UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ANA CHRISTINA DE OLIVEIRA BRINGUENTE

REENGENHARIA DE UMA ONTOLOGIA DE PROCESSO DE SOFTWARE E SEU USO PARA A INTEGRAÇÃO DE FERRAMENTAS DE APOIO AO PLANEJAMENTO DE PROJETOS

VITÓRIA
2011

ANA CHRISTINA DE OLIVEIRA BRINGUENTE

REENGENHARIA DE UMA ONTOLOGIA DE PROCESSO DE SOFTWARE E SEU USO PARA A INTEGRAÇÃO DE FERRAMENTAS DE APOIO AO PLANEJAMENTO DE PROJETOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.
Orientador: Prof. Dr. Ricardo de Almeida Falbo
Co-orientador: Prof. Dr. Giancarlo Guizzardi

VITÓRIA
2011

ANA CHRISTINA DE OLIVEIRA BRINGUENTE

REENGENHARIA DE UMA ONTOLOGIA DE PROCESSO DE SOFTWARE E SEU USO PARA A INTEGRAÇÃO DE FERRAMENTAS DE APOIO AO PLANEJAMENTO DE PROJETOS

COMISSÃO EXAMINADORA

Prof. Dr. Ricardo de Almeida Falbo
Departamento de Informática - UFES
Orientador

Prof. Dr. Giancarlo Guizzardi
Departamento de Informática – UFES
Co-orientador

Profª. Drª. Fernanda Baião
Departamento de Informática Aplicada – UNIRIO

Prof. Dr. Monalessa Perini Barcellos
Departamento de Informática – UFES

Vitória, 23 de agosto de 2011.

Ao Sinhozinho, Sinhazinha, Entojo e Pobre...

AGRADECIMENTOS

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

RESUMO

Com o crescimento do interesse na área de integração entre sistemas de software, surgiram abordagens que visam tratar este problema. De maneira geral, a integração de sistemas pode ocorrer em quatro níveis: de hardware, de plataforma, sintático e semântico. No nível semântico, foco deste trabalho, durante o processo de integração, o significado dos componentes envolvidos deve ser o mais claro possível, ou seja, o significado pretendido dos conceitos no esquema de dados, nas assinaturas das operações e dos serviços deve ser explicitado. Neste contexto, uma ontologia de domínio pode ser utilizada para definir uma representação explícita dessa conceituação compartilhada e ser usada como referência durante a integração. Este trabalho aplicou a abordagem OBA-SI, uma abordagem de integração semântica baseada em ontologia, para integrar na camada de dados ferramentas que apoiam o planejamento, controle e acompanhamento de projeto de software. Durante o processo de integração, foi utilizada uma ontologia de processo de software, a SPO (*Software Process Ontology*) para adicionar semântica aos conceitos das ferramentas envolvidas nesse processo. Para servir adequadamente como um modelo de referência, a SPO passou por um processo de reengenharia baseada na UFO (*Unified Foundational Ontology*), uma ontologia de fundamentação.

ABSTRACT

The increasing interest in the area of integration between software systems has promoted the emergence of approaches to address this issue. In general, systems integration can occur at four levels, namely hardware, platform, syntactic and semantic levels. At the semantic level, which is the focus of this work, the meaning of the involved components must be as clear as possible during the integration process. In other words, the intended meaning of concepts in the data schema, operations signatures and services should be made explicit. In this context, a domain ontology can be used to define an explicit representation of this shared conceptualization and as reference during the integration process. This work applies the approach OBA-SI, which is an approach of ontology-based semantic integration, to integrate the data layer of the tools that support the software project planning, control and tracking. During the integration process, a software process ontology (SPO) is used to add semantics to the concepts of the tools involved in this process. In order to suitably serve as a reference model, the SPO has gone through a reengineering process based on UFO (Unified Foundational Ontology).

LISTA DE FIGURAS

Figura 1 **–** Conceitos disjuntos.....20
Figura 2 – Interseção de Conceitos 20
Figura 3- Símbolos iguais que representam conceitos contidos 21
Figura 4 - Símbolos distintos com conceitos equivalentes.....21
Figura 5 – Símbolos distintos com conceitos sobrepostos 22
Figura 6 – Símbolos distintos com conceitos contidos 22
Figura 7 - Relacionamentos entre modelos de OBA-SI 26
Figura 8 - Mapeamentos verticais e horizontais 28
Figura 9 – UFO-A: Individual e Universal 38
Figura 10 – UFO-A: Substantial e Moment 38
Figura 11 – UFO-A: Sortal 39
Figura 12 – UFO-A: Tipos de moments 42
Figura 13 – UFO-A: Tipos de Relações 42
Figura 14 – Visão geral dos conceitos da UFO-A.....44
Figura 15 - UFO-B: Evento 46
Figura 16 - UFO-C: Distinção entre Agente e Objeto 47
Figura 17 – UFO-C: Intentional Moment 48
Figura 18 - Tipos de comprometimento 49
Figura 19 - UFO-C: Delegação.....50
Figura 20 - UFO-C: Ação.....51
Figura 21 – Fragmento Ontologia de Processo de Software 54
Figura 22 – Reengenharia da Ontologia de Processo de Software 55
Figura 23 – Definição de Processo Padrão 58
Figura 24 – Definição do Processo de Projeto. 59
Figura 25 - Definição do Processo de Projeto baseado em um Processo Padrão.....60
Figura 26 – Processo de Projeto Programado 62
Figura 27 - Recursos definidos para as atividades.....63
Figura 28 – Alocação de um recurso humano em um projeto.....64
Figura 29 – Alocação de Recurso Humano em uma Atividade.....65
Figura 30 – Ocorrência de Processo e de Atividade.....66
Figura 31 – Participações de Recursos na ocorrência de atividades. 67
Figura 32 – Versão atual da Ontologia de Processo de Software.....69
Figura 33 – Modelo conceitual Def-Pro – *Processo Padrão* 74
Figura 34 - Modelo conceitual Def-Pro - *Controle Processo*.....76
Figura 35 – Modelo conceitual AlocaODE.....77
Figura 36 – Modelo conceitual ControlPro 79
Figura 37 – Modelo Conceitual Endeavour 80
Figura 38 - Modelo de Integração – Definição de Processo Padrão 85

Figura 39 - Modelo de Integração – Definição de Processo de Projeto.....86
Figura 40 - Modelo de Integração – Alocação de Recurso.....87
Figura 41 – Modelo de Integração – Execução da Atividade.....88
Figura 42 - Modelo de Integração.....89

SUMÁRIO

1 INTRODUÇÃO 10
1.1 OBJETIVOS 12
1.2 HISTÓRICO DO DESENVOLVIMENTO DO TRABALHO 13
1.3 ORGANIZAÇÃO DO TRABALHO 14
2 INTEGRAÇÃO SEMÂNTICA 15
2.1 INTEROPERABILIDADE 16
2.1.1 Níveis de integração 18
2.2 ABORDAGENS DE INTEGRAÇÃO SEMÂNTICA 23
2.2.1 OBA-SI: Ontology-Based Approach For Semantic Integration 24
2.3 PLANEJAMENTO DE PROJETO E PROCESSO DE SOFTWARE 30
2.3.1 Ferramentas de Apoio ao Planejamento de Processo de Software 32
2.4 CONSIDERAÇÕES FINAIS 34
3 A ONTOLOGIA DE FUNDAMENTAÇÃO UNIFICADA 36
3.1 UFO-A 37
3.2 UFO-B 45
3.3 UFO-C 46
4 REENGENHARIA BASEADA NA UFO DA ONTOLOGIA DE PROCESSO DE SOFTWARE 52
4.1 ONTOLOGIA DE PROCESSO 53
4.2 REENGENHARIA DA ONTOLOGIA DE PROCESSO 56
4.2.1 Definição do Processo Padrão 56
4.2.2 Definição do Processo de Projeto e Agendamento 58
4.2.3 Tipos de Recurso 62
4.2.4 Alocação de Recursos Humanos 63
4.2.5 Ocorrência de Atividade 65
4.3 CONSIDERAÇÕES FINAIS 68
5 INTEGRANDO CONCEITUALMENTE FERRAMENTAS DE APOIO AO PLANEJAMENTO DE PROJETOS DE SOFTWARE 71
5.1 MODELOS CONCEITUAIS ESTRUTURAIS DAS FERRAMENTAS 73
5.1.1 Modelo Conceitual Estrutural de Def-Pro 74
5.1.2 Modelo Conceitual Estrutural de AlocaODE 77
5.1.3 Modelo Conceitual Estrutural de ControlPro 78
5.1.4 Modelo Conceitual Estrutural de Endeavour 79
5.2 MAPEAMENTOS VERTICAIS 81
5.3 MODELO DE INTEGRAÇÃO 84
5.4 CONSIDERAÇÕES FINAIS 90
6 CONCLUSÃO 92
6.1 TRABALHOS FUTUROS 94
REFERÊNCIAS 97

1 INTRODUÇÃO

O gerenciamento de projetos envolve a aplicação de conhecimento, habilidades, ferramentas e técnicas às atividades do projeto, a fim de atender aos seus requisitos. Um projeto é um empreendimento integrado e requer que cada um de seus processos seja alinhado e conectado de forma apropriada com os outros processos para facilitar sua coordenação. As ações adotadas durante um processo em geral afetam, além do processo em si, outros processos relacionados. O gerenciamento de projetos bem sucedido inclui gerenciar ativamente essas interações para cumprir os requisitos das partes interessadas (PMI, 2008).

Entre os processos de um projeto, podemos destacar os processos de Planejamento de Projeto e Acompanhamento e Controle de Projeto. Durante o planejamento do projeto, o processo do projeto deve ser definido, o cronograma estabelecido, bem como um plano de recursos humanos. Baseado neste planejamento, com o projeto em andamento, é necessário acompanhar e controlar o projeto, verificando se o planejamento está sendo cumprido, se ele precisa ser revisto e o que deve ser feito para corrigir eventuais desvios.

No contexto de projetos de software, durante o planejamento do projeto, processos de software padrão são utilizados como base para a definição de processos de projeto. Uma vez que os processos de projeto estão definidos, é necessário definir a equipe do projeto, quais os papéis que um recurso humano vai desempenhar em uma determinada atividade, quais os tipos de recursos requeridos e assim por diante. Durante a execução do projeto, as atividades devem ser controladas e monitoradas, registrando seu tempo de duração e a participação de cada recurso humano na atividade, bem como o esforço por ele despendido.

Geralmente, diferentes ferramentas são usadas para apoiar tais tarefas. Considerando que essas tarefas são iterativas e inter-relacionadas, idealmente essas ferramentas devem interoperar para apoiar efetivamente os processos de gerência de projeto.

De maneira geral, a integração de sistemas pode se dar em três camadas diferentes (IZZA, 2009): a integração na camada de dados lida com a troca de dados entre os aplicativos; a integração na camada de mensagens ou serviços trata da troca de mensagens e serviços entre os sistemas; por fim, a integração na camada de processo é responsável por tratar o fluxo de mensagens, regras de execução e definir o processo de execução global.

A integração pode acontecer, ainda, em diferentes níveis. O nível de hardware envolve diferentes computadores, redes de computadores, etc. O nível de plataforma trata de sistemas operacionais, sistemas de bancos de dados, etc. O nível sintático refere-se a como os modelos de dados e as assinaturas das operações são escritos. Por fim, o nível semântico abrange o significado pretendido dos conceitos no esquema de dados e nas assinaturas das operações. Cada um dos níveis é construído sobre o anterior, de modo que não faz sentido falar em integração no nível sintático se não existe integração no nível de plataforma (IZZA, 2009).

Para que as ferramentas trabalhem em conjunto de forma satisfatória, é necessário que elas compartilhem uma conceituação sobre o domínio de processos de software. Neste contexto, uma ontologia de domínio pode ser utilizada para definir uma representação explícita dessa conceituação compartilhada.

Uma ontologia pode ser definida como uma especificação de um vocabulário para representação de um domínio compartilhado de discurso (GUIZZARDI, 2005). Em outras palavras, uma ontologia é um modelo de domínio, composto por um conjunto de conceitos e relações. Uma ontologia é útil para fornecer uma compreensão clara sobre um determinado domínio.

Contudo, para servir adequadamente como um modelo de referência, uma ontologia de domínio deve ser construída usando uma abordagem que leve em conta explicitamente conceitos de fundamentação. Uma ontologia de domínio de referência tem o propósito de representar um domínio de discurso com clareza, veracidade e expressividade, sem considerar requisitos computacionais. O principal objetivo desse modelo de referência é ajudar modeladores a externalizar seu conhecimento sobre o domínio, tornando seus comprometimentos ontológicos explícitos, de modo a apoiar a negociação de significado e melhorar a comunicação,

aprendizado e resolução de problemas no domínio. Para que atinja tais objetivos, uma ontologia de referência deve ser construída levando em conta uma ontologia de fundamentação (GUIZZARDI et al., 2008).

1.1 OBJETIVOS

Levando em consideração as motivações apresentadas, o objetivo geral deste trabalho é integrar conceitual e semanticamente ferramentas de apoio ao processo de Gerência de Projetos de Software na camada de dados. Para a integração será utilizada, então, uma ontologia de referência do domínio de processos de software que considera as distinções ontológicas feitas em uma ontologia de fundamentação, a UFO (*Unified Foundational Ontology*) (GUIZZARDI, 2005) (GUIZZARDI et al., 2008).

Esse objetivo geral pode ser detalhado nos seguintes objetivos específicos:

- **Fazer a reengenharia da Ontologia de Processo de Software baseado na UFO:**

  Uma vez que já existe uma ontologia de processos de software desenvolvida (BERTOLLO, 2006) e parcialmente alinhada aos conceitos da UFO (GUIZZARDI et al., 2008), é um objetivo específico desta dissertação avançar na reengenharia dessa ontologia. Essa reengenharia tem o intuito de explicitar os compromissos ontológicos da ontologia de processos de software.

- **Aplicar uma abordagem de integração semântica para integrar conceitualmente as ferramentas escolhidas.**

  Neste trabalho, tem-se como ponto de partida um conjunto inicial de ferramentas a serem integradas: as ferramentas de apoio ao processo de Gerência de Projetos do ambiente ODE (*Ontology-based software Development Environment*) (FALBO et al., 2005). Assim, é necessário selecionar uma ferramenta de Gerência de Projeto que atenda a requisitos tidos como necessários para que o processo de Gerência de Projetos de

Software seja atendido após a integração. Para a integração, foi aplicada OBA-SI (CALHAU; FALBO, 2010), uma abordagem de integração semântica de sistemas de informação.

1.2 HISTÓRICO DO DESENVOLVIMENTO DO TRABALHO

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

Inicialmente, foi feito um levantamento bibliográfico relacionado ao tema Interoperabilidade Semântica. O intuito deste estudo foi levantar como o problema de interoperabilidade está sendo tratado pela comunidade científica. Durante este estudo notou-se que, apesar de existir um consenso sobre a importância do uso de ontologias para a integração semântica, pouco se relata sobre a qualidade dessas ontologias e como elas, de fato, devem ser usadas. Sendo assim, optou-se por, não apenas utilizar uma ontologia no processo de integração, mas utilizar uma ontologia de domínio que fosse elaborada com base em uma ontologia de fundamentação. Uma vez que este trabalho visa discutir a integração no nível conceitual, foi de extrema importância utilizar uma abordagem de integração semântica que enfoque os aspectos não computacionais da integração.

Como as ferramentas a serem integradas são ferramentas que apoiam a gerência de projetos de software, foi feito um estudo sobre este domínio. O estudo possibilitou reconhecer dentro do ambiente ODE quais eram os pontos que não eram devidamente apoiados pelas ferramentas e, consequentemente, quais os requisitos que a ferramenta externa deveria atender.

Uma vez que as ferramentas seriam integradas no nível conceitual, a ontologia utilizada como interlíngua entre os modelos conceituais das ferramentas deveria ser construída de maneira a maximizar a expressividade na captura de aspetos fundamentais do domínio subjacente e tornar explícitos os compromissos ontológicos. Para tal, a ontologia de processo de software passou por um processo de reengenharia baseado na UFO para atender a tais requisitos.

Com a ontologia de referência e as ferramentas a serem integradas definidas, deu-se início ao processo de integração semântica de acordo com a abordagem OBA-SI.

Parte dos resultados obtidos neste trabalho foram descritos em um artigo científico, o qual foi apresentado no XXVI Simpósio Brasileiro de Banco de Dados (SBBD 2011) e publicado no *Journal of Information and Data Management* – JIDM.

1.3 ORGANIZAÇÃO DO TRABALHO

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

- Capítulo 2 – Integração Semântica: apresenta uma revisão da literatura sobre Interoperabilidade Semântica e sobre o domínio de estudo desta dissertação: a Gerência de Projetos de Software.
- Capítulo 3 – A Ontologia de Fundamentação Unificada: apresenta a ontologia de fundamentação UFO, fazendo uma breve explicação dos conceitos necessários para este trabalho.
- Capítulo 4 – Reengenharia Baseada na UFO da Ontologia de Processo de Software: apresenta a reengenharia da ontologia de processo de Software baseada na UFO.
- Capítulo 5 – Integrando Conceitualmente Ferramentas de Apoio ao Planejamento de Projetos de Software: apresenta a iniciativa de integração semântica no nível conceitual, utilizando OBA-SI, realizada visando integrar as ferramentas do ambiente ODE com a ferramenta Endeavour.
- Capítulo 6 – Considerações Finais: apresenta as conclusões do trabalho, as contribuições, dificuldades e propostas de trabalhos futuros.

2 INTEGRAÇÃO SEMÂNTICA

Atualmente, cresce a necessidade dos sistemas de informação interoperarem de forma a apoiarem os objetivos das organizações. Podemos citar como uma das causas o crescente número de fusões entre empresas. Quando isso ocorre, sistemas que antes trabalhavam de forma independente passam a ter que trabalhar integrados para apoiar os processos de negócio satisfatoriamente. Podemos citar, ainda, ferramentas que executam tarefas complementares dentro de um processo como, por exemplo, ferramentas que apoiam diferentes fases do processo de desenvolvimento de software. Durante esse processo é comum que artefatos produzidos em uma fase sejam consumidos por outra e, portanto, é ideal que essas ferramentas sejam capazes de se comunicar entre si, sem a necessidade de intervenção manual.

Na busca por soluções para o problema apresentado acima, tem-se despendido muito tempo e dinheiro. Em 2002, estimava-se que as grandes organizações gastavam cerca de 40% do seu orçamento destinado à tecnologia com este problema e que isto aumentaria com o passar dos anos (SERAIN, 2002). Isto implicou no crescimento do interesse na área de interoperabilidade e, consequentemente, surgiram abordagens que visam solucionar este problema. As abordagens diferenciam-se entre si, entre outros fatores, pelo nível de interoperabilidade que se deseja entre os componentes de software.

Este capítulo aborda o problema de Interoperabilidade Semântica, bem como as possíveis abordagens para a sua solução. Ele está estruturado da seguinte forma: a Seção 2.1 apresenta o problema da falta de interoperabilidade; a Seção 2.2 apresenta a abordagem de integração semântica adotada neste trabalho; a Seção 2.3 apresenta um cenário de necessidade de integração de ferramentas para apoiar o planejamento de projetos de software; e a Seção 2.4, por fim, apresenta as considerações finais do capítulo.

2.1 INTEROPERABILIDADE

Interoperabilidade em um contexto amplo pode ser definida como a habilidade de operar em conjunto ou, mais especificamente no contexto de sistemas computacionais, como a capacidade de trocar dados e serviços entre aplicativos ou componentes de aplicativos (VERNADAT 1996, WEGNER, 1996). Esses aplicativos, por sua vez, podem ser classificados de acordo com a heterogeneidade existente entre eles (IZZA et al. 2005). Heterogêneo significa que cada aplicação implementa seu próprio modelo de dados e de processo. Quanto maior é a diferença entre esses modelos, sejam por conflitos sintáticos ou semânticos, mais difícil e complexo é torná-los interoperáveis (IZZA, 2009).

Há diversas dimensões de integração que ajudam a caracterizar a noção de integração e que podem ser usadas para classificar as abordagens de integração. Izza (2009) propõe quatro dimensões:

- Escopo da integração: A integração pode se dar dentro da organização, envolvendo apenas aplicações da mesma, ou entre organizações, conectando aplicações de diferentes parceiros.
- Pontos de vista da integração: A integração pode focar o ponto de vista do usuário, do projetista ou do programador. O ponto de vista do usuário preocupa-se as diferentes visões que especialistas de domínio e usuários de negócio têm. O ponto de vista do projetista enfoca os diferentes modelos usados durante o projeto de sistemas de informação (visão conceitual). Finalmente, o ponto de vista do programador refere-se à implementação dos sistemas.
- Camadas de integração: A integração de aplicativos pode ocorrer em três camadas: camada de dados, camada de mensagem ou serviço e camada de processo. A camada de dados lida com a troca de dados entre vários repositórios. Neste ponto, uma aplicação manipula os dados de outra aplicação diretamente no banco de dados, através de sua interface nativa, ignorando a lógica da aplicação. A integração na camada de mensagens

ou serviços ocorre quando duas ou mais aplicações trocam mensagens entre si. Por fim, a última camada, a camada de processo constitui a integração mais complexa, pois ela enxerga uma organização como uma série de processos relacionados e os sistemas responsáveis pela execução desse processo não podem ser vistos como ilhas de informações. Dessa forma a integração nessa camada é responsável por tratar o fluxo de mensagens, regras de execução e definir o processo de execução global.

- Níveis de integração: Como dito anteriormente, a integração pode acontecer em diferentes níveis, a saber: nível de hardware, de plataforma, sintático e semântico. O nível de hardware envolve a integração de diferentes computadores, redes de computadores etc. Neste nível é utilizado, por exemplo, protocolos de redes para que duas ou mais redes possam se comunicar. O nível de plataforma trata de sistemas operacionais, sistemas de bancos de dados etc. O nível sintático refere-se ao modo como modelos de dados e assinaturas de operações são escritos. Por fim, o nível semântico abrange o significado pretendido dos conceitos no esquema de dados e nas assinaturas das operações. Cada um dos níveis é construído sobre o anterior. Dessa forma, não faz sentido falar em integração no nível sintático se não existe integração no nível de plataforma .

Nesta dissertação, trabalha-se a integração segundo o ponto de vista do projetista, na camada de dados e no nível semântico. Interesse especial está nesta última dimensão, a qual é discutida com mais detalhes a seguir.

2.1.1 Níveis de integração

Conforme discutido anteriormente, a integração pode acontecer em diferentes níveis, a saber: nível de hardware, plataforma, sintático e semântico. Os de maior interesse para este trabalho são os níveis sintático e semântico, os quais são discutidos mais detalhadamente nesta seção.

2.1.1.1 Integração Sintática

A sintaxe define uma lista de palavras válidas em uma determinada linguagem e as regras que governam como essas palavras serão combinadas para que formem sentenças válidas (POKRAEV, 2009). Dois ou mais sistemas estão integrados sintaticamente quando eles são capazes de trocar dados e serviços. Para tal, é essencial a especificação de protocolos de comunicação que descrevam como esta troca vai ocorrer. Podemos citar XML ou SQL como padrões que proveem interoperabilidade sintática.

Quando dois sistemas estão integrados sintaticamente, é possível detectar erros sintáticos como, por exemplo, chamar um serviço passando um parâmetro de tipo incompatível com o esperado. Contudo, abordagens sintáticas não preveem erros semânticos, onde a sintaxe está correta, contudo, a semântica não. Estudos apontavam em 2005 que 70% das tentativas de integração que se limitavam a este nível não eram bem sucedidas (HALLER et al, 2005).

Lidar com o problema de integração sintática foge do escopo desta dissertação. Por esta razão nós focaremos no problema de interoperabilidade semântica

2.1.1.2 Integração Semântica

Semântica refere-se ao significado dos construtores sintáticos em uma determinada linguagem. Para exemplificar o que é semântica no contexto de sistemas computacionais, podemos pensar em um método escrito em Java com a seguinte assinatura:

```
public Double CalculaComprimentoCirculo (Double x);
```

Este método recebe como parâmetro o raio de um círculo e calcula o comprimento do mesmo. Suponha um programador que utilizará este método e que conheça a linguagem. Ele será capaz de deduzir, apenas olhando, que o método calcula o comprimento de um círculo. Contudo, uma vez que a semântica do parâmetro x não está explícita, caberá ao programador atribuir significado a esse parâmetro. Tal situação pode trazer consequências indesejadas, pois o programador pode entender que x representa o diâmetro do círculo e usar incorretamente o método. Desta forma, mesmo conhecendo a sintaxe da linguagem, ocorreria um erro, pois a semântica do parâmetro não está explícita.

Neste contexto, podemos definir semântica como sendo o mapeamento de um objeto de um sistema de informação para o objeto no mundo real que ele representa (POKRAEV, 2009). Desta forma, interoperabilidade semântica é a capacidade de dois ou mais sistemas heterogêneos e distribuídos trabalharem em conjunto, compartilhando as informações entre eles com entendimento comum de seu significado (BURANARACH, 2001).

De acordo com Pokraev (2009), ao integrar dois sistemas, podemos ter os seguintes problemas de interoperabilidade semântica:

i. *Diferentes sistemas usam o mesmo símbolo para representar conceitos disjuntos*.

   Por exemplo, um sistema usa o símbolo “Amazonas” para representar o conceito “estado venezuelano”, enquanto outro sistema

usa o mesmo símbolo para representar o conceito “estado brasileiro” (Figura 1).

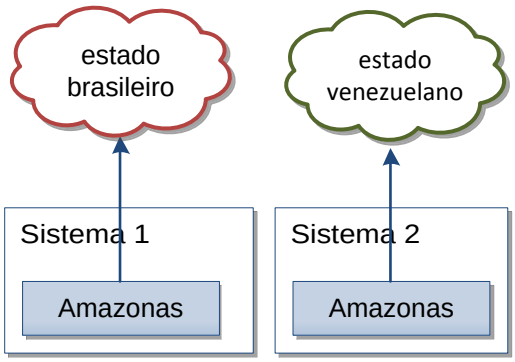

Figura 1 **–** Conceitos disjuntos

ii. *Diferentes sistemas usam o mesmo símbolo para representar conceitos que se sobrepõe.*

Por exemplo, um sistema usa o símbolo “animal” para representar o conceito “animal mamífero”, enquanto outro sistema usa o mesmo símbolo para representar o conceito “animal carnívoro”. Uma vez que nem todos mamíferos são carnívoros e vice-versa, em alguns casos, o símbolo “animal” pode se referir a entidades diferentes no mundo real (Figura 2).

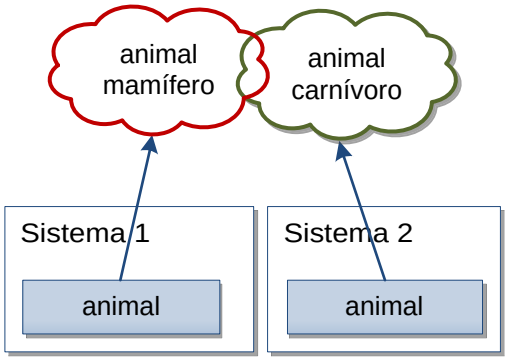

Figura 2 – Interseção de Conceitos

iii. *Sistemas diferentes usam o mesmo símbolo para representar conceitos com significados mais gerais (ou específicos).*

Por exemplo, um sistema usa o símbolo “Floresta Amazônica” para representar a “Amazônia Legal”, enquanto outro sistema usa o mesmo símbolo para representar o conceito “Floresta Amazônica” (a floresta está presente em nove nações) (Figura 3).

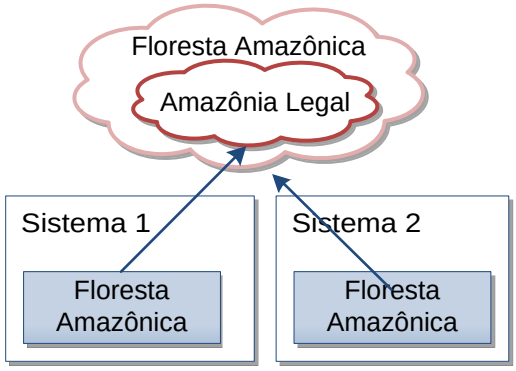

Figura 3- Símbolos iguais que representam conceitos contidos

iv. *Sistemas diferentes usam símbolos diferentes para representar o mesmo conceito.*

Por exemplo, um sistema usa o símbolo “comprador” para representar o conceito “alguém que adquire bens ou serviços” enquanto outro sistema utiliza o símbolo “cliente” para representar o mesmo conceito (Figura 4).

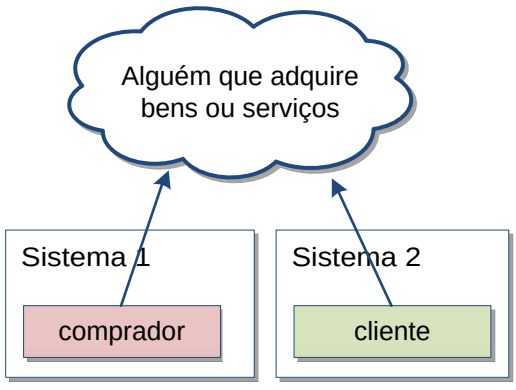

Figura 4 - Símbolos distintos com conceitos equivalentes

v. *Diferentes sistemas usam símbolos diferentes para representar conceitos com sobreposição de significados*.

Por exemplo, um sistema usa o símbolo “empregado” para representar o conceito “alguém que trabalha na empresa”, enquanto outro sistema usa o símbolo “cliente” para representar o conceito “alguém que adquire um produto da empresa” (Figura 5).

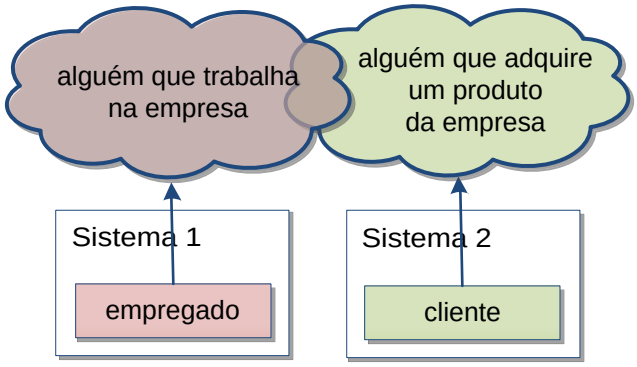

Figura 5 – Símbolos distintos com conceitos sobrepostos

vi. *Diferentes sistemas usam símbolos diferentes para representar conceitos com significados mais gerais ou específicos*.

Por exemplo, um sistema pode usar o símbolo “espírito-santense” para representar o conceito “pessoa que nasceu no Espírito Santo”, enquanto outro sistema pode utilizar o símbolo “brasileiro” para representar o conceito “pessoa que nasceu no Brasil” (Figura 6).

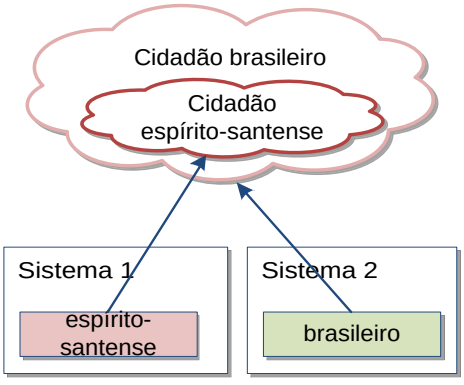

Figura 6 – Símbolos distintos com conceitos contidos

As incompatibilidades semânticas foram apresentadas para pontuar quais os tipos de problemas semânticos que podem surgir quando dois sistemas

computacionais estão sendo integrados. É importante salientar que esses problemas acontecem não apenas com os conceitos, mas também com os relacionamentos que existem entre os conceitos.

2.2 ABORDAGENS DE INTEGRAÇÃO SEMÂNTICA

Para cada um dos níveis de interoperabilidade citados anteriormente, existem diferentes abordagens utilizadas para que ela seja alcançada. Para o nível de integração semântica, que é o foco desta dissertação, as abordagens baseiam-se na representação e exploração da semântica dos sistemas de informação durante o processo de integração. Algumas delas apoiam tanto a integração na camada de dados quanto de serviços e processos (IZZA; VINCENT; BURLAT, 2005; BOURAS et al. 2006; POKRAEV et al. 2006). Outras abordagens cobrem apenas determinadas camadas (TEKTONIDIS et al., 2005, JIN; CORDY, 2006). Existem abordagens que usam serviços web (*web services*) e outras que focam no uso de uma linguagem de ontologia.

Entretanto, essas abordagens, em sua maioria, focam nos aspectos tecnológicos da solução de integração. Calhau e Falbo (2010), por sua vez, defendem que a integração semântica deve ser independente de como a integração será implementada. Assim como no processo de desenvolvimento de software, o processo de integração deve iniciar no nível conceitual. Apenas depois de considerar os requisitos de integração e construir um modelo conceitual da integração, os aspectos tecnológicos devem ser considerados.

Este trabalho usa a visão proposta em OBA-SI (CALHAU; FALBO, 2010), ou seja, focar nas questões conceituais da integração semântica. A seguir detalharemos a abordagem OBA-SI, com foco na integração semântica na camada de dados.

2.2.1 OBA-SI: *Ontology-Based Approach For Semantic Integration*

A Abordagem baseada em Ontologias para a Integração Semântica (*Ontology-Based Approach for Semantic Integration* – OBA-SI), analogamente ao processo de desenvolvimento de software, defende o uso de um processo composto das seguintes atividades: análise, projeto, implementação e testes. OBA-SI, no entanto, está centrada na primeira fase, a análise da integração, dando apenas algumas orientações para a fase de projeto. De acordo com OBA-SI, a semântica deve ser definida no início do processo de integração, na fase de análise. Isto porque as ferramentas devem concordar semanticamente antes da concepção e implementação de uma solução específica. Desta forma, OBA-SI procura ser independente da tecnologia e de uma solução de integração específica. Durante a fase de análise, mapeamentos semânticos são feitos entre os modelos conceituais das ferramentas que estão sendo integradas, usando ontologias para atribuir significado.

Na fase de análise, três artefatos são muito importantes: (i) uma ontologia de domínio de referência que descreve o domínio de integração, (ii) os modelos conceituais dos aplicativos que estão sendo integrados (modelos estruturais e comportamentais) e (iii) o modelo de processo de negócio que está sendo apoiado pelas ferramentas. A ontologia de referência, como dito anteriormente, é uma especificação independente da solução utilizada para descrever de forma clara e precisa as entidades do domínio. Ela é usada como um modelo de referência para atribuir semântica durante a análise da integração. Os modelos conceituais dos aplicativos que estão sendo integrados podem ser conhecidos a priori (por exemplo, as aplicações foram desenvolvidas pela organização e os modelos conceituais estão disponíveis), ou devem ser escavados por meio de técnicas de engenharia reversa. O modelo de processo de negócios, por sua vez, deve descrever os processos da organização envolvidos no cenário de integração, indicando, dentre outros, as atividades, os recursos humanos, as ferramentas de apoio e as entradas e saídas de cada fase do processo.

Esses modelos se concentram em diferentes aspectos da integração. A ontologia de domínio de referência, juntamente com os modelos conceituais das ferramentas, são utilizados para estabelecer a semântica das camadas de dados e de serviços. O modelo do processo de negócios é usado principalmente para tratar da integração de processos, embora também seja útil na integração dos serviços.

Como resultado da fase de análise, um modelo de integração é produzido, capturando a imagem global do cenário de integração. O modelo de integração especifica: (i) os conceitos a serem representados, (ii) como o conjunto de ferramentas integradas vai se comportar como um todo e (iii) quais as atividades do processo que serão atendidas pelas ferramentas integradas. O modelo de integração é construído com base nos requisitos de integração e os três artefatos mencionados anteriormente. De fato, o objetivo principal do modelo de integração é estabelecer mapeamentos entre os modelos das ferramentas que estão sendo integradas. Esses mapeamentos são discutidos a seguir.

a) Mapeamentos entre modelos

O objetivo de um mapeamento é relacionar os elementos de modelos distintos que são conceitualmente equivalentes. Mapeamentos ocorrem principalmente em dois níveis: vertical e horizontal.

Mapeamentos verticais (MVs) são responsáveis pela atribuição de significado para os elementos do modelo com base na ontologia de referência. Eles são independentes da solução de integração. Seu objetivo é atribuir significado para as entidades das ferramentas, relacionando-as com as entidades na ontologia de referência. Uma vez que os MVs relacionam os elementos do modelo com os conceitos da ontologia, eles são independentes do cenário de integração e podem ser usados em várias iniciativas de integração.

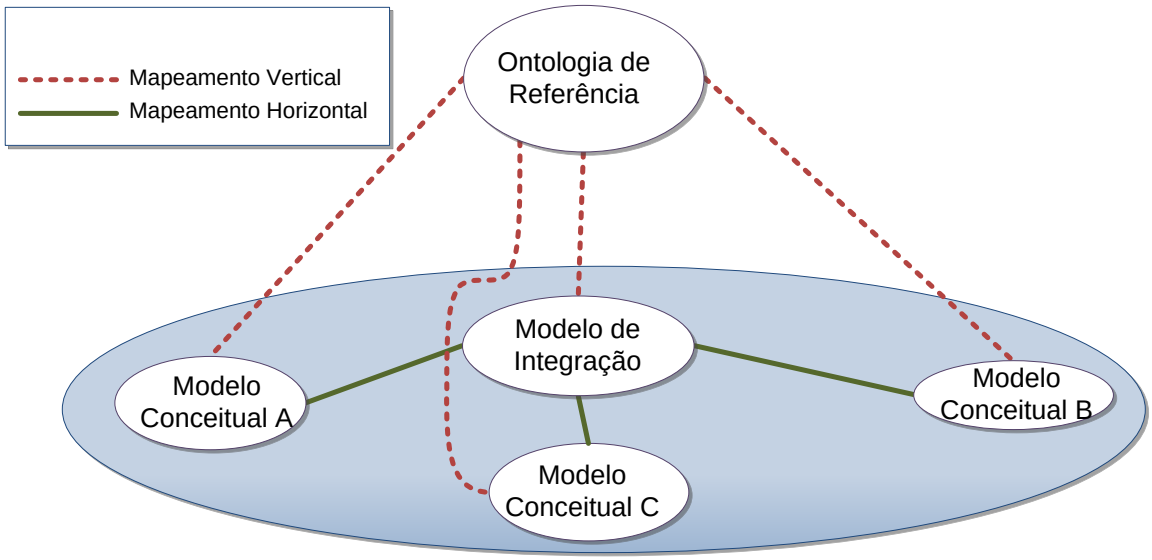

Figura 7 - Relacionamentos entre modelos de OBA-SI

Mapeamentos horizontais (MHs), por outro lado, ocorrem entre os elementos dos modelos, focando no cenário de integração. Ao estabelecer os MHs, os elementos dos modelos das ferramentas são mapeados para elementos do modelo de integração, como ilustrado na Figura 7. Os MHs definem como as ferramentas serão vistas pelo mediador de integração e como a interação entre eles ocorrerá. Idealmente, MHs devem basear-se nos MVs. No entanto, uma vez que MHs focam em um cenário de integração específico, eles nem sempre se baseiam no MVs.

b) A Fase de Análise da Integração

A fase de análise de OBA-SI abrange as seguintes atividades: (i) especificação de requisitos de integração, (ii) recuperação de modelos conceituais; (iii) atribuição semântica aos modelos conceituais estruturais das ferramentas e (iv) a modelagem da integração.

Durante a atividade de especificação de requisitos da integração, os requisitos e os objetivos da integração devem ser definidos. Inicialmente, os objetivos da integração devem ser estabelecidos. Com base neles, os requisitos podem ser levantados. O escopo da integração deve ser definido, apontando quais atividades do processo de negócio serão apoiadas e quais aplicações devem ser integradas para tal. As atividades do processo de negócio e as aplicações a serem integradas

compõem o cenário de integração. Caso os modelos de processo de negócio não estejam disponíveis, deverão ser desenvolvidos como parte do projeto de integração.

Uma vez definido o cenário de integração, devem-se obter os modelos conceituais estruturais e comportamentais das ferramentas. Precisamos também selecionar uma ontologia de referência sobre o domínio do cenário de integração que, caso não esteja disponível, deve ser desenvolvida.

O próximo passo é atribuir semântica aos modelos conceituais, definindo os mapeamentos verticais entre seus conceitos e relações e os conceitos e relações da ontologia de referência. Esses mapeamentos devem ser validados por um especialista de domínio.

Uma vez que os modelos estruturais estão semanticamente anotados, a modelagem da integração pode começar. O propósito do modelo de integração é modelar os aspectos estruturais e comportamentais da integração no nível conceitual, considerando os requisitos especificados anteriormente. É bastante semelhante a um modelo conceitual em qualquer processo de desenvolvimento. A principal diferença é que o modelo de integração é construído baseado na ontologia e nos modelos das ferramentas. Desta forma, a diferença semântica entre o modelo de integração e os modelos das ferramentas será menor.

O modelo de integração é a saída principal da fase de análise de OBA-SI. Ele é construído intercalando as abordagens *top-down* e *bottom-up*. Inicialmente, um modelo preliminar é construído com base nos requisitos, no modelo de processo de negócio e na ontologia de referência, em uma abordagem *top-down*. Em seguida, este modelo é refinado, considerando os modelos conceituais das ferramentas a serem integradas, em uma abordagem *bottom-up*. Com uma versão mais refinada do modelo de integração, os MHs devem ser estabelecidos, ou seja, é feito um mapeamento entre os conceitos dos modelos que estão sendo integrados.

A Figura 8 ilustra um cenário de integração, onde dois modelos conceituais estruturais devem ser mapeados para o modelo de integração à luz de uma

ontologia de domínio. Os MVs mapeiam os conceitos das ferramentas e do modelo de integração com a ontologia, como os mapeamentos (b2, o2) e (a2, o2). Os MHs são, então, estabelecidos com base nos MVs, como é o caso dos mapeamentos (a2, i2) e (b2, i2). No entanto, pode haver MHs, tais como (a3, i3) e (b3, i3), que não são baseadas em um MV. Isso pode ocorrer porque a ontologia a referência pode não abranger todos os elementos do modelo considerado pelos aplicativos.

No mapeamento estrutural (camada de dados), inicialmente mapeiam-se os conceitos. Uma vez estabelecido os mapeamentos entre os conceitos, devem-se definir os mapeamentos entre as relações. O procedimento para as relações é análogo, ou seja, MVs e MHs são estabelecidos e as relações correspondentes são incorporadas ao modelo estrutural de integração. No entanto, como um conceito também é definido em termos de suas relações, mapeamentos entre as relações podem influenciar os mapeamentos entre conceitos e, por isso, este é um processo iterativo.

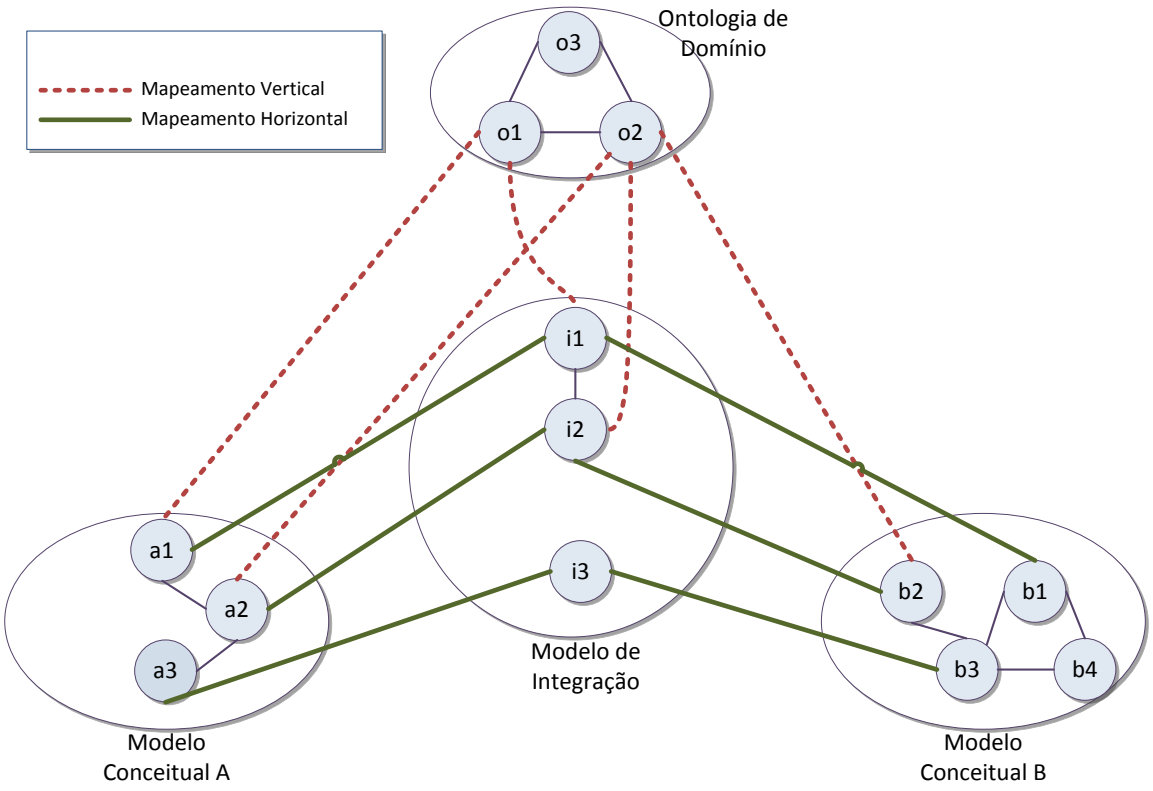

Figura 8 - Mapeamentos verticais e horizontais

c) As Fases de Design e de Implementação

Uma solução de integração pode ser concebida e implementada de várias maneiras. OBA-SI não se compromete com uma solução de integração específica. No entanto, fornece algumas diretrizes a fim de manter a consistência semântica durante o processo de integração. Primeiro de tudo, o projeto deve ser baseado na semântica estabelecida durante a fase de análise. Assim, os principais insumos para a fase de projeto são os modelos conceituais das ferramentas e o modelo de integração. Esses artefatos são independentes dos aspectos computacionais relacionados à solução de integração. No entanto, é natural que o modelo de integração seja alterado para incorporar as decisões relativas à solução de integração.

A integração de ferramentas pode acontecer sem que as mesmas sejam alteradas. Uma forma de integrá-las é construindo mediadores para intermediarem a comunicação entre as ferramentas. Um mediador deve ter uma visão global dos sistemas a serem integrados. Ele é responsável pelo intercâmbio de dados e funcionalidades entre as ferramentas. O modelo de integração fornece as informações para projetá-lo, onde o modelo estrutural capta os conceitos e as relações que serão manipuladas pelo mediador. O modelo comportamental, por sua vez, mostra as atividades do processo de negócio e os serviços das aplicações que serão integrados. Além disso, a orquestração de invocações dos serviços das ferramentas é feita através do mediador, considerando o modelo comportamental da integração. Assim, é importante que o mediador implemente o modelo de integração, a fim de ligar as ferramentas. Finalmente, os mapeamentos horizontais (estruturais e comportamentais) podem ser usados para projetar a comunicação entre as ferramentas e o mediador. A implementação desta comunicação pode ocorrer dentro do mediador ou fora dele, por meio de adaptadores que se conectam as ferramentas para o mediador.

2.3 PLANEJAMENTO DE PROJETO E PROCESSO DE SOFTWARE

De acordo com a ISO 10006:2003, a Gerência de Projetos, em geral, inclui as atividades de planejamento, organização, supervisão e controle de todos os aspectos do projeto, em um processo contínuo para atingir seus objetivos. Esses objetivos são alcançados de forma mais eficiente quando as atividades do projeto e seus recursos são gerenciados como um processo (ISO, 2003). Os Processos de Gerência de Projetos são usados para estabelecer e desenvolver planos de projeto, para avaliar o progresso do projeto em relação aos planos e controlar a execução do projeto até a sua finalização (ISO/IEC, 2008).

Processos são decompostos em partes menores. No contexto da engenharia de software, um processo pode ser decomposto em outros processos (subprocessos) ou em atividades. Um subprocesso é descrito como uma decomposição do processo que satisfaz os critérios para ser um processo, ou seja, tem um propósito, é coeso e é colocado sob a responsabilidade de uma organização, ou de uma parte dela, no ciclo de vida do software. Uma atividade é utilizada quando a unidade decomposta do processo não pode ser qualificada como um processo. Uma atividade pode ser considerada simplesmente como um conjunto de tarefas (ISO/IEC, 2008).

Os processos de projeto devem ser identificados e documentados, e as atividades devem ser realizadas e controladas de acordo com o plano do projeto (ISO, 2003). Processos de projeto de software devem ser definidos considerando as atividades a serem executadas, os recursos necessários, os artefatos de entrada e saída, os procedimentos adotados (métodos, técnicas, modelos de documento, entre outros) e o modelo de ciclo de vida que será usado (FALBO; BERTOLLO, 2009).

A definição do processo que será seguido no decorrer do projeto pode ser feita através da identificação das entradas, saídas e dos objetivos, além da atribuição de autoridade e responsabilidades para os processos, entre outros (ISO/IEC, 2008)., sendo que a organização deve considerar a experiência adquirida no desenvolvimento e uso de seus processos em projetos anteriores.

Embora diferentes projetos requeiram processos com características específicas, é possível estabelecer um conjunto de ativos de processo de software que devem estar presentes em todos os processos de projeto de uma organização. Esse conjunto de ativos de processos é chamado de processo de software padrão organizacional. O processo padrão da organização é adaptado para definir um processo cada projeto.

Outros ativos de processos organizacionais são utilizados para apoiar a adaptação e implementação de processos definidos. Um processo padrão é composto por outros processos (ou seja, subprocessos) ou elementos de processo que descrevem as atividades e tarefas para executar o trabalho de forma consistente (SEI, 2010). O uso de processos padrão é defendido por quase todos os modelos e normas de qualidade de processo de software, incluindo a ISO/IEC 12207 (ISO/IEC, 2008), o CMMI (SEI, 2010) e o MPS-BR (SOFTEX, 2011). Todos eles sugerem a utilização de um processo padrão como ponto de partida a partir do qual os processos de projeto devem ser definidos.

O principal objetivo do planejamento do projeto é definir um escopo para o projeto, refinar os objetivos e desenvolver as ações necessárias para alcançá-los (PMI, 2008). Assim, o processo de planejamento do projeto deve, dentre outros (SEI, 2010): (i) determinar o escopo do projeto e atividades técnicas, (ii) identificar as saídas dos processos, tarefas e as entregas do projeto, e (iii) estabelecer um cronograma para a execução das tarefas do projeto, incluindo os critérios de realização e os recursos necessários para realizar as tarefas do projeto. Isso envolve, entre outros, estimar o esforço requerido, definir as atividades e tarefas a serem realizadas e os recursos necessários para executá-las, estabelecer o cronograma para a realização das tarefas e alocar tarefas e atribuir responsabilidades.

No contexto da gerência de projetos, processos relacionados a recursos visam planejá-los e controlá-los, incluindo equipamentos, instalações, materiais, software, serviços, pessoal e espaço (PMI, 2008).

As pessoas que participam do projeto devem ter a autoridade e as responsabilidades bem definidas em relação à sua participação no projeto. A

autoridade delegada aos participantes do projeto deve corresponder às suas responsabilidades. A qualidade e o sucesso de um projeto dependerá do pessoal que participa desse projeto. Portanto, deve ser dada atenção especial às atividades dos processos relacionadas com o pessoal, tal como a alocação de pessoas. Ao atribuir os membros para as equipes de projeto, seus interesses, pontos fortes e fracos devem ser considerados. O trabalho ou papel a ser desempenhado deve ser compreendido e aceito pela pessoa designada (ISO, 2003).

Processos relacionados com o tempo do projeto visam determinar as dependências e a duração das atividades, e visam assegurar a conclusão do projeto no tempo adequado. Esses processos incluem o planejamento das dependências das atividades, da estimativa de duração e do desenvolvimento e controle do cronograma. A definição do cronograma pode impactar nos recursos do projeto. Na verdade, processos relacionados com o tempo e processos relacionados com recursos (incluindo processos relacionados a pessoas) são extremamente interligados. Decisões ou ações tomadas no contexto de um deles devem ser analisadas com cautela, considerando as implicações em outros processos (ISO, 2003).

2.3.1 Ferramentas de Apoio ao Planejamento de Processo de Software

O ambiente ODE (*Ontology-based software Development Environment*) (FALBO et al., 2003) é um ambiente de desenvolvimento de software (ADS) centrado em processos (ARBAOUI et al., 2002) (GRUHN, 2002) e baseado em ontologias (FALBO et al., 2004).

O ambiente é constituído de várias ferramentas. No contexto deste trabalho, nos interessam as ferramentas ligadas ao planejamento do processo de software, tais como Def-Pro, a ferramenta de apoio à definição de processos (BERTOLLO et al., 2006) (SEGRINI et al., 2006), ControlPro, a ferramenta de apoio ao acompanhamento de projetos (DAL MORO, 2005) e AlocaODE, a ferramenta de apoio à alocação de recursos (COELHO, 2007).

2.3.1.1 Def-Pro

Def-Pro (SEGRINI, 2009) é a ferramenta de definição de processo de software de ODE. Ela oferece apoio à definição de processos em níveis baseada em componentes. Ela permite que processos de software sejam definidos em dois níveis de abstração, a saber: nível de processo padrão e nível de processo de projeto.

Em ambos os níveis de abstração, a definição segue os seguintes passos: Para um componente de processo complexo, define-se quais serão os subprocessos que o compõe, os componentes de processo simples. Para cada componente de processo simples, é possível definir as suas macroatividades. Para cada atividade, sendo ela uma macroatividade ou subatividade, é possível definir subatividades, pré-atividades, artefatos produzidos e requeridos, recursos necessários e procedimentos a serem adotados. Esse conjunto de características são os ativos de uma atividade. É importante ressaltar ainda que no nível de abstração de processo padrão a definição de componentes de processo pode acontecer nos três níveis.

2.3.1.2 ControlPro

ControlPro (DAL MORO, 2005) é a ferramenta de acompanhamento de projetos de ODE. Ela permite o acompanhamento do progresso do projeto e a alteração do processo em tempo de execução. Há uma forte integração entre ControlPro e Def-Pro, conseguida pelo compartilhamento dos mesmos modelos de classes.

Usando ControlPro, um gerente de projeto pode editar os ativos de processo definidos para as atividades do projeto, definir datas de início e fim para elas, realocar recursos e controlar o andamento das atividades. Além disso, o gerente de projeto pode controlar o estado do projeto, incluindo ações para iniciar, finalizar, cancelar ou reativar um projeto .

Desenvolvedores também podem usar ControlPro para acompanhar as atividades a eles alocadas, registrar esforço despendido nelas ou criar sub-atividades para melhor organizar o seu trabalho.

2.3.1.3 AlocaODE

AlocaODE (COELHO, 2007) (PIANISSOLA, 2007) é a ferramenta de alocação de recurso do ambiente ODE. Ela permite que para cada atividade sejam alocados pessoas, ferramentas de software, hardware e sistemas de apoio.

Para ter um maior controle sobre as alocações de recursos humanos a atividades dentro do ambiente, um recurso humano é alocado para desempenhar um papel específico, é possível registrar a dedicação em termos de tempo diário do recurso humano alocado à realização da atividade, o esforço total previsto para a alocação, as datas de início e fim previstas e efetivas da alocação, além do estado da mesma.

2.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou alguns conceitos importantes sobre Interoperabilidade Semântica, tema foco desta dissertação, com o intuito de oferecer ao leitor uma visão geral acerca do tema, para que este possa dar prosseguimento à leitura dos próximos capítulos.

Foram discutidos os tipos de problemas semânticos que podem ocorrer durante a integração de ferramentas e foi apresentada uma abordagem, OBA-SI, baseada em ontologia para prover interoperabilidade semântica.

O trabalho apresentado na sequência desta dissertação abrange a integração de sistemas de apoio ao planejamento de projetos, segundo o ponto de vista do

projetista, na camada de dados e no nível semântico, seguindo parcialmente o processo proposto por OBA-SI. A integração é realizada apenas no nível conceitual e, portanto, apenas a fase de análise de OBA-SI foi realizada.

Para a realização do trabalho, foi feita a reengenharia da Ontologia de Processo de Software proposta em (BERTOLLO, 2006). Essa reengenharia foi feita com base na Ontologia de Fundamentação Unificada (Unified Foundational Ontology – UFO) (GUIZZARDI, 2005) (GUIZZARDI et al., 2008), a qual é apresentada no próximo capítulo. O Capítulo 4 apresenta a nova versão da Ontologia de Processo de Software, a qual é usada como base para o projeto de integração semântica, discutido no Capítulo 5.

3 A ONTOLOGIA DE FUNDAMENTAÇÃO UNIFICADA

Uma ontologia pode ser definida como uma especificação de um vocabulário para representação de um domínio compartilhado de discurso (GUIZZARDI apud GUIZZARDI, R.S.S, 2006). Em outras palavras, uma ontologia é um modelo de domínio, composto por um conjunto de conceitos e relações. Uma ontologia é útil para fornecer uma compreensão clara sobre um determinado domínio.

O processo de reengenharia apresentado neste trabalho (ver Capítulo 4) é baseado na UFO (*Unified Foundational Ontology*) (GUIZZARDI, 2005), uma ontologia de fundamentação. Segundo Guizzardi e Wagner (2005), uma ontologia de fundamentação "define um conjunto de categorias ontológicas independente de domínio que formam uma base geral para ontologias específicas de domínio" (GUIZZARDI; WAGNER, 2005).

UFO tem sido desenvolvida baseada em um número de teorias das áreas de Ontologias Formais, Lógica Filosófica, Filosofia da Linguagem, Lingüística e Psicologia Cognitiva. Ela é dividida em três camadas incrementais, a saber: i) a UFO-A define o núcleo do UFO, apresentada em detalhes e formalizada em (GUIZZARDI, 2005). Ela inclui os conceitos relacionados a endurants; ii) a UFO-B define, com base na UFO-A, os conceitos relacionados a perdurants e, por fim, iii) a UFO-C define, sob a UFO-A e a UFO-B, conceitos relacionados às esferas de coisas intencionais e sociais.

As seções seguintes apresentam, respectivamente, os conceitos de UFO-A, UFO-B e UFO-C que são importantes para a compreensão completa deste trabalho. O conteúdo apresentado nessas seções foi elaborado com base nos seguintes trabalhos (GUIZZARDI, 2005; GUIZZARDI, R.S.S, 2006; GUIZZARDI et al., 2008; ZAMBORLINI, 2011).

Por questões de legibilidade, nos diagramas apresentados neste trabalho, apenas terão destaque os conceitos que não foram previamente apresentados (os demais conceitos ficarão em um tom mais claro de cinza).

3.1 UFO-A

A UFO-A trata particularmente de entidades chamadas endurantes (*endurants)*. A diferença entre endurantes e perdurantes (*perdurants),* este último tratado na UFO-B, pode ser intuitivamente compreendida em termos da distinção entre objetos e processos, respectivamente. A UFO-A é proposta por Guizzardi (2005). Devido à dificuldade de se traduzir alguns termos, eles são mantidos em língua inglesa nos diagramas e no texto, porém alguns são traduzidos no texto quando convém, tanto nessas seções quanto nas demais.

Uma distinção fundamental nessa ontologia ocorre entre a categoria **Individual** e a categoria **Universal**. **Universal** é a categoria ou tipo geral que representa os padrões de características presentes em diferentes indivíduos. Por exemplo, este tipo se aplica aos conceitos ou categorias de indivíduos Pessoa, Adulto e Cachorro, que são padrões de características comuns presentes em certos indivíduos. Por outro lado, **Individual** é a categoria ou tipo geral que se aplica aos indivíduos, que são entidades que existem na realidade e possuem uma identidade única. Por exemplo, este tipo se aplica aos indivíduos Eugênio e Rex. Além disso, cada indivíduo num domínio deve ser instância de pelo menos um universal. Por exemplo, Eugênio é um indivíduo do domínio que instancia os universais Pessoa e Adulto, enquanto Rex instancia o universal Cachorro.

Classifica-se, ainda, Universal como **Monadic Universal** ou **Relation Universal**. O tipo **Monadic Universal** é a categoria que se aplica aos conceitos que são padrões aplicados a indivíduos singulares, enquanto o tipo **Relation Universal** se aplica às relações, que são padrões aplicados a grupos de dois ou mais indivíduos. A Figura 9 apresenta esses conceitos.

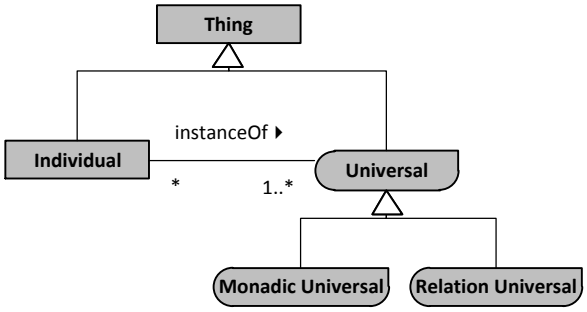

Figura 9 – UFO-A: Individual e Universal [1]

A principal distinção de tipos de universais monádicos e tipos de indivíduos em UFO é a distinção entre **substantials** e **moments**. **Substantial** representa as entidades do mundo real que persistem no tempo, mantendo as suas identidades. São exemplos de **Substantials** entidades físicas e sociais do dia-a-dia, como uma pessoa, uma bola, um cão, oceano atlântico e o presidente da república. Indivíduos podem ainda ser especializados ainda em **Situations** (situações). Situações são entidades complexas constituídas possivelmente por vários objetos (incluindo outras situações), sendo tratadas aqui como um sinônimo para o que é chamado na literatura de estado de coisas (state of affairs), ou seja, uma porção da realidade que pode ser compreendida como um todo. São exemplos de situações: "João Paulo estar gripado e com febre", e "Patricia estar casada com João Paulo que trabalha para a UFES".

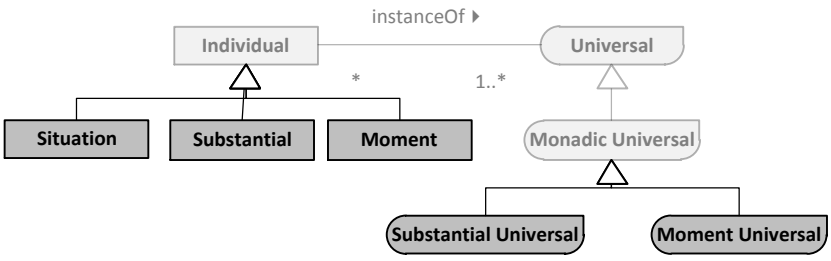

Figura 10 – UFO-A: Substantial e Moment

i. Substantials

A entidade **Substantial** (GUIZZARDI 2005) é especializada em **Sortal** e **Mixin**, conforme apresentado no diagrama da Figura 11. Os universais do tipo

[1] As entidades do tipo Universal serão representados no diagramas com uma forma arredondada para favorecer a compreensão dos diagramas.

Sortal são tais que agregam indivíduos com o mesmo princípio de identidade. Por exemplo, supondo que a impressão digital defina a identidade de uma pessoa, são universais do tipo Sortal os conceitos Pessoa, Cliente Pessoa Física e Adulto, uma vez que todos agregam indivíduos que possuem o mesmo princípio de identidade, como João, Maria e José. Em contraste, universais do tipo Mixin são tais que agregam indivíduos com princípios de identidade diferentes. Seguindo o mesmo exemplo, supondo também que o CNPJ defina a identidade de uma empresa, então o conceito *Item Assegurável*, que agrega pessoas e empresas, é um universal do tipo Mixin.

Na sequência do diagrama, os universais do tipo Sortal podem ainda ser classificados como Rigid Sortal ou Anti Rigid Sortal. Universais do tipo Rigid Sortal são conceitos rígidos como Pessoa e Empresa, cujos indivíduos devem instanciá-los enquanto existirem, por exemplo, João Paulo é necessariamente instância de Pessoa enquanto existir. Por outro lado, universais do tipo Anti Rigid Sortal são conceitos anti-rígidos como Cliente e Professor, cujos indivíduos podem eventualmente instanciá-los ou não enquanto existirem. Por exemplo, João Paulo pode ser instância do conceito professor em um determinado instante e deixar de ser professor em outro.

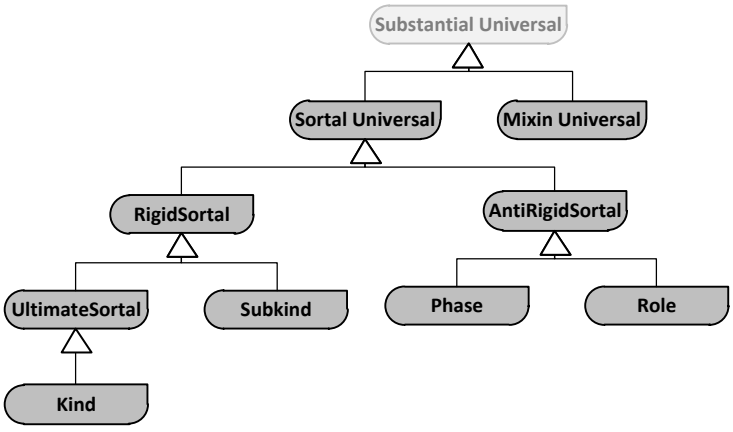

Figura 11 – UFO-A: Sortal

Os universais do tipo Rigid Sortal são ainda classificados como Ultimate Sortal quando proveem o princípio de identidade aos seus indivíduos, ou como Subkind quando apenas herdam este princípio. Por exemplo, considerando-se a impressão digital como o princípio de identidade provido a toda instância do

conceito Pessoa, este é do tipo Ultimate Sortal, enquanto os conceitos Homem e Mulher são do tipo Subkind pois, além de serem rígidos, especializam o conceito Pessoa e, portanto, herdam dele o princípio de identidade. O tipo Ultimate Sortal, especializa-se em Espécie (Kind) que categoriza conceitos cujas instâncias são complexos funcionais.

Os universais do tipo Anti-Rigid Sortal, por sua vez, como mostra a Figura 11, são classificados como **Phase** ou **Role**. Conceitos do tipo Phase (Fase) são tais que sua instanciação é determinada por uma propriedade intrínseca do indivíduo. Por exemplo, João instancia a fase Adulto se sua idade (propriedade intrínseca) é maior que 18 anos. Já os conceitos do tipo Role (Papel), como Cliente Pessoa Física ou Esposo, são relacionalmente dependentes, ou seja, sua instanciação é determinada por uma propriedade relacional do indivíduo. Por exemplo, João instancia o papel de Esposo se está casado com (propriedade relacional) Maria.

ii. Moments

Moment, por sua vez, denota o que é chamado de objetificação (instanciação) de uma propriedade (e não tem nenhuma relação com o sentido de instante temporal como o termo é comumente utilizado). Exemplos de **Moments** são: a cor de uma cadeira, uma carga elétrica e um sintoma de um determinado paciente (GUIZZARDI, 2005) (GUIZZARDI et al., 2008). Uma característica comum aos **Moments** é que todos estes são dependentes de **Substantials**, somente podendo existir em outros Substantials, como, por exemplo, uma carga elétrica somente pode existir em algum condutor elétrico, ou uma ligação covalente somente pode existir se os átomos conectados existirem. Os **Moments** são existencialmente dependentes de outros **Substantials**.

É dito que um indivíduo *x* é dependente existencialmente (*ed*) de um outro indivíduo *y* se, e somente se, por questão de necessidade, *y* existir sempre que *x* existir. A dependência existencial é uma relação modalmente constante, isto

é, se *x* for dependente de *y* esta relação existe entre estes dois indivíduos específicos em todos os mundos possíveis em que *x* existe.

Dependência existencial pode também ser usada para diferenciar **Relational Moments** (propriedades relacionais) de **Intrinsic Moments** (propriedades intrínsecas). Os **Intrinsic Moments** dependem de apenas um **Substantial** para existirem e são intrínsecas a ele. São exemplos de Intrinsic Moments a cor de um objeto, a dor de cabeça de uma pessoa e a temperatura de um ser. Se essa propriedade não é mensurável, ela é chamada de **Mode**, por exemplo, a dor de cabeça de João. Diferentemente dos **Intrinsic Moments**, os **Relational Moments**, também chamados de **Relators**, dependem de um conjunto de Indivíduos para existirem como, por exemplo, o Relator tratamento médico depende para existir dos Indivíduos Paciente e Médico.

A relação de dependência existencial que existe entre um Moment x e um Individual y na qual x depende de y é a relação de **Inherence** (inerência). Assim, para que um Individual x seja um Moment de um outro Individual y, a relação i(x,y), ou seja, x é inerente a y, deve existir entre os dois. Por exemplo, a inerência prende a existência de um sorriso a um rosto, ou a carga em um condutor específico ao próprio condutor. Nessa ontologia, admite-se que Moments podem ser inerentes a outros Moments. Um exemplo de um Moment que é inerente a um outro moment é a gravidade de um sintoma. A partir dessa relação, pode-se dizer que os Individuals que não são inerentes a outros Individuals são denominados de Substantials.

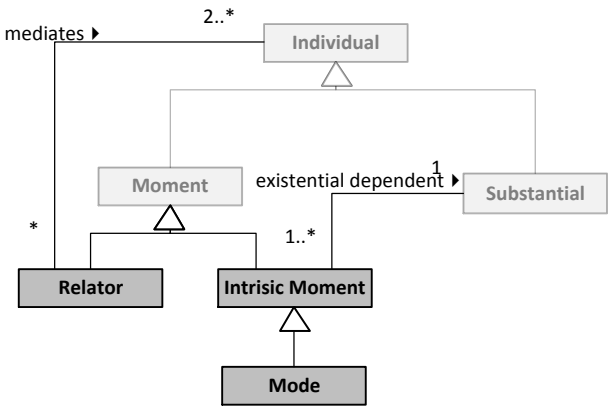

Figura 12 – UFO-A: Tipos de moments

iii. Relations

Retomando a distinção inicial dos tipos de universais (monádicos e de relação), o tipo **Relation Universal** é a categoria geral que se aplica aos universais de relação. Esta categoria é subdividida em **Formal** e **Material** (GUIZZARDI; WAGNER, 2008), como pode ser visualizado na Figura 13.

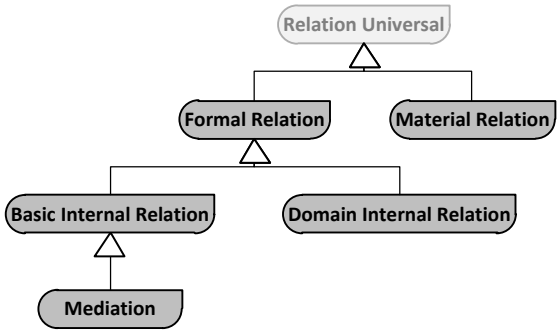

Figura 13 – UFO-A: Tipos de Relações

O tipo **Material Relation** se aplica às relações que dependem de algum interventor para valer, a saber, um indivíduo do tipo **Relator**. Por exemplo, a material relation *casado* existente entre Manuel e Anna vale enquanto existir o relator Casamento de Manuel e Anna. Contrariamente, as relações do tipo **Formal Relation** valem pela simples existência dos indivíduos relacionados. Por exemplo, a relação todo-parte entre Manuel e seu Cérebro vale sempre que ambos existirem.

As relações do tipo Formal podem ser ainda classificadas como **Basic Internal Relation** (Relação Básica Interna) ou **Domain Formal Relation** (Relação Formal de Domínio). O tipo **Basic Internal Relation** aplica-se a relações formais internas ditas de dependência existencial que têm representação entre as categorias de indivíduos da UFO. Este tipo pode ser especializado em **Mediation** (Mediação), que se aplica à relação **mediates** (de mediação) que define os indivíduos do tipo **Relator**. Por sua vez, o tipo **Domain Formal Relation** se aplica às relações formais que são específicas de domínio e que, por isso, não são representadas entre os tipos de indivíduos da UFO, mas entre os conceitos específicos de domínio, assim como as do tipo **Material Relation**.

A Figura 14 apresenta um panorama geral dos conceitos apresentados da UFO-A. Entretanto, é importante ressaltar que esta não é a versão completa da mesma.

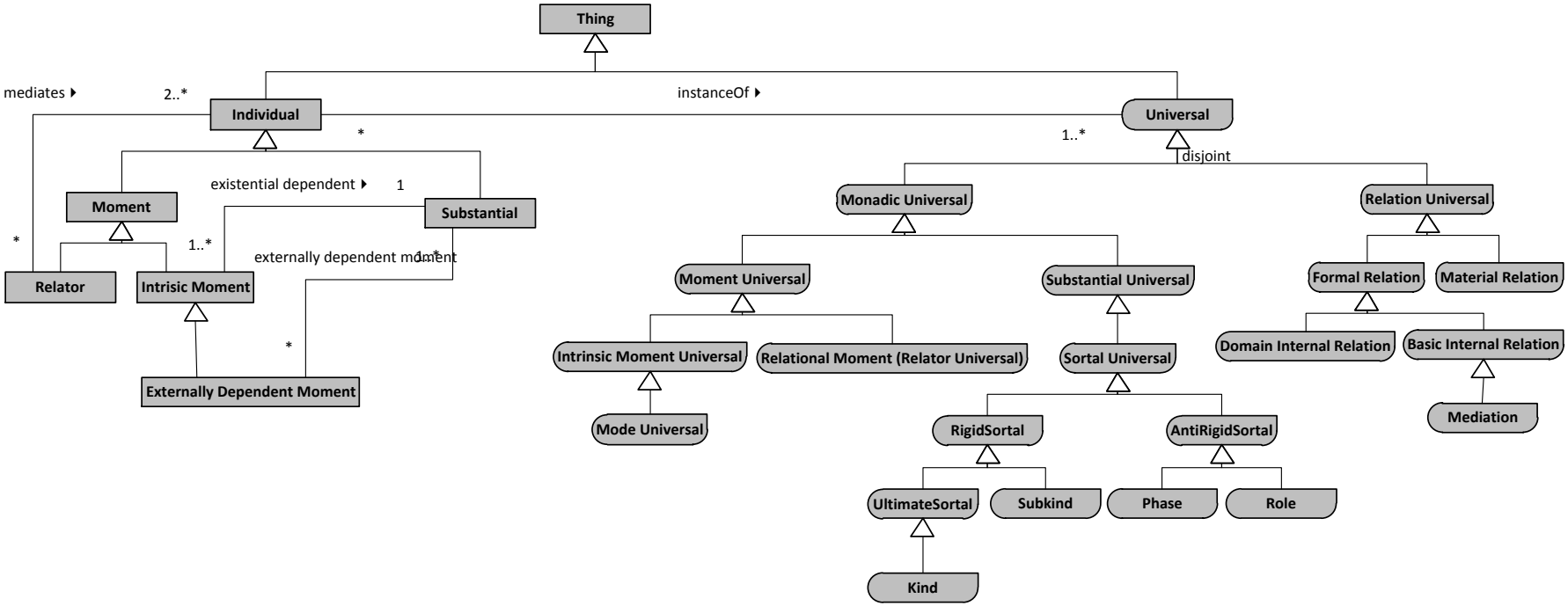

Figura 14 – Visão geral dos conceitos da UFO-A

3.2 UFO-B

UFO-B trata de Perdurants ou, mais intuitivamente, de Eventos (**Event**). Eventos (ou ocorrências) são indivíduos compostos de partes temporais. Eles acontecem no tempo no sentido de se estenderem no tempo acumulando partes temporais. São exemplos de eventos: uma conversa, uma partida de futebol, a execução de uma sinfonia e um processo de negócio. Em qualquer momento em que um evento está presente, apenas algumas de suas partes temporais estarão presentes. Como uma consequência, eventos não podem sofrer mudanças no tempo no sentido genuíno, uma vez que nenhuma de suas partes temporais mantém sua identidade ao longo do tempo.

Eventos são possíveis transformações de uma **situação** para outra na realidade, i.e., eles podem alterar o estado de coisas da realidade de um pré-estado (pre-state) para outro (post-state) em um **intervalo temporal**. Eventos são entidades ontologicamente dependentes no sentido de, para existirem, dependerem existencialmente de seus participantes. Seja o evento *e*: o ataque de Brutus a César. Nesse evento, há a participação de César, Brutus e da faca usada no ataque. Neste caso, *e* é composto da participação individual de cada uma dessas entidades. Cada uma dessas **participações** é por si própria um **evento** que pode ser **complexo** ou **atômico**, mas que existencialmente depende de um único substancial. Em UFO-B, ser atômico e ser instantâneo são noções ortogonais, i.e., **participações atômicas** podem se estender no tempo, bem como eventos instantâneos podem ser compostos de múltiplas participações (instantâneas). Em suma, o modelo da Figura 15 mostra essas duas perspectivas sob as quais eventos podem ser analisados, a saber: como entidades que se estendem no tempo com certas estruturas mereológicas (i.e., eventos simples ou complexos), e como entidades ontologicamente dependentes que podem envolver um número de participações individuais.

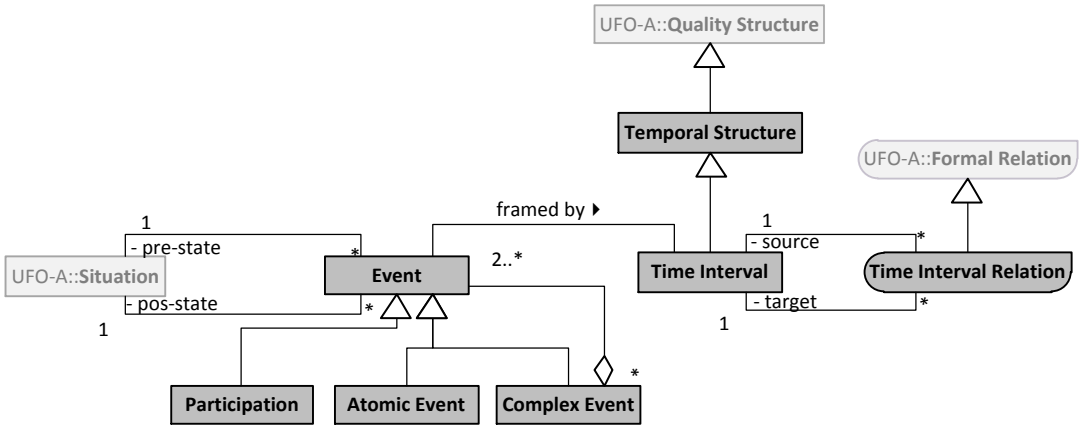

Figura 15 - UFO-B: Evento

3.3 UFO-C

A terceira camada da UFO é uma ontologia de entidades sociais (tanto endurants quanto perdurants), construída sobre UFO-A e UFO-B. Conforme mostra a Figura 16, uma das principais distinções feitas na UFO-C é entre agentes (**Agent**) e objetos não-agentivos (**Object**). Um agente é um substancial que cria ações (**Action**), percebe eventos (Event) e que pode possuir tipos especiais de modos, chamados de **intentional modes**. Os agentes podem ser físicos (**Physical Agent**) (por exemplo, uma pessoa) ou sociais (**Social Agent**) (por exemplo, uma organização). Um objeto, por outro lado, é um substancial incapaz de perceber eventos ou ter **intencional modes**. Objetos também podem ser divididos em objetos físicos (**Physical Object**) (por exemplo, um livro, um carro) e objetos sociais (**Social Object**) (por exemplo, dinheiro, língua). A descrição normativa (**Normative Description**) é um tipo de objeto social, que define uma ou mais regras/normas reconhecidas por, pelo menos, um agente social e que pode definir entidades sociais como universais (p.ex., tipos de comprometimentos sociais) e papéis sociais, tais como presidente, ou pedestre. Uma descrição plano (**Plan Description**) é um tipo especial de descrições normativas que descreve planos complexos (**Complex Action Universal**).

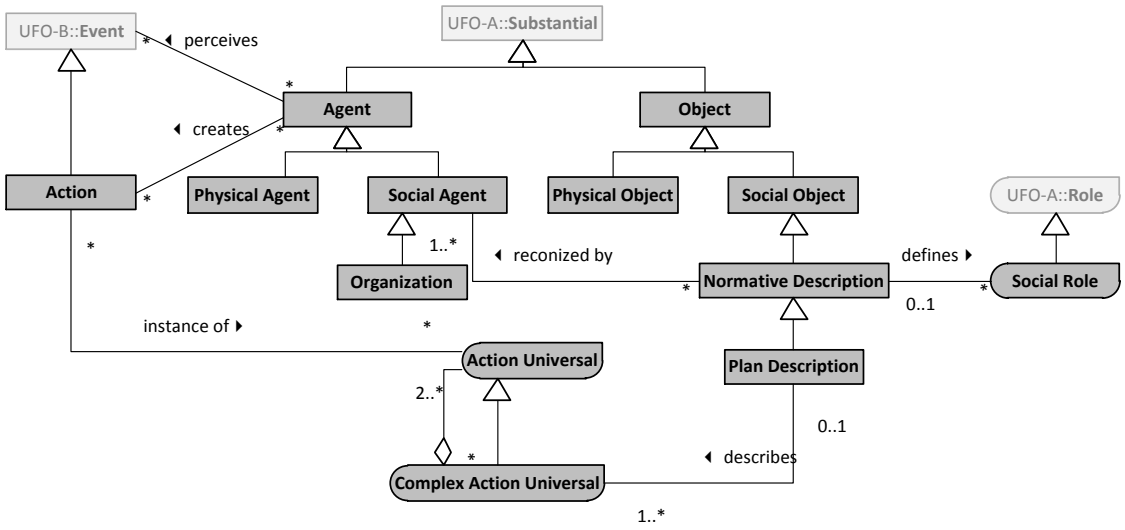

Figura 16 - UFO-C: Distinção entre Agente e Objeto

Como dito anteriormente, agentes são substanciais que podem possuir tipos especiais de modos, chamados de **intentional moments**. Intencionalidade deve ser entendida em um contexto mais amplo do que a noção de “alguma coisa que se intenciona”, mas como a capacidade de certas propriedades de certos indivíduos de se referir a possíveis situações (**Situation**) na realidade (SEARLE apud GUIZZARDI, FALBO, GUIZZARDI, 2008). Todo modo intencional tem um tipo, a saber: crença (**belief**), desejo (**desire**), intenção (**intention** ou **internal commitment**) – e um conteúdo proposicional que é um objetivo (**goal**), sendo este último uma representação abstrata de uma classe de situações referenciadas por esse modo intencional. Assim, “alguma coisa que se intenciona” é um tipo específico de intencionalidade denominada intenção (**intention**).

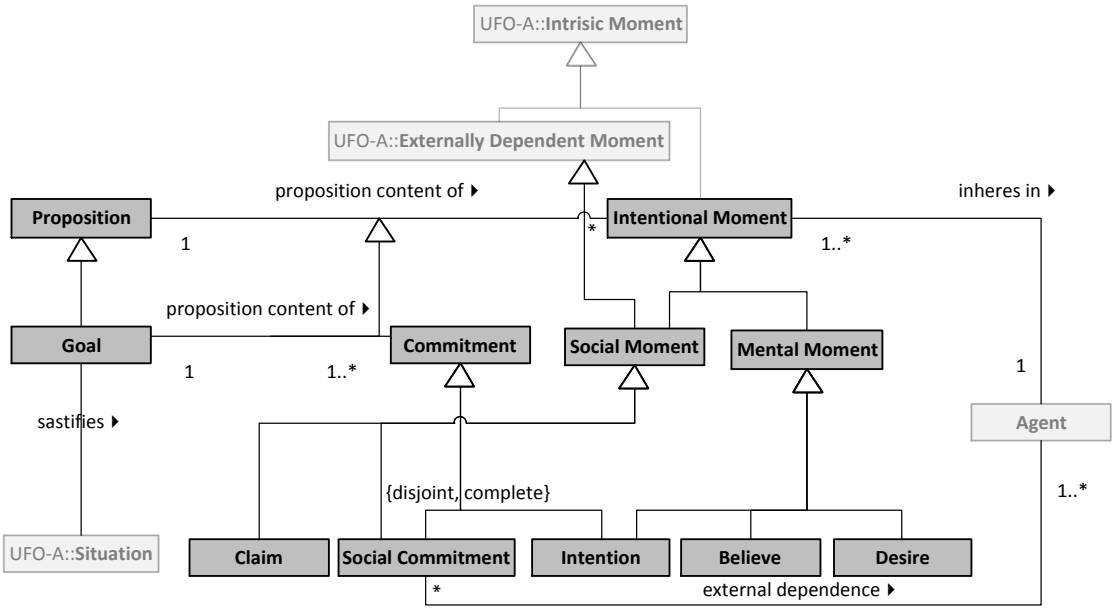

Figura 17 – UFO-C: Intentional Moment

Além de comprometimentos internos (intenções), existem também comprometimentos sociais (**social commitment**). Um comprometimento social é o comprometimento de um agente A com outro agente B. Como um momento externamente dependente (**externally dependent moment**), o comprometimento social é inerente ao agente A e é externamente dependente de B. O comprometimento social do agente A com o agente B necessariamente causa a criação de um comprometimento interno em A e a reivindicação social (**claim**) de B com A. Comprometimentos internos, ou intenções (**intentions**), por sua vez, fazem com que o agente realize ações.

Comprometimentos podem ser atômicos (**atomic commitment**) ou complexos (**complex commitment**). Um comprometimento complexo é composto de dois ou mais comprometimentos. Um tipo especial de comprometimento é um compromisso (**appointment**). Um compromisso é um comprometimento cujo objetivo (conteúdo proposicional) refere-se explicitamente a um intervalo de tempo (**time interval**). A Figura 18 sumariza os tipos de comprometimentos.

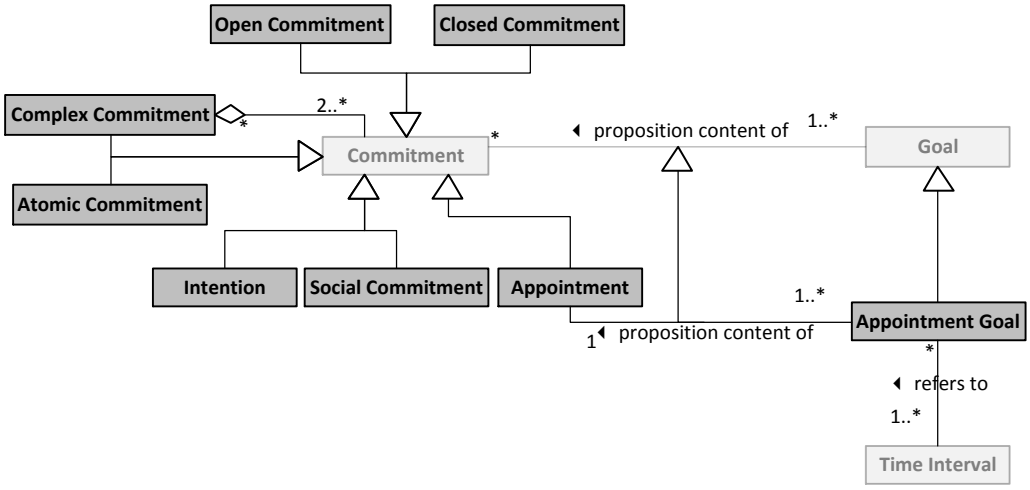

Figura 18 - Tipos de comprometimento

Por fim, comprometimentos podem ser abertos (**open commitments**) ou fechados (**closed commitments**), sendo a diferença que, no caso do último, o agente deve cumprir o comprometimento pela execução de uma ação específica. Diz-se, então, que um comprometimento C é baseado em um plano P tal que C é cumprido pelo agente A se, e somente se, A ativamente provocar uma situação S que satisfaz o conteúdo proposicional de C e por meio de uma ação p´ que é uma instância de P. Por exemplo, quando o professor João compromete-se com o departamento de avaliar os alunos matriculados em sua turma, este comprometimento é um comprometimento aberto, pois não existe um plano a ser seguido para que esse comprometimento seja cumprido. Em contrapartida, caso o professor João comprometa-se a avaliar os alunos através da aplicação de duas provas, este é um comprometimento fechado, pois, para alcançá-lo, ele deverá, necessariamente, aplicar duas provas. Comprometimentos abertos e fechados podem explicar as noções de delegação aberta e fechada, respectivamente.

Delegação (**Delegation**) é um tipo especial de relação material (**material relation**) derivada de um social relator **delegatum**. Quando um agente A (**delegator**) delega um objetivo a um agente B (**delegatee**), B se compromete (**comprometimento social**) com o agente A. O agente A, por sua vez, passa a ter o direito de reivindicar (claim) de B o cumprimento do que foi delegado. O par de comprometimento / reivindicação compõe o delegatum a partir do qual a delegação é derivada, como mostrado na Figura 19.

Compromissos e reivindicações sempre formam um par que se refere ao mesmo conteúdo proposicional, e um relator social é um exemplo de um relator composto por dois ou mais pares de compromissos associados / reclamação.

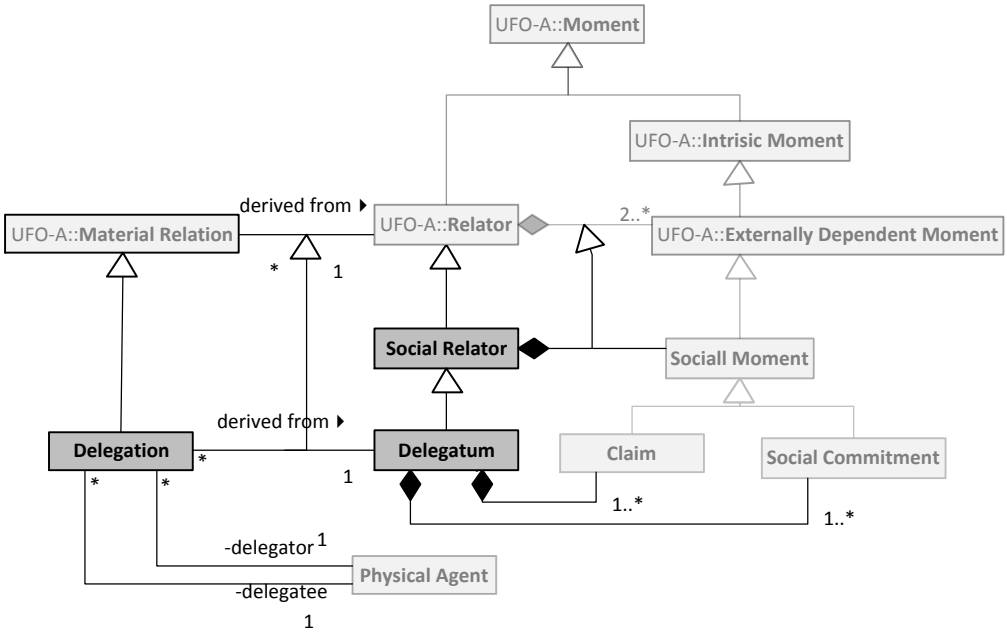

Figura 19 - UFO-C: Delegação.

Finalmente, as ações (**action**) são eventos intencionais, ou seja, eles têm a finalidade específica de satisfazer alguma intenção (por exemplo, um processo de negócio). Como eventos, ações podem ser atômicas (**atomic action**) ou complexas (**complex action**). Já uma ação complexa é composta de duas ou mais participações (**Participation**). Essas participações, por sua vez, podem ser intencionais (sendo, portanto, elas próprias ações) ou eventos não intencionais. Por exemplo, o ataque de Brutus a César inclui a participação intencional de Brutus e a participação não intencional da faca. Em outras palavras, nem toda participação de um agente é considerada uma ação, mas somente as participações intencionais, aqui denominadas contribuições de ação (**action contributions**). Apenas agentes (entidades capazes de possuir modos intencionais) podem realizar ações. A Figura 20 apresenta esses conceitos.

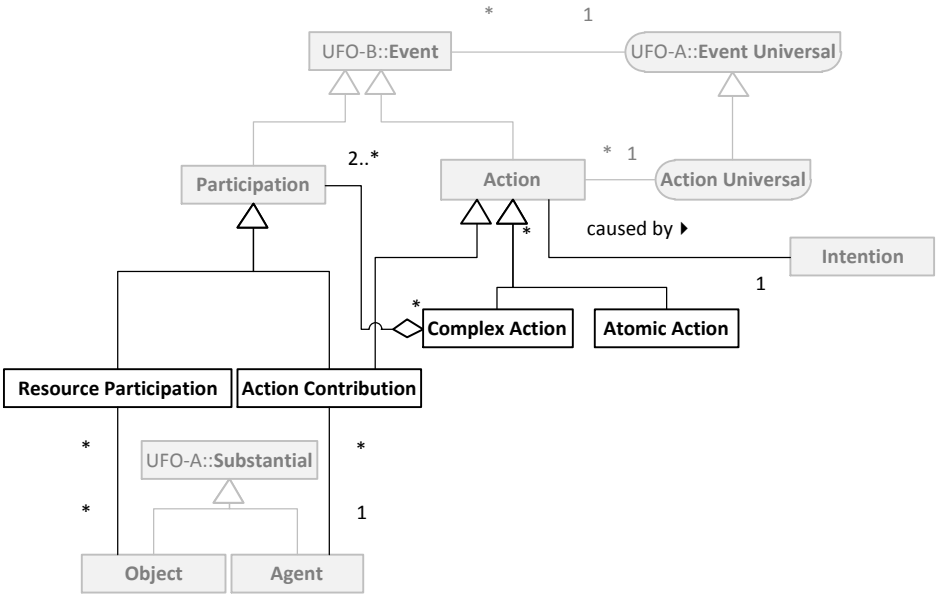

Figura 20 - UFO-C: Ação.

4 REENGENHARIA BASEADA NA UFO DA ONTOLOGIA DE PROCESSO DE SOFTWARE

Este trabalho integra conceitualmente duas ferramentas que apoiam o planejamento do processo de software (Seção 2.3). De acordo com OBA-SI (ver Seção 2.2.1) uma ontologia de domínio deve ser utilizada para atribuir significado aos modelos conceituais das ferramentas que estão sendo integradas para que, posteriormente, eles possam ser mapeados semanticamente.

Uma Ontologia de Processo de Software (*Software Process Ontology* - SPO) é usada como a ontologia de referência no processo de integração dessas ferramentas. Essa ontologia foi originalmente desenvolvida em (BERTOLLO, 2006) com o objetivo de estabelecer uma conceituação comum para que as organizações de software pudessem falar a respeito de processo de software. Em (GUIZZARDI et al. 2008) essa ontologia foi objeto de uma reengenharia parcial, onde alguns conceitos da SPO foram mapeados para conceitos da UFO com o objetivo de tornar explícitos os compromissos ontológicos subjacentes. Entretanto, essa reengenharia não cobriu todos os conceitos necessários para discutir o domínio de planejamento de projetos de software, além de apresentar alguns problemas, como discutido mais adiante.

Este capítulo apresenta um passo à frente na reengenharia da SPO com base na UFO, com o intuito de cobrir os conceitos necessários para discutir o domínio de planejamento de projetos de software. Ele está estruturado da seguinte forma: a Seção 4.1 apresenta a versão original da ontologia de processo de software e a versão produzida em sua primeira reengenharia; a Seção 4.2 apresenta a nova versão, resultante da reengenharia realizada neste trabalho; e a Seção 4.3, por fim, apresenta as considerações finais do capítulo.

4.1 ONTOLOGIA DE PROCESSO

Como discutido na Seção 2.3, durante o planejamento de projetos, é necessário definir o processo do projeto, agendar as atividades e alocar os recursos humanos que irão executá-las. Para controlar e monitorar o projeto de forma eficiente, é necessário acompanhar o cumprimento dessas atividades e o tempo gasto em sua execução.

Durante a definição do processo de software para um projeto, o gerente de projeto deve identificar as atividades que deverão ser realizadas para alcançar os objetivos do projeto. Isso é feito adaptando um processo padrão da organização, considerando as características do projeto e de sua equipe. O processo do projeto é a base para as atividades de gerência de projeto. Após definir o processo, o gerente de projeto deve definir a duração de cada atividade e alocar as pessoas que irão executá-las.

A Ontologia de Processo de Software (*Software Process Ontology* - SPO) foi originalmente desenvolvida em (BERTOLLO, 2006). Essa ontologia foi dividida em quatro subontologias, a saber: ontologia de processo, de atividade, de recursos e de procedimentos de software. A Figura 21 mostra um fragmento da primeira versão dessa ontologia com alguns dos conceitos das subontologias de atividade, recursos e processos de software. Esse fragmento foi selecionado, porque neste trabalho estamos interessados na parte da conceituação da SPO que é relevante para o planejamento de projetos.

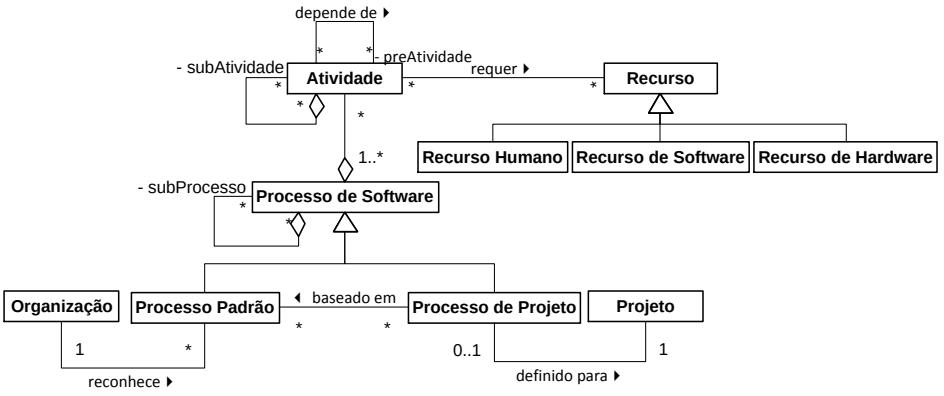

Figura 21 – Fragmento Ontologia de Processo de Software

Um **Processo de Software** pode ser decomposto em **Atividades** ou em outros processos, chamados de **subprocessos**. Uma **atividade** requer **recursos** (humanos, software e hardware) para ser executada. Uma **atividade** pode ser decomposta em **subatividades** e pode depender de outras atividades, chamadas de **pré-atividades**. Um **processo padrão** é um processo reconhecido pela **organização** e que estabelece os requisitos básicos dos **processos de projeto** que serão executados nessa organização. O **processo de projeto** refere-se ao processo definido para um determinado **projeto**, considerando suas particularidades.

A versão original da SPO foi objeto de uma reengenharia parcial em (GUIZZARDI et al., 2008), onde alguns conceitos da SPO foram mapeados para conceitos da UFO. A Figura 22 mostra um fragmento da versão da ontologia de processo depois de passar pela reengenharia. Assim no Capítulo 3, as entidades do tipo Universal serão representadas nos diagramas com uma forma arredondada para favorecer a compreensão dos diagramas.

Interpretando a versão original da SPO em termos da UFO, Guizzardi, Falbo e Guizzardi (GUIZZARDI et al., 2008) apontaram que na versão original as noções de Ação (*Action)* e Universal de Ação (*Action Universal* (*Plan*)) foram colapsadas. Para resolver este problema, foram introduzidos os conceitos de **Ocorrência de Atividade** e **Ocorrência de Processo de Software** para denotar ações particulares que ocorrem em intervalos de tempo específicos.

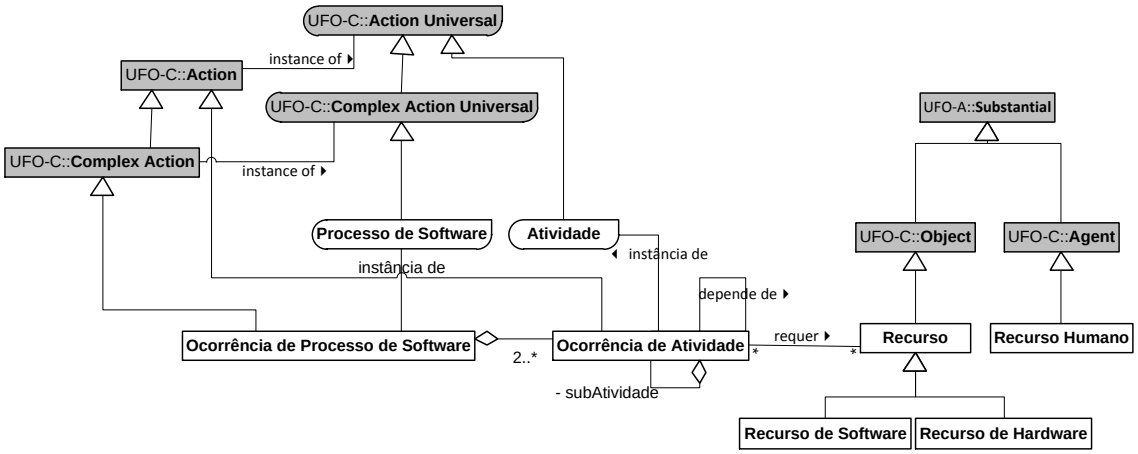

Figura 22 – Reengenharia da Ontologia de Processo de Software

Além disso, uma **Ocorrência de Processo de Software** é uma instância de um **Processo de Software**, que é, por sua vez, um universal de ação complexa (*Complex Action Universal)*. Analogamente, uma **Ocorrência de Atividade** é uma instância de **Atividade**, que é um universal de ação (*Action Universal)*.

A noção de **Recurso** na versão original da SPO foi mapeada para a noção de objeto não agentivo (*Object)* em UFO e o conceito de **Recurso Humano** deixou de ser considerado um **Recurso** e passou a ser considerado um agente (*Agent).* Um **Recurso Humano** não pode ser usado, modificado, criado ou eliminado por uma **Ocorrência de Atividade**, enquanto **Recursos de Hardware** e **de Software** são usados em uma **Ocorrência de Atividade**.

Isso possibilitou a distinção entre participações de objeto e participações de agentes. Assim, a relação **requer** na versão original de SPO subjulga diferentes tipos de participação na ocorrência de atividade, a saber: participação de objeto (*Object Participation*), ou seja, o uso de um recurso de software ou de um recurso de hardware, e contribuição de ação (*Action Contribution*), ou seja, a participação de um recurso humano na ocorrência de atividade. Outras mudanças foram feitas, mas elas estão fora do escopo deste trabalho.

Apesar de uma importante análise ontológica ter sido feita em (GUIZZARDI et al., 2008), há muitos aspectos que permaneceram em aberto. O que é um processo padrão? Qual é a diferença de uma atividade que é definida no processo de projeto

e aquela que é agendada e finalmente executada? Como podemos tratar a alocação de pessoas? Para responder a estas e outras perguntas, demos um passo à frente no processo de reengenharia da SPO.

4.2 REENGENHARIA DA ONTOLOGIA DE PROCESSO

Na primeira iniciativa de reengenharia da SPO, algumas questões importantes não foram abordadas, dentre elas: (i) a distinção entre os processos de software padrão e processos de projeto, (ii) a distinção entre os diversos tipos de atividades: atividades planejadas, atividades programadas e ocorrências de atividade. Avançamos nesses aspectos e abordamos o problema da alocação de pessoas para realizar as atividades programadas.

Contudo, no decorrer do processo de reengenharia, percebemos que seria útil utilizar uma ontologia que abordasse os conceitos do domínio de Organizações de Software. Desta forma, utilizamos a Ontologia de Organizações de Software (*Software Enterprise Ontology* - SEO) desenvolvida em (VILLELA et al., 2005) e que em (BARCELLOS; FALBO, 2009) também passou por um processo de reestruturação baseada na UFO.

Na sequência, apresentamos a reengenharia da SPO. Os conceitos da SEO são explicados à medida que são usados.

4.2.1 Definição do Processo Padrão

Embora a versão original da SPO faça a distinção entre processo padrão e processo de projeto, a reengenharia realizada em (GUIZZARDI et al., 2008) não levou em conta essa distinção. Como discutido anteriormente, um processo padrão refere-se a um processo genérico definido pela organização e que estabelece os requisitos básicos para os processos a serem executadas na organização. Olhando

para a conceituação estabelecida pela UFO, um **processo padrão** deve ser visto como um Universal de Ação Complexa (C*omplex Action Universal*), descrito por um plano. Desta forma, introduzimos o conceito de **Documento de Definição de Processo Padrão** como a descrição do plano (*Plan Description*) que descreve o **processo padrão**. Como uma descrição normativa (*Normative Description*), o **documento de definição de processo padrão** define as regras para realização dos projetos da **organização.**

De acordo com o domínio (ver Seção 2.3), um **processo padrão geral** é um **processo padrão** composto de **processos padrões específicos**, permitindo que a organização defina **subprocessos**. Por exemplo, o processo padrão de uma organização pode ser decomposto em dois subprocessos padrão, o subprocesso de Desenvolvimento de Software e o subprocesso de Gerência de Projetos. Um **processo padrão específico** (por exemplo, o Processo Padrão de Gerência de Projeto), por sua vez, é decomposto em **atividades padrão**, que são universais de ação (*Action Universal*). Enquanto universais de ação, as atividades padrão podem ser **atividades padrão simples** ou **complexas**. Por exemplo, o subprocesso de Desenvolvimento de Software é decomposto nas atividades Análise de Requisitos e Projeto de Sistema. A atividade de Análise de Requisitos, por sua vez, pode ser decomposta nas subatvidades Levantamento de Requisitos e Modelagem Estrutural.

A Figura 23 mostra esses conceitos na nova versão da SPO. Os conceitos da UFO estão destacados de cinza na figura e os relacionamentos de subtipo são utilizados para mapear os conceitos e relações da SPO com os conceitos e relações da UFO. Vale ressaltar que a forma de estruturação dos processos e atividades padrão é devida ao domínio de processo de software. Desta forma, ela não é diretamente inspirada na UFO, embora esteja alinhada a ela.

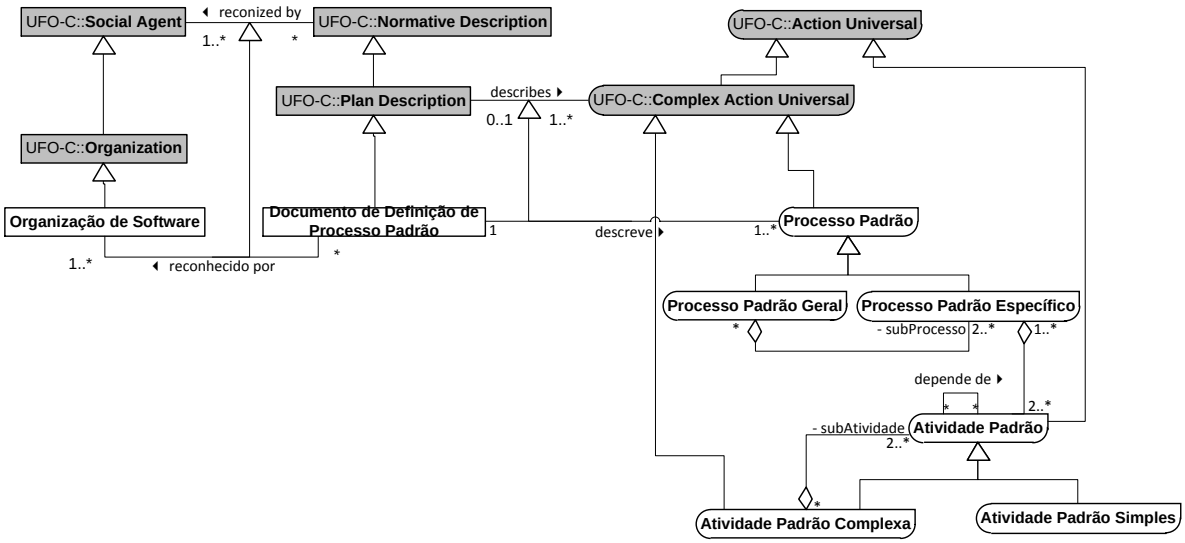

Figura 23 – Definição de Processo Padrão

Neste trabalho, tentou-se ser mais preciso do que na primeira iniciativa de reengenharia (Guizzardi et al. 2008) com relação às multiplicidades. Aspectos como a suplementação fraca (*weak supplementation*), por exemplo, foram considerados. Assim, um **processo padrão geral** deve ser composto de pelo menos dois **processos padrão específicos**. O mesmo se aplica a **atividades padrão complexas**.

4.2.2 Definição do Processo de Projeto e Agendamento

Em contraste com **Processo Padrão**, **Processo de Projeto** refere-se aos processos definidos para projetos específicos, considerando suas particularidades. Quando um gerente de projeto define um **processo de projeto**, ele está realizando um ato comunicativo que cria o comprometimento interno (*Internal Commitment - Intention*) da **equipe do projeto**: o comprometimento de executar as atividades definidas no processo de projeto. Assim, seguindo a conceituação de UFO, um **processo de projeto** é um tipo de intenção (*Intention*), como mostrado na Figura 24.

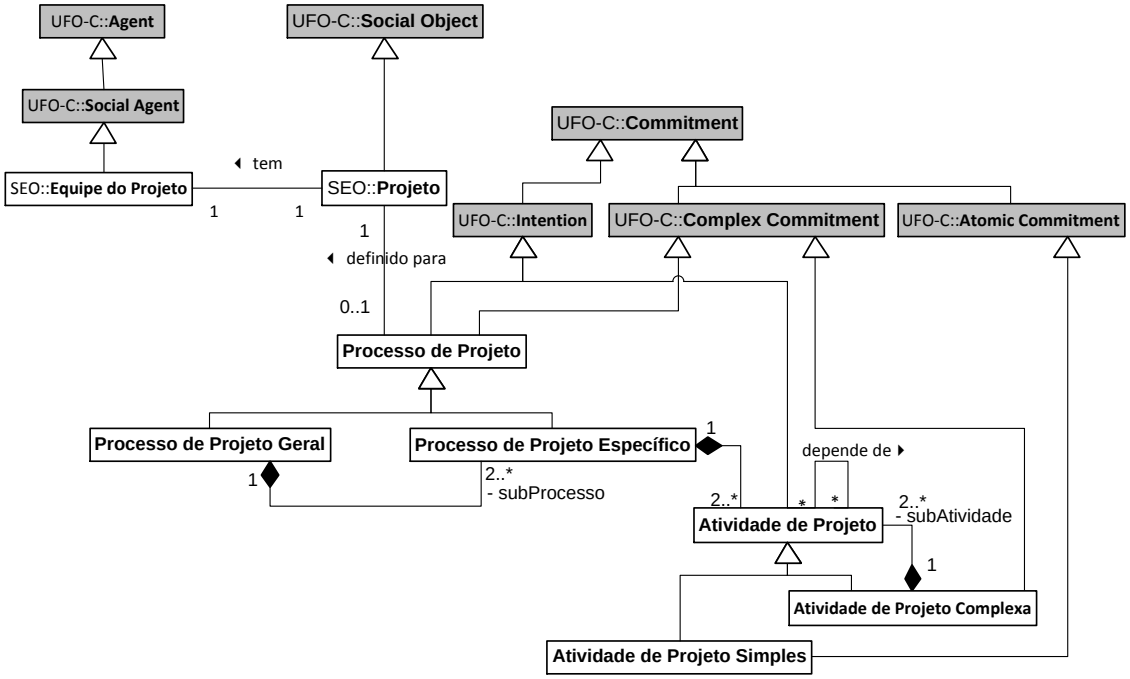

Figura 24 – Definição do Processo de Projeto.

Analogamente ao processo padrão, um **processo de projeto geral** é um **processo de projeto** que é composto de **processos de projeto específicos**, permitindo que a organização defina **subprocessos**. Um **processo de projeto** específico, por sua vez, é decomposto em **atividades de projeto**, que também são comprometimentos. Desta forma, um **processo de projeto** é sempre um comprometimento complexo (*Complex Commitment*), uma vez que é composto por **subprocessos** ou por **atividades de projeto**. Uma **atividade de projeto** pode ser **complexa**, ou seja, é composta por outras **subatividades** ou pode ser **simples**. **Atividades de projeto complexas** são, de acordo com a UFO, comprometimentos complexos (*Complex Commitment*) e as **atividades de projeto simples** são comprometimentos atômicos (*Atomic Commitment*). Uma **atividade de projeto** pode depender de outras atividades para acontecer.

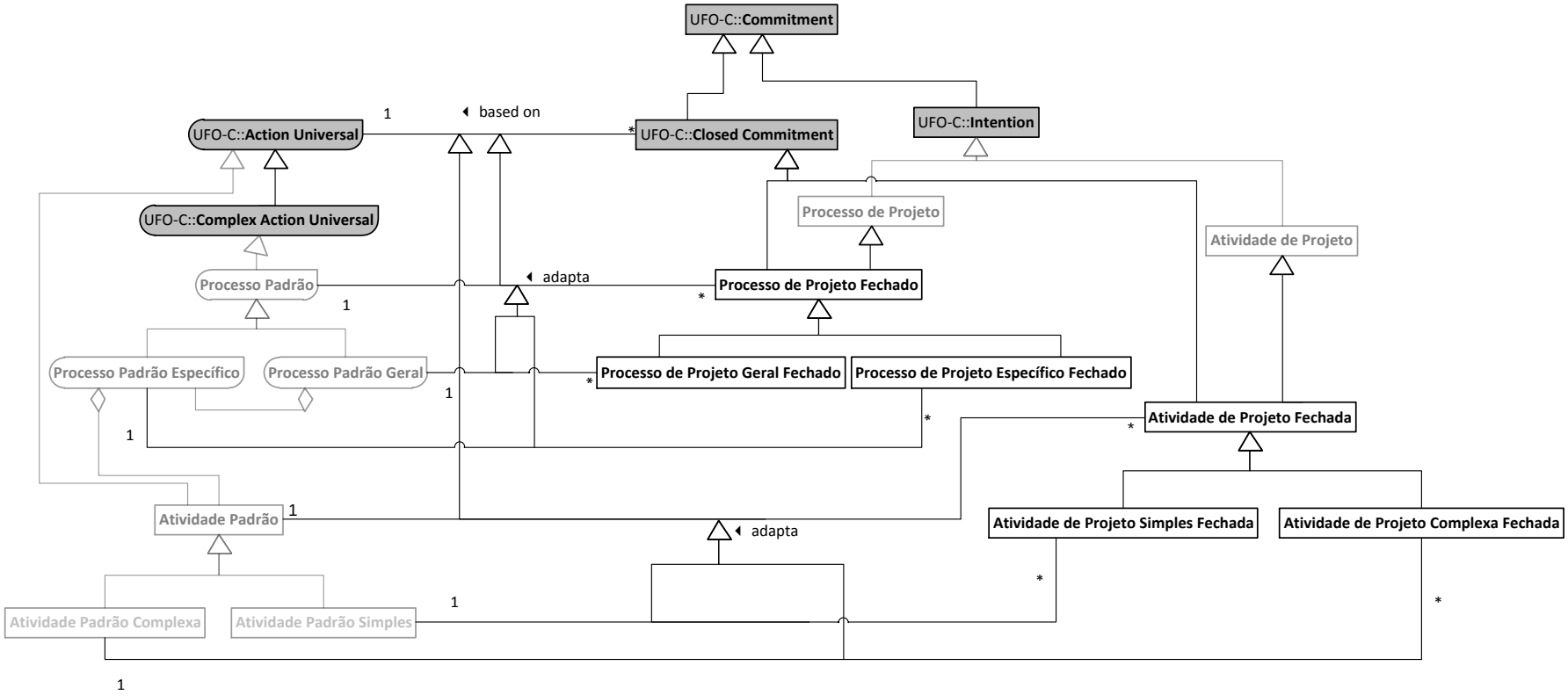

Figura 25 - Definição do Processo de Projeto baseado em um Processo Padrão

Uma **Atividade de Projeto Simples** pode ser definida adaptando uma **atividade padrão simples**, considerando as características do projeto. Quando isto ocorre, ela é uma **atividade de projeto simples fechada**, como mostra a Figura 25. Uma **atividade de projeto complexa**, por sua vez, só é fechada se, além de adaptar uma **atividade padrão complexa**, for composta por **atividades de projeto fechadas**. O mesmo acontece com um **processo de projeto**, que para ser fechado, além de ter que ser baseado em um **processo padrão**, todas as suas partes também devem ser fechadas. Um **processo de projeto fechado** ou uma **atividade de projeto fechada** são, de acordo com a UFO, um comprometimento fechado *(Closed Commitment)*, uma vez que a organização deve executar uma ação específica para cumprir o comprometimento. É importante salientar que, para um processo/atividade ser fechado, todas as suas partes também devem ser, mas o contrário não é obrigatório.

Desta forma, um subprocesso de projeto baseado em um subprocesso padrão é fechado quando todas as suas (sub)atividades são definidas tomando por base alguma (sub)atividade padrão desse subprocesso padrão.

Uma vez definido o **processo de projeto**, seja ele fechado ou não, é necessário programar quando cada processo e cada atividade deverão ocorrer. Um **processo programado** (ou uma **atividade programada**) é um processo (ou uma atividade), cujas datas de início e de fim de sua execução estão definidas, ou seja, definem-se os intervalos de tempo (*Time Interval*) em que cada (sub)processo e cada (sub)atividade do projeto deverão ocorrer. Em termos de UFO, isto corresponde a compromissos (*Appointments*), ou seja, comprometimentos cujo conteúdo proposicional refere-se explicitamente a um intervalo de tempo, como mostra a Figura 26.

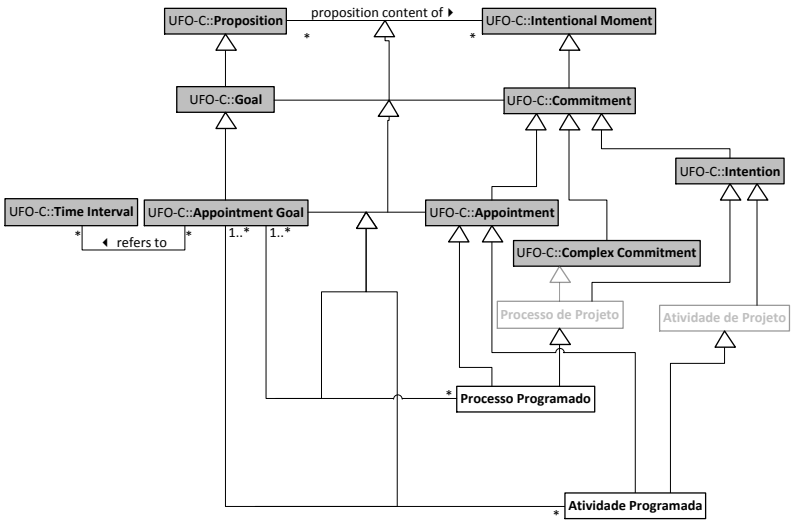

Figura 26 – Processo de Projeto Programado

4.2.3 Tipos de Recurso

Uma **atividade padrão** indica os papéis que os recursos humanos vão desempenhar ao executar as atividades desse tipo. Um **papel de recurso humano** é descrito pelo **plano de cargos** da organização. Um **papel de recurso humano** é um papel social (*Social Role*) definido por uma descrição normativa (*Normative Description*), que é o **plano de cargos**, e reconhecido por uma organização (*Organization*), a **organização de software**.

Uma **atividade padrão** indica também os **tipos de produto de software** (por exemplo, uma ferramenta de modelagem UML) e os **tipos de equipamento de hardware** (por exemplo, um notebook) que serão utilizados. O mesmo ocorre na definição do **processo de projeto**: cada **atividade de projeto** indica os **tipos de produto de software** e de **equipamento de hardware** que serão utilizados e quais os papéis que os recursos humanos irão desempenhar ao realizar as ações (ocorrências de atividade) para satisfazer o comprometimento.

Tipos de produto de software e **tipos de equipamento de hardware** são tipos de objeto (*Object Kind*) em UFO. Vale a pena ressaltar que na definição de um **processo padrão**, ou mesmo na definição de um **processo de projeto**, não estamos definindo qual produto de software ou qual equipamento de hardware específico será usado, mas apenas os tipos desses objetos (*Object Kind*) e, portanto, não estamos falando de recursos, mas sobre os tipos de recursos. O mesmo se aplica aos recursos humanos. Novamente, não estamos alocando pessoas, estamos apenas definindo os papéis sociais (**papel de recurso humano**) que as pessoas devem assumir ao realizar as ocorrências de atividade correspondentes (Seção 4.2.5). A Figura 27 mostra esses conceitos.

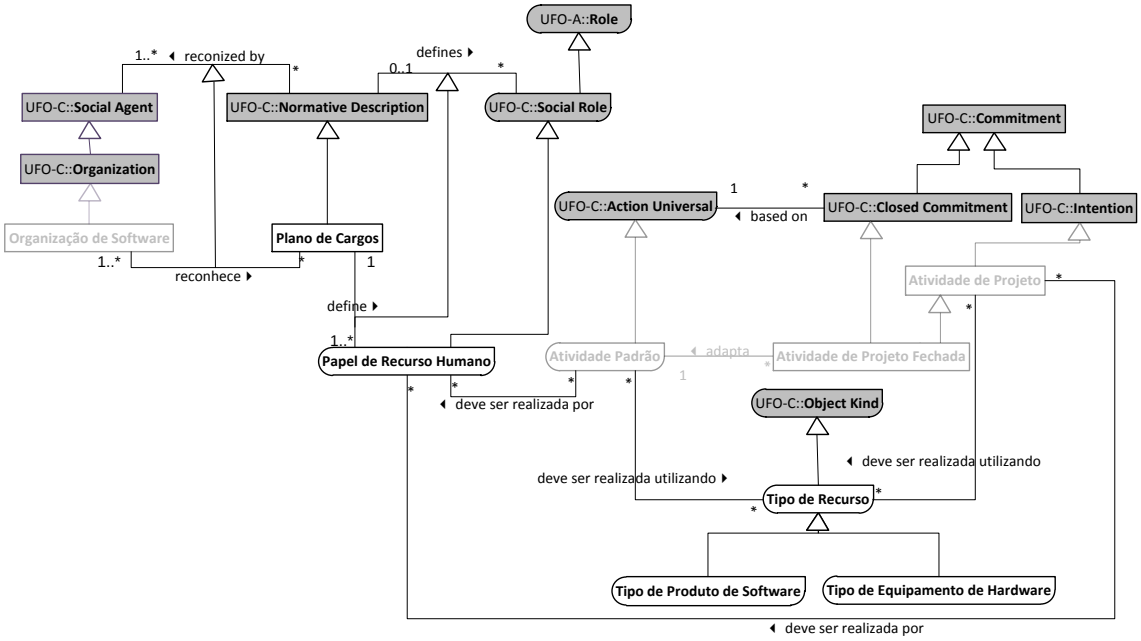

Figura 27 - Recursos definidos para as atividades

4.2.4 Alocação de Recursos Humanos

Como discutido na Seção 2.3, para que um projeto seja bem-sucedido, uma atenção especial deve ser dada às tarefas relacionadas à equipe do projeto, tais como a seleção de membros para as equipes e a alocação deles nas atividades do

projeto. Um **recurso humano** é selecionado para uma **equipe de projeto** para desempenhar um papel específico descrito pelo **plano de cargos**. Assim, de acordo com a UFO, a **alocação de recurso** (atribuição de uma pessoa para uma **equipe de projeto**) é um *relator* social (*Social Relator*) que conecta o **recurso humano,** desempenhando um papel específico (**Participante de Projeto**), com a **equipe do projeto**. A Figura 28 mostra esses conceitos.

É importante ressaltar que um recurso humano pode ser alocado diversas vezes para um mesmo projeto, onde cada alocação determina o papel que ele poderá desempenhar. Assim, é possível que João seja alocado para desempenhar o papel de Analista de Sistema e, neste mesmo projeto, ser também alocado para desempenhar o papel de Projetista.

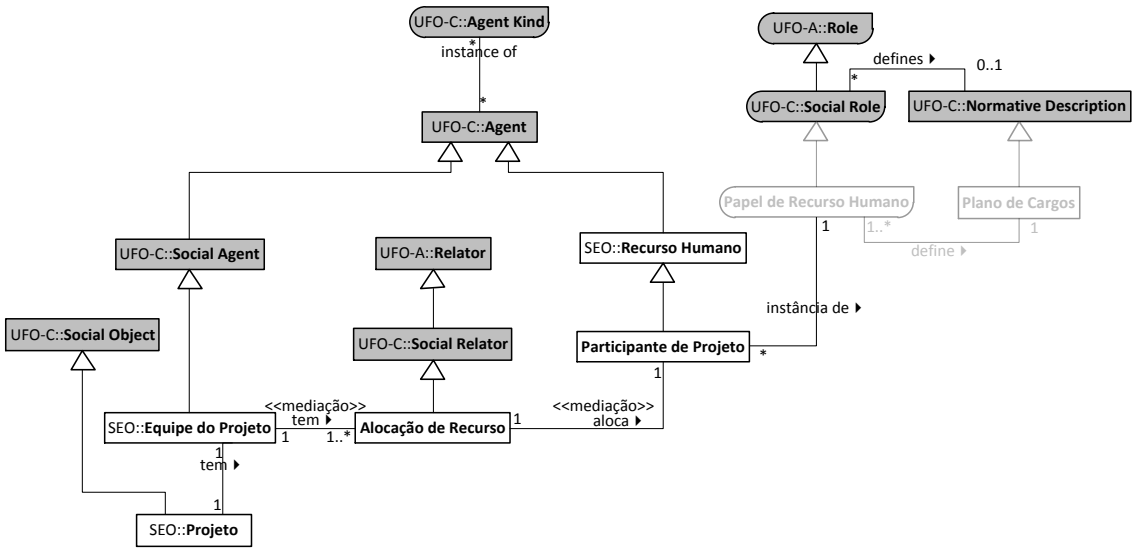

Figura 28 – Alocação de um recurso humano em um projeto

Uma vez que o **recurso humano** esteja participando de um **projeto**, ele pode ser alocado para atividades desse projeto. Neste caso, a **equipe do projeto** está delegando ao **participante do projeto** a responsabilidade de alcançar os objetivos da **atividade do projeto** seguindo um plano específico e, portanto, a alocação do **participante do projeto** na **atividade do projeto** é um tipo de delegação (*Delegation*) na UFO, em que a **equipe do projeto** é o delegante (*delegator*) e o **recurso humano** é o delegado (*delegatee*). Em outras palavras, a alocação do recurso para a atividade deriva da relação material entre **recurso humano alocado**,

equipe do projeto e os objetivos da **atividade de projeto**. A Figura 29 mostra o modelo que captura esta conceituação.

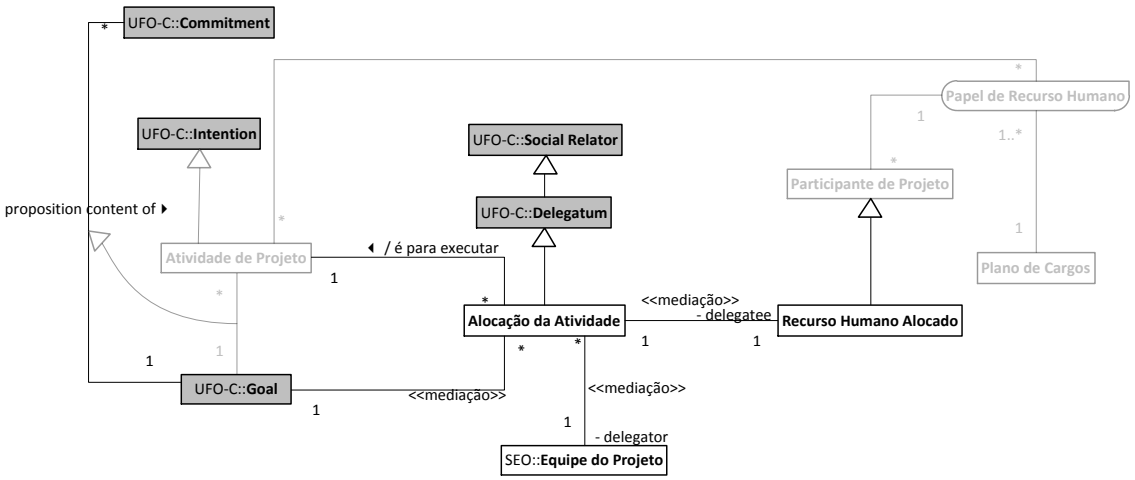

Figura 29 – Alocação de Recurso Humano em uma Atividade

4.2.5 Ocorrência de Atividade

Finalmente, ações são executadas no contexto do projeto, a fim de atingir os objetivos. Como apontado em (GUIZZARDI et al., 2008), **ocorrências de processo** e de **atividade** são ações (*Action*) em UFO, causadas pela intenção do recurso humano alocado. Ações são eventos intencionais, ou seja, elas têm a finalidade específica de satisfazer o conteúdo proposicional de uma intenção (*Intention*). Ocorrências de processo são ações complexas (*Complex Action*) causadas por **processos programados**, enquanto **ocorrências de atividade** são ações causadas por **atividades programadas**. A Figura 30 mostra esses conceitos.

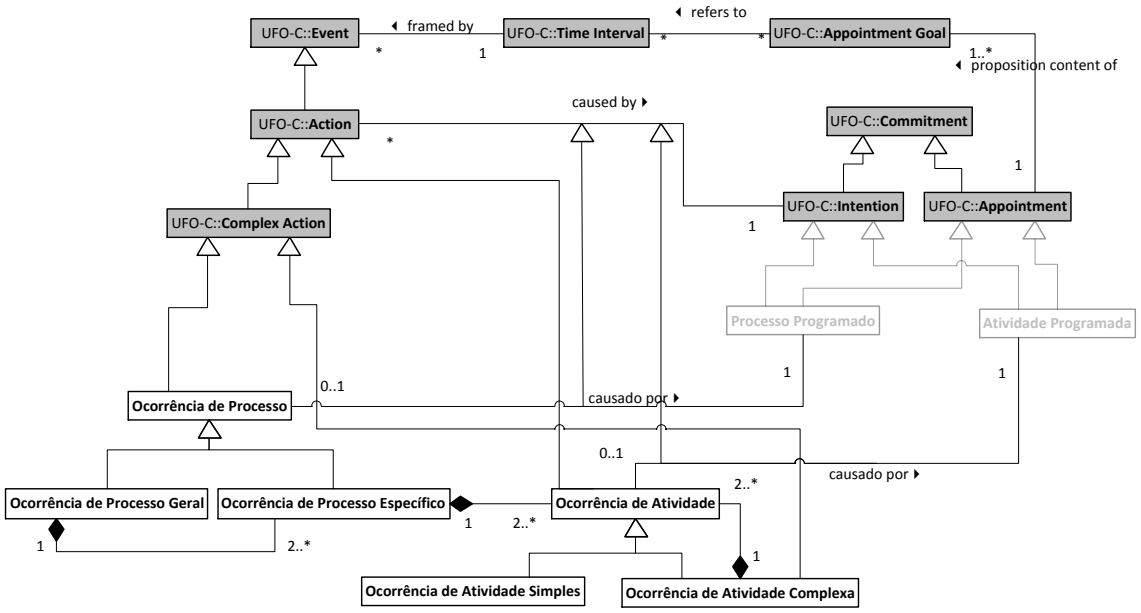

Figura 30 – Ocorrência de Processo e de Atividade.

Assim como as **atividades do projeto** (e, consequentemente as atividades programadas), as ocorrências de atividade podem ser complexas ou simples. Neste caso, as **ocorrências de atividade complexas** são ações complexas (*Complex Action*), pois são compostas de **ocorrências de atividade simples. Ocorrências de atividade simples**, por sua vez, podem ser uma **participação do recurso humano** alocado ou pode ser a **participação de recurso**, onde o recurso pode ser um **equipamento de software** ou um **produto de hardware.**

Para ajudar no entendimento dos tipos de participação, é apresentado um exemplo ilustrativo. Neste exemplo uma organização possui como uma das atividades do processo de desenvolvimento, a atividade de Análise de Requisitos. Esta atividade por sua vez é composta pelas subatividades Levantamento de Requisitos e Modelagem Estrutural. Durante a execução da subatividade Modelagem Estrutural, o Analista de Sistemas Antonio Carlos, que é o recurso humano alocado para esta atividade, utiliza a ferramenta de software Astah* para construir seus modelos. Neste exemplo a ação modelagem estrutural é composta pela contribuição de ação do Antonio Carlos e pelo uso da ferramenta Astah*.

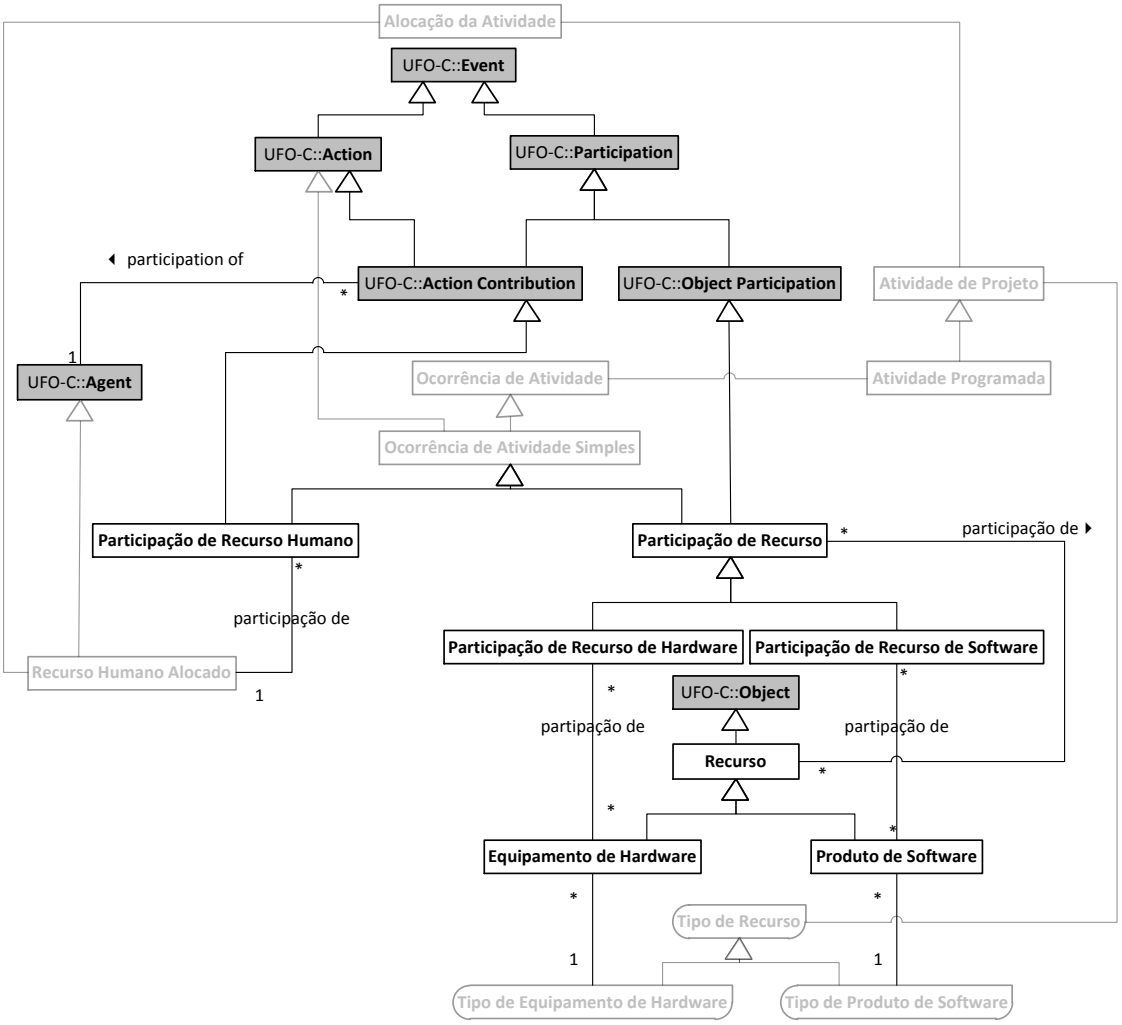

Figura 31 – Participações de Recursos na ocorrência de atividades.

Desta forma, de acordo com a UFO, a **participação do recurso humano** é uma contribuição de ação (*Action Contribution*), pois é a participação intencional de um agente (*Agent*). Em contrapartida, a **participação de recurso**, que é um objeto (*Object*), é uma participação de objeto (*Object Participation*), pois é um objeto sendo usado em uma ação. Capturar essas distinções é muito importante para o controle do projeto, pois quase todas as organizações exigem que os seus recursos humanos registrem o tempo que gastaram para realizar uma determinada tarefa. Participações de recursos humanos, enquanto eventos, ocorrem em um intervalo de tempo e tais distinções permitem capturar este aspecto.

4.3 CONSIDERAÇÕES FINAIS

Neste capítulo, apresentamos os avanços feitos na reengenharia do fragmento da Ontologia de Processo de Software (SPO), originalmente definida em (BERTOLLO, 2006), dando um passo à frente nas distinções feitas em (GUIZZARDI et al., 2008). Esta versão foi obtida a partir do alinhamento entre os conceitos e relações definidas na SPO com os conceitos e as relações da Ontologia de Fundamentação Unificada (UFO). O foco era abordar a conceituação relacionada a processos de software que fossem relevantes para o planejamento de projetos de software.

A Figura 32 mostra o panorama geral[2] da nova versão da Ontologia de Processo de Software, obtida como resultado da reengenharia realizada neste trabalho. Contrastando-a com a versão original da ontologia (Figura 21) ou até mesmo com a versão da ontologia após ter passado pelo primeiro processo de reengenharia (Figura 22), podemos ver que existia muito conhecimento que não estava explícito no modelo.

[2] As multiplicidades do modelo foram omitidas para deixar o diagrama mais limpo.

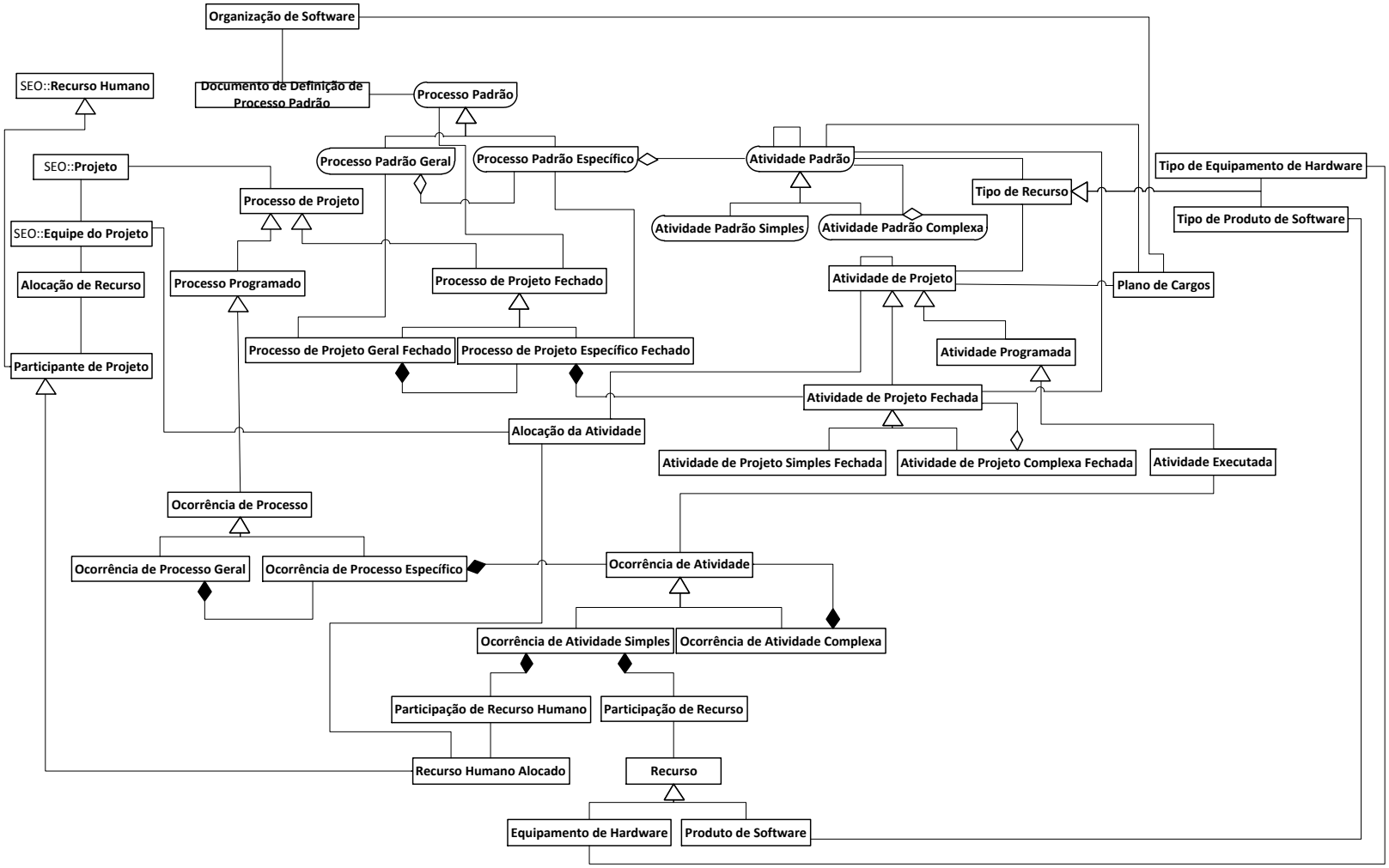

Figura 32 – Versão atual da Ontologia de Processo de Software

O uso de UFO provou ser útil para identificar problemas e para a condução da reengenharia da ontologia, especialmente descrevendo os compromissos ontológicos que estavam implícitos. Tal processo explicitou a diferença entre processo padrão, processo de projeto, processo agendado e ocorrência de processo. Ficou claro que um processo padrão é um plano descrito por uma descrição normativa reconhecida pela organização. Um processo de projeto é um compromisso da equipe do projeto com a organização, enquanto um processo programado é um compromisso que se refere a um intervalo de tempo. Finalmente, as ocorrências do processo são ações complexas. Essas distinções também se aplicam ao nível de atividade. Além disso, tratamos da alocação de recurso humano para a equipe do projeto e posteriormente a alocação dele para uma atividade deste projeto. Por fim, distinguimos as participações dos recursos humanos das participações de recursos de software e de hardware dentro de uma atividade.

O processo de reengenharia na Ontologia de Processo de Software foi feito com o intuito de explicitar os compromissos ontológicos do modelo. Esta ontologia é utilizada no processo de integração semântica de ferramentas de apoio ao planejamento de projetos de software, apresentado no capítulo a seguir.

5 INTEGRANDO CONCEITUALMENTE FERRAMENTAS DE APOIO AO PLANEJAMENTO DE PROJETOS DE SOFTWARE

Um dos objetivos deste trabalho é integrar conceitualmente ao ambiente ODE uma ferramenta que apoie a elaboração de cronogramas, funcionalidade ainda não devidamente tratada pelas ferramentas do ambiente descritas no Capítulo 2. Para apoiar tal função de forma efetiva, o conjunto de ferramentas deve ser capaz de possibilitar que os interessados não apenas manipulem os dados referentes ao cronograma, mas que também possam visualizá-los de forma efetiva.

As ferramentas do ambiente ODE já possibilitam o cadastro das informações necessárias para a elaboração do cronograma do projeto. A ferramenta de Definição de Processo, Def-Pro, permite definir o processo do projeto, ou seja, permite que se estabeleçam as atividades do projeto. Para cada atividade, podem ser definidos os tipos de recursos necessários para a sua realização.

A ferramenta AlocaODE, por sua vez, permite que a equipe do projeto seja definida, bem como que um recurso humano seja alocado para uma atividade específica, definindo o período de alocação. Outros tipos de recurso também podem ser alocados: para cada atividade, é possível estabelecer quais os produtos de software e equipamentos de hardware serão utilizados durante a realização de uma atividade.

Por fim, ControlPro, a ferramenta de acompanhamento de projeto do ambiente ODE, permite o acompanhamento do progresso do projeto e a alteração do processo em tempo de execução. A partir desta ferramenta, é possível controlar o andamento do projeto, bem como definir datas de início e fim planejadas para as atividades do projeto, bem como as datas efetivas de início e fim das ocorrências das atividades.

Pode-se observar que as ferramentas do ambiente proveem os dados necessários para apoiar a definição do cronograma. Entretanto, não existe uma ferramenta que apoie especificamente essa tarefa. Atualmente, para que um interessado tenha acesso ao cronograma do projeto, é necessário que cada

atividade seja consultada individualmente, o que não é prático nem eficaz. Desta forma, a ferramenta externa a ser integrada a ODE deve ser capaz de, tomando por base os dados existentes no ambiente, apoiar a elaboração e acompanhamento do cronograma.

Além dos aspectos funcionais, a ferramenta externa a ser integrada ao ambiente ODE também deve atender a requisitos não-funcionais para estar em linha com os padrões atuais do Projeto ODE. ODE é um sistema desenvolvido para Web, com código aberto e que utiliza a linguagem Java, uma linguagem independente de plataforma. Portanto, a ferramenta selecionada deve respeitar algumas restrições, a saber:

- Deve ser um sistema Web, com o intuito de facilitar a implementação da integração;
- Com relação ao código, ele deve ser aberto, o desenvolvimento deve respeitar o padrão MVC para facilitar a posterior integração e a linguagem deve ser independente de plataforma;
- É desejável que a documentação dos modelos conceituais e do código esteja disponível.

Tomando por base os requisitos funcionais e não-funcionais, selecionou-se a ferramenta Endeavour (http://endeavour-mgmt.sourceforge.net/). Endeavour é um sistema web, de código aberto e desenvolvido em Java (linguagem independente de plataforma), que provê apoio à Gerência de Projetos. O código respeita o padrão MVC, ou seja, separa as camadas de modelo, visão e controle. E, por fim, o modelo conceitual está disponível junto ao código. Endeavour apresenta o cronograma do projeto na forma de um gráfico de Gantt, incluindo as datas de início e fim para cada atividade. Além disso, permite que sejam especificados os tipos de relacionamento de dependência entre as atividades, além de mostrar o andamento do projeto.

Uma vez que as ferramentas foram definidas, é necessário integrá-las conceitualmente. De acordo com OBA-SI, três artefatos são muito importantes na fase de análise da integração, (i) uma ontologia de domínio de referência, (ii) os modelos conceituais das ferramentas a serem integradas e (iii) o modelo do processo de negócio que deverá ser apoiado pelas ferramentas.

No capítulo anterior, reestruturamos a ontologia de processo de software com o intuito de alinhá-la aos conceitos da UFO. A reestruturação foi feita para explicitar os compromissos ontológicos e, dessa forma, refletir com fidedignidade os conceitos do mundo real para ser usada como a ontologia de domínio de referência durante o processo de integração.

Este capítulo apresenta a integração conceitual, a qual limita-se à integração na camada de dados, no nível conceitual, de Endeavour às ferramentas de apoio à Gerência de Projetos de ODE. Ele está estruturado da seguinte forma: a Seção 5.1 apresenta os modelos conceituais das ferramentas a serem integradas; a Seção 5.2 discute os mapeamentos verticais realizados; a Seção 5.3 apresenta o modelo de integração resultante; finalmente, a Seção 5.4 apresenta as considerações finais do capítulo.

5.1 MODELOS CONCEITUAIS ESTRUTURAIS DAS FERRAMENTAS

Um dos artefatos necessários para a integração conceitual, segundo OBA-SI, são os modelos conceituais das ferramentas. O modelo conceitual de cada ferramenta deve ser mapeado para a ontologia de referência (mapeamento vertical). Isso é feito para atribuir semântica aos conceitos das ferramentas a serem integradas.

Uma vez que as ferramentas do ambiente ODE compartilham modelos conceituais, com frequência um conceito presente em uma ferramenta também está presente em outra. Aliás, esta é a premissa do original do Projeto ODE: se as ferramentas são construídas compartilhando uma mesma conceituação, então sua integração é facilitada. Ainda que as ferramentas Def-Pro, AlocaODE e ControlPro compartilhem o mesmo modelo conceitual, para facilitar a leitura, apresentamos o modelo conceitual de forma seccionada, dando ênfase aos conceitos utilizados por cada uma delas.

A seguir são apresentados os modelos conceituais das ferramentas envolvidas no processo de integração.

5.1.1 Modelo Conceitual Estrutural de Def-Pro

A ferramenta de definição de processo Def-Pro é organizada em três pacotes principais: Conhecimento, Processo Padrão e Controle Processo. O primeiro contém classes que representam o conhecimento de uma organização acerca de processos de software, incluindo tipos de processos, tipos de atividade, tipos de recursos etc. O segundo refere-se aos conceitos do nível de processo padrão. Finalmente, o terceiro trata dos conceitos relativos a processos de projeto. A Figura 33 apresenta o diagrama de classes do pacote Processo Padrão. As classes do pacote Conhecimento utilizadas são apresentadas e descritas do contexto dos outros dois pacotes. Elas são sempre iniciadas com o prefixo K.

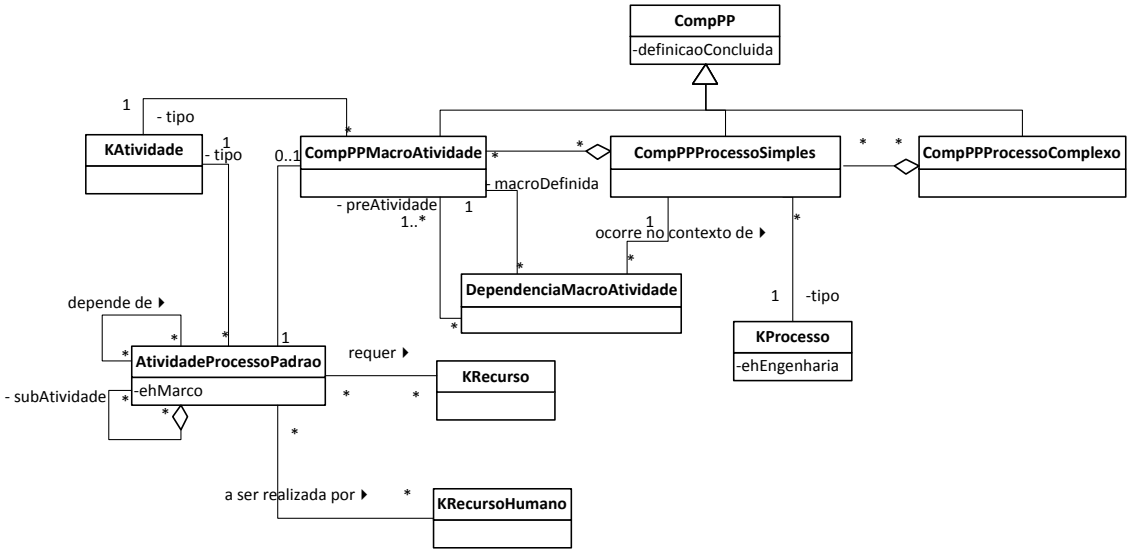

Figura 33 – Modelo conceitual Def-Pro – *Processo Padrão*

Def-Pro utiliza a ideia de definição de processos padrão por meio de componentes. No nível de abstração de processo padrão, os componentes de processos são denominados CompPPs (*CompPP*). Um processo padrão definido para uma organização é dito um processo padrão complexo (*CompPPProcessoComplexo*) e é composto por subprocessos, ditos processos padrão simples (*CompPPProcessoSimples*). Estes, por sua vez, são compostos por macroatividades (*CompPPMacroatividade*). Um CompPP de macroatividade refere-

se a uma atividade padrão (*AtividadeProcessoPadrao*), sendo que essa atividade pode ser decomposta em subatividades padrão.

Para cada atividade do processo padrão, devem ser definidos os tipos de recursos requeridos (*KRecurso*), os tipos de recursos humanos que irão realizar a atividade (*KRecursoHumano*) e a ordem de precedência entre as atividades. Destaca-se que, para os CompPPs de macroatividades, as relações de precedência são definidas sempre no contexto de um CompPP de processo simples. Assim, no contexto de um determinado CompPP de processo simples, um CompPP de macroatividade qualquer pode ser precedido ou preceder outros CompPPs de macroatividade que também compõem o CompPP de processo simples. Essas relações de precedência entre CompPPs de macroatividade no contexto de um CompPP de processo simples são definidas e registradas por meio de conectores de CompPP de macroatividade (*DependenciaMacroAtividade*).

O pacote Controle Processo envolve, dentre outros, a funcionalidade relativa à definição de processos de projeto. A Figura 34 apresenta as classes relevantes para a definição de processos de projeto. Como o pacote Controle Processo é bastante complexo e envolve preocupações que vão além da definição de processos de projeto, aqui apresentamos apenas a porção de classes relevantes para este trabalho.

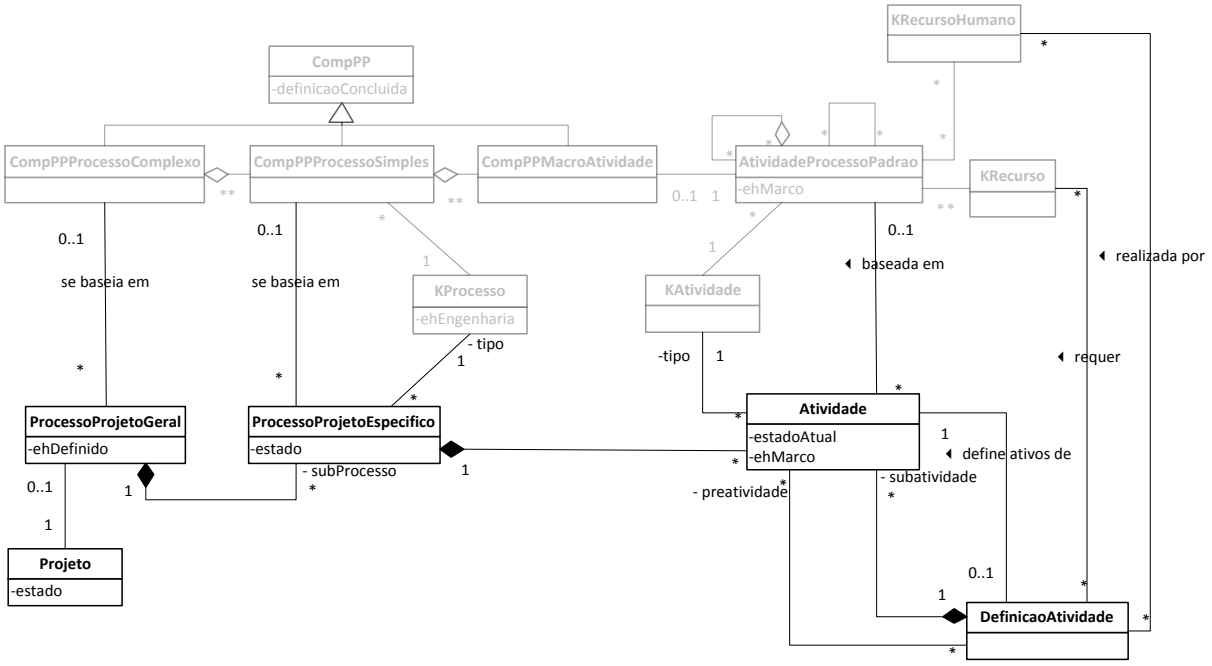

Figura 34 - Modelo conceitual Def-Pro - *Controle Processo*

Um processo de projeto (*ProcessoProjetoGeral*) é sempre definido para um projeto, a partir de um processo padrão complexo (*CompPPProcessoComplexo*). Um processo de projeto geral (*ProcessoProjetoGeral*) pode ser composto por processos de projeto específicos (*ProcessoProjetoEspecifico*). Os processos de projeto específicos possuem um tipo (*KProcesso*) e são compostos por atividades (*Atividade*). Um processo de projeto específico pode ser definido com base em um processo padrão simples.

Uma atividade (*Atividade*) tem um tipo (*KAtividade)* e pode ser originada de uma atividade do processo padrão (*AtividadeProcessoPadrao*). Para uma atividade, é possível definir seus ativos (*DefinicaoAtividade*) , incluindo, dentre outros, os tipos dos recursos necessários para a sua realização (*KRecurso*), os papéis necessários para sua execução (*KRecursoHumano*), a ordem de precedência entre elas e as suas subatividades.

5.1.2 Modelo Conceitual Estrutural de AlocaODE

A ferramenta AlocaODE possibilita que os recursos sejam alocados para um projeto e, posteriormente, para uma atividade. Após um projeto (*Projeto*) ter seu processo de software definido, é possível definir a equipe (*Equipe*) do projeto e alocar recursos para suas atividades. Um recurso humano (*RecursoHumano*) pode desempenhar diversos papéis (*KRecursoHumano*) na equipe em que está alocado (*AlocacaoEquipe*). No que se refere à alocação de recursos humanos em uma atividade específica (*AlocacaoRH*), é possível definir o papel que a pessoa (*Recurso Humano*) desempenhará na atividade, de acordo com os seus papéis desempenhados na equipe (*KRecursoHumano*). Além disso, é possível alocar ferramentas de software (*FerramentaSoftware*) e equipamentos de hardware (*RecursoHardware*) para uma determinada atividade, de acordo com os tipos de recursos (*KRecurso*) de software e de hardware requeridos pela mesma. A Figura 35 mostra um diagrama de classes parcial da alocação de recursos humanos.

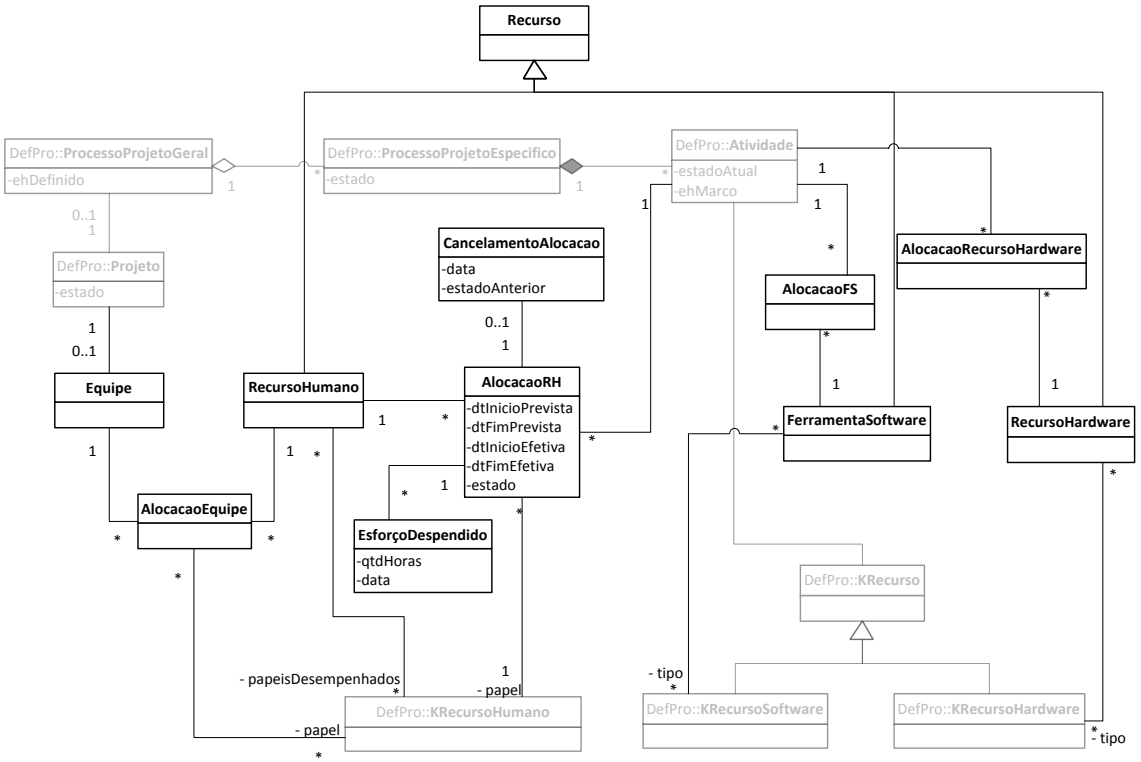

Figura 35 – Modelo conceitual AlocaODE

Quando uma alocação de recurso humano é definida para uma atividade ainda não iniciada, essa alocação está apenas *Definida*. Após uma atividade ser iniciada, todas as alocações de recursos humanos da mesma passam a estar aptas a serem iniciadas. Quando o primeiro esforço despendido (*EsforçoDespendido*) na alocação a uma atividade é registrado, essa alocação passa a estar *Em Andamento*. Uma alocação de recurso humano (*AlocacaoRH*) pode ser cancelada (*CancelamentoAlocacao*), encerrada ou retornada para andamento para que ajustes sejam feitos, indicando que algo ainda precisa ser feito pelo recurso humano naquela atividade. Além disso, é possível anular um cancelamento de uma alocação de recurso humano.

5.1.3 Modelo Conceitual Estrutural de ControlPro

Durante o acompanhamento de um projeto, todas as definições feitas para ele estão sujeitas a alterações. Assim, a ferramenta ControlPro opera sobre todo conjunto de conceitos apresentados nos diagramas de classes das Figura 34 e Figura 35. Ela permite o acompanhamento do progresso do projeto, ou seja, é através dela que um projeto é executado (*ExecucaoProjeto*). Um projeto pode ser iniciado, cancelado ou finalizado. O mesmo ocorre com as atividades do projeto, que têm sua execução (*ExecucaoAtividade*) controlada a partir da ferramenta. Uma atividade pode ser iniciada, finalizada, cancelada, suspensa ou encerrada. A Figura 36 apresenta esses conceitos.

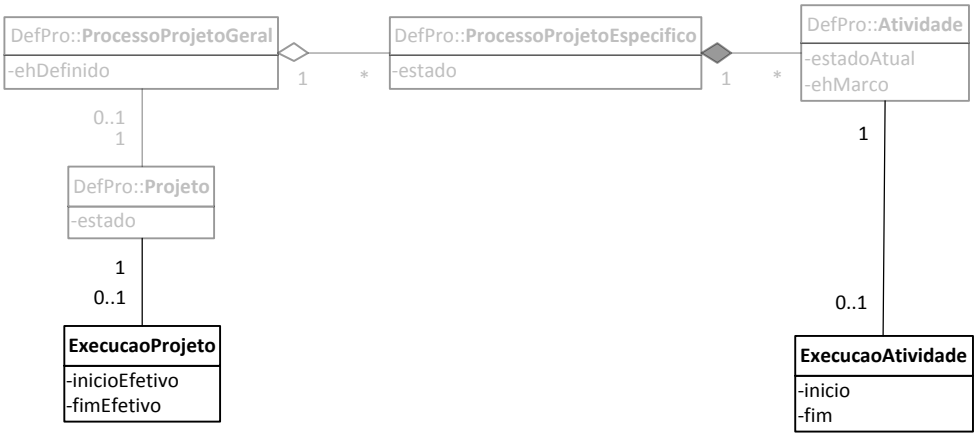

Figura 36 – Modelo conceitual ControlPro

5.1.4 Modelo Conceitual Estrutural de Endeavour

A ferramenta Endeavour foi construída para apoiar a gerência de projetos ágeis. Para cada projeto (*Project*) é possível definir quais tarefas (*Task*) são necessárias para alcançar seus objetivos e quais membros do projeto (*Project Member*) vão atuar nesse projeto. Para cada tarefa, é possível definir um ou mais membros do projeto que serão responsáveis por ela. Além disso, é possível determinar quando a tarefa deve iniciar e terminar, além de permitir que seja registrado o andamento da tarefa. Um membro de projeto pode desempenhar apenas um único papel (*Project Member.role*) e este papel independe dos projetos para o qual está alocado. A Figura 37 mostra um diagrama de classes parcial de Endeavour, apresentando esses conceitos.

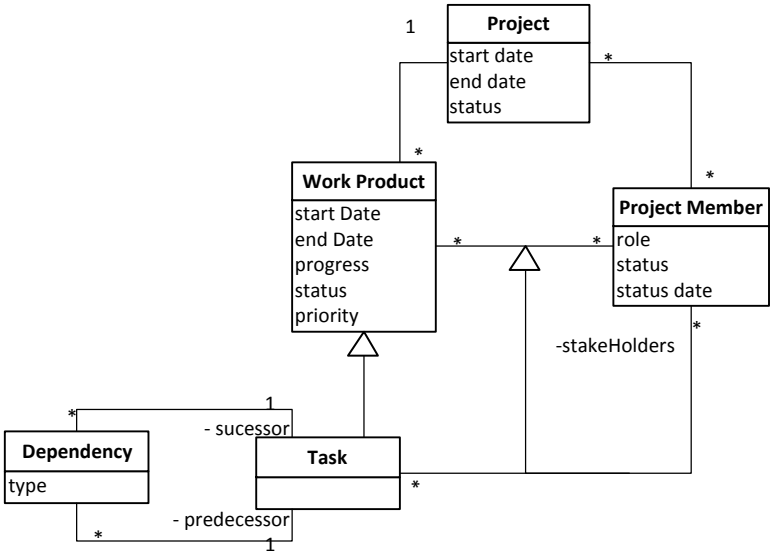

Figura 37 – Modelo Conceitual Endeavour

Uma tarefa pode se relacionar com outras por meio de dependências. Existem quatro tipos de dependência, a saber:

- *Finish-to-Start*: O sucessor (*Sucessor*) não pode iniciar antes do predecessor (Predecessor) terminar.
- *Start-to-Start*: O sucessor não pode iniciar antes do predecessor ter iniciado.
- *Finish-to-Finish*: O sucessor não pode terminar antes do predecessor ter terminado.
- *Start-to-Finish*: O sucessor não pode terminar antes do predecessor ter iniciado.

5.2 MAPEAMENTOS VERTICAIS

Como parte da integração conceitual, é necessário realizar os mapeamentos verticais, ou seja, relacionar os conceitos dos modelos conceituais das ferramentas com os conceitos da ontologia de processo. De acordo com OBA-SI, o objetivo desses mapeamentos é atribuir significado às entidades das ferramentas, relacionando-as com as entidades da ontologia de referência.

Para realizar os mapeamentos, todos os conceitos das ferramentas foram analisados e mapeados para conceitos da ontologia de referência, a SPO. A Tabela 1 apresenta os mapeamentos dos conceitos das ferramentas do ambiente ODE (Def-Pro, AlocaODE e ControlPro) para os conceitos da SPO, a ontologia de referência.

Tabela 1 – Mapeamentos verticais: SPO - ODE

SPO	**ODE**
Def-Pro – Processo Padrão[3]	
-	KProcesso
-	KAtividade
-	CompPPMacroAtividade
-	CompPP
Tipo de Recurso	KRecurso
Papel de Recurso Humano	KRecursoHumano
-	DependenciaMacroAtividade
Processo Padrão Geral	CompPPProcessoComplexo
Processo Padrão Específico	CompPPProcessoSimples
Atividade Padrão	AtividadeProcessoPadrao.sub == null
Atividade Padrão Complexa	AtividadeProcessoPadrao.sub != null
Def-Pro – Controle Processo	
Projeto	Projeto
Processo de Projeto Geral	ProcessoProjetoGeral
Processo de Projeto Específico	ProcessoProjetoEspecifico

[3] A divisão feita na tabela é apenas para melhorar a legibilidade, uma vez que no ambiente ODE os conceitos de uma ferramenta são pertinentes às outras, como discutido anteriormente.

Tabela 1 – Mapeamentos verticais: SPO - ODE

Atividade de Projeto	Atividade. ExecucaoAtividade == null
Atividade de Projeto Complexa	Atividade.subAtividade != null
-	DefinicaoAtividade
AlocaODE	
Equipe de Projeto	Equipe
Recurso Humano	Recurso Humano.AlocacaoEquipe == null
Participante de Projeto	(Recurso Humano.AlocacaoEquipe != null) && (RecursoHumano.AlocacaoRH == null)
Alocação de Recurso	AlocaçãoEquipe
Alocação da Atividade	AlocacaoRH
Recurso Humano Alocado	(RecursoHumano.AlocacaoRH != null)
-	Cancelamento Alocacao
-	AlocacaoFS
-	AlocacaoRecursoHardware
Produto de Software	FerramentaSoftware
Equipamento de Hardware	RecursoHardware
-	EsforçoDespendido
ControlPro	
Atividade Programada	Atividade. ExecucaoAtividade != null && (Atividade.estado == “inativo” || Atividade.estado == “aguardando autorizacao”)
Ocorrência de Atividade	Atividade. ExecucaoAtividade != null && (Atividade.estado == “em execucao” || Atividade.estado == “concluido”)

Podemos observar na Tabela 1 que, em alguns casos, o mesmo conceito do ambiente ODE é mapeado para mais de um conceito na SPO. Para representar estes casos, colocamos as condições que, quando satisfeitas, o mapeamento existe. Por exemplo, quando uma atividade do ambiente ODE tem subatividades (Atividade.subAtividade != null), o conceito **Atividade** é mapeado para o conceito **Atividade de Projeto Complexa** da SPO. Entretanto, quando uma atividade não tem subatividades (Atividade.subAtividade == null) não podemos afirmar, apenas com esta informação, que ela seja simples ou complexa. Assim, neste caso, a atividade é considerada simplesmente uma **Atividade de Projeto** (SPO).

É possível observar ainda que alguns conceitos do ambiente ODE não foram mapeados para nenhum conceito da SPO. Isso ocorre, por exemplo, com o conceito

CompPP, pois a ontologia não foi desenvolvida pensando na componentização do processo de software.

A Tabela 2 apresenta os mapeamentos dos conceitos da ferramenta Endeavour para os conceitos da SPO.

Tabela 2 – Mapeamentos verticais: SPO - Endeavour

SPO	Endeavour
Projeto	Project
-	Work Product
Recurso Humano	Project Member.Project == null
Participante de Projeto	Project Member.Project != null && Project Member.Task == null
Recurso Humano Alocado	Project Member.Task != null
Atividade de Projeto	Task.start Date == null
Atividade de Projeto Programada	Task.start Date != null && Task.status == “Pending”
Ocorrência de Atividade	Task.start Date != null && (Task.status == “In Progress” || Task.status == “Completed” || Task.status == “Reopened”)
“*depende de*”	Dependency
Papel de Recurso Humano	Project Member.role

Analisando os mapeamentos realizados para os dois sistemas (ODE e Endeavour), é possível observar dois tipos de mapeamentos. O primeiro, e mais simples, são os mapeamentos que acontecem de forma direta, ou seja, um conceito da ferramenta é mapeado para exatamente um conceito da ontologia de referência. Isto ocorre, por exemplo, com o conceito **Projeto** do ambiente ODE e com o conceito **Project** da ferramenta Endeavour. Ambos são mapeados para o conceito **Projeto** da SPO.

A segunda forma de mapeamento observado ocorre quando um conceito da ferramenta é mapeado para dois conceitos distintos da ontologia de referência, dependendo dos seus relacionamentos/atributos. Isto ocorre, por exemplo, com o conceito RecursoHumano de ODE. Quando ele não está alocado em uma equipe, ou seja, RecursoHumano.AlocacaoEquipe == null, então o conceito **RecursoHumano** é mapeado para o conceito **Recurso Humano** da SPO. Entretanto se ele estiver

alocado a uma equipe, ou seja, RecursoHumano.AlocacaoEquipe != null, então o conceito recurso humano é mapeado para o conceito Participante de Projeto da SPO. Na ferramenta Endeavour isto ocorre com o conceito **Task**. Quando uma tarefa (**Task**) ainda não começou, ou seja, a sua data de início (**startDate**) é vazia, ela é mapeada para o conceito **Atividade de Projeto** da SPO. Em contrapartida, quando ela já iniciou, ela é mapeada para o conceito **Ocorrência de Atividade** da SPO.

Em alguns casos, ainda, um conceito da ferramenta não foi mapeado para um conceito/relação equivalente na ontologia. Isto ocorre com o conceito **EsforçoDespendido** do ambiente ODE. Apesar de não ter sido mapeado, ele é um conceito importante para o domínio e deve ser representado no modelo de integração. O conjunto de esforços despendidos (**EsforçoDespendido**) de uma alocação de recurso humano compõe a participação desse recurso humano na correspondente atividade (**Participação de Recurso Humano** em SPO). Neste caso, o modelo de integração deve considerar tais conceitos. A seção seguinte apresenta o modelo de integração.

5.3 MODELO DE INTEGRAÇÃO

O modelo de integração é o principal artefato produzido durante a fase de análise de integração. Ele captura a imagem global do cenário de integração. O modelo de integração é construído com base nos mapeamentos verticais. De fato, o objetivo principal do modelo de integração é estabelecer mapeamentos horizontais, ou seja, o mapeamento entre os modelos das ferramentas que estão sendo integradas.

Com base nos mapeamentos feitos entre as ferramentas e a ontologia de referência, a SPO, criou-se o modelo de integração. A seguir apresentamos o modelo de integração, dividido em porções para facilitar o entendimento.

i. Modelo de Integração referente à Definição de Processos Padrão

O modelo apresentado na Figura 38 mostra uma porção que captura os conceitos utilizados pela ferramenta DefPro. Os conceitos em destaque são os conceitos das ferramentas que não são mapeados para um conceito da ontologia de referência. Tais conceitos reforçam a necessidade do modelo de integração, pois, caso fosse usada apenas a ontologia de referência, não seria possível capturar esses conceitos. Desta forma, esses conceitos são incorporados ao modelo de integração com o intuito de diminuir a distância entre esse modelo e o modelo das ferramentas.

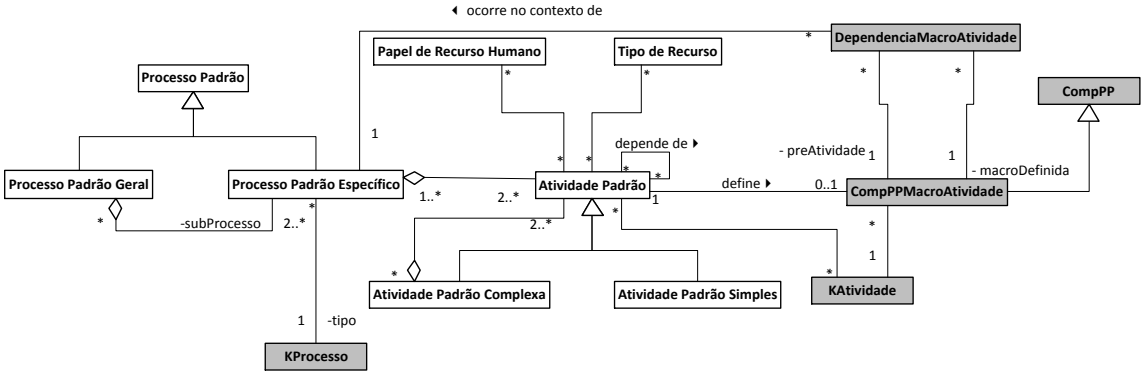

Figura 38 - Modelo de Integração – Definição de Processo Padrão

Uma **KAtividade** define o tipo da **atividade padrão**, assim como um **KProcesso** define o tipo do **Processo Padrão Específico**. Um **CompPPMacroAtividade** é definido por uma **AtividadePadrão** e também tem um tipo associado a ele (**KAtividade**).

ii. Modelo de Integração referente à Definição de Processos de Projeto

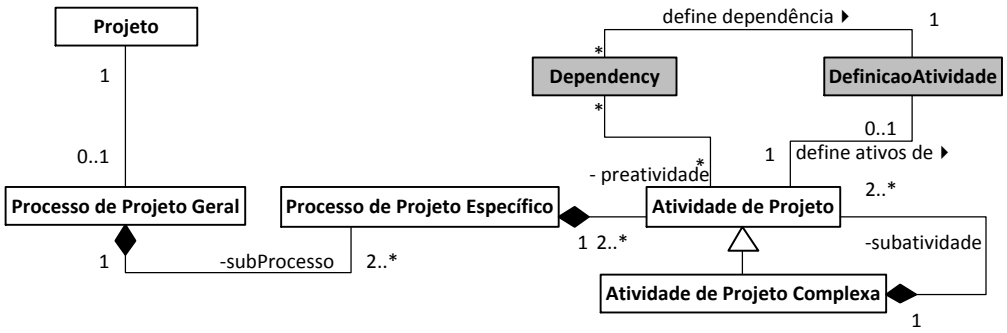

Figura 39 - Modelo de Integração – Definição de Processo de Projeto

A ontologia SPO, assim como o ambiente ODE, permite que, para cada atividade (**Atividade de Projeto**), sejam determinadas de quais atividades ela depende. Entretanto, na ontologia não existe distinção entre os tipos de dependência, como faz a ferramenta Endeavour. Desta forma, o conceito **Dependency** da ferramenta Endeavour foi adicionado ao modelo de integração para permitir que esta informação seja capturada.

iii. Modelo de Integração referente à Alocação de Recurso

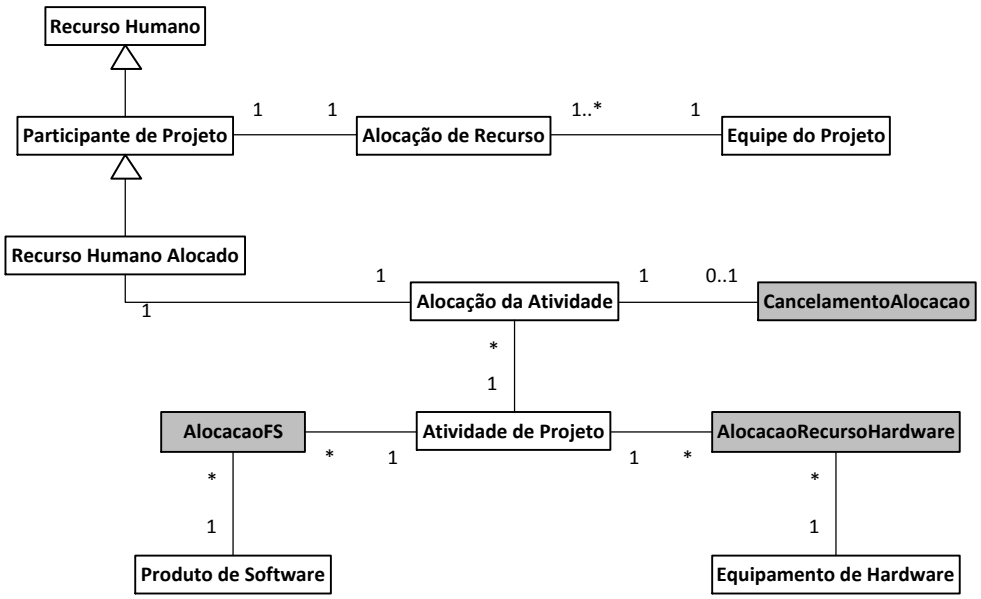

Figura 40 - Modelo de Integração – Alocação de Recurso

Uma atividade, além de poder ter recursos humanos alocados, pode ter também produtos de software e equipamentos de hardware a ela alocados. Desta forma, foram adicionados os conceitos **AlocacaoFS** e **AlocacaoRecursoHardware** do ambiente ODE. Uma alocação de recurso humano pode, ainda, ser cancelada. Para representar isso, foi adicionado o conceito **CancelamentoAlocacao**.

iv. Modelo de Integração referente à Execução do Projeto

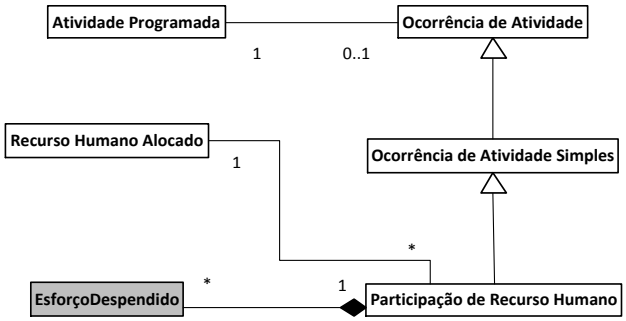

Figura 41 – Modelo de Integração – Execução da Atividade

Durante a execução de uma atividade, é necessário registrar o esforço despendido nela. Para tal, foi adicionado ao modelo o conceito EsforçoDespendido. Note que, apesar deste conceito ser da ferramenta AlocaODE, o registro de esforço só acontece quando uma atividade está sendo executada.

O modelo de integração é o produto principal da fase de análise de OBA-SI. Ele foi construído a partir dos mapeamentos feitos entre as ferramentas e a ontologia de referência. Os conceitos que não puderam ser mapeados para a ontologia de referência foram adicionados ao modelo de integração. Isto ocorre, por exemplo, com o conceito **CompPP**. Em contrapartida, os conceitos da ontologia que não são utilizados pelas ferramentas foram removidos do mesmo, como aconteceu com o conceito **Plano de Cargos**. Desta forma, a diferença semântica entre o modelo de integração e os modelos das ferramentas tornou-se menor. A Figura 42 apresenta o modelo de integração completo[4].

[4] As multiplicidades do modelo foram omitidas para deixar o diagrama mais limpo.

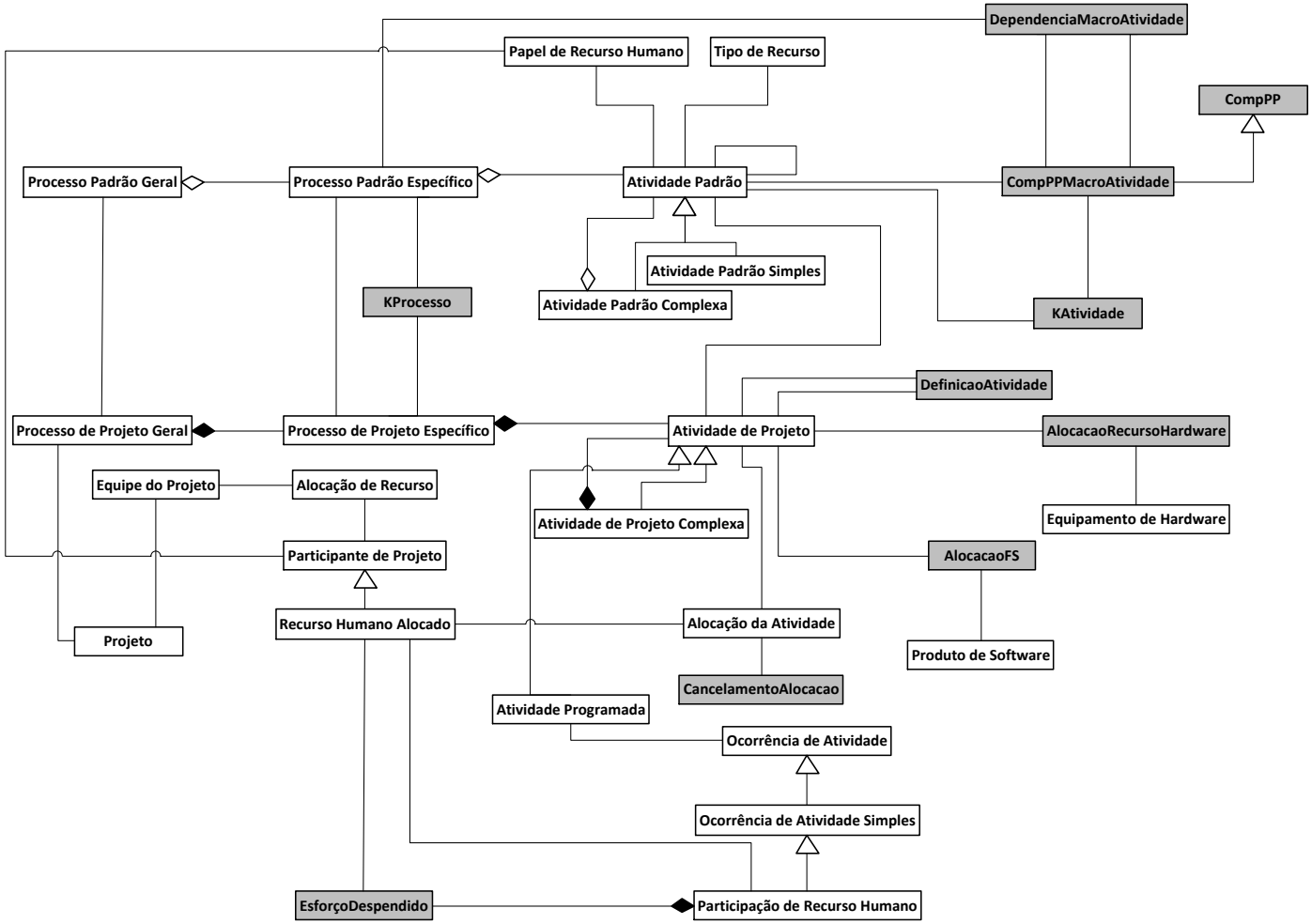

Figura 42 - Modelo de Integração

5.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o processo de integração semântica das ferramentas do ambiente ODE e da ferramenta Endeavour. Para tal utilizou-se a abordagem OBA-SI. OBA-SI defende que a integração semântica deve acontecer, primeiro, no nível conceitual. Nesta visão, o processo de integração é comparado ao processo de desenvolvimento de software, onde, antes de efetivamente integrar, é necessário levantar os requisitos da integração, analisá-los para só depois considerar aspectos tecnológicos.

Desta forma, para a análise da integração, após os modelos conceituais das ferramentas terem sido obtidos, determinou-se os mapeamentos verticais com o intuito de adicionar semântica para os elementos desses modelos. Foi observado que os mapeamentos verticais podem ocorrer, pelo menos, de duas formas distintas. Como dito anteriormente, uma delas é quando o mapeamento é direto, ou seja, quando exatamente um conceito da ferramenta é mapeado para um conceito da ontologia. A outra forma ocorre quando um conceito da ferramenta é mapeado para mais de um conceito na ontologia, dependendo do seu estado (conjunto de atributos e relações). Este último ocorre quando a ferramenta colapsa dois conceitos em um só e por isso é mais delicado de ser tratado, pois o conceito tem que ser analisado de acordo com os possíveis contextos em que ele pode estar inserido.

Pode ocorrer, ainda, de um conceito não ser mapeado para um conceito da ontologia. Isto pode ocorrer por uma escolha de modelagem diferente da feita na ontologia de referência, ou pode ser um reflexo da abrangência da ontologia de referência, já que esta é tipicamente um modelo consensual e, por isso, muitas vezes não cobre todas as particularidades tratadas por uma ferramenta. Em ambos os casos, é necessário analisar como esses elementos devem ser tratados dentro do modelo de integração.

Por fim, a importância da reengenharia da ontologia pode ser constatada ao verificar que nos mapeamentos horizontais foram utilizados conceitos que nas versões anterior da SPO não existiam, como é o caso de **Atividade Programada**,

que é um conceito que surgiu na versão da SPO desenvolvida no contexto deste trabalho.

6 CONCLUSÃO

Atualmente, tem crescido o interesse na área de interoperabilidade. Este crescimento é devido à necessidade dos sistemas interoperarem, de forma a apoiarem os objetivos das organizações. Como consequência desse crescente interesse, surgiram abordagens que visam tratar este problema.

De maneira geral, a integração de sistemas pode ocorrer em três camadas diferentes: camada de dados, camada de mensagens ou serviços e camada de processo. Ela pode acontecer, ainda, em diferentes níveis: nível de hardware, nível de plataforma, nível sintático e nível semântico (IZZA, 2009).

No nível semântico, foco deste trabalho, durante o processo de integração, o significado dos componentes envolvidos deve ser o mais claro possível, ou seja, o significado pretendido dos conceitos no esquema de dados, nas assinaturas das operações e serviços deve ser explicitado. Desta forma, interoperabilidade semântica é a capacidade de dois ou mais sistemas heterogêneos e distribuídos trabalharem em conjunto, compartilhando informações com entendimento comum de seu significado.

Neste contexto, uma ontologia de domínio pode ser utilizada para definir uma representação explícita dessa conceituação compartilhada. Contudo, para servir adequadamente como um modelo de referência, uma ontologia de domínio deve ser construída usando uma abordagem que leve em conta explicitamente conceitos de uma ontologia de fundamentação.

Este trabalho aplicou a abordagem OBA-SI, uma abordagem de integração semântica baseada em ontologias, no domínio de planejamento, controle e acompanhamento de projetos de software. Devido à complexidade desse domínio e ao fato de suas atividades serem iterativas e interrelacionadas (PMI, 2008), quando uma organização utiliza mais de uma ferramenta para apoiar o planejamento de projetos, idealmente elas devem interoperar para que sejam eficazes na sua função.

Este trabalho teve seu objetivo principal integrar conceitual e semanticamente ferramentas de apoio ao processo de Gerência de Projetos de Software na camada de dados. Durante o processo de integração, foi utilizada uma ontologia de processo de software, a SPO (*Software Process Ontology*). Ela foi utilizada para adicionar semântica aos conceitos das ferramentas envolvidas no cenário de integração.

Enquanto uma ontologia de domínio de referência, o ideal é que a SPO fosse construída baseada em uma ontologia de fundamentação. Apesar de parte da SPO ter passado por um processo de reengenharia para alinhar os seus conceitos com uma ontologia de fundamentação, a UFO (*Unified Foundational Ontology*), em (GUIZZARDI et al. 2008), a porção necessária para o domínio da integração ainda não estava satisfatoriamente fundamentada. Desta forma, foi dado um passo à frente na reengenharia da SPO baseada na UFO antes de se iniciar o processo de integração em si.

No que se refere ao cenário de integração, parte das ferramentas, as ferramentas do ambiente ODE, estava definida a priori. Entretanto a ferramenta externa ao ambiente não. Desta forma foi necessário selecionar uma ferramenta que apoiasse as atividades que as ferramentas do ambiente não apoiavam, a saber as atividades de definição e acompanhamento de cronograma. Para tal, a ferramenta Endeavour foi selecionada.

Em suma, são contribuições deste trabalho:

- ***Reengenharia Parcial da Ontologia de Processo de Software***: Foi feita uma reengenharia da porção da Ontologia de Processo de Software envolvendo os conceitos relevantes para o planejamento, acompanhamento e controle de projetos. Essa reengenharia foi baseada na UFO.
- ***Integração Semântica das ferramentas de apoio ao planejamento de software***: as ferramentas de apoio ao planejamento, acompanhamento e controle de projeto foram conceitualmente integradas, no nível semântico e na camada de dados. O processo de integração seguiu a abordagem OBA-SI e utilizou a versão proveniente da reengenharia da

Ontologia de Processo de Software como a ontologia de referência. Como artefato final, foi produzido o modelo conceitual estrutural de integração.

Durante o processo de reengenharia da ontologia foram utilizados, em sua maioria, conceitos da UFO-C e alguns conceitos da UFO-B. Os conceitos da UFO-B, assim como os da UFO-A estão bem definidos, o que facilita o entendimento e, consequentemente, o uso desses conceitos. Entretanto isto não ocorre com a UFO-C. Este fragmento da ontologia de fundamentação ainda está sendo desenvolvido e vários de seus conceitos e relações não estão bem definidos. Esta dificuldade é potencializada pelo fato de não existir uma publicação que se proponha a mostrar informações deste fragmento como um todo. Desta forma, o estudo de UFO-C é feito baseado em artigos distintos e que nem sempre estão coesos.

Antes de iniciar o processo de integração das ferramentas foi necessário selecionar a ferramenta externa ao ambiente ODE. Entretanto, uma das expectativas com relação a OBA-SI, era que essa abordagem desse algum tipo de orientação nesta atividade, uma vez que ela ressalta a importância de existir uma fase de análise antes da integração em si. A falta de orientação abre a possibilidade para que o usuário da abordagem selecione o sistema considerando pontos pouco relevantes, por exemplo, apenas aspectos tecnológicos. Entretanto, é possível levantar questões que devem ser consideradas na maior parte dos casos, de maneira formal ou não, tais como: Quais os requisitos que o conjunto de ferramentas integradas deve atender? Quais são os requisitos atendidos pelas ferramentas já definidas? Quais os requisitos funcionais que a ferramenta a ser selecionada deve atender? E os requisitos não funcionais?

6.1 TRABALHOS FUTUROS

A integração semântica apresentada neste trabalho ocorreu no nível semântico e conceitual da camada de dados. Como continuação deste trabalho sob a perspectiva da iniciativa de integração, existem duas possíveis frentes de trabalho

principais. A primeira é dar prosseguimento à iniciativa de integração em si, buscando obter, como resultado final, as ferramentas do ambiente ODE integradas à ferramenta Endeavour na camada de dados. A outra possibilidade é considerar a integração nas camadas de serviço e processo. Neste caso, daria continuidade ao processo de integração conceitual, para só então integrá-las efetivamente (projeto e implementação de uma solução de integração). Para apoiar esta continução, sugere-se seguir a nova versão de OBA-SI, descrita em (CALHAU, 2011).

No que se refere à Ontologia de Processo de Software é necessário avançar na sua reengenharia. A reengenharia feita no contexto deste trabalho teve alguns aspectos limitantes. O primeiro deles diz respeito ao domínio que a ontologia deveria abranger. Ou seja, o processo de reengenharia focou apenas nos conceitos relacionados ao planejamento e controle de projetos, mais especificamente a definição de processos, planejamento de tempo e alocação de recursos. Entretanto uma vez que a SPO abrange outros aspectos do domínio de processo de software, a sua reengenharia também deve abranger. Além desse aspecto limitante, o processo de integração semântica apontou algumas falhas na ontologia, como, por exemplo, o fato da ontologia não considerar as relações de Allen (ALLEN, 1983), apesar destas estarem presentes na UFO e serem importantes para o domínio. Essas relações permitiriam fazer distinções semanticamente mais ricas do que as propostas na ferramenta Endeavour. Essas falhas foram evidenciadas durante a construção do modelo de integração. Conceitos e relações dos modelos conceituais que tiveram que ser adicionados ao modelo de integração apontam possíveis conceitos que deveriam ser tratados pela ontologia e não foram.

Como discutido anteriormente, OBA-SI não trata do processo de escolha das ferramentas a serem integradas. Desta forma, uma melhoria para OBA-SI seria incorporar diretrizes para a seleção dos sistemas a serem integrados, como, por exemplo, a exploração dos objetivos da organização para o levantamento dos requisitos funcionais das ferramentas.

Por fim, uma linha de pesquisa interessante é melhorar as distinções dos problemas semânticos apresentados por Pokraev (2009) tomando por base a UFO-A. Para cada um dos problemas apresentados por ele, seria interessante investigar

quais os possíveis motivos do conflito semântico. Por exemplo, o conceito **Recurso Humano** da SPO pode ser especializado em **Participante de Projeto**, que é um papel social de **Recurso Humano**. No ambiente ODE esses dois conceitos são colapsados no conceito **RecursoHumano**. A ideia é que, a partir da melhoria das distinções dos problemas semânticos, fossem propostos padrões de solução para cada tipo de problema.

REFERÊNCIAS

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

ARBAOUI, S., et al. A **Comparative Review of Process-Centered Software Engineering Environments.** Annals of Software Engineering 14, 311–340, 2002.

BARCELLOS M. P.; FALBO R.A. **Using a Foundational Ontology for Reengineering a Software Enterprise Ontology**. 2009.

BERTOLLO, G. **Definição de Processos em um Ambiente de Desenvolvimento de Software**. Dissertação de Mestrado (Mestrado em Informática), Universidade Federal do Espírito Santo, Vitória - Brasil, Junho 2006.

BERTOLLO, G., SEGRINI, B., FALBO, R. A. **Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias**. In: V Simpósio Brasileiro de Qualidade de Software, 2006, Vila Velha, Brasil. Anais...2006. p. 72-86.

BOURAS, A. et al. **Business Process Fusion based on Semantically-enabled Service-Oriented Business Applications**", In: Interoperability for Enterprise Software and Applications Conference (I-ESA´06), March 2006, Bordeaux, France. Great Britain: ISTE, pp. 157–168.

BURANARACH, M. **The Foundation for Semantic Interoperability on the World Wide Web**. PhD Thesis. Department of Information Science and Telecommunications School of Information Sciences, University of Pittsburgh, Novembro, 2001.

CALHAU, R. F; FALBO, R. A. **An Ontology-based Approach for Semantic Integration**. EDOC 2010.

CALHAU, R. F; **Uma Abordagem Baseada em Ontologias para a Integração Semântica de Sistemas**. Dissertação (Mestrado em Informática) – Programa de

Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, 2011 (em elaboração).

COELHO, A. G. N. C. A. **Apoio à Gerência de Recursos em ODE**. Monografia (Graduação em Ciência da Computação) – Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, 2007.

DAL MORO, R. **CONTROLPRO:** Uma Ferramenta de Apoio ao Acompanhamento de Projetos de Software em ODE. Monografia (Graduação em Ciência da Computação) – Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, 2005.

FALBO, R. A., NATALI, A.C.C., MIAN, P.G., BERTOLLO, G., RUY, F.B., **ODE: Ontology-based software Development Environment**, Proceedings of the IX Congreso Argentino de Ciencias de la Computación, p. 1124-1135, La Plata, Argentina, October 2003.

FALBO, R.; BERTOLLO, G. **A software process ontology as a common vocabulary about software processes**. International Journal of Business Process Integration and Management (IJBPIM), v. 4, p. 239-250, 2009.

FALBO, R.A., RUY, F. B., PEZZIN, J., MORO, R. D. **Ontologias e Ambientes de Desenvolvimento de Software Semânticos**. Actas de las IV Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, JIISIC'2004, Volumen I, pp. 277-292. Madrid, España, Noviembre 2004

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

GRUBER, T.R., **Towards principles for the design of ontologies used for knowledge sharing**, *Int. J. Human-Computer Studies*, v. 43, n. 5/6. 1995.

GRUHN, V. **Process-Centered Software Engineering Environments - A Brief History and Future Challenges**. Annals of Software Engineering 14, 363–382, 2002.

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

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

GUIZZARDI, G. **Theoretical Foundations and Engineering Tools for Building Ontologies as Reference Conceptual Models**. 2010.

GUIZZARDI, G., **The Role of Foundational Ontology for Conceptual Modeling and Domain Ontology Representation.** Keynote Paper, Proceedings of 7th International Baltic Conference on Databases and Information Systems (DB&IS), Vilnius, IEEE Press, 2006

GUIZZARDI, G.; FALBO, R. A.; GUIZZARDI, R. S. S. **The Importance of Foundational Ontologies for Domain Engineering: The case of the Software Process Domain** (A importância de Ontologias de Fundamentação para a Engenharia de Ontologias de Domínio: o caso do domínio de Processos de Software – in portuguese). IEEE Transactions Latin America, 2008.

GUIZZARDI, G.; WAGNER, G. **Some Applications of a Unified Foundational Ontology in Business Modeling**. In Rosemann, M. and Green, P., editors, Ontologies and Business Systems Analysis, pages 345–367. Idea Group, London, UK. 2005.

GUIZZARDI, R. S. S. **Agent-Oriented Constructivist Knowledge Management.** PhD thesis, University of Twente, The Netherland. 2006.

HALLER, A., GOMEZ, J.M., BUSSLER, C., **Exposing Semantic Web Service Principles in SOA to solve EAI Scenarios**, Workshop on Web Service Semantics - WWW 2005, Chiba, Japan. 2005.

ISO, ISO 10006: Quality management systems – Guidelines for quality management in projects, Second edition, 2003.

ISO/IEC, ISO/IEC 12207: Systems and Software Engineering – Software Life Cycle Processes, Second edition, 2008.

IZZA, S., **Integration of industrial information systems: from syntactic to semantic integration approaches**. Enterprise Information Systems,3:1,1 – 57. 2009.

IZZA, S., VINCENT, 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. Geneva, Switzerland: 2005.

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.

PIANISSOLA, T. L., **Uso de Serviços Semânticos para Apoiar a Identificação de Recursos Humanos Baseada em Competências**. Monografia (Graduação em Ciência da Computação) – Departamento de Informática, Universidade Federal do Espírito Santo, 2007.

PMI, **A guide to the project management body of knowledge**, 4 ed, 2008.

POKRAEV, S. V., **Model-Driven Semantic Integration of Service-Oriented Applications**. 2009. Pg 20.

POKRAEV, S; QUARTEL, D.; STEEN, M. W. A.; REICHERT, M; **Semantic Service Modeling: Enabling System Interoperability**, Proc. Int'l Conf. on Interoperability for Enterprise Software and Applications (I-ESA'06), Bordeaux, France, March 2006.

SEARLE, J., **Mind, Language and Society**, Basic Books, 2000.

SEGRINI, B. M. **Definição de Processos de Baseada em Componentes**. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, 2009.

SEGRINI, B. M., BERTOLLO, G., FALBO, R. A. **Evoluindo a Definição de Processos de Software em ODE**. In: XX Simpósio Brasileiro de Engenharia de Software, 2006, Florianópolis, Brasil. Anais da XIII Sessão de Ferramentas do SBES 2006, 2006. p. 109-114

SEGRINI. B. M. **Definição de Processos em um Ambiente de Desenvolvimento de Software**. Dissertação de Mestrado, Mestrado em Informática, UFES.

SEI, CMMI for Development, Version 1.3, Technical Report CMU/SEI-2010-TR-033 November, 2010.

SERAIN, D., **Middleware and Enterprise Application Integration**, Springer, 2002.

SOFTEX, MPS.BR – **Melhoria de Processo do Software Brasileiro: Guia Geral**, Versão 1.0, disponível em www.softex.br/mpsbr, 2009.

TEKTONIDIS, D., BOKMA, A., OATLEY, G., SALAMPASIS, M.; **ONAR - An Ontology-based Service Oriented Application Integration Framework**, In: Proceedings of the 1st International Conference on Interoperability of Enterprise Software and Applications, Geneva, Switzerland, February 2005. London: Springer-Verlag, 65–74.

USCHOLD, M., GR¨UNINGER, M., **Ontologies**: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

VERNADAT, F.B., 1996. **Enterprise modelling and integration**: Principles and applications. London: Chapman & Hall.

VILLELA, K., ROCHA, A.R., TRAVASSOS, G.H. et al.: **The Use of an Enterprise Ontology to Support knowledge Management in Software Development Environments**. Journal of the Brazilian Computer Society, Porto Alegre, Brazil, 11(2): 45-59 (2005).

WEGNER, P., 1996. **Interoperability. ACM Computing Survey**, 28(1), 285-287.

ZAMBORLINI, V. C. **Estudo De Alternativas De Mapeamento De Ontologias Da Linguagem Ontouml Para Owl: Abordagens Para Representação De Informação Temporal**. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, 2011.