B e C. Isso faz com

que o relacionamento entre os elementos dos modelos estruturais, baseados na UFO-A, e os conceitos dos modelos comportamentais, baseados nas partes B e C da UFO, pareçam tratar de coisas desconexas, sendo necessário um conhecimento mais profundo da UFO para identificar essas ligações.

Apesar das dificuldades apresentadas, este trabalho dá um passo importante em direção a uma representação consistente do conhecimento de tarefa. A abordagem proposta permite criar modelos ontologicamente bem fundamentados e por utilizar perfis da UML, torna-se atrativa, uma vez que a UML é um padrão. Esses fatos não contribuem apenas para uma potencial boa aceitação do perfil proposto, mas também para o reúso de modelos de tarefa durante o desenvolvimento. Muitos trabalhos vêm mostrando as vantagens do uso de modelos de processos e de tarefas no desenvolvimento de software, porém essa utilização ainda não é expressiva. Este trabalho provê diretivas para o reúso de modelos e defende que isso possibilita a diminuição de esforços, além de possibilitar uma maior consistência nos modelos gerados, por ajudar a evitar ambiguidades e omissões tão comuns no levantamento de requisitos.

6.2. PERSPECTIVAS FUTURAS

Diante dos problemas citados na seção anterior, alguns trabalhos podem ainda ser realizados para complementar o trabalho aqui apresentado.

No que se refere ao perfil E-OntoUML, outros trabalhos podem ser realizados, estendendo-o de forma que ele seja capaz de representar mais conceitos das partes B e C da UFO que sejam relevantes para a representação de ontologia de tarefas. Além dos elementos definidos no perfil, tentou-se representar outros conceitos da UFO por meio de elementos de modelo da UML, porém não se obtiveram conclusões maduras o suficiente para serem acrescentadas a este trabalho. A saber, foram estudadas formas de representar eventos não intencionais, objetivos, pré e pós-condições, algumas relações temporais entre eventos e exceções. Essas análises devem ser retomadas e deve-se, também, considerar os conceitos restantes da UFO e os diversos elementos de modelos para diagramas de atividades que ainda não foram explorados neste estudo inicial.

Dentre esses aspectos pendentes, a representação de objetivos merece grande atenção. Toda tarefa é realizada para atingir um certo objetivo e esse fato tem impacto muito grande na modelagem de tarefas. Diferentes intenções requerem formas variadas de executar uma mesma tarefa. Logo, questões de variabilidade e intencionalidade de tarefas devem ser

investigadas, assim como seus impactos no reúso de tarefas. É importante, ainda, verificar se a Modelagem de Objetivos (BRESCIANI et al., 2004) pode ajudar em aspectos de modelagem de objetivos de tarefas.

É importante, também, sistematizar as abordagens apresentadas, tanto a abordagem de representação quanto de reutilização do conhecimento de tarefa. Isso pode ser feito com a definição mais explícita de passos, regras e heurísticas. Porém antes de sistematizá-las é importante realizar estudos experimentais com o intuito de melhor avaliar as abordagens propostas. Com a utilização das mesmas em situações reais e mais complexas, novas correspondências podem ser encontradas e possíveis problemas identificados e resolvidos. Nesse sentido pode ser útil considerar, num primeiro momento, os modelos de CommonKADS (BREUKER; VAN DE VELDE, 1994), por se tratarem de modelos já utilizados com sucesso em outras abordagens.

Tendo uma abordagem mais sistemática, ainda se poderia considerar a construção de ferramentas para tratar uma abordagem dirigida a modelos (*Model Driven Development* – MDD) (MELLOR et al., 2004) para construção de software a partir de ontologias de domínio, de tarefas e de classes de aplicação que sejam fundamentadas na ontologia de fundamentação UFO. Assim poder-se-iam gerar modelos comportamentais e estruturais de sistemas a partir de ontologias.

Há, ainda, outra linha de pesquisa interessante que é o estudo de como o perfil proposto pode apoiar a integração de sistemas. A partir de ontologias de domínio, de tarefa e de classe de aplicação, pode ser definida uma abordagem de integração de processos e de sistemas, visando a uma integração não apenas no nível de dados, como ocorre na integração por meio de ontologias de domínio, mas também no nível de controle.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBERTS, L. K., **YMIR: an Ontology for Engineering Design**. University of Twente, 1993.

ALLEN, J.F. **Maintaining Knowledge about Temporal Intervals.** Communications of the ACM, v. 26, issue 11, pp. 832-843, Novembro, 1983.

ALMEIDA, J. P. A., GUIZZARDI, G., SANTOS , P. S., **Applying and Extending a Semantic Foundation for Role-Related Concepts in Enterprise Modelling,** Enterprise Information System, (aprovado, em publicação), 2009.

ARANGO, G. **A brief introduction to domain analysis**. In Proceedings of the 1994 ACM Symposium on Applied Computing, Phoenix, Arizona, United States, March 06 - 08, 1994.

BOOCH, G., RUMBAUGH, J. , JACOBSON, I. **Unified Modeling Language User Guide**, 2nd Edition, Addison-Wesley Professional, 2005.

BOTTAZZI, E.; FERRARIO, R. **Preliminaries to a DOLCE Ontology of Organizations**, International Journal of Business Process Integration and Management, 2006.

BREUKER, J.; VAN DE VELDE, W. **CommonKADS Library for Expertise Modelling**. IOS Press. Amsterdam, Netherlands, 1994.

BRESCIANI, P., GIORGINI, P., GIUNCHIGLIA, F., MYLOPOULOS, J., PERINI, A. **Tropos: An Agent-Oriented Software Development Methodology**. International Journal of Autonomous Agents and Multi Agent Systems, v. 8, nº. 3, pp. 203–236, 2004.

BRUAUX, S., KASSEL, G., MOREL, G.: **A Clarification of the Ontological Status of "Knowledge Roles".** 18th International Conference on Database and Expert Systems Applications, DEXA´2007. DEXA Workshops, 2007, pp. 529-533.

CARDOSO, E. C. S., ALMEIDA, J. P. A., GUIZZARDI, G. **Uma Experiência com Engenharia de Requisitos baseada em Modelos de Processos**, Proceedings of the XI Ibero-American Workshop on Requirements Engineering and Software Environments, Recife, PE, Brazil, 2008.

CHANDRASEKARAN, B. **Design Problem Solving: A Task Analysis,** AI Magazine, v. 11, issue 4, pp.59-71, 1990**.**

CHANDRASEKARAN, B., JOSEPHSON, J.R., BENJAMINS, R. **The Ontology of Tasks and Methods**, In Proceedings of the 11th Knowledge Acquisition Modeling and Management Workshop, KAW'98, Banff, Canada, April 1998.

CHANDRASEKARAN, B. ,JOSEHSON, J. R., BENJAMINS, V. R. **What Are Ontologies, and Why Do We Need Then?** IEEE Intelligent Systems, v. 14 nº 1, pp. 20-26, 1999.

CHUANG, T., FANG, K., **Ontology-Based for Requirement Analyze and Design-Knowledge Network Learning Diagnosis Analysis System**. In Proceedings of the 2007 international Conference on Convergence information Technology. (ICCIT). IEEE Computer Society, Washington, November, 2007,

CHRISTEL, M.G.; KANG, K.C. **Issues in Requirements Elicitation**. Software Engineering Institute, 1992. Apud: PRESSMAN, R. S. Engenharia de Software. 5ª. ed., Rio de Janeiro: McGrawHill, 2002.

COTA, R.I., MENEZES, C.S., FALBO, R.A. **Modelagem Organizacional Utilizando Ontologias e Padrões de Análise** Memorias de VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software - IDEAS'2004, pp. 56-67, Arequipa, Perú, Maio 2004.

CRANEFIELD, S. PURVIS, M., **UML as an Ontology Modelling Language**, In: Proceedings of the IJCAI-99, Workshop on Intelligent Information, 16th International Joint Conference on AI, Stockholm, Sweden, July 1999.

CRUBÉZY, M., MUSEN, M. A., **Ontologies in Support of Problem Solving**, Handbook on Ontologies, pp. 321-342, Springer, 2003.

CRUZ, P.O.S.: **Heurísticas para Identificação de Requisitos de Sistemas de Informações a partir de Modelos de Processos**, Tese de Mestrado, UFRJ, Rio de Janeiro, RJ, 2004.

CRUZ, G. G., GOMES, A. G., CASTRO, J. B. **Mapeando Diagramas da Teoria da Atividade em Modelos Organizacionais Baseados em i*,** In Proceedings of the VII Workshop on Requirements Engineering (WER 2004), pp. 39-50, Tandil, Argentina, 2004.

DRAGAN, G**.**, DRAGAN, D., VLADAN, D. **Model Driven Architecture and Ontology Development**, Springer, 2006.

DEVEDZIC, V. **Software Patterns**. In S. K. Chang (Ed.) Handbook of Software Engineering and Knowledge Engineering v. 2 , pp. 645-671, Singapore, 2002.

DING, Y. **A review of ontologies with the Semantic Web in view**, Journal of Information Science, v. 27, n°. 6. pp. 377-384, 2001.

ERICKSON, J., **A decade and more of UML: An overview of UML semantic and structural issues and UML field use**. Journal of Database Management, v. 19, pp. i-vii, EUA, 2008.

ERIKSSON, H. E.; PENKER, M., **Business Modeling with UML: Business Patterns at Work**. New York: Wiley Publishers, 2000.

ESTRADA, H., MARTÍNEZ, A., PASTOR, O., SÁNCHEZ, J. **Generación de Especificaciones de Requisitos de Software a partor de Modelos de Negocios: Un Enfoque**. Anais do Workshop em Engenharia de Requisitos, pp. 177-193 Valencia, Espanha, Novembro 11-12, 2002.

FALBO, R.A. **Experiences in Using a Method for Building Domain Ontologies**. Proc. of the 16th International Conference on Software Engineering and Knowledge Engineering, International Workshop on Ontology In Action. Banff, Canada, 2004.

FALBO, R.A. ; MARTINS, A.F. ; SEGRINI, B. M. ; BAIOCO, G. ; MORO, R. D. ; NARDI, J.C. . **Um Processo de Engenharia de Requisitos Baseado em Reutilização de Ontologias e Padrões de Análise**. In: Jornada Iberoamericana de Engeniería del Software e Engeniería del Conocimiento, JIISIC, Lima, 2007.

FALBO, R. A. ; NARDI, J. C. . **Evolving a Software Requirements Ontology**. In: XXXIV Conferencia Latinoamericana de Informática - CLEI´2008, 2008, Santa Fe. Anales de XXXIV Conferencia Latinoamericana de Informática, pp. 300-309, 2008.

FENSEL, D. et al. **The Unified Problem-solving Method Development Language UPML**. Knowledge and Information Systems Journal (KAIS), v.5, nº 1, pp. 83-131, 2003.

FOWLER, M. **Analysis Patterns: Reusable Object Models**. Addison-Wesley Professional Computing Series, 1997.

GENNARI, J. H., TU, S. W., ROTHENFLUH, T. E., AND MUSEN, M. A. **Mapping Domains to Methods in Support of Reuse**. International Journal of Human Computer Studies, v. 41, pp. 399-424, 1994.

GIMENES, I.M.S., HUZITA, E.H.M. **Desenvolvimento Baseado em Componentes: Conceitos e Técnicas**. Editora Ciência Moderna, 2005.

GÓMEZ-PÉREZ, A., FERNÁNDEZ-LÓPEZ, M. CORCHO, O., **Ontological Engineering**, Springer-Verlag, 2004.

GÓMEZ-PÉREZ, A., BENJAMINS, V.R. **Overview of knowledge sharing and reuse components: ontologies and problem-solving methods**. In: Proceedings of IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends, in conjunction with the Sixteenth International Joint Conference on Artificial Intelligence, August, Stockholm, Sweden, 1999.

GRUBER, T.R. **A translation approach to portable ontology specifications**, Knowledge Acquisition, v. 5, pp 199–220, 1993.

GRUBER, T. **Toward Principles for the Design of Ontologies Used for Knowledge Sharing**. International Journal of Human-Computer Studies, v. 43, nº 5/6, pp. 907-928, 1995.

GUARINO, N. **Formal Ontology and Information Systems**. In: Formal Ontologies in Information Systems, N. Guarino (Ed.), IOS Press, pp. 3 -15, 1998.

GUARINO, N. **Understanding, building and using ontologies.** International Journal of Human-Computer Studies, v. 46,nº 2-3, pp. 293–310, 1997.

GUARINO, N., GIARETTA, P. **Ontologies and Knowledge Bases: Towards a Terminological Clarification**. In N. Mars (ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing 1995. IOS Press, Amsterdam, pp. 25-32, 1995.

GUIZZARDI, G. **Uma Abordagem Metodológica de Desenvolvimento para e com Reuso Baseada em Ontologias Formais de Domínio**, Dissertação de Mestrado, Departamento de Ciência da Computação, Universidade Federal do Espírito Santo, 2000.

GUIZZARDI, G. **Ontological Foundations for Structural Conceptual Models**, Universal Press, The Netherlands, 2005.

GUIZZARDI, G., **On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models**, Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV, Olegas Vasilecas, Johan Edler, Albertas Caplinskas (Editors), IOS Press, Amsterdam, 2007.

GUIZZARDI, G., FALBO, R.A., GUIZZARDI, R.S.S. **Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology**, Anais do XI Worshop Iberoamericano de Ambientes de Software e Engenharia de Requisitos, Recife, Brazil, 2008a.

GUIZZARDI, G., FALBO, R.A., GUIZZARDI, R.S.S. **A importância de Ontologias de Fundamentação para a Engenharia de Ontologias de Domínio: o caso do domínio de Processos de Software**, Revista IEEE América Latina, v. 6, p. 244-251, 2008b.

GUIZZARDI, G.; WAGNER, G. **Some Applications of a Unified Foundational Ontology in Business Modeling**. Ontologies and Business Systems Analysis, Michael Rosemann and Peter Green (Eds.). IDEA Publisher, 2005.

GUIZZARDI, R.S.S. **Agent-oriented constructivist knowledge management**. PhD thesis, University of Twente. The Netherland, 2006.

HERRE, H., HELLER, B., BUREK, P., HOEHNDORF, R., LOEBE, F., MICHALEK, H. **General Formal Ontology (GFO) a foundational ontology integrating objects and**

processes, Part 1: Basic Principals, version 1.0.1**,** Onto-Med Report 8, University of Leipzig, 2006.

IEEE-SA STANDARDS BOARD IEEE Std 1233-1998: **IEEE Guide for Developing System Requirements Specifications**. Dezembro de 1998.

IKEDA, M, SETA, K., KAKUSHO, O., MIZOGUCHI, R. **Task ontology: ontology for building conceptual problem solving models**. Proceedings of ECAI98 Workshop on Applications of ontologies and problem-solving models, pp. 126-133, 1998.

KAIYA, H., SAEKI, M. **Using Domain Ontology as Domain Knowledge for Requirements Elicitation**, pp.189-198, 14th IEEE International Requirements Engineering Conference (RE'06), 2006.

KAPTELININ, V. **Activity Theory: Implications for Human-Computer Interaction.** In Context and Consciousness - Activity Theory and Human- Computer Interaction, MIT Press, pp. 104-116, 1996.

KNIGHT, D.M. **Elicitação de Requisitos de Software a Partir do Modelo de Negócio**, Dissertação de Mestrado em Informática, Instituto de Matemática e Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ – Brasil,Abril de 2004

KOTONYA, G.; SOMMERVILLE, I. **Requirements Engineering: Process and Techniques**. 1º Edição. Editora Wiley. 1998.

LENAT, D., GUHA, R.V. **Building Large Knowledge based Systems: Representation and Inference in the CYC Project,**Addison Wesley, Reading, Massachusetts, 1990.

LIU, W., FANG, K. **Using IDEF0/Petri Net for Ontology-Based Task Knowledge Analysis: The Case of Emergency Response for Debris-Flow**, hicss, vol. 4, pp.76c, Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06) Track 4, 2006.

LU, S., PARIS, C., LINDEN, K. V. **Toward the Automatic Construction of Task Models from Object-Oriented Diagrams**, Proceedings of the IFIP, Seventh Working Conference on Engineering for Human-Computer Interaction, v. 150, p. 169 – 189, Deventer, The Netherlands, 1998.

LU, S., PARIS, C., LINDEN, K. V., COLINEAU, N., **Generating UML Diagrams From Task Models**; In the proceedings of the Fourth Annual International Conference of the New Zealand chapter of the ACM's SIGCHI, CHINZ'03, pp 9-14, Dunedin, New Zealand, July 3-4 2003.

MARTINS, L. E. G. **Uma Metodologia de Elicitação de Requisitos de Software Baseada na Teoria da Atividade**. Dissertação de Doutorado, Unicamp, Campinas, Agosto, 2001.

MARTINS, A. F., FALBO, R. A. **Models for Representing Task Ontologies**. Proceedings of the 3rd Workshop on Ontologies and their Applications (WONTO'2008), Salvador, Brasil, October 2008.

MCGUINNESS, D. L. **Ontologies Come of Age**. In Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press, 2003.

MELLOR, S. J., SCOTT, K., UHL, A., WEISE D. **MDA Distilled**. Addison Weseley Professional, 2004.

MIAN, P.G.; FALBO, R.A. **Supporting Ontology Development with ODEd**, Journal of the Brazilian Computer Science, vol. 9, nº. 2, pp. 57-76, November, 2003.

MIZOGUCHI, R. TIJERINO, Y., IKEDA, M., **Task Analysis Interview Based on Task Ontology**, Expert Systems with Applications, v. 9, no. 1, pp. 15-25, 1995a.

MIZOGUCHI, R., VANWELKENHUYSEN, J., IKEDA, M. **Task Ontology for Reuse of Problem Solving Knowledge**. In Building and Knowledge Sharing (2nd International Conference on Very Large-Scale Knowledge Bases), Enschede, The Netherlands, pp. 46–59, 1995b.

MONTABERT, C., BUSSERT, D., GIFFORD, S., CHEWAR, C. M., MCCRICKAR D. S. **Supporting Requirements Reuse in Notification Systems Design Through Task Modeling**. In Proceedings of the 11th International Conference on Human-Computer Interaction (HCII '05), Las Vegas NV, July 2005.

MOTTA, E., LU, W.: **A Library of Components for Classification Problem Solving**. In: Proceedings of Pacific Rim Knowledge Acquisition Workshop (PKAW 2000), Sydney, Australia, December 2000.

NARDI, J.C. **Apoio de Gerência de Conhecimento à Engenharia de Requisitos em um Ambiente de Desenvolvimento de Software**, Dissertação de Mestrado, UFES, 2006.

NILES, I., PEASE, A. **Towards a standard upper ontology**. In Proceedings of the international Conference on Formal ontology in information Systems (FOIS '01), Ogunquit, Maine, USA, 2001.

NOY, N. F., MCGUINNESS, D. L. **Ontology Development 101: A Guide to Creating Your First Ontology**. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.

O'LEARY, D. E. **Using AI in Knowledge Management: Knowledge Bases and Ontologies**. IEEE Intelligent Systems, v. 13, n. 3, p. 34-39, May, 1998.

OLIVÉ, A. **Conceptual Modeling of Information Systems.** Springer-Verlag New York, Inc, 2007.

OMG, **Business Process Modeling Notation Specification**, Final Adopted Specification, February, 2006.

OMG, **Unified Modeling Language (OMG UML)**, Superstructure, V2.1.2 Meta-modelo da UML, November., 2007.

PFLEEGER, S. L. **Software Engineering – Theory and Practice**. Saddle River, New Jersey: Prentice Hall, Upper, 1998.

POMPILHO, S. **Análise Essencial: Guia Prático de Análise de Sistemas.** IBPI Press, Editora Infobook, Rio de Janeiro, 1995.

PRIETO-DIAZ, R.; **Status Report: Software Reusability**; Software, IEEE v. 10, Issue 3, pp. 61 – 66, May 1993.

PRESSMAN, R. S. **Engenharia de Software**. 5ª. ed. Rio de Janeiro: McGrawHill, 2002.

RAJPATHAK, D., MOTTA, E., ROY, R., **A Generic Task Ontology for Scheduling Applications**. In Proceedings of the International Conference on Artificial Intelligence (IC-AI'2001), Nevada, Las Vegas, USA, 2001.

RAJPATHAK, D., MOTTA. E. **An Ontological Formalization of the Planning Task**, Proceedings of the International Conference on Formal Ontologies in Information Systems (FOIS'04), pp. 305-316, Torino, Italy, 2004.

ROBERTSON, S. R.; ROBERTSON, J. **Mastering the Requirements Process**. Addison-Wesley, 1999.

RUSSELL, N., VAN DER AALST, W.M.P., TER HOFSTEDE, A.H.M., WOHED, P. **On the suitability of UML 2.0 activity diagrams for business process modeling,** Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modeling, Hobart, Australia, pp. 95-104, 2006.

SCHNEIDER, G., WINTERS, J., **Applying Use Cases – A Practical Guide**, Addison-Wesley, 1998.

SCHREIBER, A. T., WIELINGA, B. J., DE HOOG, R., AKKERMANS, J. M., VAN DE VELDE , W. **CommonKADS: A comprehensive methodology for KBS development**. IEEE Expert, v. 9, issue 6, p. 28-37, 1994.

SCHREIBER, G., WIELINGA, B., JANSWEIJER, W. **The KAKTUS Viewon the 'O' Word**. In Proceedings of IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada.v, 1995.

SOMMERVILLE, I. **Engenharia de Software**. 6a. ed. São Paulo, SP: Addison-Wesley, 2003.

STÖRRLE, H., HAUSMANN J. **Towards a Formal Semantics of UML 2.0 Activities**, In Proceedings German Software Engineering Conference, v. P-64 of LNI, pp. 117-128, 2005.

TAVARES, D.B., **Procedimento de análise para validação de diagrama de classes de domínio baseado em análise ontológica,** Dissertação de Mestrado em Ciência da Computação, Universidade Federal de Viçosa, Viçosa, MG, 2008.

TRAN, V. X., TSUJI, H, **OWL-T: An Ontology-based Task Template Language for Modeling Business Processes**. Fifth International Conference on Software Engineering Research, Management and Applications, SERA´2007, pp. 101-108, 2007.

ULLMANN, S. **Semantics: An Introduction to the Science of Meaning**., Basil Blackwell, Oxford, 1972 apud GUIZZARDI, G. Ontological Foundations for Structural Conceptual Models, Universal Press, The Netherlands, 2005.

USCHOLD, M. **Building ontologies: towards a unified methodology**. In: Proceedings of Expert Systems'96, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK, pp. 16–18 December 1996.

VALERIO, A., SUCCI, G., AND FENAROLI, M. **Domain analysis and framework-based software development**. *SIGAPP Appl. Comput. Rev.* 5, 2 ,Sep. 1997.

VAN HEIJST, G., SCHREIBER, A. T., WIELINGA, B. J. **Using Explicit Ontologies in KBS Development**. International Journal of Human and Computer Studies, v. 46, pp. 183-292, 1997.

VAN LAMSWEERDE, A**. Requirements engineering in the year 00: A research perspective**. In Proceedings 22nd International Conference on Software Engineering, Invited Paper, ACM Press, June 2000.

VAN WELIE, M., VAN DER VEER, G. C., ELIËNS, A. **An Ontology for Task World Models**, Proceedings of DSV-IS98, Abingdon, UK, Springer-Verlag, pp. 3-5, 1998.

WANG, X., CHAN, C.W., **Ontology Modeling in UML**. In Proceedings of OOIS 2001, Calgary, Canada, Springer-Verlag, 59-68, 2001.

WHITE, S. A. **Process Modeling Notations and Workflow Patterns**, BPTrends, March 2004.

WIEGERS, K.E. **Software Requirements**. 2nd ed. Redmond, Washington: Microsoft Press, 2003.

WIELINGA, B. J. AND SCHREIBER, A. T. **Reusable and sharable knowledge bases: a European perspective**. In Proceedings of Proceedings of First International Conference on Building and Sharing of Very Large-Scaled Knowledge Bases. Tokyo, Japan Information Processing Development Center, 1993.

YU, E. S., MYLOPOULOS, J. **An Actor Dependency Model of Organizational Work – With Application to Business Process Reengineering**. In Proceedings of the Conference on Organizational Computing Systems (COOCS'93), pp. 258-268, Califórnia, United States, 1993.

ZAMBORLINI, V. C. **Implementação de uma Ontologia de Referência de Eletrocardiograma com Análise da Perda de Expressividade**, Projeto Final de Graduação, Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, Espírito Santo, fevereiro, 2008.

ZLOT, F., **Conhecimento de Tarefa em Ambientes de desenvolvimento de Software Orientados a Domínio**, Dissertação de mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, junho, 2002.

ZLOT, F., OLIVEIRA, K. M., ROCHA, A.R., **Modeling Task Knowledge to Support Software Development**, Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'2002), pp. 35-42, Ischia, Italy, 2002.

ZONG-YONG, L., ZHI-XUE, W., YING-YING, L., YUE, W., YING, L., **Towards a Multiple Ontology Framework for Requirements Elicitation and Reuse**, Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), v. 1, pp. 189-195 Beijin, China, 2007.