UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
MESTRADO EM INFORMÁTICA

RODRIGO FERNANDES CALHAU

UMA ABORDAGEM BASEADA EM ONTOLOGIAS PARA INTEGRAÇÃO SEMÂNTICA DE SISTEMAS

VITÓRIA, AGOSTO 2011

RODRIGO FERNANDES CALHAU

UMA ABORDAGEM BASEADA EM ONTOLOGIAS PARA INTEGRAÇÃO SEMÂNTICA DE SISTEMAS

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, AGOSTO 2011

RODRIGO FERNANDES CALHAU

UMA ABORDAGEM BASEADA EM ONTOLOGIAS PARA INTEGRAÇÃO SEMÂNTICA DE SISTEMAS

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 26 de agosto de 2011.

COMISSÃO EXAMINADORA

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

_____
Prof. Crediné Silva de Menezes, D.Sc.
Universidade Federal do Espírito Santo (UFES)

_____
Prof. Juliano Lopes de Oliveira, D.Sc.
Universidade Federal de Goiás (UFG)

VITÓRIA, AGOSTO 2011

DEDICATÓRIA

Dedico esta dissertação aos meus pais, Geraldo e Marta.

AGRADECIMENTOS

Agradeço a **Deus**, pela Vida,
Aos meus **antepassados**, pela vida,
Aos meus **pais** pela vida e amor.
Aos meus **irmãos**, amo vocês!
À Jerdda pela companhia,
Aos **amigos** do laboratório NEMO, em especial a Ana e Veruska,
Aos meus amigos da *Seicho-No-Ie*.
Ao mestre Masaharu Taniguch, pelo ensinamento divino,
Aos **professores** que colaboram com minha formação,
Em especial ao meu orientador **Ricardo Falbo** (e seu reservatório infinito de paciência) por todo apoio nesses 5 anos de orientação,

Agradeço também a **CAPES** (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior)**,** pelo apoio financeiro, viabilizado por meio de uma bolsa de mestrado**.**

Essa dissertação é Nossa!

Todas as vidas são ramificações da mesma Vida. Portanto, na essência, tudo já está integrado por natureza. Basta a nós manifestarmos concretamente essa unidade inerente a tudo.

RESUMO

Geralmente, sistemas são construídos por diferentes desenvolvedores, em diferentes momentos, sem uma preocupação com o estabelecimento de um significado comum aos itens comunicados pelos mesmos. Em geral, cada sistema é executado separadamente e implementa o seu próprio modelo de dados e de processo. Esses modelos não são compartilhados entre os sistemas, o que abre espaço para diversos tipos de conflitos, incluindo conflitos sintáticos e semânticos. Essa heterogeneidade é considerada uma das maiores dificuldades no problema da integração. Neste contexto, ontologias podem ser usadas como uma interlíngua para mapear conceitos e serviços usados por diferentes sistemas, que acessariam dados e serviços por meio de ontologias compartilhadas.

Neste presente trabalho defende-se a ideia de que a integração semântica é uma tarefa complexa e bastante subjetiva e, por isso, deve ocorrer em um nível mais alto de abstração. Tomando por base esta consideração, foi desenvolvida OBA-SI (*Ontology-Based Approach for Semantic Integration*), uma abordagem de integração semântica de sistemas que concentra esforços na modelagem conceitual e na análise dos requisitos de integração. Nessa abordagem, a integração semântica é realizada em um alto nível de abstração, provendo acordo semântico entre os sistemas no nível conceitual. OBA-SI lida com a integração nas três camadas de integração: dados, serviços e processos. Para tal, modelos conceituais dos sistemas (representando sua estrutura e comportamento), bem como do processo de negócio por eles apoiado, são comparados à luz de ontologias, usadas para atribuir semântica aos itens compartilhados entre os sistemas no apoio ao processo de negócio considerado. Os modelos são compatibilizados por meio de mapeamentos entre seus elementos. Todo esse processo de atribuição de semântica e uso de ontologias é independente da solução da integração. A fim de se testar a presente abordagem, foi realizado um estudo de caso no qual ela foi aplicada na integração semântica de dois sistemas de Gerência de Configuração de Software, usando ontologias de domínio e de tarefa.

Palavras-chave: Integração Semântica de Sistemas, Interoperabilidade Semântica, Ontologias, Gerência de Configuração de Software.

ABSTRACT

In order to properly support the whole organization's business process, systems should be integrated. However, integration is a complex problem, especially for large and dynamic organizations. In general, each system runs independently and implements its own data and process models. These models are not shared between systems, leading to several conflicts, including technical, syntactical and, especially, semantic conflicts. This heterogeneity is considered one of the major difficulties of the integration problem. In this context, ontologies can be used as an interlingua to map concepts and services used by the different enterprise applications, in a scenario of access to data and services via a shared ontology.

In this work we present an Ontology-Based Approach for Semantic Integration (OBA-SI) that concentrates efforts in the integration requirements analysis and modeling. In this approach, integration is addressed in a high level of abstraction, looking for getting semantic agreement between the systems at the conceptual level. A premise of the proposed approach is that the assignment of semantics and the use of ontologies must be independent of the integration solution itself. The proposed approach deals with integration in three layers: data, service and process. To accomplish this, conceptual and behavioral models of the systems to be integrated are compared in the light of a reference ontology, which is responsible for assigning semantics to the shared items of the systems. Thus, to illustrate our approach, we present a real case of its application in integrating the version control system Subversion and a system for controlling changes, in order to support a Software Configuration Management process.

Keywords: Semantic System Integration, Ontologies, Semantic Interoperability, Software Configuration Management.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1 - Introdução

Atualmente, a integração de sistemas é apontada como um problema complexo para a maioria das organizações. Esse problema se amplifica na medida em que, cada vez mais, surge a necessidade de os sistemas trabalharem em conjunto (IZZA, 2009). É comum organizações usarem diferentes sistemas para apoiar seus processos de negócio, sendo que, geralmente, cada uma desses sistemas é desenvolvido por fabricantes diferentes. A integração de sistemas em uma organização pode trazer inúmeros benefícios para a mesma, mas para disso, muitas vezes, cada sistema precisa compartilhar funcionalidades e dados com outros sistemas (POKRAEV, 2009).

Entretanto, os sistemas geralmente não são construídos com intuito de conversarem entre si. Na maioria das vezes, são construídos isoladamente, em diferentes contextos, sem compartilharem um significado comum dos itens manipulados por eles. Além de executarem separadamente, os sistemas geralmente implementam modelos de dados distintos e que não são compartilhados, o que abre espaço para conflitos nos níveis sintático e semântico. Essa heterogeneidade entre os sistemas a serem integrados é considerada uma das maiores dificuldades no problema da integração (IZZA, 2009).

Sistemas são construídos por pessoas e procuram ser uma representação da realidade. Eles embutem uma conceituação que pode variar em função das pessoas envolvidas na sua construção, de seus requisitos ou até mesmo em função de aspectos mais subjetivos como, por exemplo, a cultura ou a cognição humana.

Um fator chave para a integração semântica é ter sistemas compartilhando um entendimento comum do significado dos termos e serviços manipulados pelos sistemas. Para tal, deve-se ter uma representação compartilhada de dados e tarefas do universo de discurso de interesse. Neste contexto, ontologias podem ser usadas como uma interlíngua para mapear conceitos e serviços usados por diferentes sistemas, que acessariam dados e serviços por meio de ontologias compartilhadas.

A utilização de semântica na integração de sistemas tem sido apontada como estratégia promissora (IZZA, 2009). Através da atribuição de semântica aos modelos dos sistemas, é possível atribuir significado aos mesmos e, assim, compatibilizá-los de acordo com esse significado. Deste modo, os sistemas podem conversar entre si sem que haja “falsos acordos”. Muitas vezes, acha-se que os sistemas estão em concordância com o

significado das coisas, estabelecendo assim um falso acordo. Se um falso acordo não for descoberto no início do projeto de integração, provavelmente a implementação da integração será incorreta (POKRAEV, 2009).

A integração de sistemas como um todo é uma tarefa bastante complexa, envolvendo diferentes tipos de preocupações. A solução de integração não envolve apenas a integração de conceituações de sistemas por meio de uma conceituação comum. Ela, como um tipo de produto de software, possui requisitos que devem ser levantados, analisados e satisfeitos. Além disso, a solução de integração deve ser projetada, implementada, testada e implantada. Isso sem falar que, principalmente em domínios mais complexos, pode ser que os próprios envolvidos na integração possuam visões diferentes do mesmo domínio e das tarefas sendo apoiadas.

Com a construção isolada de sistemas, os mesmos não compartilham modelos de dados e de processos comuns. Em decorrência disso, ao integrar os sistemas, conflitos semânticos podem ocorrer. O objetivo da integração semântica é solucionar tais conflitos semânticos decorrentes da heterogeneidade dos sistemas.

Podem-se dividir os problemas de integração semântica em dois tipos: (i) conflitos de representação e (ii) conflitos de conceituação. Com relação aos conflitos de representação, busca-se identificar símbolos que possuem o mesmo significado, mesmo sendo distintos. Por exemplo, os símbolos *Empregado* e *Funcionário*, pertencentes a dois sistemas a serem integrados, são diferentes, mas representam uma mesma conceituação, a de uma pessoa que possui um contrato de emprego com a empresa. Já com relação ao segundo problema busca-se compatibilizar as diferentes visões da realidade, identificando conceitos que são equivalentes e representam a mesma entidade no mundo real. Por exemplo, o conceito de funcionário pode variar de um sistema para o outro. Em um sistema ele pode ser uma pessoa que possui contrato de emprego com uma empresa e em outro pode ser uma pessoa que recebe salário de uma empresa (POKRAEV, 2009).

Para que ocorra com qualidade, a integração não deve ser feita de maneira *ad-hoc*. É importante a adoção de uma abordagem que ajude a diminuir a complexidade dessa tarefa, fornecendo, entre outros, etapas bem estabelecidas, separação de preocupações, diminuição da subjetividade e redução de problemas de entendimento e comunicação. Este trabalho visa ajudar a minimizar alguns dos problemas inerentes à integração

semântica. Ele procura atacar, sobretudo, o segundo tipo de problema de integração semântica citado anteriormente, que decorre das diferenças de conceituação da realidade.

1.1. Objetivos do Trabalho

O objetivo geral deste trabalho é desenvolver uma abordagem de integração semântica que apoie a tarefa de integração semântica de sistemas que foram construídos isoladamente e que possuem diferentes conceituações, buscando solucionar o problema de heterogeneidade semântica existente entre eles e ajudando, sobretudo, na identificação e solução de conflitos semânticos.

Um dos objetivos da integração semântica é compatibilizar as conceituações de diferentes sistemas, o que por si só é uma tarefa subjetiva. Por meio de representações explícitas de tais conceituações, a subjetividade inerente a essa tarefa pode ser diminuída consideravelmente. Dessa forma, é importante que uma abordagem de integração semântica se baseie na explicitação do conhecimento tácito relativo aos sistemas, ao processo, bem como ao próprio domínio e tarefas envolvidos. Uma vez explicitado, é possível uma análise de conceituações, um melhor entendimento delas, bem como uma melhor comunicação entre os envolvidos, ajudando no estabelecimento de um consenso ou na identificação de conflitos semânticos.

O intuito da abordagem proposta não é de explicitar apenas as conceituações dos sistemas a serem compatibilizadas, mas também representar de modo adequado as relações existentes entre tais conceituações. Assim, um objetivo específico deste trabalho é relacionar as conceituações envolvidas de modo explícito, promovendo discussões entre as partes envolvidas e um maior entendimento da solução da integração.

Um ponto importante na integração de um conjunto de sistemas refere-se ao conhecimento relativo aos domínios e tarefas envolvidos, bem como a representação desse conhecimento e reutilização do mesmo em iniciativas de integração que ocorram no mesmo domínio ou envolvendo as mesmas tarefas. A abordagem parte da premissa que, para que a conceituação dos sistemas seja compatibilizada, é importante o uso de conceituações gerais referentes ao domínio e às tarefas envolvidas.

Existem diversas abordagens de integração semântica, entretanto muitas delas focam muita atenção na solução de aspectos tecnológicos da integração semântica como, por exemplo, aplicações em SOA (*Service-Oriented Architecture)*, soluções em BPM (*Business Process Management)* ou em EAI (*Enterprise Application Integration)*. Porém a solução da integração semântica está relacionada com a compatibilização das conceituações dos sistemas envolvidos e, por isso, independe de aspectos tecnológicos ou soluções específicas de integração. Pelo contrário, uma das premissas deste trabalho é que a integração semântica ocorra primeiramente e isoladamente em um alto nível de abstração, de modo independente do projeto e implementação da solução de integração.

A separação da integração semântica de seu projeto e implementação é um modo inclusive de diminuir a complexidade de tal tarefa, uma vez que ocorre uma separação de preocupações. Outra maneira de diminuir essa complexidade é seguir um processo de integração bem estabelecido para a realização da integração. Por meio dele, a subjetividade da tarefa é diminuída em função da objetividade fornecida pelo processo de integração. Este trabalho tem como objetivo específico propor um processo de integração semântica com o intuito de facilitar essa tarefa.

Por fim, outro objetivo específico deste trabalho é envolver a integração semântica nas três camadas de integração: dados, serviço e processo (IZZA, 2009). Essa divisão em camadas não é obrigatória para se efetuar a integração, mas é apenas uma maneira de enxergar a integração de modo a facilitar sua realização. A integração semântica na camada de dados se preocupa com o significado dos tipos de entidades compartilhados entre os sistemas, ou seja, se eles representam o mesmo tipo de entidade no mundo real. A integração semântica na camada de serviço se preocupa com o significado dos serviços compartilhados entre os sistemas. Por fim, a integração semântica na camada de processo se preocupa com o significado atribuído aos elementos envolvidos no apoio ao processo oferecido pelos sistemas.

Muitos trabalhos possuem como foco a integração semântica de dados, que relativamente é a mais explorada na literatura, como, por exemplo, os referenciados em (PARK; RAM, 2004), (MAIER; AGUADO; BERNARAS, 2003) e (WACHE, 2001), que são da área de integração semântica em banco de dados, além de abordagens baseadas em tecnologias de web semântica como RDF e OWL. Entretanto é importante que haja um compartilhamento da semântica de elementos comportamentais, tais como serviços, insumos, produtos, tarefas e atores, para que não ocorram conflitos semânticos

envolvendo tais elementos. Um sistema, por exemplo, pode consumir um serviço de outro sistema que faz uma coisa diferente do que realmente se espera (conflito na integração de serviço). Ou ainda, um sistema pode ser usado para apoiar um processo por meio de seus serviços, mas na verdade realiza algo diferente do que se espera (conflito na integração de processo). Para que não ocorram tais tipos de conflitos semânticos, este trabalho tem por objeto usar conhecimento de domínio para ajudar a integração semântica na camada de dados e também usar conhecimento a respeito de tarefas para apoiar a integração semântica nas camadas de serviço e processo.

Para demonstrar a aplicação da abordagem proposta, uma iniciativa de integração é realizada, envolvendo a integração de dois sistemas de apoio à Gerência de Configuração de Software: Subversion (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2011) e GCS-ODE (CALHAU, 2009). Assim, também é um objetivo específico deste trabalho realizar a integração semântica, no nível conceitual, desses sistemas.

1.2. Contexto e Histórico do Trabalho

A necessidade de desenvolver uma abordagem de integração semântica teve origem na realização de uma iniciativa de integração de sistemas de apoio à Gerência de Configuração de Software (GCS), realizada em (CALHAU, 2009). Esse trabalho ocorreu no contexto do ambiente ODE (FALBO et al., 2005), no qual diversas ferramentas de apoio ao desenvolvimento de software são construídas e integradas com base em ontologias do domínio de Engenharia de Software. Essa estratégia é a de desenvolvimento integrado, no qual os sistemas são todos construídos com base nas mesmas ontologias, o que facilita a integração das mesmas.

Entretanto, nem sempre é possível a construção de um determinado sistema para ser integrado ao ambiente devido à complexidade que tal tarefa pode envolver. No contexto do desenvolvimento de uma ferramenta (GCS-ODE) para apoiar o processo de GCS, surgiu a necessidade de se integrar essa ferramenta a um sistema externo ao ambiente (o Subversion), o qual foi construído de modo isolado e sem compartilhar nenhuma conceituação com as ferramentas desenvolvidas em ODE.

O intuito dessa integração era usar o sistema de controle de versão Subversion (SVN) para controlar as versões dos artefatos produzidos em ODE. Da mesma forma que ocorre no desenvolvimento integrado, no qual os sistemas são construídos e integrados com

base em ontologias, na integração de sistemas desenvolvidos isoladamente, como o caso de SVN e GCS-ODE, ontologias também podem desempenhar um importante papel.

Para ajudar na realização dessa integração, tinha-se em mãos uma ontologia de domínio de GCS, proposta em (ARANTES; FALBO; GUIZZARDI, 2007), que considerava informações importantes do domínio, inclusive as relacionados ao controle de versão. Na dada ocasião, tal ontologia foi usada na integração de SVN a GCS-ODE.

A integração realizada envolveu as camadas de dados e serviços, porém a integração semântica envolveu apenas a camada de dados, a qual tomou por base a ontologia de domínio citada. Além disso, a integração semântica ocorreu de maneira *ad-hoc*, sem seguir nenhuma diretriz pré-estabelecida.

As dificuldades encontradas e um estudo de trabalhos de integração semântica despertaram para a necessidade de uma abordagem de integração semântica baseada em ontologias que fosse independente de tecnologia, que propusesse um processo de integração e abrangesse as três camadas de integração. Além disso, a integração deveria ocorrer em um nível mais alto de abstração. Ela deveria ser independente de como a solução de integração seria posteriormente projetada e implementada. Como qualquer processo de desenvolvimento de software, o processo de integração deveria começar pelo nível conceitual. Apenas depois de considerar os requisitos da integração e construir o modelo conceitual da integração, os aspectos tecnológicos deveriam ser considerados (fase de projeto).

Tomando por base esta consideração, foi desenvolvida OBA-SI (*Ontology-Based Approach for Semantic Integration*), uma abordagem de integração semântica de sistemas que concentra esforços na modelagem conceitual e na análise dos requisitos de integração. Sistema é um conceito bastante amplo que envolve não apenas sistemas computacionais ou de informação. Tudo pode ser considerado um sistema. Como OBA-SI foca na compatibilização de conceituações inerentes a sistemas, na teoria, essa abordagem poderia ser usada na integração de qualquer tipo de sistema. Mas na prática, ela foi desenvolvida para ser aplicada na integração semântica de sistemas de informação.

Nessa abordagem, a integração semântica é realizada em um alto nível de abstração, provendo acordo semântico entre os sistemas no nível conceitual. OBA-SI lida com a

integração nas três camadas de integração: dados, serviços e processos. Para tal, modelos conceituais dos sistemas (representando sua estrutura e comportamento), bem como do processo de negócio por eles apoiado, são comparados à luz de ontologias, usadas para atribuir semântica aos itens compartilhados entre os sistemas no apoio ao processo de negócio considerado. Os modelos são compatibilizados por meio de mapeamentos entre seus elementos. Todo esse processo de atribuição de semântica e uso de ontologias é independente da solução da integração.

Para avaliar a utilidade da abordagem proposta, ela foi reaplicada na própria integração entre SVN e GCS-ODE. Essa integração semântica foi refeita neste trabalho com base na abordagem proposta. Para tal, foi criada uma ontologia da tarefa que descreve o processo de Gerência de Configuração, também apresentada nesta dissertação, responsável por atribuir semântica às camadas de serviço e de processo.

Como principais contribuições deste trabalho, podem-se citar:

- A abordagem de integração OBA-SI, a qual:
    - Apoia a integração semântica em um alto nível de abstração, independente de aspectos computacionais;
    - Faz uso de ontologias de domínio e de tarefa como descrição da realidade e base para a compatibilização das conceituações dos sistemas.
    - Propõe um processo de integração semântica que facilita a sua aplicação.
- A ontologia de tarefa de Gerência de Configuração construída para aplicar a abordagem proposta no estudo de caso. Essa ontologia:
    - Foi criada usando OntoUML, uma linguagem bem fundamentada para modelagem de ontologias;
    - Ajuda no entendimento preciso do processo de gerência de configuração, representando suas tarefas, agentes envolvidos e objetos mais importantes;

    - Pode ser usada como base para a integração semântica de sistemas no universo de discurso da gerência de configuração.

- A integração semântica realizada entre Subversion e GCS-ODE.

1.3. Organização da Dissertação

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

- **Capítulo 2** – Integração Semântica de Sistemas: apresenta uma revisão da literatura, abordando os seguintes temas: Integração de Sistemas, Níveis e Camadas de Integração, Integração Semântica, Ontologias e Integração Semântica, e Abordagens de Integração Semântica.

- **Capítulo 3** – OBA-SI: Uma Abordagem de Integração Semântica de Sistemas Baseada em Ontologias: apresenta a abordagem de integração OBA-SI, iniciando com uma motivação para uma abordagem de integração semântica de nível conceitual e segue apresentando os níveis de abstração, modelos, formas de mapeamento e o processo de integração propostos por OBA-SI. Este capítulo apresenta, ainda, uma comparação com trabalhos correlatos.

- **Capítulo 4** – Uma Ontologia da Tarefa de Gerência de Configuração: apresenta a ontologia da tarefa de Gerência de Configuração desenvolvida neste trabalho para apoiar a integração semântica nas camadas de serviços e processo do estudo de caso apresentado no Capítulo 5.

- **Capítulo 5** – Estudo de Caso: Integração Semântica de Sistemas de Apoio à Gerência de Configuração de Software: apresenta um estudo de caso de integração semântica usando OBA-SI.

- **Capitulo 6** - Considerações Finais: apresenta as conclusões, procurando destacar as contribuições deste trabalho, bem como as perspectivas de trabalhos futuros.

Capítulo 2 - Integração Semântica de Sistemas

Cada vez mais, a passagem do mundo real para o mundo virtual proporciona os meios adequados para que as organizações organizem e agilizem seus processos internos. Cada vez mais, as organizações estão buscando meios para encontrar modelos capazes de integrar suas soluções, para alcançar sucesso nos negócios. Integração é uma das palavras de ordem no mundo atual (CHIAVENATO, 2004).

Integração de sistemas depende em grande extensão do quanto os sistemas concordam. O objeto de concordância pode incluir formato dos dados, convenções de interfaces gráficas, uso das mesmas funções ou outros aspectos da construção dos sistemas (THOMAS et. al, 1992).

Neste contexto, a integração de sistemas traz inúmeros benefícios à organização, dentre eles: o processo de negócio pode ganhar maior flexibilidade e adaptabilidade, a sua execução se torna mais eficiente e rápida, diminui a incidência de erros e aumenta o poder de tomada de decisão da empresa (POKRAEV, 2009). Envolvendo a totalidade da organização, o conjunto de sistemas integrados compõe um único sistema capaz de manter o fluxo de processos e controlar e integrar as diversas transações internas da organização. Os resultados são: maior eficiência, menores custos, maior rapidez e cliente satisfeito (CHIAVENATO, 2004).

Hoje a integração de sistemas é um problema complexo para a maioria das organizações, pois cada vez mais surge a necessidade de os sistemas trabalharem em conjunto (IZZA, 2009). Atualmente, as organizações vêm usando um crescente número de sistemas para apoiar seus processos, tais como Sistemas de Planejamento de Recursos da Organização (*Enterprise Resource Planning* - ERP) e Sistemas de Apoio à Gestão de Relacionamentos com o Cliente (*Customer Relationship Management*- CRM),. Cada um deles tipicamente compartilha funcionalidades e dados com outros (POKRAEV, 2009).

Entretanto a integração não é uma tarefa simples. Os sistemas utilizados nas organizações, geralmente, não são construídos com o intuito de serem integrados com outros sistemas. Eles geralmente são desenvolvidos por fabricantes distintos que muitas vezes não possuem essa preocupação. Devido a esse fato, os sistemas, em sua grande parte, são enquadrados como sistemas HAD, ou seja, são heterogêneos, autônomos e

distribuídos. Além disso, na maioria das vezes, sistemas funcionam como caixas pretas, não sendo possível acessar seu conteúdo nem mesmo acessar sua interface (BUSSLER, 2003).

Dentre os problemas de integração, o que vem sendo considerado o mais desafiador por especialistas é o da heterogeneidade (IZZA, 2009). Sistemas, em sua maioria, não compartilham o mesmo modelo de dados ou de comportamento, dificultando, assim, a comunicação entre eles (BUSSLER, 2003). Tais modelos representam conceituações (maneiras de ver e pensar) de entidades do mundo real que estão inerentes no sistema. Atribuir semântica a tais modelos é explicitar as conceituações que eles representam, o que pode ajudar no entendimento e na relação dos mesmos. Através da atribuição de semântica aos modelos dos sistemas, é possível explicitar e entender melhor o significado inerente aos mesmos e assim compatibilizá-los de acordo com esse significado. Deste modo, os sistemas podem conversar entre si sem que haja “falsos acordos” (POKRAEV, 2009). Sem explicitar a semântica dos modelos dos sistemas e torná-la clara para os envolvidos na integração, pode-se erradamente achar que existe uma concordância com o significado de coisas envolvidas na integração. É importante que se identifique falsos acordos no início do projeto de integração, caso contrário provavelmente a implementação da integração será incorreta (POKRAEV, 2009).

Na integração semântica, ontologias possuem um importante papel. Como são modelos formais de representação do conhecimento consensual, elas podem servir como base na atribuição de semântica aos sistemas a serem integrados. Entretanto, apesar da grande importância do uso de semântica na integração de sistemas, grande parte das abordagens ainda foca apenas na integração sintática, ou seja, na integração estrutural, sem significado atribuído. Como exemplo de abordagens sintáticas podem-se citar middlewares como CORBA (*Common Object Request Broker Architecture*) e soluções SOA (*Service-Oriented Architecture)*.

Este capítulo tem por objetivo apresentar um referencial teórico a respeito da integração de sistemas, principalmente a integração semântica, mostrando suas motivações e seus problemas. Ele foi feito como resultado de um estudo dirigido sobre integração semântica, o qual foi baseado em revisão sistemática da literatura, realizado em conjunto com Bringuente (2011). Ele está dividido da seguinte forma: na Seção 2.1 são apresentados o contexto e a motivação para a integração de sistemas. Na Seção 2.2, é feita uma discussão sobre integração de sistemas e suas dimensões. Já a Seção 2.3

trata especificamente dos níveis de integração, com ênfase no nível semântico de integração. Na Seção 2.4 é abordado o uso de ontologias na integração semântica de sistemas. Na Seção 2.5, comenta-se acerca de algumas abordagens de integração de sistemas e, por fim, a Seção 2.6 apresenta as conclusões do capítulo.

2.1. Contexto e Motivação para a Integração de Sistemas

Segundo a teoria geral de sistemas, um sistema é um todo, composto de elementos relacionados que interagem entre si, a fim de atingir um objetivo. Quando a interação ocorre de modo adequado, há sinergia. Sinergia é o princípio que afirma que o todo é maior que a soma das partes, ou seja, o todo possui características próprias que não podem ser encontradas nas partes (CHIAVENATO, 2004). Portanto, a integração de sistemas oferece mais do que a simples junção deles poderia oferecer.

De fato, os benefícios de uma integração não se encontram em um sistema isoladamente. Integração não é uma propriedade de sistemas sozinhos, mas de suas relações em um ambiente. Assim, o ponto fundamental de uma integração são as relações entre sistemas e as propriedades dessas relações (THOMAS et. al, 1992).

Nas últimas décadas, as empresas vêm usando um crescente número de diferentes sistemas para suportar seus processos de negócio. Exemplos mais comuns dessas aplicações são os sistemas de Gerência de Relacionamentos com Clientes (*Customer Relationship Management*- CRM), Contabilidade Financeira (*Financial Accounting* - FA), Planejamento de Recursos da Organização (*Enterprise Resource Planning* - ERP) e Informação Logística (*Logistic Information*). Além disso, é muito frequente que a organização desenvolva, ainda, várias aplicações específicas para apoiar seus processos de negócio (POKRAEV, 2009).

Sistemas de informações corporativos são geralmente compostos de múltiplas aplicações e visam solucionar certos requisitos da organização, automatizando certas tarefas de um domínio particular da organização (IZZA, 2009). Uma aplicação corporativa é caracterizada por: componentes de software, contexto de uso (onde a aplicação será utilizada), funcionalidades disponíveis (correspondentes às funcionalidades providas que darão apoio automatizado às tarefas da organização), dados manipulados (dados usados e produzidos) e recursos usados (recursos humanos, de software, de hardware etc) (LEROUX 2004, REIX 1995 apud IZZA, 2009).

As aplicações coorporativas, como dito, são geralmente desenvolvidas de forma isolada, cada uma por um fabricante específico, o qual segue seus próprios padrões de desenvolvimento e de nomenclatura. Dessa forma, não há uma preocupação em seguir uma padronização para que os sistemas possam ser integrados com os de outros fabricantes. Devido a esse fato, as aplicações corporativas geralmente são caracterizadas como aplicações HAD, isto é, Heterogêneas, Autônomas e Distribuídas (BUSSLER 2003). A Figura 2.1 ilustra as dimensões das aplicações corporativas e as soluções mais comumente utilizadas.

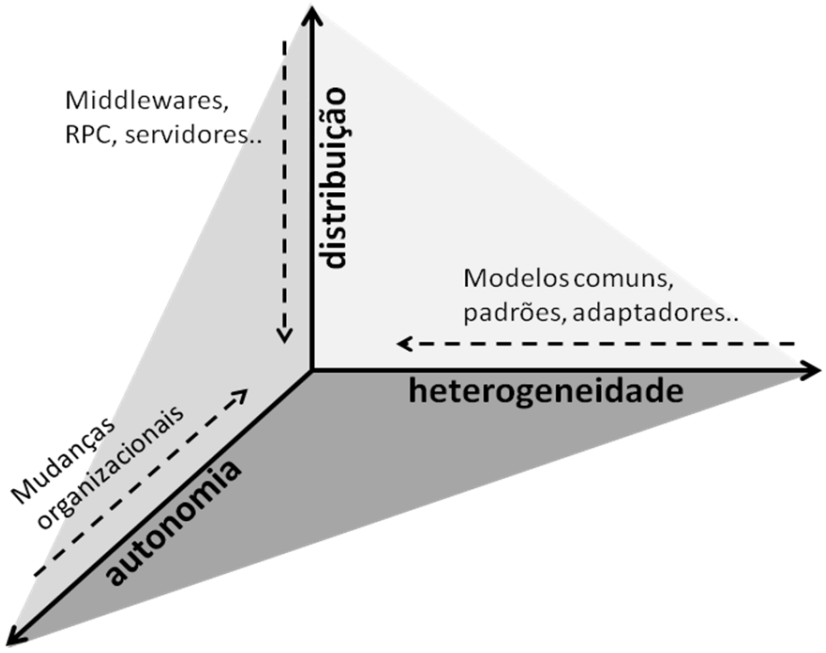

Figura 2.1-Principais dimensões de sistemas corporativos, adaptado de IZZA (2009)

Aplicações heterogêneas são aquelas que possuem modelos de dados e de processo (comportamento) implementados de forma independente, definindo conceitos relevantes da sua própria maneira. Um sistema pode, por exemplo, implementar cliente como sendo uma pessoa física e outra como sendo um parceiro comercial (BUSSLER 2003). Dessa forma, aplicações heterogêneas possuem uma "visão da realidade" diferente da mesma porção da realidade. Já as aplicações autônomas são aquelas que executam de modo independente das outras aplicações, isto é, a aplicação segue seu protocolo de interação de modo independente. Elas atualizam seus estados internos e evoluem de versão independentemente de outras aplicações (BUSSLER 2003). Já as aplicações distribuídas

são aquelas que implementam seu próprio modelo de dados em seu próprio repositório. Elas usam diferentes meios para atualizar ou recuperar estados (BUSSLER 2003).

Diversas soluções para integração de aplicações estão sendo providas focando nessas principais características das aplicações corporativas. Por exemplo, para se integrar sistemas autônomos, são necessárias soluções que lidem com comportamentos assíncronos, como por exemplo, *middlewares* assíncronos. Na integração de sistemas distribuídos, a maior dificuldade diz respeito ao controle de transações, com soluções sendo dadas principalmente pela computação distribuída. Com relação à heterogeneidade, esta é considerada o maior problema na hora de se integrar sistemas corporativos. Para solucionar essa questão devem ser providos meios para resolver conflitos sintáticos e semânticos (IZZA, 2009).

Além das características HAD das aplicações corporativas, elas geralmente são consideradas imutáveis no contexto da organização. Aplicações são, na maioria das vezes, tratadas como caixas pretas (interface não disponível) ou como caixas cinzas (interface disponível), impedindo a alteração das mesmas para que sejam integradas. Entretanto, mesmo quando as aplicações corporativas são do tipo caixa branca (código acessível e modificável), a modificação para a integração é um risco considerável e, por isso, na maioria das vezes, a alteração é inapropriada (IZZA, 2009).

Em resumo, aplicações são entidades de software independentes que são definidas de forma isolada e são operadas de forma autônoma. Qualquer forma de integração entre elas ocorre externamente às mesmas. Isso significa que elas não ficam cientes do fato de estarem sincronizadas de alguma forma com outras aplicações (BUSSLER 2003). Essa característica dificulta a integração. Apesar disso, com o número crescente de aplicações usadas em uma organização, torna-se imprescindível a utilização de meios para que os sistemas conversem entre si e apoiem o processo de melhor forma.

2.2. Integração de Sistemas

Integração de sistemas trata do grau em que os mesmos estão de acordo. O objeto dessa concordância pode incluir formato de dados, convenção da interface com usuário, o uso de serviços comuns e outros aspectos da construção dos sistemas (THOMAS et. al, 1992). A integração visa combinar componentes de modo a formar um novo sistema que represente um todo, melhorando a sinergia e fazendo com que os objetivos da

organização sejam atingidos de modo mais eficiente e produtivo (WESTON, 1993; VERNADAT, 1996 apud IZZA, 2009). Um conjunto de sistemas está integrado se cada um deles funciona como parte de um todo único, consistente e coerente (THOMAS et. al, 1992).

O conceito de interoperabilidade entre sistemas também é bastante usado. Entretanto há certa confusão acerca de sua definição e a sua relação com o conceito de integração, como é discutido em (IZZA, 2009). Portanto, o termo integração será usado aqui como um termo mais genérico que inclui o conceito de interoperabilidade. Pokraev (2009), com base em diversas definições de interoperabilidade, definiu suas principais características. Segundo ele, interoperabilidade possui as seguintes características:

- Envolve múltiplas entidades (sistemas, componentes, organizações, indivíduos, etc);
- É relacionada à habilidade de interagir (operar em conjunto, comunicar, trocar informações ou conhecimento, prover e aceitar serviços);
- Requer pouco ou nenhum conhecimento das características específicas das entidades de interação;
- Visa atingir um determinado objetivo.

Portanto, segundo Pokraev (2009), interoperar, no contexto de integração de sistemas, pode ser considerado a habilidade de sistemas interagirem entre si (usarem serviços uns dos outros), visando atingir um determinado objetivo, sem que eles conheçam informações específicas uns dos outros.

Izza (2009) pesquisou diversas abordagens de integração de sistemas e, a partir dessa pesquisa, organizou as abordagens de acordo com dimensões de integração, ilustradas na Figura 2.2. Como se pode ver, são quatro as dimensões de integração, a saber: nível de integração, camada de integração, ponto de vista da integração e escopo da integração.

Na dimensão de escopo, as abordagens são divididas em dois principais tipos: Intra-organizacional, com foco na integração de aplicações internas à organização; Extra-organizacional, que trata da integração de aplicações de diferentes parceiros (B2B - *business to business* - ou B2C - *business to customer*).

Já a dimensão ponto de vista da integração divide as abordagens em função das seguintes visões: visão programador (ou visão interna da integração), visão do projetista (ou visão conceitual da integração) e visão do usuário (ou visão externa da integração) (IZZA, 2009), como ilustra a Figura 2.3.

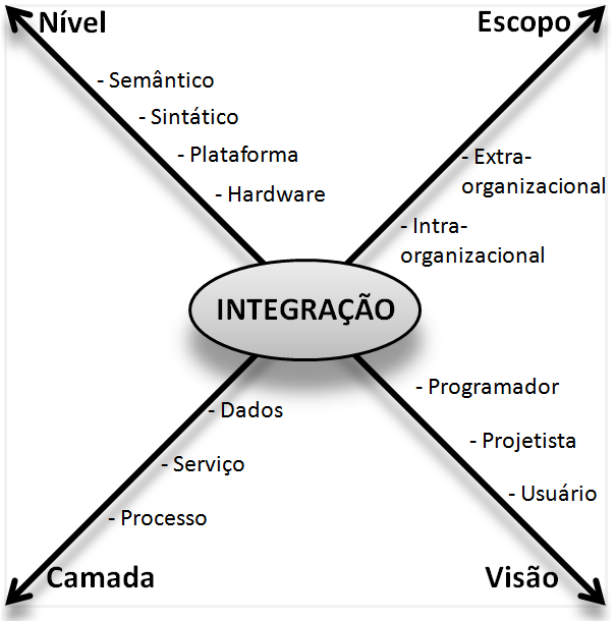

Figura 2.2 -Dimensões de Integração, adaptado de (IZZA, 2009)

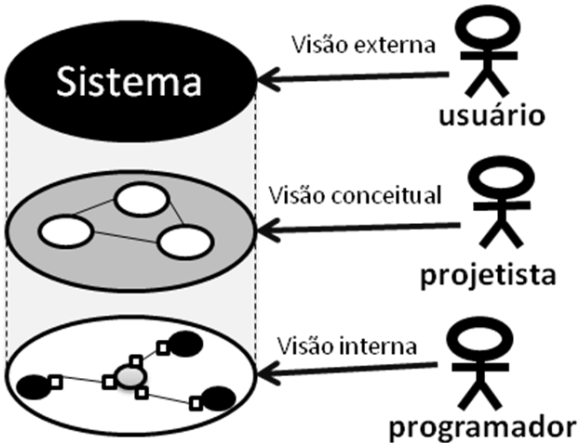

Figura 2.3 -Visões de uma integração de sistemas

Enquanto a visão interna e conceitual busca integrar os sistemas do modo mais fácil, a visão externa, do usuário, busca usar o conjunto integrado de sistemas da maneira mais simples, de preferência abstraindo aspectos internos da integração (THOMAS et al., 1992).

A divisão da integração em camadas varia de acordo com a literatura. Izza (2009) dividiu a integração em camada de dados, de mensagem (ou serviço) e de processo. Entretanto alguns textos também definem a camada de apresentação como Thomas et al. (1992), que não será abordada neste trabalho. A camada de dados refere-se à transferência de dados entre as bases de dados das aplicações (IZZA, 2009). O seu objetivo é garantir que toda informação no conjunto integrado de sistemas é gerenciada como um todo consistente, independentemente de onde ela está localizada e por quem é operada e transformada (THOMAS et. al, 1992). A camada de serviço é responsável pela troca de mensagens entre as aplicações. Seu objetivo é garantir uma flexível combinação de serviços entre os sistemas. Uma vez que serviços consomem dados, a integração de serviço também envolve a de dados (THOMAS et. al, 1992). Por fim, a camada de processo busca relacionar os sistemas de modo a definir a execução de processos de negócio da organização. Seu objetivo é garantir que sistemas interajam eficientemente a fim de apoiar o processo definido (THOMAS et. al, 1992). A Figura 2.4 ilustra essa dimensão de integração.

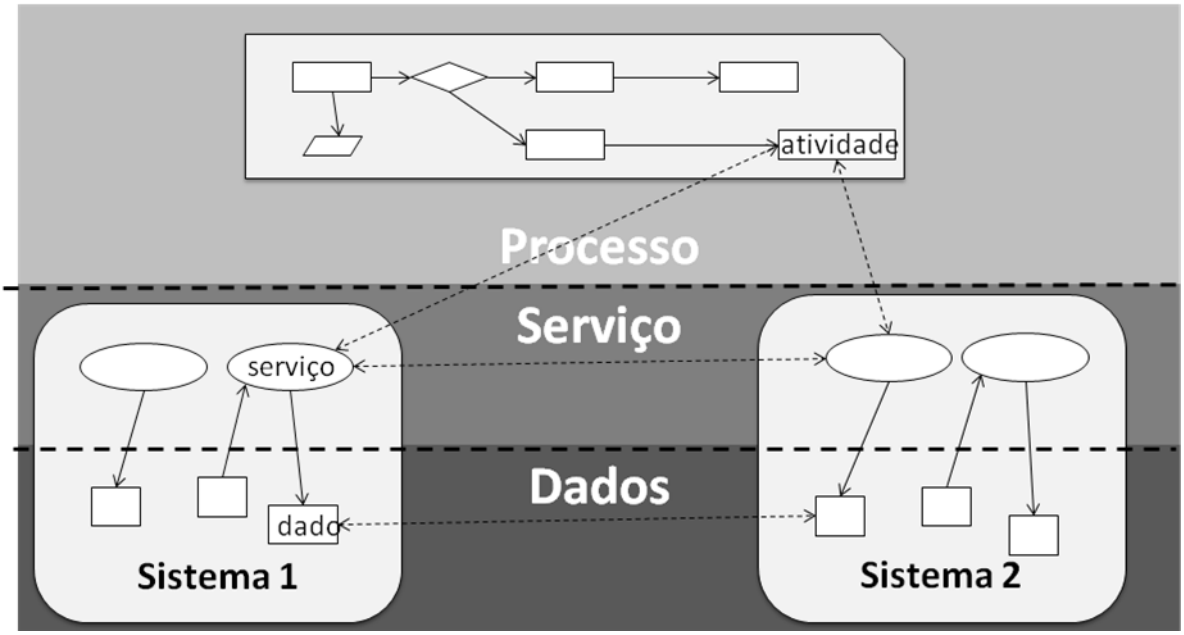

Figura 2.4 -Camadas de Integração de sistemas

Por fim, os níveis de integração são divididos em nível de hardware, de plataforma, sintático e semântico. O nível de hardware lida com as diferenças existentes entre componentes no nível físico (relativo ao hardware e redes). O nível de plataforma lida com

as diferenças existentes entre sistemas operacionais, sistemas de gerenciamento de banco de dados e *middlewares*.

Já o nível sintático lida com a maneira como modelos de dados e assinaturas das operações são escritas, ou seja, com a sua estrutura. A integração no nível sintático está relacionada com os mecanismos de comunicação entre sistemas (BUSSLER, 2003). Por último, o nível semântico envolve o significado dos conceitos nos esquemas de dados ou assinatura das operações. A integração semântica de sistemas será abordada mais detalhadamente adiante. É importante salientar que, para que haja integração em um nível, é necessário que ela ocorra no nível inferior. Por exemplo, para acontecer a integração no nível semântico, deve ter ocorrido integração no nível sintático (IZZA, 2009). Todas as atribuições de semântica são realizadas em cima de elementos sintáticos (BUSSLER, 2003). A Figura 2.5 apresenta uma ilustração dos níveis de integração.

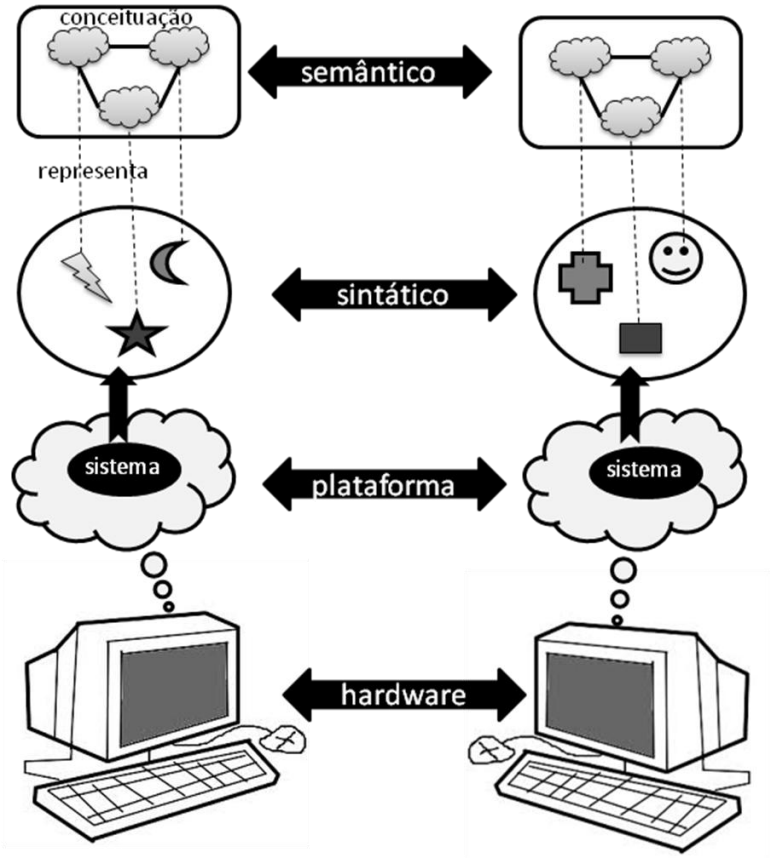

Figura 2.5 -Ilustração dos níveis de integração entre sistemas

Pokraev (2009) define um nível de integração que viria depois da integração semântica, chamado por ele de interoperabilidade pragmática. Esse nível de integração

está relacionado com a integração de comportamento dos sistemas e que depende dos outros tipos de integração para ocorrer.

2.3. Níveis de Integração

Conforme citado anteriormente, a integração de sistemas pode ocorrer em vários níveis. Os níveis de integração vão desde os níveis mais baixos, mais operacionais, como o de hardware e o de plataforma, até os níveis mais altos de integração, relativos a aspectos linguísticos, como os níveis sintático, semântico e pragmático. Os níveis mais baixos da integração estão relacionados com aspectos tecnológicos da integração, enquanto os mais altos estão relacionados com aspectos estruturais, semânticos e comportamentais da integração (IZZA, 2009).

Pokraev (2009) utiliza a definição de linguagem de Morris (1938 apud POKRAEV, 2009) para definir os níveis mais altos de interoperabilidade. Segundo ele, uma linguagem é composta de sintaxe, semântica e pragmática. A sintaxe estabelece o conjunto de símbolos (vocabulário), bem como as relações entre eles (gramática). A semântica é responsável por relacionar os símbolos às entidades do mundo real que eles representam. Já a pragmática relaciona os símbolos com a interpretação humana.

A interoperabilidade sintática visa garantir que os sistemas envolvidos em uma comunicação usem o mesmo vocabulário (conjuntos de símbolos) e mesma gramática (conjunto de regras de combinação dos símbolos) (POKRAEV, 2009). Ela permite a cooperação entre sistemas independentemente das diferenças técnicas de cada um (PARK; RAM, 2004). Já a interoperabilidade semântica visa garantir que um símbolo tenha o mesmo significado (referencia a mesma coisa no mundo real) para todos os sistemas que utilizem o símbolo (POKRAEV, 2009). Assim, enquanto a interoperabilidade sintática ocorre no nível estrutural, envolvendo aspectos tecnológicos da integração, a interoperabilidade semântica ocorre no nível de conhecimento (PARK; RAM, 2004).

2.3.1. Integração Semântica

Sistemas possuem diferentes formas de serem acessados, as quais estão relacionadas com a integração sintática. Mas, independentemente dos mecanismos, é importante expor a semântica por trás deles (BUSSLER, 2003). Existem diferentes tipos de abordagens de integração sintática de sistemas como middlewares, que permite a

troca de mensagens e objetos entre sistemas, como CORBA (*Common Object Request Broker Architecture*), as abordagens baseada no conceito de serviços, como SOA (*Service-Oriented Architecture)*, soluções baseadas nos sistemas de gestão do processo de negócio, como BPM (*Business Process Management)* e as baseadas no princípio de arquitetura de sistemas usados no processo de integração de aplicações corporativas, como EAI (*Enterprise Application Integration)*. Até mesmo padrões como XML, que possibilita troca de informações entre sistemas, e UML um padrão de modelagem, são consideradas abordagens de integração no nível sintático (IZZA, 2009).

Apesar do grande número de abordagens para se integrar sistemas, a maioria delas não provê uma descrição semântica dos serviços e funcionalidades (sua interface). Isso dificulta a integração já que dois sistemas apenas podem interoperar se possuem uma compatibilidade de significado das coisas (semântica) (POKRAEV, 2009).

Para isso é necessário identificar a semântica intrínseca aos sistemas. Quando a semântica intrínseca a um sistema não está bem definida, o responsável pela integração deve fazer um esforço mental para entender e extraí-la. Com isso, ele deve compreender o significado de estruturas de baixo nível para realizar a integração semântica de modo adequado. Por outro lado, quando a semântica está bem definida, não é necessário esse esforço por parte do responsável da integração, diminuindo problemas de interpretação da semântica (BUSSLER, 2003).

No contexto de integração de sistemas, semântica é o significado atribuído aos itens de um sistema, como dados, conceitos e funcionalidades. Sistemas podem ser considerados representações do mundo real. Na sua criação, são selecionados entidades e fenômenos (e suas respectivas propriedades) do mundo real que são importantes para o sistema (POKRAEV, 2009). Explicitar o que cada item de um sistema representa em termos de entidades e fenômenos do mundo real é justamente atribuir semântica a ele. Isso ajuda a resolver diferenças implícitas de significado, perspectivas e premissas criando um ambiente semanticamente compatível, baseado no acordo de conceitos entre sistemas (PARK; RAM, 2004).

Antes de se integrar dois ou mais sistemas é necessário que se analise a possibilidade de integração. Por meio de suas descrições (de seus serviços), o responsável pela integração deverá decidir se ela poderá acontecer ou não, verificando, dentre outras coisas, se eles possuem ou não compatibilidade semântica, ou seja, se eles concordam

ou não com os significados atribuídos (POKRAEV, 2009). Essa tarefa ocorre no nível de conhecimento, que muitas vezes é tácito, o que torna a tarefa de integração mais complexa devido à subjetividade da mesma (PARK; RAM, 2004).

Identificar se um conjunto de sistemas é compatível é uma tarefa bastante difícil e a falta de descrição semântica dificulta ainda mais a sua descoberta. É importante que essa identificação ocorra o quanto antes. Quanto mais tarde isso for descoberto no projeto de integração, maiores serão os impactos decorrentes de incompatibilidades semânticas descobertas (POKRAEV, 2009).

2.3.2. Problemas na integração semântica

Como foi dito, sistemas são representações do mundo real e, para os sistemas conversarem entre si, essas visões do mundo devem ser compatíveis. Entretanto, cada sistema geralmente possui uma visão (conceituação) diferente do mundo real, dificultando a comunicação entre eles, como ilustra a Figura 2.6. Nela há dois sistemas, cada qual com sua diferente visão da realidade. Cada um dos sistemas representa uma parte da realidade, possuindo uma parte comum, normalmente representada de modo diferente por ambos. É justamente nessa parte comum em que deverá ocorrer a integração.

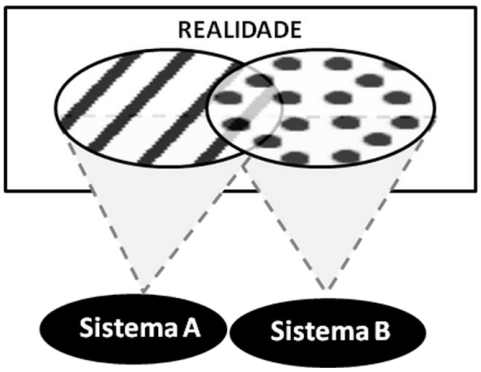

Figura 2.6 - Falta de compatibilidade semântica na integração entre sistemas com diferentes representações da realidade

Conflitos sintáticos ou estruturais ocorrem quando diferentes sistemas armazenam dados e informações em diferentes estruturas. Já os conflitos semânticos ocorrem quando o conteúdo dos itens de informação não possui o mesmo significado pretendido. Eles ocorrem sempre que dois sistemas não usam a mesma interpretação das coisas, ou seja, quando itens parecem possuir o mesmo significado, mas na verdade não possuem (WACHE 2001).

Segundo Noy (2004), os conflitos semânticos podem estar no nível da linguagem ou no nível dos modelos. Os conflitos referentes à linguagem podem ser causados por inexpressividade e diferenças na sintaxe da linguagem. Os conflitos no nível de modelo ocorrem quando um mesmo termo representa diferentes conceitos ou quando diferentes termos são usados para representar o mesmo conceito ou quando há uso de diferentes paradigmas de modelagem, de diferentes convenções ou de diferentes níveis de granularidade (NOY, 2004).

Segundo Pokraev (2009), os problemas de interoperabilidade semântica ocorrem principalmente devido a falsos acordos ou então diferenças na abstração (conceituação). O primeiro problema ocorre basicamente quando símbolos estabelecidos como iguais são usados para representar coisas diferentes no mundo real ou símbolos estabelecidos como diferentes são usados para representar a mesma coisa. Já a diferença de abstração ocorre devido à diferença de representação de um mesmo conceito ou de seus relacionamentos.

A Figura 2.7 ilustra os problemas semânticos de falso acordo mais comuns (POKRAEV, 2009). Eles podem ser divididos em dois tipos, os de *falsa equivalência* (os três primeiros), ou os de *falsa diferença* (os três seguintes). A seguir cada um dos casos é descrito.

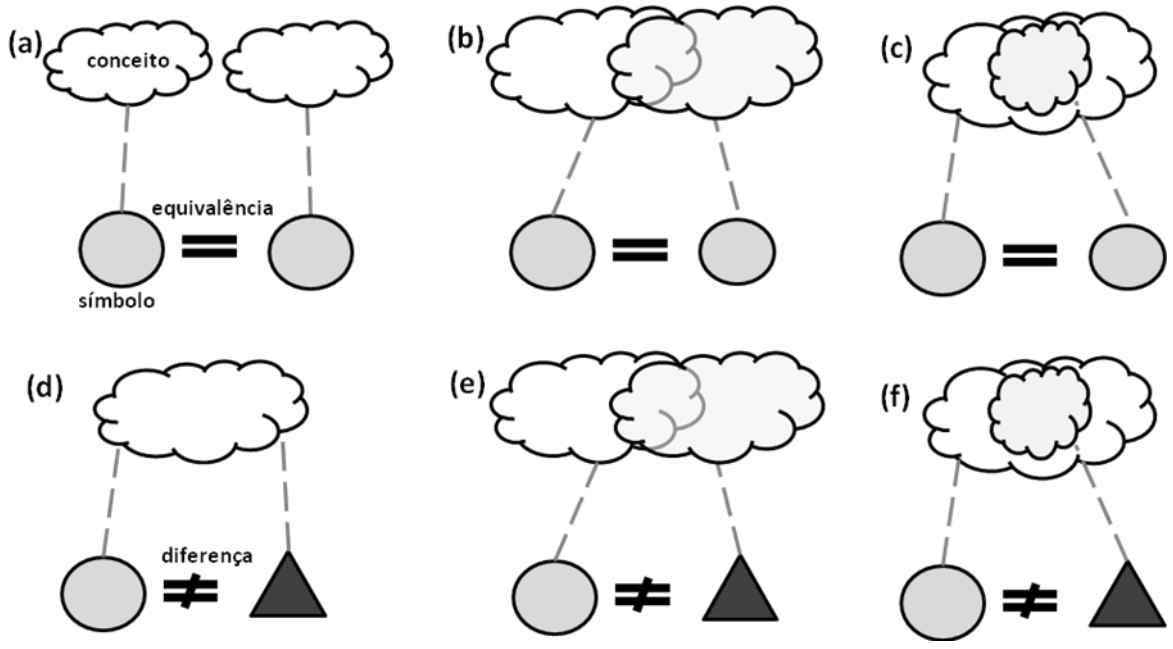

Figura 2.7– Alguns tipos de conflitos semânticos descritos em (POKRAEV, 2009)

O problema (a) é o de falsa equivalência total. Sistemas usam símbolos equivalentes, mas que representam conceitos com significado disjunto. Por exemplo, dois sistemas usam o símbolo *banco*, mas em um deles o símbolo representa um tipo de assento enquanto no outro representa um tipo de empresa. Os problemas (b) e (c) são problemas de falsa equivalência parcial, ou seja, sistemas usam símbolos equivalentes, mas que representam conceitos com significados parcialmente equivalentes (POKRAEV, 2009). Para exemplificar o caso (b), suponha dois sistemas que usam o símbolo *pessoa*, entretanto um deles se refere ao conceito legal de pessoa, uma pessoa física, e no outro se refere ao conceito de pessoa como sendo um tipo de ente racional. Existem pessoas legais, mas que não são entes racionais, como as que estão em estado vegetativo, da mesma forma que existem pessoas que são entes racionais, mas não são pessoas legais. Para exemplificar o caso (c), suponha que esse mesmo símbolo *pessoa*, em um sistema representa o conceito de pessoa legal (tanto jurídica quanto física) enquanto no outro representa apenas o conceito de pessoa física.

O problema (d) é o da falsa diferença total e ocorre quando símbolos estabelecidos como distintos se referem a conceitos totalmente equivalentes. Por exemplo, os símbolos Aluno e Estudante, definidos por dois sistemas, representam ambos o conceito referente a uma pessoa matriculada em uma instituição de ensino. Por fim, os problemas (e) e (f) são ditos problemas de falsa diferença parcial. Eles ocorrem quando símbolos diferentes representam conceitos que possuem um grau de equivalência (POKRAEV, 2009).

Outro tipo de problema citado por Pokraev (2009) é o decorrente de diferentes conceituações de uma mesma entidade do mundo real. Segundo esse mesmo autor, geralmente conceitos são definidos a partir de outros conceitos (por meio de relacionamentos e atributos). Por exemplo, o conceito *Pai* pode ser definido a partir dos conceitos *Homem* e *Ancestral*. Com isso, um mesmo conceito pode ser definido de diferentes formas. Por exemplo, o conceito *Empregado* pode ser definido como pessoa que recebe salário de uma companhia ou então como pessoa que possui um contrato empregatício estabelecido com ela. Nesse caso, um mesmo conceito *Empregado* é definido por diferentes conceituações como ilustra a Figura 2.8. Esse tipo de problema é o foco deste trabalho e será abordado mais adiante nesta dissertação.

Os conflitos apresentados em (POKRAEV, 2009) são conflitos que ocorrem no nível de classes de elementos (ou nível do esquema). Entretanto, os conflitos semânticos também podem ocorrer no nível de dados, entre indivíduos (PARK; RAM, 2004). Conflitos

semânticos desse tipo geralmente ocorrem na integração de bases de dados, na qual devem-se relacionar elementos equivalentes. Um exemplo de integração nesse nível ocorre entre os dados “Vitória” e “*VIX”* (dois símbolos distintos que referenciam a mesma entidade no mundo real, no caso a cidade de Vitória), que pertencem a bases de dados distintas.

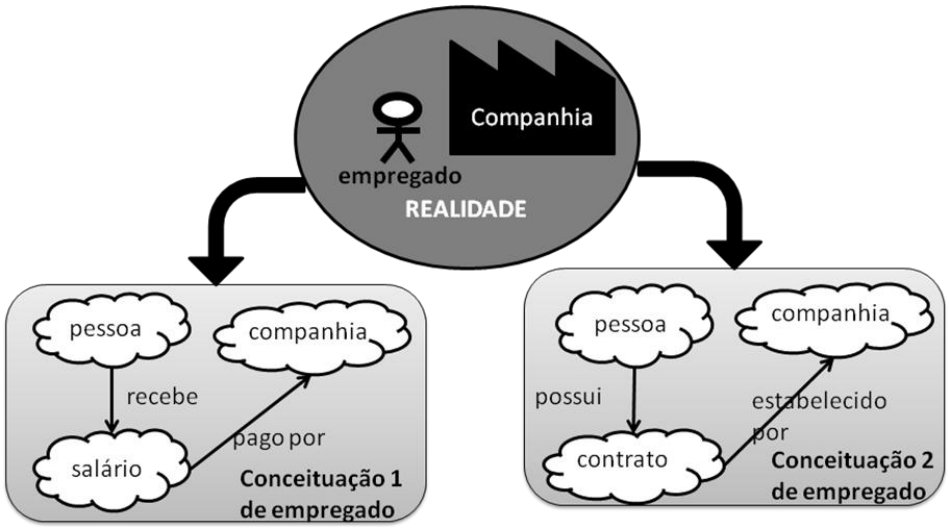

Figura 2.8 - Diferenças nas conceituações de uma mesma entidade do mundo real

2.4. Ontologias e Integração Semântica de Sistemas

Obstáculos à interoperabilidade entre sistemas surgem, principalmente, do fato deles serem geralmente criados de forma independente e não compartilharem a mesma semântica para a terminologia usada para descrever o domínio de interesse. Diferentes termos podem estar sendo usados para representar o mesmo conceito. Sem uma definição explícita do significado dos termos envolvidos é difícil saber a correspondência entre os conceitos de diferentes sistemas.

Entretanto, a atribuição de semântica aos sistemas para que eles conversem não é uma tarefa simples. Devido à subjetividade na atribuição de semântica, é muito difícil que algum método consiga solucionar completamente a questão da interoperabilidade semântica. E por causa desses julgamentos subjetivos, a automatização total torna-se praticamente inviável (PARK, RAM, 2004).

A atribuição de semântica pode sim ser feita com base no conhecimento tácito de uma organização. Entretanto essa não é a melhor maneira devido à subjetividade desse

conhecimento. Faz-se necessário, então, utilizar uma linguagem para representar esses modelos mentais em termos de um artefato concreto e, assim, poder documentar, comunicar e analisar adequadamente as conceituações e abstrações (GUIZZARDI, 2007). Técnicas de representação do conhecimento podem ser usadas para especificar formalmente pelo menos parte do conhecimento tácito de um domínio. Tornando esse conhecimento explícito, a tarefa de atribuir significado torna-se menos subjetiva, facilitando a integração semântica e a automatização (PARK; RAM, 2004).

O conhecimento de um domínio pode ser representado usando diversos tipos de linguagens para esse fim. Cada uma das linguagens de representação de conhecimento pode possuir diferentes níveis de formalização e expressão (POKRAEV, 2009). Uma linguagem de modelagem deve prover aos seus usuários um conjunto de primitivas de modelagem que possa expressar diretamente os conceitos relevantes do domínio de uma forma concisa, completa e não ambígua (GUIZZARDI, 2007). Apesar de ser mais complexa e mais difícil de usar, quanto maior for o poder de expressão de uma linguagem, mais conhecimento tácito pode ser capturado. Isso permite uma melhor atribuição de semântica aos sistemas, pois o conhecimento do mundo real é mais detalhado. Com isso, é mais fácil distinguir entidades e fenômenos diferentes, evitando falsos acordos.

Dessa forma, a utilização de uma representação formal do conhecimento do domínio facilita a atribuição de semântica. Modelos de conhecimento do domínio podem servir de base para a atribuição de semântica aos sistemas. Para tal, o ideal é utilizar um modelo de domínio que seja consensual. É justamente esse o papel de ontologias na atribuição de semântica aos sistemas, como ilustra a Figura 2.9.

O papel das ontologias é fornecer uma visão consensual da realidade, possibilitando que os sistemas entendam uns aos outros e, assim, conversem entre si. Para que isso ocorra, é necessário traduzir a linguagem de um sistema particular para a linguagem de outro, usando a ontologia como uma interlíngua,.

Ontologias podem ser usadas na tarefa de integração para descrever a semântica das fontes de informação e tornar seu conteúdo explícito. Elas podem ser usadas para identificar e associar as correspondências semânticas dos conceitos dos sistemas (WACHE, 2001).

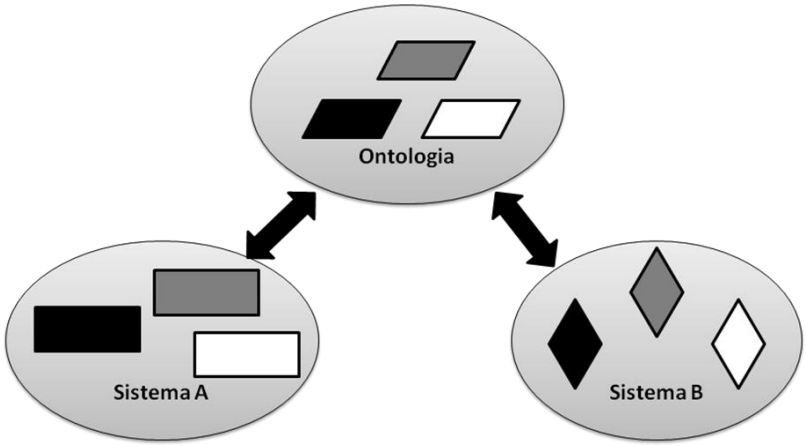

Figura 2.9 - Papel das ontologias na integração de sistemas: fornecer uma visão comum da realidade e servir como uma interlíngua

2.4.1. Ontologias

Uma ontologia é uma especificação explícita e formal de uma conceituação compartilhada, ou seja, que é consenso entre várias partes. É um artefato independente de tecnologias que explicita de forma não ambígua o conhecimento do domínio que é consenso, facilitando a sua análise, entendimento e comunicação.

Em (GUIZZARDI, 2007), é feita uma discussão sobre os diversos "tipos" de ontologias. Segundo Guizzardi (2007), *Ontologia* é uma disciplina originária da Filosofia como um ramo da meta-física que pode ser descrita como "A ciência do ser enquanto ser". *Ontologia Formal* também faz parte da disciplina de Filosofia e estuda teorias gerais que não são específicas de nenhum campo da ciência, ou seja, independentes de domínio (meta-conceituações). *Ontologia de Referência* é a sua representação como artefato concreto, com objetivo de ser entendida por humanos. *Ontologia de Fundamentação* é uma ontologia ao mesmo tempo de referência e formal, ou seja, uma ontologia de referência independente de domínio. *Ontologia de Domínio* é uma ontologia de referência que representa a conceituação de um domínio específico, feito com base em uma ontologia de fundamentação. Já as *Ontologias Leves* são versões de ontologias focadas em garantir propriedades computacionais desejáveis. Seu objetivo é ser entendida / utilizada por máquinas. São também chamadas de Implementação de Ontologia e para representá-las utiliza-se uma linguagem de implementação com a expressividade convenientemente restringida.

É importante ressaltar que ontologias não são artefatos dependentes da integração. Seu objetivo é descrever a realidade independentemente das particularidades de uma integração, embora muitas abordagens atuais a considerem como um modelo criado especificamente para interoperabilidade semântica. Para a criação e implementação de ontologias de qualidade, devem ser usados métodos de Engenharia de Ontologia, analogamente à necessidade dos métodos de Engenharia de Software para se desenvolver software de qualidade (GUIZZARDI, 2007).

Guarino (1998) propõe uma divisão de ontologias em tipos de acordo com nível de generalidade delas, como ilustra a Figura 2.10. As *Ontologias de Fundamentação* são ontologias que descrevem conceitos muito gerais, independentes de domínio ou tarefa particular, tais como espaço, tempo, matéria, objeto, evento, ação etc. As *Ontologias de Domínio* descrevem o vocabulário relacionado a um domínio genérico como, por exemplo, medicina ou automobilismo. Já as *Ontologias de Tarefa* descrevem tarefas ou atividades genéricas como, por exemplo, venda ou locação. Já as *Ontologias de Aplicação* descrevem conceitos relativos a um domínio particular e a uma tarefa particular ao mesmo tempo. Elas são, portanto, especializações das ontologias de tarefa e de domínio.

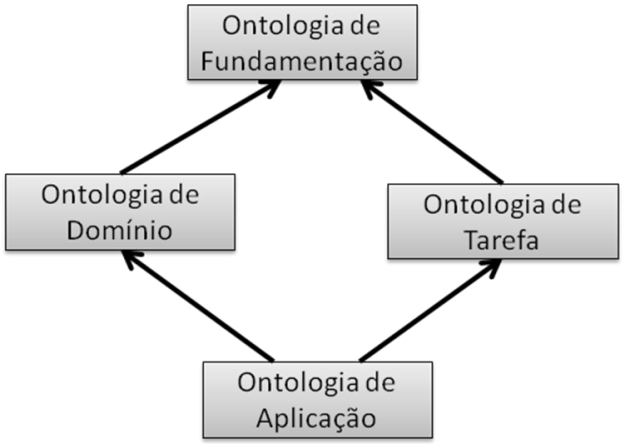

Figura 2.10 - Tipos de Ontologia (adaptado de (GUARINO, 1998))

Ontologias de domínio definem um vocabulário comum sobre um domínio específico. Elas têm sido usadas vastamente na resolução de problemas em várias áreas de computação, mas o mesmo não ocorre com ontologias de tarefa (MARTINS, 2009). Segundo a definição de Guarino (1998), ontologias de tarefa capturam o conhecimento genérico de uma tarefa. Martins (2009) reforça que essa visão de ontologias pode decompor uma tarefa em duas perspectivas interrelacionadas: (i) a perspectiva

comportamental, que enfoca a decomposição da tarefa em sub-tarefas e a modelagem do fluxo de controle; e (ii) a perspectiva estrutural, que envolve a modelagem dos papéis de conhecimento a serem desempenhados em uma tarefa. É bom salientar que essas duas perspectivas são complementares.

Segundo Guarino (1998), ontologias de tarefa e de domínio idealmente devem ser criadas com base em ontologias de fundamentação. Ontologias de fundamentação servem como base para estabelecer consenso e negociação entre humanos. Elas têm sido usadas de forma satisfatória para melhorar a qualidade de linguagens de modelagem e modelos conceituais (GUIZZARDI, 2005). Um exemplo de ontologia de fundamentação é a Ontologia de Fundamentação Unificada (*Unified Foundational Ontolgy* - UFO) (GUIZZARDI, 2005) (GUIZZARDI et al., 2008).

OntoUML

Em geral, qualquer modelo estrutural pode ser usado para representar ontologias de domínio, como, por exemplo, diagramas de classe UML (GUIZZARDI, 2005). Entretanto, a maioria das linguagens usadas não é adequada para isso, pois não são ontologicamente bem fundamentadas (GUIZZARDI, 2005). Para resolver essa questão, Guizzardi (2005) propôs um perfil UML chamado OntoUML, composto de um conjunto de estereótipos que representam distinções ontológicas definidas na UFO. Dentre tais estereótipos estão tipos (*kind*), papéis (*role*), fases (*phase*), categorias (*category*) e misturas (*mixin*) .

A Figura 2.11 ilustra um exemplo de um modelo criado usando OntoUML. Como mostra a figura, a classe Pessoa é do tipo *kind*. Esse estereótipo representa classes cujas instâncias possuem o mesmo princípio de identidade. As instâncias são rígidas, pois sempre pertencerão ao mesmo tipo, e existencialmente independentes. No exemplo, uma instância de Pessoa nunca deixará de ser uma pessoa.

Diferentemente de *kind*, o estereótipo *role* representa um tipo não rígido, ou seja, um objeto pode pertencer a esse tipo por um tempo, mas pode deixar de pertencer depois. No exemplo ilustrado, a classe *Estudante*, subclasse de *Pessoa*, é uma classe do tipo *role*. Isso significa que uma instância de Pessoa pode em um determinado momento se tornar uma instância de Estudante e depois deixar de ser. Outra característica de *role* é que ele representa classes existencialmente dependentes. No exemplo, instâncias da classe *Estudante* apenas existem quando relacionadas com instâncias da classe

Instituição de Ensino. No exemplo, o relacionamento "*estuda em*" entre Estudante e Instituição de Ensino é um exemplo de relação material. Relações materiais precisam de um objeto (*relator*) para existir e, por isso a relação é chamada de *material*. O objeto que intermedeia a relação material "*estuda em*" é uma matrícula. A classe *Matricula*, portanto, faz o papel de mediação entre os envolvidos na relação material e é representada pelo estereótipo *relator*. Ainda que mostrado na Figura 2.11, relações materiais são derivadas e, portanto, não precisam ser representadas em um modelo. Nos modelos apresentados nesta dissertação, as relações materiais não serão mostradas.

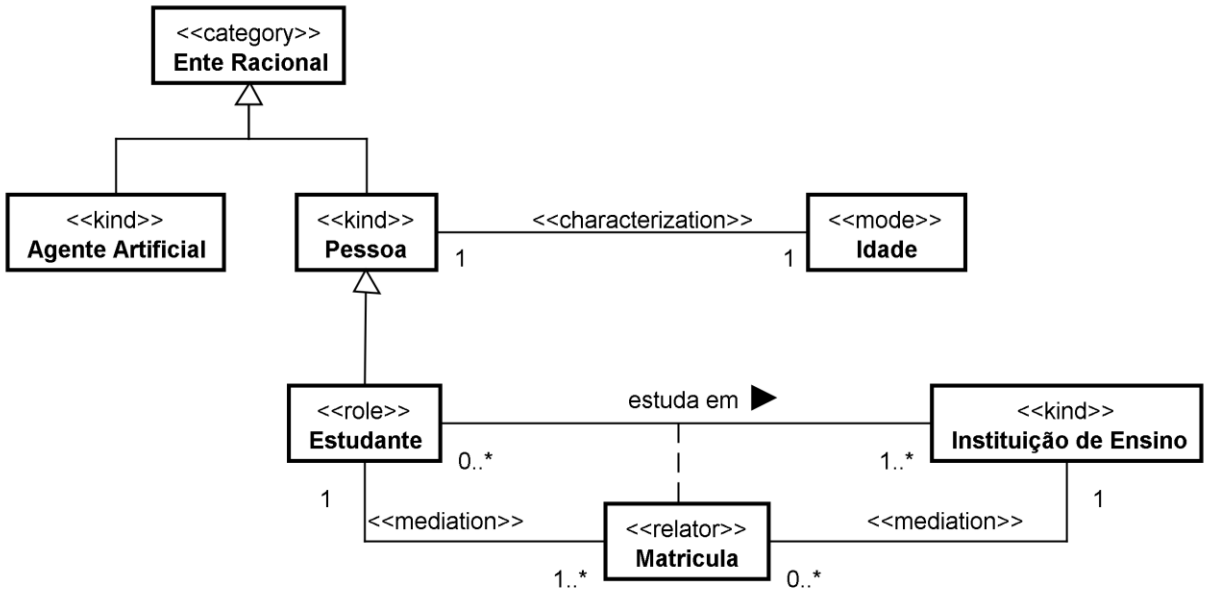

Figura 2.11 - Exemplo de modelo construído usando OntoUML

Mode é outro tipo de estereótipo de OntoUML. Ele é colocado em classes que representam alguma propriedade de outra classe. No exemplo, *Idade* é uma classe do tipo *mode*, uma vez que representa uma propriedade de Pessoa.

Category, como definido em (GUIZZARDI, 2005), representa uma classe que agrupa diferentes classes que possuem propriedades em comum. Por exemplo, Pessoa e Agente Artificial são classes distintas, com diferentes critérios de identidade, mas ambas representam objetos racionais. Essa propriedade em comum é representada por meio da classe *Ente Racional*, do tipo *category*.

RoleMixin possui uma função parecida com a de *category*, mas ao invés de agrupar classes do tipo *kind*, agrupa classes do tipo *role*. Um *rolemixin* representa um agrupamento de diferentes papéis com propriedades em comum (GUIZZARDI, 2005).

O uso de uma linguagem de modelagem como a OntoUML pode ajudar bastante na solução de problemas de integração semântica relativos a aspectos estruturais e pode ser muito útil na modelagem de domínio. Todavia também é importante a representação de tarefas usando linguagens de modelagem expressivas, baseadas em ontologias de fundamentação. Com esse intuito, de melhorar a representação do conhecimento de tarefa, Martins (2009) propôs uma extensão de OntoUML para representar informações comportamentais. Em seu trabalho, Martins propõe o uso de algumas distinções ontológicas de UFO aplicadas através de estereótipos em diagrama de atividades da UML. Tal extensão é chamada de E-OntoUML. Um exemplo de modelo em E-OntoUML é ilustrado na Figura 2.12.

A figura apresenta a modelagem da tarefa de locação. O modelo é composto de duas perspectivas: a estrutural (à esquerda) e a comportamental (à direita). Na perspectiva estrutural são mapeados os papéis de conhecimento, no exemplo: *Item*, *Locação*, *Locador* e *Locatário*. Ela é representada por meio de um diagrama de classes em OntoUML. Locação é um *relator* que possui relações de mediação com *Item* (elemento que sofre locações), *Locador* (responsável por ceder o item locado) e *Locatário* (aquele que aluga o item). Tais elementos são do tipo *rolemixin*, uma vez que representam papéis que podem ser realizados por diferentes tipos de classes, em função do domínio de aplicação.

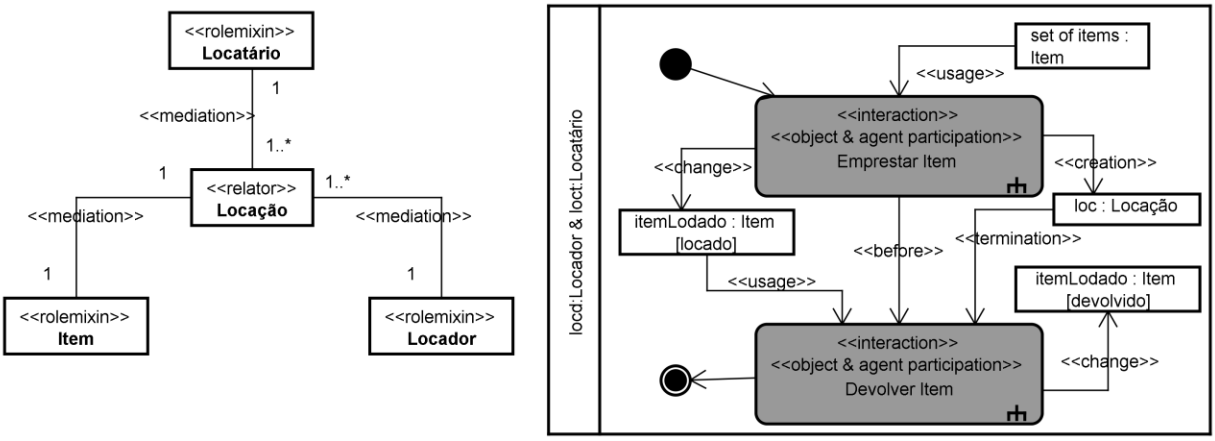

Figura 2.12 - Exemplo de modelo criado usando E-OntoUML

Já a perspectiva comportamental decompõe a tarefa de locação em atividades. Como está ilustrado, inicialmente ocorre a ação Emprestar Item, que utiliza um conjunto de itens existentes, altera a situação de um deles para locado e cria um objeto locação. Esse item passa a estar locado até a realização da ação de devolver item. Na ação Devolver Item, a situação do item locado é alterada novamente para disponível e o objeto locação é finalizado.

Ambas as ações envolvem a participação de agentes (locador e locatário), bem como de objetos (item), como explicita o estereótipo *object & agent participation*. Por meio delas ocorre também a interação entre locador e locatário, como mostra o estereótipo *interaction*. Por fim, essas ações são complexas não só no efeito, mas também pelo fato de serem divisíveis em subações de agentes, sendo, portanto, representadas pelo elemento de modelo Atividade de Chamada (MARTINS, 2009), o qual representa uma chamada a uma *Atividade de Agente*.

As ações de emprestar e devolver são sequenciais (uma só começa quando a outra termina), entretanto a precedência é não imediata, como indica o estereótipo *before*. Além dos estereótipos citados, existem os relativos à participação dos objetos nas ações. Uma participação pode ser de quatro tipos: criação (*creation*), utilização (*usage*), finalização (*termination*) e alteração (*change*). Existem outras distinções propostas em (MARTINS, 2009), entretanto elas não são importantes para este trabalho.

2.5. Abordagens de Integração Semântica de Sistemas

A integração de sistemas pode ser vista como um processo que visa construir um sistema integrador, a fim de satisfazer requisitos do usuário, usando alguma metodologia de integração. O sistema integrador é responsável por ligar os sistemas existentes e compensar as diferenças entre eles por meio da adição de sistemas adicionais chamados de adaptadores (POKRAEV, 2009). A Figura 2.13 ilustra essa situação

Muitas abordagens de integração de sistemas já foram propostas, sendo a maioria delas focada principalmente em aspectos tecnológicos e estruturais. Entretanto já existem várias abordagens que consideram a semântica para integrar sistemas, como (IZZA, 2009), (POKRAEV, 2009), (BOURAS, 2006), (TEKTONIDIS, 2005) e (PARK; RAM, 2004). O objetivo delas é tratar as diferenças semânticas (de significado) entre entidades e serviços compartilhados por aplicações.

Conforme discutido anteriormente, um grande problema para a integração semântica é a falta de uma especificação explícita e compartilhada do significado do conteúdo corporativo. Para que ocorra a integração semântica, o significado deve estar explicito de alguma forma. Ele deve ser derivado do consenso das pessoas e dos termos de uso comum (USCHOLD & GRUNINGER, 2002 *apud* IZZA, 2009). Ontologias podem ser usadas para formalizar e explicitar esse conhecimento consensual. Relacionamentos entre itens dos sistemas e itens da ontologia esclarecem a semântica dos sistemas e ajudam a identificar a correspondência semântica entre os sistemas (WACHE, 2001).

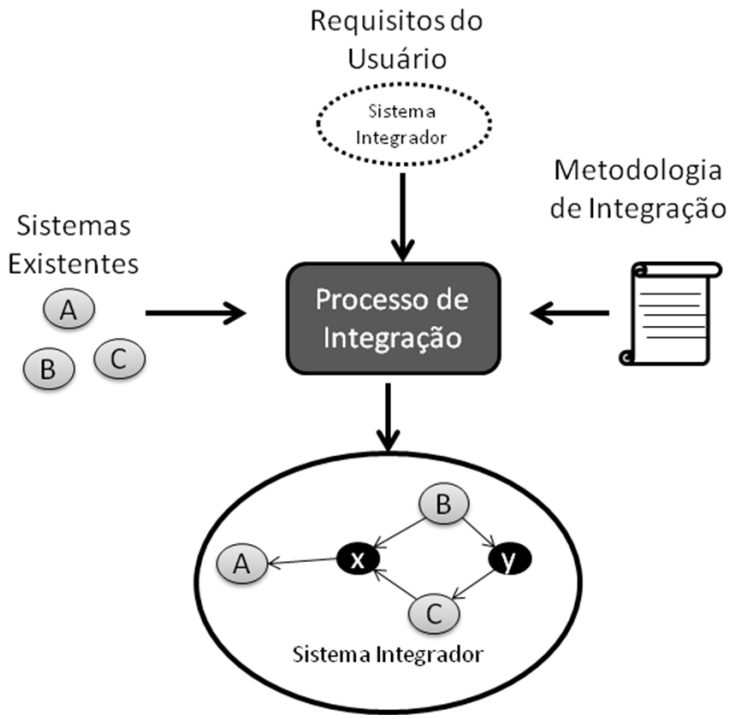

Figura 2.13 - Ilustração do Processo de Integração, adaptado de (POKRAEV, 2009)

A atribuição de semântica aos itens de um sistema pode ocorrer de diferentes maneiras. Wache (2001) aponta que, em geral, abordagens usam modelos conceituais de três maneiras na integração semântica de sistemas: (i) usando um único modelo global para representar todos os sistemas; (ii) usando um modelo para representar cada sistema e relacionando tais modelos entre si, e; (iii) usando uma abordagem híbrida. Na abordagem híbrida é usado um modelo geral, podendo ser uma ontologia, como na primeira abordagem, e cada sistema possui um modelo que o descreve, como na segunda abordagem. Só que, ao invés de os modelos dos sistemas serem relacionados

entre si, como na segunda abordagem, eles são relacionados por meio da ontologia (WACHE, 2001).

A ontologia é responsável por fornecer as primitivas do universo de discurso dos sistemas a serem integrados. Como cada termo dos modelos conceituais dos sistemas está definido com base nessas primitivas por meio de relações com a ontologia, a comparação entre eles torna-se mais fácil. Dessa forma, os modelos conceituais dos sistemas são relacionados entre si por meio da ontologia (WACHE, 2001).

O uso de ontologia na integração e interoperabilidade iniciou-se na última década, no contexto da integração de banco de dados (WACHE, 2001), sendo seu uso restrito à integração na camada de dados. Atualmente, ontologias também são usadas no contexto da integração baseada em serviços, principalmente através de Serviços Web Semânticos (IZZA, 2009). Os principais trabalhos relacionados ao apresentado nesta dissertação são apresentados e comparados à abordagem aqui proposta no próximo capítulo.

2.6. Conclusões

Como discutido por Izza (2009), a maioria das abordagens de integração ainda foca na integração de sistemas no nível sintático. Entretanto, a integração no nível semântico tem ganhado mais e mais atenção (IZZA, 2009). Enquanto a integração sintática ocorre no nível estrutural, envolvendo diferentes tecnologias, a integração semântica ocorre no nível de conhecimento (PARK; RAM, 2004), considerando que sistemas devem compartilhar significado.

Embora existam trabalhos que foquem a integração semântica, a semântica ainda é uma importante lacuna que caracteriza a maioria das abordagens (IZZA, 2009). A semântica deve ser tratada corretamente, a fim de permitir que sistemas trabalhem com seu potencial completo. Para isso, ontologias podem constituir um modo interessante de lidar com a heterogeneidade de significado.

A maioria dos autores considera que ontologias representam o coração do futuro da integração de sistemas (IZZA, 2009). Elas podem ser usadas para estabelecer um entendimento comum de um universo de discurso, servindo como uma interlíngua para a comunicação entre sistemas.

Atualmente, a integração semântica vem sendo um dos maiores desafios da integração. Apesar de existirem abordagens que consideram a semântica na integração de sistemas, a maior parte delas foca apenas em uma camada (geralmente a camada de dados e mais recentemente a camada de serviço). Além disso, elas enfocam demasiadamente aspectos tecnológicos usando, por exemplo, tecnologias da Web Semântica para a anotação de dados e serviços web. A maioria dessas abordagens não faz o melhor uso de ontologias na atribuição de semântica.

Na verdade, um dos maiores gargalos da integração semântica é como fazer mapeamentos e é importante que haja melhorias nos aspectos metodológicos da integração. No próximo capítulo, é apresentada uma abordagem de integração semântica que considera as camadas de integração de dados, serviço e processo e que foca na atribuição de significado por meio de ontologias. Essa atribuição é feita no nível conceitual, independentemente de aspectos tecnológicos. Na abordagem proposta, é usado o que Guizzardi (2007) chama de ontologia de referência, ou seja, uma ontologia que é construída com o único objetivo de fazer a melhor descrição de uma porção da realidade, seja ela um domínio ou uma tarefa genérica. Uma ontologia de referência é um tipo especial de modelo conceitual, um artefato de engenharia com o requisito adicional de representar um modelo consensual em uma comunidade. É uma especificação independente de solução, com o propósito de tornar mais clara e precisa a descrição de entidades de domínio, ajudando na comunicação, aprendizado e na solução de problemas.

Capítulo 3 - OBA-SI: Uma Abordagem de Integração Semântica de Sistemas Baseada em Ontologias

Uma das maiores dificuldades na hora de se integrar sistemas ocorre devido à heterogeneidade dos modelos dos sistemas. Isso ocorre, pois geralmente cada sistema é criado separadamente, em contextos distintos e por diferentes desenvolvedores, implementando o seu próprio modelo de dados e de processo. Dificilmente existe uma preocupação com o estabelecimento de um significado comum aos itens comunicados pelos mesmos. Assim, não há uma compatibilidade semântica inerente aos modelos dos sistemas, o que abre espaço para diversos tipos de conflitos sintáticos e semânticos. (IZZA, 2009).

Uma solução para lidar com esse problema é introduzir mediadores que considerem tanto os dados quanto o processo sendo apoiado pelos sistemas envolvidos. Mediadores de dados visam transformar dados enviados por um sistema em dados passíveis de compreensão pelo sistema destino, mantendo a sua semântica. Um mediador de processo, por sua vez, visa solucionar problemas de heterogeneidade de operações e invocação, mantendo a semântica correspondente (IZZA, 2009).

Um fator chave para a integração semântica é ter sistemas compartilhando um entendimento comum do significado dos termos e serviços manipulados pelos sistemas. Para tal, deve-se ter uma representação compartilhada de dados e tarefas do domínio de interesse. Neste contexto, ontologias podem ser usadas como uma interlíngua para mapear conceitos e serviços usados por diferentes sistemas, que acessariam dados e serviços por meio de ontologias compartilhadas.

Conforme discutido no Capítulo 2, muitos estudos enfocam a integração semântica de sistemas, vários deles usando ontologias (IZZA, 2009). Entretanto, muitos desses estudos dão muita atenção a aspectos tecnológicos da solução, o que acaba interferindo na abordagem de integração semântica propriamente dita, já que aspectos tecnológicos estão impregnados nas soluções propostas.

Neste trabalho defende-se a ideia de que a integração semântica é uma tarefa complexa e bastante subjetiva e, por isso, deve ocorrer em um nível mais alto de abstração. Ela deve ser independente de como a solução de integração será projetada e implementada. Como qualquer processo de desenvolvimento de software, o processo de

integração deve começar pelo nível conceitual. Apenas depois de considerar os requisitos da integração, composto dos sistemas a serem integrados, domínio e tarefas envolvidos bem como atividades do processo a serem apoiadas, e construir o modelo conceitual da integração, os aspectos tecnológicos devem ser considerados (fase de projeto).

Tomando por base esta consideração, foi desenvolvida OBA-SI (*Ontology-Based Approach for Semantic Integration*), uma abordagem de integração semântica de sistemas que concentra esforços na modelagem conceitual e na análise dos requisitos de integração. Nessa abordagem, a integração semântica é realizada em um alto nível de abstração, provendo acordo semântico entre os sistemas no nível conceitual. OBA-SI lida com a integração nas três camadas de integração: dados, serviços e processos. Para tal, modelos conceituais dos sistemas (representando sua estrutura e comportamento), bem como do processo de negócio por eles apoiado, são comparados à luz de ontologias, usadas para atribuir semântica aos itens compartilhados entre os sistemas no apoio ao processo de negócio considerado. Os modelos são compatibilizados por meio de mapeamentos entre seus elementos. Todo esse processo de atribuição de semântica e uso de ontologias é independente da solução da integração.

Este capítulo apresenta OBA-SI e está organizado da seguinte forma: a Seção 3.1 apresenta a motivação para o desenvolvimento de OBA-SI; na Seção 3.2 é apresentada uma visão geral da abordagem e seus níveis de abstração; na Seção 3.3 são discutidos os modelos que OBA-SI utiliza no processo de integração; a Seção 3.4 discute os mapeamentos propostos por OBA-SI para atribuição de semântica aos elementos dos modelos; a Seção 3.5 apresenta o processo de integração proposto por OBA-SI, com foco na etapa de análise da integração; na Seção 3.6 é discutido como OBA-SI trabalha as diferentes camadas de integração (dados, serviço e processo); na Seção 3.7 OBA-SI é comparada com outras abordagens de integração semântica; finalmente, na Seção 3.8, são apresentadas as conclusões do capítulo.

3.1. Motivação para uma Abordagem de Integração Semântica de Nível Conceitual

Sistemas são construídos por pessoas e são uma espécie de representação da realidade. A conceituação de um sistema pode variar em função das pessoas envolvidas na sua construção, de seus requisitos ou até mesmo em função de aspectos mais

subjetivos como, por exemplo, a cultura ou a cognição humana. Ao se tentar integrar um conjunto de sistemas, tenta-se fazer com que eles conversem entre si para apoiarem, juntos, um mesmo processo de negócio. A Figura 3.1 ilustra essa situação, na qual dois sistemas se comunicam para juntos apoiarem um processo.

Figura 3.1 – Apoio Automatizado a um Processo de Negócio

Entretanto, um importante problema é que os sistemas geralmente são construídos de modo independente, sem seguir algum tipo de padrão, e por isso não compartilham explicitamente uma conceituação comum a priori. Com isso, não é possível que eles se comuniquem diretamente uns com os outros, pois, nesse caso, não estão claras as concordâncias e distinções entre as conceituações de cada sistema.

A integração semântica de sistemas busca solucionar esse problema por meio da explicitação de uma conceituação comum, possibilitando uma relação mais precisa entre as conceituações de cada sistema. A conceituação comum em uma integração semântica requer o estabelecimento de um vocabulário comum responsável por explicitar o consenso e, portanto, é usada como uma referência para tornar mais claros os significados das conceituações dos sistemas. Com isso, é possível identificar as concordâncias e as distinções de significado entre as conceituações dos sistemas, possibilitando a comunicação. A Figura 3.2 ilustra essa situação, na qual dois sistemas integrados e o processo que eles apoiam compartilham uma conceituação comum, a qual é responsável por atribuir semântica às conceituações específicas de cada um deles.

O ideal seria que os sistemas, ao serem construídos, se baseassem em uma conceituação compartilhada, facilitando uma possível integração semântica. Como isso tipicamente não ocorre, geralmente ela deve ser estabelecida a posteriori, a partir das conceituações dos sistemas já desenvolvidos. Entretanto a construção de uma conceituação comum para uma integração não é trivial. Por exemplo, ela não pode ser nem muito abstrata nem detalhada excessivamente. Caso seja muito abstrata, ela não será capaz de prover distinções de maneira satisfatória. Assim a conceituação dará margem à ocorrência de interpretações duplas, falsas concordâncias e ausências de

conceitos considerados pelos sistemas. Entretanto, também não pode ser muito detalhada, pois dessa forma torna mais difícil a obtenção de consenso.

Figura 3.2 – Atribuição de Semântica aos Sistemas e ao Processo por meio de uma Conceituação Comum

Com relação à conceituação comum, pode-se considerar basicamente a existência de dois níveis: um relativo à conceituação mais geral, que é mais abstrato e independente de iniciativas de integração, e outro mais específico, relativo ao escopo de uma iniciativa de integração específica, como ilustra a Figura 3.3. Na figura, a conceituação geral pode ser referente a tarefas genéricas ou a um domínio de discurso, onde se realizam as integrações representadas e cada conceituação específica considera particularidades de uma iniciativa de integração (requisitos, sistemas e processos a serem apoiados).

Ao integrar um conjunto de sistemas, são envolvidos diferentes tipos de preocupações. Primeiramente a conceituação referente a cada sistema deve ser compatibilizada por meio de uma conceituação comum. Essa tarefa é bastante subjetiva e pode ser bastante complexa em função da complexidade do domínio onde os sistemas estão inseridos. Além disso, essa integração semântica deve ser implementada para que permita efetivamente a comunicação dos sistemas em si. Como a integração semântica engloba o desenvolvimento de um produto de software, atividades como levantamento de requisitos, análise da integração, projeto, implementação, teste e implantação são importantes.

Assim, a integração semântica não pode ser realizada de qualquer maneira. É importante que se siga um método, com etapas claras, de modo que ela ajude na realização de tal tarefa, diminuindo, assim, a subjetividade e a complexidade. É

importante que o método ajude a integração semântica tanto na compatibilização de conceituações dos sistemas, quanto no desenvolvimento do produto de software responsável por materializar a integração fornecendo, entre outros, etapas bem estabelecidas, separação de preocupações, diminuição da subjetividade e redução de problemas de entendimento e comunicação. Com o intuito de ajudar a minimizar alguns dos problemas inerentes à integração semântica, foi proposta OBA-SI.

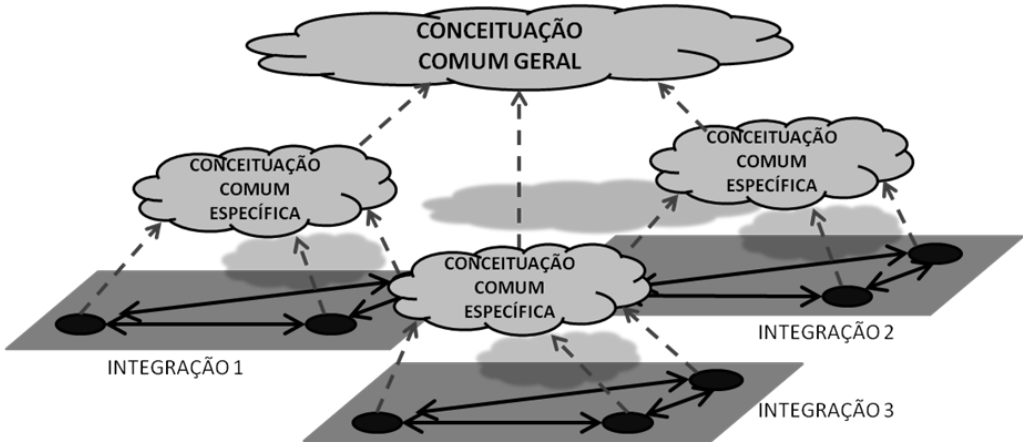

Figura 3.3 - Tipos de conceituações comuns a serem explicitadas na integração semântica em OBA-SI

3.2. Visão em Níveis de Abstração de OBA-SI

OBA-SI é uma abordagem de integração semântica de sistemas baseada em ontologias. Como ilustra a Figura 3.4, OBA-SI trata a integração em diferentes níveis de abstração, iniciando-se no nível superior, o mais abstrato e independente da integração, até chegar ao inferior, onde a integração é projetada e implementada. Esses níveis procuram destacar a separação de interesses que a abordagem proporciona. Na figura, cada seta representa a projeção de um nível superior em um nível inferior, levando em consideração novas preocupações. Desses três níveis, o superior, que está acima da linha pontilhada, é independente da iniciativa de integração em questão, enquanto os dois inferiores, abaixo dessa linha, incorporam preocupações específicas (sistemas, processos e requisitos no nível de integração semântica e tecnologia no nível de projeto e implementação). Além disso, os dois níveis mais altos (em cinza claro) são níveis conceituais, e portanto independentes de tecnologia, enquanto o último nível não.

O nível ontológico fornece os modelos de referência, os quais são independentes da iniciativa de integração, a saber ontologias de domínio e ontologias de tarefa. Nesse

nível, esses modelos de referência devem ser integrados para servir de base para a integração semântica no nível subjacente.

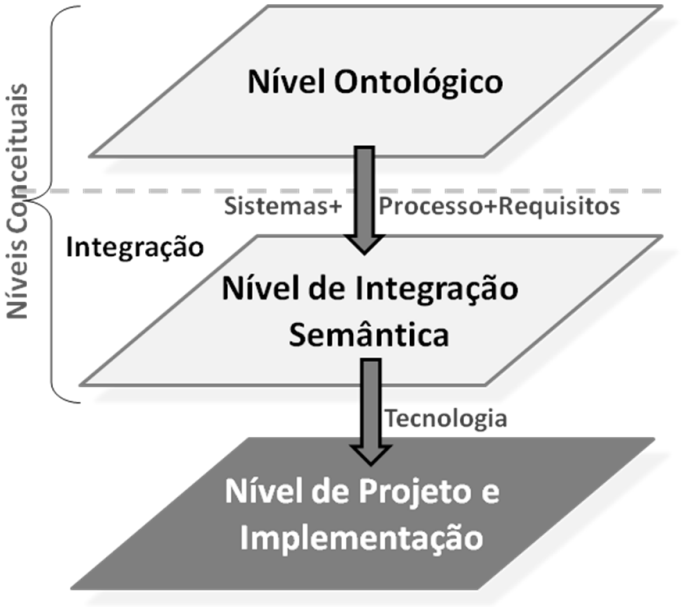

Figura 3.4 – Níveis de Integração em OBA-SI

O nível de integração semântica, por sua vez, adiciona novas preocupações: os sistemas a serem integrados, o processo de negócio a ser apoiado e os requisitos a serem satisfeitos. O foco é a atribuição de semântica, de modo a descrever conceitualmente os sistemas e o processo de negócio, bem como a conceituação comum relativa à integração. Por meio dos requisitos, o escopo de atuação de cada sistema é delimitado. A representação das entidades nesse nível é conceitualmente rica, pois é focada para a solução de conflitos semânticos. Esse nível é, portanto, focado no entendimento do significado atribuído pelos sistemas, pelo processo e pelo modelo de integração como um todo. Por fim, o último nível é referente ao projeto e à implementação da integração. Nesse nível estão presentes os sistemas e o processo e é nele que a integração deve ser concretizada usando a tecnologia estabelecida.

É interessante notar a separação de interesses proporcionada por OBA-SI. À medida que se desce o nível de abstração da integração, novas preocupações vão sendo adicionadas, o que promove alterações nos modelos pertencentes ao nível anterior. No início, a única preocupação é descrever e integrar o domínio e as tarefas genéricas relacionadas. Depois, preocupa-se também com o entendimento da semântica dos

sistemas e do processo como um todo. Por fim, são considerados os aspectos tecnológicos.

3.3. Modelos de OBA-SI

Pode-se pensar que um dos principais objetivos da integração semântica é o estabelecimento de uma *conceituação comum* que permita a comunicação entre os sistemas de modo satisfatório. Essa conceituação comum deve compatibilizar as diferentes conceituações envolvidas na integração, dentre elas as dos sistemas envolvidos e do processo de negócio considerado.

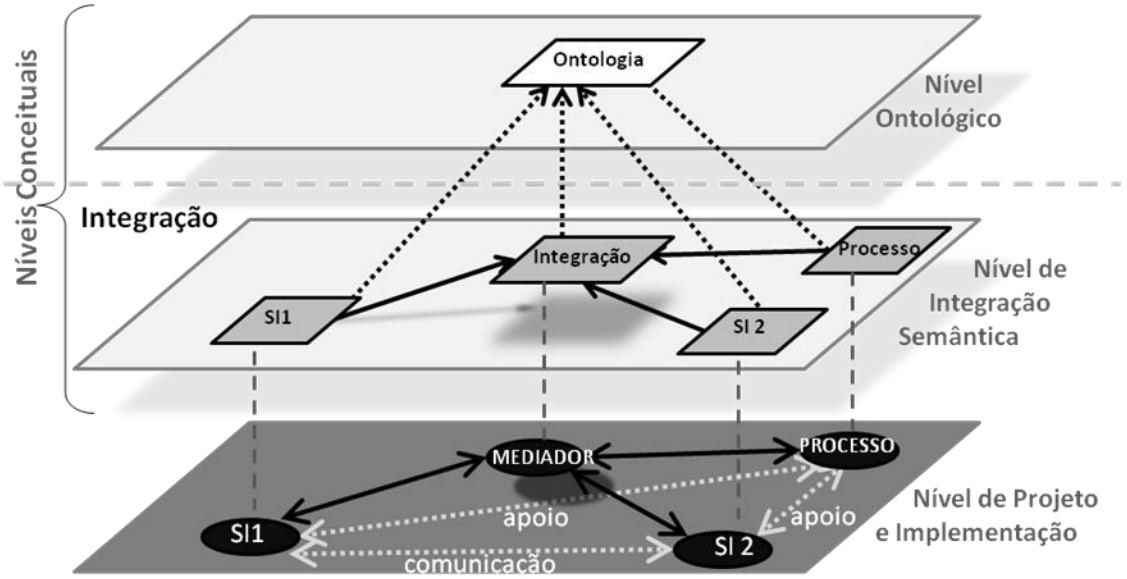

Figura 3.5 - Visão Geral de OBA-SI

Entretanto, essas conceituações geralmente estão implícitas, o que dificulta a sua compatibilização. Para facilitar a integração, em OBA-SI, são usados modelos com o intuito de explicitar as conceituações envolvidas. Eles são compostos basicamente de conceitos, relacionamentos e propriedades na modelagem estrutural e atividades, insumos, produtos e atores na modelagem comportamental. Os modelos de OBA-SI são modelos conceituais, já que visam explicitar conceituações envolvidas na integração durante a etapa de análise da integração e são independentes de aspectos tecnológicos.

Em OBA-SI, a integração é desenvolvida usando-se diferentes modelos em cada nível de abstração, como mostra a Figura 3.5. Os modelos propostos em OBA-SI possuem diferentes funções, a saber:

(i) *Modelos de Referência*: são os modelos de OBA-SI que explicitam as conceituações comuns e são usados para atribuir significado a elementos dos modelos de sistemas e processo. Eles estabelecem um vocabulário a ser tomado como base durante a integração, envolvendo tanto aspectos estruturais quanto comportamentais.

   a. *Ontologias*: modelos pertencentes ao nível ontológico, usados para representar a conceituação comum mais geral, referente ao domínio de aplicação e às tarefas genéricas envolvidas no universo de discurso dos sistemas e processo a serem integrados. As ontologias de domínio são utilizadas na integração semântica na camada de dados (entidades e relacionamentos dos sistemas), enquanto as ontologias de tarefa são usadas na integração semântica nas camadas de serviço e de processo.

   b. *Modelos de Integração*: são modelos baseados nas ontologias, mas pertencentes ao nível de integração semântica. Eles são usados para representar a conceituação mais específica de uma iniciativa de integração, considerando particularidades da mesma, envolvendo as conceituações dos sistemas a serem integrados, do processo a ser apoiado e considerando os requisitos da integração. São usados como referência para anotar semanticamente outros modelos usados na integração.

(ii) *Modelos de Entidades de Integração (MEI)*: são modelos conceituais que descrevem os elementos dos sistemas e do processo considerados na integração. Seu intuito é permitir a integração de sistemas e processo no nível conceitual. Representam apenas a parte de uma conceituação de um sistema ou processo que está envolvida e será compartilhada com outros sistemas em uma iniciativa de integração.

   a. *Modelos Conceituais de Sistemas* (MCS): descrevem informações relativas aos sistemas a serem integrados (funcionalidades e dados). A parte estrutural do modelo conceitual de sistema é responsável por representar

conceitos e relações com base nos dados manipulados pelo sistema. Já a parte comportamental do modelo conceitual de sistema deve se preocupar com o comportamento geral do sistema, em especial, a ordem em que as funcionalidades são executadas, tornando explícito o processo que é apoiado pelo sistema.

b. *Modelo Conceitual do Processo* (MCP): Descreve o processo da organização a ser apoiado com foco na perspectiva comportamental, representando as atividades que compõem o processo, bem como os ativos de processo envolvidos, tais como artefatos consumidos e produzidos e os recursos utilizados.

No contexto deste trabalho, é importante deixar clara a distinção realizada entre ontologias e modelos de integração semântica. Ontologias são modelos mais gerais pertencentes ao nível ontológico e por isso são “externas” à iniciativa de integração. Por serem mais gerais, elas podem ser reutilizadas em várias iniciativas de integração que venham a ocorrer no mesmo universo de discurso.

Entretanto, cada iniciativa de integração possui suas particularidades que variam de acordo com os sistemas, o processo e os requisitos da integração. Devido a essas particularidades, as ontologias podem não ser suficientes para detalhar a integração em um nível semanticamente adequado. Por essa razão, OBA-SI propõe a utilização dos *Modelos de Integração*.

A Figura 3.6 ilustra como os modelos de referência e os modelos de integração estão relacionados em OBA-SI. Conforme ilustrado, ontologias são usadas como referências em várias iniciativas de integração. Entretanto, em cada iniciativa de integração, também é usado um modelo de integração que considere suas particularidades.

O modelo de integração é construído com base tanto nas ontologias (construção *top-down*) quanto nos modelos conceituais de sistemas e no modelo de processo (construção *bottom-up*), sempre levando em conta os requisitos da integração. Ele também pode ser baseado no conhecimento do próprio analista de integração e dos especialistas de domínio. A estratégia *top-down* incorpora a semântica geral das ontologias. Já a estratégia *bottom-up* permite considerar particularidades dos sistemas e

do processo na representação da conceituação comum da integração. Através dela ocorre o detalhamento semântico em um nível suficiente para que os sistemas possam conversar sem que haja conflitos. Durante a construção do modelo de integração, deve-se procurar solucionar possíveis conflitos semânticos existentes entre os sistemas e o processo.

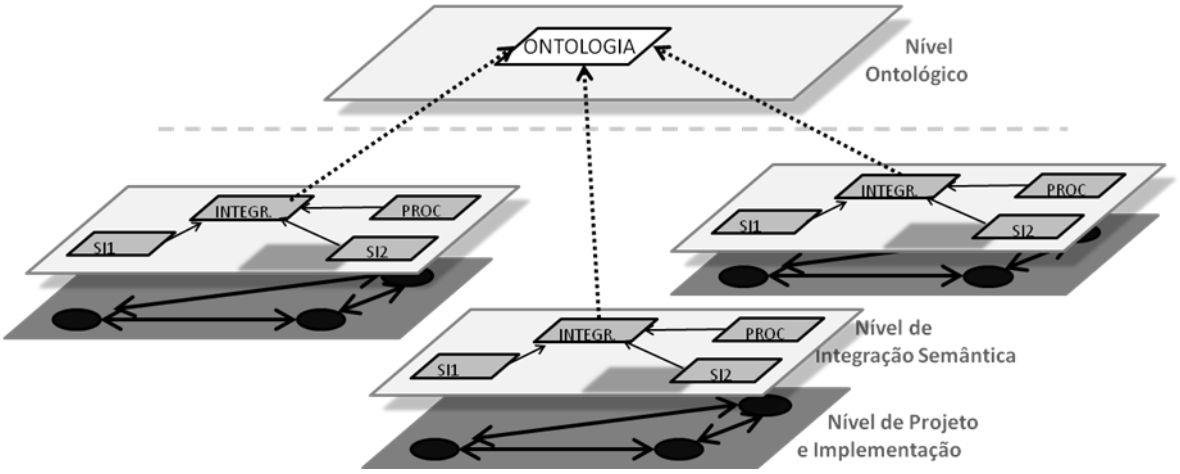

Figura 3.6 - Relação entre modelos envolvidos no nível semântico em OBA-SI

O modelo de integração é, portanto, uma espécie de adaptação da ontologia considerando particularidades da integração. Adaptação não apenas no sentido de detalhar mais, mas também no sentido de abstrair algumas informações também, em função dos requisitos da integração. Por ser mais específico, o modelo de integração é de difícil reutilização, ao contrário das ontologias. Dessa forma, cada iniciativa de integração deve desenvolver seu próprio modelo de integração.

Como visto nesta seção, modelos conceituais são usados em OBA-SI para descrever sistemas e processos a serem integrados. Entretanto, eles por si só não resolvem os conflitos semânticos. Para tratar esta questão, OBA-SI sugere o uso de mapeamentos para atribuir semântica aos sistemas e ao processo.

3.4. Mapeamentos Semânticos

O propósito de um mapeamento semântico é relacionar um elemento de modelo a outro que represente o seu significado. Assim, para explicitar o significado dos elementos dos modelos de sistemas e do modelo de processo, eles são mapeados para os elementos dos modelos de referência. Como em OBA-SI há dois tipos de modelos de

referência, ontologias e modelos de integração, os mapeamentos são também categorizados em dois tipos: mapeamentos verticais e mapeamentos horizontais.

Os mapeamentos verticais são responsáveis por atribuir significado aos modelos do nível de integração semântica (modelos conceituais dos sistemas, modelo conceitual do processo e o modelo de integração) com base em ontologias. Eles são responsáveis por atribuir um significado genérico e independente da iniciativa de integração específica. Eles são chamados de verticais, pois ocorrem entre os elementos pertencentes ao nível da integração semântica e os elementos do nível acima, o nível ontológico, e, portanto, os mapeamentos verticais tomam por base uma fonte externa ao contexto da integração. É bom salientar que, por estarem relacionados com as ontologias, eles também são independentes do cenário de integração e com isso podem ser reutilizados.

Já os mapeamentos horizontais são responsáveis por atribuir significado aos modelos de sistemas e do processo com base no modelo de integração. Eles são responsáveis por atribuir um significado mais específico, considerando as particularidades da integração. Eles são, portanto, focados no cenário de integração específico, sem, no entanto, desconsiderar a semântica mais geral. Por isso, em OBA-SI, advoga-se que os mapeamentos horizontais sejam baseados nos mapeamentos verticais, ainda que isso não ocorra necessariamente em todos os casos, já que os mapeamentos horizontais são focados em um cenário de integração específico. Eles são chamados de horizontais justamente por serem uma atribuição semântica que ocorre entre modelos de um mesmo nível.

A Figura 3.7, na parte (1), ilustra um exemplo de mapeamento vertical. Nela são apresentados uma ontologia, dois modelos conceituais de sistemas (referentes aos sistemas A e B), um modelo de processo e um modelo de integração. Nessa ilustração, por exemplo, os elementos *a1* e *b1* dos sistemas *A* e *B*, respectivamente, possuem o mesmo significado geral, representado pelo elemento *o1* da ontologia. Na parte (2), os elementos *a1* e *b1* (dos sistemas A e B) estão relacionados ao elemento *i1* do modelo de integração, por meio dos mapeamentos horizontais (*a1*, *i1*) e (*b1*, *i1*). Eles, portanto, possuem o mesmo significado específico (mapeamento horizontal), representado por *i1*. Nesse exemplo, *a1* e *b1* também possuem o mesmo significado geral.

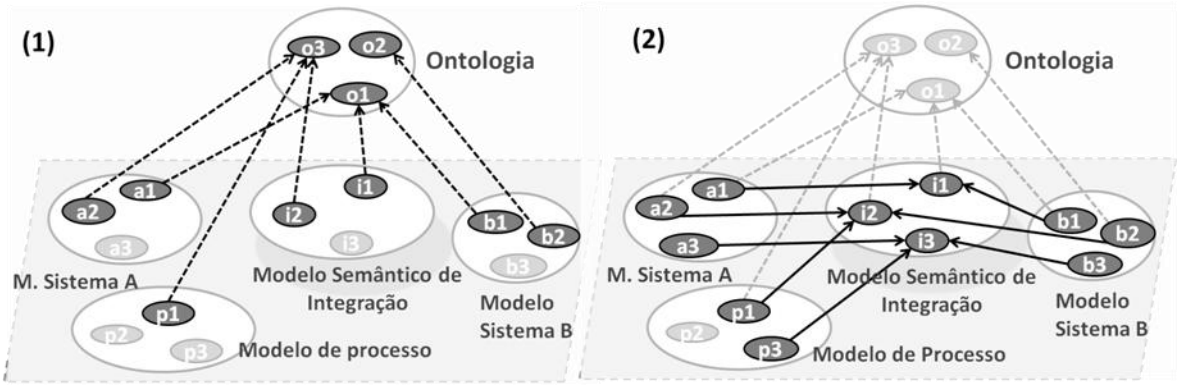

Figura 3.7 – (1) Mapeamentos Verticais e (2) Mapeamentos Horizontais

3.5. Processo de Integração Semântica

Ao se realizar uma iniciativa de integração é gerado um produto de software. Por meio dele o conjunto de sistemas é integrado. Eles passam então a se comportar como um todo, como se fossem um único sistema. Uma questão importante a ser tratada por uma abordagem de integração semântica de sistemas é a definição de um processo que conduza o desenvolvedor na realização desta tarefa. Uma iniciativa de integração deve gerar um produto de software, que foi analisado, projetado e implementado. Dessa forma, a abordagem de integração semântica não deve considerar essas etapas de modo isolado. Ao contrário, a integração de sistemas pode ser vista como um tipo de processo de desenvolvimento de software, composto de fases de levantamento de requisitos, análise, projeto, implementação, testes e implantação, como ilustra a Figura 3.8.

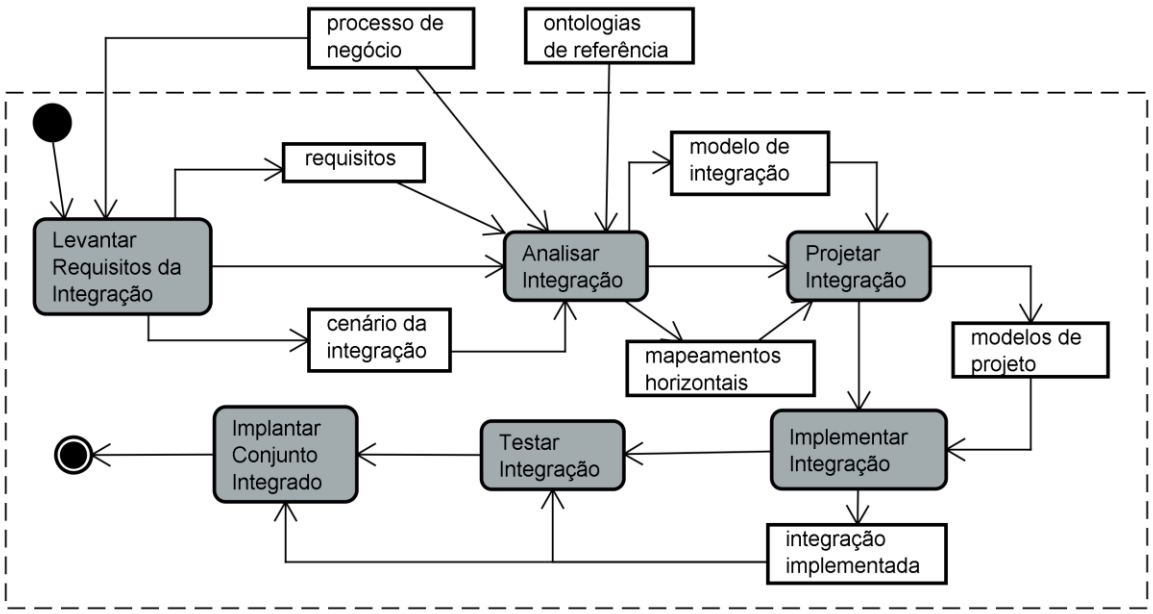

Figura 3.8 – Processo de Integração Semântica

OBA-SI possui como premissa que a integração deve ocorrer primeiramente no nível semântico. Isto porque os sistemas devem "concordar" semanticamente antes de serem integrados. Assim, em OBA-SI, diferentemente de um processo de desenvolvimento convencional, antes que a integração seja projetada e implementada, é realizada a integração no nível semântico, por meio dos mapeamentos semânticos. Por isso, OBA-SI é centrada na atividade de análise da integração, onde a semântica deve ser definida. Por conseguinte, ela fornece apenas orientações para as outras fases do processo de desenvolvimento. É importante salientar que o processo de integração apresentado não é estritamente sequencial como está representado. Ele também pode ser realizado de modo iterativo, principalmente nas atividades que envolvem o levantamento de requisitos e análise da integração.

3.5.1. Levantamento de Requisitos da Integração

Como mostra a Figura 3.8, o processo de integração começa com a etapa de levantamento de requisitos da integração, quando os requisitos e objetivos da integração devem ser estabelecidos. A principal questão a ser respondida é: por que é necessário integrar sistemas? Quais são os processos de negócio que serão apoiados pelos sistemas integrados? Baseado nos objetivos, os requisitos podem ser levantados. Essa atividade é responsável por definir o **cenário de integração**, indicando:

A. As atividades do processo de negócio que serão apoiadas;
B. Os sistemas a serem integrados para apoiar essas atividades;
C. Os domínios envolvidos no cenário de integração e as tarefas genéricas relacionadas à integração.

Essa atividade utiliza o modelo de processo de negócio e gera como produtos o cenário de integração e os requisitos. Ela é realizada pelo analista da integração, podendo ter a participação de clientes e especialistas de domínio.

As principais informações dessa atividade são resumidas abaixo:

- ***Objetivo***: Definir o cenário de integração e estabelecer o escopo da iniciativa de integração;
- ***Insumo(s)***: modelo de processo de negócio;
- ***Produto(s)***: cenário de integração e requisitos da integração;
- ***Participante(s)***: analista da integração;

Para ilustrar o processo de integração, suponha um cenário hipotético no qual se deseja integrar dois sistemas de biblioteca de duas instituições de ensino que se fundiram. Para esse exemplo, o cenário de integração é ilustrado na Tabela 3.1, apresentando os sistemas a serem integrados, as atividades a serem apoiadas, bem como os domínio e tarefas genéricas envolvidas.

Tabela 3.1 – Exemplificação de um Cenário de Integração

CENÁRIO DE INTEGRAÇÃO	
SISTEMAS	Sistemas de Bibliotecas das Instituições A e B
ATIVIDADES	Realizar Empréstimo de Itens, Devolver Itens Emprestados, Efetuar Notificação de Entrega, Realizar Pagamento de Multas
DOMÍNIOS/TAREFAS ENVOLVIDAS	Domínio de Instituição de Ensino, Domínio de Biblioteca, Tarefa de Empréstimo, Tarefa de Notificação, Tarefa de Pagamento

Ambos os sistemas apoiam, dentre uma gama de funcionalidades, as funcionalidades envolvidas na integração capturadas no modelo de casos de uso. Tais funcionalidades devem ser integradas, entretanto as conceituações referentes a elas possuem algumas distinções que deverão ser compatibilizadas por meio da integração semântica.

3.5.2. Análise da Integração

Depois do levantamento de requisitos da integração, é realizada a análise da integração. De maneira análoga à fase de análise de requisitos tradicional, a análise da integração é responsável por analisar e modelar os requisitos da integração, especificar as funcionalidades que serão providas e os conceitos envolvidos, bem como estabelecer como será o comportamento geral do conjunto integrado de sistemas. Seu principal produto é o modelo da integração. Os modelos de análise no contexto da abordagem OBA-SI são modelos conceituais, como os modelos de análise tradicionais, e baseados nos requisitos da iniciativa de integração. Eles também servem de base para o projeto e implementação da integração.

Entretanto, diferentemente dos modelos de análise tradicionais, que são baseados apenas nos requisitos, os modelos de análise em OBA-SI devem estar de acordo com as ontologias para que a integração semântica seja facilitada. Além disso, devido à utilização de um conjunto de sistemas, aspectos relacionados a eles também devem ser considerados na construção do modelo de análise. A satisfação dos requisitos levantados deve ocorrer de forma a se adequar ao cenário de integração para que não haja uma distância grande entre o que se quer (requisitos) e o que se tem (sistemas). O modelo de integração é, portanto, construído seguindo uma mistura de estratégias *top-down* e *bottom-up*.

A fase de análise em OBA-SI também tem como objetivo a realização da integração semântica no nível conceitual, ou seja, estabelecer as equivalências semânticas entre sistemas e processo. Para isso, o modelo de integração é utilizado para atribuir semântica aos modelos conceituais de sistemas e do processo por meio de mapeamentos semânticos, à luz de ontologias de referência. Esse é o grande diferencial da fase de análise em OBA-SI. A Figura 3.9 apresenta as atividades envolvidas nesta fase.

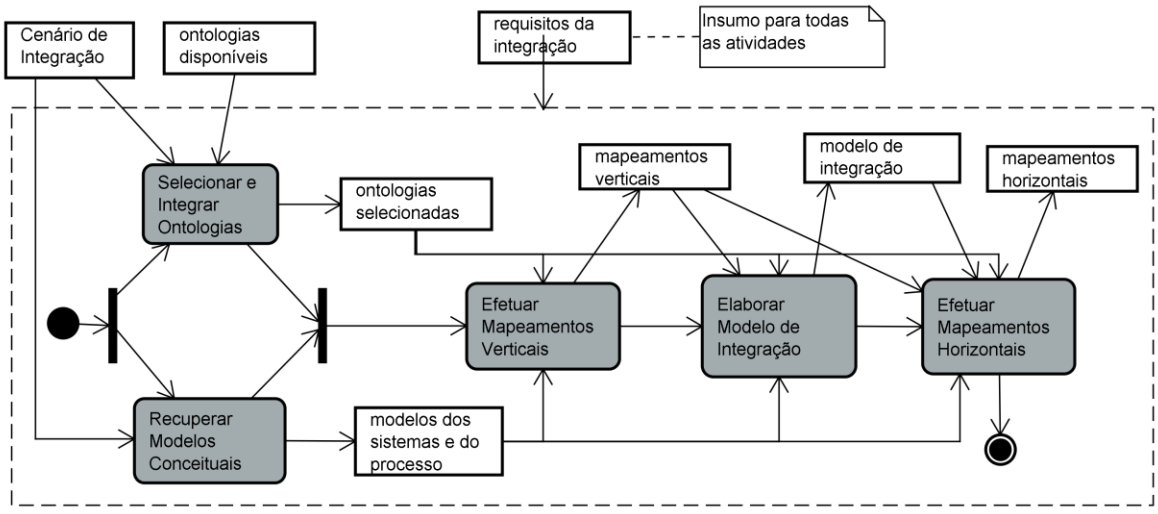

Figura 3.9 - Atividades da Fase de Análise da Integração

Recuperar Modelos Conceituais

Esta etapa é responsável por recuperar os modelos conceituais dos sistemas listados no cenário de integração, bem como do processo de negócio a ser apoiado. Não é necessário recuperar todo o modelo conceitual dos sistemas envolvidos, mas apenas a parte deles envolvida no cenário de integração. Para se determinar a parte dos sistemas cujo modelo conceitual será recuperado, tomam-se como base os requisitos da integração e casos de uso descritos no levantamento de requisitos da integração.

As principais informações dessa atividade são resumidas abaixo:

- ***Objetivo***: Essa etapa é responsável por recuperar os modelos conceituais dos sistemas listados no cenário de integração, bem como do processo de negócio;
- ***Insumo(s)***: cenário de integração, requisitos da integração;
- ***Produto(s)***: modelos conceituais dos sistemas e do processo;
- ***Participante(s)***: analista da integração;

No exemplo hipotético da integração dos sistemas de biblioteca, deseja-se integrar as funcionalidades relativas à realização de empréstimo, de notificações de entrega e de gerência de multas. Com base nesse escopo, seriam recuperadas as parte dos modelos conceituais correspondentes dos dois sistemas. A Figura 3.10 apresenta parte dos

modelos conceituais dos dois sistemas de bibliotecas a serem integrados (apenas a parte estrutural está sendo considerada, para facilitar a exemplificação).

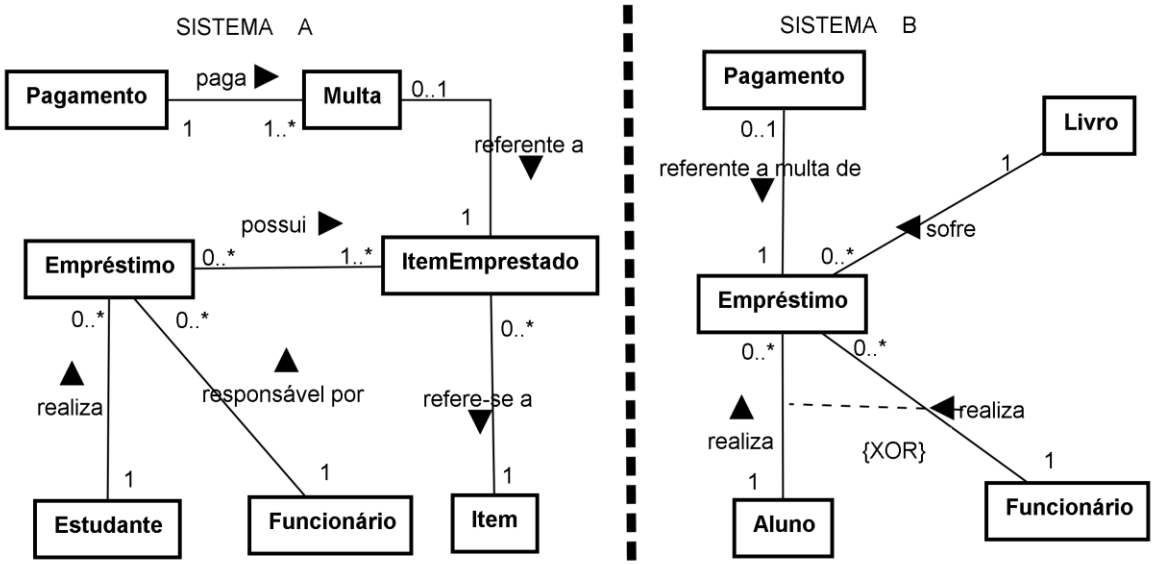

Figura 3.10 - Modelos Conceituais dos Sistemas A (à esquerda) e Sistema B (à direita)

Como mostra o modelo do sistema A, empréstimos de itens da biblioteca são realizados por estudantes possuindo como responsável um funcionário. Empréstimos podem sofrer multa, caso sejam devolvidos com atraso. Já o modelo do sistema B representa que alunos e funcionários podem realizar empréstimos de livros. Caso haja atraso na devolução, pagamentos de multas de empréstimos são realizados.

Selecionar e Integrar Ontologias

Essa etapa é responsável por selecionar as ontologias de referência a serem adotadas, considerando o domínio estabelecido no cenário de integração, bem como as tarefas genéricas identificadas. Em princípio, devem-se utilizar ontologias existentes. Entretanto, se não houver ontologias de referência disponíveis, elas deverão ser desenvolvidas. Depois de selecionadas, as ontologias de diferentes domínios e de diferentes tarefas devem ser integradas entre si. Uma vez que não existem muitas ontologias de referência disponíveis, no caso de as ontologias não existirem, elas devem ser desenvolvidas.

As principais informações dessa atividade são resumidas abaixo:

- ***Objetivo***: selecionar e integrar ontologias de referência para a integração;
- ***Insumo(s)***: cenário de integração, requisitos da integração, ontologias disponíveis;
- ***Produto(s)***: ontologias selecionadas e integradas;
- ***Participante(s)***: analista da integração, especialistas de domínio;

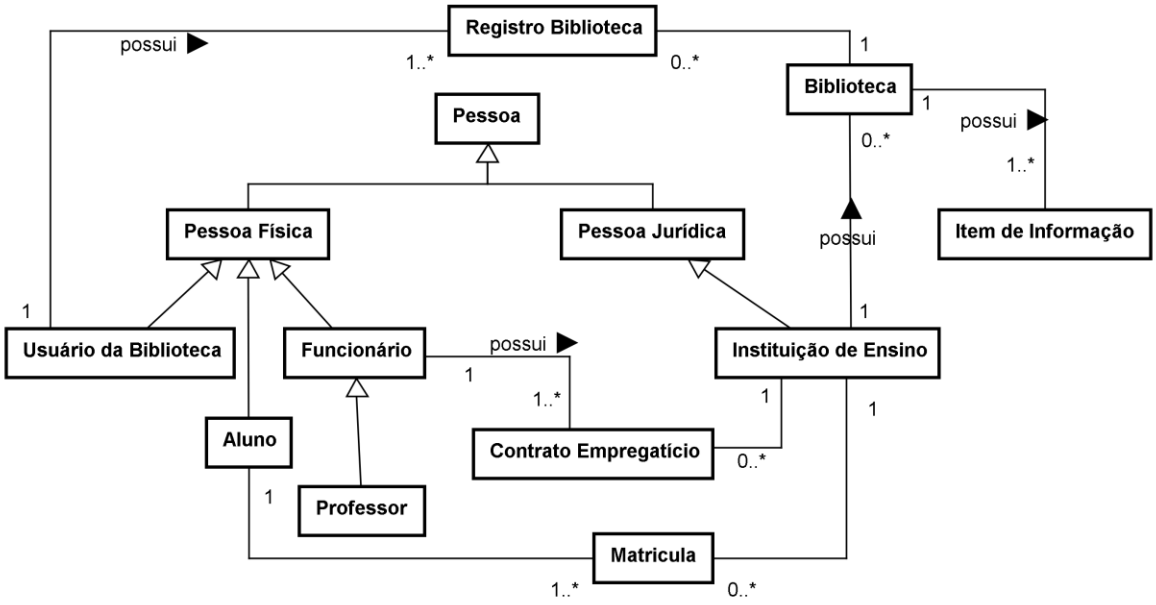

Figura 3.11- Ontologia de Domínio Selecionada para Integração

No exemplo de integração de sistemas de biblioteca, devem-se selecionar ontologias que envolvam os domínios e tarefas listados no cenário de integração. Além disso, as ontologias devem envolver os principais conceitos identificados na atividade “Recuperar Modelos Conceituais”, tais como *Funcionário*, *Estudante*, *Pagamento* e *Empréstimo*. No exemplo foi selecionada uma ontologia de domínio de instituição de ensino, apresentada na Figura 3.11 e descrita na, e uma ontologia de tarefa referente à tarefa de empréstimo, apresentada na Figura 3.12 e descrita na Tabela 3.3.

Tabela 3.2 – Descrição dos conceitos da ontologia de domínio

CONCEITO	DESCRIÇAO
Pessoa	Ser que têm direitos e obrigações
Pessoa Física	Ser humano que têm direitos e obrigações
Pessoa Jurídica	Ser social abstrato que têm direitos e obrigações
Aluno	Pessoa física vinculada a uma instituição de ensino por meio de uma matrícula
Funcionário	Pessoa física vinculada a uma instituição de ensino por meio de um contrato
Professor	Tipo de funcionário de uma instituição
Usuário da Biblioteca	Pessoa física que está autorizada a realizar empréstimos em uma biblioteca
Instituição de Ensino	Tipo de pessoa jurídica
Contrato	Formalização do vínculo empregatício entre funcionário e a instituição de ensino
Matrícula	Formalização do vínculo de um aluno com uma instituição de ensino
Biblioteca	Unidade organizacional responsável por gerenciar itens de informação da instituição de ensino
Registro Biblioteca	Formalização do registro de uma usuário da biblioteca

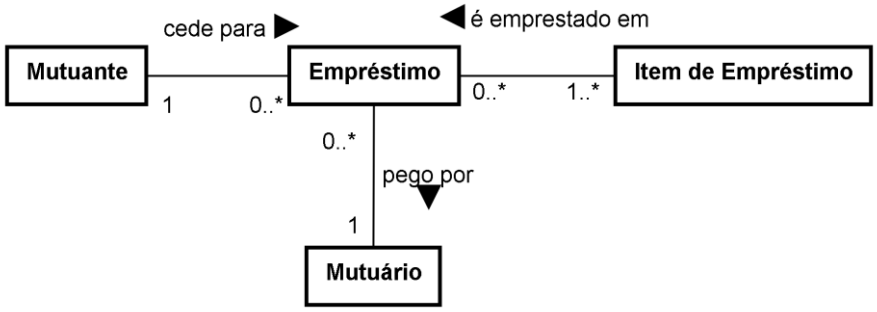

Figura 3.12 - Ontologia de Tarefa Selecionada para Integração (Apenas modelo estrutural)

Tais ontologias não envolvem todos os domínios e tarefas requeridos, mas assim mesmo foram integradas para servir de referência para a integração semântica dos sistemas de biblioteca. A Figura 3.13 ilustra o resultado da integração das ontologias selecionadas. Ela foi gerada da junção de uma ontologia de domínio de Instituição de Ensino e uma ontologia de tarefa de Empréstimo (conceitos em cinza). Entretanto, apenas a parte estrutural dela está sendo apresentada, para simplificar a exemplificação. Através da integração da ontologia de tarefa com a ontologia de domínio é estabelecido quem desempenha os papéis representados na ontologia de tarefa.

Tabela 3.3 – Descrição dos conceitos das ontologias de tarefa selecionada

CONCEITO	DESCRIÇÃO
Item de Empréstimo	Coisa que pode ser emprestada
Mutuário	Aquele que recebe qualquer coisa por empréstimo.
Mutuante	Aquele que cede alguma coisa por empréstimo
Empréstimo	Compromisso entre duas partes formalizando que um determinado item está sendo emprestado ao mutuário pelo mutuante do empréstimo.

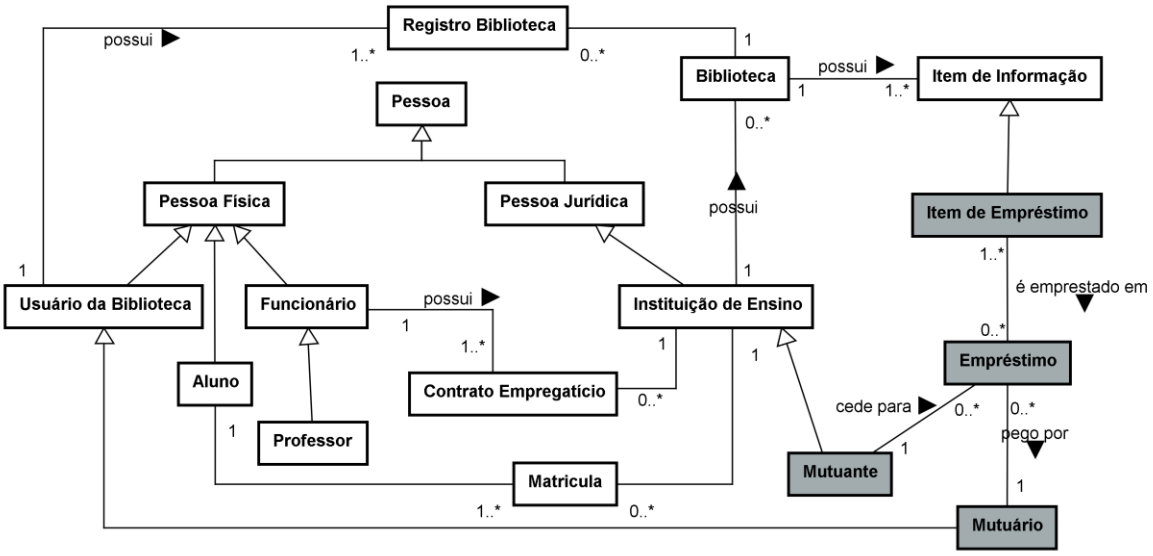

Figura 3.13 – Integração das ontologias de referência selecionadas para integração (ontologia de tarefa em destaque)

Efetuar Mapeamentos Verticais

Esta etapa tem por objetivo atribuir semântica aos modelos de sistemas por meio de mapeamentos verticais entre seus elementos de modelo e os elementos de modelo das ontologias de referência. Cada conceito é mapeado separadamente e depois são comparados para se reavaliar as equivalências.

As principais informações dessa atividade são resumidas abaixo:

- ***Objetivo***: atribuir semântica aos modelos conceituais dos sistemas e do processo com base em ontologias;

- ***Insumo(s)***: requisitos da integração, ontologias disponíveis, modelos conceituais dos sistemas e processo;
- ***Produto(s)***: conjunto de mapeamentos verticais;
- ***Participante(s)***: analista da integração, especialistas de domínio;

No caso hipotético considerado como exemplo, os elementos de modelo *Empréstimo* (Sistema A) e *Empréstimo* (Sistema B) foram mapeados para o conceito *Empréstimo* (Ontologia), como mostra a Tabela 3.4. Os elementos de modelo *Pagamento* (Sistemas A e B) e *Livro* (Sistema B), destacados de cinza na tabela, não receberam mapeamento vertical, pois não foram cobertos pela ontologia. O elemento *Item* (Sistema A) foi mapeado para o conceito *Item de Empréstimo* (Ontologia) e, por fim, os elementos *Estudante* (Sistema A) e *Aluno* (Sistema B) foram mapeados para o conceito *Aluno* (Ontologia).

Apesar de os mapeamentos verticais explicitarem a conceituação de cada elemento, eles podem não ser suficientes para justificar equivalências ou identificar conflitos, como ocorre no caso das linhas destacadas em cinza escuro na tabela.

Tabela 3.4 – Ilustração dos Mapeamentos Verticais

SISTEMA A	SISTEMA B	ONTOLOGIA
Empréstimo	Empréstimo -	Empréstimo
Pagamento	Pagamento	-
Funcionário	Funcionário	Funcionário
-	Livro	-
Item	-	Item de Empréstimo
Estudante	Aluno	Aluno

Elaborar Modelo de Integração

Esta etapa refere-se à construção do modelo de integração. Ele é construído de modo que cada elemento dos modelos conceituais dos sistemas e processo que estejam sem mapeamento vertical receba um significado. Ele pode refinar alguns conceitos da ontologia, caso seja necessária uma distinção semântica maior.

As principais informações dessa atividade são resumidas abaixo:

- ***Objetivo***: construir o modelo de integração;

- ***Insumo(s)***: requisitos da integração, mapeamentos verticais, ontologias, modelos conceituais dos sistemas e do processo;
- ***Produto(s)***: modelo de integração;
- ***Participante(s)***: analista da integração, especialistas de domínio, clientes;

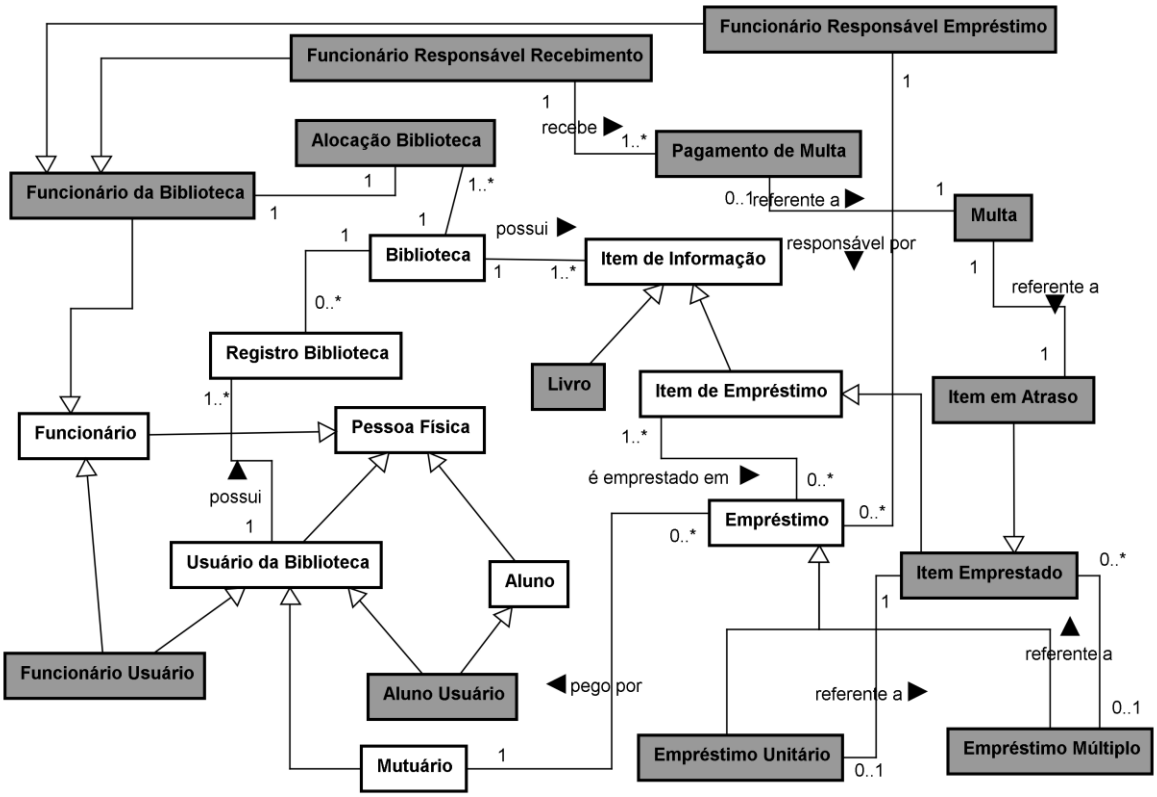

Figura 3.14 –Modelo de Integração (conceitos adicionais em destaque)

Dando prosseguimento ao exemplo, a ontologia não foi suficiente para cobrir os elementos de modelo *Pagamento* e *Livro*, que deverão ser cobertos pelo modelo de integração. Para envolver tais elementos, foram adicionados os conceitos *Pagamento de Multa* e *Livro* no modelo de integração, como mostra a Figura 3.14. (alguns conceitos da ontologia foram omitidos).

Já os elementos *Empréstimo* (dos sistemas A e B) e *Funcionário* (dos sistemas A e B), apesar de terem recebido mapeamentos verticais, suas conceituações não foram suficientemente explicitadas. Por exemplo, o elemento *Empréstimo* no sistema A é referente a um conjunto de itens, enquanto no sistema B é referente a um único livro. *Funcionário* no Sistema A é responsável por um empréstimo, enquanto que no Sistema B

ele pode realizar empréstimos. Para tornar essas distinções mais claras, foram adicionados os conceitos *Funcionário da Biblioteca*, *Funcionário Responsável por Empréstimo*, *Funcionário Usuário, Empréstimo Unitário e Empréstimo Múltiplo* no modelo de integração. Além deles, foram adicionados os conceitos *Multa*, *Item em Atraso* e *Funcionário Responsável por Recebimento* para tratar adequadamente o conceito *Multa* do Sistema A.

Efetuar Mapeamentos Horizontais

A última tarefa da análise da integração é a realização de mapeamentos horizontais. Nessa etapa, cada elemento dos modelos conceituais dos sistemas e do modelo de processo deve ser mapeado para um elemento correspondente do modelo de integração. Todos os elementos dos modelos conceituais dos sistemas e processo devem ser mapeados.

As principais informações dessa atividade são resumidas abaixo:

- ***Objetivo***: realizar mapeamentos horizontais;
- ***Insumo(s)***: requisitos da integração, mapeamentos verticais, modelo de integração, modelos conceituais dos sistemas e do processo;
- ***Produto(s)***: mapeamentos horizontais;
- ***Participante(s)***: analista da integração, especialista de domínio, clientes;

A Tabela 3.5 ilustra os mapeamentos horizontais realizados no exemplo da integração dos sistemas de biblioteca. Como a tabela mostra, por meio dos mapeamentos horizontais, algumas distinções semânticas foram mais elaboradas. Com o modelo de integração ficou clara a distinção entre os diferentes papéis que um funcionário desempenha nos sistemas A e B. Funcionário (Sistema A) foi mapeado para Funcionário Responsável por Empréstimo (modelo de integração), que representa um funcionário da instituição de ensino alocado em uma biblioteca que é responsável por um empréstimo. Por outro lado, Funcionário (Sistema B) foi mapeado para Funcionário Usuário (modelo de integração), que representa um funcionário que também é usuário da biblioteca.

O modelo de integração também ajudou a distinguir o elemento Empréstimo presente em ambos os sistemas. Empréstimo (Sistema B) foi mapeado para Empréstimo Unitário (modelo de integração), enquanto Empréstimo (Sistema A) foi mapeado para Empréstimo Múltiplo (modelo de integração).

Tabela 3.5- Ilustração dos Mapeamentos Horizontais

SISTEMA A	SISTEMA B	MODELO DE INTEGRAÇÃO
Empréstimo	-	Empréstimo Múltiplo
-	Empréstimo	Empréstimo Unitário
Pagamento	Pagamento	Pagamento de Multa
Multa	-	Multa
Funcionário	-	Funcionário Responsável por Empréstimo
-	Funcionário	Funcionário Usuário
-	Livro	Livro
Item	-	Item de Empréstimo
Estudante	Aluno	Aluno Usuário

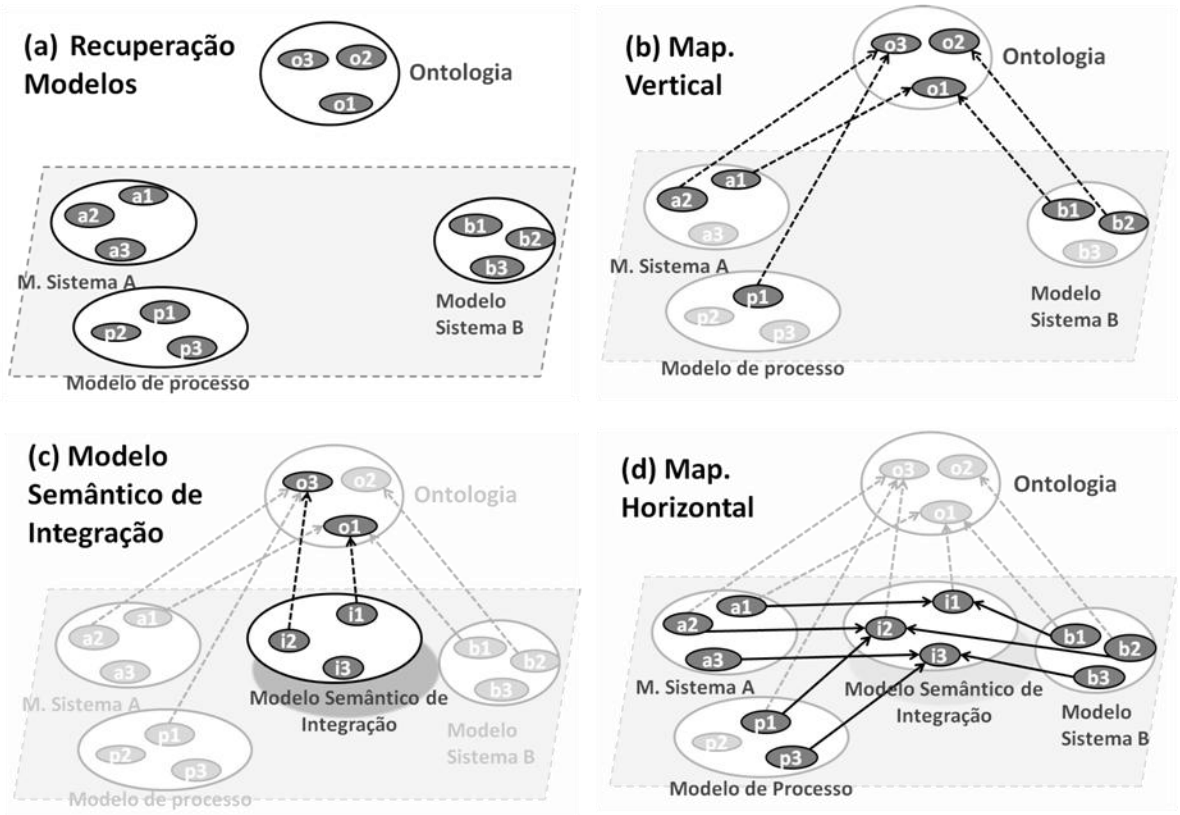

Figura 3.15 - Ilustração das atividades da fase de Análise da Integração

O modelo de integração também foi útil na atribuição de semântica dos elementos Pagamento (Sistema A) e Pagamento (Sistema B), os quais foram mapeados para Pagamento de Multa (Modelo de Integração). O modelo de integração tornou também mais específica a semântica de Estudante (Sistema A) e Aluno (Sistema B), que foram

mapeados para Aluno Usuário (modelo de integração), que representa um aluno da instituição que está registrado como usuário da biblioteca.

A Figura 3.15 ilustra as principais atividades da fase de análise da integração. Como está representado, a parte (a) da figura ilustra os modelos conceituais de sistemas, do processo de negócio a ser apoiado e das ontologias selecionadas; na parte (b) são ilustrados os mapeamentos verticais; na parte (c) é ilustrada a criação do modelo de integração; por fim, na parte (d) são ilustrados os mapeamentos horizontais.

3.5.3. Projeto e Implementação da Integração

Esta atividade tem por objeto especificar *como* ocorrerá a integração entre os sistemas. A solução de integração pode ser projetada e implementada de diversas formas. OBA-SI não se compromete com nenhuma solução específica de integração.

Essa etapa é responsável por:

- Definir e especificar a arquitetura da integração, bem como os seus componentes.
- Especificar como será a troca de dados e serviço entre os sistemas, com base na integração semântica realizada no nível de análise;

Os modelos de análise produzidos na fase anterior são independentes de aspectos computacionais relativos à solução da integração. Entretanto, desde que a fase de projeto leve em conta esses aspectos, é natural que esses artefatos sejam alterados, como em qualquer processo de desenvolvimentos de software. Assim, os modelos citados podem ser alterados para incorporar decisões relativas à solução da integração, dando origem ao modelo de projeto da integração.

Como tipicamente os sistemas a serem integrados não podem ser modificados, a integração de um conjunto de sistemas geralmente ocorre fora deles. Uma possível solução para tratar este problema é utilizar mediadores responsáveis por interligar os sistemas. Nessa estratégia, cada sistema não conversa com o outro diretamente, mas sim através de um mediador. O mediador deve, portanto, possuir uma visão geral dos sistemas a serem integrados, sendo responsável pelas trocas de dados e funcionalidades entre os sistemas. Por isso, o modelo de integração é a fonte mais importante para o projeto de um mediador, já que esse modelo captura a visão geral dos sistemas a serem

integradas com base na semântica geral da integração. Os mapeamentos horizontais, por exemplo, podem ser tomados como base para a tradução de dados entre os sistemas e o mediador.

Além disso, a orquestração dos serviços dos sistemas também deve ser realizada pelo mediador, levando em consideração a parte comportamental do modelo de integração. Os mapeamentos horizontais também podem ser usados para projetar a comunicação entre os sistemas e o mediador. A implementação dessa comunicação pode ocorrer dentro do mediador ou fora dele, por meio de adaptadores que conectam os sistemas ao mediador, como em (JIN; CORDY, 2006).

3.6. OBA-SI e as Camadas de Integração

Durante uma iniciativa de integração, mapeamentos verticais e horizontais são realizados entre os elementos dos modelos envolvidos, a saber: conceitos, relações entre conceitos (modelo estrutural), tarefas, insumos e produtos (modelo comportamental). Como já mencionado anteriormente, a integração envolve basicamente três camadas: dados, serviço e processo..

3.6.1. Atribuindo Semântica às Camadas de Integração

A integração semântica deve abranger todas as camadas de integração como ilustra a Figura 3.16. Ela ilustra a integração de dois sistemas nas diferentes camadas de integração para apoiarem um processo de negócio. São ilustradas as ligações entre dados, serviços e processo (entre serviços dos sistemas e atividades do processo de negócio). Como está ilustrado, a integração semântica está abrangendo todas as camadas por meio de mapeamentos semânticos entre elementos dos sistemas / processo e os modelos de referência (ontologias e modelo de integração).

Integração semântica na camada de dados

Em OBA-SI, a integração semântica na camada de dados é feita por meio de mapeamentos semânticos entre os modelos estruturais dos sistemas e a ontologia de referência. Para a camada de dados, ontologias de domínio são a referência principal, pois são elas que contêm as informações estruturais do domínio. Além delas, o modelo estrutural das ontologias de tarefa, referente aos papéis de conhecimento, também pode

ser usado de modo integrado com uma ontologia de domínio. Esses modelos são compostos de conceitos e relações entre conceitos.

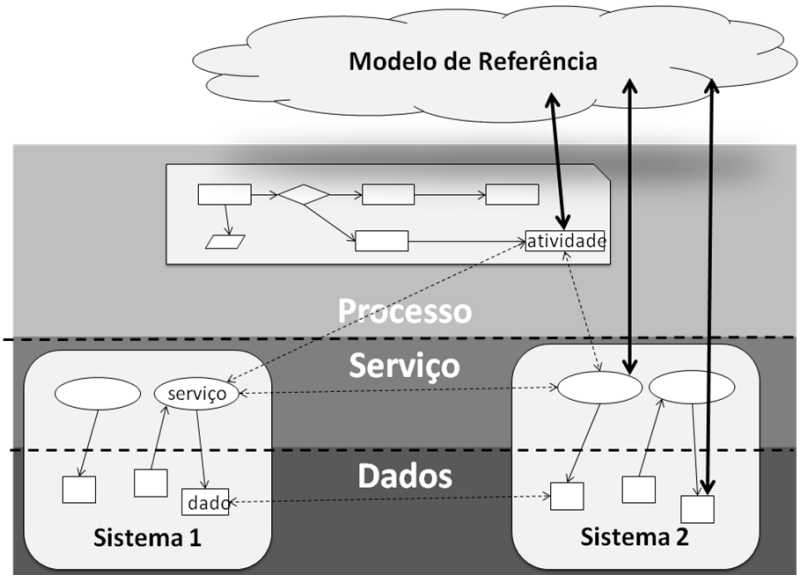

Figura 3.16 - Integração Semântica nas diferentes camadas de integração

Em OBA-SI, a integração semântica na camada de dados não engloba os dados (instâncias) propriamente ditos, mas apenas os conceitos do sistema. O foco de OBA-SI não é dizer, por exemplo, que o dado “Rio de Janeiro”, em uma base de dados, é equivalente ao dado “RJ”, em outra base de dados. Mas sim estabelecer uma equivalência entre o conceito de *Cidade* presente nos sistemas. Entretanto, a partir da integração semântica entre conceitos, o estabelecimento de equivalências entre dados é facilitado. Nesse caso, a relação de equivalência não ocorreria apenas entre o conceito Cidade dos sistemas, mas também de suas relações. Todas essas relações devem estar refletidas no nível de instância.

A Figura 3.17 ilustra a integração semântica na camada de dados em OBA-SI. A parte (a) da figura ilustra os modelos estruturais de dois sistemas, recortados em função do cenário de integração, além da ontologia de domínio. Na partes (b) e (c) da figura, estão ilustrados os mapeamentos verticais referentes aos sistemas A e B.

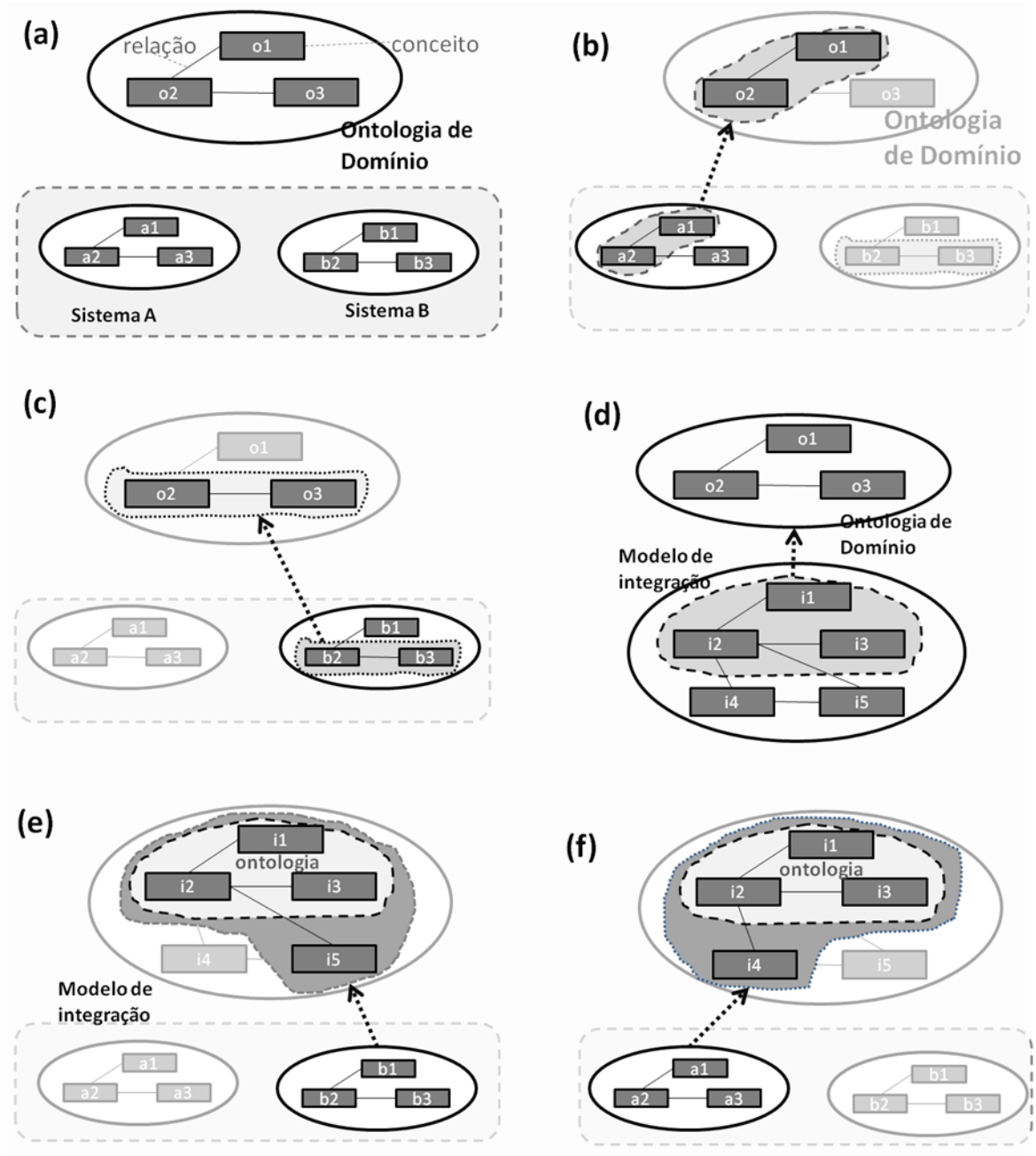

Figura 3.17 - Integração semântica estrutural

Na parte (d) está ilustrada a criação do modelo de integração (parte estrutural), o qual toma por base a ontologia de domínio, acrescentando alguns elementos necessários para tratar conceitos específicos dos sistemas. Por fim, nas partes (e) e (f) estão ilustrados os mapeamentos horizontais dos sistemas A e B ao modelo de integração. Vale ressaltar que os modelos apresentados são compostos apenas de conceitos e relações entre conceitos.

Integração Semântica nas Camadas de Serviço e Processo

Como foi a ilustrado na Figura 3.16, na integração nas camadas de serviço e processo, dois ou mais sistemas comunicam-se entre si para apoiar um processo de negócio. Com relação à integração semântica em tais camadas, OBA-SI faz uso de ontologias de tarefa, das partes comportamentais dos modelos dos sistemas (descrevendo serviços, insumos, produtos e o fluxo de controle) e do modelo de processo de negócio da organização.

É importante salientar que a parte dos sistemas modelada é relativa ao cenário de integração considerando os requisitos da integração levantados. A parte (a) da Figura 3.18 ilustra esses modelos. São ilustrados a ontologia de tarefa, os modelos parciais dos sistemas e o modelo de processo de negócio. De posse desses modelos, os modelos de entidades de integração (os de sistemas e do processo de negócio) são mapeados para a ontologia de tarefa, a qual é responsável por descrever tarefas genéricas, independentes de domínio, e, por isso, usada para atribuir semântica às funcionalidades dos sistemas e às atividades do processo de negócio. A parte (b) da Figura 3.18 ilustra um exemplo de mapeamento vertical de comportamento, no qual é atribuída semântica à parte dos modelos de sistemas e do processo de negócio.

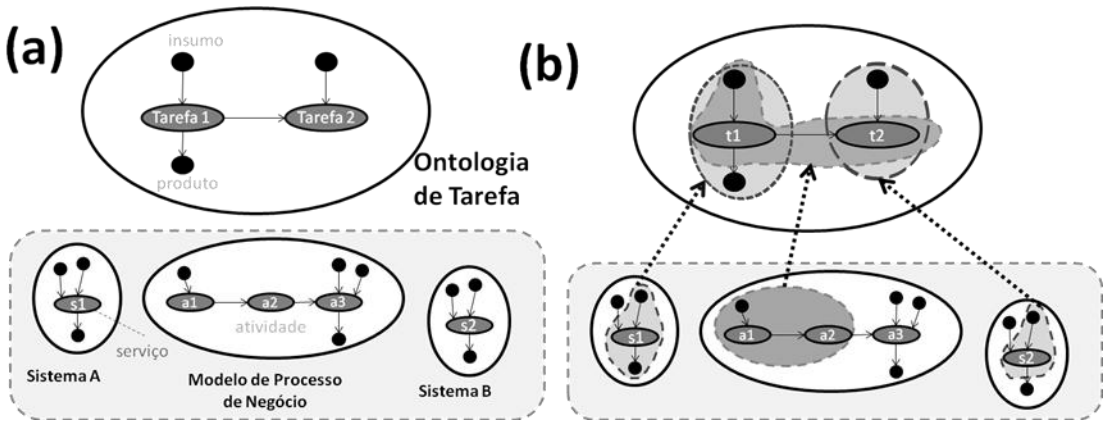

Figura 3.18 – Modelos para a integração semântica de serviço e processo

Da maneira análoga à integração na camada de dados, a ontologia de tarefa pode não ser capaz de cobrir todos os elementos considerados no cenário de integração. Assim, cabe ao modelo de integração cobrir todos os elementos dos modelos dos sistemas em questão. A parte (a) da Figura 3.19 ilustra a criação de um modelo de integração com base em uma ontologia de tarefa, a fim de cobrir suas lacunas. Já a parte

(b) ilustra o mapeamento horizontal do sistema A para o modelo de integração. Note que agora todos os seus elementos estão sendo cobertos pelo modelo de integração. A parte (c) ilustra os mapeamentos horizontais do sistema B e, por fim, a parte (d) ilustra os mapeamentos horizontais do modelo de processo de negócio.

Vale notar que o modelo de integração é basicamente uma extensão da ontologia de tarefa com informações adicionais referentes aos modelos parciais dos sistemas e do processo de negócio considerados no cenário de integração. Ele deve ser construído com o intuito de relacionar esses modelos, a fim de solucionar conflitos semânticos entre eles, estabelecendo uma semântica comum para a integração.

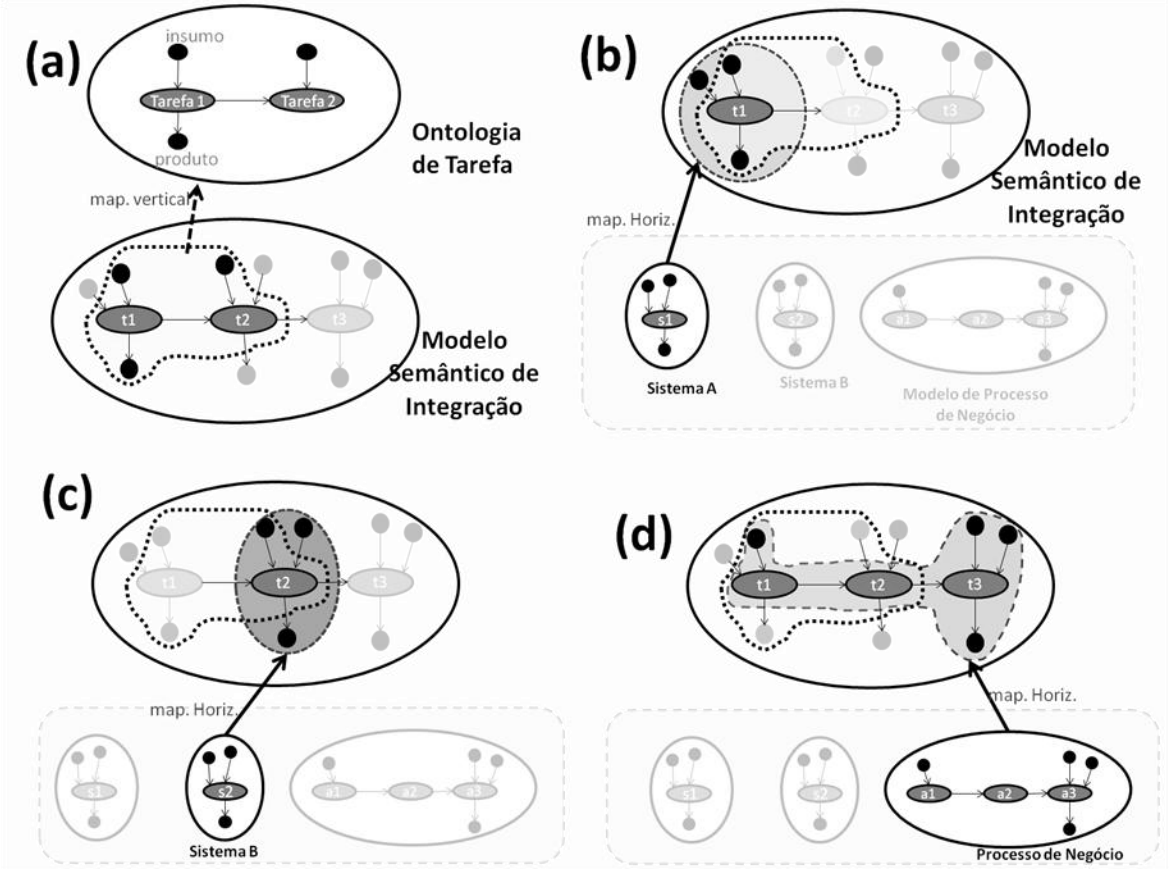

Figura 3.19 – Criação do modelo de integração e estabelecimento dos mapeamentos horizontais

Por fim, a Figura 3.20 apresenta uma visão geral da integração semântica no que se refere à parte comportamental da integração (envolvendo as camadas de serviço e de processo). Ela apresenta a ontologia de tarefa, os modelos de entidade de integração, os mapeamentos verticais e os mapeamentos horizontais. É importante destacar que nem

todos os elementos pertencentes aos modelos de sistemas ou ao modelo do processo possuem um mapeamento vertical, uma vez que geralmente a ontologia não consegue cobrir todos os elementos envolvidos na integração. Entretanto, em OBA-SI, todos os elementos envolvidos na integração devem possuir um mapeamento horizontal.

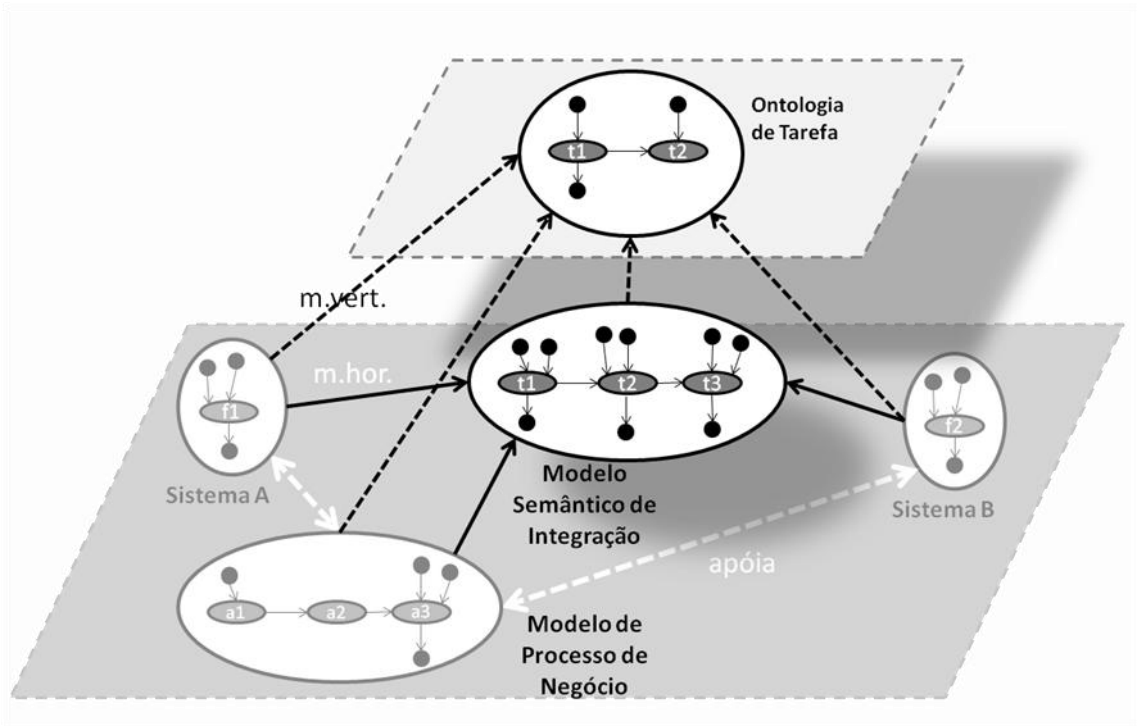

Figura 3.20 – Integração Semântica de Sistemas - Camadas de Serviço e Processo

3.6.2. Ordem de mapeamentos nas camadas de integração

A realização dos mapeamentos semânticos em cada camada de integração não necessariamente deve seguir uma ordem rígida, uma vez que o processo de mapeamento é iterativo. Entretanto, para facilitar essa tarefa, OBA-SI sugere uma ordem de mapeamento que pode ser realizada iterativamente. Assim, os mapeamentos em OBA-SI começam pelos mapeamentos de elementos estruturais dos modelos para depois tratar o mapeamento dos elementos comportamentais. Os mapeamentos estruturais estão relacionados à camada de dados e são responsáveis por atribuir significado a conceitos e relações. Já os mapeamentos comportamentais estão relacionados às camadas de serviço e de processo e são responsáveis por atribuir semântica aos elementos comportamentais dos modelos: funcionalidades ou atividades, produtos e insumos.

A Figura 3.21 apresenta o subprocesso de mapeamentos de modelos (atividades Efetuar Mapeamentos Verticais e Efetuar Mapeamentos Horizontais da fase de Análise da Integração de OBA-SI), explicitando a ordem em que os mapeamentos devem ser realizados, a saber: (i) conceitos, (ii) relações (i e ii se referem à parte estrutural), (iii) insumos e produtos aos respectivos conceitos, (iv) funcionalidades e (v) insumos e produtos (iii, iv e v se referem à parte comportamental). É bom salientar que essa ordem de mapeamento é seguida tanto para mapeamentos verticais quanto para os horizontais.

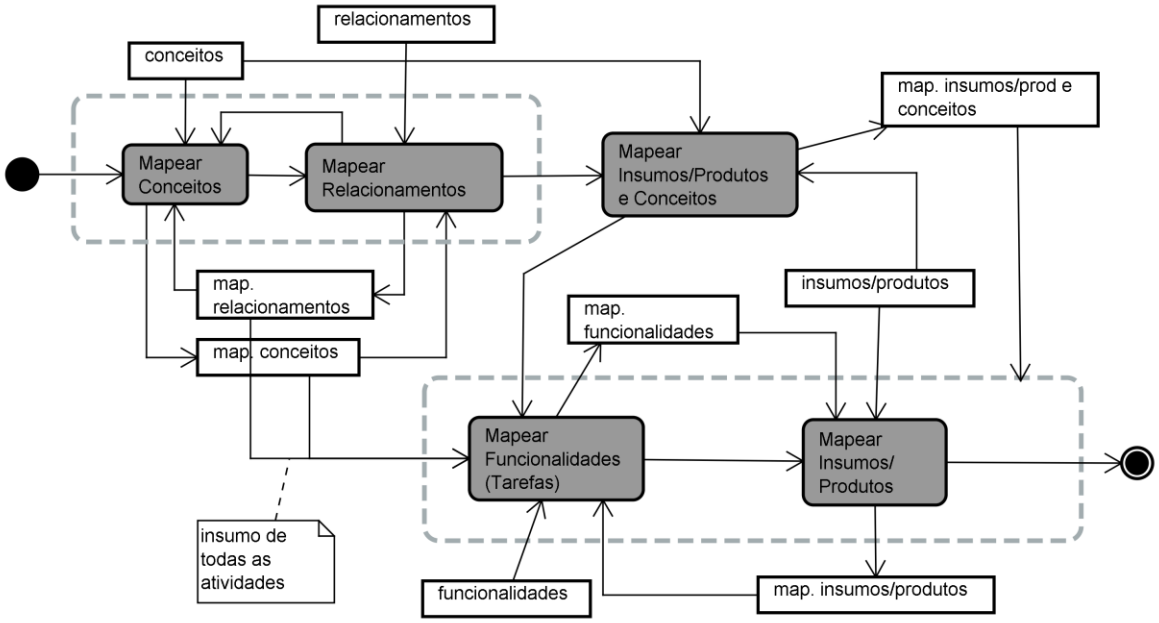

Figura 3.21 - Subprocesso de Mapeamento entre Elementos de Modelos

No mapeamento (vertical ou horizontal) entre dois modelos quaisquer, inicialmente, os conceitos pertencentes à parte estrutural dos modelos são analisados e mapeados entre si de acordo com o seu significado. Uma vez que os mapeamentos entre conceitos equivalentes foram estabelecidos, os relacionamentos entre eles são mapeados, de modo a identificar relacionamentos que são equivalentes. Para isso, os mapeamentos entre conceitos são fundamentais. Eles podem fornecer pistas de quais relacionamentos são equivalentes.

Seja o exemplo da Figura 3.22, na qual os conceitos *Livro* e *Empréstimo* do modelo do Sistema A do exemplo hipotético tem um relacionamento (*Livro* sofre *Empréstimo*) e que na ontologia existam dois conceitos *Item de Empréstimo e Empréstimo* relacionados (*Item de Empréstimo* é emprestado em *Empréstimo*). O conceito Livro (Sistema A) não foi mapeado com Item de Empréstimo (Ontologia), mas

Empréstimo (Sistema A) foi mapeado para Empréstimo (Ontologia). Nesse caso, existe uma equivalência entre os relacionamentos "*Livro* **sofre** *Empréstimo*" e "*Item de Empréstimo* **é emprestado em** *Empréstimo*", uma vez que ambos indicam que algo sofreu um empréstimo. Na parte B da figura é apresentado um exemplo de mapeamento de relacionamentos que supostamente seriam equivalente, pois os conceitos envolvidos o são, mas que na verdade não são.

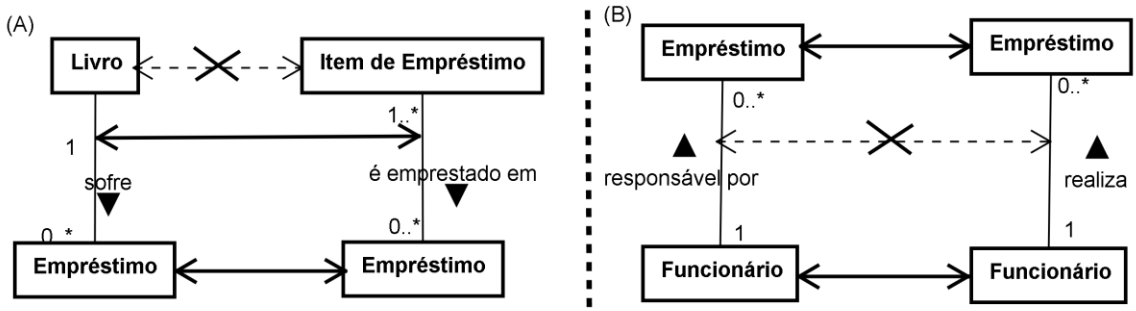

Figura 3.22 – Ilustração do mapeamento entre relações

É importante salientar também que, como os conceitos também são definidos em função dos seus relacionamentos, mapeamentos entre relacionamentos podem influenciar os mapeamentos entre conceitos e, por isso, o mapeamento estrutural é um processo iterativo.

Depois de realizar o mapeamento estrutural entre os modelos, passa-se ao mapeamento comportamental. Ele é responsável pela atribuição de semântica às funcionalidades dos sistemas, bem como respectivos insumos e produtos. O mapeamento comportamental é baseado no estrutural e está relacionado com a integração semântica nas camadas de serviço e de processo. Como resultado dessa etapa, a semântica das funcionalidades dos sistemas, bem como de seus insumos e produtos, é estabelecida, da mesma forma que ocorre entre conceitos e relacionamentos no mapeamento estrutural. Entretanto, no caso do mapeamento comportamental, a identificação das correspondências geralmente não é tão direta como no mapeamento estrutural.

O foco da abordagem de integração semântica não está em propor uma maneira rígida de realizar a atribuição semântica. Apesar disso, OBA-SI aponta algumas diretrizes para guiar a integração semântica de funcionalidades, a saber: (i) análise da estrutura dos modelos comportamentais (ordem em que as funcionalidades ocorrem); (ii) análise dos

insumos e produtos das funcionalidades; (iii) análise dos passos internos da funcionalidade.

A análise da estrutura dos modelos comportamentais pode ajudar a identificar funcionalidades e tarefas que ocorrem em um mesmo contexto. Funcionalidades ou tarefas relacionadas geralmente ocorrem em situações similares, possuindo, por exemplo, funcionalidades / tarefas similares que ocorrem antes ou depois. No exemplo hipotético, ao se integrar os sistemas de biblioteca, as funcionalidades “Devolver Item Emprestado” (Sistema A) e “Devolver Livro” (Sistema B) ocorrem depois das funcionalidades “Efetuar Empréstimo de Item” (Sistema A) e “Emprestar Livro” (Sistema B). Nesse exemplo, elas possuem certa similaridade, pois ocorrem em um mesmo contexto. Além disso, a análise de insumos e produtos das funcionalidades/tarefas também pode fornecer importantes pistas sobre como realizar os mapeamentos, já que funcionalidades que são similares geralmente manipulam dados do mesmo tipo.

Antes de se realizar os mapeamentos comportamentais, é necessário que os insumos e produtos envolvidos estejam relacionados com os respectivos conceitos da parte estrutural do modelo de sistema. Uma vez que os insumos e produtos estejam relacionados com os conceitos que eles representam, as funcionalidades/tarefas podem ser mapeadas entre si. Para isso, o analista de integração deve selecionar funcionalidades/tarefas que possuam a mesma semântica. Como essa é uma tarefa bastante subjetiva, o analista pode se basear nos insumos e produtos dos serviços e compará-los, a fim de auxiliar a integração. Se os insumos e produtos de duas funcionalidades são correspondentes, a chance de as funcionalidades também serem é considerável.

Para identificar insumos e produtos que sejam equivalentes, pode-se tomar como base os conceitos que eles representam. Se, por exemplo, dois insumos representam conceitos equivalentes, a chance deles também serem equivalentes é considerável. Entretanto, assim como no caso dos relacionamentos, não há garantia de que isso ocorra. O mais importante na análise de equivalência entre insumos e entre produtos é relacionar com base no papel desempenhado pelo insumo/produto na funcionalidade em questão. Seja o exemplo de mapeamento das duas funcionalidades anteriormente citadas “Efetuar Empréstimo de Item” e “Emprestar Livro”. Ambas possuem como insumo um item a ser emprestado. Entretanto, esses insumos estão relacionados a conceitos não equivalentes

(Item e Livro, respectivamente). Apesar disso, eles podem ser mapeados, pois possuem o mesmo papel nas respectivas funcionalidades (item a ser emprestado).

A análise dos passos internos de uma funcionalidade também pode ajudar bastante na identificação de funcionalidades similares. Entretanto, essa análise pode ser difícil de ser realizada justamente pela dificuldade de se saber como a funcionalidade é realizada internamente pelo sistema. Mesmo assim, essa análise pode ajudar, principalmente se as funcionalidades implementarem alguma tarefa mais complexa, como por exemplo, um algoritmo ou cálculo. Através da análise de como as funcionalidades realizam uma tarefa, podem-se identificar similaridades entre elas. A Figura 3.23 ilustra um exemplo de mapeamento comportamental.

Ela apresenta mapeamentos entre insumos, conceitos e funcionalidades (serviços) pertencentes aos modelos conceituais comportamentais de dois sistemas. Em OBA-SI o mapeamento entre modelos de sistemas ocorre por meio do modelo de integração. Mas nesse exemplo o mapeamento é ilustrado diretamente para facilitar o entendimento relativo à ordem de mapeamento entre os elementos. Na parte (a) de figura são mostrados os modelos a serem mapeados. Cada um deles composto de uma parte estrutural (em cima) e outra comportamental (em baixo). Como ilustra a parte (b), primeiramente é realizado o mapeamento estrutural. Os conceitos Empréstimo (Sistema A) e Empréstimo (Sistema B) dos modelos foram mapeados entre si, entretanto o mesmo não ocorreu com os conceitos Item e Livro. Depois foi realizado o mapeamento entre os relacionamentos.

Feitos os mapeamentos estruturais, passa-se aos mapeamentos comportamentais. Inicialmente, os insumos e produtos de cada um dos serviços devem ser mapeados para conceitos pertencentes ao modelo estrutural. Na parte (c) estão ilustrados esses relacionamentos. Por fim, devem ser feitos os mapeamentos entre os elementos dos modelos comportamentais. Os serviços *Efetuar Empréstimo de Item* e *Emprestar Livro* foram mapeados entre si como mostra a figura.

Com relação aos mapeamentos entre insumos e produtos, os relacionamentos entre os conceitos correspondentes podem fornecer pistas de mapeamentos entre eles. Por exemplo, os produtos *empréstimo* (Sistema A) e *empréstimo* (Sistema B) foram mapeados entre si e esse mapeamento está refletido entre os conceitos correspondentes. Entretanto, no mapeamento entre insumos e entre produtos, o que deve ser considerado

principalmente é o papel deles nas funcionalidades. O mapeamento entre os insumos *item* (Sistema A) e *livro* (Sistema B) ilustra isso. Nesse exemplo, os insumos, apesar de representarem conceitos diferentes, possuem papéis similares nas funcionalidades correspondentes e, por isso, foram mapeados.

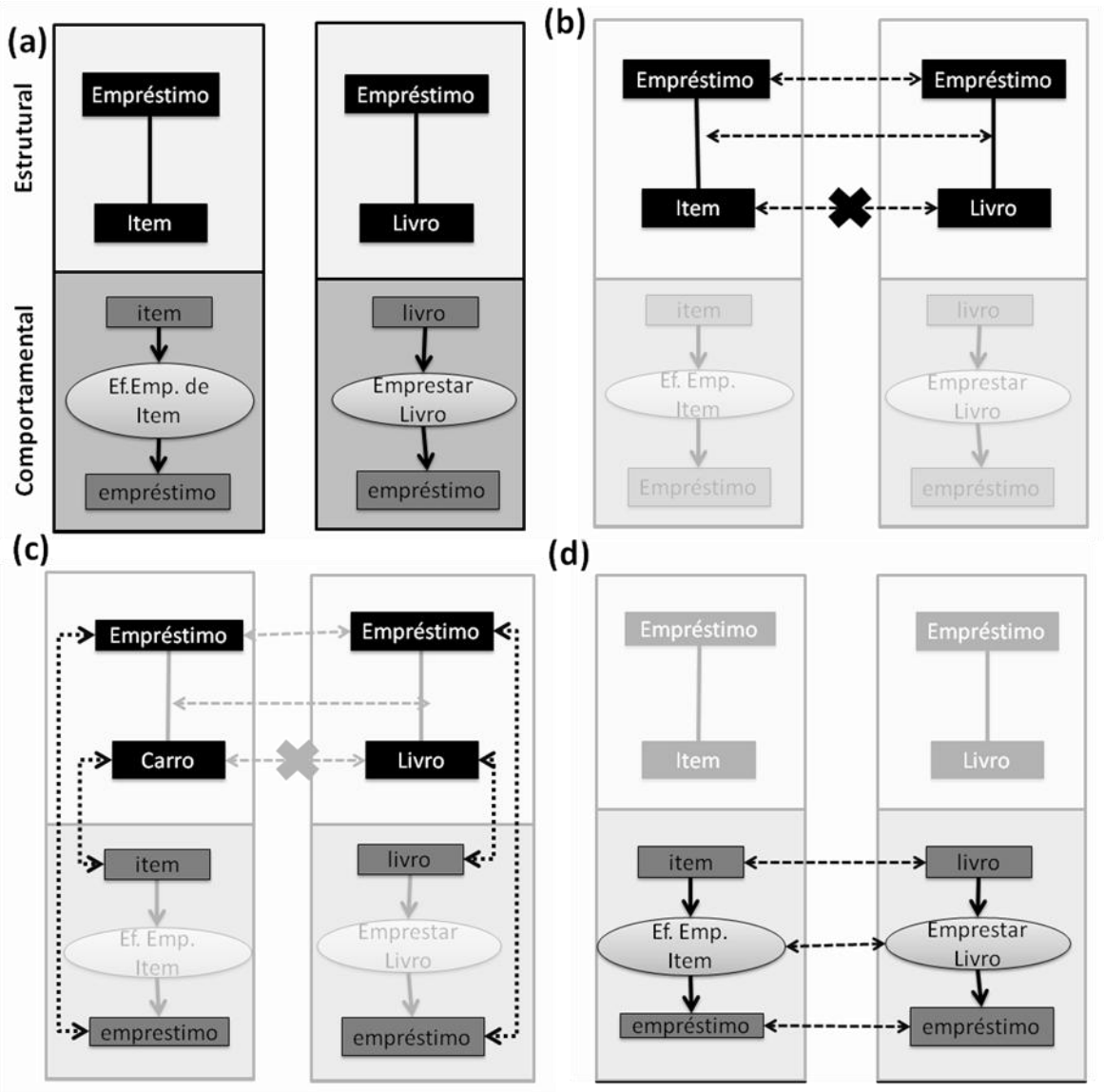

Figura 3.23 – Ordem dos Mapeamentos entre Modelos

3.7. Trabalhos Correlatos

Conforme citado em (GUARINO, 1998), considerando a dimensão temporal, ontologias podem ser usadas em tempo de desenvolvimento ou de execução. OBA-SI faz uso de ontologias em tempo de desenvolvimento da solução de integração, focando no

nível conceitual de integração. Outras abordagens, como ODSOI (IZZA et al, 2005), usam ontologias em tempo de execução, uma vez que são focadas em aspectos tecnológicos. A seguir algumas abordagens mais semelhantes a OBA-SI são discutidas brevemente.

Izza et al. (2005) propuseram ODSOI (*Ontology-Driven Service-Oriented Integration*). ODSOI estende a tecnologia de serviços web (*web services*) adicionando uma camada semântica, por meio da qual é possível adicionar semântica aos serviços web. A principal função dessa camada é definir a semântica dos serviços e realizar a mediação semântica no contexto de integração de aplicações corporativas (*Enterprise Application Integration* - EIA). Assim como OBA-SI, ODSOI também propõe um processo de integração. Do mesmo modo que OBA-SI, esse processo de integração cobre as camadas de dados, serviço e processo, apesar de o primeiro protótipo da arquitetura ser restrito apenas a camada de dados da integração. Considerando o processo de integração de ODSOI, a principal diferença com relação ao processo de integração semântica proposto neste trabalho é a integração de serviços ocorre em tempo de execução.

Outra abordagem relacionada a OBA-SI é o *framework* baseado em ontologias para EAI chamado ONAR (TEKTONIDIS, 2005). ONAR utiliza tecnologias de web semântica com o intuito de enriquecer a semântica envolvida na troca de informação. Seu processo de integração considera o uso de ontologias tanto em tempo de desenvolvimento quanto em tempo de execução. Considerando o tempo de desenvolvido, o processo de integração de ONAR propõe duas fases que são bastante alinhadas com as ideias de OBA-SI, chamadas de fase de conceituação e fase de associação. Na fase de conceituação de ONAR, um caso de integração é definido (similar ao cenário de integração de OBA-SI) e entidades que precisam participar do caso de integração são conceituadas em função de ontologias OWL. O conjunto de entidades associadas a conceitos dá origem ao esquema conceitual da integração. Já na fase de associação, o analista associa os conceitos da ontologia aos recursos dos repositórios dos sistemas que participam da integração, de modo bastante similar com o proposto em OBA-SI.

Entretanto, existem duas diferenças. Primeiro, OBA-SI considera ontologias de referência de domínio e ontologias de tarefa para atribuir semântica aos elementos de modelo. Como mencionado por (GUIZZARDI et al., 2009), uma ontologia de referência deve ser usada, ajudando na representação do domínio sem considerar requisitos computacionais. O principal objetivo de uma ontologia de referência é ajudar modeladores

a exteriorizar o conhecimento tácito sobre o domínio para tornar explícitos os comprometimentos ontológicos, facilitando a negociação do significado, as tarefas de comunicação sobre o domínio, aprendizado e solução de problemas da melhor maneira.

A mesma ontologia de referência pode dar origem a diferentes ontologias leves, em diferentes linguagens (como OWL), e assim satisfazer diferentes conjuntos de requisitos não funcionais. Definir a linguagem mais adequada para codificar a ontologia de referência é, dessa forma, uma escolha a ser feita na fase de projeto da integração, levando em consideração tanto o propósito da aplicação final quanto o equilíbrio entre a expressividade e o tratamento computacional. A segunda diferença é com relação à fase de associação. OBA-SI fornece diretrizes para a realização delas, enquanto ONAR apenas menciona que conceitos da ontologia devem ser relacionados aos recursos do repositório do sistema. Por fim, ONAR trata a integração apenas nas camadas de dados e serviços, enquanto OBA-SI também inclui aspectos relacionados à camada de processo.

Uma abordagem bastante interessante é proposta em (POKRAEV, 2009). Dentre as abordagens estudadas, esta é a mais completa entre as que consideram a análise da integração. Uma característica importante dessa abordagem é que ela faz uso de técnicas de Arquiteturas Dirigidas a Modelos (*Model Driven Architecture* – MDA). Da mesma forma que OBA-SI, essa abordagem considera a integração tanto nas camadas de dados e serviços quanto na de processo e ela também ocorre em um alto nível de abstração, por meio de modelos independentes de plataforma (*Platform Independent Models* - PIMs). A extração dos PIMs que representam os modelos dos sistemas é feita automaticamente por meio do processamento de modelos específicos de plataforma (*Platform Specific Models* - PSMs) que implementam os sistemas. Nessa abordagem, ontologias também são usadas para atribuir semântica aos modelos que representam sistemas, como ocorre em OBA-SI. A abordagem também possui um modelo geral usado para modelar a integração. Esse modelo de integração é também independente de plataforma (PIM). Contudo, essa abordagem não foca muita atenção nos mapeamentos semânticos e a integração semântica ocorre de modo bastante simples, relacionando apenas conceitos. Em OBA-SI a atribuição de semântica busca ser mais completa, envolvendo mapeamentos entre conceitos, relações, serviços/tarefas/atividades e respectivas entradas e saídas. Além disso, OBA-SI considera o uso de ontologias de tarefa.

A Tabela 3.6 apresenta um resumo comparativo entre OBA-SI e as abordagens citadas anteriormente. Como se pode notar, o principal diferencial entre OBA-SI e as

outras abordagens é o uso de ontologias de referência, que enriquece a qualidade da integração semântica, o uso de ontologias de tarefa, que permite a atribuição de semântica a serviços e ao processo, e as diretrizes de mapeamento. E as principais desvantagens são a ausência de estratégias para a solução da integração, bem como a falta de uso de ontologias na forma de modelos computacionais (ontologias leves), o que impossibilita o uso de ontologias em tempo de execução, bem como a automatização em determinadas etapas ou até mesmo a realização de outros recursos, como a simulação.

Tabela 3.6- Comparações entre Abordagens

	NÍVEL		CAMADA			ONTOLOGIA				Processo	
	Conceitual	Solução	Dado	Serviço	Processo	Referência	Leve	Domínio	Tarefa/Serviço	Processo Integração	Diretrizes de Mapeamento
OBA-SI	X		X	X	X	X		X	X	X	X
ONAR	X	X	X				X	X			
ODSOI		X	X	X	X		X		X	X	
POKRAEV	X	X	X	X	X		X	X		X	

3.8. Conclusão

Este capítulo apresentou OBA-SI, uma abordagem de integração semântica de sistemas baseada em ontologias. Seu intuito é sistematizar o processo de integração, sugerindo etapas a serem seguidas. Podem-se destacar como características importantes de OBA-SI:

- A integração semântica é tratada antes do projeto e implementação da integração, uma vez que antes de a integração ser solucionada, é preciso verificar se os sistemas são semanticamente compatíveis.
- O processo de integração é tratado como um processo de desenvolvimento de software, uma vez que é gerado um produto de software que materializa a integração entre os sistemas.

- OBA-SI mescla abordagens de integração *top-down* (baseada principalmente nos requisitos e em ontologias de referência) e *bottom-up* (baseada principalmente nos sistemas). Isso é importante já que, para modelar a conceituação comum envolvida na integração, é importante considerar tanto o domínio e as tarefas envolvidos (independentes da integração), como os sistemas a serem integrados (dependentes da integração).

- OBA-SI trata as camadas de integração de dados, serviços e processo com o propósito de não haver conflitos de entendimento na troca de dados e serviços entre os sistemas nem no apoio ao processo pelos sistemas, evitando, assim, que dados e serviços sejam usados para um propósito que não podem atender.

- OBA-SI promove a utilização e o reúso do conhecimento independente da integração, por meio de ontologias de domínio e de tarefa, de modo que o entendimento do universo de discurso pelos envolvidos seja bastante rica.

- A atribuição de semântica nas camadas de serviço e processo é feita usando ontologias de tarefa, uma vez que é um artefato independente da integração e descreve as tarefas envolvidas de modo genérico.

- OBA-SI separa interesses de uma iniciativa de integração por meio de diferentes níveis de abstração, estimulando que os envolvidos primeiramente entendam o universo de discurso da integração para depois identificar as compatibilidades semânticas e, finalmente solucionar tecnologicamente a integração.

- OBA-SI é focada na integração de sistemas HAD (Heterogêneos, Autônomos e Distribuídos), que não são passíveis de alteração, pois são os sistemas mais problemáticos na hora de se realizar a integração semântica e os que mais precisam de uma abordagem para isso.

OBA-SI propõe poucas etapas adicionais ao processo de desenvolvimento de software tradicional, mas que são importantes para a realização da integração semântica. Seu intuito é apresentar uma abordagem em que a integração semântica é tratada em um nível mais alto de abstração. Assim, ela procura não se comprometer com linguagens de modelagem, linguagens de implementação, metodologias, tecnologias ou arquiteturas de integração e, por isso, pode ser combinada com outras abordagens, sobretudo as mais

focadas na solução e em aspectos tecnológicos, uma vez que são abordagens complementares.

Para uma avaliação da utilidade da abordagem OBA-SI, proposta neste capítulo, foi realizado um estudo de caso onde sistemas de gerência de configuração são integrados para apoiar o processo de gerência de configuração. No capítulo seguinte é apresentada uma ontologia de tarefa de gerencia de configuração criada para esse estudo de caso e no Capítulo 5 é apresentada a integração.

Capítulo 4 - Uma Ontologia da Tarefa de Gerência de Configuração

A gerência de configuração (GC) é um processo fundamental no desenvolvimento de produtos complexos. Ela provê diretrizes técnicas e administrativas para gerenciar o ciclo de vida de um produto e de seus itens de configuração (ICs). A GC orienta e controla a evolução da configuração de um produto, provendo meios para prevenir a desordem em seu desenvolvimento (ISO 10007, 2003). Esse controle ocorre através de um processo que identifica e define os ICs do produto, controla as alterações nos ICs durante o ciclo de vida do produto, registra e relata o estado dos mesmos e verifica a consistência de tais itens (ISO 10007, 2003) (MIL-HDBK-61A, 1997).

Este capítulo apresenta uma ontologia de tarefa que descreve aspectos comportamentais da Gerência de Configuração. Ela foi desenvolvida para ser usada como referência na integração semântica de sistemas de GC, principalmente nas camadas de processo e de serviço. Por ser uma tarefa bastante extensa, a ontologia foca principalmente na atividade de controle de alteração, que é uma parte fundamental da GC. Para representar a parte estrutural da ontologia, foi usada a linguagem de modelagem OntoUML (GUIZZARDI, 2005)..

A ontologia proposta neste capítulo representa as informações mais importantes relativas ao processo de GC, tais como conceitos, relações, tarefas, agentes, entradas e saídas. Além disso, cada informação possui uma classificação ontológica atribuída pela própria linguagem de modelagem usada, provendo, assim, uma representação mais precisa e clara da tarefa de GC. A sua construção foi baseada principalmente em sistemas de GC existentes (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2011), (CAETANO, 2004), (TRAC, 2011), normas (ISO 10007, 2003) (MIL-HDBK-61, 1997), livros (LEON, 2000) (PRESSMAN, 2006) e na ontologia de Gerência de Configuração de Software proposta em (ARANTES; FALBO; GUIZZARDI, 2007).

Vale ressaltar que neste texto é adotado o termo ontologia de tarefa por ser este bastante utilizado na área de ontologias devido às definições de Guarino (1998). Contudo, a ontologia proposta se refere, de fato, ao processo de GC e, portanto, um termo mais apropriado seria ontologia de processo.

Este capítulo está organizado da seguinte forma: a Seção 4.1 apresenta uma motivação para o desenvolvimento de uma ontologia da tarefa de GC; a Seção 4.2

apresenta uma breve revisão de GC, a Seção 4.3 apresenta a ontologia de tarefa; a Seção 4.4 descreve a relação da ontologia de tarefa criada com a ontologia de domínio de Gerência de Configuração de Software proposta em (ARANTES; FALBO; GUIZZARDI, 2007); finalmente, a Seção 4.5 apresenta as conclusões do capítulo.

4.1. Motivação

Com o intuito de ajudar na integração semântica, um grande número de ontologias de domínio tem sido desenvolvido em diferentes áreas (GUIZZARDI, 2005), inclusive no domínio de Gerência de Configuração, como é o caso da ontologia proposta em (ARANTES; FALBO; GUIZZARDI, 2007). Apesar do crescimento do uso de ontologias de domínio para a solução de problemas de integração semântica, o mesmo não ocorre com ontologias de tarefas (MARTINS, 2009). Porém, na integração semântica, é importante considerar também o conhecimento relativo a aspectos comportamentais.

Conforme discutido no Capítulo 2, a integração de sistemas pode ocorrer em três camadas distintas, a saber: camada de dados, de serviços e de processo. A integração na camada de dados lida com o compartilhamento de dados entre sistemas. Já a integração na camada de serviço se preocupa em como os sistemas vão compartilhar serviços. Finalmente, a camada de processo vê a organização como um conjunto de processos relacionados e é responsável por gerenciar o fluxo de mensagens, implementando regras e definindo a execução do processo como um todo.

Dessa forma, a integração semântica também deve englobar o significado relativo a conceitos, serviços e também a processos (IZZA, 2009). Um fator importante para isso é o compartilhamento de uma conceituação comum não apenas referente a aspectos estruturais (conceitos e relações), que permite integração na camada de dados, mas também que envolva aspectos comportamentais (serviços, tarefas, insumos, produtos, atores). Ontologias de tarefa podem desempenhar esse papel, sendo usadas para atribuir semântica a elementos como serviços, funcionalidades, tarefas e informações relacionadas, ajudando na integração semântica nas camadas de serviço e processo.

Idealmente, ontologias de domínio e de tarefa devem ser baseadas em ontologias de fundamentação (ontologias de alto nível) (GUARINO, 1998). Entretanto, as linguagens usadas para representar ontologias geralmente não são apropriadas para isso, pois não são ontologicamente bem fundamentadas (GUIZZARDI, 2005). A utilização de linguagens bem fundamentadas ontologicamente pode trazer diversos benefícios, influenciando a

qualidade da representação de conhecimento, tanto de domínio quanto de tarefa, podendo influenciar também na qualidade da integração semântica.

Assim, o desenvolvimento de ontologias de domínio e de tarefa bem fundamentadas ontologicamente pode ser bastante importante no contexto da GC visto que é uma tarefa complexa que envolve diferentes tipos de atividades e preocupações. O intuito da ontologia de tarefa de GC apresentada neste capítulo é ajudar na integração semântica de sistemas de apoio ao processo de GC.

4.2. O Processo de Gerência de Configuração

O principal objetivo da GC é controlar a evolução de produtos (ISO 10007, 2003). A GC é uma disciplina que aplica técnicas e procedimentos administrativos para: (i) identificar e documentar características físicas e funcionais de itens de configuração (IC); (ii) controlar alterações nessas características; (iii) armazenar e reportar as modificações realizadas e; (iv) verificar a compatibilidade de modificações realizadas com o requisitos especificados (IEEE, 2003).

Para facilitar o controle da evolução do produto, ele é dividido em itens. Um item é um termo genérico usado para representar partes de um produto ou informações geradas no seu desenvolvimento. Um item de produto cuja configuração é gerenciada é chamado de Item de Configuração (IC). A evolução de ICs ocorre por meio de procedimentos formais para se manter o controle e a organização. Os ICs possuem uma maneira de serem identificados, fazendo parte da configuração do produto (ISO 10007, 2003).

Um IC pode possuir diferentes estados enquanto ele evolui. O termo versão representa um estado específico de um IC em um determinado momento do desenvolvimento do produto (LEON, 2000). O produto como um todo também pode possuir diferentes estados chamados de configuração. Configuração é geralmente definida como um conjunto de itens que formam o produto (ISO 10007, 2003) (ESTUBLIER, 2000). A ISO 10007 (2003) define configuração como sendo um conjunto de características físicas e funcionais que descreve o produto em um determinado momento.

Linha base é outro conceito importante da GC. Uma linha base é basicamente uma configuração do produto que foi revisada e designada a servir de base para o desenvolvimento futuro (ISO 10007, 2003) (IEEE, 2003). É uma referência no

desenvolvimento formalmente definida em um determinado estágio do ciclo de vida do produto (ESTUBLIER, 2000).

Considerando o processo de GC, ele é apresentado em diferentes formas em diferentes referências. Tomando por base principalmente (ISO 10007, 2003), (MIL-HDBK-61A, 1997) e (ESTUBLIER, 2000), as principais atividades do processo de GC podem ser resumidas da seguinte forma:

- Identificação de Configuração: identifica itens do produto a serem controlados, estabelece critérios de seleção de ICs e suas versões, e estabelece padrões de numeração, ferramentas e técnicas a serem usadas para controlar os itens;
- Controle de Versão: combina procedimentos e ferramentas para gerenciar as diferentes versões de um IC criadas durante seu desenvolvimento;
- Controle de Alteração: atua na gerência de alterações realizadas durante o ciclo de vida do produto e envolve: (i) solicitação de alterações, (ii) avaliação de alterações; (iii) checkout, (iv) realização de alteração, (v) revisão de alteração e (vi) checkin dos itens modificados.

Além dessas atividades, a gerência de configuração possui atividades que envolvem o planejamento da GC, auditorias de configuração e também o relato do status da configuração.

4.3. Uma Ontologia da Tarefa de Gerência de Configuração

O processo de GC é, em geral, bastante complexo e extenso. Ele possui diferentes preocupações, dentre elas o controle de versões do produto, o relato desses estados, a realização de auditorias e o controle de alterações, sendo essa última uma das tarefas mais importantes da GC. O controle de alterações está diretamente ligado com o controle da evolução do produto, que ocorre por meio de alterações. A maioria das demais atividades citadas ocorrem em decorrência das alterações, com algumas exceções como, por exemplo, a identificação de ICs. Por essa razão, a tarefa de controle de alteração o foco da ontologia apresentada.

O controle de alteração envolve diferentes atividades, dentre elas: identificação e solicitação da alteração, avaliação da solicitação de alteração, retirada (*checkout*) dos

itens para alteração, implementação da alteração solicitada, registro (*checkin*) dos itens alterados e verificação dos itens alterados.

Ontologias de tarefa visam capturar o conhecimento genérico acerca de uma tarefa, envolvendo duas perspectivas complementares (MARTINS, 2009): (i) decomposição de tarefas em sub-tarefas e fluxo de controle (representação de informações comportamentais), e (ii) papéis de conhecimento a serem desempenhados pelos conceitos de domínio na tarefa em questão (representação de informações estruturais).

Como mencionado, OntoUML foi usada para representar papéis de conhecimento através de diagramas de classes e E-OntoUML foi usada para representar a perspectiva comportamental por meio de diagramas de atividades.

4.3.1. Papéis de Conhecimento Centrais da Gerência de Configuração

A Figura 4.1 mostra o modelo estrutural da ontologia de tarefa de GC proposta. Nesse modelo, **Item** representa diferentes elementos que compõem um produto e que podem ser gerenciados. Este conceito foi representado como uma categoria, uma vez que representa elementos que possuem um princípio de identidade distinto. Ou seja, itens podem representar diferentes tipos (*kinds*), tais como documentos e ferramentas, que estão sujeitos a alteração e podem ter sua configuração gerenciada. Os itens de um produto que estão tendo sua configuração gerenciada são representados pelo conceito **Item de Configuração** (IC). Assim, o conceito IC está representado como sendo um papel desempenhado por um **Item** mediante uma **Seleção de Configuração** realizada por um **Gerente de Configuração**. A relação de seleção de itens foi representada como uma relação material, ou seja, composta de um *relator* (**Seleção de Configuração**). O *relator* Seleção de Configuração representa, por sua vez, um objeto que formaliza quais itens de um produto que estão sendo gerenciados.

Um IC possui uma ou mais **Versões**, que representam estados específicos da evolução do IC. À medida que um IC vai evoluindo, novas versões dele vão sendo criadas. O conceito **Versão** foi representado como sendo uma propriedade do tipo *mode*, uma vez que está intrinsecamente relacionado ao item.

ICs e suas versões podem ser atômicos ou compostos. As versões de um IC composto são compostas de outras versões e são chamadas de **Configuração**. Já as versões de um IC atômico também são atômicas. Além disso, as relações de composição

entre **IC Composto** e **Item de Configuraçã**o e entre **Configuração** e **Versão** são do tipo *componentOf*, indicando que cada parte desempenha um papel diferente no todo.

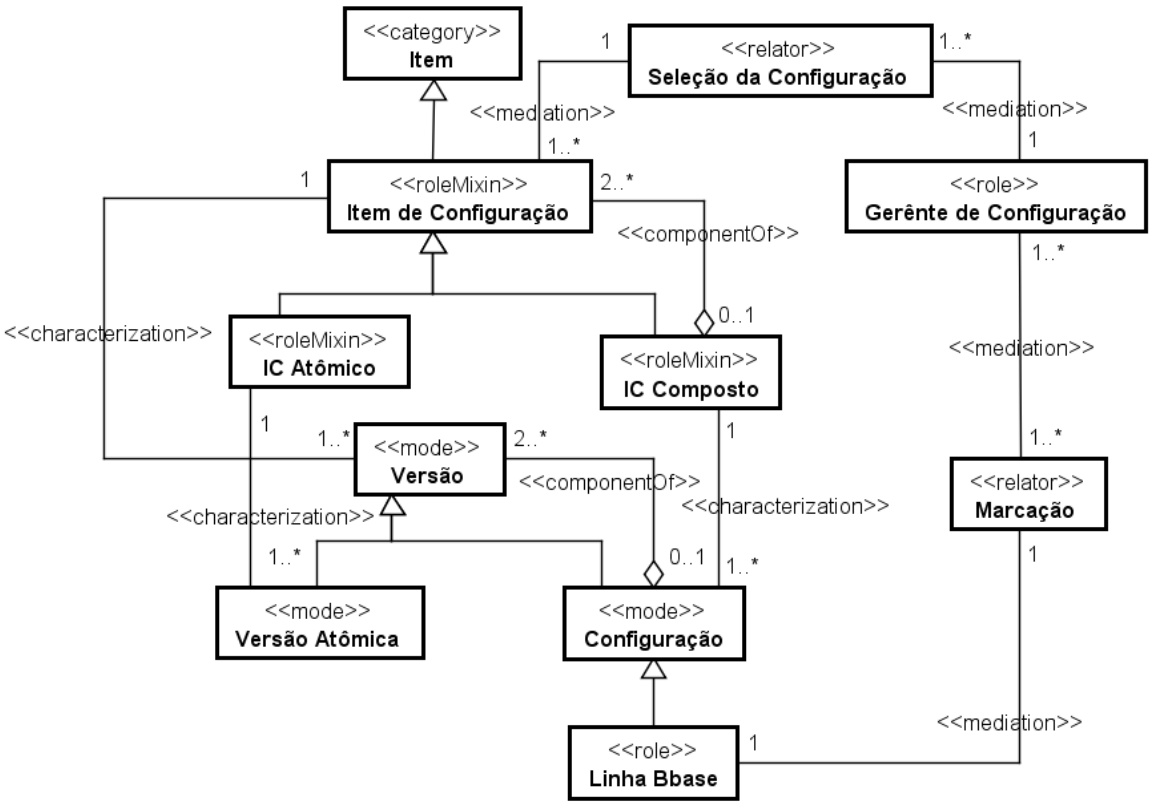

Figura 4.1 - Modelo Estrutural da Ontologia de Gerência de Configuração.

Configurações podem desempenhar o papel de **Linha Base** quando uma **Marcação** é feita por um ou mais **Gerentes de Configuração**. Esse evento é representado por uma relação material e o relator Marcação representa um registro formal do mesmo. Para facilitar o entendimento, alguns conceitos foram omitidos nesse diagrama. Por exemplo, os papéis de responsável por linha base e responsável por configuração (papéis de **Gerente de Configuração**) não foram representados. É importante lembrar que os relacionamentos de herança na UML, por padrão, são incompletos (*incomplete*) e disjuntos (*disjoint*) e o tipo nesse caso é omitido. O tipo do relacionamento de herança será mostrado somente quando a herança for completa (*complete*) ou sobreposta (*overlapping*).

É bom salientar que os modelos estruturais de uma ontologia de tarefa são incompletos por natureza, uma vez que não representam propriamente quais elementos desempenham os papéis de conhecimentos (MARTINS, 2009). Conforme discutido no

Capítulo 2, em OntoUML, classes do tipo *role* devem ser especializações de classes do tipo *kind*, Contudo, as classes do tipo *role* representadas na Figura 4.1 (p.ex., **Gerente de Configuração**) não estão associadas a nenhuma classe do tipo *kind*. Dessa forma, os modelos da ontologia de tarefa só estarão ontologicamente corretos, quando seus conceitos forem relacionados a conceitos de uma ontologia de domínio, que indicará quem vai desempenhar os papéis especificados na ontologia de tarefa. Isso é feito na Seção 4.4. O diagrama da Figura 4.1 representa as informações referentes a duas tarefas: selecionar itens de configuração e marcar linha base. Essas tarefas não ocorrem dentro de um fluxo de controle definido no processo de Gerência de Configuração e, por isso, não foram representadas em um modelo comportamental. Mas como está indicado, a atividade "Marcar Linha Base" é realizada por um **Gerente de Configuração**, indicando a **Configuração** a ser marcada e a tornando uma **Linha Base**. A atividade "Selecionar Itens de Configuração" também é realizada pelo **Gerente de Configuração**, que seleciona um conjunto de **Itens** que terão sua configuração gerenciada, dando origem a um conjunto de **Itens de Configuração**.

A Figura 4.2 ilustra um conjunto de item de diferentes tipos (documentos, diagramas, código fonte, produto de software).

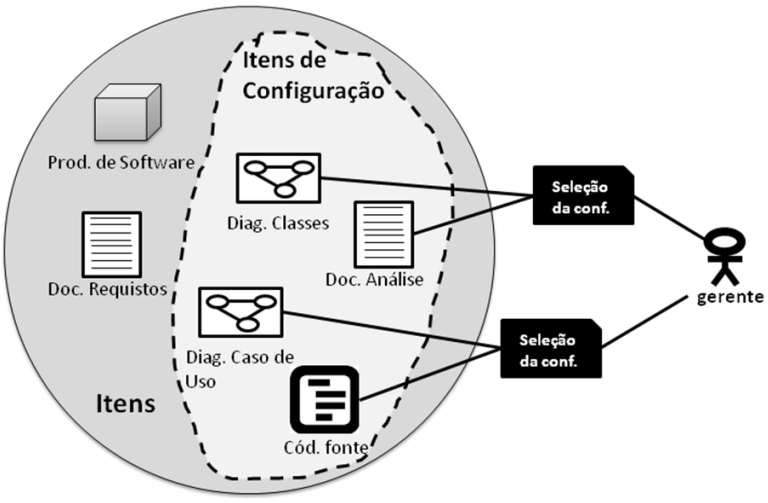

Figura 4.2 - Ilustração dos conceitos *Item*, *Item de Configuração* e *Seleção da Configuração*

Alguns deles estão sob gerência de configuração, enquanto outros não. Os que estão (dentro da parte pontilhada), estão relacionados com entidades do tipo seleção de configuração. Já a Figura 4.3 ilustra a composição de Itens de Configuração e Versões. Cada um dos dois ICs destacados (Diagrama de Classes e Documento de Análise)

representa um IC atômico, mas que juntos formam um IC composto. Tanto os ICs atômicos quanto o composto possuem versões, sendo que os ICs atômicos possuem versões atômicas, enquanto o IC composto possui uma configuração. A figura ainda ilustra a marcação de uma configuração feita por um gerente de configuração. A configuração que sofre a marcação é destacada de preto, representando uma linha base. Essa configuração (como as outras) é composta de outras versões, como está representado por meio de setas tracejadas.

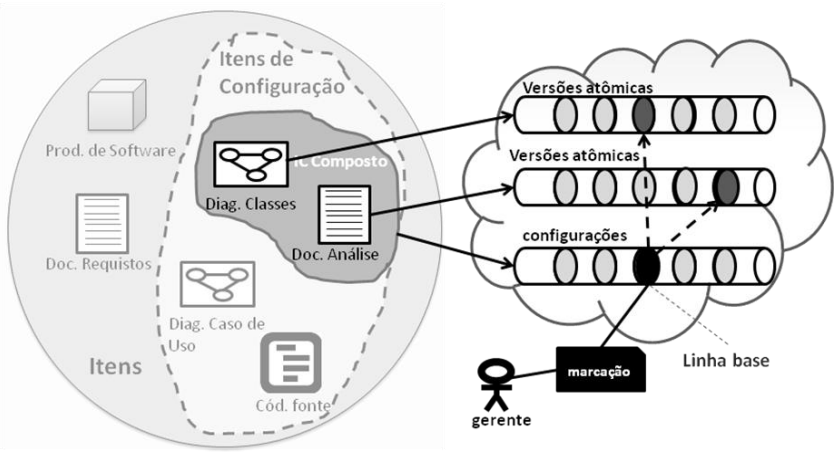

Figura 4.3 - Ilustração dos conceitos Item de Configuração Compostos, Configurações e Linha Base

4.3.2. Controle de Alteração

Como dito anteriormente, o controle de alteração é uma tarefa essencial da GC. Ela é composta de atividades de solicitação de alteração, avaliação de solicitação, implementação e verificação da implementação. Foram representados para essa parte três diagramas. Um diagrama de classes referente à parte estrutural, apresentado na Figura 4.4, e dois diagramas de atividades referentes ao fluxo de controle, apresentados na Figura 4.5 e na Figura 4.6.. O primeiro diagrama apresenta as atividades gerais do controle de alteração, citadas anteriormente. Já o segundo diagrama é um detalhamento da atividade de implementação da alteração, composta das ações *Realizar Checkout*, *Realizar Modificação* e *Realizar Checkin*.

Como mostra a Figura 4.4, um **Solicitador** realiza uma **Solicitação de Alteração** em um conjunto de versões de itens um produto, as quais passam a desempenhar o papel de **Versões Submetidas a Alteração**, indicando as **Alterações** a serem realizadas. Uma **Alteração** é uma especificação de um problema a ser resolvido ou de

uma melhoria a ser feita no produto. Esse evento é representado pelo *relator* **Solicitação de Alteração**, do qual deriva uma relação material ternária entre o solicitador, as alterações solicitadas e as versões submetidas.

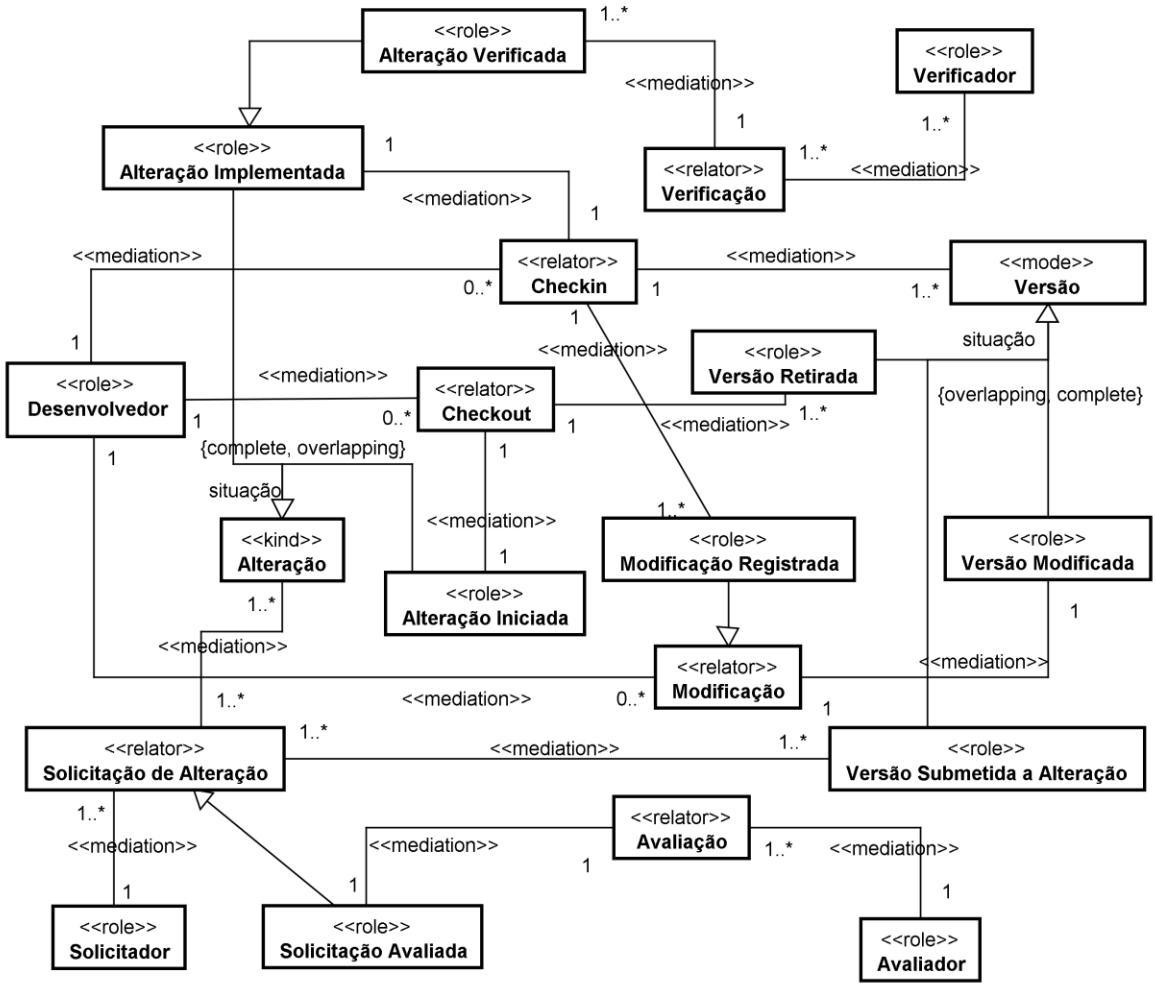

Figura 4.4 - Diagrama de Classes relativo aos papéis de conhecimento envolvidos no Controle de Alteração.

Um **Avaliador** é responsável por avaliar uma solicitação de alteração realizada. Essa ação é representada por uma relação material cujo relator é uma **Avaliação**. Neste momento, a solicitação passa a ser uma **Solicitação Avaliada**. **Desenvolvedor** é o papel responsável por implementar uma **Alteração** requisitada. Para tal, o **Desenvolvedor** realiza um **Checkout** de um conjunto de **Versões** de acordo com uma Alteração. Neste momento, as versões passam a desempenhar o papel de **Versões Retiradas** e a alteração passa a ser uma **Alteração Iniciada**. O *relator* **Checkout** representa o registro

dessa ação, relacionando o responsável, as versões retiradas e a alteração que foi iniciada.

A seguir, o desenvolvedor pode modificar as versões retiradas. O *relator* **Modificação** é uma representação dessa ação, registrando o **Desenvolvedor** que realizou a modificação e a **Versão Modificada**.

Em seguida, o desenvolvedor é responsável por registrar as novas versões criadas. O relator **Checkin** representa essa ação, registrando o **Desenvolvedor** responsável pelo *checkin*, as modificações que foram registradas (**Modificação Registrada**), o conjunto de **Versões** geradas e a alteração correspondente, qual passa a ser uma **Alteração Implementada**.

Por fim, o papel **Verificador** é responsável por verificar se uma **Alteração Implementada** foi realizada de acordo com o especificado. Essa ação é representada por meio do relator **Verificação** e, quando ela ocorre, a alteração se torna uma **Alteração Verificada**.

Para simplificação do diagrama, os papéis do Desenvolvedor foram omitidos, a saber: *Responsável por Checkout*, *Responsável por Modificação* e *Responsável por Checkin*.

O diagrama de classes da Figura 4.4 representa a perspectiva estrutural do controle de alteração. Entretanto essa perspectiva precisa ser complementada com a perspectiva comportamental da tarefa, representada pelo diagrama de atividades da Figura 4.5.

O diagrama de atividades da Figura 4.5 foi dividido em partições. Em cada uma delas está representada a atuação de um papel diferente. O processo inicia-se com a ação “Solicitar Alteração”, realizada por um **Solicitador**. Essa ação tem como entrada um conjunto de **Versões** e gera como saídas uma **Solicitação de Alteração**, a **Alteração** a ser realizada e a mudança do conjunto de versões selecionadas, que passa a ser tratado como um conjunto de **Versões Submetidas a Alteração**.

Em seguida (mas não imediatamente depois), um **Avaliador** avalia a **Solicitação de Alteração** realizada. Para fazer a avaliação, o **Avaliador** utiliza a **Solicitação de Alteração** e sua correspondente **Alteração** e gera uma **Avaliação**, além de alterar a situação da solicitação para **Solicitação Avaliada**. Caso a avaliação seja por rejeitar a

solicitação, a atividade é finalizada; caso contrário, a implementação da alteração pode ser realizada.

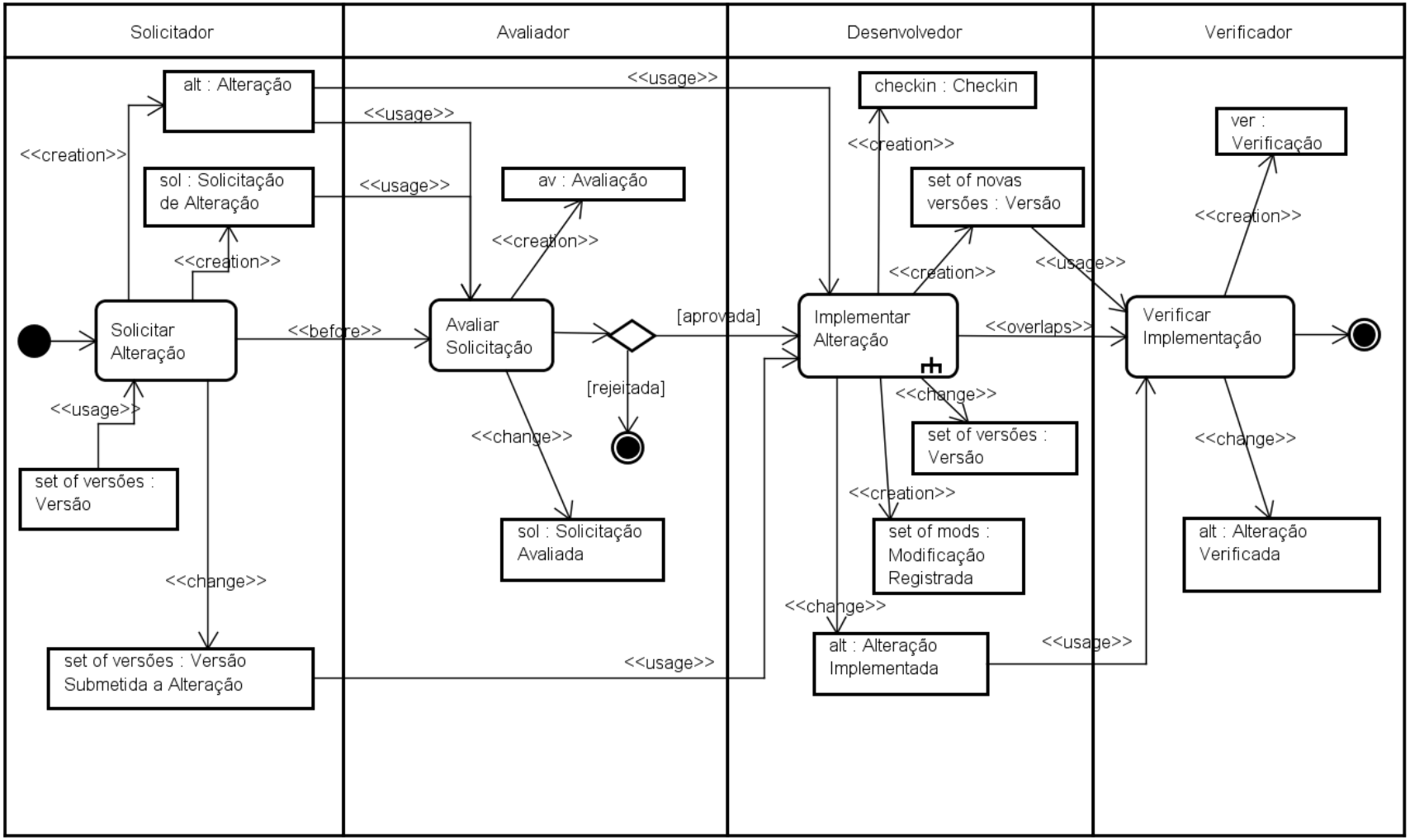

Figura 4.5 -Diagrama de Atividades da Tarefa de Gerência de Configuração

Para implementar uma alteração, o **Desenvolvedor** utiliza a especificação da **Alteração** e as **Versões Submetidas a Alteração**. Como resultado, são geradas as novas **Versões** dos itens do produto. Além disso, a **Alteração** passa a ser considerada uma **Alteração Implementada**.

Se sobrepondo parcialmente à realização da implementação, o **Verificador** pode verificar uma **Alteração Implementada**, checando se a alteração que foi implementada está em conformidade com o especificado. Para tal, o **Verificador** utiliza as novas **Versões** geradas e a **Alteração Implementada**. Como saída desta atividade, a alteração está verificada e é criado um registro da **Verificação**.

Como dito acima, as ações de implementação e de verificação se sobrepõem, sendo que a implementação sempre inicia primeiro que a verificação, como indica o estereótipo *overlaps*.

Ainda com relação ao diagrama da Figura 4.5, vale ressaltar que todas as ações são do tipo Ação de Agente com Objeto, sendo que a ação “Implementar Alteração” é também uma Atividade de Agente e, portanto, precisa ser detalhada em outro diagrama. O modelo comportamental apresentado na Figura 4.6 mostra esse detalhamento. Todas as ações nesse diagrama são do tipo Ação de Agente com Objeto, com precedência não imediata (estereótipo *before*) e desempenhadas pelo **Desenvolvedor** (omitido no diagrama).

Inicialmente o desenvolvedor realiza o *checkout* de um conjunto de versões a serem modificadas. Para realizar essa atividade, são necessários a alteração que será realizada e o conjunto das versões a serem retiradas. Como resultado da ação, é gerado o *relator* **Checkout**, que representa um registro da retirada de versões. Além disso, a alteração é iniciada e as versões são retiradas.

Após o *checkout*, o desenvolvedor modifica as versões retiradas anteriormente, gerando um conjunto de versões modificadas. Para cada versão modificada, é criado um registro de sua **Modificação**.

Finalmente, após a realização de todas as modificações necessárias, o desenvolvedor realiza o *checkin* das versões modificadas. Neste momento, é criado um conjunto de novas versões, baseadas nas versões modificadas, e é criado também o registro da ocorrência do **Checkin**. Além disso, as modificações passam a estar registradas, devido ao *checkin* realizado, e as versões modificadas voltam para sua situação de versões apenas, uma vez que as modificações foram incorporadas nas novas versões criadas.

A ação “Realizar Checkin” também destrói o relator **Checkout**, responsável por indicar que uma versão está em alteração. Essa destruição está representada pelos fluxos de objetos com estereótipo *termination*.

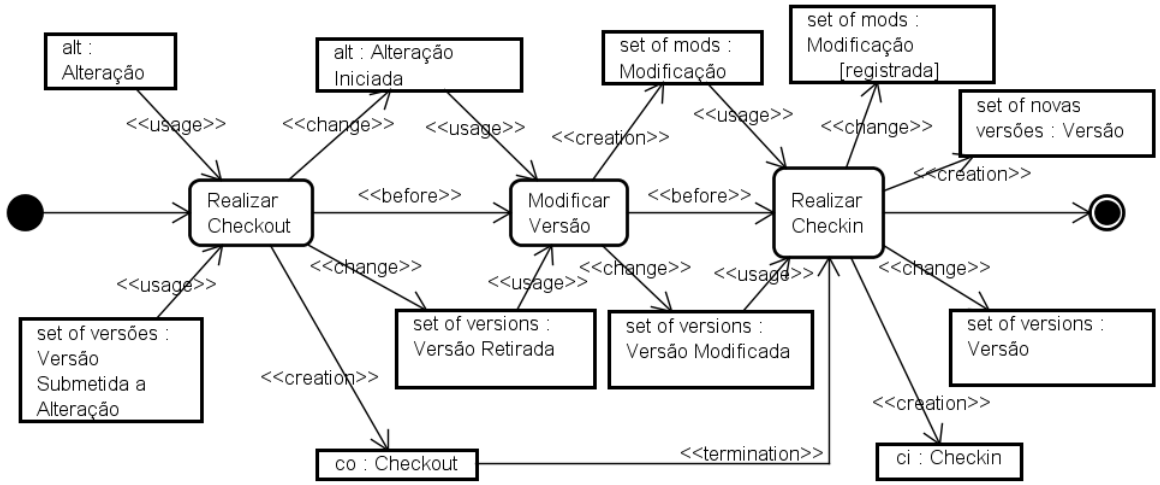

Figura 4.6 -Diagrama de atividades da atividade complexa “Implementar Alteração”.

Na realização do processo de controle de alteração, à medida que as ações vão sendo realizadas, novos objetos vão sendo criados e o estado de outros vai sendo alterado. A Figura 4.7 representa os estados de objetos relativos ao controle de alteração em função do tempo.

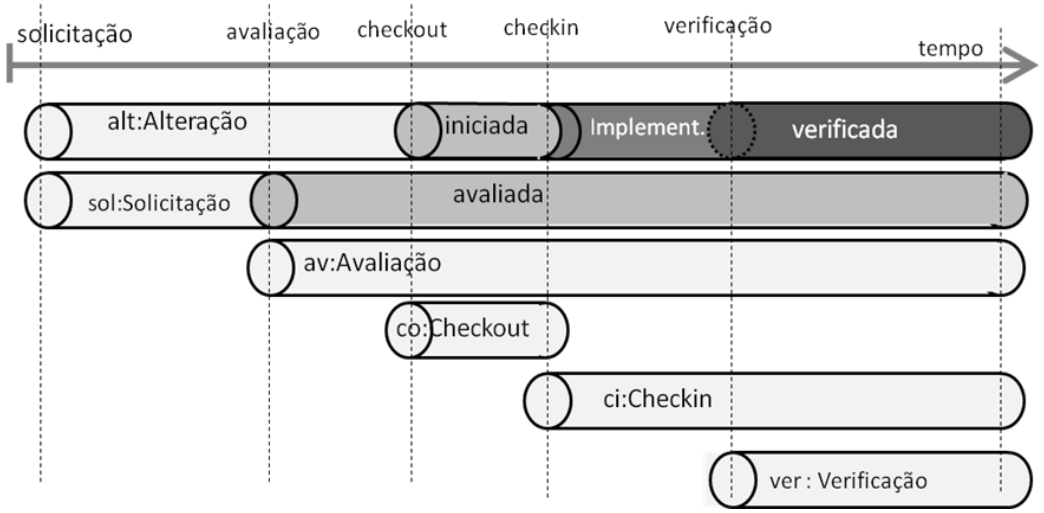

Figura 4.7-Estados dos objetos relativos aos conceitos alteração, solicitação, avaliação, checkout, checkin e verificação, em função do tempo

Inicialmente, a alteração *alt* e a solicitação *sol* são criadas quando da ocorrência da atividade “Solicitar Alteração”. Depois, por meio da atividade ”Avaliar Solicitação”, é criada a avaliação *av*, a qual é relacionada à solicitação *sol*. A solicitação passa, então, a ser uma solicitação avaliada. Quando a atividade “Implementar Alteração” é realizada, é realizado um

checkout *co*, fazendo com que a alteração desempenhe o papel de alteração iniciada. Depois alteração *alt* passa a desempenhar o papel de alteração implementada, devido a um *checkin* relacionado à alteração. Por fim, com a realização da atividade “Verificar Implementação”, a verificação *ver* é criada e relacionada à alteração *alt*, que passa a desempenhar o papel de alteração verificada.

A Figura 4.8 ilustra a realização atividade de implementação, conforme descrito pela ontologia proposta.

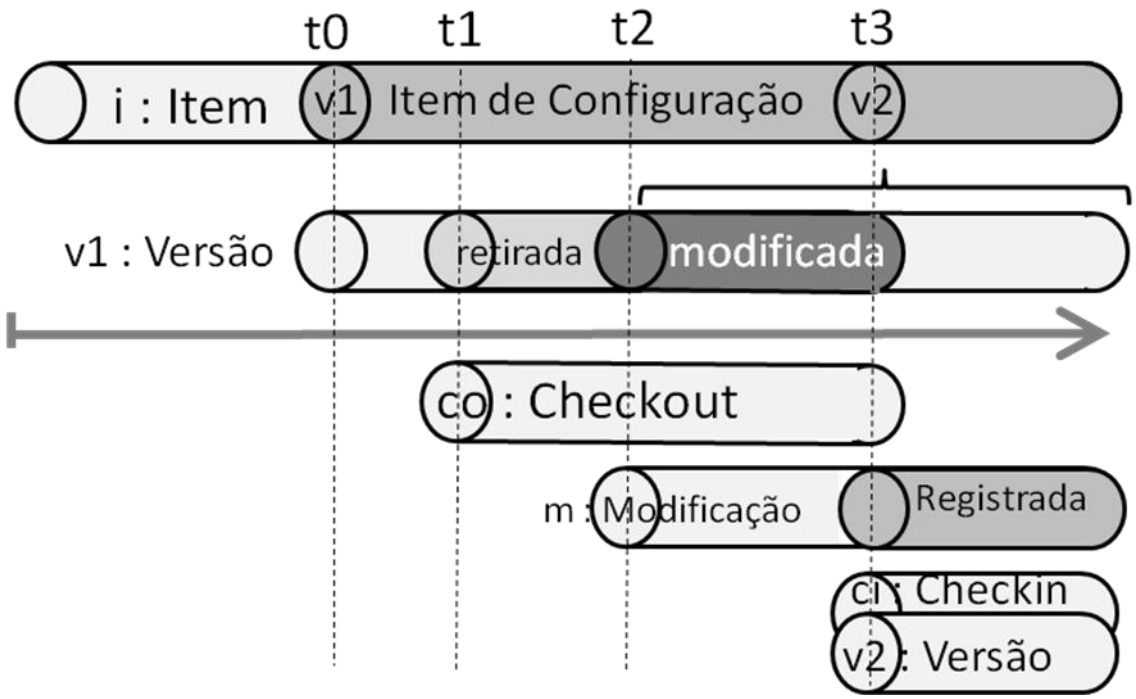

Figura 4.8 -Estados das instâncias dos conceitos Item, Versão, Checkout, Checkin e Modificação

Inicialmente o item *i* não desempenha nenhum papel. No tempo *t0*, devido à ocorrência de uma seleção de itens a terem sua configuração gerenciada, o item *i* passa a desempenhar o papel de **Item de Configuração**, possuindo uma **Versão** inicial *v1*. Neste momento, a versão *v1* não desempenha nenhum papel. No tempo *t1*, quando um *checkout* é realizado, a versão *v1* é retirada para modificação. Um relator referente ao *checkout* é criado e relacionado à versão *v1* que passa a desempenhar o papel de **Versão Retirada** (apenas versões retiradas podem ser modificadas). Com a modificação de *v1*, no tempo *t2*, o relator modificação *m* é criado e relacionado a *v1*. A versão *v1* passa, então, a desempenhar o papel de **Versão Modificada**. Por fim, no tempo *t3*, quando ocorre o *checkin*, o *relator co* referente ao *checkout* é destruído. Além disso, o relator *m* referente à modificação é marcado como registrado. Com isso a versão *v1* deixa de desempenhar o papel de retirada (apenas versões

retiradas podem sofrer checkin). Além disso, o checkin cria uma nova versão do item *i* chamada *v2*. Um relator checkin *ci* é criado e relacionado à versão *v2*.

Observe que os diagramas de classe da ontologia apresentada modelam os papéis de conhecimento envolvidos na tarefa de GC. Entretanto eles não especificam quem desempenha esses papéis, que são importantes na integração semântica. Como foi dito, a ontologia de tarefa, por modelar aspectos comportamentais, está diretamente relacionada com a integração nas camadas de serviço e processo. Porém a utilização apenas de ontologias de tarefa pode não ser suficiente para a integração semântica, pois ela não considera informações acerca do domínio. Dessa forma, além de ontologias de tarefa, é importante considerar ontologias de domínio, principalmente para ajudar na integração semântica na camada de dados.

Assim, para uma melhor integração semântica, neste trabalho a ontologia de tarefa de GC apresentada foi combinada com uma ontologia de domínio de Gerência de Configuração de Software, conforme discutido a seguir.

4.4. Relacionando a Ontologia de Tarefa a uma Ontologia de Domínio

Ontologias de tarefas são responsáveis por representar informações de tarefas genéricas, independentes do domínio no qual elas estão inseridas, tais como as tarefas de locação, diagnóstico e venda. Essas tarefas possuem informações independentes de domínio que estão relacionadas ao fluxo de controle e aos papéis de conhecimento da tarefa.

No caso da ontologia de tarefa de GC, não foi estabelecido quem desempenha os papéis de conhecimento representados por ela. Por exemplo, não se estabeleceu quem desempenha o papel de solicitador, de desenvolvedor, avaliador nem o de gerente de configuração. Além disso, a ontologia não estabelece quais são os tipos de itens que podem ter sua configuração gerenciada. Ou seja, ela deixa em aberto os agentes bem como os objetos envolvidos, já que essas informações são dependentes de domínio.

A ontologia de tarefa de GC poderia ser relacionada a diferentes domínios como, por exemplo, de construção civil ou criação musical. Os papéis de conhecimento seriam representados por diferentes elementos em função do domínio ao qual a tarefa de GC for relacionada. Se, por exemplo, o domínio for de criação musical, os itens poderiam ser músicas, partituras, letras, álbum; o papel de desenvolvedor poderia ser representado por um compositor musical. Já no domínio de construção civil, os itens controlados poderiam ser

plantas, maquetes, ferramentas ou o próprio edifício e o papel de desenvolvedor poderia ser representado por engenheiros, arquitetos e pedreiros.

O intuito deste trabalho é a integração semântica de Sistemas de Gerência de Configuração de Software. Dessa forma é interessante relacionar a ontologia de tarefa de GC com o domínio de desenvolvimento de software. Existe uma ontologia de domínio de Gerência de Configuração de Software definida em (ARANTES; FALBO; GUIZZARDI, 2007) que pode ser usada para este fim. A Figura 4.9 apresenta a ontologia de domínio de GCS proposta em (ARANTES; FALBO; GUIZZARDI, 2007).

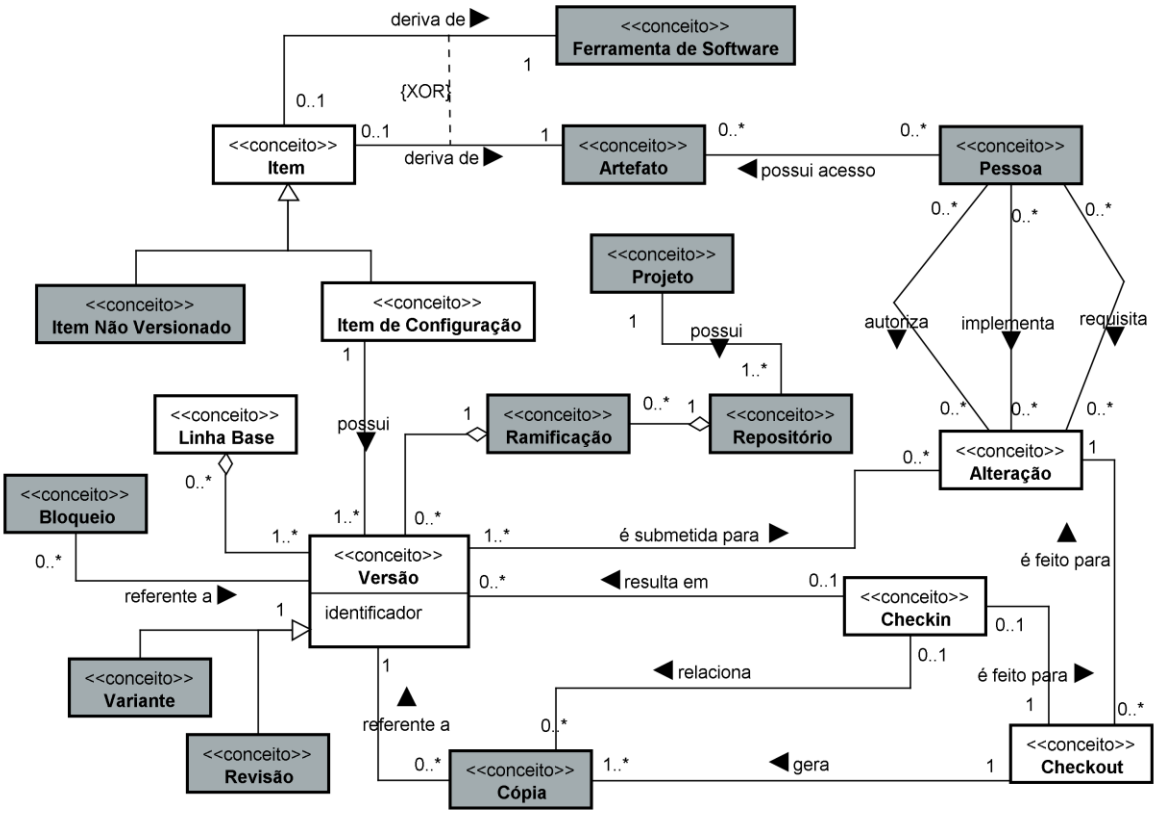

Figura 4.9– Ontologia de domínio de Gerência de Configuração de Software e o conceitos adicionados por ela (em destaque) (ARANTES; FALBO; GUIZZARDI, 2007)

Como as duas ontologias lidam com a gerência de configuração, existe uma interseção de conceitos equivalentes entre elas. Os conceitos em branco na ontologia de domínio representam esses conceitos, a saber: item, item de configuração, versão, alteração, checkin e checkout. Esses conceitos são considerados conceitos centrais da GC. Já os conceitos destacados de cinza representam conceitos que não estavam sendo considerados pela ontologia de tarefa, a saber: pessoa, artefato, ferramenta de software, projeto, repositório, ramificação, revisão, variante, cópia, bloqueio e item não versionado. Esses conceitos, por

sua vez, são mais específicos do domínio de desenvolvimento de software (ou representam conceitos não contemplados na ontologia proposta neste trabalho).

Como a figura mostra, um projeto possui um ou mais repositórios que são responsáveis por armazenar as versões dos itens de um projeto. Cada repositório é composto de ramificações que representam linhas de desenvolvimento diferentes. Cada ramificação é composta por versões de itens de configuração. Os itens de configuração representam itens do projeto que estão sendo controlados. Itens podem ser derivados de artefatos ou de ferramentas de software. Itens de configuração são compostos de versões que representam um estado de evolução do mesmo. Versões podem ser de dois tipos: Variantes (versões paralelas a outras) ou Revisões (versões que substituem outras). Versões podem estar bloqueadas para alteração por meio de um bloqueio.

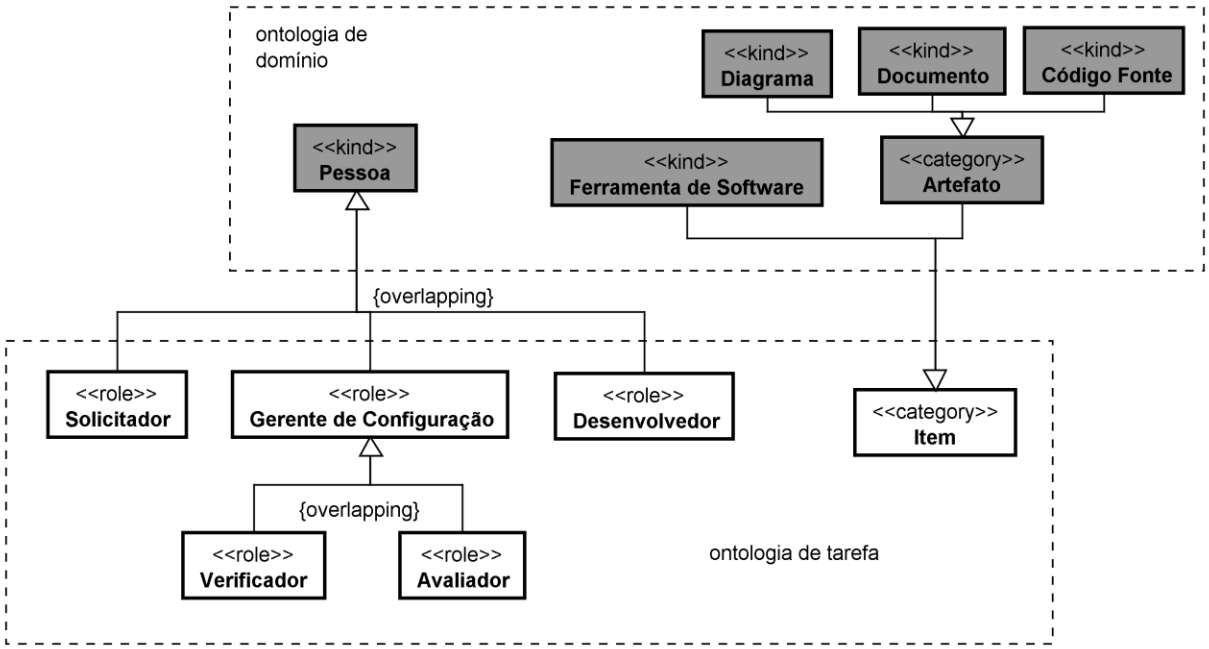

Figura 4.10– Ligação de complementação entre papéis da tarefa de GC e conceitos da GCS (representação dos papeis de conhecimento)

Uma linha base é um conjunto de versões de um projeto que serve como ponto de referência de um estado de desenvolvimento do produto. Uma pessoa possui acesso a artefatos e pode autorizar, implementar ou requisitar alterações nos mesmos. Uma alteração é responsável por submeter um conjunto de versões que serão modificadas por meio de cópias locais. As cópias das versões são geradas por meio de um checkout. As alterações nas versões são feitas por meio de suas cópias e registradas por meio de um checkin. A Figura 4.10 explicita a relação de complementação das duas ontologias, além das relações de equivalência apresentadas anteriormente.

Como mostra a figura, os papéis de desenvolvedor, gerente de configuração, verificador, avaliador e solicitador (da ontologia de tarefa) são desempenhados por uma pessoa (da ontologia de domínio). A ontologia também é responsável por fornecer os itens que podem estar sob gerência de configuração. Ferramentas de software e artefatos são os tipos que podem ser gerenciados, lembrando que artefatos podem ser documentos, diagramas e código-fonte.

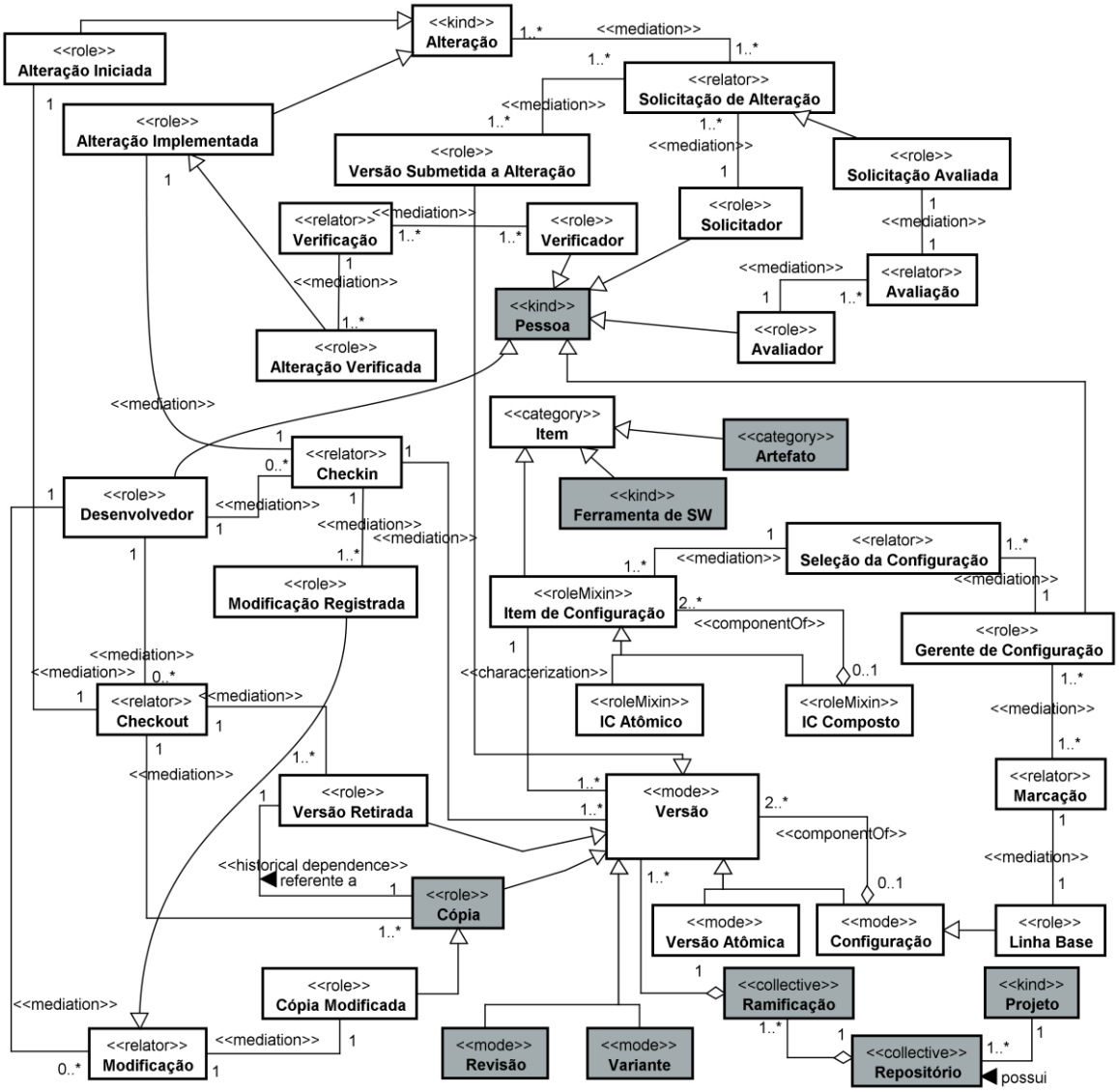

Figura 4.11 - Modelo estrutural da ontologia de classe de aplicação

Como resultado da integração da ontologia de domínio com o modelo estrutural da ontologia de tarefa, tem-se uma ontologia de classe de aplicação, cujo modelo estrutural é apresentado na Figura 4.11. No diagrama da Figura 4.11 os conceitos da ontologia de domínio adicionados ao modelo estrutural da ontologia de tarefa estão em destaque, a saber:

Cópia, *Pessoa*, *Artefato*, *Ferramenta de Software*, *Ramificação*, *Repositório*, *Projeto*, *Revisão* e *Variante*.

Os demais conceitos da ontologia de domínio já estavam presentes na ontologia de tarefa, a saber: *Item*, *Item de Configuração*, *Versão*, *Linha Base*, *Checkin*, *Checkout* e *Alteração.* Vale a pena ressaltar que, por conta de uma análise ontológica proveniente do uso de OntoUML, que incorpora distinções ontológicas de uma ontologia de fundamentação, algumas relações foram alteradas para alinhar as duas ontologias, como é o caso das relações entre Item e Artefato e entre Item e Ferramenta de Software. Além disso, foram omitidos nesse diagrama os tipos das especializações, bem como seus critérios, uma vez que tal informação foi apresentada nos outros diagramas. Esse modelo estrutural é usado como base para a iniciativa de integração semântica apresentada no capítulo seguinte.

Com relação ao modelo comportamental, ele também sofre alterações com a junção com a ontologia de domínio. No caso dessa integração, apenas o modelo da Figura 4.6 sofre uma pequena alteração. A tarefa de *Realizar Checkout* agora também gera um conjunto de cópias referentes às versões retiradas, a tarefa *Modificar Versão* consome essas cópias, modificando-as, e elas depois são registradas em *Realizar Checkin*.

4.5. Conclusão

Dentre os benefícios da integração e alinhamento entre as ontologias de domínio e tarefa de GC, está o enriquecimento da ontologia de domínio que, além de não estar fundamentada em uma ontologia de fundamentação, não considerava informações comportamentais, principalmente as relativas ao controle de alteração. A ontologia de tarefa, por exemplo, detalha alguns relacionamentos da ontologia de domínio, tais como os relacionamentos entre Pessoa e Alteração (“autoriza”, “implementa”, “requisita”). Na ontologia de domínio não estão representados os papéis das pessoas envolvidas nessas relações, nem as ações correspondentes, na forma de *relators*.

Outro benefício da integração das ontologias é a junção de informações comportamentais e estruturais permitindo uma maior abrangência da integração semântica nas camadas de dados, serviço e processo. Por fim, a ontologia de tarefa descreve mais detalhadamente os conceitos de controle de alteração, enquanto a ontologia de domínio abrange mais detalhadamente os conceitos relacionados ao controle de versão. Isso é muito importante neste trabalho, visto que as ontologias serão usadas na integração de ferramentas para apoiar tanto o controle de alteração quanto o controle de versão.

A ontologia de classe de aplicação resultante é bastante útil na solução de problemas de integração semântica no contexto de sistemas de apoio ao processo de GC. Principalmente porque os sistemas de GC, na maioria dos casos, são desenvolvidos de modo isolado e para apoiar apenas uma parte do processo de GC. Para uma melhor eficiência no desenvolvimento de software, esses sistemas devem trabalhar em conjunto para apoiar o processo como um todo.

O próximo capítulo mostra uma utilização da ontologia de classe de aplicação proposta neste trabalho na solução de problemas de integração semântica entre dois sistemas de apoio à GC.

Capítulo 5 - Estudo de Caso: Integração Semântica de Sistemas de Apoio à Gerência de Configuração de Software

No capítulo 3 foi apresentada a abordagem OBA-SI (*Ontology-Based Approach for System Semantic Integration*). OBA-SI é uma abordagem de integração semântica que concentra esforços na análise de requisitos da integração. Na proposta apresentada, a integração deve ocorrer em um alto nível de abstração, com intuito de realizar a integração semântica entre os sistemas no nível conceitual antes de se projetar e desenvolver a solução da integração. Uma premissa da abordagem é que a atribuição de semântica usando ontologias deve ser independente da solução da integração. A integração ocorre em três camadas: dados, serviço e processo. Para tal, os modelos conceituais e comportamentais dos sistemas e do processo de negócio são comparados à luz da ontologia de referência, que é usada para atribuir semântica.

O cenário geral de integração também é válido para organizações de software. Para apoiar todo processo de software, tipicamente vários sistemas são usados, cada um deles apoiando uma ou mais atividades relacionadas. Dessa forma para ilustrar a abordagem apresentada no Capítulo 3, é apresentado neste capítulo um caso real de integração dos sistemas de controle de versão Subversion (SVN) (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2011) e apoio à Gerência de Configuração de Software do ambiente ODE, GCS-ODE (CALHAU, 2009), com o intuito de apoiar o processo de Gerência de Configuração de Software (GCS). Para isso, foi usada a ontologia da tarefa de Gerência de Configuração em conjunto com a ontologia de domínio de GCS, apresentados no Capítulo 4.

Este capítulo está organizado da seguinte forma: Seção 5.1 apresenta o contexto e a motivação para esta iniciativa de integração; a Seção 5.2 descreve o cenário de integração, apresentando os modelos conceituais dos sistemas e do processo considerados; a Seção 5.3 apresenta o processo de integração; por fim, na Seção 5.4 são apresentadas as conclusões do capítulo.

5.1. Contexto da Iniciativa de Integração

Esta iniciativa de integração tem como intuito a integração de ferramentas ao ambiente ODE. ODE é um ambiente de desenvolvimento de software baseado em ontologias e que, com base nelas, integra diferentes ferramentas de apoio à Engenharia de Software. O objetivo de tais ontologias no contexto do Projeto ODE é descrever o domínio de Engenharia de

Software de modo claro e preciso, facilitando a integração de ferramentas desenvolvidas no contexto do projeto (FALBO; RUY; DAL MORO, 2005). Neste sentido, ontologias vinham sendo usadas como especificações de conceituações que podem ser reutilizadas no desenvolvimento de diferentes ferramentas produzidas no contexto do projeto.

Dentre as ferramentas que compõem o ambiente ODE, existia uma ferramenta de gerência de configuração proposta em (NUNES, 2005). Essa ferramenta provia funcionalidades básicas para apoiar a gerência de configuração de software, principalmente o controle de alteração, suporte a solicitações e controle de acesso. Entretanto ela não provia suporte a outras atividades da gerência de configuração, tal como o controle de versão, que é uma importante tarefa no desenvolvimento de software.

Iniciou-se então um estudo para prover tais funcionalidades de controle de versão na ferramenta uma vez que o controle de versão é uma parte importante da gerência de configuração. O sistema de GCS de ODE foi então reformulado, dando origem a uma nova versão chamada GCS-ODE (CALHAU, 2009).

Entretanto, prover funcionalidades de controle de versão não é uma tarefa fácil, pois pode envolver tarefas complexas tais como armazenamento eficiente de artefatos e suas versões em um repositório, sincronização de alterações que ocorrem em paralelo, acesso remoto ao repositório, controle de concorrência e transação para salvar alterações de arquivos.

Existem diversos sistemas usados no desenvolvimento de software que já desempenham esse papel. Decidiu-se então, após um levantamento de sistemas de controle de versão, integrar o sistema *Subversion* a GCS-ODE para prover tais funcionalidades. Como resultado dessa integração, obteve-se um sistema de GCS mais completo tendo o GCS-ODE atuando principalmente no apoio ao controle de alteração e *Subversion* apoiando principalmente o controle de versão (CALHAU, 2009).

A integração realizada envolveu as camadas de dados e serviços, porém a integração semântica envolveu apenas a camada de dados, a qual tomou por base a ontologia de domínio citada. Além disso, a integração semântica ocorreu de maneira *ad-hoc*, sem seguir nenhuma diretriz pré-estabelecida.

As dificuldades encontradas e um estudo de trabalhos de integração semântica despertaram para a necessidade de uma abordagem de integração semântica baseada em ontologias que fosse independente de tecnologia, que propusesse um processo de integração

e abrangesse as três camadas de integração. Além disso, a integração deveria ocorrer em um nível mais alto de abstração. Ela deveria ser independente de como a solução de integração seria posteriormente projetada e implementada.

5.2. Descrição do Cenário de Integração

O cenário de integração envolve os sistemas Subversion e GCS-ODE, a serem integrados para apoiar o processo de GCS definido no âmbito do Núcleo de Estudos em Modelagem Conceitual e Ontologias – NEMO, na UFES. Subversion é um famoso sistema de código aberto de controle de versão de arquivos, que gerencia arquivos e diretórios, bem como, suas respectivas alterações ao longo do tempo. GCS-ODE apoia o controle de alterações dentro do ambiente ODE (*Ontology-based Development Environment*) e oferece funcionalidades para solicitação e avaliação de alteração, avaliação de impacto de alteração e de controle de acesso. A integração desses sistemas tem o intuito de apoiar especificamente o processo de GCS definido, o qual procura atender aos resultados esperados desse processo no modelo MPS-BR, nível F (SOFTEX, 2011).

O principal objetivo da integração SVN/GCS-ODE é integrar funcionalidades de controle de alteração com de controle de versão. Ela surgiu da necessidade de controlar a versão de modo eficiente dos artefatos gerenciados no ambiente ODE (do conteúdo deles). Esse é o principal requisito dessa integração. A partir de tal requisito, novos requisitos foram sendo definidos à medida que estudos de sistemas de controle de versão foram sendo feitos. Através deles, percebeu-se a necessidade de permitir em GCS-ODE que alterações ocorressem de modo simultâneo, por meio de mecanismos de sincronização de alteração, e que elas ocorressem de modo isolado, localmente, sem impactar as versões estáveis do produto. Outros requisitos da integração que foram identificados: registrar linhas base do produto e os estados de suas ramificações, bem como do conteúdo dos arquivos correspondentes.

Para a realização da integração seguindo a abordagem proposta, foi necessário um estudo dos sistemas e do processo envolvidos na integração, o que deu origem a modelos conceituais que são responsáveis por descrevê-los. Tais modelos, apresentados a seguir, junto com a ontologia apresentada no capitulo anterior, são a base para a integração semântica, conforme propõe OBA-SI.

5.2.1. Processo de negócio

O processo de GCS considerado no cenário de integração é basicamente composto das seguintes atividades: *Planejamento da GCS*, *Identificação de ICs*, *Controle da Configuração*, *Efetuação de Auditoria de Configuração* e *Gerencia de Liberação e Entrega*. Cada uma de tais atividades está relacionada em um fluxo de controle além de serem compostas de sub-atividades. A Figura 5.1 apresenta as principais atividades do processo de GCS, apresentando o fluxo de controle, bem como os produtos e insumos de cada atividade.

O processo inicia com o planejamento da gerência de configuração, atividade que tem como insumo o plano de projeto e gera um plano de GCS. Depois é realiza a atividade identificar itens de configuração. Essa atividade tem como objetivo selecionar os itens que serão colocados sob Gerência de Configuração e definir a sua constituição e o nível de controle apropriado. Para tal, ela utiliza o plano de GCS e um conjunto de artefatos e gera como produto um conjunto itens de configuração (ICs) identificados, bem como uma alteração no plano de GCS, através da adição dos ICs nela identificados. Depois ocorre a atividade de controle de configuração, qual, a partir do conjunto de ICs identificados, gera um conjunto de solicitações realizadas, modificações realizadas, linhas base atualizadas e um conjunto de novas versões de IC registrados. Por fim, ocorrem as atividades de efetuar auditoria de configuração e gerenciar liberação e entrega.

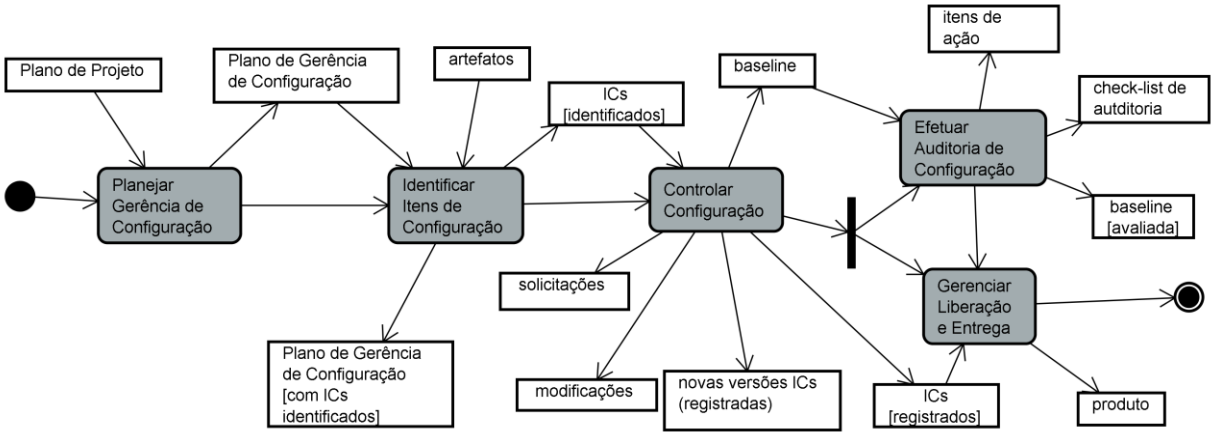

Figura 5.1 - Modelo comportamental do processo de negócio de GCS

Tais atividades citadas são compostas de sub-atividades, que serão omitidas aqui, com exceção da atividade de controlar configuração, que é a atividade central da realização da integração, junto com a identificação da configuração. Ela visa controlar a evolução dos ICs, por meio do controle de solicitações de modificação, analisando o seu impacto e

notificando os afetados, de modo a evitar retrabalho e efeitos colaterais indesejáveis, e por meio do controle das versões produzidas como resultados das modificações.

Conforme mostra a Figura 5.2, a atividade de controle de configuração é composta das seguintes sub-atividades: *Efetuar Solicitação de Modificação*, *Classificar Solicitação*, *Avaliar Impacto da Modificação*, *Avaliar Solicitação*, *Retirar ICs*, *Implementar Modificação*, *Acompanhar Modificação*, *Registrar Modificação* e *Verificar Modificação*.

Solicitações e modificações são geradas quando uma solicitação de modificação é efetuada. A solicitação é classificação na etapa seguinte, enquanto a modificação é avaliada na etapa de avaliação de impacto, onde a proposta de implementação é gerada. A solicitação classificada é avaliada na etapa de avaliação de solicitação, que também gera uma avaliação e atribui um responsável pela modificação. Depois, itens de configuração envolvidos com a modificação são retirados. Eles são, então, modificados na etapa de implementação da modificação e registrados na etapa de registro de modificação, quando são geradas novas versões deles. Por fim, tais ICs registrados são verificados com base na proposta de implementação.

Outras atividades envolvidas na etapa de controle da configuração, mas que não foram representadas nos modelos por não estarem dentro do fluxo de controle, são: *Criar Linha Base*, *Atualizar Linha Base* e *Registrar Novos ICs e Versões*. A identificação de ICs e versões a serem registrados, por exemplo, pode ocorrer a qualquer momento, e a criação de linhas base ocorre em períodos pré-estabelecidos no projeto, não estando, portanto, dentro de um fluxo de controle bem definido.

É possível notar que os modelos apresentados que representam o processo de negócio de GCS envolvem apenas a parte comportamental, estabelecendo apenas atividades, insumos, produtos e fluxo de controle. Algumas dessas atividades serão apoiadas por sistemas, principalmente as referentes ao controle de versão e ao controle de alteração.

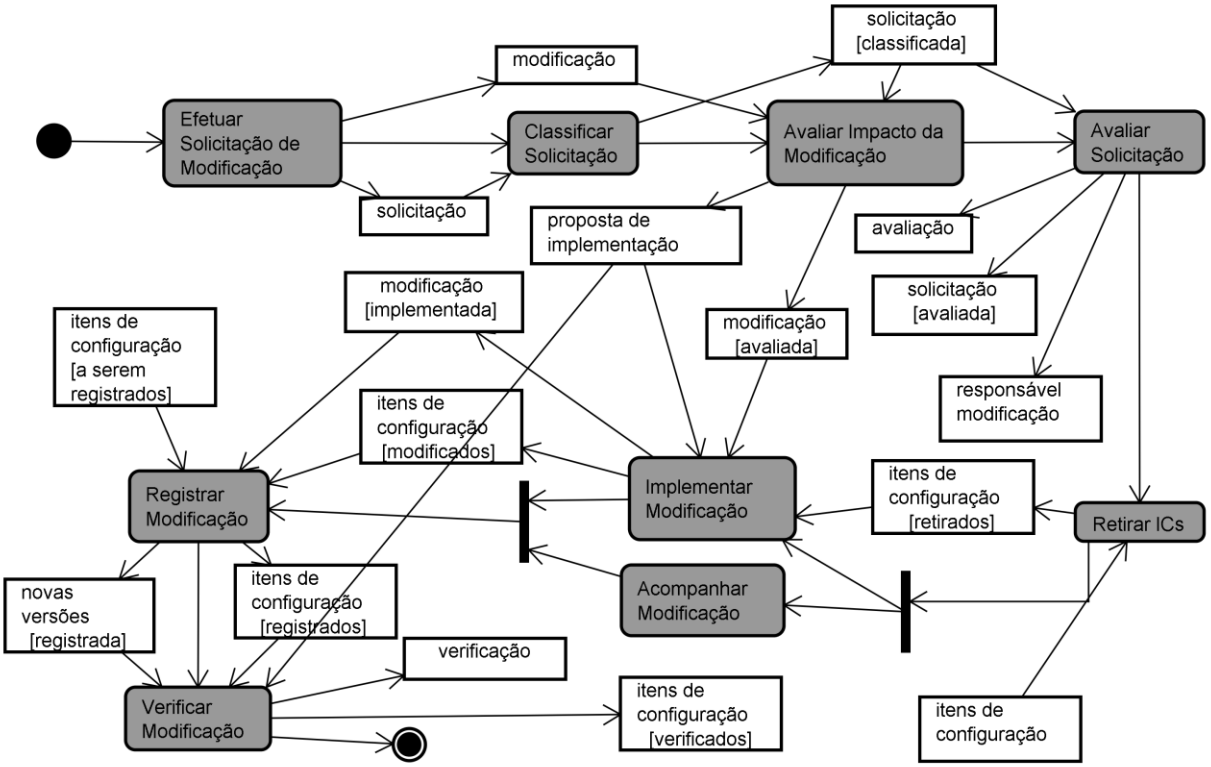

Figura 5.2 - Sub-atividades da atividade de Controlar Configuração, do Processo de GCS

5.2.2. Subversion

O Subversion (SVN) foi um dos sistemas escolhidos para apoiar o processo descrito na subseção anterior. SVN (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2011) é um sistema de controle de versão de arquivos e diretórios. Ele permite o gerenciamento de arquivos e diretórios ao longo do tempo por meio de um repositório central. Tal repositório pode ser acessado através da rede, permitindo que vários usuários tenham acesso a ele, a partir de máquinas diferentes.

O SVN usa a estratégia chamada de *copia-modifica-mescla* na realização de alterações nos arquivos. Nessa estratégia, cada usuário trabalha de forma simultânea e independente através de cópias de trabalho, um espelho local dos arquivos e diretórios de um repositório criado no momento de um *checkout*. Eles realizam as alterações em suas cópias locais, podendo inclusive adicionar, remover ou alterar a estrutura do sistema de arquivos. Para registrar as alterações no repositório (*efetuar commit*), eles devem antes mesclá-las com as alterações registradas por outros usuários. SVN oferece, ainda, ajuda na mesclagem de alterações em um mesmo arquivo.

O SVN permite que desenvolvedores trabalhem em cima do mesmo conjunto de arquivos ao mesmo tempo, visto que possuem mecanismos de mesclagem de alterações no registro das alterações, ou seja, oferecem controle de sincronização (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2011). Os modelos conceituais do SVN não eram conhecidos *a priori* e por isso foi necessária a escavação deles para a realização da integração para a realização da abordagem de integração semântica.

Apesar de SVN ser uma ferramenta de código aberto, é difícil obter seu modelo conceitual diretamente, visto que além do código ser grande, ele possui muitos detalhes de implementação que deveriam ser abstraídos. Dessa forma, para a extração do modelo conceitual do SVN, foram analisados principalmente os metadados das cópias de trabalho e os arquivos com o esquema XML de saída de linha de comando do SVN.

Modelo Estrutural

A Figura 5.3 apresenta o modelo conceitual estrutural extraído do SVN.

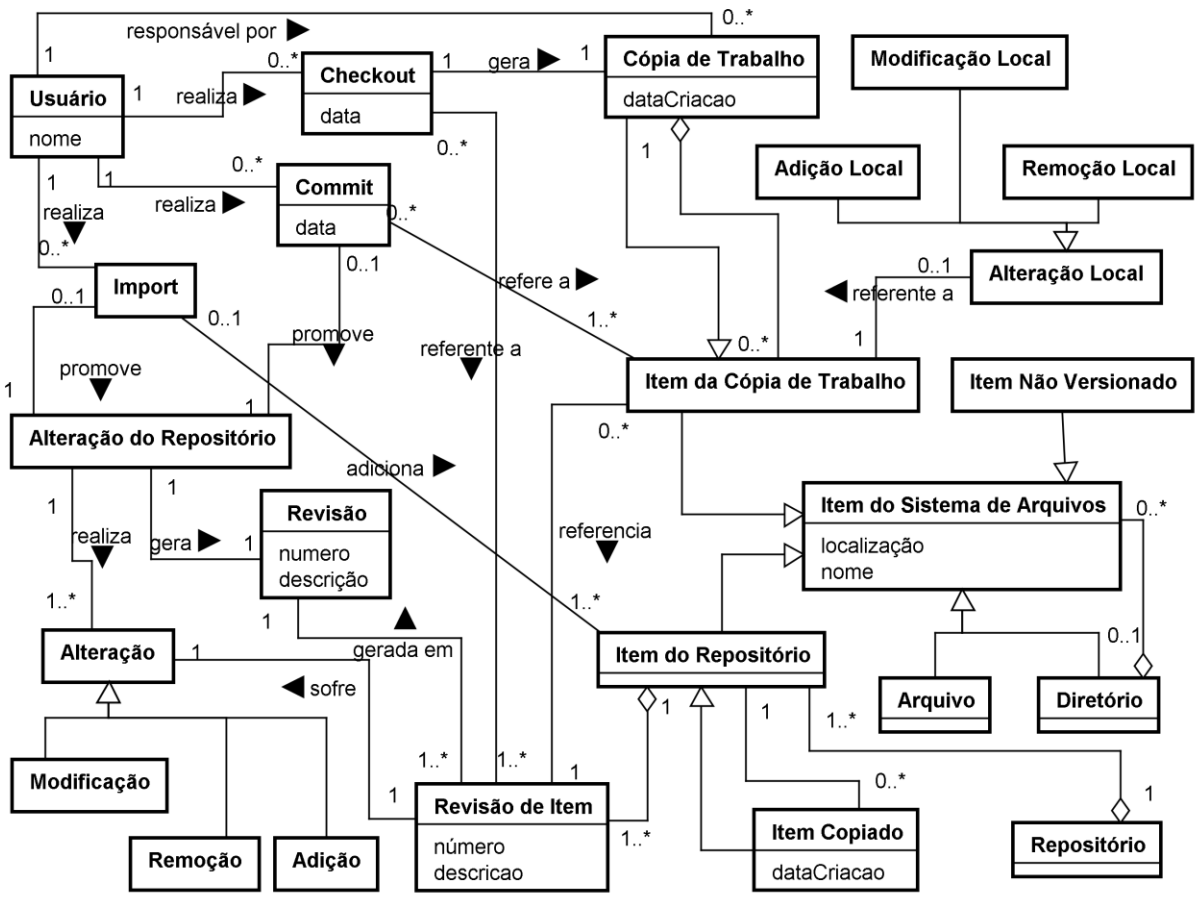

Figura 5.3 - Modelo Conceitual do Subversion

Um dos conceitos principais é o conceito de **Item do Sistema de Arquivos** (Item do S.A.), uma vez que o SVN é focado nele. Tal conceito representa arquivos e diretórios que compõem o sistema de arquivos, podendo estar dentro do repositório (**Item do Repositório**), onde suas versões e alterações são controladas, ou dentro de uma cópia de trabalho (**Item da Cópia de Trabalho** – Item da C.T.), representando localmente um item do repositório, ou fora deles (**Item Não Versionado** – Item N.V.). Um item do repositório pode ser uma cópia de outro (**Item Copiado**) ou não. Eles são compostos de uma ou mais **Revisões de Item,** que representam os estados deles. A revisão de um item é gerada em uma **Revisão** do repositório (um *snapshot* dele) em função de uma determinada **Ação** (adição, remoção ou modificação) realizada no item do repositório em uma **Alteração do Repositório**. As alterações nos itens do repositório ocorrem por meio de cópias locais (**Item da Cópia de trabalho**). Essas cópias locais são itens do sistema de arquivos gerados por meio de um **Checkout** realizado pelo usuário. O checkout gera uma cópia de trabalho que é um diretório onde as cópias locais são armazenadas. **Alterações Locais** são realizadas em itens da cópia de trabalho. Tais alterações podem ser uma **Modificação Local**, **Adição Local** ou uma **Remoção Local**. Depois de efetuadas localmente, as alterações são registradas por meio de um **Commit** realizado por um usuário. Itens não versionados do sistema de arquivo podem ser adicionados por meio de um **Import**.

Na Figura 5.3 foram omitidos os conceitos referentes aos arquivos e diretórios que são do tipo item não versionado (**Arquivo Não Versionado** e **Diretório Não Versionado**), do tipo item da Cópia de Trabalho (**Arquivo da C.T.** e **Diretório da C.T.**) e do tipo item do repositório (**Arquivo do Repositório** e **Diretório do Repositório**).

Modelo comportamental

SVN possui funcionalidades que permitem colocar itens (arquivos ou diretórios) em um repositório (como, por exemplo, as funcionalidades *Add* e *Import*) ou então remover itens dele (como a funcionalidade *Remove*). Os itens de um repositório são alterados por meio de cópias feitas pela funcionalidade *SVN Checkout*. Essa funcionalidade gera cópias locais que podem ser alteradas pelos desenvolvedores. Tais alterações são registradas posteriormente por meio da funcionalidade *Commit*. SVN ainda apoia outras funcionalidades como a de criação de repositórios, de atualização de cópias de trabalho, realização de cópias em itens do repositório ou de itens na cópia de trabalho. A Figura 5.4 mostra as funcionalidades do SVN relevantes para o cenário de integração sendo descrito.

Como a figura ilustra, é apresentada parte do modelo comportamental do SVN responsável pela realização de alterações. Como está apresentada na figura, a realização de alterações em SVN ocorre por meio das ações *Checkout*, *Realizar Alterações Locais*, *Atualizar Cópia de Trabalho* e *Commit*. Como está representado, um diretório do repositório é selecionado para o *checkout*, gerando uma cópia de trabalho, bem como um conjunto de itens da cópia de trabalho (arquivos e diretórios). Tais itens sofrem mais tarde alterações locais na ação *Realizar Alterações Locais*. Essa ação representa a realização de modificações em, adições de e remoções de itens da cópia de trabalho. Tais itens depois são atualizados e, por fim, são registrados por meio de um *Commit*, que gera uma alteração no repositório, novas revisões dos itens alterados, uma nova revisão do repositório e novos itens do repositório.

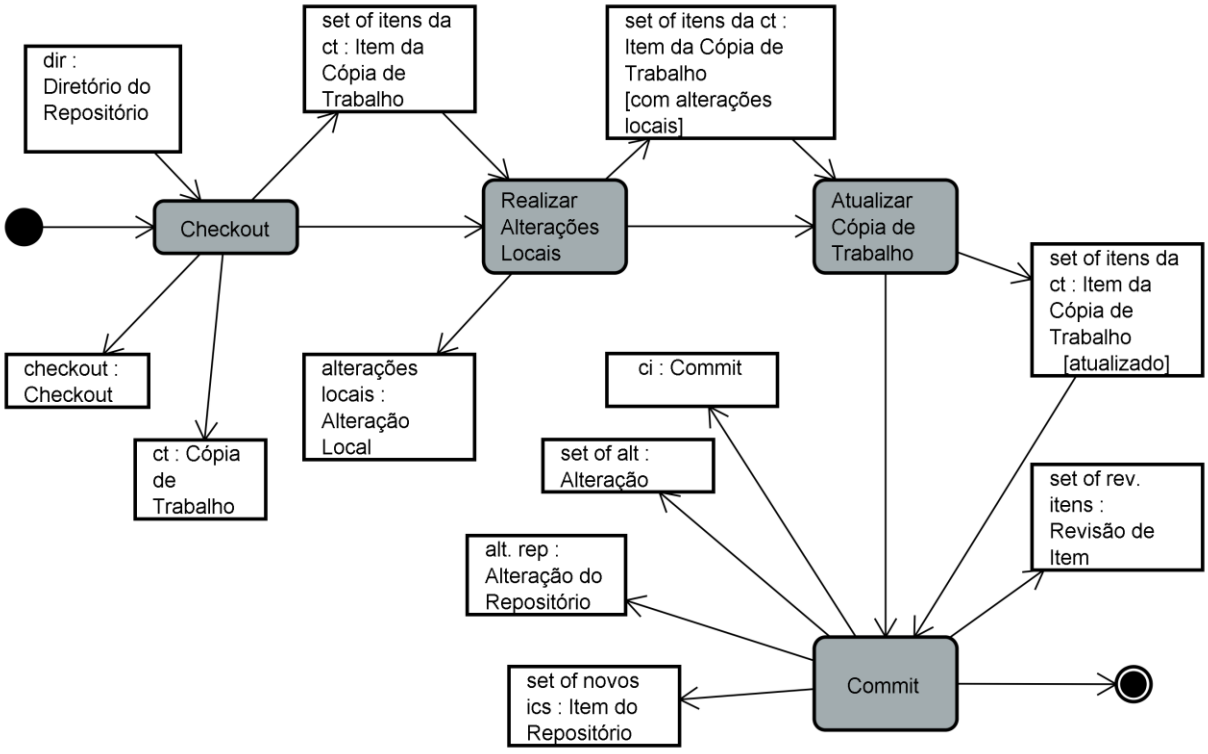

Figura 5.4 - Parte do modelo comportamental de SVN (referente a realização de alterações)

Algumas funcionalidades de SVN não foram representadas no modelo comportamental por não se encaixarem no fluxo de controle, pois elas não estão diretamente ligadas à realização de uma alteração de um conjunto de itens. Entretanto tais atividades podem fazer parte da integração. São elas:

- *Funcionalidades de Controle do Repositório:*
    - Copiar Item do Repositório;
    - Importar Diretório não Versionado;

- Remover Item do Repositório;
- Criar Repositório.

- *Funcionalidades de alteração de cópia de trabalho:*
  - Adicionar Item Não Versionado à Cópia de Trabalho;
  - Remover Item da Cópia de Trabalho;
  - Modificar Item da Cópia de Trabalho.

A funcionalidade de criação de um repositório é a primeira a ocorrer, mas ocorre uma única vez. Depois que o repositório é criado, as funcionalidades de controle do repositório podem ocorrer a qualquer instante. A qualquer momento, por exemplo, é possível importar itens no repositório. Já as funcionalidades de alteração de cópia de trabalho ocorrem após um *checkout*, no contexto da funcionalidade *Realizar Alterações Locais* .

5.2.3. GCS-ODE

GCS-ODE (CALHAU, 2009) é uma ferramenta de apoio à Gerência de Configuração de Software, focada na gerência de alterações nos artefatos de ODE. Tais artefatos podem ter sido produzidos internamente ao ambiente, por alguma ferramenta do próprio ambiente, ou externamente, por alguma ferramenta não integrada ao ambiente, sendo, nesse caso, representados na forma de arquivos. No caso dos artefatos internos, eles são representados na forma de objetos, sendo salvos no banco de dados.

Por meio de GCS-ODE é possível gerenciar alterações, bem como os artefatos a serem alterados. O controle de alteração em GCS-ODE ocorre por meio de solicitações realizadas por desenvolvedores e avaliações de alteração realizadas por gerentes de configuração.

GCS-ODE possui, ainda, suporte a avaliação de impacto de alteração através de mapas de dependência entre artefatos de um projeto e da organização, usando para isso a infra-estrutural de gerência de conhecimento provida pelo ambiente ODE (CALHAU; ARANTES; FALBO, 2008). Além disso, oferece funcionalidades de controle de acesso, atribuindo acesso de escrita ou leitura para conjuntos de usuários do ambiente.

GCS-ODE apesar de ser focado no controle de alteração, também possui algumas atividades que apoiam o controle de versão, de modo bastante simplista. São registradas informações sobre as versões de um artefato sob gerência de configuração. Também existem funcionalidades para criação de ramificações, contendo um conjunto de versões, e para a definição de linhas base de um projeto. Tais funcionalidades são providas de modo simplista, possuindo o intuito apenas de armazenar algumas informações sobre versões, ramificações e

linhas base. Como a ferramenta foi construída no contexto do Projeto ODE, seus modelos são conhecidos e não foi preciso escavá-los.

Modelo Estrutural

A Figura 5.5 apresenta o modelo estrutural de GCS-ODE.

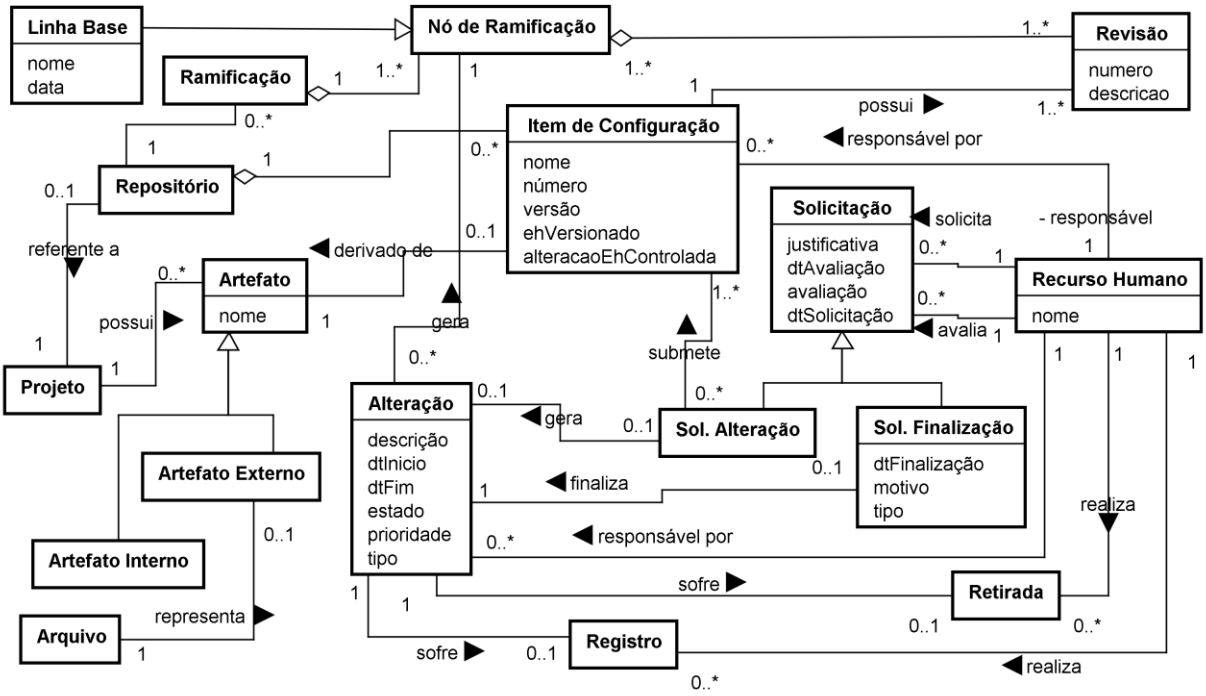

Figura 5.5 - Modelo conceitual de GCS-ODE (CALHAU, 2009)

Como está ilustrado na figura, um **Projeto** em ODE pode ser composto por um **Repositório** contendo os **Itens de Configuração** do projeto. No caso de GCS-ODE, os itens que são gerenciados são **Artefatos** produzidos e consumidos no projeto. Eles podem ser **Artefatos Internos**, produzidos por uma ferramenta interna, ou **Artefatos Externos**, produzidos externamente ao ambiente e representados na forma de **Arquivos**. Itens de configuração possuem um nível de controle, que pode ser versionado, possuir controle de alteração ou ambos. Um IC ainda possui um **Recurso Humano** que é responsável. ICs que são versionados possuem uma ou mais **Revisões** que representam informações específicas de um estado do IC. O repositório em GCS-ODE é composto de **Ramificações**, representando diferentes linhas de evolução. Cada ramificação é composta de estados chamados de **Nó de Ramificação**. Um nó de ramificação é composto de revisões e pode ser marcado como sendo uma linha base do projeto.

ICs podem ser submetidos a **Alterações**. Alterações são solicitadas por meio de **Solicitações de Alteração**. Uma solicitação é realizada por um recurso humano e avaliada por outro recurso humano responsável por essa tarefa. Em uma avaliação a alteração pode ser aprovada ou rejeitada. Para serem finalizadas, as alterações também necessitam de uma solicitação de finalização de alteração, realizada por um recurso humano que é um desenvolvedor.

Modelo Comportamental

GCS-ODE é um sistema que apoia grande parte das atividades de controle de configuração do processo apresentado na Subseção 5.2.1. A Figura 5.6 apresenta as principais funcionalidades de GCS-ODE para apoiar essas atividades.

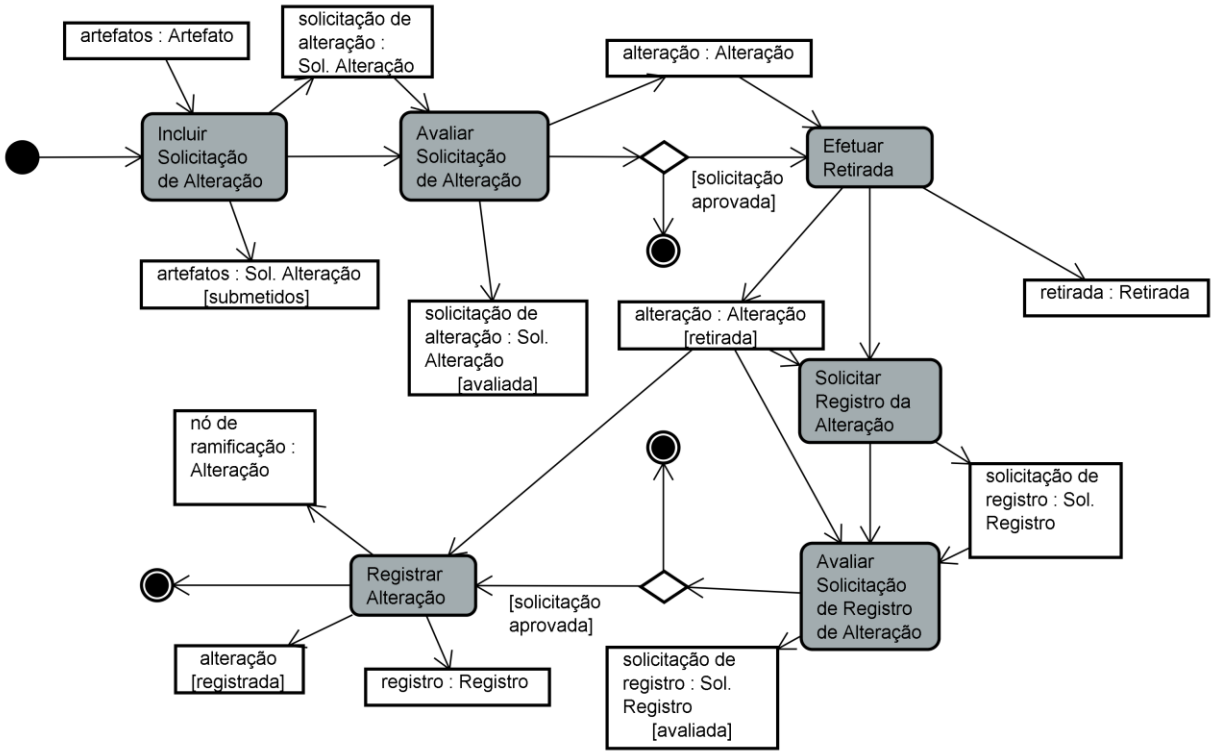

Figura 5.6 - Parte do modelo comportamental de GCS-ODE

Para tal estão envolvidas funcionalidades referentes à inclusão de solicitação de alteração, avaliação da solicitação de alteração, retirada de alteração, solicitação do registro da alteração, avaliação da solicitação de registro de alteração e registro de alteração. Inicialmente, artefatos são selecionados para serem submetidos a uma alteração durante a inclusão de uma solicitação de alteração. Uma solicitação de alteração é gerada e depois é avaliada na atividade de avaliação da solicitação. Caso seja aprovada, é criada uma alteração. Tal alteração é retirada na atividade seguinte. A alteração passa, então, a estar

retirada e depois que as modificações referentes a ela são terminadas, uma solicitação de registro de alteração deve ser realizada para que a alteração seja registrada. Caso a solicitação de registro de alteração seja aprovada, a alteração é registrada, gerando novas revisões dos ICs submetidos, um novo nó de ramificação e um registro *do checkin* são criados.

Além das funcionalidades citadas, existem outras funcionalidades de GCS-ODE que não foram representadas, pois não obedecem ao fluxo de controle apresentado. São elas: Criar Ramificação de Repositório, Marcar Nó de Ramificação, Colocar Artefato sob Gerência de Configuração e Criar Repositório do Projeto. Tais funcionalidades são de controle do repositório e ocorrem de modo independente da realização de alterações, ilustrada na figura. A qualquer momento, é possível criar ramificações para o projeto, marcar nós como linha base ou colocar artefatos sob GC.

5.3. Integrando *Subversion* e GCS-ODE

De acordo com OBA-SI, a integração semântica deve ocorrer primeiramente no nível conceitual por meio de mapeamentos semânticos entre os modelos conceituais e as ontologias de referência. O mapeamento para a ontologia tem como propósito explicitar o significado dos conceitos presentes nos sistemas e no processo de negócio a serem integrados, facilitando a integração semântica. A integração semântica primeiramente ocorre entre elementos estruturais (conceitos e relações), para então passar a considerar elementos comportamentais (tarefas, serviços, entradas e saídas).

5.3.1. Levantamento dos Requisitos da Integração

A primeira etapa para a integração foi o levantamento dos requisitos da integração. Tendo em mãos o processo de GCS do NEMO, levantaram-se as atividades a serem apoiadas. Com base nisso, definiu-se o cenário de integração, apresentado na Tabela 5.1. Nessa tabela são listados os sistemas a serem integrados e as atividades do processo de GCS a serem apoiadas, bem como os domínios e tarefas envolvidos e os requisitos da integração. Em função do escopo levantado, foi realizada a etapa de recuperação dos modelos conceituais, apresentados na seção 5.2. Em paralelo a isso, também com base no cenário de integração, é realizada a seleção das ontologias a serem usadas. Na presente iniciativa de integração foram usadas a ontologia de tarefa de Gerência de Configuração definida no Capítulo 4, e a ontologia de domínio de GCS proposta em (ARANTES; FALBO; GUIZZARDI, 2007), também apresentada no Capítulo 4.

Tabela 5.1 – Cenário de Integração entre Subversion e GCS-ODE

CENARIO DE INTEGRAÇAO	
SISTEMAS	*Subversion* e GCS-ODE
ATIVIDADES	*Identificar Itens de Configuração:* • Selecionar ICs; • Descrever ICs; • Definir Responsável por IC; *Controlar Configuração:* • Criar Baseline; • Efetuar Solicitação de Modificação; • Analisar Impacto da Modificação; • Avaliar Solicitações; • Atribuir Solicitação a Responsável; • Retirar Itens; • Implementar Modificação; • Registrar Itens; • Verificar Modificação em Relação à proposta de Implementação; • Atualizar *Baseline*.
DOMÍNIOS e TAREFAS ENVOLVIDAS	Domínio de Gerência de Configuração de Software, Tarefa de Gerência de Configuração
REQUISITOS	• Integrar funcionalidades do controle de alteração de GCS-ODE com as de controle de versão de SVN; • Realizar o controle de versão de artefatos de software de GCS-ODE que estão sob formato de arquivo por meio do SVN; • Controlar alterações em arquivos versionados em SVN; • Apoiar a avaliação de impacto de uma alteração; • Permitir alteração simultânea de artefatos dos artefatos de GCS-ODE; • Apoiar o registro de linhas base e dos estados dos respectivos artefatos de GCS-ODE; • Apoiar a gerência de ramificações bem como registro dos estados da ramificação. • Exigir que alterações dos artefatos de GCS-ODE ocorram localmente antes de serem registradas.

Depois da seleção das ontologias, foi feita a integração das mesmas, também discutida no capítulo anterior. Essa integração deu origem a uma ontologia de classe de aplicação (a classe de aplicações de apoio à Gerência de Configuração de Software), a qual foi usada como referência para a integração. Após ter em mãos a ontologia de classe de aplicação de referência e os modelos conceituais dos sistemas a serem integrados, passa-se à etapa de mapeamentos verticais.

5.3.2. Mapeamentos Verticais

Mapeamentos Verticais de Conceitos

Os mapeamentos verticais possuem o intuito de explicitar a semântica da conceituação dos sistemas em relação à semântica das ontologias de referência. Eles ocorrem entre os modelos conceituais dos sistemas e as ontologias selecionadas para a integração. Por meio de tais mapeamentos, é atribuída uma semântica aos conceitos dos sistemas usando para isso uma mesma referência, o que permite uma comparação entre as conceituações dos sistemas envolvidos na iniciativa de integração. A Tabela 5.2 apresenta os mapeamentos verticais dos conceitos de GCS-ODE e SVN com os conceitos da ontologia de classe de aplicação de referência. Cada linha da tabela é composta de um conceito da ontologia e o(s) conceito(s) dos sistemas relacionados. Entretanto, alguns conceitos dos sistemas não estão relacionados a nenhum conceito da ontologia.

Tabela 5.2 – Mapeamentos Verticais de Subversion e GCS-ODE

ONTOLOGIA	SUBVERSION	GCS-ODE
Pessoa *Agente físico responsável por desempenhar papéis de Solicitador, Gerente de Configuração (Avaliador e Verificador) e Desenvolvedor no processo de GCS.*	**Usuário** *Pessoa responsável pela realização de alterações no repositório, tais como adição, remoção e modificação de itens.*	**Recurso Humano** *Pessoa responsável por ICs, por alterações, por solicitar e avaliar solicitações.*
Item *Elemento genérico que pode ser selecionado para ter sua configuração gerenciada.*	**Item do Sistema de Arquivo** *Arquivos e diretórios de um sistema de arquivo.*	----------
Artefato *Entidades consumidas e produzidas em um projeto*	----------	**Artefato** *Entidades consumidas e produzidas em um projeto*
Item de Configuração *Item que foi selecionado para ser gerenciado*	**Item do Repositório** *Tipo de item do sistema de arquivo armazenado em um repositório cujas revisões são controladas.*	**Item de Configuração** *Artefato do projeto armazenado em um repositório cuja alteração é controlada.*
Item de Configuração Atômico *Item de Configuração que não é composto de nenhum outro.*	**Arquivo do Repositório** *Item do Sistema de Arquivo controlado que é do tipo Arquivo.*	----------
Item de Configuração Composto *IC composto de outros ICs.*	**Diretório do Repositório** *Item do Sistema de Arquivo controlado que é do tipo Diretório.*	----------

Tabela 5.1 – Mapeamentos Verticais de Subversion e GCS-ODE (Continuação)

ONTOLOGIA	SUBVERSION	GCS-ODE
Versão *Propriedade que representa o estado de um IC.*	**Revisão de Item** *Elemento que representa um estado específico de um item do repositório.*	**Revisão** *Estado específico de um artefato gerenciado.*
Linha Base *É o papel desempenhado por uma configuração marcada de um item composto, indicando que a mesma pode servir de referência para desenvolvimento posterior.*	**----------**	**Linha Base** *Estado de uma ramificação que foi marcado para servir de referência.*
Projeto *Esforço temporário de um conjunto de pessoas para o desenvolvimento de um produto*	**----------**	**Projeto** *Esforço temporário de um conjunto de pessoas para o desenvolvimento de um produto de software*
Repositório *Conjunto de ramificações que organizam as versões dos ICs de um projeto.*	**Repositório** *Entidade que representa o local de armazenamento de itens do sistema de arquivo que estão controlados.*	**Repositório** *Entidade que representa o local de armazenarmento dos ICs de um projeto.*
Ramificação *Conjunto de versões de um repositório que está em uma mesma linha de evolução.*	**----------**	**Ramificação** *Conjunto de ICs que evoluem conjuntamente.*
Seleção da Configuração *Registro do ato de selecionar itens para serem gerenciados, transformando-os em ICs.*	**Import** *Registro do ato de colocar itens não versionados no repositório para serem controlados.*	**----------**
Solicitação de Alteração *Requisição de alteração feita por um solicitador para alteração em um conjunto de versões.*	**----------**	**Solicitação de Alteração** *Requisição realizada por um recurso humano para realização de uma alteração em um conjunto de ICs, com informações relativas à sua avaliação associadas.*
Avaliação *Registro da ação realizada por um avaliador de avaliar uma solicitação de alteração*	**----------**	**Solicitação de Alteração** *Requisição realizada por um recurso humano para realização de uma alteração em um conjunto de ICs, com informações relativas à sua avaliação associadas.*
Alteração *Especificação de uma modificação a ser realizada em ICs que pode ou não ser implementada.*		**Alteração** *Registro das informações relativas a uma alteração feita em um conjunto de ICs.*

Tabela 5.1 – Mapeamentos Verticais de Subversion e GCS-ODE (Continuação)

Alteração Implementada *Alteração especificada que foi implementada e registrada por meio de um checkin.*	**Alteração do Repositório** *Registro da alteração de um repositório contendo um conjunto de modificações registradas em itens do repositório.*	**----------**
Checkout *Registro do retiro de um conjunto de versões por um desenvolvedor para realização de uma alteração, a qual é iniciada.*	**Checkout** *Registro da ação feita por um usuário de gerar cópias locais de itens do repositório para realizar alterações nos mesmo.*	**Retirada** *Registro da retirada de ICs para alteração feita por um recurso humano*
Cópia Versão gerada a partir de uma versão retirada no momento do checkout, a qual vai ser objeto efetivamente de alteração.	**Item Cópia de Trabalho** *Item do sistema de arquivo gerado a partir de um checkout de um item do repositório e que representa a sua cópia.*	**----------**
Modificação *Registro da ação de modificação de uma cópia de uma versão retirada*	**Modificação Local** *Registro de alguma modificação sofrida por um item da Cópia de Trabalho.*	**----------**
Checkin *Registro do registro (no sentido de checkin) de uma alteração em um conjunto de ICs.*	**Commit** *Registro da ação feita por um usuário de registrar as alterações feitas em um item da cópia de trabalho no repositório*	**Registro** *Registro da devolução de ICs retirados feita por um recurso humano*
Modificação Registrada *Registro da ação de modificação de uma versão retirada e que foi registrada por meio de um checkin.*	**Modificação** *Registro de alguma modificação sofrida por um item do repositório*	**----------**
Revisão Versão que substitui outra na linha de evolução de um IC	**Revisão** Estado do diretório raiz do repositório SVN e que representa o estado de todo repositório	**----------**

Através dos mapeamentos verticais, obtêm-se relações mais gerais entre os conceitos dos sistemas, uma vez que a ontologia é geral e não considera particularidades da integração. Conforme apresentado na Tabela 5.2, *Recurso Humano* (GCS-ODE) e *Usuário* (SVN) foram mapeados para *Pessoa* (Ontologia). Ambos os conceitos do sistema representam uma pessoa que desempenha os papéis envolvidos nas tarefas de gerência de configuração. Em GCS-ODE, um recurso humano é responsável por realizar as tarefas de solicitação, avaliação,

seleção da configuração e implementação. No SVN, por sua vez, um usuário é responsável por fazer modificações em itens do repositório e por adicionar arquivos e diretórios no repositório.

Item do Sistema de Arquivo (SVN) foi mapeado para *Item* (Ontologia), pois representa o que pode ser gerenciado no contexto do SVN. *Artefato* (GCS-ODE), por sua vez, foi mapeado para Artefato (Ontologia), pois ambos representam entidades produzidas e usadas no contexto de um projeto. *Item do Repositório* (SVN) e *Item de Configuração* (GCS-ODE) foram mapeados para o conceito *Item de Configuração* (Ontologia), uma vez que representam itens cuja configuração está sendo gerenciada.

Arquivo do Repositório (SVN) foi mapeado para *IC Atômico* (Ontologia), já que um arquivo do repositório não é composto de nenhum outro item do repositório. Já *Diretório do Repositório* (SVN) foi mapeado para um *IC Composto* (Ontologia), uma vez que representa um item de configuração composto de outros (Itens de Repositório, no caso).

Os conceitos *Revisão de Item* (SVN) e *Revisão* (GCS-ODE) foram mapeados para *Versão* (Ontologia), já que ambos representam estados de itens de configuração. *Linha Base* (GCS-ODE), por sua vez, foi mapeada para *Linha Base* (Ontologia). Apesar da equivalência entre esses conceitos, algumas distinções conceituais não foram totalmente representadas na ontologia. *Linha Base* (GCS-ODE) é a marcação de um estado de uma ramificação e *Linha Base* (Ontologia) é a marcação de um estado de um IC composto e a ontologia não fala a respeito de uma relação entre *IC Composto* e *Ramificação*. Já o conceito *Projeto* (GCS-ODE) foi mapeado para *Projeto* (Ontologia).

Repositório (SVN) e *Repositório* (GCS-ODE) foram mapeados para *Repositório* (Ontologia). Apesar de *Repositório* (SVN) ser composto de *Arquivos* e *Diretórios*, e de *Repositório* (GCS-ODE) ser composto de *Artefatos*, eles são conceitualmente equivalentes. O conceito *Ramificação* (GCS-ODE), por sua vez, foi mapeado para *Ramificação* (Ontologia), já que ambos representam uma linha de evolução distinta de um conjunto de ICs.

O conceito *Import* (SVN) foi mapeado para *Seleção da Configuração* (Ontologia), uma vez que representa o registra da ação de selecionar itens não gerenciados para se tornarem itens de configuração. O conceito *Solicitação de Alteração* (GCS-ODE) foi mapeado para *Solicitação de Alteração* (Ontologia), já que ambos formalizam uma intenção de realizar uma alteração em um conjunto de itens de configuração. O mesmo conceito *Solicitação de Alteração* (GCS-ODE) também foi mapeado para *Avaliação* (Ontologia), pois uma solicitação de alteração em GCS-ODE também armazena informações de uma avaliação.

O conceito *Alteração* (GCS-ODE) foi mapeado para *Alteração* (Ontologia), uma vez que representa informações referentes ao que deve ser realizado em uma alteração de itens. Já *Alteração do Repositório* (SVN) foi mapeado para *Alteração Implementada* (Ontologia), uma vez que representa alterações em itens de configuração que foram registradas.

Checkout (SVN) e *Retirada* (GCS-ODE) foram mapeados para *Checkout* (Ontologia), uma vez que representam a retirada de itens de configuração para alteração. *Item da Cópia de Trabalho* (SVN) foi mapeado para *Cópia* (SVN), uma vez que representa uma cópia de uma versão de IC retirada para alteração. *Alteração Local* (SVN) foi mapeado para Modificação (Ontologia), uma vez que representa modificações não registradas de uma versão retirada. Um detalhe importante é que o conceito de alteração local em SVN também representa adição ou remoção de itens da cópia de trabalho, e não apenas modificação dos itens da Cópia de Trabalho. Vale ressaltar que o significado atribuído pela ontologia nem sempre está totalmente de acordo, como nesse caso, no qual o conceito no sistema engloba também outros significados.

O conceito *Commit* (SVN) e o conceito *Registro* (GCS-ODE), por sua vez, foram mapeados para o conceito *Checkin* (Ontologia), já que representam o registro da alteração. Por fim, o conceito *Alteração* (SVN) foi mapeado para *Modificação Registrada* (Ontologia), pois representa uma alteração realizada em uma versão retirada e que foi registrada. Da mesma forma que ocorre com o conceito *Alteração Local*, que envolve modificações, adições e remoções, esse também envolve essas ações.

Por meio dos mapeamentos verticais foi possível explicitar conceituações relativas ao universo de discurso da gerência de configuração que estão presentes nos sistemas de forma implícita. Por exemplo, em SVN foi possível explicitar que *checkouts* e *commits* ocorrem no contexto de uma alteração, que itens da cópia de trabalho representam cópias de versões retiradas para uma alteração que já foi iniciada.

Tabela 5.3 – Resumos dos Mapeamentos Verticais Estruturais

ONTOLOGIA	SUBVERSION	GCS-ODE
Pessoa	Usuário	Recurso Humano
Item	Item do Sistema de Arquivo	----------
Artefato	----------	Artefato
Item de Configuração	Item do Repositório	Item de Configuração
Item de Configuração Atômico	Arquivo do Repositório	----------
Item de Configuração Composto	Diretório do Repositório	----------
Versão	Revisão de Item	Revisão
Revisão	Revisão	----------
Linha Base	----------	Linha Base
Projeto	----------	Projeto
Repositório	Repositório	Repositório
Ramificação	----------	Ramificação
Seleção da Configuração	Import	----------
Solicitação de Alteração	----------	Solicitação de Alteração
Avaliação	----------	Solicitação de Alteração
Alteração		Alteração
Alteração Implementada	Alteração do Repositório	----------
Checkout	Checkout	Retirada
Cópia	Item Cópia de Trabalho	----------
Modificação	Modificação Local	----------
Checkin	Commit	Registro
Modificação Registrada	Modificação	----------

Com relação ao GCS-ODE, os mapeamentos verticais também explicitaram conceituações da tarefa de GC que estavam implícitas, como por exemplo, os papéis de desenvolvedor, avaliador, solicitador e responsável pela configuração que são realizados por recursos humanos, além de explicitar, por exemplo, que o conceito *Linha Base* representa um papel desempenhado por um estado da ramificação mediante uma marcação.

Apesar de a ontologia estar bem fundamentada e ter coberto a maior parte das conceituações dos sistemas, ela não cobre todos os conceitos dos sistemas, como já era de se esperar. No caso da iniciativa de integração em questão, vários elementos de modelo dos sistemas não tiveram correspondência com conceitos da ontologia, a saber: *Artefato Interno* e *Artefato Externo* (GCS-ODE), *Arquivo* (SVN e GCS-ODE), *Diretório* (SVN), *Revisão* (SVN), *Item Copiado* (SVN) e *Nó de Ramificação* (GCS-ODE).

Em especial, vale o registro de que há uma correspondência bastante próxima entre conceitos de GCS-ODE e a ontologia de referência. Isso decorre do fato de uma primeira versão de GCS-ODE ter sido desenvolvida usando como base a conceituação provida por

uma primeira versão da ontologia de domínio de Gerência de Configuração de Software em (NUNES, 2005).

Mapeamentos Verticais de Relacionamentos

Depois de realizado os mapeamentos verticais dos conceitos dos sistemas com a ontologia, foram realizados os mapeamentos verticais das relações entre conceitos. Tais mapeamentos foram responsáveis por explicitar nos modelos conceituais dos sistemas informações que antes não estavam representadas. Elas se referem principalmente a papéis realizados pelas pessoas nos sistemas, como também a informações referentes às ações realizadas por eles. A Tabela 5.4 apresenta alguns dos mapeamentos verticais entre relacionamentos do SVN e relações da ontologia de classe de aplicação. Os relacionamentos destacados com um asterisco representam relacionamentos derivados no modelo.

Tabela 5.4 – Mapeamentos Verticais de Relacionamentos - Subversion

ONTOLOGIA			SUBVERSION		
CONCEITO	RELAÇÃO	CONCEITO	CONCEITO	RELAÇÃO	CONCEITO
Pessoa (Gerente de Configuração)	**realiza**	Seleção da Configuração	Usuário	**realiza**	Import
Pessoa (Desenvolvedor)	**realiza**	Checkout	Usuário	**realiza**	Checkout
Pessoa (Desenvolvedor)	**realiza**	Modificação	Usuário	**responsável por***	Alteração Local
Pessoa (Desenvolvedor)	**realiza**	Checkin	Usuário	**Realiza**	Commit

Os mapeamentos foram feitos considerando, sobretudo, o mapeamento dos conceitos que estão sendo por eles relacionados (ver Tabela 5.3). Vale destacar apenas que Gerente de Configuração e Desenvolvedor são papéis desempenhados por Pessoas na ontologia e Usuário (SVN) corresponde a Pessoa (Ontologia) e, por isso, esses conceitos foram considerados também equivalentes.

O relacionamento “Usuário **é responsável por** Alteração Local” é derivado de outras associações no modelo do SVN. Lá, Usuário é responsável por Cópia de Trabalho, a qual é composta de Itens da Cópia de Trabalho. Uma Alteração Local é sempre referente a um Item da Cópia de Trabalho e, portanto, é possível derivar o relacionamento Usuário é responsável por Alteração Local.

Por meio dos mapeamentos verticais dos relacionamentos de SVN, foi possível explicitar, por exemplo, os papéis desempenhados pelas pessoas no SVN (*Usuário* em SVN). Um *Usuário* (SVN) desempenha o papel de *Desenvolvedor* ao realizar as ações de *Commit*, *Checkout* e *Alteração Local*, enquanto desempenha o papel de *Gerente da Configuração* ao selecionar a configuração.

A Tabela 5.5 apresenta os demais mapeamentos de relacionamentos realizados, os quais foram obtidos de maneira análoga aos da Tabela 5.4 e não são comentados em maiores detalhes.

Tabela 5.5 – Mapeamentos Verticais de Relacionamentos – Subversion (Continuação)

ONTOLOGIA			SUBVERSION		
CONCEITO	RELAÇÃO	CONCEITO	CONCEITO	RELAÇÃO	CONCEITO
Item de Configuração	**possui**	Versão	Item do Repositório	**composto de**	Revisão de Item
Item de Configuração	**selecionado por**	Seleção de Configuração	Item do Repositório	**adicionado por**	Import
Versão	**gerada em*** (gerada no *checkin* referente à)	Alteração Implementada	Revisão de Item	**gerada em*** (uma *revisão* que é gerada por uma)	Alteração de Repositório
Versão	**gerada por*** (checkin referente à)	Modificação Registrada	Revisão de Item	**sofre**	Alteração
Versão (Retirada)	**sofre**	Checkout	Revisão de Item	**sofre**	Checkout
Checkout	**gera**	Cópia	Checkout	**gera*** (*cópia de trabalho* que é composta de)	Item da CT
Cópia	**referente a**	Versão (Retirada)	Item da CT	**referencia**	Revisão de Item
Cópia (Modificada)	**sofre**	Modificação	Item da CT	**sofre**	Alteração Local (Modificação Local)
Checkin	**registra**	Alteração Implementada	Commit	**promove**	Alteração do Repositório
Checkin	**referente a*** (*modificação registrada* referente a)	Cópia (Modificada)	Commit	**referente a**	Item da CT
Alteração Implementada	**relacionada a*** (*checkin* referente a)	Modificação (Registrada)	Alteração do Repositório	**realiza**	Alteração (Modificação)

A Tabela 5.6 apresenta os mapeamentos verticais de relacionamento de GCS-ODE.

Tabela 5.6– Mapeamentos Verticais de Relacionamento de GCS-ODE

ONTOLOGIA			GCS-ODE		
CONCEITO	RELAÇÃO	CONCEITO	CONCEITO	RELAÇÃO	CONCEITO
Pessoa (Solicitador)	**realiza**	Solicitação de Alteração	Recurso Humano	**solicita**	Solicitação (de Alteração)
Pessoa (Avaliador)	**avalia**	Solicitação	Recurso Humano	**avalia**	Solicitação (de Alteração)
Pessoa (Responsável por Configuração)	**responsável por**	Item de Configuração	Recurso Humano	**responsável por**	Item de Configuração
Pessoa (Desenvolvedor)	**realiza**	Checkout	Recurso Humano	**realiza**	Retirada
Pessoa (Desenvolvedor)	**realiza**	Checkin	Recurso Humano	**realiza**	Registro
Item de Configuração	**possui**	Versão	Item de Configuração	**possui**	Revisão
Item de Configuração	**representa papel desempenhado por**	Artefato	Item de Configuração	**derivado de**	Artefato
Alteração (Implementada)	**gera*** (sofre checkin, que registra)	Versão	Alteração	**gera*** (nó de ramificação que é composto de)	Versão
Alteração (Iniciada)	**sofre**	Checkout	Alteração	**sobre**	Retirada
Alteração (Implementada)	**sofre**	Checkin	Alteração	**sobre**	Registro
Solicitação de Alteração	**submete**	Versão (Submetida a Alteração)	Sol.Alteração	**submete*** (item de configuração composto de)	Revisão
Solicitação de Alteração	**referente a**	Alteração	Sol.Alteração	**gera**	Alteração
Repositório	**referente a**	Projeto	Repositório	**referente a**	Projeto
Repositório	**composto de*** (ramificação composta de versão de)	Item de Configuração	Repositório	**composto de**	Item de Configuração
Repositório	**composto de**	Ramificação	Repositório	**possui**	Ramificação

Dentre os relacionamentos apresentados na Tabela 5.6, vale a pena comentar alguns deles. O relacionamento "Recurso Humano **é responsável por** Item de Configuração" (GCS-ODE) foi mapeado para o relacionamento "Gerente de Configuração (Responsável por Configuração) **é responsável por** Item de Configuração" (Ontologia), o qual não está mostrado na ontologia de classe de aplicação, uma vez que é um relacionamento material derivado do *relator Seleção da Configuração*. Situação análoga ocorre com o relacionamento "Recurso Humano **avalia** Solicitação" (GCS-ODE), que foi mapeado para o relacionamento "Avaliador **avalia** Solicitação" (Ontologia), o qual não está mostrado na ontologia de classe de aplicação, uma vez que ele é um relacionamento material derivado do *relator Avaliação*. Esse mapeamento, em especial, é responsável por explicitar, ainda, que nesse relacionamento de GCS-ODE existem informações referentes à avaliação. No caso do modelo de GCS-ODE tais informações estão na classe solicitação (atributos *avaliacao*, referente a uma descrição da mesma, e *dtAvaliacao*, referente à data de avaliação).

Os mapeamentos verticais de relacionamentos de GCS-ODE também explicitaram os papéis desempenhados por uma pessoa em GCS-ODE (*Recurso Humano)*. Uma pessoa em GCS-ODE desempenha o papel de *Solicitador* quando solicita uma alteração, de *Avaliador* quando realiza uma avaliação de uma solicitação, de *Responsável por Configuração* quando é responsável por um IC, e de *Desenvolvedor*, quando realiza uma *Retirada* ou um *Registro*.

Os mapeamentos também explicitaram que uma alteração em GCS-ODE que sofre retirada desempenha o papel de *Alteração Iniciada*, enquanto uma que sofre um registro, gerando versões por meio de um nó de ramificação, desempenha o papel de *Alteração Implementada*. Por fim, as revisões selecionadas para uma solicitação de alteração desempenham o papel de *Versão Submetida a Alteração*.

Os mapeamentos de relacionamentos explicitaram principalmente informações referentes a papéis desempenhados por certos conceitos. Foram também explicitadas informações sobre ações que não estavam representadas nos modelo conceituais dos sistemas, como por exemplo, as ações de avaliação e seleção da configuração, que são conceitos presentes no modelo de GCS-ODE, mas que não estão explícitos.

Entretanto, da mesma forma que os mapeamentos verticais de conceitos, nem todos os relacionamentos envolvidos na integração têm correspondentes na ontologia, sobretudo os relativos a conceitos que também não foram cobertos. Esses relacionamentos, quando parte do cenário de integração, têm de ser tratados no modelo de integração, usando mapeamentos horizontais. Um exemplo de relacionamento não coberto é o relacionamento “*Recurso*

Humano ***é responsável por*** *Alteração*", presente no modelo conceitual de GCS-ODE, e que deverá ser coberto pelo modelo de integração. Outros relacionamentos não cobertos pelos mapeamentos verticais são: "Projeto **possui** Artefato" (GCS-ODE), "Ramificação **é composta de** Nó de Ramificação" (GCS-ODE) e "Nó de ramificação **composto de** Revisão" (GCS-ODE).

Mapeamentos Verticais Comportamentais

Depois dos mapeamentos verticais estruturais (entre conceitos e relacionamentos), são feitos os mapeamentos verticais comportamentais que envolvem os mapeamentos entre tarefas (serviços), entre insumos/produtos e os respectivos conceitos, e por fim os mapeamentos de insumos entre si e produtos entre si. As Tabela 5.7 a Tabela 5.14 apresentam os mapeamentos verticais comportamentais realizados. Elas mostram os mapeamentos verticais envolvendo os serviços dos sistemas, atividades do processo e tarefas da ontologia. Para cada mapeamento de tarefa, foram realizados os mapeamentos dos respectivos insumos e produtos. Para facilitar a visualização, os mapeamentos de serviços/tarefas foram representados de cinza escuro, os mapeamentos de insumos em cinza claro e os de produtos em branco. Os mapeamentos entre insumos/produtos aos respectivos conceitos são apresentados posteriormente, na descrição dos mapeamentos realizados.

Como mostra a Tabela 5.7, a atividade *Efetuar Solicitação de Alteração* (Processo) e o serviço *Incluir Solicitação de Alteração* (GCS-ODE) foram mapeados para a tarefa *Solicitar Alteração* (Ontologia), uma vez que representam a realização de solicitações de alteração. Com relação ao mapeamento de insumos, os insumos *set of versões: Versão* (Ontologia) e *artefato: Artefato* (GCS-ODE) foram mapeados entre si, uma vez que representam o que se deseja alterar.

Tabela 5.7 – Mapeamentos Verticais Comportamentais: Solicitar Alteração

ONTOLOGIA	PROCESSO	SVN	GCS-ODE
Solicitar Alteração	**Efetuar Solicitação de Alteração**	**---**	**Incluir Solicitação de Alteração**
set of versões: Versão	---	---	artefatos: Artefato
set of versões: Versão Submetida a Alteração	---	---	artefatos: Artefato [submetido]
alt: Alteração	modificação	---	**---**
sol: Solicitação de Alteração	solicitação	---	solicitação de alteração: Solicitação Alteração

Com relação aos mapeamentos entre produtos, *set of versões: Versão Submetida a Alteração* (Ontologia) e *artefato: Artefato [submetido]* (GCS-ODE) foram mapeados, uma vez que representam o que foi submetido à alteração. Os produtos *alt: Alteração* (Ontologia) e *modificação* (Processo) foram mapeados, pois representam a descrição do que foi solicitado. Por fim, o insumo *solicitação* (Processo) e *solicitação de alteração: Solicitação de Alteração* (GCS-ODE) foram mapeados para *sol: Solicitação de Alteração (Ontologia)*, uma vez que representam a solicitação de alteração em si.

A Tabela 5.8 mostra o mapeamento entre a atividade *Avaliar Solicitação* (Processo) e o serviço *Avaliar Solicitação de Alteração* (GCS-ODE) que são mapeados com a tarefa *Avaliar Solicitação* (Ontologia).

Tabela 5.8- Mapeamentos Verticais Comportamentais: Avaliar Solicitação

ONTOLOGIA	PROCESSO	SVN	GCS-ODE
Avaliar Solicitação	**Avaliar Solicitação**	**---**	**Avaliar Solicitação de Alteração**
alt: Alteração	---	---	**---**
sol: Solicitação de Alteração	solicitação [classificada]	---	solicitação de alteração : Sol. Alteração
av: Avaliação	avaliação	---	avaliação: Avaliação
sol: Solicitação Avaliada	solicitação [avaliada]	---	solicitação de alteração : Sol. alteração [avaliada]
---		---	alteração

Com relação aos mapeamentos de insumos, o insumo *solicitação de alteração*: Sol. Alteração (GCS-ODE) e o insumo *solicitação* (Processo) foram mapeados para o insumo *sol:* Solicitação (Ontologia), uma vez que representam o que será avaliado.

Com relação aos mapeamentos entre produtos, os produtos *avaliação* (Processo) e *avaliação:* Avaliação (GCS-ODE) foram mapeados para *av*: Avaliação (Ontologia), uma vez que representam o registro da ação de avaliação, enquanto o produto *solicitação de alteração:* Sol. Alteração [avaliada] (GCS-ODE) foi mapeado para *sol* : Solicitação Avaliada (Ontologia), uma vez que representa a mudança de estado da solicitação.

A Tabela 5.9 mostra o mapeamento das tarefas relacionadas à de retirada de itens para alteração. *Retirar ICs* (Processo), *Checkout* (SVN) e *Retirar Alteração* (GCS-ODE) foram mapeados para a tarefa *Realizar Checkout* (Ontologia), uma vez que desempenham essa

função. Com relação aos mapeamentos de insumos, *itens de configuração* (Processo), *dir: Diretório do Repositório* (SVN) foram mapeados para o insumo set of *versões:* Versão (Ontologia), uma vez que representam o que será retirado para alteração. Já o insumo *alteração:* Alteração (GCS-ODE) foi mapeado para o insumo *alt:* Alteração (Ontologia), uma vez que representa a razão da retirada.

Tabela 5.9 - Mapeamentos Verticais Comportamentais: Realizar Checkout

ONTOLOGIA	PROCESSO	SVN	GCS-ODE
Realizar *Checkout*	**Retirar ICs**	**Checkout**	**Retirar Alteração**
alt: Alteração	---	---	alteração : Alteração
set of versões: Versão	itens de configuração	dir: Diretório do Repositório	---
co: Checkout	---	checkout : Checkout	retirada: Retirada
alt: Alteração Iniciada	---	---	alteração : Alteração [retirada]
Set of copias: Copia		set of itens da ct: Item da Cópia de Trabalho	---
set of versões: Versão Retirada	itens de configuração [retirados]		
---	---	ct: Cópia de Trabalho	---

Com relação aos mapeamentos de produtos, os produtos *checkout*: Checkout (SVN) e *retirada:* Retirada (GCS-ODE) foram mapeados para o produto *co:* Checkout (Ontologia), uma vez que representam um registro da ação realizada. O produto *itens de configuração [retirados]* (Processo) foi mapeados para o produto *set of versões*: Versão Retirada (Ontologia), uma vez que representam o que foi retirado. O produto *set itens da ct: Item da Cópia de Trabalho (SVN)* foi mapeado para *set of copias: Copia* (Ontologia), uma vez que representa uma cópia do que foi retirado para alteração. Por fim, o produto *alteração: Alteração* [*retirada]* (GCS-ODE) foi mapeado para *alt:* Alteração Iniciada (Ontologia), uma vez que representa a alteração do estado de uma alteração para iniciada.

A Tabela 5.10 mostra os mapeamentos da atividade *Implementar Modificação* (Processo) e do serviço *Realizar Alterações locais* (SVN), os quais foram mapeados para a tarefa *Modificar Versões* (ontologia), uma vez que representam a realização da alteração propriamente dita, nos itens retirados para ela.

Com relação aos mapeamentos dos insumos relacionados, os insumos *itens de configuração [retirado]* (Processo) e *set of itens da ct: Item da Cópia de Trabalho* (SVN) foram mapeados para *set of copias: Copia* (Ontologia), uma vez que representam copias do que foi retirado para ser alterado. O insumo *proposta de implementação* (Processo) foi mapeado para o insumo *alt:* Alteração Iniciada (Ontologia), uma vez que representa como deverá ser realizada a alteração.

Com relação aos mapeamentos de produto, o produto *modificação [implementada]* (Processo) e o produto *alterações locais:* Alteração Local (SVN) foram mapeados para *set of mod* (Ontologia), uma vez que representam as modificações realizadas nos itens. Já os produtos *itens de configuração [modificados]* (Processo) e set of *itens da ct:* Item da Cópia de Trabalho [com alterações locais] (SVN) foram mapeados para *set of copias:* Cópia Modificada (Ontologia), uma vez que representam o que foi modificado.

Tabela 5.10 - Mapeamentos Verticais Comportamentais: Modificar Versões

ONTOLOGIA	PROCESSO	SVN	GCS-ODE
Modificar Versões	**Implementar Modificação**	**Realizar Alterações Locais**	
set of copias: Copia	itens de configuração [retirado]	set of itens da ct: Item da Cópia de Trabalho	
alt: Alteração Iniciada	proposta de implementação		
---	modificação [avaliada]		
set de mods: Modificação	modificação [implementada]	alterações locais: Alteração Local	
set of copias: Copia Modificada	itens de configuração [modificados]	set of itens da ct: Item da Cópia de Trabalho [com alterações locais]	

A Tabela 5.11 mostra os mapeamentos entre a atividade *Registrar Modificação* (Processo), o serviço *Commit* (SVN) e o serviço *Registrar Alteração* (GCS-ODE), que foram mapeados para a tarefa *Realizar Checkin* (Ontologia), uma vez que representam o registro das alterações realizadas nos itens retirados.

Com relação aos mapeamentos dos respectivos insumos, o insumo *itens de configuração [modificado]* (Processo), o insumo *set of itens da ct: Item da Cópia de Trabalho* [atualizada] (SVN) e o insumo *alteração: Alteração [retirada]* (GCS-ODE) foram mapeados

para o insumo *set of copias : Copia Modificada* (Ontologia), uma vez que representam o que foi modificado e será registrado.

Com relação aos mapeamentos de produto, o produto *ci*: Commit (SVN) e o produto *registro:* Registro (GCS-ODE) foram mapeados para o produto *ci:* Checkin (Ontologia), uma vez que representam o registro da ação realizada. O produto *novas versões [registrada]* (Processo), o produto *set of ver. item: Revisão de Item* (SVN) e o produto *nó de ramificação: Nó de Ramificação* (GCS-ODE) foram mapeados para *versões geradas* (Versão - Ontologia), uma vez que representam novos estados dos itens modificados a partir do registro dos mesmos. Por fim, o produto *set of alt: Alteração* (SVN) e o produto *alteração: Alteração [registrada]* (GCS-ODE) foram mapeados para o produto *set of mods: Modificação Registrada* (Ontologia), uma vez que representam as modificações que foram realizadas nos itens e que foram registradas.

Tabela 5.11 - Mapeamentos Verticais Comportamentais: Realizar Checkin

ONTOLOGIA	PROCESSO	SVN	GCS-ODE
Realizar *Checkin*	**Registrar Modificação**	**Commit**	**Registrar Alteração**
set of mods: Modificação	modificação [implementada]	---	---
set of cópias : Cópia Modificada	itens de configuração [retirados]	set of itens da ct: Item da Cópia de Trabalho [atualizada]	alteração: Alteração [retirada]
ci: Checkin	-----	ci: Commit	registro: Registro
set of novas: Versão	novas versões [registrada]	set of ver. item: Revisão de Item	nó de ramificação: Nó de Ramificação
set of mods: Modificação Registrada	itens de configuração [registrados]	set of alt : Alteração	alteração: Alteração [retirada]
---		set of novos ics : Item do Repositório	---
---		alt rep.: Alteração do Repositório	---
---	---	---	

A Tabela 5.12 mostra os mapeamentos da atividade *Verificar Modificação* (Processo), a qual foi mapeada para a tarefa *Verificar Implementação* (Ontologia). O insumo *proposta de implementação* (Processo) foi mapeado para o insumo *alt*: *Alteração Implementada*

(Ontologia), uma vez que representa o que será tomado como base na realização da verificação com relação ao que deveria ter sido realizado. Por fim, o insumo *novas versões [registrada]* (Processo) foi mapeado para *set of novas versões: Versão* (Ontologia), uma vez que representa o resultado da implementação realizada que será verificado. Com relação ao mapeamento de produtos, *verificação* (Processo) foi mapeado para *verificação:* Verificação (Ontologia), pois representam o registro da ação realizada.

Tabela 5.12 - Mapeamentos Verticais Comportamentais: Verificar Implementação

ONTOLOGIA	PROCESSO	SVN	GCS-ODE
Verificar Implementação	**Verificar Modificação**	**---**	**---**
Alt: Alteração Implementada	proposta de implementação	---	---
set of novas: Versão	novas versões [registrada]	---	---
verificação: Verificação	verificação	---	---
alt: Alteração Verificada. '.	---	---	---

A Tabela 5.13 mostra os mapeamentos entre *Identificar IC* (Processo), *Importar Diretório Não Versionado* (SVN) e *Colocar Artefato sob Gerência de Configuração* (GCS-ODE), que foram mapeados para *Selecionar ICs* (Ontologia), uma vez que representam a seleção de itens para se tornarem itens de configuração.

Tabela 5.13 - Mapeamentos Verticais Comportamentais: Selecionar ICs

ONTOLOGIA	PROCESSO	SVN	GCS-ODE
Selecionar ICs	**Identificar IC**	**Importar Diretório Não Versionado**	**Colocar Artefato sob Gerência de Configuração**
set of itens : Item	set of artefatos	dir. não ver.: Diretório Não Versionado	artefatos: Artefato
sel: Seleção da Configuração	---	Import: Import	---
set of ics: Item de Configuração	itens de configuração	novos itens do repositório: Item do Repositório	set of ics: Item de Configuração
---	plano de gerência de configuração [modificado]	---	---
---	---	alteração do repositório:	---

		Alteração do repositório	

Nessa tarefa, os insumos *set of artefatos* (Processo), *dir não ver.: Diretório Não Versionado* (SVN) e *artefatos*: *Artefato* (GCS-ODE) foram mapeados para o insumo *set of itens*: *Item* (Ontologia), uma vez que representam os itens que serão colocados sob gerência de configuração. Os produtos *import*: *Import* (SVN) e *sel: Seleção da Configuração* (Ontologia) foram mapeados, uma vez que representam o registro da ação realizada. Já os produtos *itens de configuração* (Processo), *novos itens do repositório:* Item do Repositório (SVN) e *set of ics*: *Item de Configuração* (GCS-ODE) foram mapeados para *set of ics: Item de Configuração* (Ontologia), uma vez que representam os ICs que foram selecionados para serem controlados.

Por fim, a atividade *Criar Linha base* (Processo) e o serviço *Marcar Nó de Ramificação como Linha Base* (GCS-ODE) foram mapeadas para *Marcar Configuração como Linha Base* (Ontologia), como mostra a Tabela 5.14.

Tabela 5.14 - Mapeamentos Verticais Comportamentais: Marcar Configuração como Linha Base

ONTOLOGIA	PROCESSO	SVN	GCS-ODE
Marcar Configuração como Linha Base	**Criar Linha Base**	---	**Marcar Nó de Ramificação como Linha Base**
conf: Configuração	versões [registradas]	---	nó de ramificação: Nó de Ramificação
marc: Marcação		---	---
lb: Linha Base	linha base	---	linha base: Linha Base

Os insumos *versões registradas* (Processo) e *nó de ramificação*: *Nó de Ramificação* (GCS-ODE) foram mapeados para o insumo *configuração: Configuração* (Ontologia), pois representam o que será marcado como linha base. Os produtos *linha base* (Processo) e *linha base*: *Linha Base* (GCS-ODE) foram mapeados para o produto *linha base*: *Linha Base* (Ontologia), pois representam a linha base gerada.

Através dos mapeamentos verticais de comportamento foi possível explicitar diversas informações sobre o comportamento dos sistemas e do processo de negócio que antes não estavam explícitas. Por exemplo, através deles, foi possível ter um melhor entendimento da

semântica dos serviços do SVN. *Checkouts* realizados no SVN são responsáveis por retirar versões específicas dos itens de configuração (localizados no diretório do repositório informado) para a realização de uma alteração iniciada. Isso se dá, especificamente no caso do SVN, por meio de cópias (itens da cópia de trabalho), que representam as versões retiradas dos ICs. A realização de alterações locais no SVN é uma maneira de se realizar modificações em versões de ICs retiradas (por meio de seus representantes, os itens da cópia de trabalho), gerando modificações, inicialmente não registradas, que ficam registradas localmente nos itens da cópia de trabalho. Por meio da ontologia foi explicitado também que um *Commit* realizado no SVN é responsável por registrar as modificações (alterações locais) realizadas nas versões retiradas e modificadas por meio de seus representantes, os itens da cópia de trabalho. As modificações de cada item da cópia de trabalho são então registradas por meio do Commit, que registra no repositório as alterações realizadas localmente, dando origem a novas versões do IC retirado para alteração (novas revisões dos itens do repositório). Além disso, foi explicitado que um *Import* efetuado em SVN é uma maneira de selecionar Itens de Configuração, que estão na forma de arquivos. Essa seleção ocorre por meio de um diretório não versionado, contendo arquivos não versionados, que é selecionado, sendo então adicionado no repositório, passando a ser versionado. Como foi ilustrado através do Subversion, por meio dos mapeamentos verticais foi possível tornar mais clara a semântica por trás dos serviços oferecidos pelos sistemas.

Como ocorreu com outros tipos de mapeamentos verticais, a ontologia de referência não foi capaz de cobrir todos os serviços e atividades envolvidos na integração, como por exemplo, os serviços "*Criar Ramificação de Repositório"* e "*Criar Repositório do Projeto"* de GCS-ODE ou "*Copiar Item do Repositório*" e "*Criar Repositório*" de SVN. Tais serviços não receberam mapeamentos verticais e, por isso, não tiveram sua semântica explicitada.

A próxima etapa do processo de integração é a elaboração do modelo de integração, o qual deverá abranger todos os serviços que a ontologia não foi capaz de abranger. Como o modelo de integração representa tanto os aspectos comportamentais quanto os aspectos estruturais da integração, ele deve envolver também os conceitos e relacionamentos que não foram mapeados para a ontologia.

5.3.3. Modelo de Integração

Depois de se realizar os mapeamentos verticais, passa-se ao desenvolvimento do modelo de integração. Ele é responsável por servir de referência para atribuição de semântica em uma iniciativa de integração, considerando particularidades da mesma.

O esboço inicial da parte estrutural do modelo de integração é o fragmento da ontologia de referência que contém os conceitos e relações usados nos mapeamentos verticais estruturais. De maneira análoga, o esboço inicial da parte comportamental do modelo de integração é o fragmento da ontologia de referência que contém as tarefas, insumos e produtos usados nos mapeamentos verticais comportamentais. A partir desse esboço inicial, o modelo de integração deve ser alterado para comportar os elementos dos sistemas que não foram mapeados com a ontologia ou dos que foram, mas que a conceituação atribuída ainda não ficou bem especificada. Assim, ele é composto, basicamente, dos elementos da ontologia de referência, mais os elementos dos sistemas não cobertos por ela.

Devido à sua complexidade, o modelo de integração será apresentado em partes nesta seção. Os conceitos adicionados no modelo de integração serão representados em cinza escuro. Elementos de modelo em cinza claro já foram definidos, mas estão sendo usados na definição de outros. Diferentemente da ontologia, o modelo de integração não foi construído usando OntoUML. Foi usada apenas a UML, sem as distinções semânticas de UFO. Entretanto OntoUML poderia ser usado, o que poderia trazer benefícios adicionais.

Na construção do modelo de integração estrutural, procurou-se envolver os seguintes conceitos que não estavam presentes na ontologia de referência: *Arquivo* (SVN e GCS-ODE), *Diretório* (SVN), *Revisão* (SVN), *Nó de Ramificação* (GCS-ODE), *Artefato Interno* (GCS-ODE), *Artefato Externo* (GCS-ODE) e *Item Copiado* (SVN).

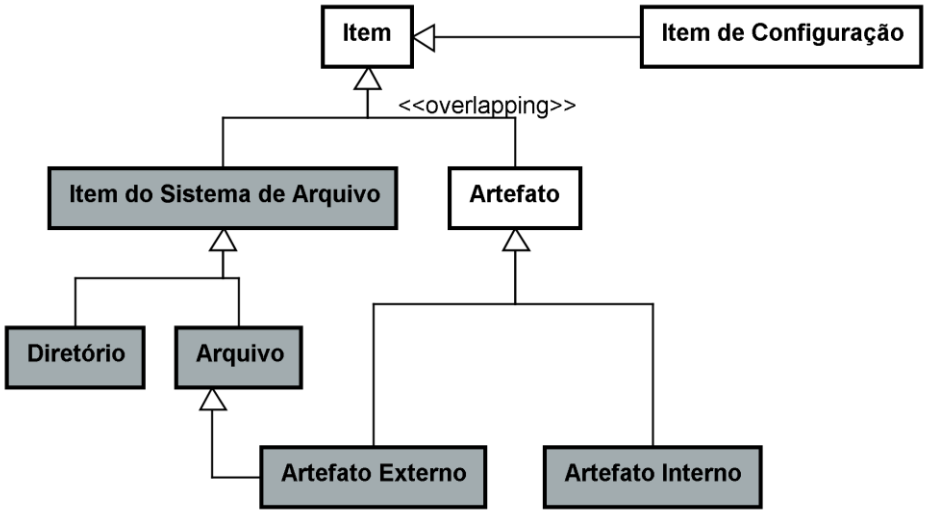

Figura 5.7– Modelo de Integração: Tipos de Itens

A Figura 5.7 apresenta uma parte do modelo de integração que adiciona os elementos de modelo *Item do S.A.* (item do sistema de arquivo), *Arquivo*, *Diretório*, *Artefato Interno* e

Artefato Externo. Todos eles são tipos de item e, portanto, representam diferentes tipos de elementos que podem ter sua configuração gerenciada, quando se tornam ICs.

Um *Item* pode representar um *Item do Sistema de Arquivo* (SVN) ou um *Artefato* (GCS-ODE), sendo que essa hierarquia é do tipo *overlapping*, ou seja, uma instância de *Item do S.A.* pode ser instância de *Artefato* também (no caso, instância de *Artefato Externo*, já que apenas artefatos externos são representados na forma de arquivos). *Artefatos* (GCS-ODE) podem ser externos (representados na forma de *Arquivo*) ou internos. *Itens do S.A.* podem ser *Diretórios* ou *Arquivos*.

A Figura 5.8 apresenta a parte do modelo de integração que introduz os elementos de modelo *Item Copiado* e *Nó de Ramificação*, de modo a tratar os respectivos elementos *Item Copiado* (SVN) e *Nó de Ramificação* (GCS-ODE). *Item Copiado* representa um IC que foi originado a partir de uma cópia de outro IC em uma versão específica dele. Ele é usado para representar o conceito *Item Copiado* (SVN). O conceito Ramificação, apesar de já pertencer à ontologia, foi representado também como sendo um tipo de IC Composto originado de uma configuração. Essa nova concepção foi adicionada para permitir a introdução do elemento de modelo Nó de Ramificação, que é definido como sendo uma configuração referente a uma ramificação.

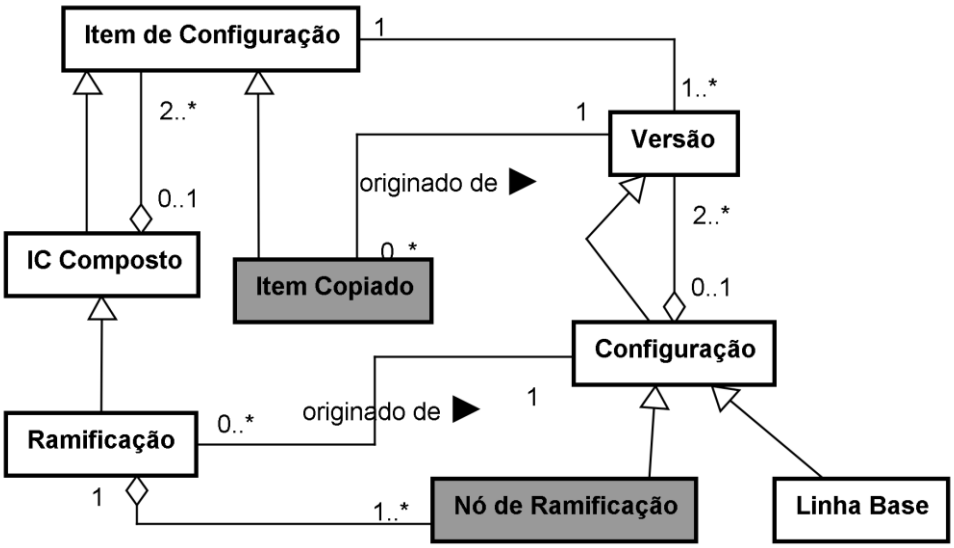

Figura 5.8–Modelo de Integração: Item Copiado e Nó de Ramificação

Além de envolver a introdução de conceitos relativos aos sistemas que não foram contemplados pela ontologia, o modelo de integração pode adicionar novos relacionamentos

para contemplar relacionamentos não cobertos pela ontologia. A Figura 5.9 apresenta um exemplo de adição de um relacionamento que não foi coberto pela ontologia.

Com relação aos outros relacionamentos que não foram contemplados pela ontologia, a saber "Ramificação **é composta de** Nó de Ramificação" (GCS-ODE), "Nó de ramificação **composto de** Revisão" (GCS-ODE), não foi preciso adicionar novos relacionamentos, uma vez que, com a adição de conceitos referentes a Nó de Ramificação no modelo de integração, os referentes relacionamentos foram cobertos.

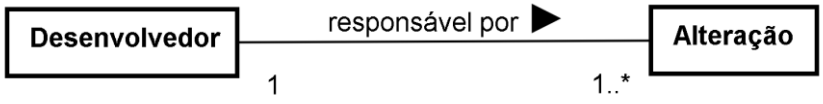

Figura 5.9 – Adição de Relacionamentos ao Modelo de Integração

Tabela 5.15 – Tarefas adicionais do Modelo de Integração Comportamental

TAREFA	DESCRIÇÃO
Criar Cópia de IC	Tarefa realizada por um gerente de configuração que é responsável por criar um Item Copiado (um IC gerado a partir de uma versão específica de item de configuração)
ic de origem: Item de Configuração	Item de configuração a partir do qual é criada uma cópia
versão de origem: Versão	Versão do IC usada como base para a cópia
item copiado: Item Copiado	cópia gerada
Criar Repositório	Tarefa realizada pelo gerente de configuração que é responsável por criar um repositório para um projeto
projeto: Projeto	Projeto para o qual o repositório será criado
repositório: Repositório	Repositório criado
Criar Ramificação	Tarefa responsável por criar uma ramificação a partir de uma configuração
conf. origem : Configuração	Configuração a partir da qual será criada uma ramificação
ram: Ramificação	Ramificação gerada

O modelo de integração também deve envolver os aspectos comportamentais da integração para envolver os serviços dos sistemas que não tiveram uma semântica atribuída pela ontologia. Por exemplo, os serviços "*Criar Ramificação de Repositório"* e "*Criar Repositório do Projeto"* de GCS-ODE ou "*Copiar Item do Repositório*" e "*Criar Repositório*" de SVN não foram contemplados pela ontologia de referência. A Tabela 5.15 mostra as tarefas

adicionais criadas para abranger os serviços citados. As tarefas são representadas pelas células em cinza escuro, os insumos pelas que estão de cinza claro e os produtos pelas células em branco. Como a tabela apresenta, foram representadas no modelo de integração comportamental três novas tarefas, referentes à criação de cópias de IC, criação de repositório e criação de ramificação.

5.3.4. Mapeamentos Horizontais

Depois da criação do modelo de integração, é realizada a etapa de mapeamentos horizontais, como mostra a Tabela 5.16.

Tabela 5.16 – Mapeamentos Horizontais entre Conceitos

MODELO DE INTEGRAÇÃO	SUBVERSION	GCS-ODE
Item do Sistema de Arquivo *Arquivos e diretórios de um sistema de arquivo*	**Item do Sistema de Arquivo** *Arquivos e diretórios de um sistema de arquivo*	------
Artefato *Elementos consumidos e produzidos em um projeto*	------	**Artefato** *Elementos consumidos e produzidos em um projeto*
Artefato Interno *Tipo de artefato que foi produzido internamente no ambiente representado na forma de objetos*	------	**Artefato Interno** *Tipo de artefato que foi produzido internamente no ambiente representado na forma de objetos*
Artefato Externo *Item do sistema de arquivo que representa um artefato*	------	**Artefato Externo** *Tipo de artefato que foi produzido externamente ao ambiente e que é representado na forma de arquivo*
Diretório *Tipo de item do sistema de arquivo responsável por estruturar e organizar um conjunto de Itens do Sistema de Arquivo*	**Diretório** *Tipo de item do sistema de arquivo responsável por estruturar um conjunto de Itens do Sistema de Arquivo*	------
Nó de Ramificação *Tipo de configuração que representa a situação de uma ramificação em um dado momento*	------	**Nó de Ramificação** *Conjunto de versões que representa determinado estado de uma ramificação*
Item Copiado *Item de Configuração gerado a partir de uma cópia de outro IC numa versão específica*	**Item Copiado** *Item do repositório criado a partir da cópia de outro item do repositório*	------

Esses mapeamentos são responsáveis por tratar aspectos particulares da corrente iniciativa de integração. Eles são baseados nos mapeamentos verticais e tendem a mantê-los. Assim, são apresentados apenas os mapeamentos horizontais que não têm correspondência com os mapeamentos verticais. Uma vez feitos os mapeamentos horizontais de conceitos, devem-se mapear os relacionamentos. A Tabela 5.17 apresenta os mapeamentos horizontais de relacionamentos que não têm correspondência com os mapeamentos verticais de relacionamentos já efetuados.

Tabela 5.17 – Mapeamentos Horizontais de Relacionamentos

MODELO DE INTEGRAÇÃO			GCS-ODE		
CONCEITO	RELAÇÃO	CONCEITO	CONCEITO	RELAÇÃO	CONCEITO
Desenvolvedor	**é responsável por**	Alteração	Recurso Humano	**é responsável por**	Alteração
Ramificação	**possui**	Nó de Ramificação	Ramificação	**é composta de**	Nó de Ramificação
Configuração (Nó de Ramificação)	**é composta de**	Versão	Nó de Ramificação	**composto de**	Revisão

Mapeamento Horizontal Comportamental

Depois de realizado o mapeamento horizontal estruturtal, é realizado o mapeamento horizontal comportamental. A Tabela 5.18 apresenta os mapeamentos horizontais comportamentais realizados no contexto desta iniciativa de integração. Da mesma forma que a tabela que apresenta as novas tarefas introduzidas no modelo de integração (Tabela 5.15), na Tabela 5.18 as tarefas/serviços estão representadas nas linhas em cinza escuro, os insumos nas linhas que estão de cinza claro e os produtos nas linhas em branco.

No mapeamento relativo à tarefa "*Criar Cópia de IC*" (Modelo de Integração), o insumo *item do repositório*: Item do Repositório (SVN) foi mapeado para o insumo *ic de origem: Item de Configuração* (Modelo de Integração), representando o IC a ser copiado. O insumo *revisão: Revisão* (SVN) foi mapeado para o insumo *versão: Versão* (Ontologia), uma vez que ambos representam a versão a partir da qual a cópia será criada.

Por meio dos mapeamentos horizontais de comportamento, foi possível identificar serviços de SVN e GCS-ODE que são semanticamente equivalentes e que poderão ser integrados entre si, como é o caso do serviço de criação de repositório presente em SVN e GCS-ODE.

Tabela 5.18 - Mapeamentos horizontais de comportamento

MODELO DE INTEGRAÇÃO	PROCESSO	SVN	GCS-ODE
Criar Cópia de IC	---	**Copiar Item do Repositório**	
ic de origem: Item de Configuração	---	Item do repositório: Item do Repositório	
versão: Versão	---	revisão:Revisão	
copia**:** Item Copiado	---	item copiado:Item Copiado	
---	---	alteração do repositório	---
Criar Repositório	---	**Criar Repositório**	**Criar Repositório do Projeto**
projeto: Projeto	---	---	projeto:Projeto
repositório:Repositório	---	repositório:Repositório	repositório:Repositório
Criar Ramificação		**---**	**Criar Ramificação**
conf. origem : Configuração		---	no de origem: Nó de ramificação
ram: Ramificação		---	ram: Ramificação

Como dito anteriormente, GCS-ODE é um sistema focado no controle de alteração, enquanto o SVN tem foco no controle de versão. Na presente iniciativa de integração, o SVN é usado por GCS-ODE para fazer controle de versão dos artefatos externos gerenciados. GCS-ODE não faz nenhum controle sobre o conteúdo de tais artefatos que estão na forma de arquivos. GCS-ODE apenas manipula as informações referentes ao gerenciamento da configuração de tais artefatos. Com a integração com SVN, os conteúdos dos artefatos externos de GCS-ODE (arquivos) são armazenados em repositórios do SVN. Tais arquivos são acessados, alterados e têm suas versões controladas por meio do SVN. Assim, eles são alterados por meio de cópias de trabalho, como trabalha o SVN.

Há de se realçar que haverá replicação de dados no SVN e em GCS-ODE. Assim, é muito importante que haja um controle de redundância entre os sistemas e que se estabeleça quais serviços são equivalentes e que devem ser sincronizados. Por exemplo, ao ser criado um repositório em GCS-ODE, o mesmo deve ocorrer em SVN; ao ser criada uma ramificação em GCS-ODE, uma ramificação (diretório copiado) também deverá ser criada em SVN; ao se criar uma linha base em GCS-ODE, os arquivos referentes deverão ser marcados de alguma forma no repositório SVN; quando um desenvolvedor retirar uma alteração em GCS-ODE, o SVN deve realizar a operação de *checkout* dos arquivos correspondentes, gerando uma cópia

de trabalho; finalmente, quando um desenvolvedor, usando GCS-ODE, registra uma alteração, as alterações realizadas na cópia de trabalho devem ser registradas por meio de um *commit* no SVN.

Por meio dos mapeamentos verticais e horizontais estabelecidos, é possível identificar quais serviços são equivalentes e que, portanto, precisam ser sincronizados de modo a garantir a operação consistente dos sistemas. Além disso, com a integração dos sistemas, algumas informações serão replicadas em ambos. Com o uso dos mapeamentos entre conceitos e relacionamentos, é possível identificar os conceitos/relacionamentos replicados, de modo a se buscar mantê-los sincronizados e consistentes.

5.3.5. Projeto e Implementação da Integração

Uma vez desenvolvido o modelo de integração e realizados os mapeamentos horizontais, podem ser realizados o projeto e a implementação da solução de integração.

Para que GCS-ODE e SVN possam se comunicar, é necessário que os elementos compartilhados sejam traduzidos. A solução proposta consiste em desenvolver um mediador responsável por traduzir os dados compartilhados entre os sistemas e que controla a invocação de serviços, como ilustra a Figura 5.10.

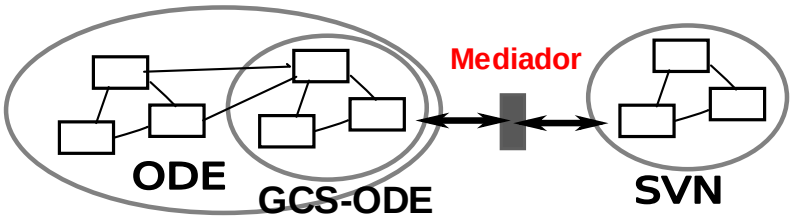

Figura 5.10 –Mediador: responsável pela comunicação entre GCS-ODE e SVN

Como mostra a figura, GCS-ODE é uma ferramenta interna do ambiente ODE (compartilhando conceitos com o mesmo), enquanto o SVN é um sistema externo ao ODE. GCS-ODE e SVN se comunicam através de um mediador, responsável por traduzir os conceitos de GCS-ODE para SVN (e vice-versa), de modo que os sistemas possam se comunicar de maneira adequada. O mediador também é responsável pela integração de serviços e processo, possibilitando a troca de funcionalidades e a execução orquestrada das mesmas, em função com o processo de negócio.

Para realizar a tradução de dados entre os sistemas, o mediador deve armazenar informações sobre os mapeamentos, permitindo a identificação de correspondências entre os

dados dos sistemas. Por exemplo, para um objeto do tipo *Ramificação* em GCS-ODE, o mediador deve registrar a localização do *Diretório* correspondente no *Repositório* do SVN. Para um objeto *Revisão* em GCS-ODE, por sua vez, o mediador deve armazenar a localização do *Arquivo* correspondente no *Repositório* do SVN e a *Revisão* a ele relacionada.

Os dados dos sistemas integrados são armazenados, tipicamente, em três lugares: no banco de dados de ODE, nas cópias de trabalho criadas pelo SVN e nos repositórios do SVN. É responsabilidade do mediador considerar essa questão na tradução de dados entre os sistemas, de modo a tratar a integração na camada de dados.

É importante citar que, em função de requisitos não funcionais, decisões de projeto que alterem os mapeamentos feitos, podem ser tomadas para garantir algum atributo de qualidade no sistema resultante. Por exemplo, apesar de os conceitos de *Repositório* em GCS-ODE e *Repositório* no SVN serem conceitualmente equivalentes, durante o projeto a classe *Repositório* de GCS-ODE pode ser relacionada a um *Diretório do Repositório* de SVN. Tal decisão está relacionada à usabilidade do sistema, pois seria bastante complexo criar um repositório SVN para cada projeto criado em ODE, visto que a criação de um repositório SVN não é uma tarefa simples, envolvendo fatores como o sistema operacional utilizado, e sendo necessário configurar do acesso remoto do mesmo. Assim sendo, pode ser uma decisão de projeto considerar que os artefatos da organização seriam armazenados em um único repositório SVN.

Com relação ao projeto da integração comportamental, o mediador deve utilizar funcionalidades fornecidas pelo SVNKit (http://svnkit.com/), uma biblioteca que provê os serviços SVN em Java, para a realização das funcionalidades de controle de versão de SVN. Um fator importante a ser considerado na integração comportamental nesta iniciativa de integração é a manutenção da integridade das informações replicadas nos diferentes locais de armazenamento. A Figura 5.11 ilustra um exemplo dessa situação.

Uma revisão *r1* de um item de configuração no banco de dados do ambiente ODE representa uma revisão de item *f1* de um arquivo localizado no repositório SVN. O arquivo *a*, pertencente à cópia de trabalho *c.t.1*, representa, inicialmente, uma cópia de *f1* que possui alterações locais. No registro das alterações da cópia de trabalho *c.t.1*, o mediador deve manipular as informações no banco de dados de ODE e usar o SVNKit para manipular as informações contidas na cópia de trabalho *c.t.1* e no repositório SVN. Neste caso, o mediador deve criar uma nova revisão *r2* e solicitar ao SVNKit que efetue o registro da alteração. O SVNKit, por sua vez, deve alterar o estado do item da cópia de trabalho *a* para indicar que

suas alterações foram registradas, criar uma nova versão *f2* do arquivo no repositório e fazer o arquivo *a* referenciar *f2*.

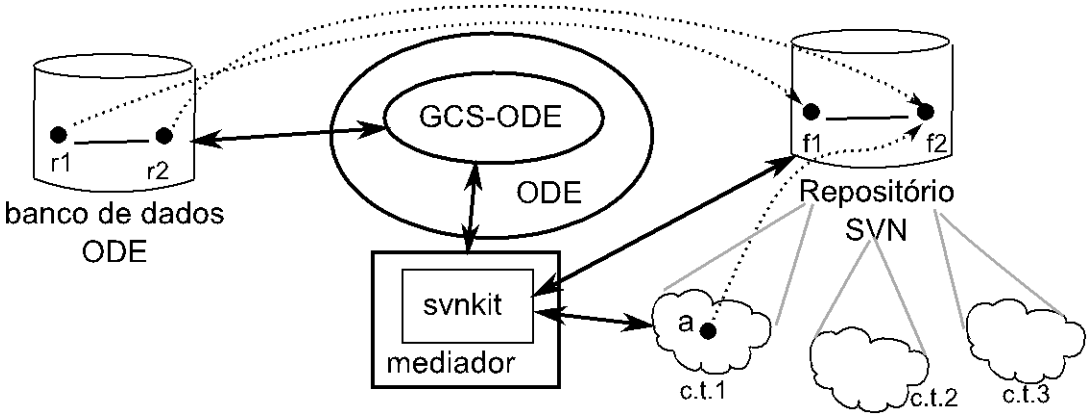

Figura 5.11 – Localização dos Objetos de GCS-ODE e integração de controle

5.4. Conclusão

Os sistemas de GCS existentes geralmente apoiam apenas parte do processo de GCS. Se for desejado fornecer um apoio mais abrangente a esse processo é necessário integrar diferentes sistemas, tais como sistemas de controle de versão e de controle de alteração. Entretanto, a integração entre tais sistemas, sobretudo no nível semântico, não ocorre de modo direto, uma vez que eles não compartilham uma conceituação comum do domínio e das tarefas de GCS.

Este capítulo mostrou uma aplicação prática da abordagem de integração semântica proposta neste trabalho, mostrando como suas diretrizes podem ajudar em um caso prático de interação entre sistemas. A iniciativa de integração envolveu o sistema de controle de versão Subversion e o sistema de controle de alteração GCS-ODE, este último desenvolvido no contexto do ambiente ODE.

A abordagem de integração semântica ajudou a explicitar a conceituação, muitas vezes implícita de cada sistema, por meio de modelos de referência (ontologia de referência). Isso permitiu uma comparação entre as conceituações e seus elementos, facilitando a identificação de elementos equivalentes e que devem ser integrados.

A integração semântica deve ocorrer antes do projeto da solução de integração, evitando um entendimento equivocado do significado dos elementos dos sistemas a serem integrados.

OBA-SI, a abordagem de integração semântica aplicada na iniciativa de integração discutida neste capítulo, mostrou ser bastante útil para a solução de problemas de integração no nível semântico. Entretanto, ela ainda possui aspectos a serem melhorados. Tais informações são discutidas a seguir, no capítulo conclusivo desta dissertação.

Capítulo 6 - Considerações Finais

É comum nos diversos tipos de organizações, o uso de diferentes tipos de sistemas para apoiarem conjuntamente os processos de negócios da organização. Para que esse apoio ocorra de modo eficiente, é importante que os sistemas estejam integrados, compartilhando dados e serviços no apoio a um processo. Neste contexto, a integração de sistemas traz inúmeros benefícios à organização, dentre eles: o processo de negócio pode ganhar maior flexibilidade e adaptabilidade, a sua execução se torna mais eficiente e rápida, diminui a incidência de erros e aumenta-se o poder de tomada de decisão da empresa (POKRAEV, 2009).

Entretanto, geralmente tais sistemas não são desenvolvidos para trabalharem em conjunto. Eles a priori não estão aptos a se comunicarem entre si para juntos apoiarem o processo de negócio. Na maioria das vezes, sistemas funcionam como caixas pretas, não sendo possível acessar seu conteúdo e, às vezes, nem mesmo sua interface.

Por trás dos dados e serviços compartilhados por sistemas estão os modelos de dados e de comportamento dos mesmos, construídos em função de uma conceituação da realidade. Geralmente cada sistema é construído com base em uma conceituação da realidade distinta, havendo, assim, divergências nos modelos de dados e de comportamento.

Esse é o problema da heterogeneidade, vem sido considerado o mais desafiador na integração de sistemas, sobretudo aqueles construídos isoladamente (IZZA, 2009). Para que esse problema seja solucionado e que os sistemas trabalhem em conjunto, é preciso que, antes de tudo, as diferentes conceituações que eles possuem sejam compatibilizadas, obtendo, assim, um consenso a respeito da semântica de cada conceituação específica.

Entretanto, realizar a integração semântica de um conjunto de sistemas de modo a compatibilizar tais conceituações pode ser uma tarefa um tanto subjetiva e complexa. Tais conceituações encontram-se presentes nos sistemas de modo implícito. Assim, para que ocorra a integração entre sistemas por meio do alinhamento de suas conceituações, é preciso primeiro torná-las explícitas e depois mapeá-las.

Este trabalho apresentou uma abordagem de integração semântica de sistemas baseada em ontologias, que visa sistematizar o trabalho de explicitação e compatibilização de conceituações de sistemas. Na abordagem apresentada a integração semântica é realizada por meio de modelos conceituais, responsáveis por explicitar as conceituações de cada sistema, e ontologias do domínio e das tarefas envolvidas. A compatibilização das

conceituações dos sistemas em OBA-SI ocorre por meio de mapeamentos semânticos. Eles são responsáveis por comparar os modelos conceituais dos sistemas sob a luz de uma ontologia de referência.

Por meio do uso de ontologias de referência para atribuir semântica aos modelos conceituais dos sistemas, o problema de conflito semântico relativo às diferenças de conceituação de uma mesma entidade do mundo real, ilustrado na Figura 2.8, é reduzido. A solução desse tipo de conflito semântico é o foco de OBA-SI e sua estratégia é o uso de ontologias de qualidade, com expressividade suficiente para que as diferentes conceituações da mesma porção da realidade sejam relacionadas e comparadas, facilitando que conflitos desse tipo sejam resolvidos. Já os conflitos semânticos que ocorre entre símbolos e suas diferentes conceituações, ilustrados na Figura 2.7, não são o foco deste trabalho. Para solução desse tipo de conflito, pode-se usar em conjunto com OBA-SI a abordagem proposta por Pokraev (2009).

A integração semântica em OBA-SI ocorre em níveis, possibilitando a separação de preocupações e que a integração ocorra em um alto nível de abstração. A integração começa pelo nível ontológico, que é independente da iniciativa de integração e busca unicamente representar os domínios e as tarefas envolvidos. Depois a integração é realizada no nível de análise da integração, que soluciona a integração semântica de modo independente de tecnologia e de soluções específicas, para depois ocorrer no nível de projeto e implementação, o qual considera tais aspectos. Deve-se ressaltar que a solução da integração semântica em OBA-SI, ocorrendo primeiramente no nível de ontologia, se baseia no entendimento do domínio e tarefas em questão para prover uma integração semântica de qualidade.

Um dos principais diferenciais de OBA-SI, com relação às abordagens de integração semântica estudadas, é o uso de ontologias de referência. OBA-SI considera ontologias de referência de domínio e de tarefa para atribuir semântica aos elementos de modelo. Como mencionado em (GUIZZARDI et al., 2009), uma ontologia de referência deve ser usada, ajudando na representação do domínio sem considerar requisitos computacionais. O principal objetivo de uma ontologia de referência é ajudar modeladores a exteriorizar o conhecimento tácito sobre o domínio para tornar explícitos os comprometimentos ontológicos, facilitando a negociação do significado, as tarefas de comunicação sobre o domínio, aprendizado e solução de problemas da melhor maneira.

OBA-SI faz uso de ontologias de domínio e tarefa para atribuição de semântica a uma iniciativa de integração, considerando tais artefatos como independentes da integração, construídos unicamente com o intuito de descrever o domínio e tarefas envolvidos da melhor maneira possível, podendo ser reutilizados em várias iniciativas de integração. Uma desvantagem nessa estratégia é com relação ao uso de linguagens bastante expressivas, como OntoUML, que por um lado é responsável por descrever a realidade de modo claro e preciso, mas por outro são de difícil utilização. Na realização do estudo de caso percebeu-se essa dificuldade na criação da ontologia que foi bastante custosa devido ao uso de OntoUML. Por um lado ela proporciona um modelo com bastante expressividade e qualidade, mas por outro existe uma complexidade para sua construção, uma vez que suas primitivas bem fundamentadas são de difícil entendimento.

Ontologias em OBA-SI são usadas para explicitar a conceituação comum que está implícita nos sistemas a serem integrados. Entretanto, por ser independente da iniciativa de integração, ontologias podem não ser suficientes para tal. Dessa forma, OBA-SI divide a conceituação comum em dois níveis: um geral e independente da iniciativa de integração e outro específico e dependente dela. Ontologias são usadas para representar a conceituação geral, enquanto OBA-SI faz o uso dos modelos de integração para representar a conceituação comum mais específica, envolvendo toda a conceituação dos sistemas a serem integrados.

OBA-SI procura envolver as três camadas de integração: dados, serviço e processo. A integração semântica na camada de dados é feita por meio de mapeamentos semânticos entre os modelos estruturais dos sistemas, tomando como base principalmente a ontologia de referência. Com relação à integração semântica nas camadas de serviço e processo, OBA-SI faz uso de ontologias de tarefa para atribuir semântica aos elementos de modelos comportamentais dos sistemas (descrevendo serviços, insumos, produtos e o fluxo de controle) e do modelo de processo de negócio da organização, possibilitando a identificação de tarefas/serviços, insumos e produtos que são equivalentes.

Uma iniciativa de integração deve gerar um produto de software, que foi analisado, projetado e implementado. Para diminuir a complexidade do processo de integração, OBA-SI propõe um processo de integração. A integração de sistemas é vista como um tipo de processo de desenvolvimento de software, composto de fases de levantamento de requisitos, análise, projeto, implementação, testes e implantação. Entretanto, como OBA-SI possui como premissa que a integração deve ocorrer primeiramente no nível semântico, antes de ser projetada e implementada, ela é centrada na atividade de análise da integração, onde a

semântica deve ser definida. Por conseguinte, ela fornece apenas orientações para as outras fases do processo de desenvolvimento, sendo uma das limitações da abordagem.

OBA-SI baseia-se em mapeamentos semânticos para se estabelecer as concordâncias, mas o procedimento de mapeamento não é seu foco. OBA-SI se preocupa principalmente em fornecer passos metodológicos para a realização da integração semântica no nível conceitual. Dessa forma, OBA-SI pode ser combinada com outras abordagens que são focadas na realização de mapeamentos semânticos, como as apresentadas em (NOY, 2004) e (IZZA, 2009).

Com relação ao processo de integração de OBA-SI, ele dá ênfase à análise da integração. Essa tarefa é bastante subjetiva e, por isso, pode ser difícil a realização de mapeamentos automáticos, como preconizado por várias abordagens. Dessa forma, o processo proposto é inteiramente manual. Entretanto, isso não é um problema em si, uma vez que OBA-SI é uma abordagem para integração de sistemas existentes em uma organização em um cenário de integração previamente definido. Essa seria uma limitação se OBA-SI estivesse interessada na integração de serviços em tempo de execução. Mas esse não é o foco de OBA-SI, para isso outras abordagens podem ser usadas em conjunto com OBA-SI.

Enfim, podem-se destacar como as principais contribuições deste trabalho:

- Desenvolvimento de OBA-SI, uma abordagem de integração semântica de sistemas, baseada em ontologias de referência;
- Desenvolvimento de uma Ontologia da Tarefa de Gerência de Configuração;
- Realização da análise da integração, segundo OBA-SI, entre as ferramentas GCS-ODE e SVN no apoio ao processo de Gerência de Configuração de Software.

Parte dos resultados obtidos neste trabalho está no artigo "*An Ontology-based Approach for Semantic Integration*", apresentado e publicado nos anais da *14th IEEE International Enterprise Distributed Object Computing Conference* (EDOC 2010) (CALHAU; FALBO, 2010).

6.1. Perspectivas Futuras

Como todo projeto de pesquisa, o trabalho realizado nesta dissertação possui diversos pontos a serem melhorados. Principalmente, é importante que OBA-SI seja aplicada em

diferentes iniciativas de integração, de modo que ela possa ganhar mais maturidade por meio do *feedback* obtido, sobretudo em domínios de entendimento complexo e que possam dar margem a diversos problemas de integração semântica.

Outra perspectiva futura poderia ser explorar mais o uso de ontologias de fundamentação no processo de integração semântica. Notou-se no estudo de caso que o seu uso melhorou a qualidade da integração semântica dos sistemas. Seu uso permitiu um maior entendimento do domínio de GCS e da tarefa de GC, trazendo benefícios posteriores na integração. Em diversas situações, o uso de OntoUML promoveu uma maior reflexão e entendimento a cerca dos conceitos do domínio devido à necessidade de categorizá-los em relação às distinções semânticas da Ontologia de Fundamentação Unificada (UFO) (GUIZZARDI, 2005). Dessa forma, seria interessante um estudo mais aprofundado sobre o benefício do uso de ontologias de fundamentação na integração semântica, podendo inclusive ser proposto diretrizes para seu uso. O uso da ontologia de fundamentação poderia ser explorado também na etapa de extração dos modelos conceituais dos sistemas e também na representação do modelo de integração.

Além disso, é interessante que o processo de integração seja estendido de modo a apoiar também a criação de ontologias, uma vez que dificilmente se têm ontologias já construídas como OBA-SI assume. Uma estratégia interessante é possibilitar a construção de ontologias para uma integração específica e com o tempo ir refinando-a de modo a ficar cada vez mais independente de solução e mais comprometida como uma descrição da realidade. Alias, também seria interessante que OBA-SI desse apoio à atualização e evolução das ontologias, uma vez que é comum a identificação de novos conceitos ou melhorias ao usá-las na integração de sistemas.

Outra perspectiva futura deste trabalho é melhorar a atribuição semântica na integração comportamental. Uma possibilidade é estudar o uso de classificação semântica de ações (ou até mesmo classificação semântica de verbos) para melhorar a atribuição de semântica na integração comportamental. Através do uso de uma categoria semântica de ações, a identificação de serviços equivalentes poderia ser facilitada. Por exemplo, no mapeamento dos serviços “Realizar Empréstimo” e “Efetuar Empréstimo”, através do uso de tal classificação, poderia ser identificada uma relação mais precisa entre as ações “Realizar” e “Efetuar”, fornecendo mais expressividade na integração de serviços.

Outra maneira de se melhorar a atribuição de semântica na integração comportamental é explorar ainda mais o uso de ontologias de tarefa. No estudo de caso, foi

criada uma ontologia de tarefa de Gerência de Configuração. Entretanto, essa tarefa, apesar de ser genérica, é composta de diversas tarefas que também são genéricas como, por exemplo, solicitar, avaliar, verificar, implementar e modificar. Se a ontologia de tarefa de GC fosse construída com base em ontologias de tarefas que descrevessem essas atividades, ela ganharia mais expressividade o que ajudaria na integração.

OBA-SI poderia ser usada em conjunto com outras abordagens, principalmente as que possuem um enfoque mais tecnológico, como (BOURAS et al, 2006), (TEKTONIDIS et al, 2005) e (POKRAEV, 2009), . A combinação de OBA-SI com tais abordagens poderia trazer um retorno bastante benéfico para a abordagem. A integração realizada em OBA-SI em um nível conceitual poderia ser implementada usando linguagens de implementação de ontologias como OWL, o que permitiria simular a integração, fazer inferências e verificações, ou então com abordagens de EAI para ajudar na integração de processo ou SOA para ajudar na integração de serviços. SOA junto com processo de negócio, por exemplo, poderia prover uma solução interessante em conjunto com OBA-SI. Ele poderia ser usado para abstrair os sistemas a serem integrados por meio de serviços compartilhados e reusáveis. A semântica relativa a tais serviços, aos dados e ao processo envolvido seria atribuída por OBA-SI e necessitaria ser replicado na solução usando SOA também, servindo como uma referência na hora de combinar e usar serviços, trocar dados e saber qual serviço usar para cada atividade do processo. OBA-SI, portanto, pode ser combinada com outras abordagens, servindo como base para o entendimento de elementos a serem combinados e compartilhados na solução de integração semântica.

Inclusive também seria interessante também que OBA-SI também apoiasse a fase de projeto da integração, ao invés de estar apenas focada na análise da integração, Nesse caso, poderia ser criada uma especialização da abordagem para sua solução em uma arquitetura específica, como, por exemplo, SOA. Nessa especialização de OBA-SI para SOA, seriam consideradas especialidades no processo de integração relativas a esse tipo de arquitetura.

Referências Bibliográficas

ARANTES, L.O., Falbo, R.A., Guizzardi, G.: **Evolving a Software Configuration Management**, 2nd Workshop on Ontologies and Metamodeling Software and Data Engineering, Brazil, (2007)

BOURAS, A. . G. P. . F. A. . P. S. . A. S. . M. G. **Business Process Fusion based on Semantically-enabled Service-Oriented Business Applications**. Interoperability for Enterprise Software and Applications Conference (I-ESA´06), Bordeaux, France, Março 2006. 157-168.

BRINGUENTE, A. C.: **Reengenharia de uma Ontologia de Processo de Software e seu Uso para Integração de Ferramentas de Apoio ao Planejamento de Projetos**. Dissertação de Mestrado, Universidade Federal do Espírito Santo (UFES), Vitória (2009)

BUSSLER, C.: **The Role of Semantic Web Technology in Enterprise Application Integration,** Volume: 51, Issue: 4, Pages: 1-7, Data Engineering (2003).

CAETANO, C.; CVS: **Controle de Versões e Desenvolvimento Colaborativo de Software**. Editora Novatec (2004).

CALHAU, R.F., ARANTES, L.O., FALBO, R.A.: **Uso de Gerência de Conhecimento para Apoiar a Rastreabilidade e a Avaliação de Impacto de Alterações**, XXII Simpósio Brasileiro de Engenharia de Software – SBES´2008, Campinas, Brasil, 2008.

CALHAU, R.F., FALBO, R.A.: **GCS-ODE: Uma Ferramenta de Apoio à Gerência de Configuração de Software Integrada ao Ambiente ODE**, Sessão de Ferramentas do XXII Simpósio Brasileiro de Engenharia de Software – SBES´2008, Campinas, Brasil, 2008.

CALHAU, R.F.; **GCS-ODE: Uma Ferramenta de Apoio À Gerência de Configuração de Software Integrada ao Ambiente ODE**, Trabalho de Conclusão de Curso, Universidade Federal do Espírito Santo, UFES, 2009.

CALHAU, R.F.; FALBO, R.A.; **An Ontology-Based Approach for Semantic Integration**. Proceedings 14th IEEE International Enterprise Distributed Object Computing Conference, Vitória, Brasil (2010).

CHIAVENATO, Idalberto; **Introdução à Teoria Geral da Administração**; 7a. Edição, Ed. Elsevier, 2004

COLLINS-SUSSMAN B., FITZPATRICK B. W., PILATO C. M.: **Version Control with Subversion**. Disponível em: <svnbook.red-bean.com/nightly/en>. Acesso em: 07 ago. 2011.

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.

ESTUBLIER, J., **Software Configuration Management: A Roadmap**, In: Proc. of the Future of Software Engineering, ICSE'2000, Ireland, (2000).

FALBO, R.A., RUY, F.B., MORO, R.D.; **Using Ontologies to Add Semantics to a Software Engineering Environment**. In: 17th International Conference on Software Engineering and Knowledge Engineering - SEKE'2005, Taipei, China, p. 151-156. (2005)

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

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. ISBN 978-1-58603-640-8.

GUIZZARDI, G., LOPES, M., BAIÃO, F.A., FALBO, R.A.; **On the Importance of Truly Ontological Distinctions for Ontology Representation Languages: An Industrial Case Study in the Domain of Oil and Gas**. In: 14th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD), Amsterdam, The Netherlands, 2009. p. 1-13.

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

IAN THOMAS, H.; NEJMEH, B.A; **Definitions of Tool Integration for Environments**, IEEE Software, 9(2):29–3, 1992

IEEE, 1990, Std 610.12 - **IEEE Standard Glossary of Software Engineering Terminology**, Institute of Electrical and Electronics Engineers (1990).

IEEE. SWEBOK - **Guide to the Software Engineering Body of Knowledge**. IEEE Computer Society, 2004.

ISO 10007, **Quality Management - Guidelines for Configuration Management**, International Organization for Standardization (2003).

IZZA, S. **Integration of industrial information systems from syntactic to semantic integration approaches**. Enterprise Information Systems, Vol. 3, No. 1, Fevereiro 2009. 1-57.

IZZA, S.; VICENT, L., BURLAT, P.; **Ontology-Based Approach for Application Integration**, First International Conference on Interoperability of Enterprise Software and Applications – Interop-ESA´05, Geneva, Switzerland, February 2005.

JASPER, R., Uschold, M.; **A framework for understanding and classifying ontology applications**. IJCAI-99, Ontology Workshop, Stockholm, 1999.

JIN, D.; CORDY, J. R. **A Service-Sharing Methodology for Integrating COTS-Based Software Systems**. 5th International Conference on COTS-Based Software Systems, Orlando, Florida, 2006.

LEON, A., **A Guide to Software Configuration Management Norwood**, MA, Artech House Publishers (2000).

MARTINS, A.F.: **Construção de Ontologias de Tarefa e sua Reutilização na Engenharia de Requisitos**, Dissertação de Mestrado, Universidade Federal do Espírito Santo (UFES), Vitória (2009)

MARTINS, A.F.: **Construção de Ontologias de Tarefa e sua Reutilização na Engenharia de Requisitos**, Dissertação de Mestrado, Universidade Federal do Espírito Santo (UFES), Vitória (2009)

MIL-HDBK-61A, **Military Handbook, Configuration Management Guidance**, Department of Defense of United States of America, 1997

NOY, N. S., **Semantic Integration: A Survey Of Ontology-Based Approaches**, ACM SIGMOD Record, vol. 33, nº4, p.65-70, 2004

NUNES, V.B. **Integrando Gerência de Configuração de Software**, Documentação e Gerência de Conhecimento em um Ambiente de Desenvolvimento de Software, Mestrado em Informática, UFES, Abril 2005.

PARK, J.; RAM, S.; **Information Systems Interoperability: What lies Beneath?**, ACM Transactions on Information Systems, vol. 22, pg. 595-632, 2004

POKRAEV, S. et al. **Semantic Service Modeling: Enabling System Interoperability**. Proc. Int'l Conf. on Interoperability for Enterprise Software and Applications (I-ESA'06), Bordeaux, France, Março 2006.

POKRAEV, S. V.; **Model-driven Semantic Integration of Service-Oriented Applications**, PhD thesis, University of Twente, 2009

PRESSMAN, Roger. **Software Engineering: A Practitioner's Approach**, 6ªedição, Mc Graw Hill, 2005.

TEKTONIDIS, D. . B. A. . O. G. . S. M. ONAR - **An Ontology-based Service Oriented Application Integration Framework**. Proceedings of the 1st International Conference on Interoperability of Enterprise Software and Applications, Geneva, Switzerland, Fevereiro 2005. 65–74.

THOMAS, I.; NEJMEH, B. A.; **Definitions of Tools Integration for Environments**. Published in: Journal IEEE Software Volume 9 Issue 2, March 1992.

TRAC. Disponível em: < http://trac.edgewall.org/ >. Acesso em: 01 ago. 2011.

WACHE, H; VOGELE, T; S., H; **Ontology-Based Integration of Information – An Survey of Existing Approaches**, Workshop of Ontologies and Information Sharing, 2001, pp. 108-117