UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
MESTRADO EM INFORMÁTICA

PAULO SÉRGIO DOS SANTOS JÚNIOR

UMA ABORDAGEM DE DESENVOLVIMENTO BASEADA EM MODELOS DE ARQUITETURA ORGANIZACIONAL DE TI: DA SEMÂNTICA AO DESENVOLVIMENTO DE SISTEMAS

VITÓRIA, DEZEMBRO 2009

PAULO SÉRGIO DOS SANTOS JÚNIOR

UMA ABORDAGEM DE DESENVOLVIMENTO BASEADA EM MODELOS DE ARQUITETURA ORGANIZACIONAL DE TI: DA SEMÂNTICA AO DESENVOLVIMENTO DE SISTEMAS

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

VITÓRIA, DEZEMBRO 2009

PAULO SÉRGIO DOS SANTOS JÚNIOR

UMA ABORDAGEM DE DESENVOLVIMENTO BASEADA EM MODELOS DE ARQUITETURA ORGANIZACIONAL DE TI: DA SEMÂNTICA AO DESENVOLVIMENTO DE SISTEMAS

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

Aprovada em 16 de dezembro de 2009.

COMISSÃO EXAMINADORA

_____

Prof. Dr. João Paulo Andrade Almeida
Universidade Federal do Espírito Santo (UFES)
(Orientador)

_____

Prof. Dr. Giancarlo Guizzardi
Universidade Federal do Espírito Santo (UFES)
(Co-orientador)

_____

Prof. Dr. Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo (UFES)

_____

Profa. Dra. Lucinéia Heloísa Thom
Universidade Federal da Universidade do Rio Grande do Sul (UFRGS)

VITÓRIA, DEZEMBRO 2009

DEDICATÓRIA

Dedico esta dissertação

a meus pais, Paulo e Rosileni

e a todos que me ajudaram nessa jornada.

AGRADECIMENTOS

RESUMO

O gerenciamento das organizações é uma tarefa que envolve um nível de complexidade significante, uma vez que agrega diversos domínios de conhecimento (incluindo processo de negócios, tecnologias da informação e infra-estrutura). Para que seja possível analisar como esses fatores estão interconectados entre si e como a priorização de um deles pode ocasionar a postergação de outro, a utilização de uma Arquitetura Organizacional de TI (*Enterprise Architecture*) torna-se necessária.

Através de uma Arquitetura Organizacional de TI é possível: **(i)** capturar a essência e as evoluções do negócio e dos sistemas de informação presentes na organização; e **(ii)** realizar de maneira mais eficaz e menos onerosa o alinhamento entre tecnologia da informação e os processos de negócios pertencentes a uma ou mais organizações.

Diversas abordagens de Desenvolvimento Orientado a Modelos utilizam os modelos que representam as Arquiteturas Organizacionais de TI para o desenvolvimento de sistemas computacionais. Porém, a grande maioria das abordagens não permite que o projetista do sistema **(i)** explore a riqueza semântica dos diversos modelos da Arquitetura Organizacional de TI (além dos modelos de processos de negócio), (**ii**) parametrize transformações pré-definidas nessas abordagens para adequar as transformações ao sistema sendo desenvolvido; e **(iii)** divida o processo de desenvolvimento em etapas independente e dependente de plataforma.

Este trabalho propõe uma nova abordagem de Desenvolvimento Orientado a Modelos que visa mitigar estas limitações do estado-da-arte, através de transformações parametrizadas que exploram a riqueza semântica dos modelos das Arquiteturas Organizacionais de TI. Adicionalmente, a abordagem propõe uma divisão clara das etapas independente e dependente de plataforma.

Palavras-chave: Arquitetura Organizacional de TI, Análise Semântica, Desenvolvimento Orientado a Modelos.

ABSTRACT

The management of organizations is a highly complex activity, since it requires the use of knowledge from several knowledge domains (including business process, information technology and infrastructure). In order to analyze how these domains are interrelated an Enterprise Architecture is essential.

The main benefits of using Enterprise Architecture are: **(i)** to capture the essence and evolution of a business and its information systems; and **(ii)** to manage the alignment between the business and information systems in a cost-effective manner, possibly by revealing how business processes and information systems are interrelated.

To address the alignment between business processes and information systems, several Model-Driven Development (MDD) approaches enable designers to derive process-oriented systems directly from the business process models through automatic transformation. However, most of these approaches do not enable designers to explore the semantic richness of many Enterprise Architectural models (using only the control flow of business process models), and further define rather inflexible transformations. Moreover, many of these approaches to not clearly separate the development process into platform-independent and platform-specific steps, polluting business process models with platform concerns.

This work proposes a novel Model-Driven Development approach to address the aforementioned issues. This approach enable the designers to **(i)** profit from the semantics of Enterprise Architecture models throughout the system development process; **(ii)** to apply parameterization in pre-defined transformations and; **(iii)** to clearly divide the development process into platform-independent and platform-specific steps.

Keywords: Enterprise Architecture, Semantic Analysis, Model-Driven Development.

SUMÁRIO

CAPÍTULO 1 INTRODUÇÃO 2

1.1 Introdução 2

1.2 Motivação 3

1.3 Objetivo 6

1.4 Metodologia e Histórico do Desenvolvimento do Trabalho 7

1.5 Organização 9

CAPÍTULO 2 REFERENCIAL TEÓRICO 11

2.1 Introdução 11

2.2 Arquiteturas Organizacionais de TI 11

2.3 Ontologias 15

2.4 Desenvolvimento Orientado a Modelos 34

2.5 Sistemas de Informação Orientados a Processos de Negócio 39

2.6 Conclusão 44

CAPÍTULO 3 METAMODELOS DAS LINGUAGENS DO ARIS METHOD 45

3.1 Introdução 45

3.2 Metamodelo da Linguagem Organizacional 47

3.3 Metamodelo da Linguagem de Processo de Negócio 58

3.4 Metamodelo da Linguagem de Detalhamento de Atividades 68

3.5 ARIS Object 75

3.6 Discussão sobre Elementos Notacionais no ARIS Method 76

3.7 Discussão sobre os Metamodelos de Domínio do ARIS Method 77

3.8 Conclusão 78

CAPÍTULO 4 ANÁLISE ONTOLÓGICADAS LINGUAGENS DO ARIS METHOD . 80

4.1 Introdução 80

4.2 Abordagem de Análise Ontológica 81

4.3 Análise Ontológica do Metamodelo Organizacional 82

4.4 Análise Ontológica do Metamodelo da Linguagem de Processo de Negócio 116

4.5 Análise Ontológica do Metamodelo de Detalhamento de Atividades 128

4.6 Discussão Sobre Análise Ontológica e Trabalhos Correlatos.....132

4.7 Conclusão.....134

CAPÍTULO 5 UMA ABORDAGEM DE DESENVOLVIMENTO ORIENTADO A MODELOS UTILIZANDO ARTEFATOS DE ARQUITETURA ORGANIZACIONAL DE TI 135

5.1 Introdução 135

5.2 Componente de Padrões de Processo 142

5.3 Transformações no Nível Comportamental.....144

5.4 Transformação no Nível de Atividade (TDA).....156

5.5 Parametrizações das Transformações 160

5.6 Conclusão.....165

CAPÍTULO 6 PROVA DE CONCEITOS..... 168

6.1 Introdução 168

6.2 Implementação da Abordagem Proposta de Desenvolvimento Orientado A Modelos 171

6.3 Implementação das Transformações.....184

6.4 Aplicação das Transformações No Nível Independente de Plataforma.....186

6.5 Aplicação das Transformações no Nível de Dependente De Plataforma.....204

6.6 Etapas Manuais para o Desenvolvimento do Sistema de Workflow 217

6.7 Workflow do Processo "Triar Pacientes para Admissão no Setor".....218

6.8 Conclusão.....222

CAPÍTULO 7 CONCLUSÕES E TRABALHOS FUTUROS..... 223

7.1 Conclusões 223

7.2 Trabalhos Futuros 226

REFERÊNCIAS 228

CAPÍTULO 1 INTRODUÇÃO

1.1 INTRODUÇÃO

O gerenciamento das organizações é uma tarefa que envolve um nível de complexidade significante, uma vez que agrega diversos domínios de conhecimento[1] (incluindo processo de negócios, tecnologias da informação e infra-estrutura) (DODAF, 2007). Tais domínios, por sua vez, fornecem critérios de qualidade potencialmente conflitantes, e envolvem fatores diversos que influenciam o desempenho de uma organização. Segundo Jonkers *et al.* (2004), isso é um problema importante, porque uma mudança na estratégia e nos objetivos da organização traz grandes modificações em todas as áreas de domínio de conhecimento da organização.

Para que seja possível analisar como esses fatores estão interconectados entre si e como a priorização de um deles pode ocasionar a postergação de outro, a utilização de uma Arquitetura Organizacional de Tecnologia da Informação (TI) torna-se necessária. Arquitetura Organizacional de TI (*Enterprise Architecture*) é uma arquitetura que captura a estrutura de uma organização, suas características e funcionalidades, bem como a interação dinâmica entre os seus diversos componentes.

Uma completa e consistente Arquitetura Organizacional de TI é a chave para a diminuição dos riscos e para o gerenciamento dos diversos domínios do conhecimento presentes na organização. Isso se justifica pelo fato de que a Arquitetura Organizacional de TI possibilita a avaliação de novos investimentos (novas tecnologias de informação, mudanças na infra-estrutura, novos processos negócios, entre outros) e a avaliação dos impactos dos novos investimentos no(s) objetivo(s) da organização e nos componentes organizacionais (DODAF, 2007).

Através de uma Arquitetura Organizacional de TI é possível capturar a essência e as evoluções do negócio e dos sistemas de informação presentes na organização, tornando

[1] Os domínios de conhecimento também podem ser chamados de componentes organizacionais (DODAF, 2007).

assim a organização mais flexível, adaptativa às evoluções, e conseqüentemente, tendo melhores chances para alcançar o sucesso nos negócios (LANKHORST, 2005).

Através da utilização das Arquiteturas Organizacionais de TI é possui realizar de maneira mais eficaz e menos onerosa o alinhamento entre tecnologia da informação e os processos de negócios pertencentes a uma ou mais organizações. Este alinhamento é reconhecido como uma boa prática e de grande eficiência (LANKHORST e VAN DRUNEN, 2007)(JONKERS *et al.*, 2004). A eficiência organizacional é obtida pela boa orquestração das relações entre os componentes organizacionais e não pela especificação detalhada de cada componente individualmente.

Por fim, o conceito de Arquitetura Organizacional de TI é de tal importância acadêmica e governamental que diversos trabalhos têm abordado esse conceito na literatura. Os principais trabalhos concentram-se no desenvolvimento de *frameworks* que ajudam na construção e na modelagem da Arquitetura Organizacional de TI de uma organização(LANKHORST, 2005). A seção 2.2 apresenta maiores detalhes sobre os *frameworks* de Arquiteturas Organizacionais TI.

1.2 MOTIVAÇÃO

As Arquiteturas Organizacionais de TI são uma fonte rica de informação para o desenvolvimento de sistemas computacionais que tem por objetivo interagir com um ou mais componentes organizacionais. Afinal, através dos modelos de Arquiteturas Organizacionais de TI é possível conhecer, dentre outros:

i. a estrutura da organização (obtendo informações que descrevem como as entidades organizacionais interagem para o funcionamento e comportamento da organização) (SCHEER, 1999);

ii. as atividades necessárias em um processo de negócio; e

iii. como os membros da organização e os recursos (software e equipamentos) interagem com o processo de negócio.

Dentre as diversas abordagens de desenvolvimento de sistemas computacionais, a que mais se destaca por proporcionar um meio de utilizar modelos no curso do entendimento, projeto, construção, implantação, operação, manutenção e modificação de sistemas computacionais é o Desenvolvimento Orientado a Modelos (do inglês, *Model-Driven Development*)(OMG, 2003). Através do Desenvolvimento Orientado a Modelos é possível construir transformações que permitam agilizar o desenvolvimento de um sistema computacional. A seção 2.4 apresenta detalhes sobre o Desenvolvimento Orientado a Modelos.

Existem diversas abordagens de Desenvolvimento Orientado a Modelos que utilizam os modelos das Arquiteturas Organizacionais de TI para o desenvolvimento de sistemas computacionais, como, (MENDLINGY e NÜTTGEN, 2004), (OUYANG *et al.*, 2008), (ZIEMANN e MENDLING, 2005), (MENDLING e ZIEMANN, 2005) dentre outras. Assim como estas abordagens, este trabalho concentra-se em Sistemas de Informação Orientados a Processo. A seção 2.5 caracteriza esse tipo de sistema.

A utilização dos modelos das Arquiteturas Organizacionais de TI é motivada pelo fato de que tais modelos contêm informações úteis para o desenvolvimento de sistemas computacionais. Por exemplo, os modelos de processos de negócio representam o fluxo de controle da execução de um conjunto de atividades organizacionais que visam alcançar um objetivo da organização (OUYANG *et al.*, 2008). O detalhamento de uma atividade pode incluir a descrição das informações necessárias e produzidas pela atividade, os responsáveis pela execução da atividade dentre outras informações que tornam os modelos de processos de negócio de grande importância para as abordagens de Desenvolvimento Orientado a Modelos.

Infelizmente, a maioria das abordagens atuais de Desenvolvimento Orientado a Modelos limita-se ao uso dos modelos de processos de negócios, e dessa forma, não tira proveito das informações contidas nos outros modelos usados para descrever Arquiteturas Organizacionais de TI. Por exemplo, a maioria das abordagens não utiliza os modelos de estrutura organizacional e os modelos de detalhamento de atividades como fonte de informação para o desenvolvimento de sistemas computacionais.

Os modelos de estrutura organizacional (organogramas, por exemplo) muitas vezes possuem informações que são de grande valor para o sistema computacional que será desenvolvido, como os papéis ou posto de trabalhos da organização que desempenham alguma atividade no processo de negócio. Esses papéis e os postos de trabalhos no nível do negócio podem ser usados como base para a definição de papéis e atores no nível de sistema.

Os modelos de detalhamento de cada atividade, por sua vez, tipicamente descrevem as informações que são consumidas e produzidas em cada atividade de um processo, assim como regras de negócio que se aplicam em cada atividade e os sistemas que dão suporte à execução da atividade. Em uma abordagem orientada a modelos, as informações contidas nos modelos detalhados podem ser utilizadas para descrever os dados necessários e produzidos para uma determinada funcionalidade do sistema em realização e, até mesmo, regras de execução que influenciem diretamente em diversas funcionalidades do sistema computacional em realização.

Apesar do potencial de uso dos modelos de detalhamento de atividade, grande parte das abordagens atuais de Desenvolvimento Orientado a Modelos consideram a atividade de um processo de negócio como uma "*caixa preta*" (menor unidade de decomposição). Logo, é necessário que o responsável pelo desenvolvimento do sistema computacional acrescente as informações contidas nos modelos de detalhamento de atividade manualmente, tornando, assim, o desenvolvimento mais demorado e mais suscetível a erros.

Com a utilização de diversos modelos da Arquitetura Organizacional de TI no desenvolvimento de sistemas computacionais torna-se possível que um projetista tenha uma visão abrangente dos aspectos organizacionais do ambiente no qual o sistema computacional está inserido. Assim, é possível também considerar os impactos do sistema computacional na organização e vice-versa.

Segundo CARDOSO et al. (2008), a partir de um processo de negócio é possível definir diferentes sistemas computacionais levando em consideração as diferentes decisões de projeto e os requisitos para os mesmos. No entanto, grande parte das abordagens de Desenvolvimento Orientado a Modelos que utiliza os modelos das Arquiteturas

Organizacionais de TI para o desenvolvimento de sistemas computacionais possui transformações pré-estabelecidas e não parametrizáveis, desta forma, impossibilitando que o projetista aplique decisões nas transformações. Por exemplo, as atuais abordagens Desenvolvimento Orientado a Modelos são apenas traduções entre espaços tecnológicos (no inglês, *technological spaces*)(KURTEV *et al.*, 2002), como em (MENDLING e ZIEMANN, 2005) que propõe a transformação direta de BPEL (OASIS, 2006) em EPC (SCHEER, 1999). Com este tipo de abordagem de transformação direta não há parametrização para permitir ao projetista indicar decisões de projeto relevantes para o sistema computacional. Um exemplo deste tipo de decisão é a definição de como uma determinada atividade do modelo BPEL deve ser transformada para o modelo EPC.

Por fim, a maioria das abordagens atuais de Desenvolvimento Orientado a Modelos não explora a riqueza da semântica das linguagens de modelagem das Arquiteturas Organizacionais de TI. Desta forma, as transformações propostas por essas abordagens, não realizam um "mapeamento" adequado entre os elementos do domínio organizacional e os elementos do domínio da solução computacional. Por exemplo, a transformação proposta por (MENDLINGY e NÜTTGEN, 2004) não diferencia atividades que são executas por seres humanos e por sistemas do modelo EPC (SCHEER, 1999) para o desenvolvimento do modelo em EPML (MENDLING, 2009).

1.3 OBJETIVO

O objetivo deste trabalho é apresentar uma nova abordagem de Desenvolvimento Orientado a Modelos que utiliza os modelos de Arquiteturas Organizacionais de TI para o desenvolvimento de sistemas computacionais. A abordagem de Desenvolvimento Orientado a Modelos proposta possibilita a inclusão de diversos modelos de domínio das Arquiteturas Organizacionais de TI no desenvolvimento de sistemas computacionais, não se limitando aos modelos de processos de negócio. Desta forma, objetiva-se auxiliar o projetista no entendimento do contexto no qual o sistema computacional está incluído e, também possibilitar a utilização das informações contidas nesses modelos em transformações semi-automáticas para o desenvolvimento do sistema computacional.

As abordagens atuais de Desenvolvimento Orientado a Modelos assumem que os modelos das Arquiteturas Organizacionais de TI podem ser automatizados sem modificações. Esta premissa leva à definição de modelos de processos de negócio que misturam interesses de negócio e interesses de sistema. Diferentemente dessas abordagens, a abordagem proposta neste trabalho assume que os modelos das Arquiteturas Organizacionais de TI foram construídos para representar uma realidade organizacional e não para automatização. Desta forma, são definidos mecanismos para que o projetista possa adequar os modelos das Arquiteturas Organizacionais de TI para atender às necessidades do sistema computacional em desenvolvimento, mantendo ainda o alinhamento com a realidade organizacional capturada nos modelos originais.

Por fim, a abordagem de Desenvolvimento Orientada a Modelos proposta permite que o projetista explore a riqueza da semântica das linguagens de modelagem da Arquitetura Organizacional de TI em todas as etapas do desenvolvimento do sistema computacional.

1.4 METODOLOGIA E HISTÓRICO DO DESENVOLVIMENTO DO TRABALHO

Inicialmente foi realizado um estudo das principais abordagens de Desenvolvimento Orientado a Modelos que utilizam modelos de Arquiteturas Organizacionais de TI para o desenvolvimento de sistemas computacionais. Neste estudo buscou-se entender como tais abordagens utilizam as informações contidas nos modelos e como as decisões de projeto são incorporadas no desenvolvimento de sistemas computacionais. Constatou-se que a maioria das abordagens de Desenvolvimento Orientado a Modelos baseadas em modelos de Arquiteturas Organizacionais de TI não explora de forma abrangente as informações contidas nos modelos e não permite que o projetista aplique as decisões de projeto nas transformações.

Adicionalmente, foi realizado um levantamento bibliográfico sobre os principais frameworks de modelagem de Arquiteturas Organizacionais de TI que podem ser utilizados para o desenvolvimento de sistemas computacionais. Dentre os diversos *frameworks* (como DODAF (DODAF, 2007), TOGAF (TOGAF, 2007), ArchiMate (ARCHIMATE CONSORTIUM, 2009)) o ARIS (*Architecture of Integrated Information Systems*) (SCHEER, 1999) foi adotado neste trabalho. A escolha pelo

ARIS foi motivada por este fornecer um conjunto de linguagens de modelagem bem difundido, que permite modelar os diversos aspectos da organização como, por exemplo, aspectos organizacionais e de processo de negócio. Outro ponto importante na escolha do ARIS é a existência de diversas ferramentas que utilizam os modelos do ARIS como, por exemplo, Microsoft's Visio, BOC's ADONIS, SAP R/3, ARIS Toolset e outras da IDS Scheer (KERN e KÜHNE, 2007). Devido ao valor econômico do ARIS, diversos trabalhos acadêmicos têm sido realizados utilizando seus modelos como, por exemplo, trabalhos na área de análise ontológica (GREEN *et al.*, 2004), na área de transformações entre linguagens utilizando técnicas de desenvolvimento orientado a modelos (MENDLINGY e NÜTTGEN, 2004) (KERN e KÜHNE, 2007) dentre outros.

Com a escolha do *framework* de modelagem, foi necessário considerar os aspectos sintáticos e semânticos das linguagens de modelagem para incorporação dos modelos produzidos por essas linguagens na abordagem de Desenvolvimento Orientado a Modelos proposta. Logo, foi realizada uma busca bibliográfica sobre os metamodelos das linguagens de domínio do ARIS que são implementadas nas ferramentas de modelagem. O objetivo dessa busca foi entender de forma profunda os conceitos dos domínios representados por tais linguagens.

Infelizmente, os metamodelos das linguagens de domínio disponíveis na literatura e apresentados em (SCHEER, 1999) e (DAVIS, 2001) não descrevem as atuais implementações nas ferramentas de modelagem. Desta forma, foi necessário desenvolver no escopo deste trabalho uma abordagem de escavação para definir os metamodelos das linguagens de modelagem de domínio. Este metamodelo é também a base para a definição da semântica de cada elemento presente nos metamodelos das linguagens de modelagem de domínio.

A fim de uma melhor definição e compreensão dos conceitos de cada elemento presente nos metamodelos de domínio escavados em termos de entidades do mundo real foi necessário realizar uma análise ontológica com ontologias de fundamentação. A ontologia de fundamentação escolhida neste trabalho foi a UFO (GUIZZARDI, 2005a)(GUIZZARDI *et al.*, 2008). A UFO tem sido aplicada com sucesso na avaliação, (re) projeto e integração de modelos de linguagens de modelagem conceitual, assim

como, na definição semântica de entidades do mundo real para os elementos de tais modelos (GUIZZARDI *et al.*, 2008) .

A análise ontológica possibilitou identificar e eliminar problemas nos metamodelos das linguagens de domínio do ARIS, além de realizar distinções conceituais relevantes entre os elementos do metamodelo. Através da análise ontológica foi possível definir um conjunto mínimo de elementos e relacionamentos que permite expressar de forma exata e sem ambigüidade as conceituações presentes em cada domínio organizacional que as linguagens do ARIS visam representar.

Com uma base sintática e semântica para representação e interpretação dos modelos de Arquitetura Organizacional de TI e com um estudo aprofundado das abordagens atuais de Desenvolvimento Orientado a Modelos foi proposta uma abordagem de Desenvolvimento Orientado a Modelos que visa: **(i)** aproveitar corretamente a semântica dos elementos de modelagem contidos nos modelos das Arquiteturas Organizacionais de TI e **(ii)** permitir ao projetista aplicar as decisões de projeto nas transformações pré-definidas; e **(iii)** dividir o processo de desenvolvimento do sistema computacional em etapas independente e dependente de plataforma.

Por fim, foi realizada uma prova de conceitos da abordagem de desenvolvimento orientado a modelos proposta neste trabalho com a implementação dos metamodelos e transformações. Estes metamodelos e transformações foram aplicados no desenvolvimento de um sistema de *workflow*.

1.5 ORGANIZAÇÃO

Este trabalho é estruturado em sete capítulos. O conteúdo de cada um desses é descrito resumidamente abaixo:

- *Capítulo 2 – Referencial Teórico*: apresenta o referencial teórico a respeito de Arquiteturas Organizacionais de TI, Desenvolvimento Orientado a Modelos, Ontologias de fundamentação e Sistemas orientados a processos de negócio.

- *Capítulo 3 – Metamodelos das Linguagens do ARIS Method*: apresenta os metamodelos das linguagens de modelagem organizacional, processo de negócio e detalhamento de atividades presente no ARIS Method.

- *Capitulo 4 – Análise Ontológica dos Metamodelos dos Domínios de Modelagem Organizacional e de Modelagem de Processo de Negócio*: apresenta a análise ontológica dos elementos e relacionamentos presentes nos metamodelos dos domínios de modelagem organizacional e de modelagem de processo de negócio. Através da análise ontológica é possível interpretar cada elemento presente nos metamodelos em entidades do "mundo real" e, também, identificar problemas estruturais nas linguagens de modelagem.

- *Capítulo 5 – Uma Abordagem de Desenvolvimento Orientado a Modelos Utilizando Artefatos de Arquitetura Organizacional de TI*: descreve em detalhes os conceitos e as transformações da abordagem de desenvolvimento orientado a modelos proposta neste trabalho. O capítulo discute também as diferenças entre a abordagem proposta neste trabalho e a maioria das abordagens atuais de Desenvolvimento Orientado a Modelos com base em modelos de Arquiteturas Organizacionais de TI.

- *Capítulo 6 – Prova de conceitos*: apresenta a viabilidade da abordagem de Desenvolvimento Orientado a Modelos proposta nesse trabalho, através da construção de um sistema orientado a processos de negócio.

- *Capítulo 7 - Considerações Finais e Trabalhos Futuros*: apresenta as conclusões do trabalho, as contribuições, dificuldades e propostas de trabalhos futuros que podem ser desenvolvidos com base neste trabalho.

CAPÍTULO 2 REFERENCIAL TEÓRICO

2.1 INTRODUÇÃO

Este capítulo apresenta o referencial teórico necessário para o entendimento deste trabalho como um todo. O capítulo é organizado da seguinte forma: a seção 2.2 apresenta uma breve discussão sobre as Arquiteturas Organizacionais de TI e depois apresenta as características do framework ARIS, a seção 2.3 apresenta conceitos sobre ontologias, mais especificamente sobre a Ontologia de Fundamentação UFO, a seção 2.4 descreve os conceitos e práticas do Desenvolvimento Orientado a Modelos e, finalmente, a seção 2.5 descreve os conceitos dos sistemas computacionais que utilizam os processos de negócio como base para o seu funcionamento e desenvolvimento.

2.2 ARQUITETURAS ORGANIZACIONAIS DE TI

Uma definição ampla de Arquitetura Organizacional de TI é apresentada por LANKHORST (2005):

> *Arquitetura Organizacional de TI consiste em um conjunto completo e coerente de princípios, métodos e modelos que são utilizados no projeto e implementação de uma estrutura organizacional, processos de negócios, sistemas de informação e infra-estrutura*.

O conceito de Arquitetura Organizacional de TI é de tal importância acadêmica e governamental que diversos trabalhos acadêmicos e governamentais estão presentes na literatura. Grande parte destes trabalhos se concentra no desenvolvimento de frameworks para modelagem e estruturação de Arquitetura Organizacional de TI. Os frameworks que se destacam são:

i. **ArchiMate** (ARCHIMATE CONSORTIUM, 2009)**,** desenvolvido por um consórcio composto dos seguintes membros: Telematica Instituut, Ordina,

Radboud University Nijmegen, o Leiden Institute for Advanced Computer Science (LIACS) e o Centre for Mathematics and Computer Science (CWI).

ii. **DoDAF** (*Department of Defense Architecture Framework*), desenvolvido pelo Departamento de Defesa Norte-Americano (DODAF, 2007);

iii. **RM-ODP**, padronizado pela ISO (RM-ODP-ISO-ISO/ITU-T, 1995) e especializado na arquitetura de sistemas distribuídos;

iv. **TOGAF** (*The Open Group Architecture Framework*), (TOGAF, 2007) desenvolvido pelo The Open Group (THEOPENGROUP, 2009);

v. O framework de **Zachman** (ZACHMAN, 1987) desenvolvido por John Zachman em 1987; e

vi. **ARIS** (*Architecture of Integrated Information Systems*) (SCHEER, 1999) desenvolvido pelas pesquisas acadêmicas do professor A.W.Scheer;

Através da utilização destes frameworks, é possível documentar os conceitos do negócio, das aplicações, de infra-estrutura e outros conceitos relacionados aos mesmos que sejam relevantes, além de perspectivas e aspectos desejáveis para atender o(s) stakeholder(s) no desenvolvimento de um projeto de software(LANKHORST, 2005).

A utilização de um framework de Arquitetura Organizacional de TI possibilita não somente a estruturação do conhecimento do domínio organizacional de forma sistemática, mas também provê ferramentas para análise de desempenho organizacional, análise de impactos e também provê um meio de realizar o alinhamento entre estrutura organizacional e arquitetura de TI.

Os frameworks de Arquiteturas Organizacionais de TI são particularmente úteis no desenvolvimento de sistemas computacionais em dois aspectos: **(i)** favorecem a rastreabilidade de informações entre a arquitetura organizacional (capturada no formato de um artefato denominado modelo) e os sistemas/infra-estrutura que suportam esta arquitetura (concebidos a partir dos modelos gerados previamente) (LANKHORST, 2005) e **(ii)** permitem a geração automática de código uma vez que os modelos são

artefatos passíveis de sofrer transformações, para derivarem código para sistemas computacionais (LANKHORST, 2005).

Este trabalho está interessado nos frameworks de Arquitetura Organizacional de TI que fornecem um conjunto de linguagens de modelagem, que permita modelar os diversos aspectos da organização como, por exemplo, aspectos organizacionais e de processo de negócio. Dentre os frameworks supracitados, os que possuem tal característica são o ARIS e o ArchiMate. Devido ao foco deste trabalho, será explicado apenas o framework ARIS na seção 2.2.1. Este framework foi escolhido pela sua ampla adoção no setor produtivo e devido ao suporte à definição detalhada de processos de negócio.

2.2.1 ARIS

O ARIS (***AR****chitecture for integrated* ***I****nformation* ***S****ystems*) foi desenvolvido na universidade de Saarbrucken, Alemanha, em 1992 com o objetivo principal de permitir a descrição e desenvolvimento de sistemas de informação que estivessem integrados à estrutura da organização através de seus processos de negócio(SCHEER, 1999) .

Através do ARIS é possível: **(i)** documentar os processos de negócios existentes na organização; **(ii)** obter uma visão geral da organização para análise e projeto dos processos de negócios; e **(iii)** apoiar o projeto de sistemas de informação (LANKHORST, 2005).

O ARIS propõe uma arquitetura de modelagem de organizações chamada de ARIS Method, estruturada em cinco diferentes visões (organizacional, dados, controle, função e saída) e três diferentes camadas de abstração (requisitos, projeto e implementação)(SCHEER, 1999). Dado o escopo dos modelos suportados pelo ARIS Method, esta abordagem pode ser considerada uma técnica abrangente de modelagem de organizações e de arquiteturas de TI(LANKHORST, 2005). Cada uma das visões do ARIS Method possui uma linguagem própria.

A *visão organizacional* descreve a hierarquia organizacional, isto é, descreve a comunicação e o relacionamento entre as unidades organizacionais de uma organização, revelando também os papéis dos indivíduos na organização(SCHEER, 1999).

A *visão de dados* descreve as informações que são manipuladas na organização. Nesta visão é possível expressar como o conhecimento organizacional está organizado e armazenado (DAVIS, 2001).

A *visão de função* é utilizada para descrever as tarefas realizadas pela organização. Através dessa visão é possível expressar os objetivos organizacionais de cada tarefa realizada na organização e quais sistemas computacionais auxiliam em cada tarefa organizacional. Também é possível descrever a hierarquia das tarefas organizacionais.

A *visão de controle* descreve os processos de transformação da informação por meio de uma função ou um conjunto de funções. Como as funções representam atividades organizacionais potencialmente complexas, esta visão é usada para modelar processos de negócio na abordagem ARIS(SCHEER, 1999). Através dessa visão é possível demonstrar o relacionamento entre os processos de negócio da organização e as demais entidades da organização (entidades organizacionais, recursos, informações e entre outros aspectos).

Em relação aos níveis de abstração propostos pelo ARIS, o Nível de Requisitos é destinado a atividades de modelagem do domínio do problema, ou seja, na elicitação de conceitos e detalhes referentes ao negócio e à organização, assim como o fechamento do escopo da modelagem abarcada pelos modelos e os objetivos pelos quais se justifica a modelagem. Assim, nesse momento, a perspectiva analisada deve ser inteiramente orientada à percepção que o usuário possui do negócio, bem como sua documentação e padrões(CARDOSO, 2007). Estão presentes neste primeiro nível os modelos de cadeia de eventos orientados a processos (EPCs), diagrama de objetivos, dentre outros modelos que ajudam na modelagem do domínio do problema (CARDOSO, 2007).

O segundo nível de abstração, o nível de Especificação de Projeto, tem por objetivo construir modelos que retratem os conceitos de TI, ou seja, construir modelos que descrevam o projeto conceitual de arquitetura, requisitos funcionais de uma aplicação e infra-estrutura de um sistema, sem se ater, no entanto, a detalhes específicos a uma tecnologia (CARDOSO, 2007). Portanto, como esse nível implementa certas características do nível de Nível de Requisitos, é possível alterar-se alguns detalhes de especificação de projeto que implementem os mesmos conceitos do negócio, sem que,

no entanto, seja necessário retornar ao Nível de Requisitos para atualizá-lo, o que garante flexibilidade no processo (CARDOSO, 2007).

O último nível de abstração, o Nível de Descrição de Implementação, tem por objetivo traduzir a arquitetura descrita no Nível de Especificação de Projeto para modelos que descrevam detalhes tecnológicos dos componentes que irão implementá-la(CARDOSO, 2007). O Nível de Descrição de Implementação está intimamente ligado à tecnologia da informação e sua especificação pode ser apoiada pelos detalhes levantados no Nível de Especificação de Projeto(CARDOSO, 2007).

Tendo como base o ARIS Method, diversas ferramentas foram desenvolvidas, como por exemplo, ARIS SOA Architect, ARIS Business Architect e ARIS Toolset. Estas ferramentas provêem um ambiente para modelagem, gerenciamento dos modelos produzidos e outros serviços de gerenciamento de processos(KERN e KÜHNE, 2007).

2.3 ONTOLOGIAS

O Dicionário Webster (2004 apud GUIZZARDI 2005, GUIZZARDI 2007) define ontologia como: **(i)** um ramo da metafísica preocupado com a natureza e os relacionamentos dos seres; **(ii)** uma teoria particular sobre a natureza do ser; **(iii)** uma teoria preocupada com os tipos de entidades e, especialmente, com os tipos abstratos de entidades que são admitidos em um sistema de linguagem.

Segundo GRUBER (1993) "*uma ontologia é uma especificação formal e explícita de uma conceituação compartilhada*". Uma conceituação é uma abstração, uma visão simplificada de uma parte do mundo que se deseja representar para algum propósito. Outra definição bastante utilizada sobre ontologias é apresentada em (GUARINO e GIARETTA, 1995): "Uma ontologia é uma especificação explícita e parcial de uma conceitualização"

Na Filosofia, a palavra ontologia[2] é utilizada para representa um sistema particular de categorias que descreve uma visão do mundo (GUARINO, 1998)(GUIZZARDI, 2005a).

[2] A palavra ontologia, em grego, significa "*o estudo do ser e da existência*" (GUIZZARDI, 2005a) (GUIZZARDI, 2007).

Aristóteles foi primeiro filósofo ocidental que utilizou a ontologia para o estudo das características da realidade e dos objetos reais.

De acordo com Guizzardi (2007), na Ciência da Computação, o primeiro relato da utilização de ontologias ocorreu em um trabalho sobre fundamentos de modelagem de dados em 1967, no qual foi utilizado o conceito filosófico de ontologias. De uma maneira independente, pesquisadores na área de Inteligência Artificial começaram a fazer uso daquilo que veio a ser conhecido como Ontologias de Domínio (GUIZZARDI, 2007) e, desde então, diversas ontologias de domínio tem sido criadas em diversas áreas da Ciência da Computação. As Ontologias de Domínio descrevem um vocabulário relativo a um domínio genérico (por exemplo, de Engenharia de Software e Processos de Negócio) (GUARINO, 1998).

Um ponto importante que deve ser enfatizado é que o termo ontologia é utilizado com diferentes sentidos nas diversas áreas da Ciência da Computação (GUIZZARDI, 2007)(GUIZZARDI *et al.*, 2008). Na área de Sistema de Informação, o termo ontologia tem sido utilizado de acordo com as definições da Filosofia, isto é, como um sistema de categorias formais independente de domínio e filosoficamente bem fundamentado que pode ser usado para enunciar modelos da realidade específica de domínio (GUIZZARDI *et al.*, 2008). Em outras áreas da Ciência da Computação o termo ontologia, é em geral, utilizado: **(i)** como um artefato concreto de engenharia projetado para um propósito específico sem dar muita atenção para questões de fundamentação (GUIZZARDI, 2007)(GUIZZARDI *et al.*, 2008); e **(ii)** como uma representação de um domínio particular (por exemplo, Engenharia de Software, Processo de negócio, Medicina e entre outros domínios do conhecimento) (GUIZZARDI *et al.*, 2008).

Guarino (1998) define quatro tipos de ontologias de acordo com o seu nível de generalidade: **(i)** Ontologias de Fundamentação, **(ii)** Ontologias de Domínio, **(iii)** Ontologias de Tarefa e **(iv)** Ontologias de Aplicação. A Figura 2.1 apresenta a relação entre os tipos de ontologias. As setas na Figura 2.1 representam uma relação de "dependência" conceitual entre os tipos de ontologias, ou seja, uma ontologia de fundamentação fornece os conceitos necessários para a construção das ontologias de domínio e de tarefa, enquanto as ontologias de domínio e tarefa fornecem os conceitos necessários para a construção de ontologias de aplicação.

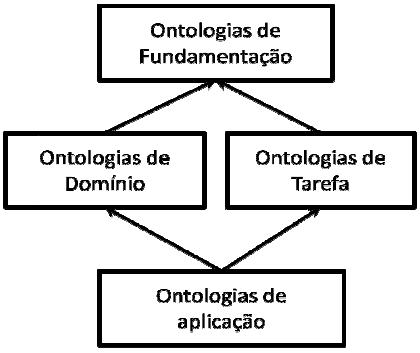


Figura 2.1 – Tipos de Ontologias de acordo com o nível de generalização (GUARINO, 1998)

As Ontologias de Fundamentação descrevem conceitos como espaço, tempo, objeto, eventos, ações e outros conceitos que são independentes de um problema ou domínio em particular (GUARINO, 1998).

As Ontologias de Domínio e Ontologias de Tarefa descrevem, respectivamente, o vocabulário relativo a um domínio genérico (por exemplo, de Engenharia de Software, processos de negócio) e o vocabulário relativo a uma tarefa ou atividade genérica (por exemplo, venda ou compra de carros) através de especializações de termos apresentados nas Ontologias de Fundamentação (GUARINO, 1998).

As Ontologias de Aplicação descrevem sobre conceitos que dependem de um domínio e uma tarefa em particular, que freqüentemente são especializações das Ontologias de Domínio e de Tarefa. Estes conceitos freqüentemente correspondem a papéis que são desempenhados pelas entidades do domínio na execução de certa tarefa ou atividade (GUARINO, 1998).

Em particular, neste trabalho estamos interessados nas Ontologias de Fundamentação e no uso destas para a melhoria da qualidade das linguagens de modelagem, conforme apresentado em (GUIZZARDI e WAGNER, 2005b). Dentre as diversas Ontologias de Fundamentação presentes na literatura, como DOLCE (BOTTAZI e FERRARIO, 2006), GFO (HERRE *et al.*, 2006), SUMO (NILES e PEASE, 2001) e Cyc(LENAT e GUHA, 1990), adota-se neste trabalho a Ontologia de Fundamentação UFO (***U**nified **F**oundational **O**ntology*), por ser uma ontologia que visa unificar as ontologias de fundamentação supracitadas, capturando as suas características positivas e sanando suas limitações(GUIZZARDI e WAGNER, 2005b).

A UFO será utilizada como base semântica para a análise ontológica dos metamodelos da linguagem Organizacional, de Processo de Negócio e de Detalhamento de Atividades do ARIS Method a serem apresentados no Capítulo 3. Através da análise ontológica será possível: **(i)** definir formalmente e rigorosamente os elementos dessas linguagens em termos de entidades do mundo real (utilizando os conceitos da UFO); **(ii)** identificar elementos inadequados das linguagens de modelagem utilizando o framework de avaliação de linguagens apresentado em(GUIZZARDI *et al.*, 2005a); e **(iii)** indicar recomendações de melhoria nas linguagens caso sejam identificados problemas na interpretação ontológica dos elementos das linguagens.

O resto desta sub-seção é dedicado a uma descrição da UFO, uma vez que seus conceitos serão relevantes para a análise ontológica conduzida no Capítulo 4.

2.3.1 UFO

A UFO (***U**nified **F**oundational **O**ntology*) tem sido desenvolvida com base em um número de teorias das áreas de Ontologias Formais, Lógica Filosófica, Filosofia da Linguagem, Lingüística e Psicologia Cognitiva (GUIZZARDI *et al.*, 2008). Esta ontologia de fundamentação tem sido aplicada com sucesso na avaliação, (re) projeto e na integração de modelos das linguagens de modelagem, assim como, na definição semântica de entidades do mundo real para os elementos de tais modelos (GUIZZARDI *et al.*, 2008)(ALMEIDA *et al.*, 2009)(SANTOS JR. *et al.*, 2010).

A UFO é dividida em três camadas que se complementam: **(i)** UFO-A é o núcleo da UFO, excluindo conceitos relacionados com eventos; **(ii)** UFO-B concentra-se nos conceitos ligados a eventos; e **(iii)** UFO-C fundamenta-se em UFO-A e UFO-B para sistematizar conceitos sociais, tais como planos, agentes, objetivos, ações, intencionalidade, compromisso e entre outros conceitos sociais (GUIZZARDI *et al.*, 2008).

As seções seguintes apresentam, respectivamente, a UFO-A, UFO-B e UFO-C. O conteúdo apresentado nessas seções foi elaborado com base nos seguintes trabalhos: (GUIZZARDI, 2005a), (GUIZZARDI *et al.*, 2005a), (GUIZZARDI e WAGNER, 2005b), (GUIZZARDI *et al.*, 2007a) e (GUIZZARDI *et al.*, 2008).

2.3.1.1 UFO-A

Uma distinção fundamental nessa ontologia ocorre entre a categoria Particular e a categoria Universal. Particulars são entidades que existem na realidade, possuindo uma identidade única. Universal, de modo contrário, são modelos independentes de espaço e tempo, que podem ser instanciados em diferentes Particulars, ou até mesmo em outros Universals. Em outras palavras, um Particular é uma instância de um Universal que define, em linguagem informal, "tipos de entidades". A Figura 2.2 apresenta um fragmento da UFO-A com a distinção entre Particular e Universal.

A entidade 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 individual, uma bola, um cão, oceano atlântico e o presidente da república.

A entidade Moment 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 particular *x* é *dependente existencialmente* (*ed*) de um outro particular *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 (Propriedade Relacional) de Intrinsic Moments (Propriedade Intrínseca). Os Intrinsic Moments dependem de apenas um Substantial para existirem. São

exemplos de Intrinsic Moments a cor de um objeto, a dor de cabeça de uma pessoa e a temperatura de um ser. Diferente dos Intrinsic Moments, os Relational Moments, também chamado de Relators, dependem de um conjunto de Particulars para existirem como, por exemplo, o Relator *tratamento médico* depende para existir dos Particulars *Paciente* e *Médico*.

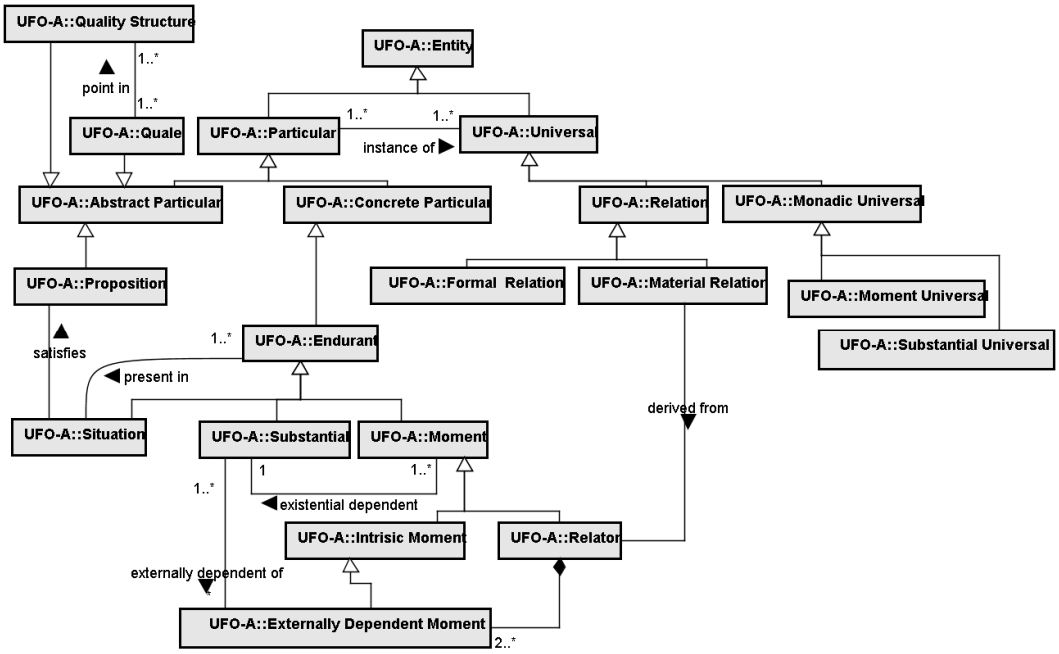


Figura 2.2 – Fragmento da UFO-A.

A relação de *dependência existencial* que existe entre um Moment *x* e um Particular *y* na qual *x* depende de *y* é a relação de Inherence (inerência- *i*). Assim, para que um Particular *x* seja um Moment de um outro Particular *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 Particulars que não são inerentes a outros Particulars são denominados de Substantials.

As categorias Substantial Universal e Moment Universal, ambas especializações de Monadic Univerval, são uma interpretação ontológica para as estruturas estáticas de um domínio. Exemplos de Substantial Universals são Maçã,

Planeta e Pessoa. Todos esses são tipos, e não Particulars – como uma pessoa específica. Da mesma maneira, exemplos de Moment Universal são a Cor, a Carga Elétrica, o Sabor e a Dor de Cabeça, que são intrínsecos a alguns Substantial Universals.

Uma tentativa de modelar a relação entre propriedades intrínsecas e suas representações na estrutura cognitiva humana, ou seja, na forma em que o ser humano vê o mundo, é representada na Teoria dos Espaços Conceituais, introduzida em(GÄRDENFORS, 2000). Essa teoria é baseada na noção de Quality Structures. A idéia é que, para diversos tipos de propriedades, existe uma Quality Structure relacionada na cognição humana. Por exemplo, a Altura e a Massa estão associadas com as estruturas unidimensionais isomórficas à semi-reta dos números positivos. Outros tipos de propriedades, como a Cor e o Sabor, são representadas por estruturas multi-dimensionais.

Em (GÄRDENFORS, 2000), a percepção, ou concepção, de uma propriedade intrínseca pode ser representada como um ponto em uma Quality Structure, e este ponto é chamado de Quale. Quality structures e Qualia – o plural de Quale – são, juntos com conjuntos, números e proposições lógicas (Propositions), exemplos de Abstract Particulars, como mostrado na Figura 2.2. Assim, resume-se que, se um Moment Universal é associado a uma Quality Structure e é inerente a um Substantial Universal, então uma instância (um Substantial) desse Substantial Universal deve possuir uma propriedade cuja concepção é um ponto na Quality Structure citada. Por exemplo, quando dizemos que "Antônio" possui *1,80* m de altura estamos nos referindo a uma instância do Substantial Universal Pessoa, sua altura (Quality) e um ponto (Quale) na Quality Structure de estruturas unidimensionais isomórficas à semi-reta dos números racionais.

A entidade Quality representa uma qualidade de um individual. O Quality está associado a uma estrutura de qualidade (Quality Structure). Por exemplo, a cor (Quality) da maçã está associada à estrutura de qualidade (Quality Structure) RGB.

Relations são entidades que conectam ("colam") outras entidades. Na literatura filosófica, duas amplas categorias de relações são tipicamente consideradas, a saber, as Material Relations (relações materiais) e Formal Relations (relações formais) (HELLER e HERRE, 2004a).

As Formal Relations relacionam duas ou mais entidades diretamente, sem nenhum indivíduo de intervenção adicional. A princípio, a categoria de Formal Relations inclui aquelas relações que dão forma a uma superestrutura matemática que garante formalmente a estrutura da ontologia UFO, incluindo as relações de dependência existencial (existential dependence - ed), inerência (inherence - *i*), parte-de (part-of - <), subconjunto-de (subset-of), instanciação (instantiation), caracterização (characterization), exemplificação (exemplification), dentre outras não discutidas aqui (ver (GUIZZARDI, 2005a)).

As Material Relations, ao contrário, têm uma estrutura material em si próprias e incluem exemplos tais como "*trabalhar em*", "*estar registrado em*", e "*ser conectado a*". Enquanto uma Formal Relation como aquela entre Paulo e seu conhecimento *x* da língua inglesa reside diretamente e sempre em todas as situações em que Paulo e *x* existirem, uma relação material de "sendo tratado por" entre José e uma unidade médica UM1 existe se uma outra entidade mediadora existir e que ela relacione José a UM1. Essas entidades são chamadas de Relators. Relators são particulars com a capacidade (poder) de conectar outras entidades. Por exemplo, um tratamento médico conecta um paciente com uma unidade médica, um registro conecta um estudante com uma instituição educacional e uma ligação covalente conecta dois átomos.

Um importante conceito para a caracterização de Relators é a noção de Foundation. Foundation pode ser vista como um tipo da dependência histórica, de maneira que, por exemplo, uma instância de "ser beijado" é fundamentada em um beijo individual, ou uma instância de "ser conectado" entre aos aeroportos é fundamentada em uma conexão de um vôo em particular. Suponha que Gian está casado com Renata. Neste caso, pode-se assumir que há um Particular Relator (Relational Moment) m_1 do tipo "casamento" que faz o intermédio entre Gian e Renata. A Foundation deste Relator pode ser, por exemplo, um evento de casamento ou de assinatura de um contrato social

entre as partes envolvidas, que dá veracidade à proposição "Gian é casado com Renata". A existência do relator depende historicamente deste evento.

Há muitos Moments que Gian adquire pela virtude de ser casado com Renata. Por exemplo, considere todas as responsabilidades legais que Gian tem no contexto desta relação. Estas propriedades recentemente adquiridas são Intrinsic Moments de Gian que, conseqüentemente, são inerentes e existencialmente dependentes dele. Entretanto, estes Moments dependem também da existência de Renata. Este tipo de Moment é denominado na ontologia UFO como Externally Dependent Moments, isto é, Intrinsic Moments que são inerentes a um indivíduo único, mas que são existencialmente dependentes de outros indivíduos: um Moment x é externamente dependente (Externally Dependent) se e somente se é existencialmente dependente de um Particular que seja independente (no sentido técnico discutido anteriormente) do portador do Moment. No exemplo de um Material Externally Dependent Moment x há sempre um Particular externo a seu portador (isto é, que não é uma de suas partes ou seus Intrinsic Moments), que é a Foundation de *x*. Mais uma vez, no exemplo dado, pode-se pensar em um determinado evento e_1 (evento do casamento ou assinatura do contrato social) em que Gian e Renata participam e em que fundamenta a existência destes Externally Dependent Moments inerentes a Gian.

Situations (situações) são tipos especiais de Endurants. São entidades complexas que são constituídas possivelmente por muitos Endurants (incluindo outras Situations). Situations são assumidas aqui como sinônimos do que é denominado na literatura de *state of affairs*, isto é, uma parcela da realidade que possa ser compreendida como um todo. Os exemplos de situações incluem "*José estar com gripe e febre*", "*José está no mesmo local que Gian enquanto Bernardo estiver no mesmo local que André*", "*José estar mais velho do que a idade mínima de aposentadoria enquanto Lula for presidente do Brasil*", "*Renata está casada com Gian que trabalha para a UFES*".

A UFO define uma relação de "estar presente em" (being present at) entre Endurants e as Situations que eles constituem. Por exemplo, pode-se assumir que José (Substancial) e os seus Intrinsic Moments m_1 (febre de José) e o m_2 (gripe de José) estão presentes na Situation s_1: "José estar com gripe e febre".

Espécie (Kind), Subespécie (Subkind), Coleção (Collection), Fase (Phase) e Papel (Role) são especializações de Sortal. Kind é um tipo que provê o Princípio de Identidade às suas instâncias, que são chamadas de complexos funcionais, ou seja, entidades cujas partes exercem diferentes papéis. Por exemplo, Pessoa é uma Espécie composta de várias partes, cada qual desempenha um papel distinto. Subkind é uma especialização de Espécie que herda desta entidade o Princípio de Identidade. Por exemplo, sendo Pessoa uma Espécie, pode-se definir Homem e Mulher como suas Subespécies. Um grupo de pessoas, porém, é uma instância do tipo Collection. Este tipo, assim como Espécie, provê o Princípio de Identidade a suas instâncias, que são, porém, conjuntos de objetos que exercem o mesmo papel. Pode-se dizer ainda que um indivíduo (ou coleção) que é instância de uma Espécie/Subespécie (ou Coleção), é necessariamente instância deste (em todos os mundos possíveis). Já o tipo Phase é um Sortal instanciado em determinado mundo ou período de tempo, mas não necessariamente em todos. Por exemplo, Criança, Adolescente e Adulto são fases da Espécie Pessoa. Também Role é um tipo de Sortal instanciado eventualmente, mais precisamente na Participação em um Evento ou numa determinada Relação. Por exemplo, Mãe é um Papel para a Subespécie Mulher mediante a existência da relação de maternidade com uma instância do Papel Filho da Espécie Pessoa(GUIZZARDI, 2005a).

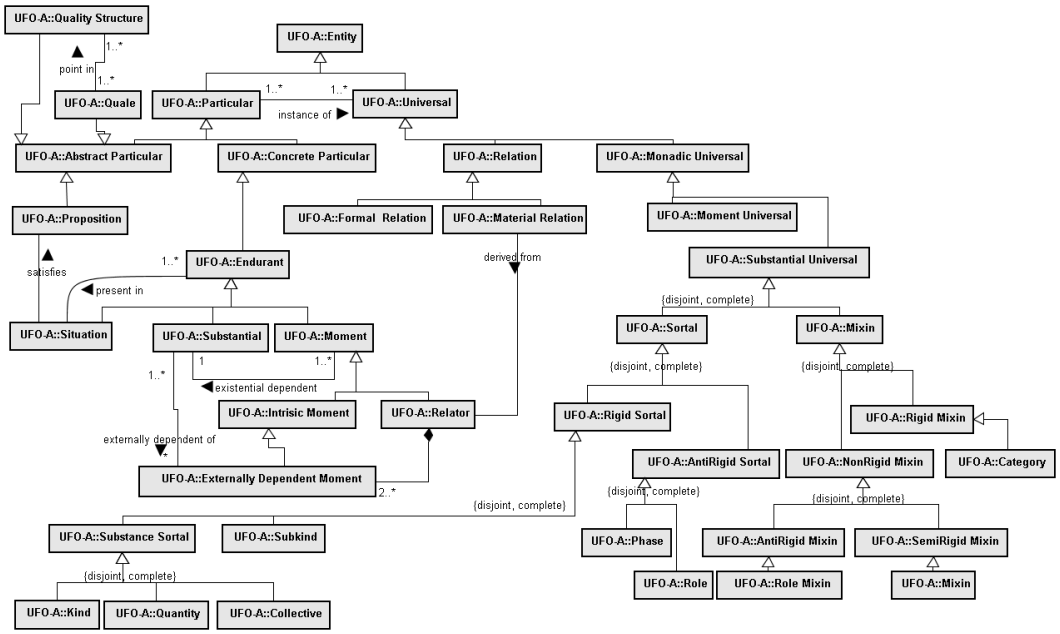


Figura 2.3 – Fragmento da UFO-A com foco nos Universais.

Por fim, Category (Categoria) é um tipo de Mixin, que classifica as entidades que pertencem a Kinds diferentes, mas que compartilham uma propriedade comum essencial (ou seja, uma propriedade que eles não podem perder). Por exemplo, uma Categoria Entidade Racional pode ser especializada pelas Espécies Pessoa e Agente Artificial (GUIZZARDI, 2005a). Mixin é o tipo de entidade que representa Dispersive Universal (Universais Dispersivos), ou seja, cobrem conceitos com diferentes princípios de Identidade (GUIZZARDI, 2005a).

2.3.1.2 UFO-B

A UFO-B faz a distinção entre indivíduos Endurants e Perdurants. Tal distinção pode ser captada a partir do entendimento de seus comportamentos em relação ao tempo. Endurants são ditos como sendo totalmente presentes em qualquer momento em que existam, ou seja, os Endurants existem no tempo, no sentido de que se é dito que, em uma circunstância c_1, um Endurant e1 tem uma propriedade P_1, e, em uma circunstância c_2, a propriedade P_2 (podendo ser incompatíveis entre si), é o mesmo Endurant e que está sendo referenciado em cada uma das situações. Por exemplo, pode-se dizer que o Particular Paulo pesa 80 Kg em c_1, mas 68 Kg em c_2. Nessas duas situações, é sempre o mesmo Particular Paulo que está sendo referenciado. Exemplos de Endurants são casas, pessoas, a Lua, um buraco e um punhado de areia.

Perdurants, ou eventos, são indivíduos compostos de partes temporais. Os Perdurants acontecem no tempo, no sentido de que eles se estendem no tempo cumulando partes temporais. Exemplos de Perdurants são uma conversa, um jogo de futebol, uma sinfonia, uma festa de aniversário, a análise de um projeto de software, a Segunda Guerra Mundial e um processo de negócio. Sempre que um Perdurant está presente, não é o caso que todas as suas partes temporais estão presentes. Por exemplo, se for considerado o evento "ocorrência de conversas em uma sala de bate-papo" em diferentes partes do tempo em que ele está ocorrendo, em cada um desses instantes de tempo, somente algumas partes temporais estarão presentes, como a participação de uma pessoa X em um determinado período, mas a ausência dele em outro. Como conseqüência, perdurants não podem exibir mudança no tempo visto que nenhuma de suas partes temporais retém sua identidade ao longo do tempo.

A principal categoria dessa ontologia é Event (Evento) – também podendo ser chamada de Perdurant ou Occurrent. Eventos podem ser atômicos (Atomic Event) ou complexos (Complex Event), dependendo de suas estruturas merológicas, ou seja, enquanto eventos atômicos não possuem subpartes, os eventos complexos são agregações de pelo menos dois outros eventos – que por sua vez podem ser atômicos ou complexos.

Eventos possibilitam transformações de uma porção da realidade para outra, ou seja, os eventos podem mudar a realidade mudando o estado de uma (pre-state) situation para uma (post-state) situation. A Figura 2.4 apresenta um fragmento da UFO-B.

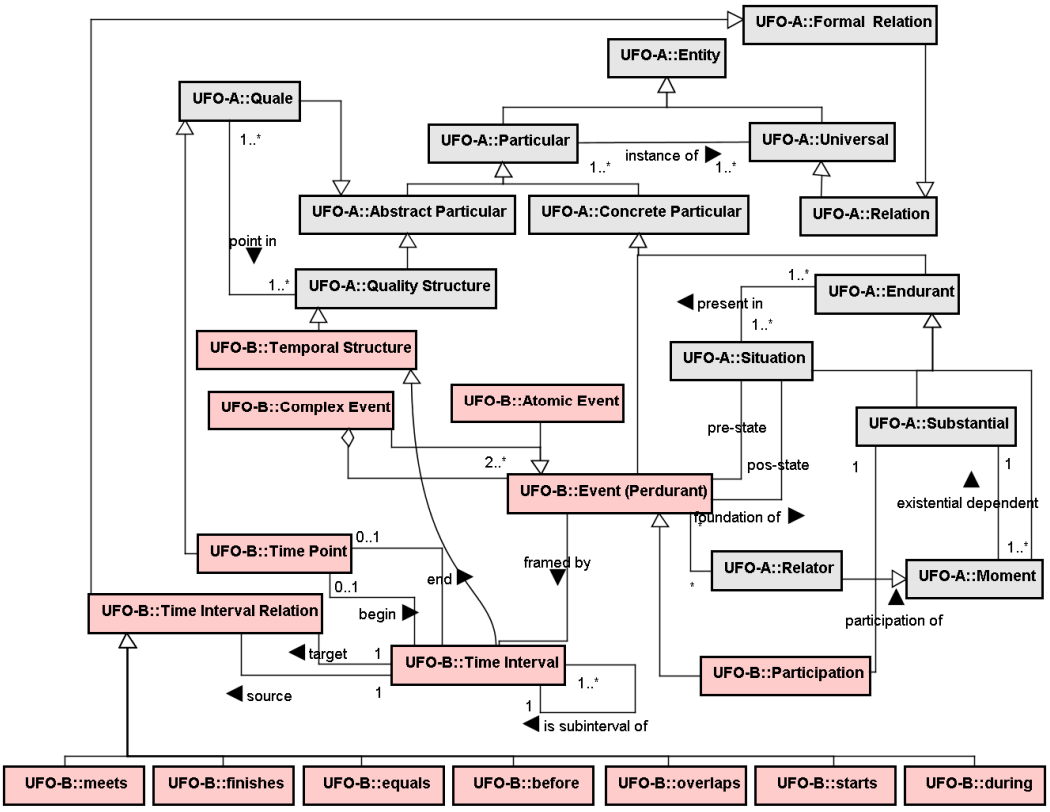


Figura 2.4 – Fragmento da UFO-B.

Os Eventos são entidades ontologicamente dependentes, no sentido que dependem existencialmente de seus participantes (Participants) para existir. Tome, por exemplo, o evento *e*: "a defesa de Gordon Banks" (na cabeçada de Pelé na Copa de 70). Nesse evento, há a participação de Pelé (cabeceando a bola), de Gordon Banks (defendendo a cabeçada) e da bola. Neste caso, *e* é composto da participação individual de cada uma

destas entidades. Cada uma destas participações é por si própria um evento, que pode ser atômico ou complexo, mas existencialmente dependente de um Particular.

É importante enfatizar que ser atômico e ser instantâneo são noções ortogonais nesta estrutura, isto é, as participações atômicas podem ser estendidas ao longo do tempo assim como um evento instantâneo pode ser composto de múltiplas participações (instantâneas). Em (MASOLO *et al.*, 2008), as propriedades especiais dos eventos são definidas em termos das propriedades espaciais de seus participantes. Ao contrário, as propriedades temporais dos Substantials são definidas em termos dos eventos nos quais participam. Analogamente ao que foi discutido com Endurants, as propriedades temporais dos eventos têm seus valores obtidos (seu Qualia) projetando estas propriedades em uma estrutura de qualidade (Quality Structure).

Aqui, define-se o espaço conceitual de tempo como sendo uma estrutura "composta de" Time Intervals (intervalos de tempo). Um intervalo de tempo, por sua vez, é "composto de" Time Points (pontos no tempo). Tais pontos no tempo podem ser representados como números reais e os intervalos de tempo como sendo um conjunto de números reais. Entretanto, podem também ser interpretados como entidades Sui Generis, como os Chronoids e os Time Boundaries da GFO(HELLER e HERRE, 2004b).

Por fim, adicionalmente, é admitida a existência de: **(i)** intervalos limitados por um ponto de início e por um de fim, ou podendo ter intervalos abertos; **(ii)** intervalos contínuos e não-contínuos; e **(iii)** intervalos com e sem duração instantes. Particularmente, esse modelo permite uma diversidade de estruturas temporais, como as de tempo linear, de ramificação, paralelo e circular. No caso de estruturas ordenadas, são utilizados os chamados Allen's Relations (ALLEN, 1983), as relações temporais entre os intervalos, de onde se deriva as relações correspondentes entre eventos. Essas relações são vistas da Figura 2.5.

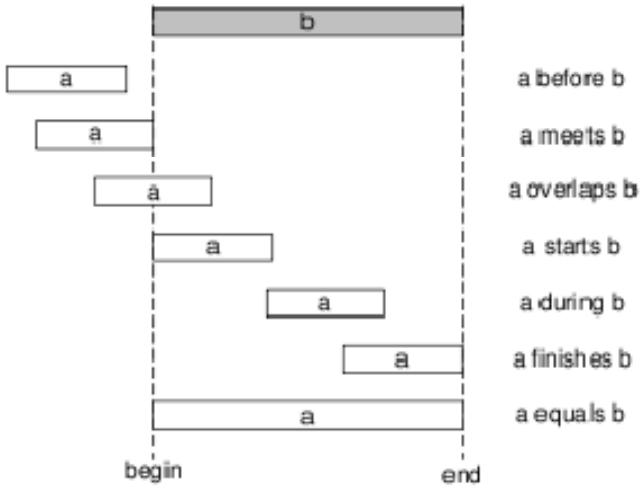


Figura 2.5 - Exemplificando os Allen's Relations: relações temporais entre intervalos de tempo.

2.3.1.3 UFO-C

A UFO-C é uma ontologia de entidades sociais (ambos Endurants e Perdurants). A UFO-C é baseada nos conceitos de Object e Moment Individual definidos na UFO-A, e no conceito de evento especificado na UFO-B. A ontologia começa a ser entendida pela distinção entre Substantials que são denominados agentes (Agents) e os objetos (Objects). A Figura 2.6 apresenta um fragmento da UFO-C.

Os agentes (Agent) são Sociais (Social Agent) como, por exemplo, uma organização ou uma sociedade; e não sociais como um software. Deve ser ressaltada a diferença entre um agente humano (Human Agent) ou biológico (Biological Agent), um agente artificial (Artificial Agent) e um agente institucional (Institutional Agent), que são, respectivamente, agentes representados por seres humanos, entidades computacionais (em geral, software), e agentes que representam organizações ou subunidades da organização (tais como departamentos, áreas e divisões). Os agentes institucionais são compostos de diversos (outros) agentes, que podem ser eles mesmos seres humanos, artificiais ou institucionais.

Agentes são Substantials que podem possuir tipos especiais de propriedades chamados Intentional Moments (Propriedades Intencionais). Como defendido em (SEARLE, 2000), intencionalidade deve ser entendida em um contexto mais amplo que a noção de "ter a intenção de algo" – deve ser entendida como a capacidade de algumas

propriedades de certos indivíduos referenciar certas situações da realidade. Todo Intentional Moment tem um tipo (e.g., Belief (Crença), Desire (Desejo), Intention (Intenção)), e um conteúdo proposicional (Proposition). O conteúdo proposicional é entendido como sendo uma representação abstrata de uma classe de situações referenciadas por um Intentional Moment. (Dessa maneira, "ter a intenção de algo" (Intending Something) é um tipo específico de intencionalidade chamada Intention).

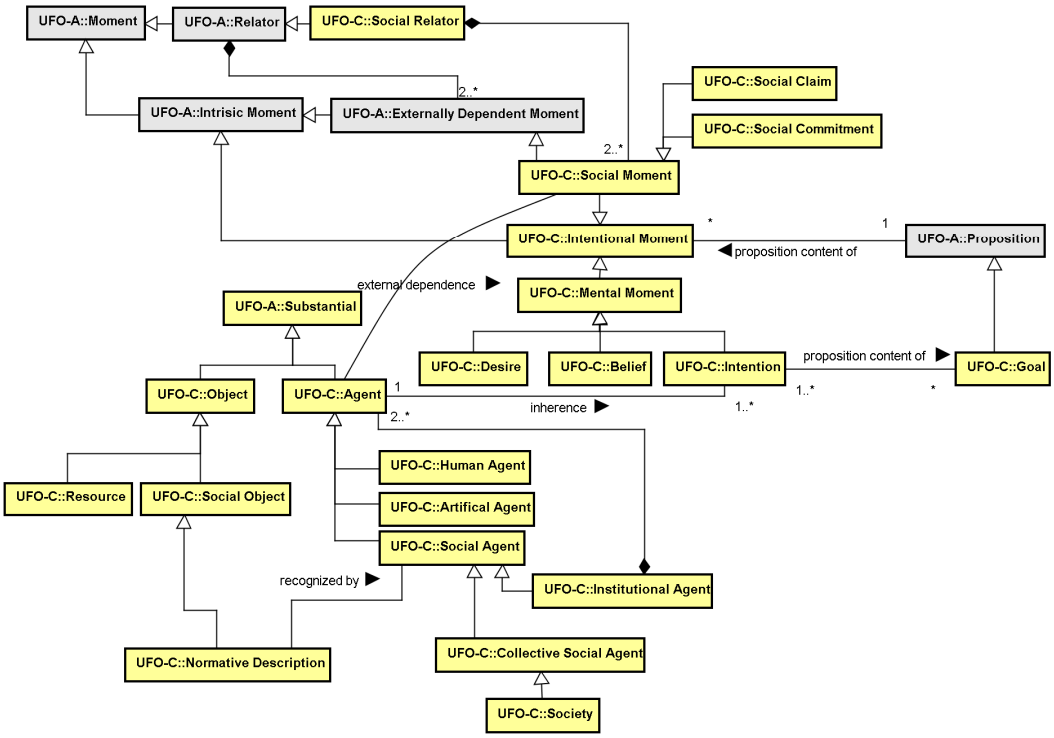


Figura 2.6 - Fragmento da UFO-C.

O conteúdo proposicional de uma intenção é um Goal (Objetivo). Uma descrição precisa da relação entre um Intentional Moment e uma Situation (da UFO-B) é a seguinte: situações da realidade podem satisfazer o conteúdo proposicional de uma intenção, ou seja, satisfazer – no sentido da lógica – a proposição representada pelo conteúdo proposicional.

Crenças podem ser justificadas por situações da realidade. Como exemplos, inclui-se a crença de que Roma é a capital da Itália e a de que a Lua orbita a Terra. Desejos e intenções podem ser satisfeitos (atingidos) ou frustrados. Enquanto um desejo expressa uma vontade de um agente que um estado do mundo torne-se real (e.g., o desejo de que

o Brasil ganhe a próxima Copa do Mundo), intenções são estados do mundo desejados pelos quais o agente busca atingir – Internal Commitments (compromissos internos, ou compromissos pessoais), como a intenção de passar de ano (CONTE e CASTELFRANCHI, 1995)(SEARLE, 2000). Por essa razão, intenções, ou compromissos internos, fazem com que os agentes executem ações.

Os objetos (Object) podem também ser categorizados em objetos sociais (Social Object) e não-sociais. Os objetos não-sociais incluem um livro, uma árvore ou um carro. Objetos sociais incluem dinheiro, linguagem e Descrições Normativas (Normative Description).

Uma Normative Description define uma ou mais regras, ou normas, reconhecidas por pelo menos um agente social (Social Agent), permitindo-se, dessa maneira, a definição de universais nominais, como tipos de propriedades sociais (e.g., tipos de compromissos sociais), objetos sociais (a coroa do rei da Espanha) e papéis sociais (Social Roles), como presidente, primeiro ministro e pedestre. Exemplos de Normative Description incluem a constituição Italiana e a regulamentação do programa de Mestrado em Informática da Universidade Federal do Espírito Santo, e também um conjunto de diretivas em como realizar certas ações dentro de uma organização – uma descrição de um plano(BOTTAZI e FERRARIO, 2006).

Actions (ações) são eventos intencionais, ou seja, eventos que instanciam um plano (Plan), ou tipo de ação (Action Universal), com o propósito específico de satisfazer (o conteúdo proposicional de) algum compromisso interno. Exemplos de ações são escrever um artigo, um processo de negócio e um ato de comunicação.

Como os eventos, as ações podem ser atômicas (Atomic Action) ou complexas (Complex Action). Uma ação complexa é composta de duas ou mais participações (Participation). As participações podem ser eventos intencionais (i.e., serem elas mesmas ações) ou eventos não-intencionais. Por exemplo, a punhalada de Brutus em Cesar inclui a participação intencional de Brutus e a não-intencional do punhal. Em outras palavras, seguindo a teoria filosófica de ações, tem-se que não é o caso que qualquer participação de um agente é considerada uma ação, mas somente aquelas participações intencionais – chamadas aqui de Action Contributions. Somente

agentes – entidades capazes de ter Intentional Moments – podem executar ações. Quando um objeto participa de uma ação é dito que existe uma Object Participation de um Object (Non-Agentive). A Figura 2.7 apresenta um fragmento da UFO-C com foco na entidade Action.

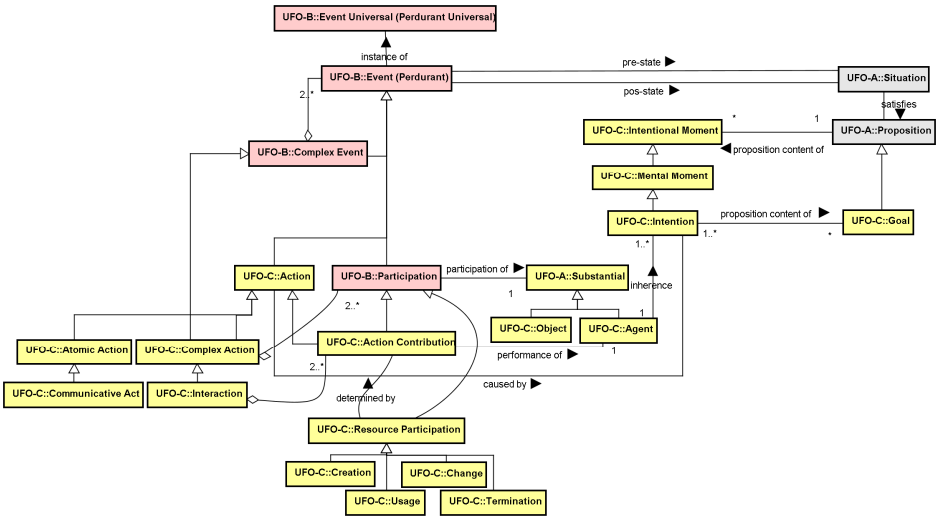


Figura 2.7 – Fragmento da UFO-C com foco na entidade Action.

Uma ação complexa, composta de Action Contributions de diferentes agentes, é chamada de Interaction (interação). Dois artistas colaborando para criar uma escultura é um exemplo de uma interação, assim como um diálogo entre dois agentes. Nesse caso citado, a escultura, bem como as ferramentas e as matérias-primas usadas para criá-la, são exemplos de Object (Non-agentive). Objetos podem participar de ações de diferentes maneiras. Há quatro diferentes modos de participação de Object (Non-agentive) (Resource Participation): criação (Creation), finalização (Termination), mudança (Change) e uso (Usage), que podem ser caracterizados do seguinte modo: seja *r* um recurso, *a* uma ação e s_1, s_2 duas situações que são, respectivamente, pre-state e post-state da ação *a*. Então, define-se:

i. Creation: a participação do Object (Non-agentive)*r* em a é uma criação se, e somente se, (*r* não está presente em s_1) e (*r* está presente em s_2) e (existe pelo menos uma Action Contribution *ac* que também é parte de *a* e que s_2 satisfaz o conteúdo proposicional (objetivo) de *ac*);

ii. Termination: a participação do Object (Non-agentive)*r* em a é uma destruição se, e somente se, (*r* está presente em s_1) e (*r* não está presente em s_2) e (existe pelo menos um Action Contribution *ac* que também é parte de *a* de tal modo que s_2 satisfaz o conteúdo proposicional de *ac*);

iii. Change: a participação de um Object (Non-agentive)*r* em *a* é uma mudança se, e somente se, existe pelo menos uma propriedade *m* tal que ((*m* é inerente a *r* em s_1 e não é inerente a *r* em s_2) ou (m não é inerente a r em s_1 e m é inerente a *r* em s_2)) e (existe pelo menos um Action Contribution *ac* que também faz parte de *a* e que s_2 satisfaz o conteúdo proposicional de *ac*);

iv. Usage: a participação de um Object (Non-agentive)que não é nenhum dos três tipos de participação anteriormente mencionados é uma participação de recurso do tipo *Usage*.

Como discutido em (GUIZZARDI *et al.*, 2007a) uma participação de recurso pode ser a causa de uma dependência de recurso (Resource Dependence) e o resultado de uma delegação de recurso (Resource Delegation) entre agentes. Em uma delegação de recurso, o agente A dá uma permissão de recurso r ao agente B. Para que isso aconteça, A deve ter o direito de conceder a permissão ao agente B e, além disso, a possibilidade de conceder o modo correto de permissão (por exemplo, permissão de usar ou modificar).

Communicative Acts (Atos de comunicação) podem ser usados criar Social Moments. Nesta visão, uma linguagem representa não somente a realidade, mas cria também uma parte da realidade (SEARLE, 2000). Assim, Social Moments são tipos de Intentional Moments que são criados pela troca de Communicative Acts e pelas conseqüências destas trocas (por exemplo, adoção de um objetivo ou uma delegação (GUIZZARDI *et al.*, 2007a). Por exemplo, suponha o aluguel de um carro em um serviço de aluguéis de carros; ao assinar um acordo de negócio, um agente executa um Communicative act (neste caso uma promessa). Este ato cria um compromisso social (Social Commitment) para essa organização: um compromisso de "retornar o carro em boas condições de uso, etc" (o conteúdo proposicional). Além disso, cria também uma reivindicação social (Social Claim) dessa organização para o agente.

Uma relação social (Social Relator) é um exemplo de um *relator* composto de dois ou mais pares de compromissos/reivindicações associados (Social Moments). Finalmente, um compromisso (Commitment) (interno ou social) é cumprido (Fulfilled) por um agente *a* se este agente executar uma ação *x* tal que o pós-estado dessa ação seja uma situação que satisfaça ao conteúdo proposicional desse compromisso.

A Figura 2.8 descreve os conceitos de dependência (Dependency) e de delegação (Delegation) na UFO-C (GUIZZARDI *et al.*, 2007a). Uma primeira diferenciação reside no fato de uma Dependency constituir uma Formal Relation e uma Delegation consistir em uma Material Relation.

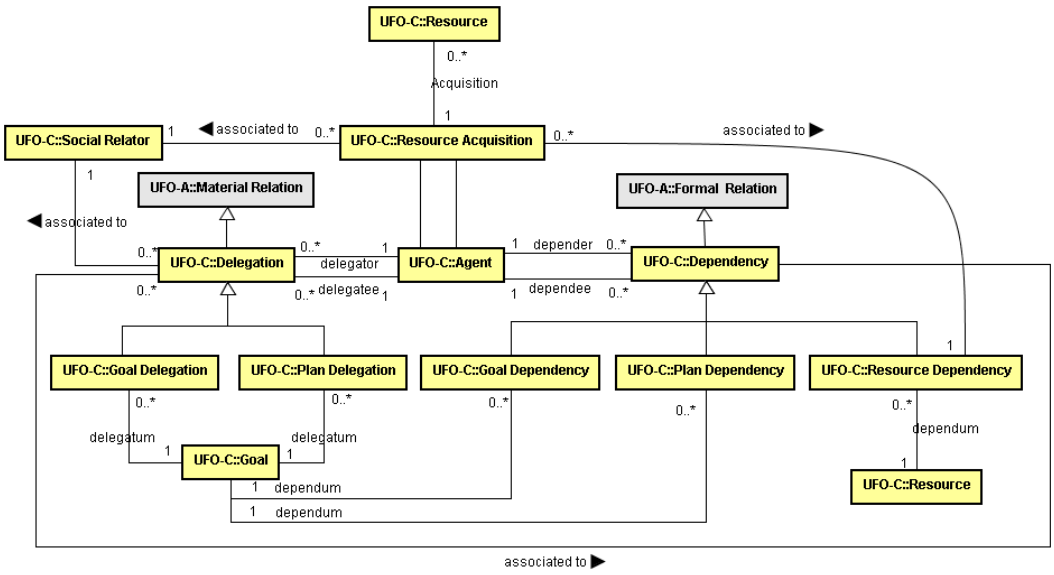


Figura 2.8 - Fragmento da UFO-C que distingue as relações de Dependência, Delegação e Aquisição, (Guizzardi et al., 2007a).

A figura acima mostra que uma dependência faz uma conexão entre dois agentes físicos, chamados de depender e dependee, e um dependum (objeto de dependência), cuja natureza define o tipo de dependência entre os agentes. Um agente A (depender) depende de um agente B (dependee) a respeito de um objetivo G se G for um objetivo do agente A, mas A não pode realizar G, mas o agente B pode alcançá-lo. Se um agente não pode realizar um objetivo, isso pode significar que esse agente não possui a capacidade de alcançar esse objetivo ou ainda pode denotar que para alcançar esse objetivo, o agente pode interferir e contradizer suas outras intenções. Dessa maneira, o

agente A decide então delegar a realização do objetivo ao agente B. Uma delegação é assim associada a uma dependência. Entretanto, como uma Material Relation, uma relação de delegação pode significar mais do que ligar elementos. Nesse caso, os elementos conectados são dois agentes (Delegator e Delegatee) e um objetivo (Delegatum), e a fundamentação desta Material Relation é o Social Relator (isto é, o par de compromissos e reivindicações) estabelecido entre os dois agentes envolvidos nesta delegação (Delegation). Ou seja, quando um agente A delega um objetivo G a um agente B, além do fato de A depender de B a respeito de G, B compromete-se a realizar G em nome do objetivo de A.

Por fim, uma delegação de um objetivo (Goal Delegation) define o que é chamado de uma delegação aberta (Open Delegation), o que significa que a decisão a respeito da estratégia para a realização do objetivo cabe ao Depender. Uma delegação de um plano (Action Universal (Plan)) é chamada de uma delegação fechada (Close Delegation), que descreve uma estratégia específica (isto é, definida pelo plano), que o Depender deve adotar para a realização do objetivo delegado.

2.4 DESENVOLVIMENTO ORIENTADO A MODELOS

Atualmente existe uma grande distância entre a etapa de elicitação de requisitos e o desenvolvimento (codificação) de sistemas computacional (ALMEIDA *et al.*, 2006a)(ALMEIDA, 2006), tornando, assim, o desenvolvimento de um aplicativo computacional demorado e custoso. O Desenvolvimento Orientado a Modelos é uma das possíveis soluções para aumentar a velocidade e diminuir o custo do desenvolvimento de um aplicativo computacional.

O Desenvolvimento Orientado a Modelos é uma abordagem de desenvolvimento de sistemas computacionais, que aumenta o poder da utilização de modelos para esta finalidade(OMG, 2003). Tal abordagem proporciona um meio de utilizar modelos no curso do entendimento, projeto, construção, implantação, operação, manutenção e modificação de sistemas computacionais(OMG, 2003). Os modelos são expressos em linguagens de modelagem que possuem sua sintaxe abstrata descritas em modelos conhecidos como metamodelos (ALMEIDA *et al.*, 2006b).

2.4.1 Processo de Desenvolvimento Orientado a Modelos

O Processo de Desenvolvimento Orientado a Modelos é caracterizado como uma seqüência de etapas de projeto que possuem como finalidade o desenvolvimento de um sistema computacional. Para cada etapa de projeto, atividades de projeto são executadas. Cada atividade de projeto contém uma atividade de transformação (do inglês, *transformation activity*) e uma atividade de avaliação (do inglês, *assessment activities*) (ALMEIDA *et al.*, 2006a).

Uma atividade de transformação é uma atividade de projeto genérica que é responsável por produzir um modelo destino (do inglês, *target model*) tendo com base um modelo origem (do inglês, *source design*) e um ou mais requisito que devem ser satisfeitos. Uma atividade de avaliação é uma atividade de projeto genérica que possui como compromisso avaliar o modelo destino produzido pela atividade de transformação. A noção de modelo origem e destino depende da etapa do projeto(ALMEIDA *et al.*, 2006b). Em princípio, as atividades de avaliação devem incluir a avaliação de conformidade (do inglês, *conformance assessment*) para avaliar se o modelo destino está em conformidade com o modelo origem(ALMEIDA, 2006). A teoria sobre o relacionamento entre conformidade e transformações é apresentada na seção 2.4.3.

Durante o processo de desenvolvimento do sistema computacional, as atividades de transformação incorporam diversas decisões de projetos que adicionam características que eventualmente estarão associadas ao sistema em realização. Diferentes decisões de projeto guiam para diferentes alternativas de realização. A redução do espaço de realização imposta pelas sucessivas decisões de projeto é apresentada na Figura 2.9.

As decisões de projeto aplicadas em cada etapa do projeto devem reunir dois requisitos para que o processo de construção do sistema computacional tenha sucesso:

i. as decisões devem contribuir para satisfazer requisitos do sistema que ainda não foram cumpridos, e

ii. as decisões de projeto devem preservar as características que estão presentes no modelo origem, isto é, o modelo destino deve estar em conformidade com o modelo origem(ALMEIDA *et al.*, 2006a).

As decisões de projeto devem eventualmente guiar o projeto de desenvolvimento do sistema computacional, a fim de definir todas as características que são necessárias para o sistema em realização. A plataforma em que o sistema computacional irá ser desenvolvido, particularmente, define como as decisões de projeto podem ser realizadas. De forma similar, as decisões de projeto definem as possíveis plataformas em que o projeto pode ser realizado (ALMEIDA *et al.*, 2006a).

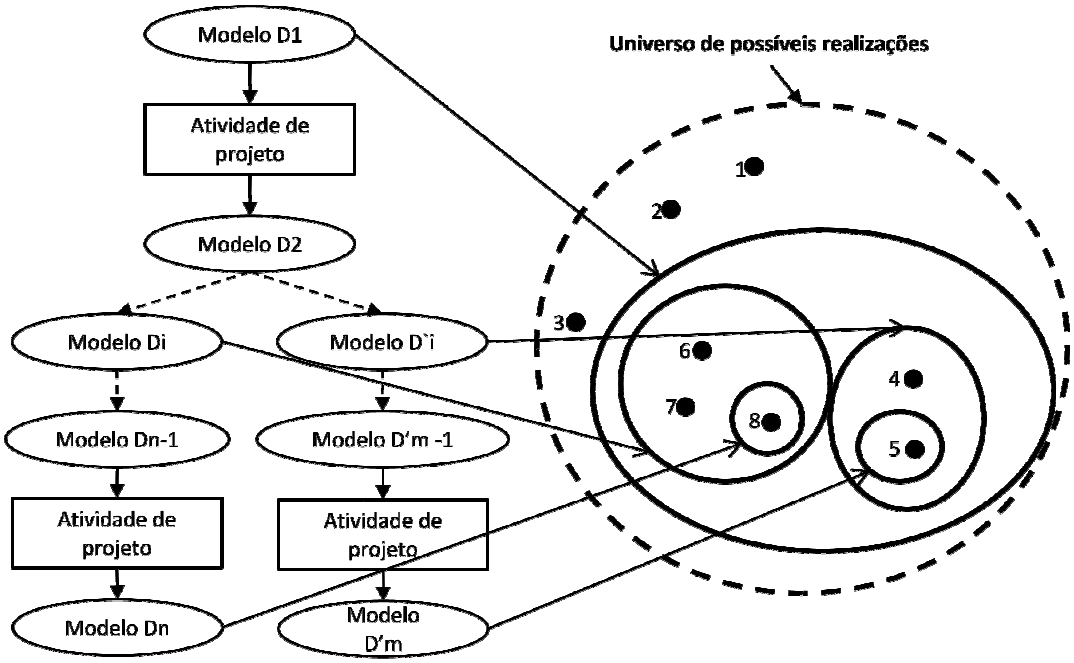


Figura 2.9 – Redução do espaço de soluções para um projeto com diferente níveis de abstração (ALMEIDA et al., 2006a).

2.4.2 Plataformas de Realização e Modelos Independentes de Plataforma

Uma plataforma de realização provê ao projetista um conjunto de construtores reusáveis. Estes construtores permitem ao projetista desenvolver um sistema computacional sem que o mesmo tenha a necessidade de se preocupar com a implementação dos construtores (ALMEIDA *et al.*, 2006a). Por prover um conjunto de construtores reusáveis, a plataforma de realização impõe um conjunto de regras no projeto do sistema computacional. Estas regras impostas pela plataforma de realização devem ser incorporadas nas etapas de projeto do sistema computacional em realização (ALMEIDA *et al.*, 2006a).

Quando os sistemas computacionais são descritos de forma a evitar depedências dos construtores de uma plataforma específica, os modelos utilizados são chamados de *modelos independentes de plataforma* (do inglês, *platform-independent models* – PIMs)(OMG, 2003). A independência de plataforma evita poluir modelos independentes de plataforma com interesses relacionados a restrições de uma plataforma específica, facilitando o reuso dos mesmos modelos para a construção de um sistema em diferentes plataformas (ALMEIDA *et al.*, 2006a).

2.4.3 Relacionamento entre Conformidade e Transformações

As regras de conformidade determinam o mínimo que deve ser preservado na etapa de projeto, ou seja, determinam o máximo de liberdade para a criação do modelo destino sem perder as decisões de projeto presentes no modelo origem. As especificações de transformação determinam o máximo que pode ser escrito na etapa de projeto, ou seja, determinam o resultado em um específico modelo destino (mínimo de libertada)(ALMEIDA *et al.*, 2006a). Isto é apresentado na Figura 2.10.

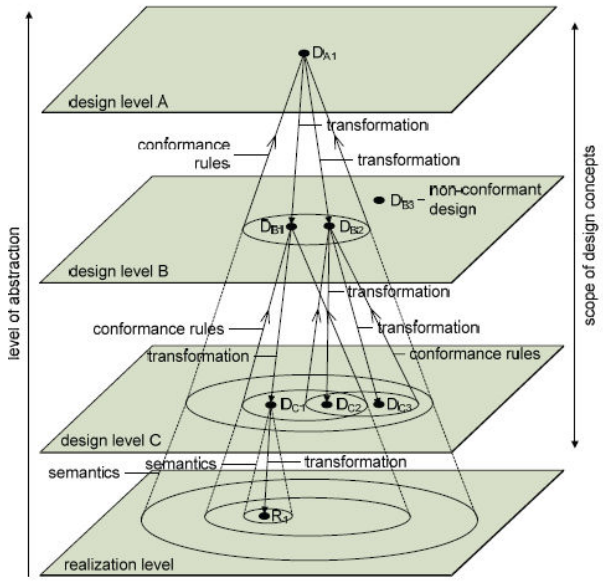


Figura 2.10 – Relação entre conformidade e transformações (ALMEIDA et al., 2006).

As regras de conformidade determinam implicitamente o conjunto de modelos destino que estão em conformidade com um modelo origem. Estas regras são "faróis" e um modelo origem "ilumina" uma área alvo no nível do modelo destino. Em contraste, uma

especificação de transformação (não-parametrizada ou determinística) é descrita como uma seta do modelo origem para o modelo destino (ALMEIDA *et al.*, 2006a).

A avaliação da conformidade pode ser realizada como conforme demonstrado na Figura 2.11. A Figura 2.11 ilustra o que é feito durante a construção de um modelo destino *Dt* a partir de um modelo origem *Ds*, com a inserção de características *c* (conseqüência das decisões de projeto que foram tomadas). Para considerar a conformidade de um modelo destino com relação a um modelo origem, devemos abstrair as características do modelo destino resultando assim em uma abstração modelo destino *Dt`*. Então, devemos checar se o modelo destino sem as características (*Dt`*) é equivalente (~) ao modelo origem(ALMEIDA *et al.*, 2006a). Para abstrair as características *c* que foram inseridas, é necessário conhecer as decisões de projeto que foram tomadas no passo de projeto que levou *Ds* a *Dt*. Se, por exemplo, durante a construção do modelo destino, uma interação foi inserida, nós devemos saber que a interação deve ser abstraída durante o processo de validação da conformidade (ALMEIDA *et al.*, 2006a).

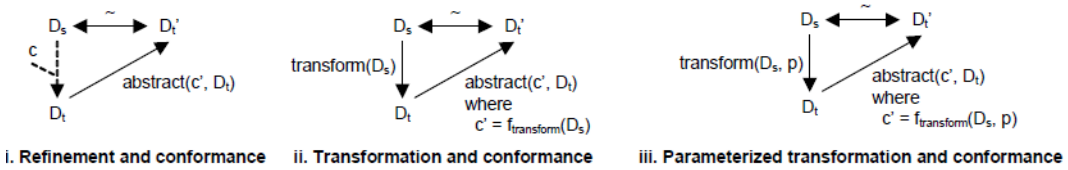


Figura 2.11 – Conformidade e transformação(ALMEIDA *et al.*, 2006a).

Uma transformação do modelo origem *Ds* para o modelo destino *Dt* é um caso especial de construção de um modelo destino. Neste caso, o modelo destino e as características inseridas são completamente determinados pelas regras de transformação e pelo modelo origem *Ds*. Conseqüentemente, as características *c'* que devemos abstrair durante a avaliação de conformidade podem ser automaticamente determinadas por uma função *ftransform* aplicada sobre o modelo origem *Ds*. Esta função deve ser definida como um complemento da transformação. Se tal função existe, é possível validar automaticamente a conformidade do modelo destino em relação ao modelo origem. A Figura 2.11 (ii) ilustra este caso.

A Figura 2.11 (iii) ilustra o caso das transformações parametrizadas. Para as transformações parametrizadas, as decisões de projeto que foram tomadas nas etapas de projeto são determinadas pelas regras de transformação, o modelo origem e os valores

dos parâmetros. Conseqüentemente, *c'* pode ser determinada por uma função *ftranform* no modelo origem *Ds* e valores dos parâmetros *p* (ALMEIDA *et al.*, 2006a).

Uma abordagem de Desenvolvimento Orientado a Modelos que preserva conformidade em suas transformações sempre produz modelos de destino que estão em conformidade com os modelos de origem. O benefício deste tipo de abordagem é que as atividades de validação de conformidade não tem que ser executadas manualmente para cada aplicação de transformação. Isto é particularmente beneficial para as abordagens de projeto iterativas em que as transformações são re-aplicadas com freqüência para lidar com as mudanças nos modelos de origem(ALMEIDA *et al.*, 2006a).

A fim de demonstrar que uma transformação de modelos garante a conformidade, é necessário demonstrar que para cada modelo origem *Ds*:

$$\text{abstract}(\text{transform}(D_s), f_{transform}(D_s)) \sim D_s.$$

Para as transformações parametrizadas, é necessário demonstrar que para cada modelo origem Ds e cada valor admissível do parâmetro de *Ds*:

$$\text{abstract}(\text{transform}(D_s), f_{transform}(D_s, p)) \sim D_s.$$

2.5 SISTEMAS DE INFORMAÇÃO ORIENTADOS A PROCESSOS DE NEGÓCIO

Um dos principais desafios enfrentados pelas organizações, atualmente, é transformar os conceitos e as idéias em produtos e serviços de maneira rápida e com menor custo e impacto possível em seus ambientes organizacionais (DUMAS *et al.*, 2007) .Para suprir estas necessidades organizacionais, diversos tipos de Sistemas de Informação (ALTER, 1999) tem surgido.

Nas décadas de 70 e 80, o desenvolvimento de sistemas de informação foi dominado pelas abordagens orientadas a dados (do inglês, *data-driven approach*). O foco da tecnologia da informação (TI) era o armazenamento, recuperação e apresentação das informações (primeiramente vista como dados) (DUMAS *et al.*, 2007). De certa forma,

a modelagem de dados foi o ponto de partida para o desenvolvimento de sistemas de informação nesta época. Porém, devido ao foco nos dados, os modelos de processos foram negligenciados. Como resultado, a lógica dos processos de negócio foi espalhada em diversos sistemas computacionais e processos manuais (DUMAS *et al.*, 2007). Além disso, os processos de negócio foram estruturados para se adaptarem às limitações dos sistemas computacionais, introduzindo assim a ineficiências ao processo de negócio como, por exemplo, alocação de recursos humanos desnecessários e uma separação "pobre" de responsabilidade para a execução das atividades organizacionais(DUMAS *et al.*, 2007).

A partir da década de 90 houve uma tendência voltada para o gerenciamento dos processos de negócio, desta forma, trazendo uma grande ênfase para os processos de negócio. Como resultado, os responsáveis pelo desenvolvimento de sistemas computacionais recorreram às abordagens orientadas a processos (do inglês, *processes–driven approach*).

Através das abordagens orientadas a processos estabeleceu o cenário para o surgimento dos sistemas de informação sensível a processo (do inglês, *process-aware information systems* - PAISs) (DUMAS *et al.*, 2007). DUMAS et al. (2007) define um sistema de informação sensível a processos como:

> "*um sistema computacional que gerencia e executa processos operacionais envolvendo pessoas, aplicativos computacionais, e/ou fontes de informações tendo como base um modelo de processo*".

Dada essa definição, pode ser visto que um editor de texto e um cliente de e-mail não são aplicações sensíveis ao processo, à medida em que estes são utilizados para facilitar a execução de uma ou mais tarefas específicas, sem tomar conhecimento do processo que a tarefa faz parte. Neste tipo de sistemas os usuários executam tarefas em uma ordem pré-estabelecida (DUMAS *et al.*, 2007).

Os sistemas de informação sensíveis a processos são construídos sobre uma infra-estrutura tecnológica que permite separar os sistemas computacionais e o processo organizacional. Exemplos destas infra-estruturas são sistemas de gerenciamento de fluxo de trabalho (do inglês, *workflow management systems*), *process-aware groupware* e algumas plataformas de integração de aplicações computacionais (do inglês, *Enterprise Application Integration Plataforms* - EAI).

Nos sistemas de informação tradicionais, o processo suportado por estes não é avaliado como um todo ou é codificado diretamente em um programa ("*hard-coded*"). A manutenção dos sistemas de informação tradicionais em que o código da aplicação é misturado com a lógica do processo é cara e propensa a erros. Durante o seu ciclo de vida, os processos requerem adaptações devido às mudanças organizacionais. As mudanças no processo em um sistema de informação tradicional requerem modificações no código fonte. Cada modificação no código fonte pode guiar a erros de programação ou a resultados inesperados. Por outro lado, o princípio central do gerenciamento de fluxo de trabalho é a separação da lógica do processo e das aplicações. As mudanças no processo podem ser feitas usando ferramentas de fluxo de trabalho sem ter a necessidade de re-escrever o código fonte do sistema (DUMAS *et al.*, 2007). Desta forma, basta apenas modificar os modelos de processo para evoluir o sistema de informação(VAN DER AALST e JABLONSKI, 2000). O mesmo princípio da remoção de funções genéricas dos aplicativos computacionais tem sido aplicada com sucesso na área Sistemas de gerenciamento de base de dados (do inglês, *Data-Base Management System*), em que a funcionalidade de gerenciar os dados (tais como processamento de pesquisas, controle de integração ou controle de concorrência) é levado para fora dos sistemas computacionais (DUMAS *et al.*, 2007).

A infra-estrutura dos sistemas de informação sensíveis a processos pode ser usada para evitar a codificação da lógica do processo nas aplicações computacionais e, deixando tais regras nos modelos de mais alto nível, facilitando, assim, o (re) projeto dos sistemas de informação. Mais do que isso, os sistemas de informação sensíveis a processo permitem o gerenciamento dos recursos das organizações de forma mais eficiente. Por exemplo, os sistema de gerenciamento de fluxo de trabalho e as plataformas de EAI permitem aos projetistas e as desenvolvedores implementarem as mudanças do processo através de modelos de processo de negócio (uma prática consistente com a MDA).

Em adição, o isolamento do gerenciamento de processo em um componente separado é consistente com o recente desenvolvimento de aplicações de integração do domínio de intra e inter-organizacional (por exemplo, *Web Services* e Arquiteturas Orientadas a Serviços) (DUMAS *et al.*, 2007).

Por fim, a construção de sistemas de informação utilizando modelos de processos permite ao Projetista uma visão global de todas as operações suportadas pelos sistemas de informação permitindo a redução de informações redundantes dentro das tarefas e provendo oportunidades para interconectar transações separadas (VAN DER AALST e VAN HEE, 2002) (JABLONSKI e BUSSLER, 1996.)(LEYMANN e ROLLER, 1999)

2.5.1 Sistemas de Gerenciamento de Fluxo de Trabalho

Um sistema de informação sensível ao processo associa usuários e outros recursos (por exemplo, sistemas computacionais) às tarefas do sistema ou, de outro ponto de vista, tarefas e sistemas computacionais são alocadas para os usuários. Além disso, é necessário um controlador das tarefas do sistema de informação. Esta funcionalidade pode ser provida por um sistema de gerenciamento de fluxo de trabalho (do inglês, *workflow management systems*) integrado ou trabalhando em parceria com um sistema de informação (DUMAS *et al.*, 2007).

O gerenciador de fluxo de trabalho visa apoiar a execução do fluxo de atividades de um processo de negócio de uma organização, de tal forma que as atividades sejam feitas no momento certo, pela pessoa certa e com o sistema computacional adequado. O foco do gerenciador de fluxo de trabalho é na estrutura do processo de trabalho e não no conteúdo individual de cada tarefa. As tarefas são suportadas por sistemas computacionais específicos. O gerenciador de fluxo de trabalho associa as pessoas aos seus sistemas computacionais para que as atividades sejam realizadas pelas mesmas, como apresentado na Figura 2.12 (DUMAS *et al.*, 2007).

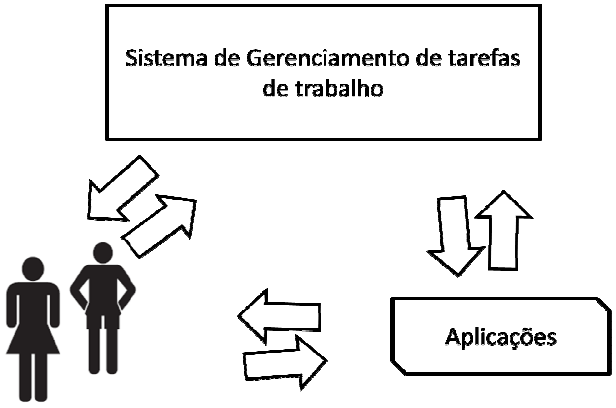


Figura 2.12 – Gerenciador de fluxo de trabalho como link entre Pessoas e Sistemas computacionais (DUMAS *et al.*, 2007).

Um sistema de fluxo de trabalho é um sistema de informação baseado em um sistema de gerenciamento de fluxo de trabalho (do inglês, *WorkFlow Management System* (WFMS)), que suporta um conjunto específico de processos de negócio através da execução de uma especificação de um processo de negócio(DUMAS *et al.*, 2007). Segundo DUMAS et al., 2007, uma especificação de processo de negócio descreve um tipo de processo de negócio que pode ser interpretado como um modelo para a execução concreta das instâncias do workflow.

Mudanças estruturais nos tipos de processos podem ser feitas utilizando ferramentas de modelagem de workflow sem a necessidade de re-escrever os sistemas computacionais. Para finalizar, o uso de Gerenciadores de Fluxo de Trabalho simplifica o desenvolvimento de sistemas computacionais, pois permitem o reuso de componentes computacionais e funcionalidades de processos, que são comuns a muitas aplicações computacionais, é provido pela arquitetura do sistema de gerenciamento de fluxo de trabalho(DUMAS *et al.*, 2007). No escopo deste trabalho, um Gerenciador de Fluxo de Trabalho específico (como por exemplo, JBOSS jBPM, SAP Workflow e Oracle BPM Studio) é considerado como uma plataforma específica. Desta forma, uma especificação de processo de negócio que pode ser executado por um Gerenciador de Fluxo de Trabalho é um modelo de processo dependente de plataforma e será o alvo das transformações a serem descritas na abordagem proposta.

2.6 CONCLUSÃO

Este capítulo apresentou um referencial teórico para embasar a abordagem de Desenvolvimento Orientado a Modelos proposta neste trabalho.

Primeiramente, foram discutidas as características que uma Arquitetura Organizacional de TI deve possuir para viabilizar a abordagem proposta. Dentre essas características, destaca-se o uso de uma linguagem de modelagem arquitetural que permita modelar diversos aspectos da organização (como, por exemplo, os processos de negócio e as interações entre as entidades organizacionais).

Dentre os diversos *frameworks* de Arquitetura Organizacional de TI foi destacado o ARIS (***AR****chitecture for integrated* ***I****nformation* ***S****ystems*), devido ao seu grande uso no meio acadêmico e organizacional. Os conceitos do ARIS Method apresentados na seção 2.2.1 serão de grande importância para a definição dos metamodelos apresentados no Capítulo 3.

Uma das características da abordagem de Desenvolvimento Orientado a Modelos proposta neste trabalho é o uso adequado da semântica dos elementos das linguagens de modelagem das Arquiteturas Organizacionais em todas as etapas da abordagem. A seção 2.3 apresentou os conceitos da ontologia de fundamentação UFO que será a base conceitual da análise ontológica apresentada no Capítulo 4.

Em relação ao Desenvolvimento Orientado a Modelos, o capítulo apresentou a definição formal desse processo de desenvolvimento de sistemas computacionais e também de outros conceitos como: transformação, parametrização e conformidade entre modelos. Tais conceitos são necessários para o entendimento por completo da abordagem de Desenvolvimento Orientado a Modelos apresentada no Capítulo 5.

Por fim, a seção 2.5 apresentou os conceitos sobre os sistemas de informação orientados a processo de negócio. Tais conceitos são fundamentais para o entendimento do tipo de sistemas de computacional que a abordagem de Desenvolvimento Orientado a Modelos proposta neste trabalho visa construir, como será ilustrado no uso da plataforma JBOSS jBPM (CUMBERLIDGE, 2007) no Capítulo 6.

CAPÍTULO 3 METAMODELOS DAS LINGUAGENS DO ARIS METHOD

3.1 INTRODUÇÃO

Conforme apresentado na seção 1.3, uma das características da abordagem de Desenvolvimento Orientado a Modelos proposta neste trabalho é possibilitar que o Projetista utilize a riqueza da semântica dos elementos de modelagem empregados para representar uma Arquitetura Organizacional de TI.

Para isto, é necessário um estudo aprofundado das linguagens de modelagem de Arquiteturas Organizacionais de TI. Este capítulo apresenta o primeiro passo deste estudo, considerando as linguagens de modelagem dos domínios de Estrutura Organizacional, Processo de Negócio e Detalhamento de Atividades do framework ARIS. A escolha por estas linguagens deve-se ao grande uso destas na modelagem de Arquiteturas Organizacionais de TI e devido à relevância dos modelos criados nestas linguagens para o desenvolvimento de sistemas orientados a processos.

Inicialmente, consideramos como fonte para a definição do ARIS Method o livro original do criador do método, Prof. Scheer (SCHEER, 1999). Apesar de Scheer (1999) apresentar a linguagem para representar cada uma das visões da metodologia ARIS, a implementação destas linguagens no ARIS Toolset não condiz com o apresentado por Scheer. Em outras palavras, há elementos de uma determinada linguagem de domínio que estão presentes na ferramenta de modelagem e não estão presentes nos modelos ou nas descrições textuais em (SCHEER, 1999) e vice-versa.

Por exemplo, o metamodelo organizacional apresentado em (SCHEER, 1999) inclui os elementos *Organizational Object* e *Profile* que não ocorrem no ARIS Toolset. Ao mesmo tempo, no ARIS Toolset inclui o elemento *System Organization Unit* que não ocorre no metamodelo organizacional apresentado em (SCHEER, 1999).

Essa deficiência na definição da linguagem cria empecilhos para o estudo e análise das linguagens do ARIS Method. Desta forma, as fontes definitivas para estudo dos

elementos sintáticos e semânticos das linguagens de domínio do ARIS são as ferramentas ARIS (ARIS Toolset, ARIS Architecture e dentre outras), uma vez que estas são de fato empregadas em larga escala no desenvolvimento dos modelos.

Como as ferramentas computacionais revelam diretamente apenas os elementos sintáticos concretos, a representação abstrata da linguagem suportada pela ferramenta deve ser "escavada" na ausência de documentação definitiva para esta representação. Esta representação da linguagem é chamada de *sintaxe abstrata* (HAREL e RUMPE, 2000) que pode ser parcialmente definida através de um metamodelo(OMG, 2003).

Para obter a sintaxe abstrata das linguagens adotas pelo ARIS Toolset, definimos uma abordagem que permite a escavação e a construção dos metamodelos das linguagens do ARIS Method. Os detalhes das etapas dessa abordagem são apresentados em(SANTOS JR. e ALMEIDA, 2008). O resultado da aplicação da abordagem são os metamodelos apresentados neste capítulo. Este capítulo discute ainda a semântica dos elementos e relacionamentos presentes nestes metamodelos.

Na ausência de especificação rigorosa da linguagem, a semântica das metaclasses de cada metamodelo de domínio foi inferida a partir da documentação on-line da ferramenta ARIS Toolset, de conhecimentos de modelagem adquiridos através da utilização do ARIS Toolset e das seguintes referências bibliográficas(DAVIS, 2001)(SCHEER, 1999).

Diferentemente das metaclasses, os metarelacionamentos não possuem semântica definida na literatura e nem na documentação on-line da ferramenta. Desta forma, a semântica inferida para os relacionamentos foi derivada a partir da semântica dos elementos relacionados, do conhecimento de modelagem adquirido através da utilização do ARIS Toolset e de exemplos de uso desses relacionamentos na literatura como em(DAVIS, 2001)(SCHEER, 1999).

Este capítulo é organizado da seguinte forma: as seções 3.2, 3.3 e 0 apresentam informalmente a semântica dos elementos e relacionamentos dos metamodelos das linguagens organizacional, de processo de negócio e de detalhamento de atividades; a seção 3.5 apresenta o conceito denominado ARIS Object que é comum a todas as

metaclasses dos metamodelos de domínio; a seção 3.6 apresenta uma discussão sobre elementos notacionais nos metamodelos de domínio; a seção 3.7 apresenta uma discussão sobre os metamodelos apresentados neste capítulo e a seção 3.8 as conclusões do capítulo.

3.2 METAMODELO DA LINGUAGEM ORGANIZACIONAL

As principais metaclasses do metamodelo da linguagem organizacional são: *Organization Unit Type*, *Organization Unit*, *Position*, *System Organization Unit Type*, *System Organization Unit*, *Location*, *Person*, *Person Type* e *Group*. A Figura 3.1 apresenta um fragmento do metamodelo organizacional com os elementos *Organization Unit*, *Organization Unit Type, Cost Center* e seus relacionamentos. Os elementos em cinza na figura são "elementos notacionais" no ARIS, ou seja, são apenas especializações simples de metaclasses. A seção 3.6 apresenta uma discussão sobre elementos notacionais no ARIS Method

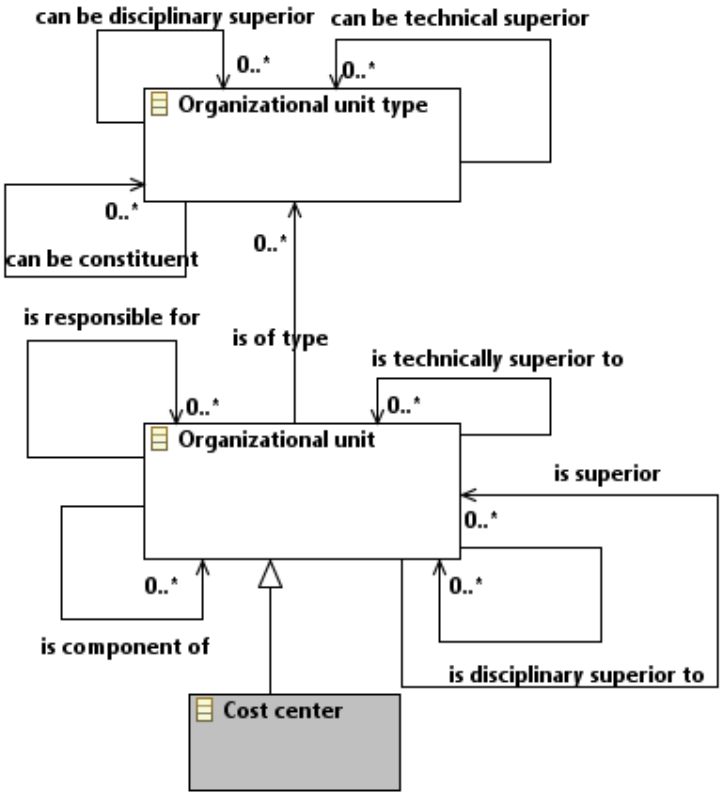


Figura 3.1 - Fragmento do metamodelo organizacional contendo Organization Unit, Organization Unit Type e seus relacionamentos.

O elemento *Organization Unit* representa uma entidade responsável pela execução de atividades que materializem os objetivos da organização. A Figura 3.2 apresenta os símbolos do elemento *Organization Unit*. O elemento *Organization Unit* possui o elemento notacional *Cost Center*.

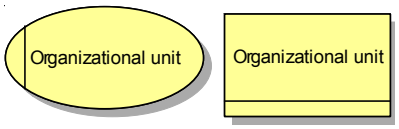


Figura 3.2 – Símbolos do elemento Organization Unit.

As unidades organizacionais (*Organization Unit*) podem ser categorizadas levando em consideração características em comum como, por exemplo, as responsabilidades e deveres que um conjunto de unidades organizacionais possuem. Desta forma, o ARIS Method permite que sejam criados tipos de unidade organizacionais que são representados pelo elemento *Organizational Unit Type*. O elemento *Organization Unit Type* possui o elemento notacional *Position Type.* A Figura 3.3 apresenta os símbolos do elemento *Organization Unit Type*.

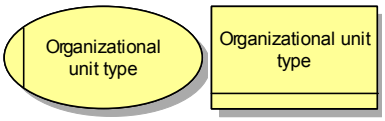


Figura 3.3- Símbolos do elemento Organization Unit Type.

O relacionamento *is of type* é utilizado para representar que uma determinada entidade "é do tipo" de outra entidade. Por exemplo, esse relacionamento é utilizado para expressar que uma *Organization Unit* é do tipo *Organization Unit Type*.

A documentação on-line , assim como (DAVIS, 2001) e (SCHEER, 1999) definem que a menor unidade organizacional é representada por uma posição (*Position*) que um indivíduo (*Person*) possui dentro da organização. As posições podem ser alocadas de acordo com o tamanho das unidades, levando em consideração as regras do negócio ou a estrutura da empresa (SCHEER, 1999). Várias posições podem ser associadas a uma unidade organizacional. As responsabilidades e deveres de uma posição (*Position*) são definidos no *Position Description* (vide Figura 3.9). A Figura 3.4 apresenta o símbolo do elemento *Position* e a Figura 3.5 apresenta um fragmento do metamodelo organizacional contendo *Position*, *Position Type*, *Organizational Unit Type* e *Organization Unit* e seus relacionamentos.

Figura 3.4 – Símbolo do elemento Position.

O relacionamento *is composed of* entre *Organization Unit* e *Position* representa que uma unidade organizacional é composta por uma ou mais positions. Por exemplo, esta relação pode ser usada para representar que o Departamento de Vendas é composto pelos positions Vendedor e Gerente de Vendas.

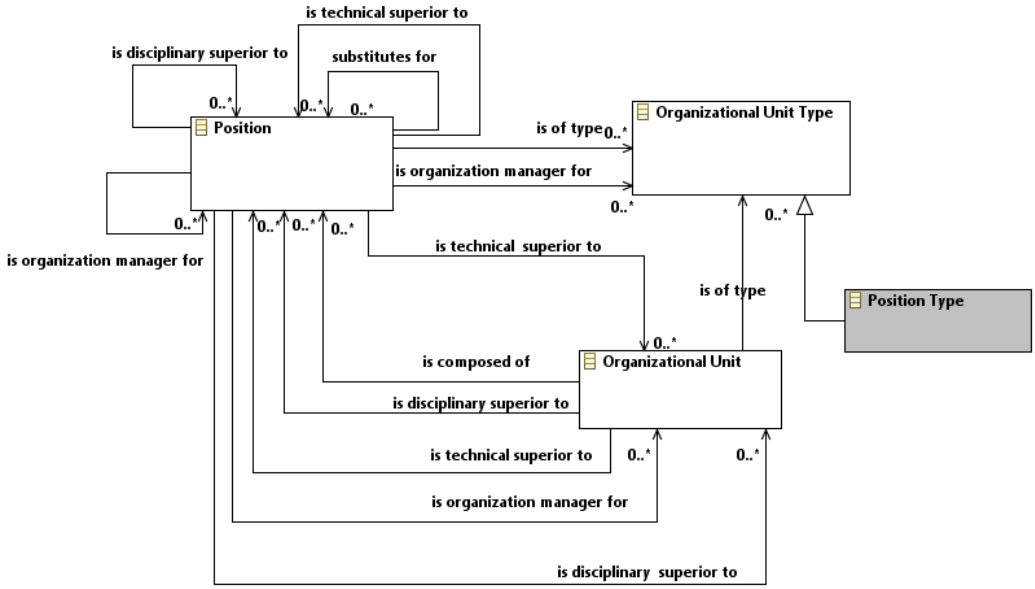


Figura 3.5 - Fragmento do Metamodelo Organizacional contendo os elemento Position, Position Type Organizacional Unit, Organizational Unit Type e seus relacionamentos.

O auto-relacionamento *is component of* do elemento *Organization Unit* representa que uma unidade organizacional é componente (é parte) de uma ou mais unidades organizacionais. Através desse relacionamento é possível modelar que o Departamento de TI e o Departamento de RH são componentes da uma empresa.

O auto-relacionamento *is responsible for* do elemento *Organization Unit* representa que uma unidade organizacional é a "responsável legal" por um ou mais unidades organizacionais. Assim, é possível expressar, por exemplo, que o Departamento de TI de uma empresa é responsável pelo departamento de Processo de Negócio da mesma.

Os relacionamentos *can be disciplinary superior* e *can be technical superior* do elemento *Organization Unit Type* especificam quais as unidades organizacionais

poderão ser consideradas superiores disciplinarmente e tecnicamente. O auto-relacionamento *can be constituent* do elemento *Organization Unit Type*, especifica quais os tipos de unidades organizacionais podem compor um outro tipo de unidade organizacional. Por exemplo, através desse relacionamento é possível expressar que o tipo de unidade organizacional Y pode ser composto pelos tipos de unidades organizacionais X e K.

Os relacionamentos e auto-relacionamentos *is disciplinary superior to*, *is technical superior to* e *is superior to,*representam, respectivamente, que uma determinada entidade é superior disciplinarmente a outra entidade; que uma determinada entidade é superior tecnicamente a outra entidade e; que uma entidade é superior hierarquicamente que a outra entidade.

A associação de uma pessoa (*Person*) a uma unidade organizacional expressa que essa pessoa é um empregado da mesma e a associação de uma pessoa (*Person*) a uma posição (*Position*) define qual o status dela na organização, as funções e as responsabilidades dentro da organização. O elemento *Person* não possui símbolo, porém, os seus elementos notacionais *External Person* e *Internal Person* possuem. A Figura 3.6 apresenta o símbolo do elemento notacional *External Person* e o símbolo do elemento notacional *Internal Person*.

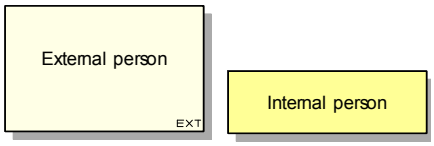


Figura 3.6 – Símbolo dos elementos notacionais do elemento Person.

A Figura 3.7 apresenta um fragmento do Metamodelo Organizacional com foco no elemento Person. O relacionamento *occupies* entre *Person* e *Position* indica que uma pessoa ocupa uma determinada posição na organização, por exemplo, Paulo ocupa a posição de Gerente no Departamento de TI.

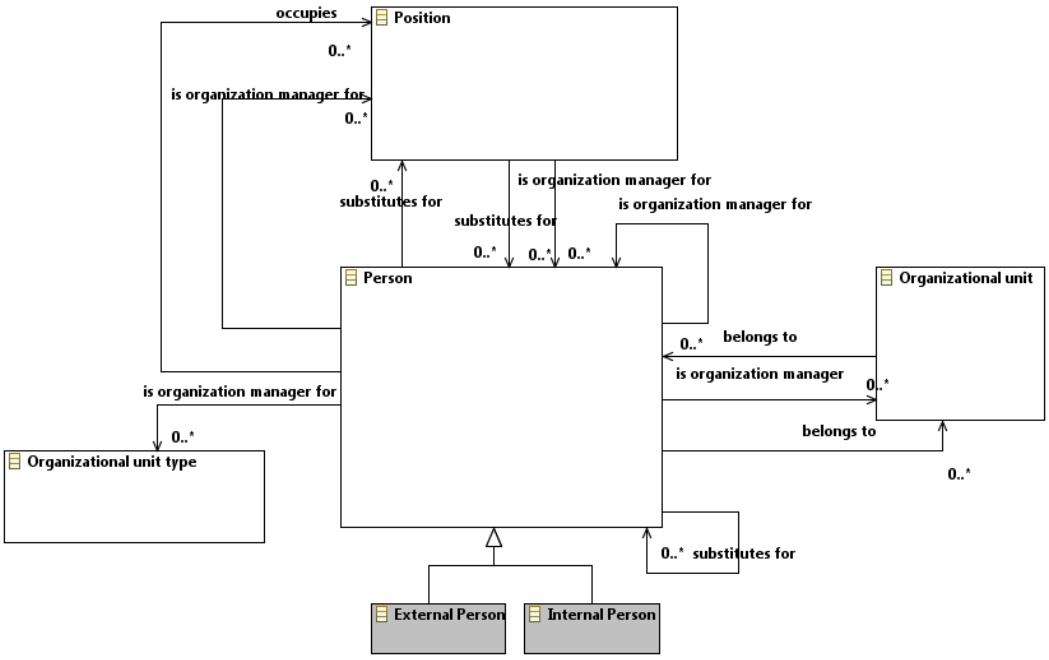


Figura 3.7 – Fragmento do Metamodelo Organizacional com foco no elemento Person.

O relacionamento *belongs to* entre *Organization Unit* e *Person* (*Organization Unit*→ *belongs to* → *Person*) representa que uma organização pertence a uma pessoa, expressando a idéia de posse ou que uma pessoa é o "responsável legal" pela organização. Por exemplo, José da Silva é o responsável "legal" pela empresa X.

O relacionamento *belongs to* entre *Person* e *Organization Unit* (*Person* → *belongs to* → *Organization Unit*) representa que uma pessoa (*Person*) faz parte de um ou mais unidades organizacionais. É possível expressar através desse relacionamento que o empregado X trabalha na unidade organizacional Y.

O relacionamento *is organization manager for* que ocorre entre *Positon* e *Person*, *Person* e *Organization Unit*, *Person* e *Organization Unit Type*, *Position* e *Organization Unit* e *Position* e *Organization Unit Type* representa que uma entidade é gerenciada por outra entidade. Por exemplo, que o Gerente de RH gerencia o Departamento de RH ou que o empregado E é gerenciado pela position P.

O relacionamento *substitutes for* que ocorre *Position* e *Person* entre representa que uma entidade pode ser substituída por um ou mais entidades. O elemento *Person* possui

um auto-relacionamento com o mesmo nome. Assim, é possível expressar que o empregado X foi substituído pelo empregado Y na organização.

Da mesma forma que as unidades organizacionais (*Organizational Unit*), as pessoas (*Person*) também podem ser classificadas levando em consideração uma característica em comum como, por exemplo, as responsabilidades e deveres de um conjunto de empregados. Desta forma, o ARIS Method permite que sejam criados tipos de pessoas (*Person Type*) como, por exemplo, gerente de departamento, líder de grupos ou gerente de projeto. A Figura 3.8 apresenta o símbolo do elemento *Person Type* e a Figura 3.9 apresenta um fragmento do metamodelo organizacional com foco no elemento *Person Type* e seus relacionamentos.

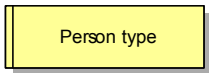


Figura 3.8 – Símbolo do elemento Person Type.

Segundo a documentação on-line, o beneficio de utilizar os tipos (como, por exemplo, *Organizational Unit Type* e *Person Type*) é possibilidade de representar regras de negócios genéricas que são impostas a unidades organizacionais reais ou a pessoas da organização. Um exemplo disso é a possibilidade de especificar que somente um certo tipo de pessoa (*Person Type*) poderá desempenhar uma certa função ou ter acesso a uma determinada informação da organização. O elemento *Person Type* possui o elemento notacional *Position Description*

O relacionamento *performs* ocorre entre *Person Type* e todos os demais elementos to metamodelos (com exceção dos elementos *System Organization Unit* e *System Organization Unit Type*) representa que uma determinada entidade desempenha (instância) um *Person Type*. Por exemplo, através desse relacionamento é possível expressar que uma pessoa desempenha um *Person Type* chamado “Gerente de projetos” em uma unidade organizacional.

O relacionamento *belongs to* representa que um *Person Type* pertence a uma unidade organizacional (*Organization Unit*) ou a um grupo (*Group*), ou seja, esse relacionamento é utilizado para expressar, por exemplo, o *Person Type* “Gerente de Projeto” pertence ao Departamento de TI.

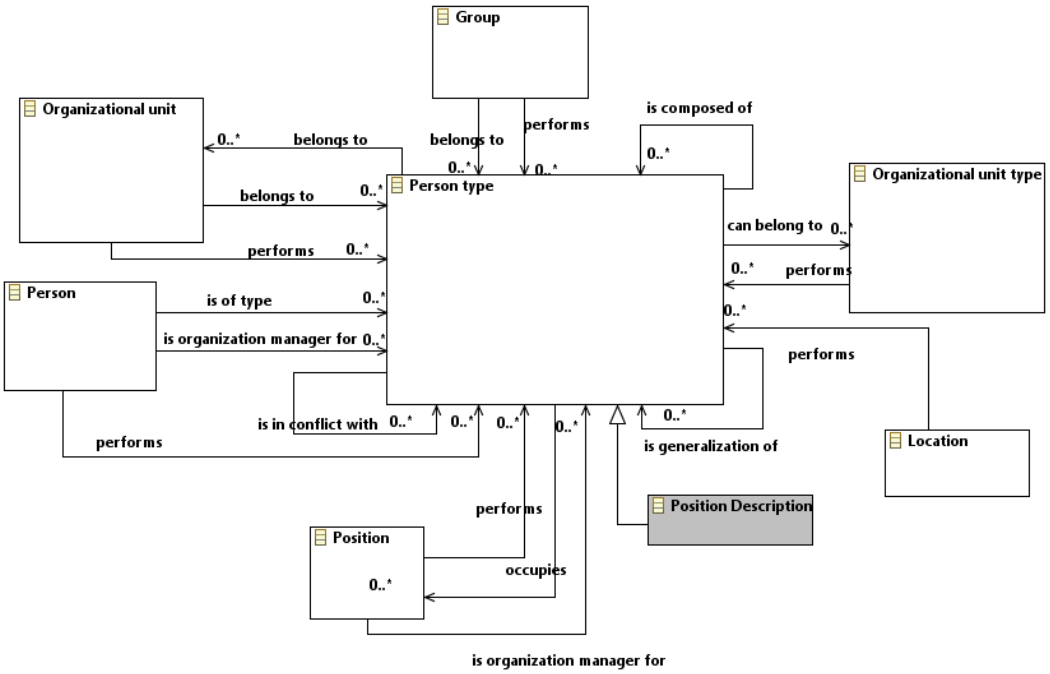


Figura 3.9 - Fragmento do metamodelo organizacional com foco no elemento Person Type e seus relacionamentos.

O relacionamento *can belongs to* entre *Person Type* e *Organization Unit Type* representa uma especificação de quais Person Type podem pertencer a um tipo de unidade organizacional. Por exemplo, é possível modelar que o Person Type "Gerente de TI" pode pertencer um tipo de unidade organizacional de TI.

O relacionamento *belongs to* entre os elementos *Group* (Grupo) e *Person Type* representa que um ou mais *Person Type* pertencem a um grupo, ou seja, fazem parte de um grupo. Assim, através desse relacionamento é possível descrever que o *Person Type* Y pertence ao grupo K.

O auto-relacionamento *is generalization of* e o auto-relacionamento *is composed of* representam, respectivamente, que um *Person Type* é uma generalização de outro *Person Type* e que um *Person Type* é composto de outros *Person Types*. Logo, através desses relacionamentos é possível expressar, respectivamente, que um *Person Type* Y é uma generalização de outros dois *Person Types* K e Z; e que o *Person Type* J é composto por outros dois *Person Types*.

O auto-relacionamento *is in conflict with* representa que um determinado *Person Type* está em conflito com outro *Person Type*. (Não há semântica detalhada para este relacionamento na literatura.)

O elemento *Location* representa a posição geográfica de uma unidade organizacional (*Organizational Unit*), de uma pessoa (*Person*), de uma posição (*Position*) ou de um recurso da organização (Colocar modelo de maquina/sistemas). Uma localização (*Location*) pode representar uma região, uma cidade, uma planta de chão de fábrica, um prédio, uma sala ou uma estação de trabalho.O elemento *Location* possui o elemento notacional *Workstation.* A Figura 3.10 apresenta o símbolo do elemento Location e a Figura 3.11 apresenta um fragmento do metamodelo de processo organizacional com foco no elemento Location e seus relacionamentos.

Os relacionamentos *at location* e *is location* representam que uma determinada entidade está localizada em uma posição geográfica. Por exemplo, esses relacionamentos podem ser utilizados para representar que o position "Modelador de Processos de negócio" está localizado na unidade organizacional "Departamento de TI".

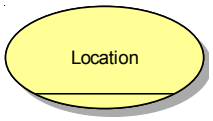


Figura 3.10 – Símbolo do elemento Location.

O auto-relacionamento *encompasses* é utilizado para representar que uma determinada região geográfica pode englobar outras regiões geográficas, por exemplo: podemos representar a localização "Brasil" como o conjunto de todas as posições geográficas dos estados brasileiros.

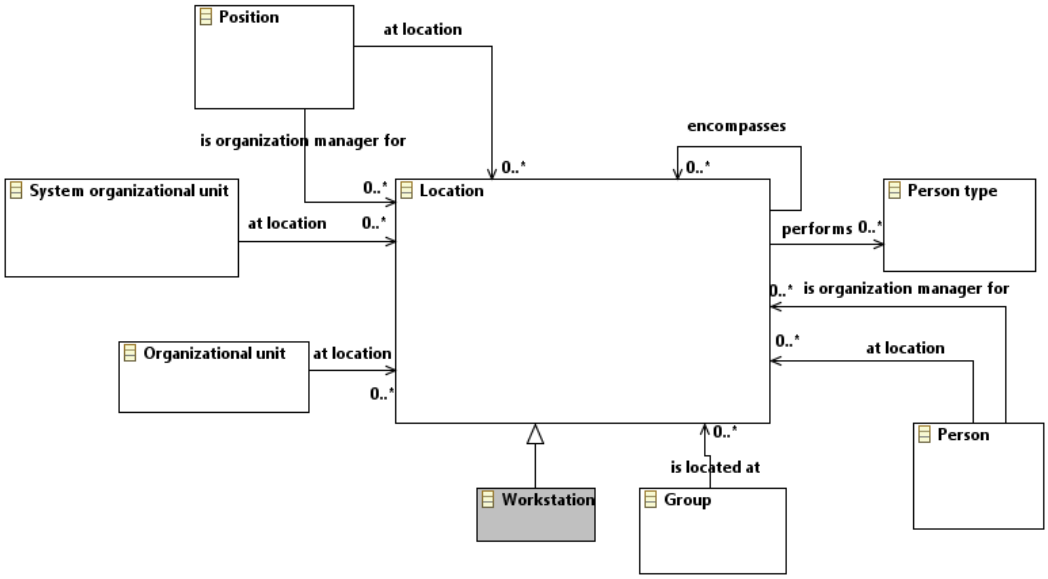


Figura 3.11 – Fragmento do metamodelo organizacional com foco no elemento Location e em seus relacionamentos.

Um grupo (*Group*) representa um grupo de empregados (*Person*) - com ou sem posições na organização - ou unidades organizacionais que estão trabalhando em conjunto por um período de tempo específico para um determinado objetivo. A Figura 3.13 apresenta um fragmento do metamodelo organizacional com foco no elemento *Group* e em seus relacionamentos e a Figura 3.12 apresenta o símbolo do elemento *Group*.

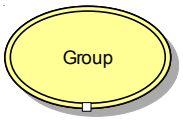


Figura 3.12 – Símbolo do elemento Group.

Os relacionamentos *is position of* e *is composed of* representam que um ou mais posições (*Position*) fazem parte de um grupo. Através desses relacionamentos é possível descrever, por exemplo, que o *Position* "Analista de processo de negócio" e o position "Analista de demandas" compõem o grupo chamado de "Grupo de validação de demandas".

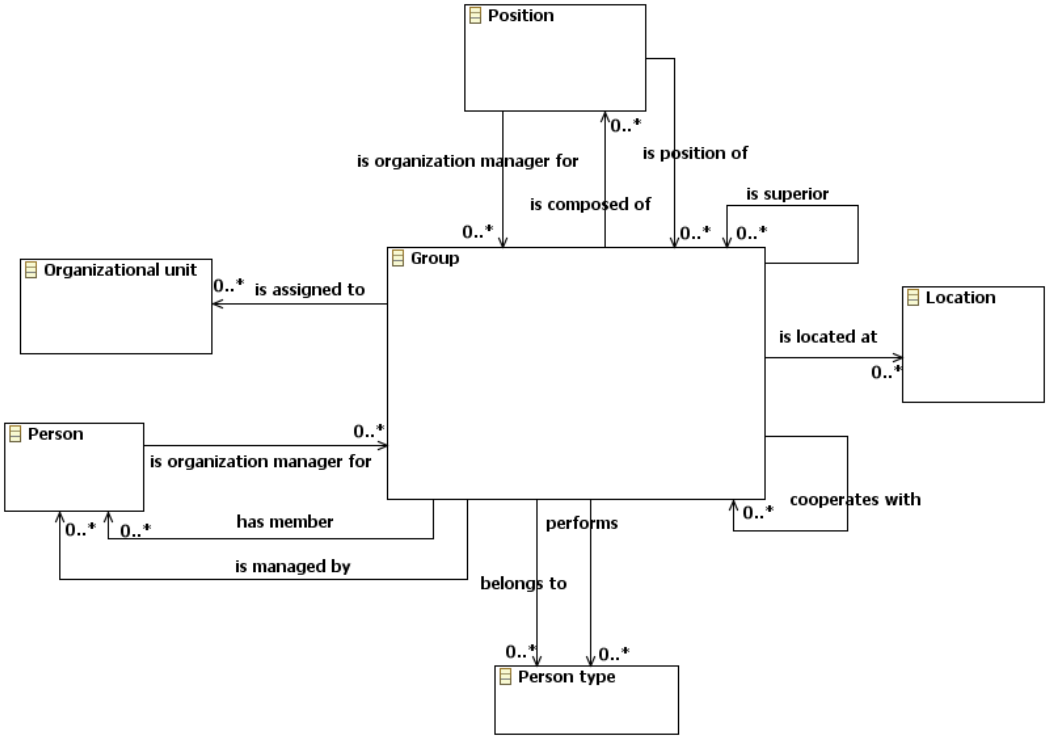


Figura 3.13 – Fragmento do metamodelo organizacional com foco no elemento Group e em seus relacionamentos.

O relacionamento *has member* representa que uma pessoa é membro de um grupo. Assim, é possível expressar que Paulo faz parte do grupo de pesquisa NEMO.

O relacionamento *is assigned to* representa que um grupo está associado a uma ou mais unidades organizacionais, ou seja, este relacionamento descreve que de alguma forma o grupo está "ligado" a organização. Este relacionamento também pode ser utilizado para expressar que o grupo é reconhecido pela organização. Por exemplo, através deste relacionamento é possível expressar que uma determinada ONG é reconhecida pelo Brasil ou que o grupo de pesquisa NEMO está associado ao Departamento de Informática da UFES.

O relacionamento *is managed by* que ocorre entre os elementos *Group* e *Person* representa que um ou mais grupos são gerenciados por uma ou mais pessoas. Por exemplo, através deste relacionamento é possível expressar que o grupo de pesquisa NEMO é gerenciado por João Paulo, Giancarlo e Ricardo Falbo.

Por fim, o relacionamento *cooperates with* representa que dois ou mais grupos trabalham juntos para que um ou mais objetivos sejam alcançados. Esse objetivo pode ser comum a todos os grupos ou pertencente a apenas um dos grupos. Por exemplo, é possível expressar que o grupo X trabalha com o grupo Y para realizar uma determinada tarefa no processo de negócio da empresa.

Infelizmente a documentação on-line do ARIS Toolset não fornece de forma clara a semântica dos elementos *System Organization Unit* e *System Organization*, assim, não conseguimos extrair a semântica destes elementos. A Figura 3.14 apresenta um fragmento do metamodelo organizacional com foco nos elementos System Organization Unit e System Organization Unit Type.

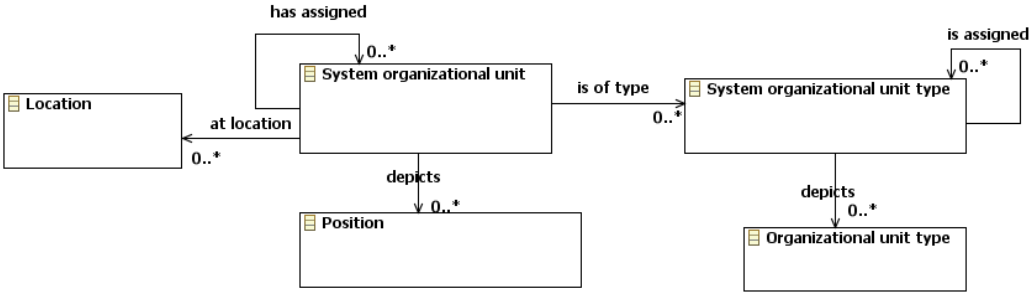


Figura 3.14 – Fragmento do metamodelo organizacional com foco nos elementos System Organization Unit e System Organization Unit Type.

Por fim, a Figura 3.15 apresenta um exemplo de modelo organizacional construído utilizando as metaclasses do metamodelo organizacional do ARIS Method. Como pode ser observado na Figura 3.15, o elemento Vendas é uma instância da metaclasse *Organization Unit*, os elementos Gerente e Vendedor são instâncias da metaclasse *Position* e as conexões entres os elementos do modelo são instâncias do meta-relacionamento *is composed of*.

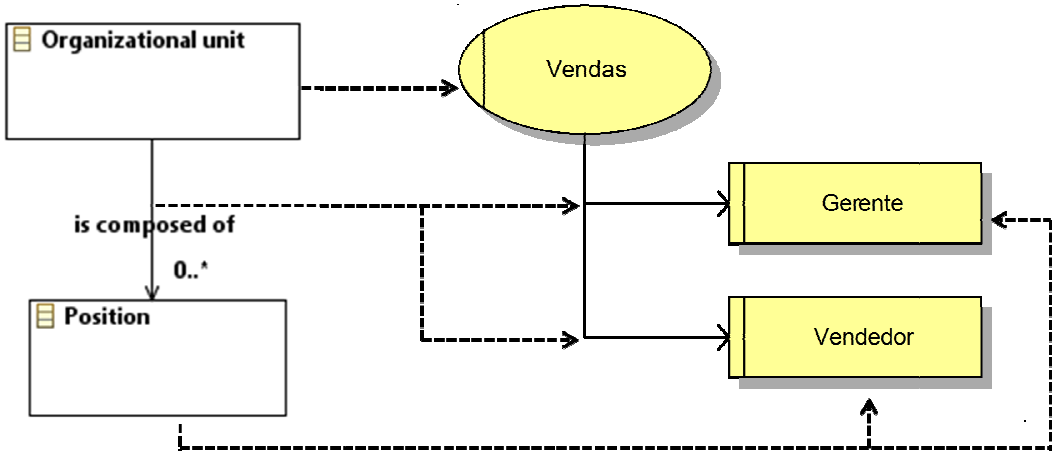


Figura 3.15 – Exemplo de Modelo Organizacional Construído Utilizando as Metaclasses do Metamodelo Organizacional .

3.3 METAMODELO DA LINGUAGEM DE PROCESSO DE NEGÓCIO

O metamodelo da linguagem de processo de negócio é formado pelas seguintes metaclasses *Participant*, *Event*, *Rule*, *Function, Application System, Application System Type, Application System Class* e.*Employee variable*.

A metaclasse *Participant* é uma classe abstrata que representa os elementos os elementos organizacionais *Organization Unit Type*, *Organization Unit*, *Position*, *Person Type*, *Person, Group*. A metaclasse Application é uma classe abstrata que representas os elementos *Application System, Application System Type, Application System Class.*

De acordo com o help do ARIS Toolset, uma *Function* é uma atividade técnica ou atividade realizada sobre um determinado objeto, que possui como finalidade apoiar um ou vários objetivos de negócios da organização. Uma atividade pode ser realizada por uma pessoa ou por um sistema de computador. A Figura 3.16 apresenta um fragmento do metamodelo de processo de negócio com o foco nos relacionamentos do elemento *Function* com o elemento *Participant*.

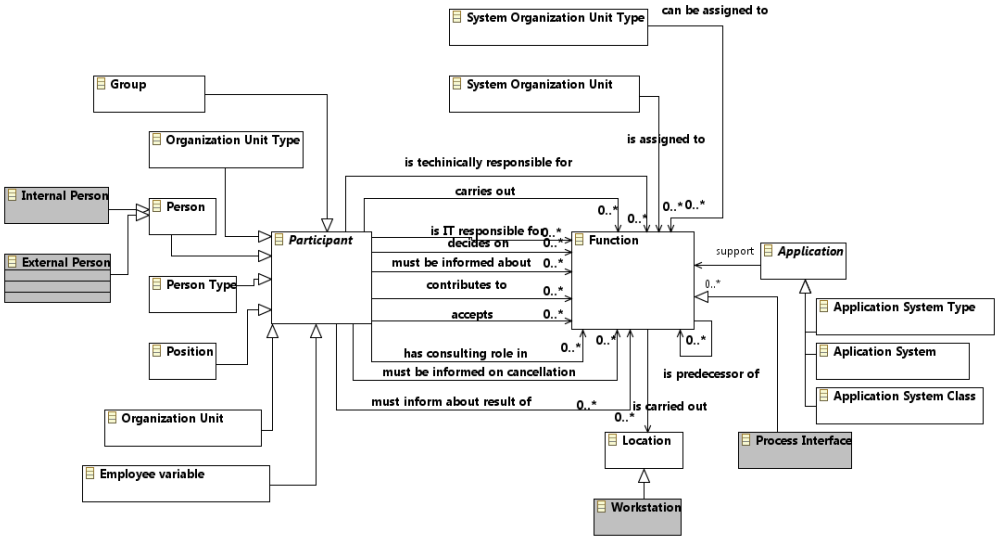


Figura 3.16 - Fragmento do metamodelo de processo com foco no elemento Function e seus relacionamentos com o elemento Participant e com o elemento Application.

As atividades possuem *inputs* (informações ou matéria-prima), e criam *outputs* (informações diferentes ou produtos) e podem consumir recursos(DAVIS, 2001). As atividades de um processo de negócio têm o propósito de atingir um ou mais objetivos (SCHEER, 1999). A Figura 3.17 apresenta o símbolo do elemento *Function*.

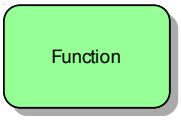


Figura 3.17 - Símbolo do elemento Function.

O elemento notacional *Process Interface*, apresentado na Figura 3.18, do elemento *function* é utilizado para representar o ponto de finalização de um processo e início de outro subseqüente processo de negócio (DAVIS, 2001) (MENDLING *et al.*, 2005a)(MENDLING *et al.*, 2005b). De maneira simplória, este elemento notacional realiza um *link* entre dois processos de negócio.

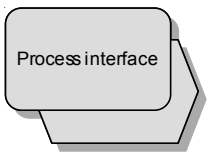


Figura 3.18 – Símbolo do elemento Process Interface.

De acordo com(MENDLING *et al.*, 2005b), existe uma especialização do elemento *function* chamada ***hierarchical function***. Esta especialização é utilizada para

representar que uma atividade do processo de negócio principal é refinada em um sub-processo de negócio do mesmo. O ARIS Method utiliza o símbolo padrão do elemento *function* para representar *hierarchical function*, ou seja, não há elemento notacional para representar que uma atividade do processo de negócio é refinada em um sub-processo de negócio. O relacionamento entre uma ou mais instâncias do elemento *Function* e um ou mais modelos de processo (sub-processos ou macro-processo) é realizado pela conceito denominado *Assignment*. Esse conceito é apresentado com mais detalhes na seção 3.5.

O elemento notacional *System Function (actual)* representa as atividades que são suportadas por aplicativos computacionais ou por computadores. A Figura 3.19 apresenta o símbolo do elemento System Function (actual) (DAVIS, 2001).

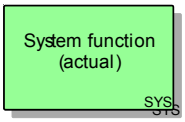


Figura 3.19 – Símbolo do elemento System function (actual).

O relacionamento *carries out* representa que um ou mais participantes do processo de negócio são os responsáveis "legais" pela execução da atividade, ou seja, que um ou mais participantes possuem o compromisso que a atividade seja realizada. É importante ressaltar que um ou mais participantes responsáveis "legais" pela realização da atividade podem ou não ser os executores da mesma. Assim é possível representar que o participante "Analista de processo de negócio" é o responsável pela conclusão da atividade "Modelar processo de negócio".

O relacionamento *contributes to* representa que um *Participant* contribui para que a atividade seja realizada (concluída), sem nenhuma responsabilidade "legal" sobre a conclusão da atividade.

Os relacionamentos *must be informed about*, *must be informed on cancellation* e *must be informed about result of* representam, respectivamente, que um ou mais participantes devem ser informados sobre a execução da atividade; que uma ou mais participantes devem ser informados quando uma atividade for cancelada; e que um ou mais participantes devem ser informados sobre o resultado da atividade.

Infelizmente não foi possível definir uma semântica clara dos seguintes relacionamentos entre *Function* e *Participant*: *is technically responsible for*, *is IT responsible for*, *decides on*, *accepts* e *has consulting role in*. Apenas podemos obter uma notação intuitiva devido ao nome destes relacionamentos.

O elemento *Employee variable* representa um espaço reservado para uma pessoa que será especificada posteriormente, e cujo envolvimento no processo já pode ser identificado.

O elemento *Application System* representa um software de computador que é utilizado para dar suporte à execução de atividade (*Function*) de um ou mais processos de negócios (DAVIS, 2001). A Figura 3.20 apresenta o símbolo do elemento *Application System*.

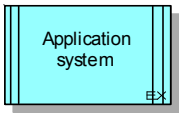


Figura 3.20 – Símbolo do elemento Application System.

O elemento *Application System Type* representa a tipificação do elemento *Application System*, ou seja, ele representa um tipo de aplicações que possuem exatamente as mesmas propriedades tecnológicas. A Figura 3.21 apresenta o símbolo do elemento *Application System Type*.

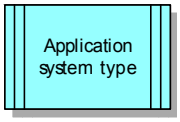


Figura 3.21 – Símbolo do elemento Application System Type.

Conforma apresentado na documentação *on-line* da ferramenta de modelagem, o elemento *Application System Class* representa uma classificação feita aos tipos de aplicações (*Application System Type*) baseados em diversos critérios de classificação. Assim, um *Application System Type* pode estar em diversas classes de aplicações de sistemas (*Application System Class*). A Figura 3.22 apesenta o símbolo do elemento *Application System Class*.

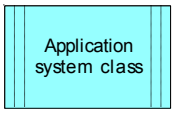


Figura 3.22- Símbolo do elemento Application System Class.

O relacionamento *supports* entre *Application* e *Function* representa que uma atividade é apoiada por uma aplicação computacional, por um tipo de aplicação computacional ou por uma classe de aplicações computacionais.

Um evento (*Event*) representa um estado que é relevante para o gerenciamento do processo e que influência ou controla o fluxo de execução de um ou mais processos de negócio. Mudanças no estado são refletidas na troca do status das informações relevantes do processo. A Figura 3.23 apresenta o símbolo do elemento *Event*.

Figura 3.23 – Símbolo do elemento Event.

Eventos disparam atividades (*Function*) e são resultados de atividades ou são criados por atores externos ao processo(SCHEER, 1999). Um evento tem um aspecto temporal e ocorre em um instante de tempo. A Figura 3.24 apresenta um fragmento do metamodelo de processo de negócio contendo os relacionamentos entre *Function*, *Event* e *Rule*.

Segundo (DAVIS, 2001) os eventos, representam as pré-condições e pós-condições para cada etapa do processo. As pré-condições representam um estado da realidade que ativam uma ou mais atividades. As pós-condições representam um estado da realidade após a execução da atividade. Os eventos podem ocorrer como um resultado do algo que uma pessoa realizou ou como resultado de uma operação de algum software (DAVIS, 2001). Por fim, de acordo com (DAVIS, 2001) um evento final em um processo pode ativar um ou mais processos da organização e, desta forma, criando um link entre dois ou mais processos.

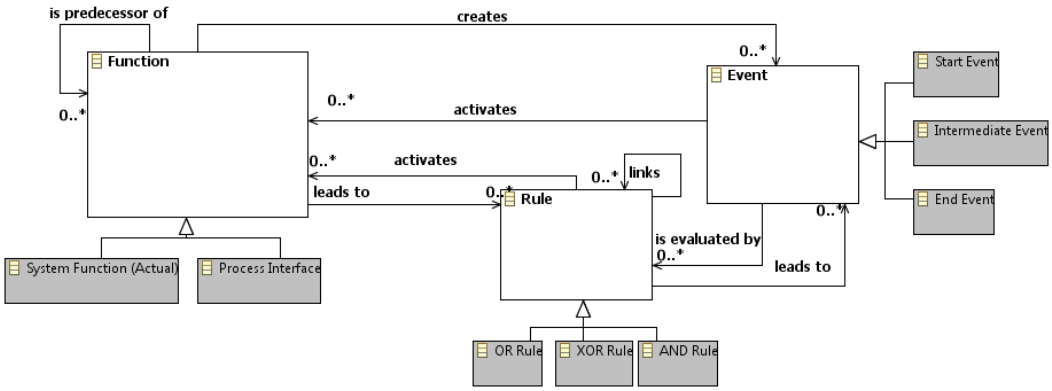


Figura 3.24 – Fragmento do metamodelo de processo contendo os relacionamentos entre Function, Event e Rule.

De acordo com (DAVIS, 2001), as atividades são "ativadas" por um ou mais eventos (*Events*). Na metodologia ARIS um evento ativa uma atividade e uma atividade sempre cria um ou mais novos eventos.

Os relacionamentos *activates* e *creates* entre *Function* e *Event* representam, respectivamente, que uma atividade é ativada por um ou mais eventos e que um ou mais funções podem criar um ou mais eventos.

O relacionamento *is predecessor of* representa que uma atividade é predecessor de outra atividade do processo de negócio. Assim, é possível expressar que uma atividade A acontece antes de uma atividade B.

O elemento *Rule* representa os operadores lógicos que permitem especificar um relacionamento lógico entre eventos (*Event*) e atividade (*Function*) em processo de negócio. As *Rules* são utilizadas para controlar o fluxo do modelo de processo, ou seja, as regras definem o caminho que fluxo do processo deve seguir, tomando como base as informações processadas nas atividades que a precedem (DAVIS, 2001).

As três regras básicas são: *OR*, *XOR* e *AND*. Qualquer regra pode está ligadas, antes ou depois, a atividades de decisão. A Tabela 3.1 apresenta as regras básicas do ARIS.

Tabela 3.1 - Regras básicas do ARIS.

Operador	Símbolo	Antes de uma função	Após uma função
OR		Qualquer evento, ou combinação de eventos, ativa	Um ou mais caminhos serão habilitados com o resultado da

		uma atividade	atividade
XOR		Um, e somente um, evento ativa uma atividade	Um, e somente um, caminho será atividade habilitado com o resultado da atividade
AND		Somente após a execução de todos os eventos a atividade será ativada	Divide o processo em dois ou mais caminhos em paralelo.

O ARIS Toolset apresenta outros elementos notacionais para o elemento Rule, conforme apresentado na tabela abaixo, no entanto, não é apresentada a semântica para os mesmos. Segundo (DAVIS, 2001), os elementos presentes na tabela abaixo "*não possuem semântica definida e em muitos casos tais elementos possuem semânticas ambíguas*". Assim, os elementos notacionais presentes na Tabela 3.2 não serão considerados neste trabalho.

Tabela 3.2 – Outras regras do ARIS.

Elemento	Elemento notacional
Rule	Gateway
	AND/XOR Rule
	XOR/AND Rule
	OR/AND Rule
	XOR/OR Rule
	OR/XOR Rule
	Rule

As atividades (*Function*) são responsáveis por tomar as decisões e as regras (*Rules*) que determinam qual a lógica do fluxo que o processo irá tomar.

O relacionamento *is evaluated by* é utilizado para especificar, em alguns casos até mesmo compor, o *Event* que é necessário para uma determinada etapa do processo de negócio seja realizada. O conector direcionado na Figura 3.25 representa o relacionamento *is evaluated by*.

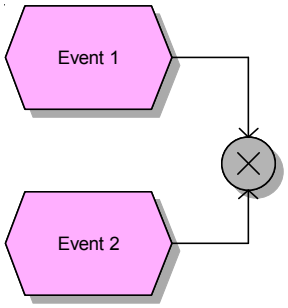


Figura 3.25 – Exemplo o relacionamento is evaluated by entre Event e Rule.

O relacionamento *activates* é responsável por ativar uma ou mais atividades do processo de negócio após o seu pré-estado ser selecionado ou composto, através do relacionamento *is evaluated by,* por algum elemento *Rule*. A Figura 3.26 apresenta um exemplo do relacionamento *activates.*

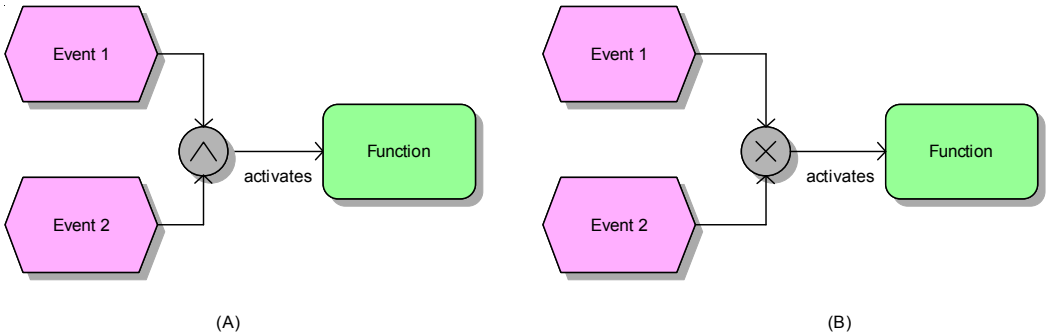


Figura 3.26 – Exemplo do relacionamento activates.

Conforme apresentado no exemplo acima, o elemento *RULE-AND* (Figura 3.26 (A)) é responsável por criar um evento (*Event*) que ativa, através do relacionamento *activates*, uma atividade (*Function*). Na Figura 3.26 (B) o elemento *RULE-XOR* é responsável selecionar um evento (*Event*) que irá ativar uma atividade.

O auto-relacionamento *links* do elemento *Rule* é utilizado para especificar regras de comportais do processo de negócio mais complexas. A Figura 3.27 apresenta um exemplo desse relacionamento.

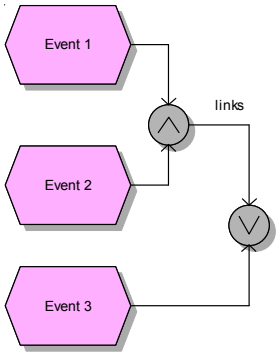


Figura 3.27 – Exemplo do relacionamento links.

No exemplo acima, o relacionamento link é utilizado para especificar uma regra de comportamento que tem como conseqüência a "construção" de um novo evento que será composto pelo Event 1, Event 2 e Event 3 ou apenas pelo Event 1 e Event 2.

O relacionamento *leads to* representa a relação entre a atividade que antecede a regra (*Rule*) e os eventos (*Event*) que são criados pela atividade. A Figura 3.28 apresenta um exemplo da relação leads to entre uma atividade e um *Rule-XOR*.

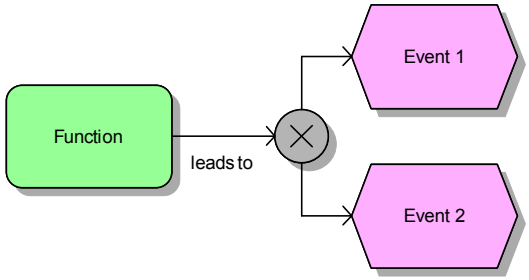


Figura 3.28 – Exemplo do relacionamento leads to.

Por fim, a Figura 3.29 e a Figura 3.30 apresentam um exemplo de modelo de processo de negócio construído utilizando as metaclasses do metamodelo de processo de negócio do ARIS Method. Como pode ser observado na Figura 3.29, os elementos "Cliente" e "Vendedor" são, respectivamente, instâncias das metaclasses Position e Person Type, enquanto os elementos "Solicitar Compra", "Analisar Solicitação de Compra", "Finalizar Comprar" e "Informar Cliente" são instâncias da metaclasse Function. A Figura 3.30 demonstrar que o elemento "XOR" é uma instância da metaclasse "Rule" e os elementos "Necessidade de compra identificada", "Pedido aprovado", "Pedido não aprovado", "Compra finalizada" e "Cliente informado" são instâncias da metaclasse Event.

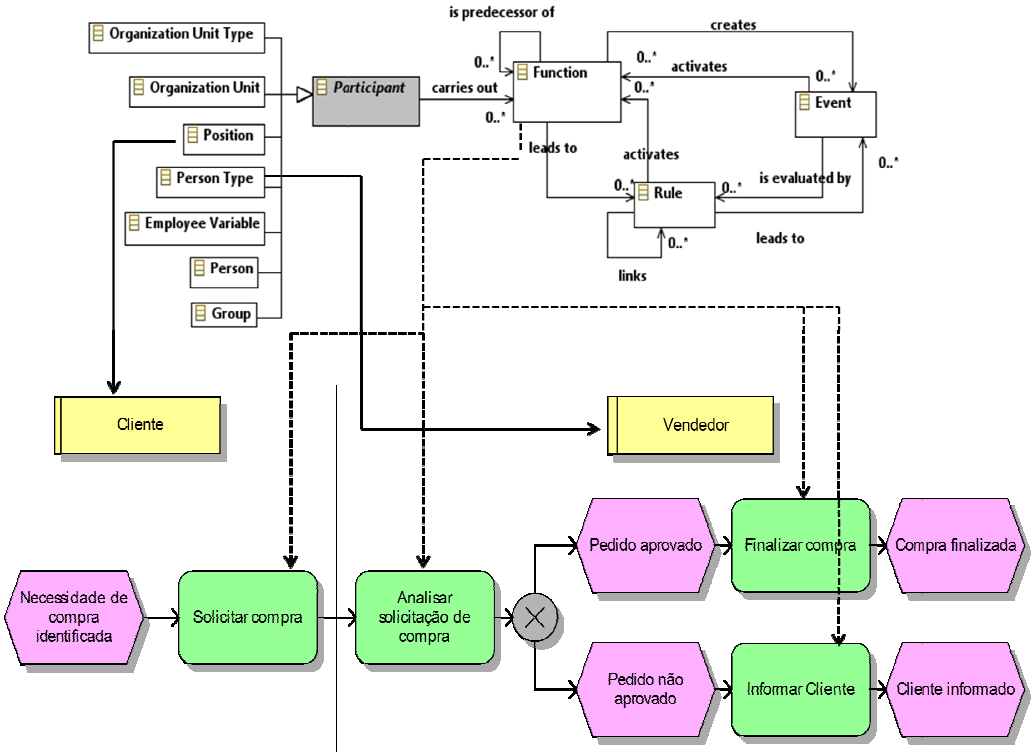


Figura 3.29 – Mapeamento entre as entidades Participant e Function do metamodelo de processo de negócio e em suas instancias em um modelo de processo de negócio.

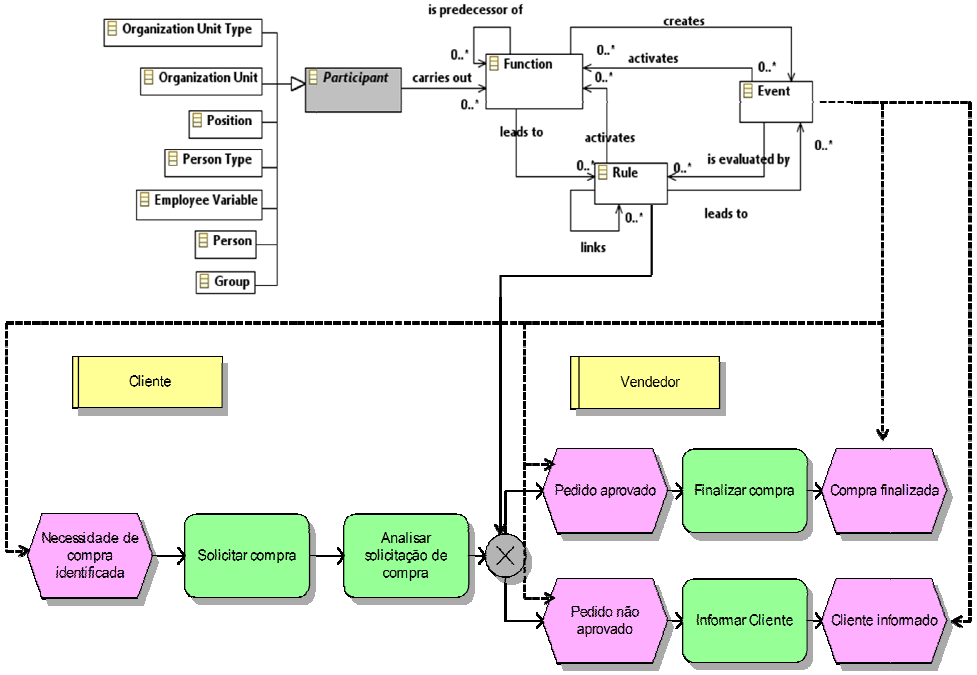


Figura 3.30 - Mapeamento entre as entidades Rule e Event do metamodelo de processo de negócio e suas instâncias em um modelo de processo de negócio.

3.4 METAMODELO DA LINGUAGEM DE DETALHAMENTO DE ATIVIDADES

O metamodelo da linguagem de detalhamento de atividades é composta pelas seguintes metaclasses: *Participation*, *Employee variable*, *Function*, *Information Carrier, Cluster*, *Technical Term, Application* e *Business Rule.*

No metamodelo de detalhamento de atividade os elementos *Participant* e *Function* possuem os mesmos relacionamentos apresentado na seção anterior.

A Figura 3.31 e a Figura 3.32 apresentam fragmentos do metamodelo da linguagem de detalhamento de atividades do ARIS Method contendo os elementos *Function*, *Technical Term* e *Cluster* e seus relacionamentos. Todas as classes em cinza na Figura 3.31 são classes abstratas. A classe abstrata *Information* é utilizada apenas para estruturar o metamodelo e capturar as características gerais dos elementos generalizados.

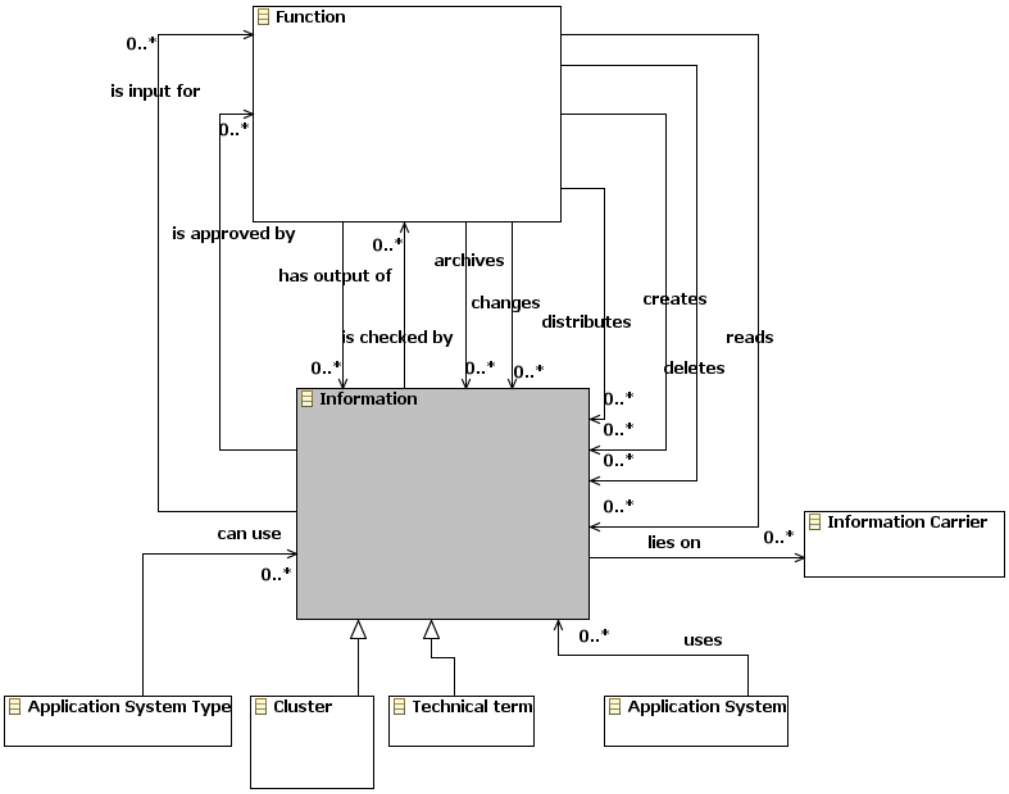


Figura 3.31 - Fragmento do Metamodelo de Detalhamento de Atividade contendo Function, Cluster e Technical Term e seus relacionamentos.

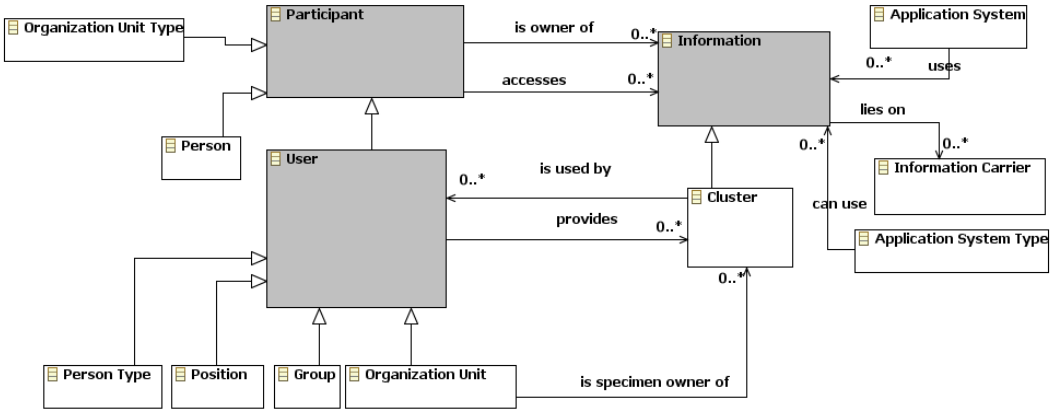


Figura 3.32 - Fragmento do Metamodelo de Detalhamento de Atividade contendo Cluster, Participant, User e seus relacionamentos

O ARIS Method possui uma linguagem de modelagem conceitual de dados chamada de eERM (*Extended Entity-Relationship Model*). Um dos principais elementos desta linguagem é o *Entity Type*, que segundo (DAVIS, 2001), representa as entidades do mundo real (por exemplo: consumidores, empregados e pessoas).

Na linguagem de detalhamento de atividades, o elemento *Cluster* é utilizado para representar uma coleção de *Entity Types* como, por exemplo, pessoas e os seus compromissos (DAVIS, 2001). A Figura 3.33 apresenta o símbolo do elemento *Cluster*.

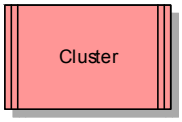


Figura 3.33 – Símbolo do elemento Cluster.

O elemento *Technical Term* representa conceitos que existem na organização e tem por objetivo descrever objetos de informação do negócio da organização. Segundo (DAVIS, 2001), este elemento é usado para modelar informações do ponto de vista da perspectiva dos negócios e o elemento *Cluster* é utilizado para a modelagem das informações do ponto de vista de dados. A Figura 3.34 apresenta o símbolo do elemento *Technical Term*.

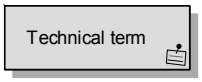


Figura 3.34 – Símbolo do elemento Technical Term.

Os relacionamentos entre o elemento *Function* e o elemento *Information,* apresentados na Figura 3.31, possuem as seguintes semânticas:

i. *has output of*: uma atividade tem como saída uma informação;

ii. *reads*: uma atividade ler uma informação;

iii. *archives*: uma atividade guarda (arquiva) uma informação;

iv. *creates*: uma atividade gera uma informação;

v. *distributes*: uma atividade distribui uma informação;

vi. *deletes*: uma atividade exclui uma informação;

vii. *changes*: uma atividade modifica uma informação;

viii. *is input for*: uma informação é entrada para uma ou mais atividades;

ix. *is checked by*: uma informação é verificada por uma ou mais atividades;

x. *is approved by*: uma informação é aprovada por uma atividade;

Os relacionamentos *is owner of* e *accesses* entre os elementos *Organization Unit*, *Organization Unit Type*, *Position*, *Person*, *Group*, *Location* e *Person Type* com o elemento *Information* representam, respectivamente, que os elementos organizacionais são donos da informação; e que os elementos organizacionais acessam as informações.

O relacionamento *is specimen owner of* entre *Organization Unit* e *Cluster* representa que um ou mais unidades organizacionais são os proprietários de um *Cluster*.

O elemento *Information Carrier* representa um meio de armazenamento de informação. De acordo com (DAVIS, 2001), dados e informações estão sempre armazenados em algum lugar. Por exemplo, as informações da organização estão armazenadas em um arquivo de computador, formulários, arquivos, relatório de negócios ou em entre outros meios que possam armazenar informação. A Figura 3.35 apresenta o símbolo do elemento *Information Carrier*.

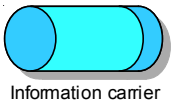
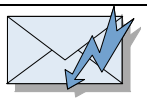
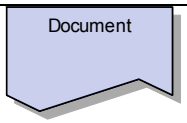
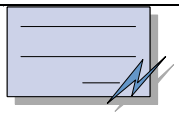
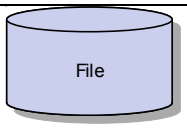


Figura 3.35 – Símbolo do elemento Information Carrier.

A Tabela 3.3 apresenta alguns elementos notacionais do elemento *Information Carrier*.

Tabela 3.3 – Elementos notacionais do Information Carrier.

Elemento notacional	Símbolo
E-mail	
Document	
Eletronic Document	
File	

Os relacionamentos *provides e is used by*. que ocorrem entre os elementos *Person Type*, *Position*, *Group* e *Organization Unit* com os elementos *cluster* e *information Carrier* representam, respectivamente, que tais elementos organizacionais fornecem o *cluster* e os meios de armazenamento; e que os clusters e os meios de armazenamento são usados por tais elementos organizacionais. A Figura 3.36 apresenta um Fragmento do metamodelo de detalhamento de atividade com foco no elemento *Information Carrier*. Na Figura 3.36 somente as classes *User*, *Information* e *Application* são classes abstratas, os demais elementos são elemento notacionais.

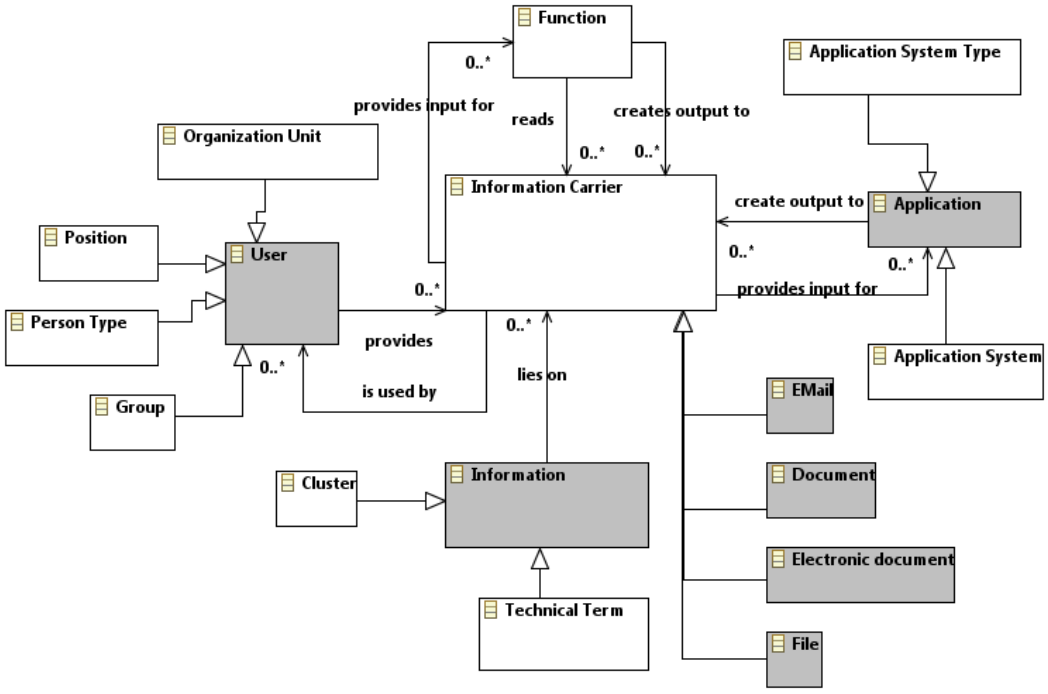


Figura 3.36 – Fragmento do metamodelo de detalhamento de atividade com foco no elemento Information Carrier.

O relacionamento *lies on* que ocorre entre *Information* e *Information Carrier* representa que as informações estão armazenadas em um portador de informação.

O relacionamento *provides input for* entre *Information Carrier* e *Function* representa que um ou mais portadores de informação provêm informações para uma ou mais atividades.

O relacionamento *reads* entre *Information Carrier* e *Function* representa que um ou mais atividades "lêem" informações que estão contidas em uma ou mais portadores de informação.

O relacionamento *creates output to* entre *Information Carrier* e *Function* representa que uma ou mais atividades criam informações para serem armazenadas em um ou mais elementos portadores de informação.

A Figura 3.37 apresenta um Fragmento do metamodelo de detalhamento de atividades com foco nos elementos *Application System*, *Application System Type* e *Application System Class*.As classes em cinza na Figura 3.37 são abstratas.

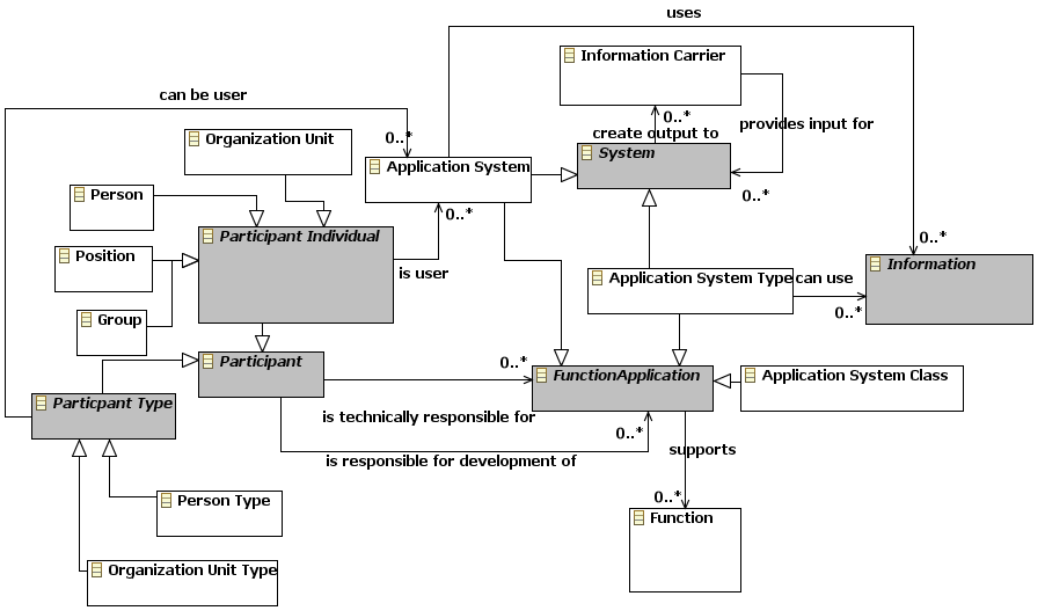


Figura 3.37 – Fragmento do metamodelo de detalhamento de atividades com foco nos elementos Application System, Application System Type e Application System Class.

Os elementos *Organization Unit Type* e *Person Type* possuem o relacionamento chamado *can be user* com o elemento *Application System*. Este relacionamento indica que esses elementos organizacionais podem ser usuários de sistemas computacionais.

Os elementos *Person*, *Position*, *Organization Unit* e *Group* possuem o relacionamento *is user* com o elemento *Application System*. Este relacionamento demonstra que este subconjunto do metamodelo organizacional são usuários de sistemas computacionais.

Os elementos *Organization Unit Type*, *Person Type*, *Person*, *Position*, *Organization Unit* e *Group* possuem os seguintes relacionamentos com os elementos *Application System*, *Application System Type* e *Application Class*:

i. *is responsible for development of*: os elementos do metamodelo organizacional listados acima são responsáveis pelo desenvolvimento de um ou mais *Application System*, *Application System Type* e *Application Class;*

ii. *is technically responsible for:* os elementos do metamodelo organizacional listados acima são tecnicamente responsáveis por um ou mais *Application System*, *Application System Type* e *Application Class;*

O relacionamento *provides input for* entre o elemento *Information Carrier* e os elementos *Application System*, *Application System Type* e *Application Class* representa que um ou mais portadores de informação provêm informações para um ou mais elementos do conjunto de sistema de informação.

O relacionamento *create output to* entre o elemento *Information Carrier* e os elementos *Application System*, *Application System Type* e *Application Class* representa que um ou mais sistemas de informação produzem informações que são armazenados em um ou mais portadores de informação.

O relacionamento *uses* entre *Application System*, *Application System Type* e *Application Class* e Information representa que os sistemas computacionais utilizam as informações contidas nos *Cluster* e nos *Technical Terms* para algum processamento.

O relacionamento *can use* entre *Application System Type* e *Information* é utilizado para especificar quais tipos de informação que um tipo de sistema computacional pode utilizar para o processamento.

Infelizmente a documentação on-line da ferramenta, (SCHEER, 1999) e (DAVIS, 2001) não apresentam a semântica para o elemento *Business Rule*.

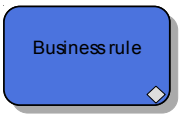


Figura 3.38 – Símbolo do elemento Business Rule.

A Figura 3.39 apresenta um fragmento do metamodelo de detalhamento de atividades com foco no elemento *Business Rule* e *Function*.

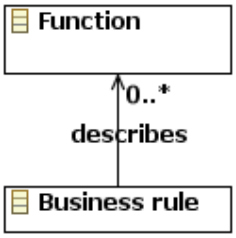


Figura 3.39 – Fragmento do metamodelo de detalhamento de atividades apresentado o relacionamento do elemento Business rule.

O relacionamento *describes* entre *Business Rule* e *Function* representa que uma ou mais atividades são descritas por uma ou mais regras de negócio da organização.

Por fim, a Figura 3.40 apresenta uma instância do Diagrama de Detalhamento de atividades construído com os elementos *Position* (Vendedor), *Function* (Analisar solicitação de compra), *Application System* (Sistema de cadastro de cliente), *Cluster* (Informação da solicitação de compra, Pedido aprovado e Pedido não aprovado) e *Information Carrier* (Solicitação de compra).

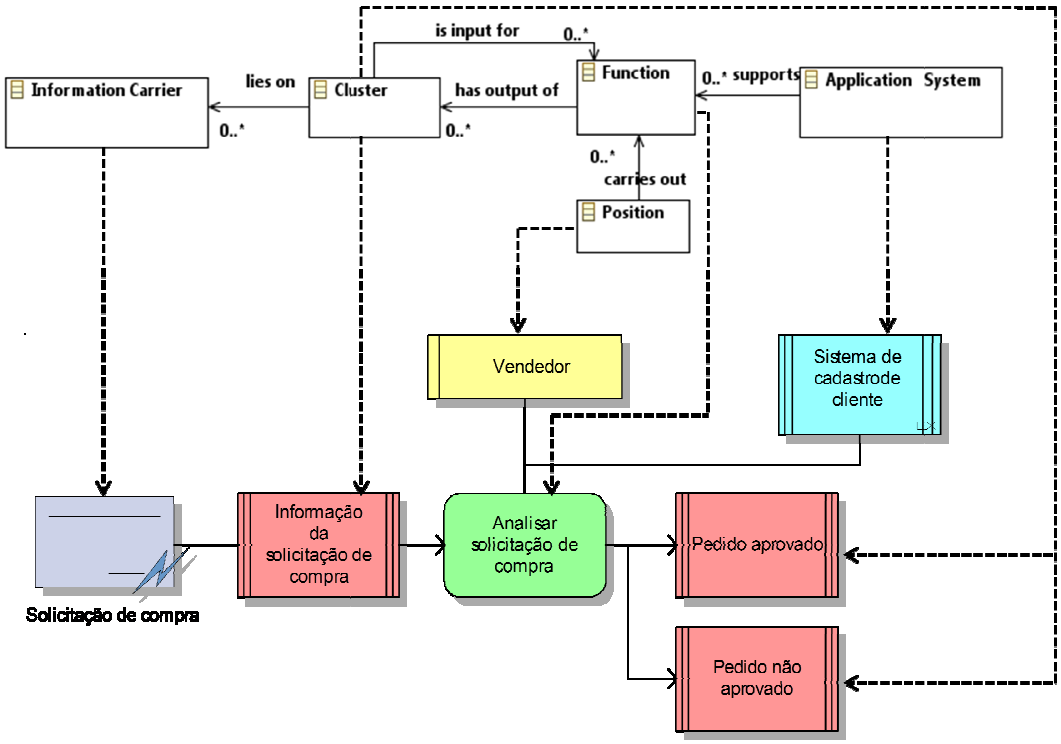


Figura 3.40 - Mapeamento entre as entidades do metamodelo de Detalhamento de Atividades e suas instâncias em um modelo de detalhamento de atividades.

3.5 ARIS OBJECT

Todas as metaclasses de cada metamodelo de linguagem de domínio do ARIS Method, apresentadas nesse capítulo, são extensões da metaclasse abstrata *ARISObject*. A Figura 3.41 apresenta um fragmento do elemento *ARISObject* contendo apenas os atributos *Name* (representando o nome do objeto) e *Description* (representando a descrição do objeto) (PIUMBINI, 2009).

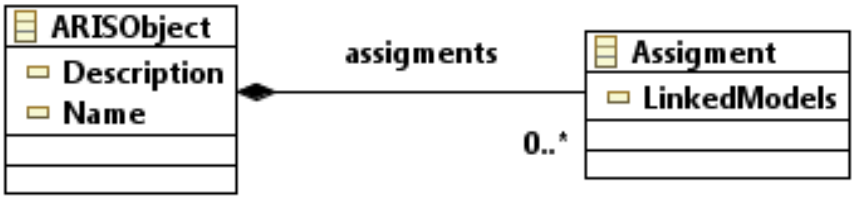


Figura 3.41 – Fragmento do elemento ARIS Object e o Elemento Assignments.

O ARIS permite que um elemento de modelagem esteja relacionado a um ou mais modelos. Por exemplo, é possível associar a um mesmo elemento *Function* um modelo de processo de negócio (EPC) e um modelo de detalhamento de atividades (FAD). Desta forma, o ARIS Method permite descrever os diversos níveis de abstração e refinamento entre os modelos de processo.

Para representar o relacionamento entre um elemento de modelagem e um modelo foi criada a metaclasse *Assigment* (Figura 3.41). O atributo *linkedModels* dessa metaclasse representa o identificador do modelo associado dentro do ARIS Toolset (PIUMBINI, 2009).

3.6 DISCUSSÃO SOBRE ELEMENTOS NOTACIONAIS NO ARIS METHOD

No ARIS Method os elementos notacionais são utilizados como uma forma de adaptar um elemento de modelagem mudando apenas a sua aparência (sintaxe concreta). Por exemplo, o ARIS Method permite ao modelador mudar a aparência de um elemento do tipo *Organization Unit Type* para *Position Type*. O mecanismo é usado de forma extensa até mesmo em filtros pré-definidos na ferramenta ARIS Toolset. O principal problema da abordagem é que o mecanismo é abusado introduzindo modificações significativas na semântica dos elementos originais. Como ilustrado na Figura 3.42, o elemento *Position Type* é considerado um elemento notacional do elemento *Organizational Unit Type*.

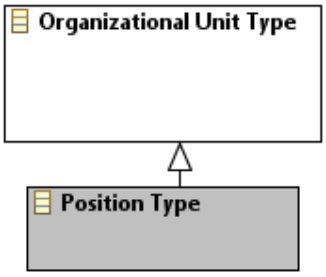


Figura 3.42 – Fragmento do Metamodelo Organizacional.

De forma sucinta, os elementos notacionais no ARIS Method são uma forma que os projetistas das linguagens encontraram de adicionar elementos semânticos nas linguagens sem precisar modificar o metamodelos dessas. No entanto, isso pode acarretar problemas na linguagem como no caso do *Position Type* e *Organizational Unit Type* onde o modelador pode usar entidades com semânticas diferentes em um mesmo objeto.

Para solucionar o problema supracitado é necessário re-engenharia dos metamodelos das linguagens de modelagem de domínio. Tal re-engenharia transformará os elementos notacionais que causam problemas na linguagem em elementos semânticos do metamodelo. Em termos de sintaxe abstrata, consideramos aqui um elemento notacional do ARIS como uma especialização de metaclasses existentes, sem introduzir novos atributos e relacionamentos.

3.7 DISCUSSÃO SOBRE OS METAMODELOS DE DOMÍNIO DO ARIS METHOD

Os metamodelos apresentados neste capítulo retratam a "real" implementação das linguagens de modelagem utilizadas no ARIS Toolset. É importante ressaltar que foi realizada apenas uma refatoração nos metamodelos para torná-los mais legíveis, sem acrescentar elementos na sintaxe abstrata como suportada pela ferramenta. Adicionalmente, neste ponto do trabalho, não foram propostas modificações para possíveis inconsistências e/ou irregularidades na sintaxe abstrata.

Um exemplo deste tipo de inconsistência pode ser observado no metamodelo da linguagem organizacional que considera o elemento *Position Type* como uma especialização de *Organization Unit Type* porém não considera o elemento *Position*

como uma especialização de *Organization Unit*, conforme demonstrado na Figura 3.5 . Além disso, foram preservados elementos com semântica indefinida para uma posterior análise. Por exemplo, foi preservado o relacionamento *performs* entre os elementos *Location* e *Person Type*, o que daria uma impressão que uma localização pode executar um papel organizacional (vide Figura 3.11).

Podemos ainda observar que os metamodelos apresentados neste capítulo são pouco restritivos sobre os elementos de modelagem, permitindo a construção de modelos com semântica indefinida. Por exemplo, o relacionamento *assigments* permite relacionar qualquer elemento com outro elemento ou modelo. No entanto, esse elemento de modelagem não possui semântica definida. Esta característica do metamodelo foi preservada para refletir adequadamente as linguagens do ARIS Method como suportadas pelas ferramentas.

Um dos princípios básicos da abordagem de Desenvolvimento Orientada a Modelos proposta neste trabalho é o conhecimento completo das linguagens de modelagem utilizadas. Logo, será necessário analisar e resolver as inconsistências das linguagens de modelagem do ARIS Method para a construção de um dialeto que permita expressar descrições arquiteturais sem ambigüidade ou inconsistências. Para atingir este objetivo, o Capítulo 4 apresenta uma análise ontológica de cada elemento do metamodelo identificando de forma sistemática as inconsistências das linguagens de modelagem e propondo soluções para os problemas identificados.

3.8 CONCLUSÃO

Os metamodelos apresentados nesse capítulo descrevem as metaclasses e os elementos notacionais das linguagens organizacional, de processo de negócio e de detalhamento de atividades do ARIS Method.

Através dos metamodelos apresentados nesse capítulo torna-se viável a construção de ferramentas de modelagem, simulação e análise de modelos de processos de negócio usando técnicas de desenvolvimento orientado a modelos (do inglês, *Model-Driven Design*).

Torna-se também viável a construção de ferramentas que implementem transformações entre linguagens distintas sem a necessidade de realizar transformações em nível de documento, como é o caso da transformação *XSLT* apresentada em (MENDLINGY e NÜTTGEN, 2004) para a conversão de EPCs do ARIS para EPML (*EPC Markup Language*)(MENDLINGY e NÜTTGEN, 2004). Com o uso de metamodelos, a manipulação de modelos é feita um nível mais alto de abstração, sem recorrer ao nível de documento.

Apesar dos metamodelos de domínio do ARIS Method serem acompanhados de texto para apresentar informalmente a semântica das linguagens, será necessária uma avaliação rigorosa destas semânticas. A avaliação também deve ser capaz de identificar elementos inadequados nas linguagens de modelagem de domínio e propor melhorias na linguagem. Esta avaliação será realizada no Capítulo 4 e será conduzida com base em ontologias de fundamentação.

A avaliação ontológica das linguagens permitirá ainda melhorar o conhecimento dos elementos das linguagens de domínio organizacional e de processo de negócio, facilitando assim a criação de transformações de Desenvolvimento Orientado a Modelos corretas e completas.

CAPÍTULO 4 ANÁLISE ONTOLÓGICADAS LINGUAGENS DO ARIS METHOD

4.1 INTRODUÇÃO

Este capítulo apresenta a análise ontológica dos metamodelos das linguagens organizacional, processo de negócio e detalhamento de atividades do ARIS Method que foram apresentados no Capítulo 3.

Através da análise ontológica será possível definir, de forma, rigorosa a semântica dos elementos dos metamodelos do ARIS Method em termos de entidades do mundo real. Para isto será utilizado as ontologias de fundamentação UFO-A, UFO-B e UFO-C (seção 2.3). Além disto, a análise ontológica permitirá identificar elementos de modelagem inadequados de cada domínio. Esta identificação será feita através de uma série de critérios sistemáticos de avaliação revelando a clareza, expressividade, completude, parcimônia, lucidez, correção e consistência das linguagens de modelagem (GUIZZARDI *et al.*, 2005a) levando à definição de recomendações de melhoria bem fundamentadas.

Para realizar e organizar a análise ontológica das linguagens de domínio organizacional, de processos de negócio e de detalhamento de atividades do ARIS Method foi necessário o desenvolvimento de uma abordagem sistemática, descrita na seção 4.2. Esta seção apresenta os passos utilizados para a análise ontológica e os conceitos utilizados para a mesma.

As seções 4.3, 4.4 e 4.5 apresentam, respectivamente, as análises ontológicas dos metamodelos da linguagem organizacional, de processos de negócios e de detalhamento das atividades. Por fim, a seção 4.6 apresenta as conclusões do capítulo.

4.2 ABORDAGEM DE ANÁLISE ONTOLÓGICA

A abordagem de análise ontológica proposta neste trabalho têm por objetivo: **(i)** interpretar os elementos das linguagens de modelagem em termos de entidades da uma ontologia de fundamentação; **(ii)** identificar possíveis problemas nas linguagem de modelagem; e **(iii)** propor soluções para os problemas das linguagens de modelagem.

A abordagem é aplicada para cada metaclasse e meta-relacionamento presente nos metamodelos de domínio organizacional, de processo de negócio e de detalhamento de atividades do ARIS Method. A Figura 4.1 apresenta a estrutura da abordagem proposta.

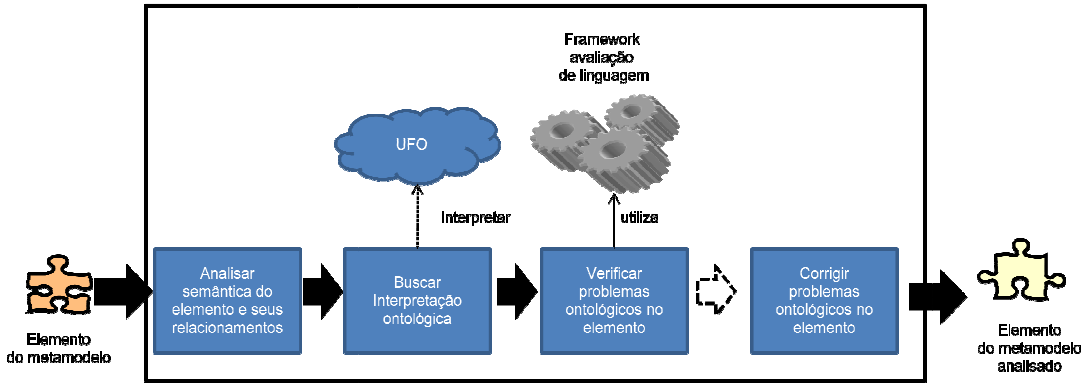


Figura 4.1 – Abordagem da avaliação ontológica.

A abordagem é composta das seguintes etapas:

i. **Analisar a semântica do elemento e seus relacionamentos**: nesta etapa é realizada uma busca na literatura com o objetivo de encontrar a semântica dos elementos e relacionamentos contidos nos metamodelos.

ii. **Buscar interpretação ontológica:** nesta etapa busca-se uma interpretação ontológica para os conceitos representados pelo elemento do metamodelo selecionado e seus relacionamentos. A interpretação ontológica será feita utilizando a semântica das entidades presentes na UFO-A, UFO-B ou UFO-C.

iii. **Verificar problemas ontológicos** no elemento: nesta etapa, verifica-se se o elemento do metamodelo selecionado possui algum problema ontológico que

possa afetar a clareza, a expressividade, a completude, a parcimônia ou a lucidez da linguagem. Para essa etapa, será necessário utilizar um framework de avaliação de linguagens de modelagem como apresentado em (GUIZZARDI *et al.*, 2005a). O framework de avaliação de linguagens irá utilizar as informações da interpretação ontológica (realizada pela segunda etapa) e as informações da semântica do elemento fornecida pelo ARIS (realizada pela primeira etapa) para verificar se há problemas ontológicos no elemento selecionado que afetam a linguagem.

iv. **Corrigir problemas ontológicos no elemento:** caso a terceira etapa encontre algum problema ontológico no elemento selecionado, a quarta etapa busca sugerir modificações na linguagem que corrijam os problemas encontrados.

Ao fim da abordagem o elemento avaliado possui uma interpretação ontológica e, caso exista um problema na linguagem que envolva o elemento analisado, são apresentados os problemas da linguagem e as possíveis soluções para os mesmos.

4.3 ANÁLISE ONTOLÓGICA DO METAMODELO ORGANIZACIONAL

Para realizar a análise ontológica dos elementos do metamodelo organizacional será utilizada a semântica das metaclasses e dos meta-relacionamentos apresentados na seção 3.2 e os conceitos da UFO discutidos na seção 4.3.1.

4.3.1 Base Ontológica para a Análise do Metamodelo Organizacional

Um Institutional Agent, como apresentado na seção 2.3.1.3, representa uma organização, ou seja, um agente que é composto de diversos (outros) agentes, que podem ser Human Agents (seres humanos), Artificial Agents (*softwares*) ou outros Institutional Agents (organizações). Desta forma, um Institutional Agent exemplifica o que é chamado de Functional Complex (GUIZZARDI, 2005a), isto é, uma entidade mereológica complexa, cujas partes desempenham diferentes papéis em relação ao todo.

O relacionamento entre o Institutional Agent e suas partes é um tipo de relacionamento todo-parte especial chamado Component-Functional Complex. Este relacionamento é utilizado para representar que as partes possuem um link funcional com o todo, e desta forma, as partes contribuem para o funcionamento (e/ou comportamento) do todo (GUIZZARDI, 2005a).

A estrutura funcional de composição de um Institutional Agent é definida da seguinte maneira(ALMEIDA *et al.*, 2009):

> *"Vamos considerar X é um Institutional Agent (ou um Institutional Agent Universal) e N seja uma Normative Description associada a X. N define para X um conjunto de funções (functions) que devem ser instanciadas para que X exista, perdure e apresente as propriedades essenciais (incluindo comportamento) associada a X (ou uma instância de X). Estas funções são designadas para um conjunto de* Social Roles *prescritos para existirem em X. Finalmente, um agente z é dito ser uma parte funcional de X (ou instância de X) se, e somente se, z instancia um dos* Social Roles *definidos na* Normative Description *N associada a X .*"

No caso dos Social Function Complexes (como as Institutional Agents**,** por exemplo) a definição do universal instanciado por um Agent é feito via Normative Description. A Figura 4.2 apresenta os relacionamentos entre o Agent e Normative Description.

Como apresentado na seção 2.3.1.3, uma Normative Description pode criar normas que são reconhecidas por uma ou mais entidades sociais. Estas normas podem ser utilizadas para criar novas entidades sociais, controlar o comportamento de entidades sociais e até mesmo para descrever como uma determinada atividade organizacional deve ser desempenhada. De acordo com (BOTTAZI e FERRARIO, 2006), as normas são divididas nos seguintes tipos:

i. Constitutive Norm: norma responsável por criar novos conceitos, papéis e indivíduos sociais na organização. Este tipo de norma também pode descrever, por exemplo, quais são os requisitos para que uma pessoa assuma um papel social ou quais os requisitos que definem qual o papel é superior a outro;

ii. Deontic Norm: norma responsável por regular o comportamento social das entidades, ou seja, este tipo de norma define as permissões, obrigações, direitos e regras de comportamento social dos agentes quando assumem um papel social reconhecido pela organização;

iii. Technical Norms: norma responsável por descrever qual o procedimento correto para que alguma atividade organizacional seja realizada. As Technical Norms tem o propósito de sugerir o comportamento social de certos papéis sociais da organização ao desempenharem alguma atividade na organização.

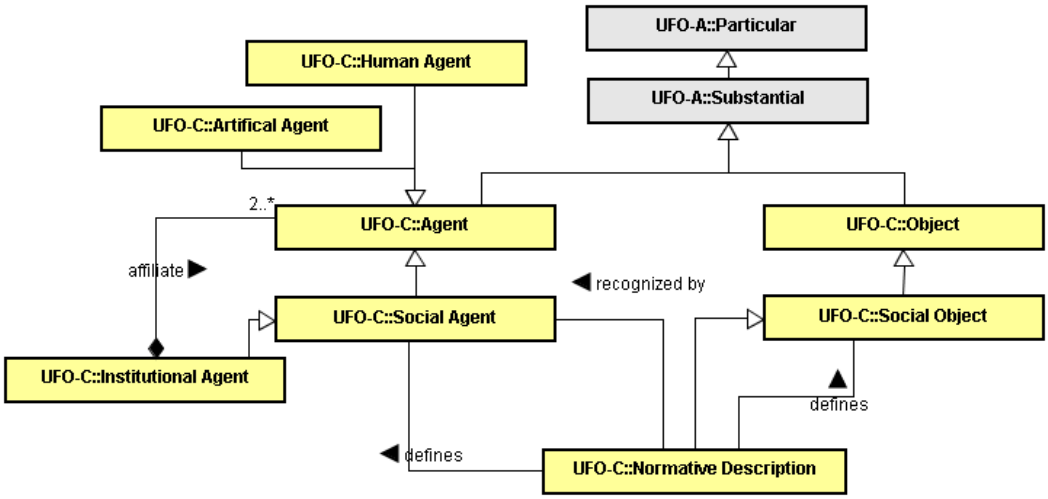


Figura 4.2 – Fragmento da UFO-C contendo a relação entre Social Agent e Normative Description.

O relacionamento is recognized by, na Figura 4.3, é utilizada para representar que as normas são reconhecidas pela organização. Através deste mesmo relacionamento todos os conceitos e papéis sociais criados por uma norma do tipo Constitutive Norm são ditos válidos ou reconhecidos no contexto da organização (BOTTAZI e FERRARIO, 2006).

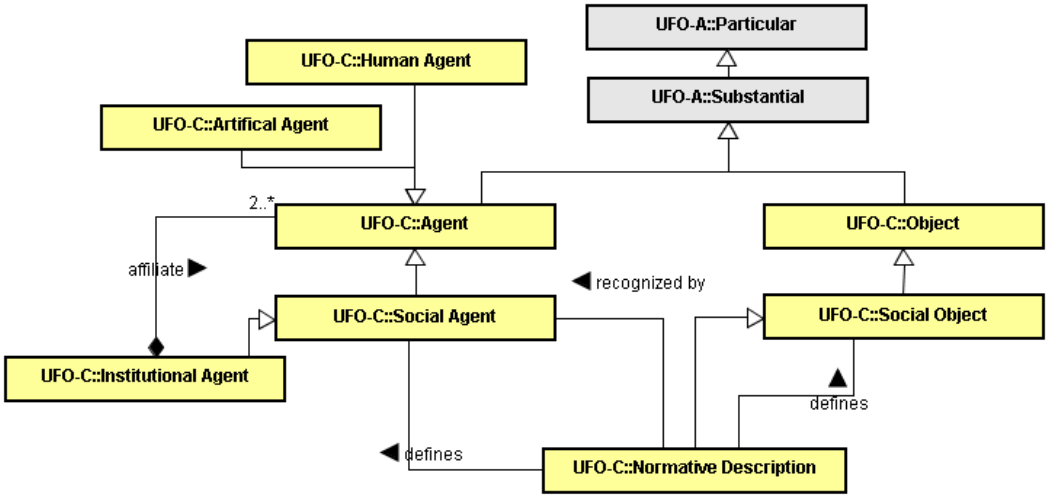


Figura 4.3 – Fragmento da UFO-C com Norm.

Segundo (BOTTAZI e FERRARIO, 2006), um conceito ou um papel social só é reconhecido pela organização quando a norma responsável por criar estes elementos é reconhecida pela organização. O relacionamento is institutionalized by, na Figura 4.4, expressa que um Papel Social é reconhecido pela organização.

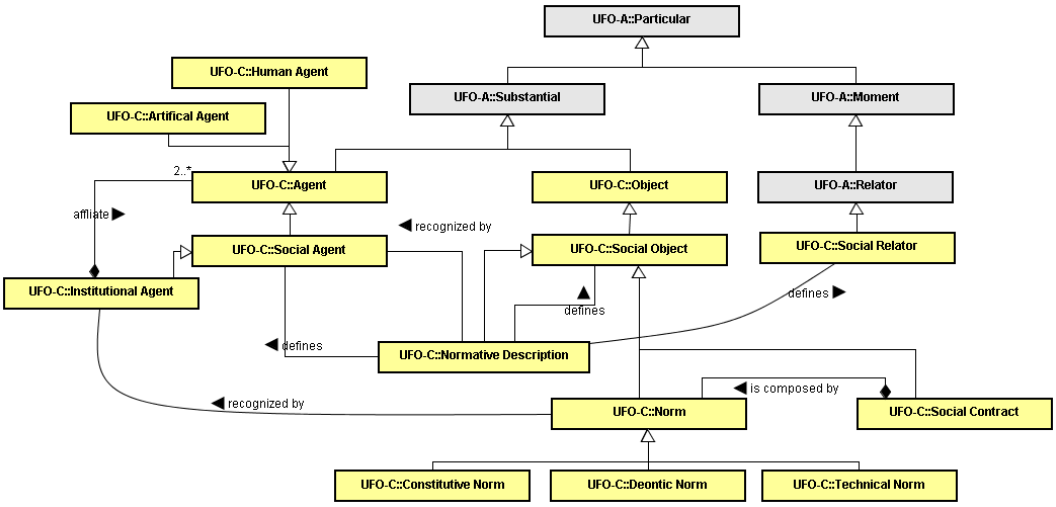


Figura 4.4 – Fragmento da UFO-C contendo o relacionamentos entre Normative Description e Social Role.

Além dos papéis sociais e normas as Normative Descriptions também definem Social Relator Universals. Veja a Figura 4.5. Os papéis são sempre definidos em um contexto ou dentro do escopo de uma relação. Assim, para que Paulo (Particular) instancie um papel social (por exemplo, estudante), ele precisa ser mediado por um Particular Relator (por exemplo, a inscrição na universidade de ensino). O Relator

Universal "Inscrição", neste caso, define as propriedades que todas as instâncias de estudante possuem no contexto deste relacionamento (ou, equivalentemente, quando instancia o papel). Do mesmo modo, para que um Particular z assuma um papel social (Social Role) na estrutura de um Institutional Agent x, o Particular z deve ser mediado por uma instância de um Social Relator Universal definido em uma Normative Description N associada a X.

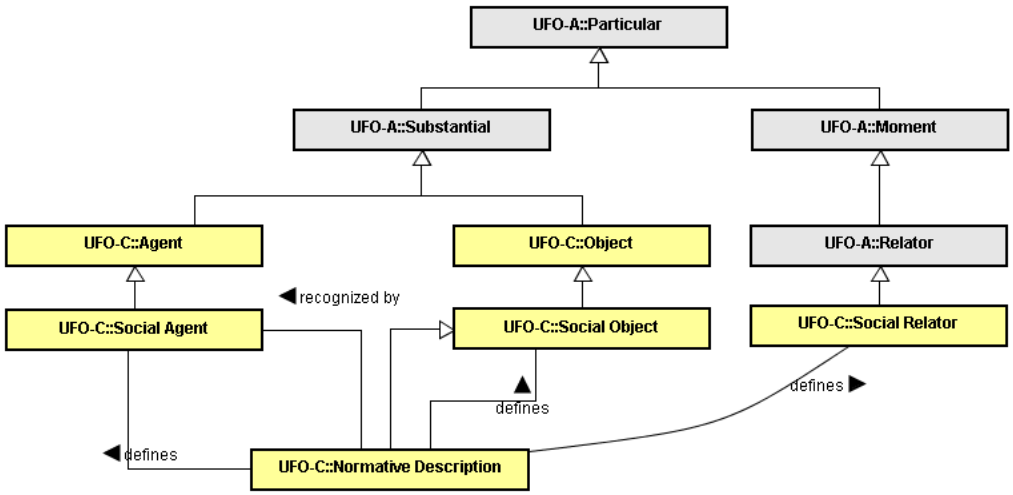


Figura 4.5 – Fragmento da UFO-C contendo o relacionamento entre Normative Description e Social Relator.

Um exemplo de Normative Description responsável por definir Social Role e Social Relator é o Social Contract (Contrato Social) (DIGNUM, 2004). De acordo com (DIGNUM, 2004), um contrato social descreve atividades, regras e obrigações específicas que estão associadas a um papel social da organização como: **(i)** limite de tempo que um determinado membro pode assumir um papel, **(ii)** acordos específicos entre a organização e o membro que assume um papel na organização; **(iii)** condições de governança, ou seja, como a organização pode governar o agente que está querendo assumir um papel social e vice-versa; **(iv)** além de definir sanções ao agente quando as normas são violadas. Logo, o Social Contract é utilizado para explicitar a expectativa de comportamento de um ou mais agentes (Human Agent, por exemplo) ao assumirem um papel social dentro de uma organização (DIGNUM, 2004).

O Social Contract é constituído de Norms. Através destas normas a organização especifica e limita o comportamento social de um agente ao assumir um papel social dentro do contexto organizacional. Os membros da organização podem possuir mais de

um contrato social e mais de um papel social dentro da organização(DIGNUM, 2004). O Social Contract prove um relacionamento (Social Relator) entre os agentes e os papéis sociais presentes na organização. O ato de assinar um Social Contract é um *declarative speech act* que serve como *foundation* para a criação de um Social Relator que liga as partes envolvidas. Quando o agente assina um Social Contract está declarando de forma explícita que aceita todos as condições organizacionais contidas no contrato social.

Um agente é admitido em uma organização através de um processo de socialização(DIGNUM, 2004). Neste processo as partes negociam os termos e condições da participação do agente na organização utilizando Social Contracts. Cada agente irá negociar a sua participação na organização e acordar a duração de sua participação, recursos e como será comunicação entre este agente e os demais agentes já pertencentes a organização, por exemplo.

Por fim, o relacionamento affiliate indica que um agente aceitou as condições descritas no Social Contract, e assim, assumiu um papel social reconhecido pela organização e tornou-se parte da organização. Vale ressaltar, que não são apenas agentes humanos (Human Agents) e outras organizações (Institutional Agent) que podem tornar-se membros de uma organização através do processo de socialização.

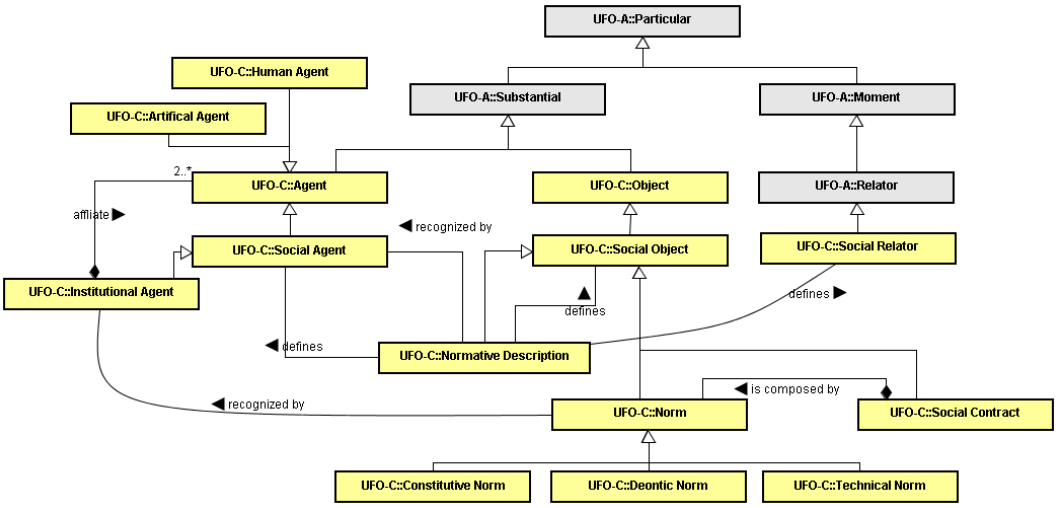


Figura 4.6 – Fragmento da UFO-C contendo Social Contract.

4.3.2 Análise Ontológica dos Elementos Position, Organization Unit, Organization Unit Type

Antes da análise ontológica dos elementos *Position*, *Organization Unit* e *Organization Unit Type*, será necessário interpretar o relacionamento *is of type*, que ocorre entre esses elementos, como demonstrado nas Figura 3.1e Figura 3.5.

O relacionamento *is of type* (tradução: *é do tipo*) é interpretado como o relacionamento chamado instanciação (instantiation) da UFO-A. O relacionamento de instanciação é um relacionamento formal (Relation Formal*)* que ocorre entre um Universal e um Particular e possui a seguinte semântica: "*quando dizemos que p é uma instância de U, estamos querendo representar que p tem a propriedade de ser um U"*(GUIZZARDI, 2005a). Logo, o relacionamento *is of type* é utilizado para expressar que um determinado elemento do metamodelo assume as propriedades de outro elemento do metamodelo. Por exemplo, as instâncias de *Position* e *Organization Unit* irão assumir propriedades de um determinado *Organization Unit Type*.

Analisando a semântica do elemento *Organization Unit,* pode-se interpretá-lo como a entidade ontológica Institutional Agent presente na UFO-C (seções 2.3.1.3), pois ambos representam o conceito social de organização (instituição).

Levando em consideração a interpretação do relacionamento *is of type* entre *Organization Unit* e *Organization Unit Type* (vide Figura 3.1) e a interpretação do elemento *Organization Unit* pode-se interpretar o elemento *Organization Unit Type* como um Institutional Agent Universal.

O elemento *Position* é definido como a menor unidade organizacional. Caso esse conceito seja interpretado literalmente e seja utilizada a interpretação do relacionamento *is of type* entre esse elemento e *Organization Unit Type*, seria possível interpretar o elemento *Position* como um Institutional AgentA Figura 4.7 apresenta as interpretações dos elementos *Organization Unit*, *Organization Unit Type* e *Position* e a interpretação do relacionamento *is of type* em relação a entidades e relacionamentos presentes na UFO-A e UFO-C. O relacionamento pontilhado com o estereótipo <<*represen*t>> indica que um determinado elemento ou relacionamento do ARIS

Method foi interpretado (representa) como uma entidade ou relacionamento da UFO-A, UFO-B ou UFO-C. Vale ressaltar que na parte superior do diagrama temos elementos conceituais da ontologia de fundamentação UFO e que na parte inferior temos elementos linguísticos do ARIS Method.

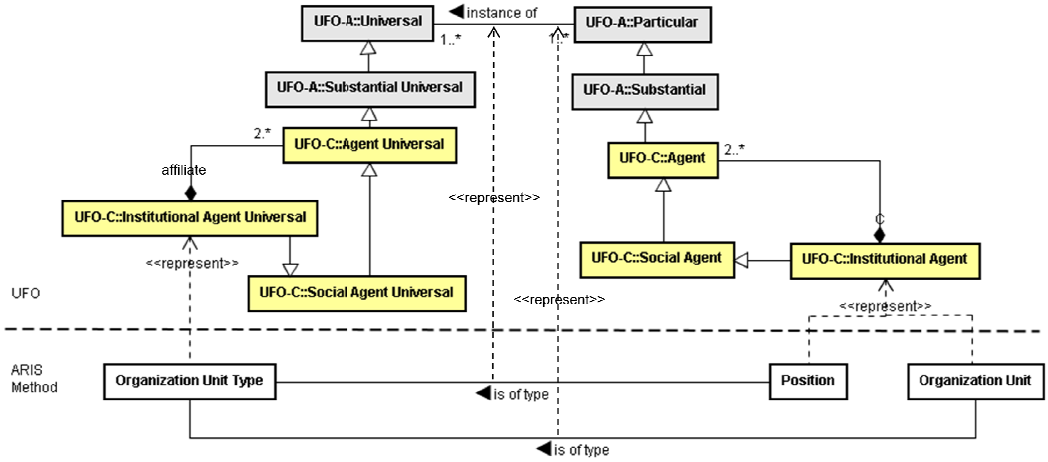


Figura 4.7 - Interpretação dos elementos Organization Unit Type, Organization Unit e Position seguindo os conceitos do ARIS Method.

4.3.2.1 Problemas Ontológicos na Interpretação do Elemento Position

Segundo o ARIS o elemento position é a menor unidade organizacional dentro de uma organização, ou seja, este não pode ser decomposto em outras partes menores. Outra característica deste elemento é que somente o tipo *Person* pode instanciar este elemento. Esta última afirmação pode ser comprovada através do relacionamento *occupies* entre *Position* e *Person*, apresentado na Figura 4.8.

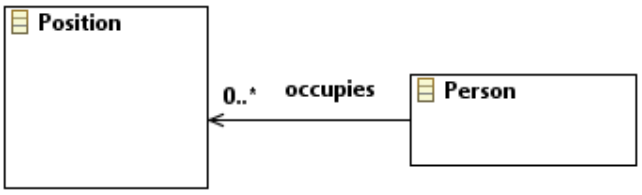


Figura 4.8 - Fragmento do metamodelo organizacional com os elementos Position e Person.

O relacionamento *occupies* possui a mesma interpretação do relacionamento *is of type*, ou seja, o relacionamento *occupies* é interpretado como um relacionamento de instanciação.

Sendo o elemento *Position* indivisível, pode-se considerá-lo como o todo que é composto por somente uma parte. Em outras palavras, o elemento *Position* representa uma unidade organizacional (Institutional Agent) que é composta por somente uma pessoa, caracterizando assim uma quebra do princípio de ***weak supplementation*** (GUIZZARDI, 2005a) e também um problema na interpretação ontológica, afinal um agente institucional é formado por partes.

O princípio de ***weak supplementation*** expressa que o todo não pode ser constituído de apenas uma parte, porque, neste caso, a parte é todo. O todo deve ser constituído de pelo menos duas partes disjuntas. Este conceito está fortemente atrelado ao conceito de agente institucional, como pode ser observado na Figura 4.9, onde é demonstrado que um Institutional Agent é composto de pelo menos dois agentes.

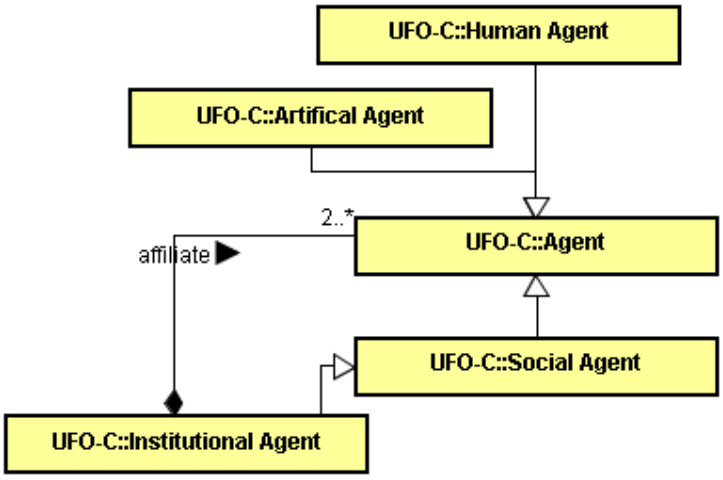


Figura 4.9 - Fragmento da UFO-C contendo Institutional Agent, Human Agent e Artificial Agent (GUIZZARDI et al., 2007).

Devido aos problemas ontológicos na interpretação do elemento *Position* é necessária uma nova interpretação para este elemento, com o objetivo de corrigir tais problemas.

4.3.2.2 Nova Interpretação para o Elemento Position

Propomos que o elemento *Position* seja interpretado com um papel social (Social Role) desempenhado, somente, por uma pessoa (Agente Humano (Human Agent)). Em outras palavras, o elemento *Position* é um Social Role que possui um relacionamento de instanciação (instantiation) somente com agentes do tipo Agente

Humano (como sugerido pelo metamodelo organizacional do ARIS Method através do relacionamento *occupies* entre *Person* e *Position* apresentado na Figura 4.8). A Figura 4.10 apresenta a nova interpretação para o elemento *Position*.

A semântica do relacionamento *occupies* é que uma pessoa pode ocupar (ser associada) a uma ou mais posições em uma organização. Em outras palavras, o relacionamento *occupies* representa que quando uma instância de *Person* assume (ocupa) uma *Position*, uma pessoa assume todos os comportamentos sociais que estão associados ao *Position*. Desta forma, o relacionamento *occupies* é interpretado como um instantiation presente na UFO-A.

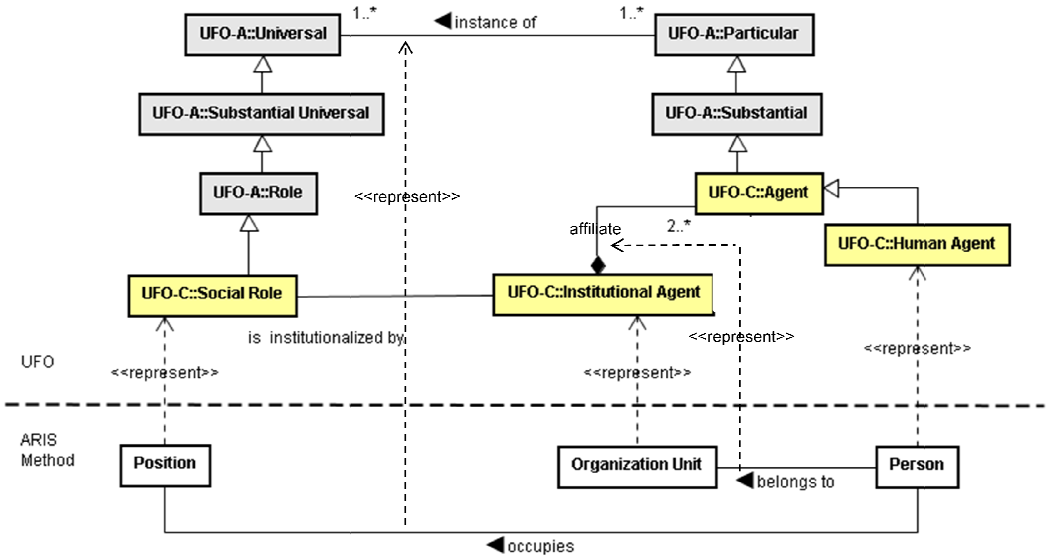


Figura 4.10 - Nova interpretação do elemento Position.

Com a nova interpretação do elemento *Position*, o problema de ***weak supplementation*** foi eliminado, pois um *Position* não é mais interpretado como uma entidade ontológica formada por partes funcionais.

Através da nova interpretação (Social Role) o elemento *Position* adquire propriedades que não existiam na interpretação antiga (Institutional Agent) como: **(i)** Social Moments, que regulam o comportamento social da pessoa que instanciar o *Position*; e **(ii)** existe um contrato social (Social Contract) que estabelece um relacionamento social (Social Relator) entre a instância de *Person* que assume o *Position* (Social

Role) com a organização na qual o Social Role pertence; e entre outras característica associadas ao Social Role.

4.3.3 Análise Ontológica do Position Type

Como pode ser observado na Figura 4.11, o elemento *Position Type* é um elemento notacional, e assim, não possui semântica definida pelo ARIS Method.

Para formular a semântica do elemento *Position Type* serão consideradas as semânticas do elemento *Organization Unit Type* e do elemento *Position*. Assim, pode-se concluir que o elemento *Position Type* é uma tipificação das menores unidades organizacionais (*Position*) que possuem características em comum.

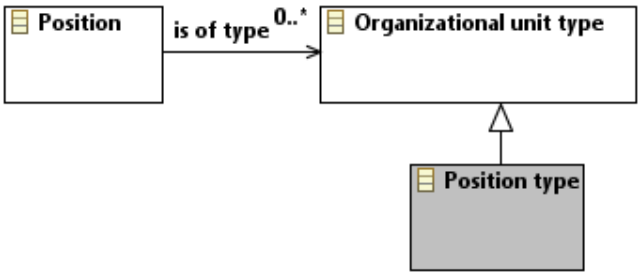


Figura 4.11 - Fragmento do Metamodelo Organizacional contendo somente Position Type, Organization Unit Type e Position

Assumindo o conceito apresentado acima para o elemento *Position Type*, a nova interpretação do elemento *Position* e a interpretação do relacionamento *is of type*, pode-se interpretar o elemento *Position Type* como um dos seguintes elementos ontológicos: **(i)** Social Role, **(ii)** Social Role Mixin ou **(iii)** High Order Universal (HOU).

Ao interpretar o *Position Type* como um Social Role, determinaríamos que o elemento *Position Type* possui a mesma interpretação que o elemento *Position*, ou seja, o elemento *Position Type* é um papel social que é instanciado apenas por agentes humanos. Desta forma, é identificado um problema de **construção em excesso** na linguagem, pois teremos dois elementos distintos com a mesma interpretação ontológica. Para resolver tal problema de acordo com esta interpretação, seria necessário retirar o elemento *Position Type* do metamodelo organizacional.

A segunda interpretação considera o elemento *Position Type* como um Social Role Mixin. Com essa interpretação, o elemento *Position* é uma especialização do elemento *Position Type*, e assim, o elemento *Position* herdaria os Social Moments do *Position Type*. O relacionamento de especialização será feito pelo relacionamento *is of type*. Para exemplificar, imagine o *Position Type* "Cliente" e os *Positions* "Pessoa Física" e "Pessoa Jurídica'. Ambos os *Positions* são especializações de *Position Type*. Cada um dos *Positions* possui os Social Moments do *Position Type*. A Figura 4.12 apresenta a interpretação do elemento *Position Type* como um Social Role Mixin.

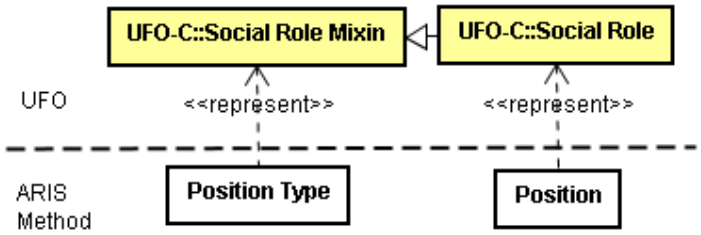


Figura 4.12 - Interpretação do elemento Position Type como um Social Role Mixin.

A terceira, e última, interpretação do elemento *Position Type* considera este elemento como um HOU, ou seja, todas as instâncias desse elemento são Universais, o que torna esta interpretação bastante plausível, pois as instâncias do elemento *Position Type* serão *Position* que também são universais (Social Role). A instanciação será feita pelo relacionamento *is of type* que ocorre de forma indireta no fragmento do metamodelo apresentado na Figura 4.11. A Figura 4.13 apresenta a interpretação do elemento *Position Type* com um HOU.

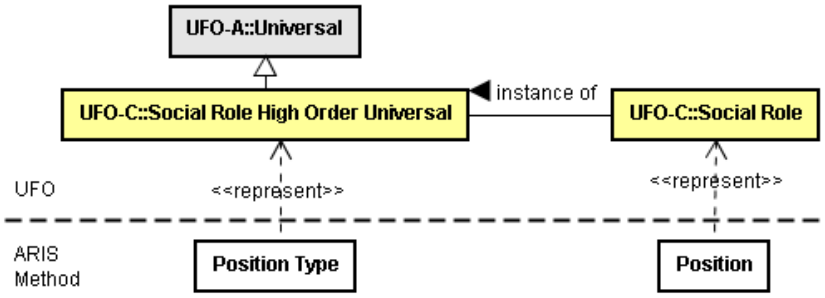


Figura 4.13 - Interpretação do Position Type como um High Order Universal.

Para exemplificar a terceira interpretação,vamos considerar o HOU "Tipo de Docente". Este elemento pode ter uma instância chamada "Docente de ensino superior" (Social

Role) e, este último pode ser instanciado em "Docente do Departamento de Informática da UFES" pelo João Paulo (*Person*), por exemplo.

4.3.4 Interpretação do Person

Segundo a documentação *on-line* da ferramenta de modelagem, o elemento *Person* representa uma pessoa que está associada a uma unidade organizacional. Esta associação expressa que a pessoa é um membro da organização e que assume um position dentro dela reconhecida por ambos. A associação de uma pessoa a uma posição (*Position*) define qual o status dela na organização, as funções e as responsabilidades dentro da organização.

Os conceitos apresentados acima são capturados pelo relacionamento *belongs to* entre *Person* e *Organization Unit* e pelo relacionamento *occupies* entre *Person* e *Position*, no fragmento do metamodelo organizacional apresentado pela Figura 4.14.

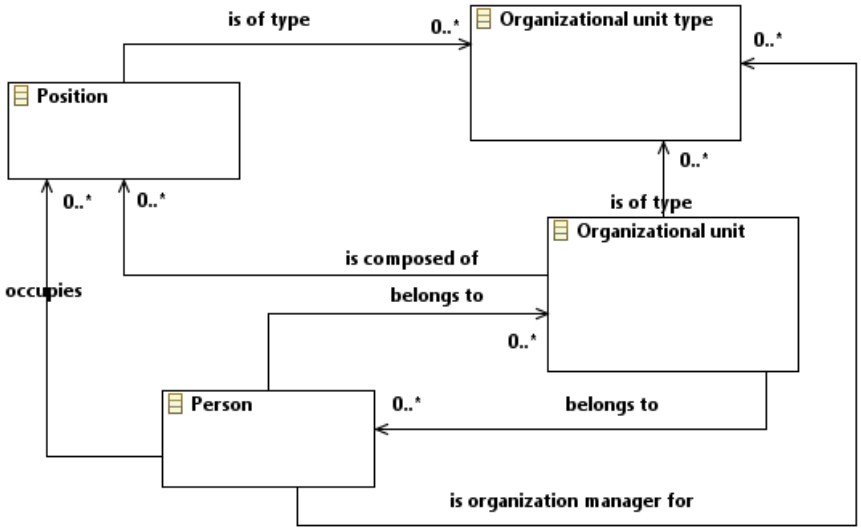


Figura 4.14 - Fragmento do metamodelo organizacional contendo alguns relacionamentos do elemento Person, Position, Organization Unit e Organization Unit Type.

O relacionamento *belongs to* entre *Person* e *Organization Unit* representa que uma pessoa pertence a uma unidade organizacional, ou seja, que uma pessoa faz parte de uma unidade organizacional. Dessa forma, este relacionamento pode ser interpretado como um relacionamento semelhante ao relacionamento affiliate que ocorre entre as entidades Agentes e Agentes Institucionais, como demonstrada na seção 4.3.2.

É dito que uma pessoa faz parte de uma organização quando ela assina um contrato social (Social Contract) com uma organização e assume (occupies) um papel social (Position no caso do ARIS e Social Role na UFO-C) dentro de uma unidade organizacional.

Por fim, analisando a semântica do elemento Person e as interpretações dos elementos *Position*, *Organization Unit* e *Organization Unit Type* é possível interpretar o elemento Person como um Agente Humano (Human Agent). A Figura 4.10 apresenta a interpretação do elemento *Person* e dos relacionamentos *occupies* e *belongs to*.

O relacionamento *occupies* que ocorre entre *Person* e *Position* foi definido e interpretado na seção 4.3.2.2.

Com a nova interpretação do elemento *Position*, o relacionamento *is of type* que ocorre entre este elemento e o elemento *Organization Unit Type* torna-se desnecessário.

O relacionamento *is composed of* entre *Organization Unit* e *Position* representa uma composição funcional entre uma unidade organizacional e uma ou mais *Positions*, ou seja, através deste relacionamento o ARIS representa quais os papéis sociais que são as partes funcionais e essenciais de uma unidade organizacional. Desta forma, este relacionamento será interpretado com o relacionamento todo-parte chamado Componet-Functional Complex. A Figura 4.15 apresenta um exemplo do relacionamento *is composed of*.

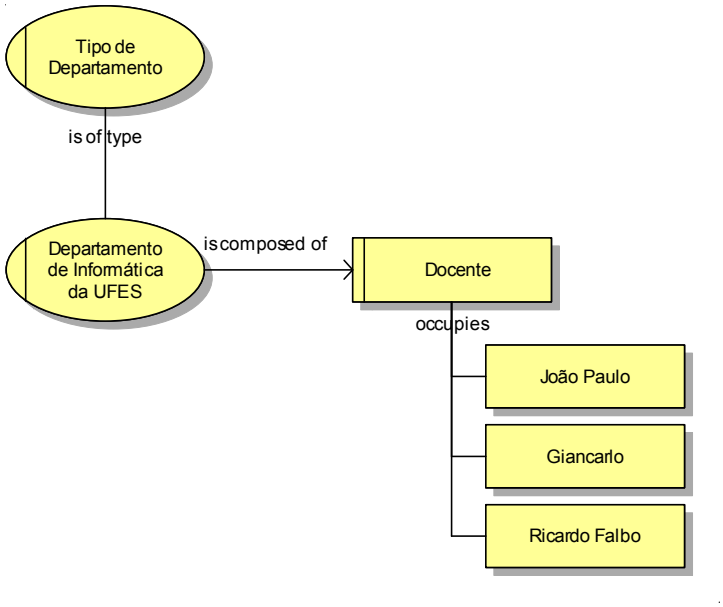


Figura 4.15 - Exemplo do relacionamento *is composed of*.

Na Figura 4.15 o relacionamento *is composed of* é o responsável por estabelecer relação funcional entre uma unidade organizacional (Departamento de Informática da UFES), seus papéis sociais (Docentes) e um "link" entre a unidade organizacional e as pessoas (João Paulo, Giancarlo e Ricardo Falbo) que instanciam os papéis sociais reconhecidos pela organização.

Os relacionamento e auto-relacionamentos *is disciplinary superior to*, *is technical superior to* e *is superior to* representam, respectivamente, que uma determinada entidade é superior disciplinarmente a outra entidade; que uma determinada entidade é superior tecnicamente a outra entidade e; que uma entidade é superior hierarquicamente que a outra entidade. Infelizmente não é possível definir o significado da expressão "*a entidade X é superior tecnicamente ou disciplinarmente do que a entidade Y*", mas é possível definir o que é superioridade hierárquica através da seguinte definição (DIGNUM, 2004):

> *"Um relacionamento de superioridade hierárquica é um relacionamento de coordenação, onde, o elemento denominado superior hierarquicamente pode delegar tarefas, obrigações ou objetivos para um elemento denominado inferior hierárquico (em relação ao elemento*

> *que delegou a tarefa, obrigação ou objetivo) e este executa o que foi lhe pedido*."

Através da definição de (DIGNUM, 2004) pode-se assumir que uma ou mais unidade organizacionais podem ser superiores hierarquicamente a um ou mais *Positions* e vice-versa. Não será feita a distinção entre os relacionamentos *is disciplinary superior to*, *is technical superior to* e *superior to*, ambos, serão considerados relacionamentos de hierarquia definidos em (DIGNUM, 2004).

Os relacionamentos *is disciplinary superior to* e *is technical superior to* e o auto-relacionamento *is superior to* são interpretados como um relacionamento material (Material Relation) da UFO-A, ou seja, existe um relator (Social Relator) que une as entidades envolvidas neste relacionamento e este relator também descreve quais as propriedades sociais (Social Moments) adquiridas pelas entidades ao realizar este relacionamento.

Para exemplificar os relacionamentos de superioridade hierárquica, no contexto organizacional pode existir um objeto social (Social Object) que descreve que a unidade de Recursos Humanos é superior a unidade de Produção ou que um o position Gerente de Recursos Humanos (instância de *Position*) é superior a unidade de Recursos Humanos.

O relacionamento *belongs to* entre *Organization Unit* e *Person* (*Organization Unit* → *belongs to* → *Person*) representa que uma organização pertence a uma pessoa, expressando a idéia de posse ou que uma pessoa é o "responsável legal" pela organização. Em outras palavras, através deste relacionamento o ARIS representa que uma pessoa (*Person*) assume um papel social (Social Role), que é reconhecido no contexto interno e externo à organização, que lhe dá responsabilidades, direitos, deveres, obrigações e outras propriedades (Social Moments) perante o ambiente interno e externo da unidade organizacional. Desta forma, esse relacionamento pode ser interpretado como um relacionamento material (Material Relation), pois é necessário um relator (um contrato social) para associar a unidade organizacional e a pessoa.

Através do relacionamento *belongs to* entre *Organization Unit* e *Person* é possível expressar, por exemplo, que Paulo é o "responsável legal" da empresa X, ou que o Presidente Luiz Inácio Lula da Silva é o "responsável legal" pela República Federativa do Brasil perante o povo brasileiro e as entidades internacionais, conforme apresentado na Figura 4.16.

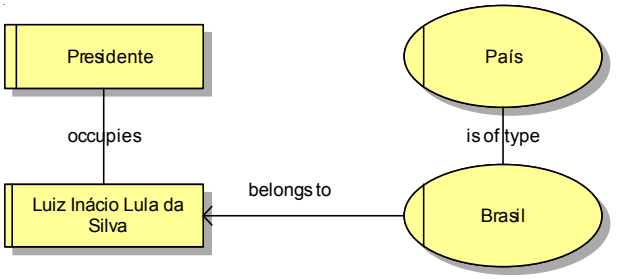


Figura 4.16 - Exemplo de utilização do relacionamento belongs to.

Como pode ser observado na Figura 4.16, quando o Luiz Inácio assume o papel social de Presidente da Republica, ele assume todas as propriedades socais deste papel, sendo que uma delas é ser o responsável legal pela República Federativa do Brasil.

O relacionamento *belongs to* entre *Person* e *Organization Unit* (*Person* → *belongs to* → *Organization Unit*) representa que uma pessoa (*Person*) faz parte de um ou mais unidades organizacionais. Através desse relacionamento, o ARIS Method tenta expressa que uma pessoa assinou um contrato social com a organização e assumiu uma posição (*Position*) dentro de uma ou mais unidades organizacionais. Desta forma, este relacionamento é interpretado com um Relacionamento Material (Material Relation), ou seja, existe um contrato social para unir as partes envolvidas e descrever as propriedades sociais que ambas receberam neste relacionamento.

Através do relacionamento *belongs to* é possível expressar que existe um "link" entre uma pessoa e uma unidade organizacional através do papel que a pessoa assume nessa unidade organizacional.

O relacionamento *is organization manager for* representa que uma entidade é gerenciada por outra entidade. Da mesma forma que os outros relacionamentos, a escavação não conseguiu encontrar o significado deste relacionamento, desta forma, não há interpretação formal para o relacionamento acima. Para permitir a análise,

assumimos a definição de coordenação definida em(DIGNUM, 2004), apresentada anteriormente, que foi utilizada para definir os relacionamentos *is disciplinary superior to*, *is technical superior to* e *superior to*. Logo, o relacionamento *is organization manager for* é interpretado como um relacionamento material, pois é necessário um relator para unir as entidades envolvidas e definir as "propriedades de gerenciamento" das entidades envolvidas.

Através do relacionamento *is organization manager for* é possível expressar, por exemplo, que situações como "Paulo é gerenciado pelo João Paulo", "Os Analista de Processo (*Position*) são gerenciados pelo Escritório de Processo (Organization Unit)" e "O gerente coordena Pedro, Evellin e Ariane". A Figura 4.17apresenta a interpretação do relacionamento *is organization manager*.

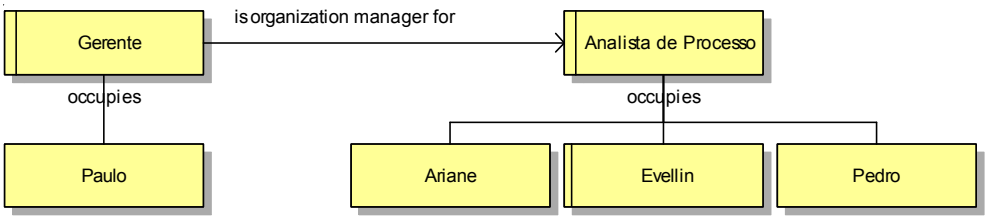


Figura 4.17 - Exemplo do relacionamento *is organization manager for*.

Como apresentado na **Erro! Fonte de referência não encontrada.**Figura 4.17, através do relacionamento *is organization manager for* que ocorrer, neste caso, entre papéis sociais que um relacionamento de superioridade hierárquica. Em outras palavras, existe um Social Moment descrito em uma Normative Description que define que o papel social Gerente é superior hierarquicamente ao papel social Analista de Processo. Sendo assim, todas as pessoas que instanciarem esses papéis irão assumir esse Social Moment.

O relacionamento *substitutes for* representa que uma entidade pode ser substituída por um ou mais entidades, ou seja, através deste relacionamento é possível expressar que Paulo, que assume o papel social *(Position*) de Analista de Sistema, foi substituído por João, por exemplo. Desta forma, concluirmos que o relacionamento *substitutes for* é um relacionamento formal.

O auto-relacionamento *is component of* do elemento *Organization Unit* representa uma composição funcional entre unidades organizacionais, ou seja, através deste relacionamento o ARIS representa quais as unidades organizacionais que são as partes funcionais e essenciais de outra unidade organizacional. Desta forma, esse relacionamento será interpretado com o relacionamento todo-parte chamado Componet-Functional Complex. Assim, este relacionamento permite expressar, por exemplo, que os estados brasileiros compõem a República Federativa do Brasil.

O auto-relacionamento *is responsible for* do elemento *Organization Unit* representa que uma unidade organizacional é a "responsável legal" por um ou mais unidades organizacionais. Em outras palavras, esse relacionamento expressa de forma implícita que uma unidade organizacional assume um papel social de ser responsável por uma ou mais unidades organizacionais da organização. Logo, o relacionamento *is responsible for* é interpretado com um relacionamento material. Através deste relacionamento é possível expressar, por exemplo, que o Departamento de Tecnologia da UFES é o responsável pelo Departamento de informática da UFES e pelo Departamento de Física da UFES.

Os relacionamentos *can be disciplinary superior* e *can be technical superior* do elemento *Organization Unit Type* especifica quais as unidades organizacionais poderão ser consideradas superiores disciplinarmente e superiores tecnicamente. Esses relacionamentos receberão a mesma interpretação dos relacionamentos *is disciplinary superior* e *is technical superior*.

Vale ressaltar que os relacionamentos *can be disciplinary superior* e *can be technical superior estão* no nível de tipo e não no nível de instância como os relacionamentos *is disciplinary superior* e *is technical superior*.

Por fim, o auto-relacionamento *can be constituent* do elemento *Organization Unit Type*, especifica quais os tipos de unidades organizacionais podem compor um outro tipo de unidade organizacional. Assim, esse relacionamento é interpretado como um componet-functional complex no nível de tipo.

4.3.5 Análise Ontológica do Elemento Person Type

O elemento *Person Type* é uma tipificação de um conjunto de pessoas que possuem as mesmas características: responsabilidades, direitos, obrigações entre outras. A relação *is of type* entre *Person* e *Person Type* no metamodelo tende a reforçar esta definição, como demonstrado na Figura 4.18.

Tendo como base que o elemento *Person* foi interpretado como um Human Agent e a existência do relacionamento *is of type* entre este elemento e o elemento *Person Type* é possível interpretar, que este último, como um Human Agent Universal.

Se estendermos a análise ontológica para os demais relacionamentos do *Person Type* (por exemplo, *peforms* e *belongs to*) é possível identificar o problema de **sobrecarga semântica** nesse elemento. Desta forma, afetando a interpretação que o *Person Type* representa um Human Agent Universal e até mesmo a interpretação do elemento *Person*. Para demonstrar o problema de sobrecarga de semântica do elemento *Person Type* será necessário realizar a análise ontológica dos relacionamentos *peforms* e *belongs to* que ocorrem entre o elemento *Person Type* e os alguns elementos do Metamodelo Organizacional.

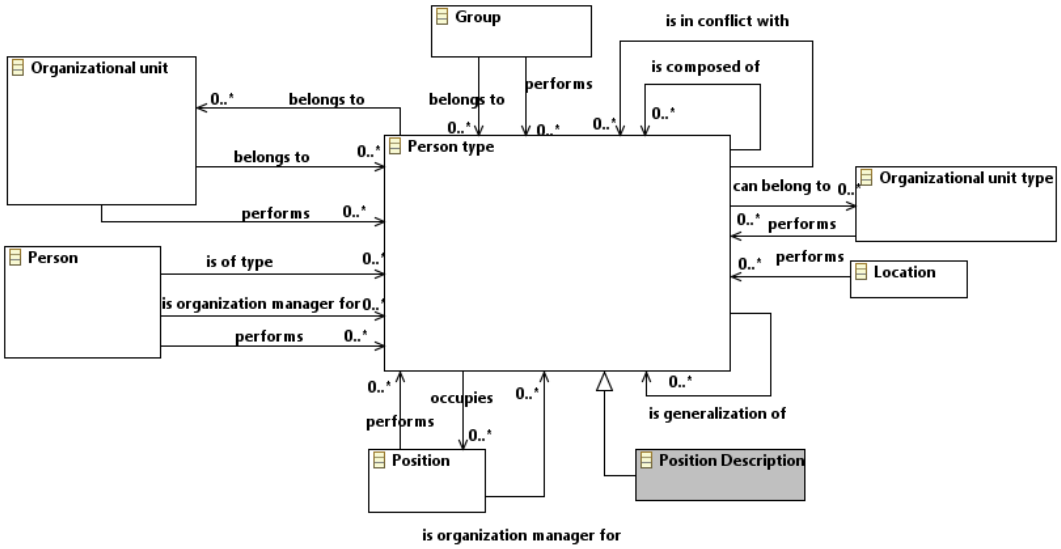


Figura 4.18 – Fragmento do metamodelo contendo os relacionamentos do elemento Person Type.

O relacionamento *performs* representa que uma determinada entidade executa (instância) um *Person Type*. Desta forma, este relacionamento é interpretado com um

relacionamento de instanciação (instantiation) presente na UFO-A. A figura abaixo apresenta um exemplo de modelo organizacional com o relacionamento *perfoms*. Através do relacionamento *performs* o elemento *Position* está assumindo todas as propriedades do elemento *Person Type*.

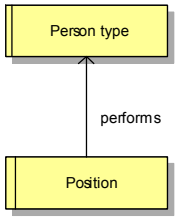


Figura 4.19 – Exemplo de relacionamento performs.

O relacionamento *belongs to* representa que um *Person Type* pertence a uma unidade organizacional (*Organization Unit*) ou a um grupo (*Group*). Desta forma, este relacionamento é interpretado como o relacionamento is institutionalized by/ recognized by (BOTTAZI, 2006) que ocorre entre unidades organizacionais e papéis sociais (Social Role). A Figura 4.20 apresenta um exemplo de relacionamento *belongs* entre um Person Type X e uma unidade organizacional Y, representando que o Person Type X é reconhecido pela unidade organizacional Y e pertence a mesma.

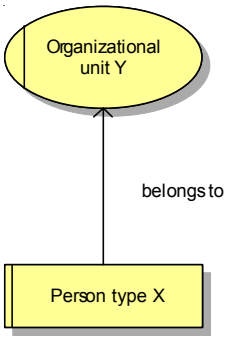


Figura 4.20 – Exemplo de relacionamento belongs to entre Person Type e Organization Unit.

Assumindo somente as interpretações dos relacionamentos *belongs to* e *perfoms* é possível interpretar o elemento *Person Type* com um Social Role Mixin, pois o elemento Person Type possui características de Social Role da UFO-C; e através do relacionamento *performs* o elemento Person Type pode ser instanciado por diversos elementos distintos do metamodelo organizacional (*Organization Unit*, *Organization Unit Type*, *Position*, *Location*, *Person* e *Group*), desta forma caracterizando a sua propriedade de Mixin.

O problema de sobrecarga semântica do elemento *Person Type* pode ser comprovado pois esse possui duas interpretações ontológicas: Human Agent Universal (através do relacionamento *is of type* com o elemento *Person*) e Social Role Mixin (através do relacionamento *perfoms* e *belongs to*).

Infelizmente, o problema de sobrecarga de construtor do elemento *Person Type* não é o único encontrado nesse elemento. Ao analisar o auto-relacionamento *is composed of,* deste elemento, foi encontrado um problema de construtor em excesso.

O problema de construtor em excesso ocorre, porque não foi encontrado nenhuma interpretação para esse relacionamento na UFO-A, na UFO-B, na UFO-C e em outros trabalhos (BOTTAZI e FERRARIO, 2006) (DIGNUM, 2004) que descreva que um Human Agent Universal ou um Social Role Mixin possam ser compostos de outros elementos do mesmo tipo. Desta forma, esse relacionamento explicita um problema de construtor em excesso no elemento *Person Type*.

Para sanar os problemas de sobrecarga semântica e construtor em excesso do elemento *Person Type* é proposta as seguintes soluções: **(i)** interpretar o elemento *Person Type* com um Social Role Mixin; **(ii)** criar um novo elemento no ARIS Method para simbolizar o Human Agent Universal (chamado Person Universal, por exemplo), e assim, "transferir" o relacionamento *is of type* que ocorre entre *Person Type* e *Person* para este novo elemento, ou seja, existiria um relacionamento *Person is of type Person Universal*; e **(iii)** retirar o relacionamento *is composed of* do elemento *Person Type*.

As solução **(i)** e **(ii)** tem por objetivo sanar o problema de sobrecarga semântica e a solução **(iii)** visa resolver o problema de construtor em excesso do elemento *Person Type*. A Figura 4.21 apresenta a interpretação do elemento *Person Type* como um Social Role Mixin.

Com a definição que o elemento *Person Type* representa um Social Role Mixin, é possível realizar a interpretação ontológica dos demais relacionamentos desse elemento.

O relacionamento *can belongs to* entre *Person Type* e *Organization Unit Type* representa uma especificação de quais tipos de papéis sociais podem pertencer a um tipo

de unidade organizacional. Desta forma, esse relacionamento é interpretado com um Componet-Functional Complex no nível de tipo.

A interpretação do relacionamento *is organization manager for* foi apresentada na seção 4.3.3.

O relacionamento *belongs to* entre os elementos *Group* e *Person Type* representa que um ou mais papéis sociais são reconhecidos pela organização como os "responsáveis legais" por um ou mais grupos. Desta forma, esse relacionamento é interpretado como um relacionamento material, pois é necessário um relator (um contrato social) para "colar" a Group a um papel social. Para exemplificar, esse relacionamento possibilita expressar, por exemplo, que o papel social "Coordenador de TI" é responsável pelo grupo que realiza as auditorias internas na empresa X.

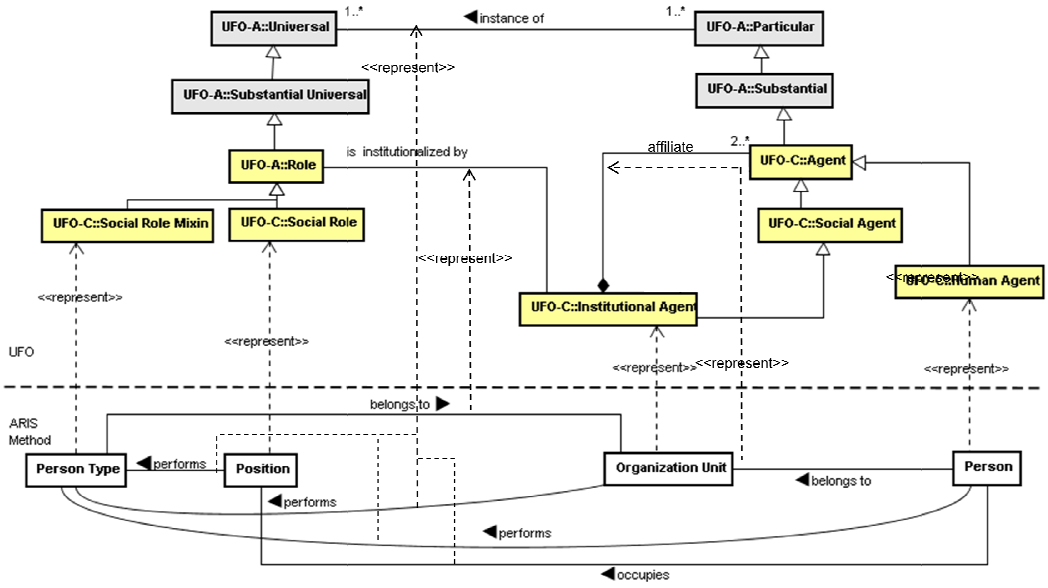


Figura 4.21 – Interpretação do elemento Person Type e dos relacionamentos performs e belongs to.

O relacionamento *occupies* entre *Person* Type e *Position* recebe a mesma interpretação do relacionamento de mesmo nome apresentado na seção 4.3.2. No caso do *Person Type*, o relacionamento *occupies* é utilizado para representar que um determinado *Person Type* pode instanciar um determinado *Position* na organização.

O auto-relacionamento *is generalization of* representa que um determinado papel social possui um conjunto de social moments que está presente em um ou mais papéis sociais

(*Person Type*), ou seja, esse relacionamento indica que um determinado papel social é generalização de outro papel social. Desta forma, o relacionamento *is generalization of* é interpretado como um relacionamento de generalização presente na UFO-A.

Através do auto-relacionamento *is generalization of* é possível expressar que o papel social "Analista financeiro" possui propriedades que são "herdados" pelo papel social "Contador financeiro", assim, o papel social "Analista financeiro" é a "*generalização*" do papel "Contador financeiro". Em outras palavras, todas as instâncias do papel social "Contador financeiro' serão instâncias do papel social "Analista financeiro", e, portanto, as instâncias de "Contador financeiro" exibem as características do papel social "Analista financeiro".

O auto-relacionamento *is composed of* representa que um papel social é composto de um ou mais papéis sociais, ou seja, através desse relacionamento o ARIS Method tenta expressar que um determinado papel social pode ser composto de Social Moments que pertencem a outros papeis sociais da organização.

A UFO não apresenta um relação (ou até mesmo conceitos) que descreva que um papel social é composto de social moments presentes em outros. Porém, é valido descrever que um papel social é composto de social moments de outros papéis social como, por exemplo, o papel social "Analista de Processo" possui características (social moments) que pertencem ao papel social "Analista de Domínio" e "Modelador de Processo", por exemplo. Isto é, o papel social "Analista de processo" é composto de social moments que pertencem aos papéis sociais "Analista de Domínio" e "Modelador de Processo". De forma, singular é possível interpretar o relacionamento *is composed of* com um relacionamento formal (Formal Relator).

Para finalizar, o relacionamento *is in conflict with* descreve que um papel social pode está em conflito com outro papel social da organização, ou seja, que os objetivos de um papel social estão em conflito com os objetivos de outros papéis sociais da organização. Assim, uma mesma entidade (pessoa ou unidade organizacional, por exemplo) não pode desempenhar os dois papéis sociais, com objetivos conflitantes, simultaneamente. O relacionamento *is in conflict with* é um relacionamento formal.

4.3.6 Análise Ontológica do Position Description

Segundo a documentação on-line da ferramenta ARIS Toolset, o elemento notacional *Position Description* é utilizado para descrever as responsabilidades e deveres de um *Position* dentro de uma organização.

Caso seja utilizada apenas a semântica apresentada acima (sem analisar do Metamodelo Organizacional), o elemento *Position Description* é interpretado como um Deontic Norm. Em outras palavras, o elemento *Position Description* representa um documento organizacional utilizado para descrever o comportamento social de um *Position*.

Utilizando a semântica proposta pela documentação on-line, a interpretação do elemento *Person Type* e o fragmento do metamodelo organizacional apresentado na Figura 4.22, é possível concluir que a interpretação que o elemento *Position Description* representa a entidade ontológica Deontic Norm é incorreta. Tal conclusão é possível, porque teríamos no nível ontológico um Social Object (*Position Description*) sendo a especialização de um Social Role Mixin (Person Type) e isso não é válido na UFO, conforme apresentado na (vide Figura 4.2).

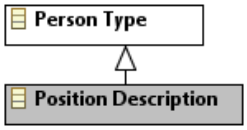


Figura 4.22 – Fragmento do Metamodelo Organizacional com elemento Person Type e Position Description

Como o elemento *Person Type* foi interpretado como um Social Role Mixin, logo, pode-se concluir que o elemento *Position Description* representa, pelo menos, um Social Role ou um Social Role Mixin. Ao analisar novamente a semântica do elemento *Position Description* e levando em consideração a hipótese que este elemento é um Role, concluo-se que o elemento *Position Description* é utilizado para representar papéis sociais (Social Role) que são instanciados por pessoas (*Person*). Assim, quando uma pessoa (*Person*) instanciar o elemento *Position Description* a mesma assume todas as responsabilidades e deveres associado a papel instanciado. A instanciação é feita através do relacionamento *performs*, herdado do elemento *Person Type*. A Figura 4.23 apresenta a interpretação do elemento *Position Description*.

Um exemplo do conceito apresentado no parágrafo acima é que podem existir no nível de processo os papéis sociais Cliente e Cliente Físico representados, respectivamente, pelos elementos *Person Type* e *Position Description*. O Cliente pode ser instanciado por uma pessoa (*Person*) ou uma unidade organização (*Organization Unit*) enquanto o Cliente Físico somente pode ser instanciado por uma pessoa (*Person*).

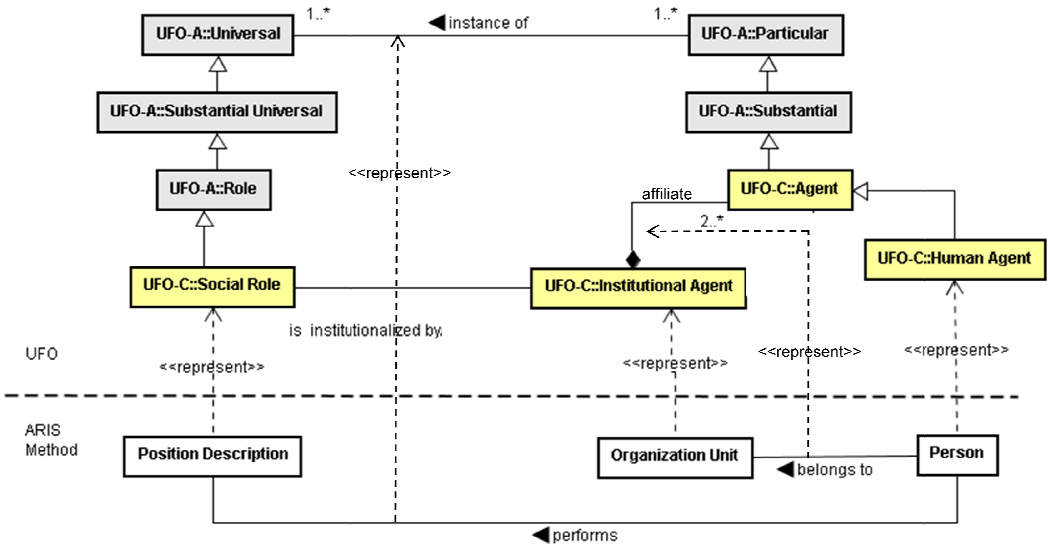


Figura 4.23 - Interpretação do elemento Position Description.

Não há diferenças ontológicas entre elemento *Position* e *Position Description*, ambos representam papéis sociais (Social Role) instanciados por pessoas (Human Agent). A diferença entre estes elemento é no uso. De acordo com Scheer (ARIS, 1999, p. 57), existem papéis sociais que estão associados diretamente ao nível organizacional, outros papéis sociais que estão associados ao nível de processo de negócio. Desta forma, o elemento *Position Description* está ligado somente ao nível de Processo de negócio e não está associado à estrutura organizacional. A seção 4.3.9 apresenta uma discussão mais detalhada sobre esse conceito. Logo, ele pode ser removido da organização e, consequentemente, do processo da negócio sem grandes impactos na estrutura organizacional.

4.3.7 Análise Ontológica do Elemento Group

O elemento *Group* representa um conjunto de empregados (*Person*) que estão trabalhando junto por um período de tempo especifico. No entanto, somente a semântica do elemento *Group* não é suficiente para realizar a análise ontológica desse elemento.

Portanto, será necessário utilizar a semântica dos relacionamentos do elemento *Group* com os demais elementos do Metamodelo Organizacional para realizar a análise ontológica do elemento. A Figura 3.13 apresenta um fragmento do metamodelo organizacional com foco no elemento *Group* e seus relacionamentos.

A primeira interpretação do relacionamento *is position of* é que este é utilizado para indicar que um ou mais posições (*Position*) fazem parte de um grupo. Estas posições podem ser papéis sociais (Social Role) que são reconhecidos pela organização (ou organizações) que o grupo faz parte (conseqüentemente, pelo grupo) ou somente pelo grupo. Esse relacionamento é utilizado no nível de instância. Através deste relacionamento é possível expressar, por exemplo, que o papel social "Pesquisador do NEMO" é um position do grupo NEMO.

O relacionamento *is composed of* pode representa que um ou mais grupos são compostos de positions (*Position*). Da mesma forma que o relacionamento *is position of,* todas as posições associadas ao grupo são reconhecidas pela organização (ou organizações) que o grupo faz parte (conseqüentemente, pelo grupo) ou somente pelo grupo. Porém, este relacionamento é utilizado no nível de tipos, ou seja, é utilizado para especificar quais os tipos de Position compõem um grupo (*Group*). O relacionamento *is composed of* permite expressar que o universal "Grupo de Pesquisa" é composto de papéis sociais do tipo "Pesquisadores"

Logo, os relacionamentos *is position of,* e *is composed* são interpretados como o relacionamento Component-Functional Complex. Sendo o primeiro utilizado no nível de instância (Particular) e o último no nível de tipo (Universal).

Outra interpretação possível para o relacionamento *is position of*, e *is composed* é que ambos sejam utilizados para representar que um determinado *position* que é parte de um grupo, sem realizar a distinção entre nível de tipo e nível de instância. Caso essa interpretação seja levada em consideração é caracterizado um problema de **construção em excesso** na linguagem. Caso contrário, é caracterizado um problema de **sobrecarga semântica**, pois o elemento *Group* está representando ao mesmo tempo um Universal e um Particular.

O relacionamento *has member* representa que uma pessoa é membro de um grupo. Uma pessoa torna-se membro de um grupo através da position que ela ocupa dentro do grupo. Através deste relacionamento é possível expressar, por exemplo, que Paulo Sérgio é membro do grupo de Pesquisa NEMO através do papel social "Pesquisador" que é reconhecido pelo grupo.

Outra interpretação para o relacionamento *has member* é que este seja utilizado para expressar quais os tipos de pessoas podem assumir os papéis sociais reconhecidos pelo grupo. Assim, o relacionamento *has member* é um relacionamento utilizado no nível de instância e no nível de tipo, caracterizando, novamente, um problema de **sobrecarga semântica** no elemento Group.

Em ambas as interpretações do relacionamento *has member* será interpretado como um Material Relation, pois é necessário um Relator para unir as duas partes.

O relacionamento *perfoms* representa que um grupo instancia um determinado papel na organização, ou seja, o grupo como um todo instancia um determinado papel social em algum processo da organização. Por exemplo, um grupo chamado de "Grupo de validação" (composto de diversos agentes e papéis sociais) pode desempenhar o papel social de "Validador" de alguma etapa do processo de negócio da empresa. O relacionamento *perfoms* é interpretado como um instantiation da UFO-A.

O relacionamento *is assigned to* representa que um grupo está associado a uma ou mais unidades organizacionais, ou seja, este relacionamento descreve que de alguma forma o grupo está associado a organização. Este relacionamento também pode ser utilizado para expressar que o grupo é reconhecido pela organização. Por exemplo, através deste relacionamento é possível expressar que uma determinada ONG é reconhecida pelo Brasil ou que o grupo de pesquisa NEMO está associado ao Departamento de Informática da UFES. O reconhecimento do grupo pela organização é feita através de uma Normative Description. Desta forma, interpretamos que este relacionamento é interpretado com um Material Relation.

O relacionamento *is managed by* que ocorre entre os elementos *Group* e *Person* representa que um ou mais grupos são gerenciados por uma ou mais pessoas. Por

exemplo, através deste relacionamento é possível expressar que o grupo de pesquisa NEMO é gerenciado por João Paulo, Giancarlo e Ricardo Falbo. Vale lembrar que este relacionamento representa de forma indireta que o elemento *Person* está associado a um *Position* e este último possui Social Moments que o permite gerenciar o grupo (*Group*). Desta forma, concluímos que o relacionamento *is managed by* é interpretado como um Material Relation, pois existe um Social Relator que une o *Group*, *Position* e *Person*.

O relacionamento *belongs to* que ocorre entre os elementos *Group* e *Person Type* representa que um ou mais papéis sociais são os responsáveis pelo grupo. Assim este relacionamento possui a interpretação semelhante do relacionamento *belongs to* que ocorre entre *Person* e *Organization Unit*.

O relacionamento *cooperates with* representa que dois ou mais grupos trabalham juntos para que um ou mais objetivos sejam alcançados. Esse objetivo pode ser comum a todos os grupos ou pertencente a apenas um dos grupos. Em ambos os casos, os grupos trabalham junto porque sozinho não conseguiriam alcançar os objetivos. Desta forma, existe um social relator (criado por uma Normative Description) que define as propriedades deste relacionamento e como os objetivos serão alcançados. Logo, este relacionamento é interpretado como um relacionamento material. Através do relacionamento *cooperates with* é possível expressar que o grupo de pesquisa NEMO trabalha em cooperação com outro grupo chamado LPRM, por exemplo.

Utilizando as interpretações dos relacionamentos interpretados acima, a semântica apresentada para o elemento *Group* pelo ARIS Method e a definição de grupo dado por (DIGNUM, 2004) o elemento *Group* pode ser interpretado como um Collective Social Agent ou como um Agent Institucional. A diferença de interpretação dependerá do uso do elemento *Group*. A Figura 4.24 apresenta a interpretação do elemento *Group* do ARIS Method em relação aos elementos da UFO-C.

A interpretação do elemento *Group* como um Collective Social Agent ocorre quando a instância do elemento *Group* é utilizado para representar uma coleção (Collection) de Social Agent do mesmo tipo como, por exemplo, um grupo formado apenas por

Human Agent ou um grupo formado por apenas pessoas que assumem o papel de gerente da organização.

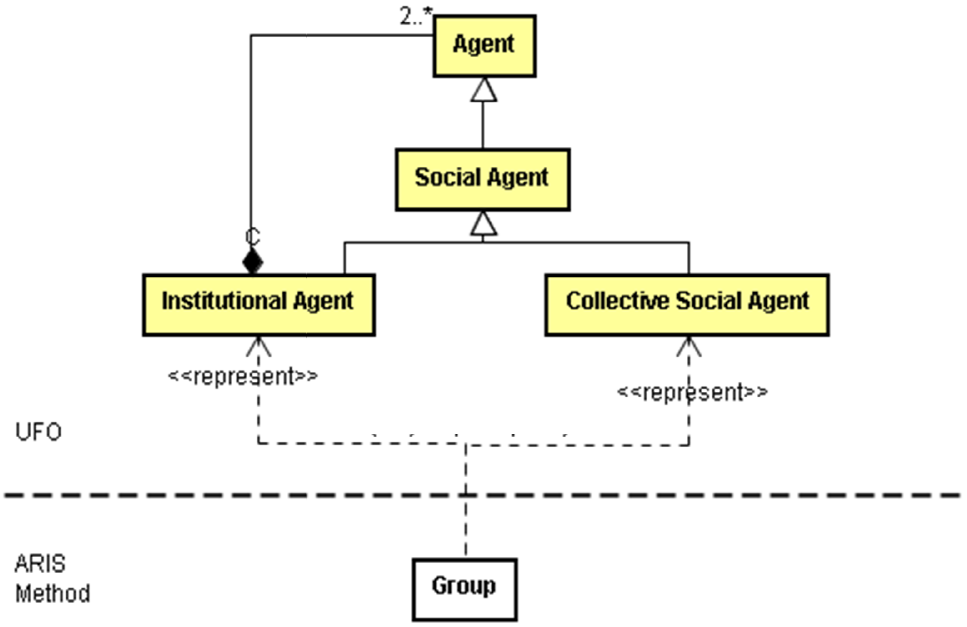


Figura 4.24 - Interpretação do elemento Group.

A interpretação do elemento *Group* como um Institutional Agent ocorre quando a instância do elemento *Group* possui características de Function Complex, ou seja, quando as partes possuem um relacionamento funcional com o grupo. Em outras palavras, todos os integrantes do grupo contribuem para o funcionamento e comportamento do grupo. Quando o elemento *Group* se comporta como um Institutional Agent assume todas as propriedades deste tipo de agente como, por exemplo, o elemento Group pode possuir uma Normative Description, pode ser sub-divida em unidade menores (chamado de sub-grupos) e entre outras características inerentes ao agentes institucionais[3].

Por fim, para solucionar o problema de sobrecarga semântica do elemento *Group* é necessário criar um novo elemento na linguagem que represente o *Group* no nível de tipos (Universal) ou criar um elemento na linguagem que *Group* no nível de instância (Particular).

[3] O ARIS não permite expressar que um grupo é constituído de sub-grupos.

4.3.8 Análise Ontológica do Elemento Location

O elemento *Location* representa uma localização geográfica de uma unidade organizacional, de um papel social (Position e Person Type), etc. Porém, somente essa informação não é suficiente para a análise ontológica do elemento *Location*, será necessário analisar os relacionamentos que esse elemento possui com os demais elementos. A Figura 3.11 apresenta um fragmento do metamodelo organizacional com foco no elemento Location e em seus relacionamentos.

Os relacionamentos *is located at* e *at location* são utilizados para representar em qual região geográfica o elemento organizacional está localizado. Pode-se dizer que é através desse relacionamento que as entidades organizacionais "recebem" o valor da sua localização geográfica.

O auto-relacionamento *encompasses* é utilizado para representar que uma determinada região geográfica pode englobar outras regiões geográficas, por exemplo: podemos representar a localização "Brasil" como o conjunto de todas as posições geográficas dos estados brasileiros.

A partir do conceito apresentado pelo ARIS Method para o elemento *Location* e a semântica dos relacionamentos supracitados é possível interpretar o elemento *Location* como um Quale de um Quality Domain de localizações.

Sendo o elemento *Location* um Quale, o relacionamento *encompasses* é interpretado como o relacionamento member of que fica entre as entidades ontológicas Quale e Quale Structure na UFO-A.

Através do relacionamento *encampasses* o ARIS Method cria uma Quality Structure utilizando os Quales (elemento Location) que pertencem a um Quality Domain de Localizações.

O relacionamento *at location* e o relacionamento *is location at* são interpretados como sendo o relacionamento *quale of* que ocorrer entre as entidades *Quality* e *Quale*. A

Figura 4.25 apresenta a interpretação do elemento *Location* e dos relacionamentos *is located at*, *at location* e *encompasses*.

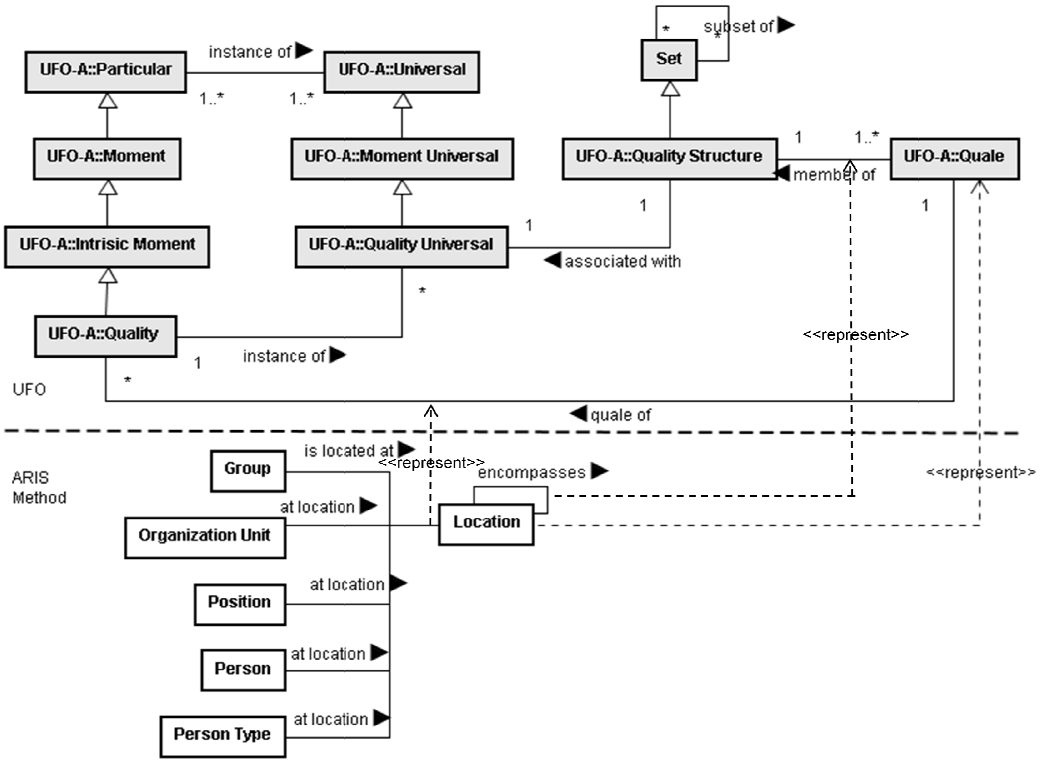


Figura 4.25 - Interpretação do Location e dos relacionamentos at location, is located at e encompasses.

Como apresentado na Figura 3.11 , o elemento *Location* apresenta um relacionamento chamado de *performs* com o elemento *Person Type* e o relacionamento *is organization manager for* com os elementos *Person* e *Position* que podem induzir que o elemento *Location* seja interpretado (erroneamente) como um Institutional Agent. Logo, existe um problema de **sobrecarga semântica**, ou seja, que o elemento *Location* esta sendo utilizado para mais de uma construção ontológica.

O relacionamento *performs* indica que o *Location* instancia um *Person Type*, neste caso, o elemento *Location* não representa uma coordenada geográfica e sim uma unidade organizacional como, por exemplo, uma planta de fábrica pode assumir o papel de responsável da produção de um determinado produto da organização. Desta forma, pode-se concluir o elemento *Location* pode ser interpretado como um Institutional Agent.

Como descrito na seção 4.3, o elemento *Organization Unit* foi interpretado como um Institutional Agent sendo desnecessário que o *Location* também seja interpretado para essa entidade da UFO-C, porque, caso os dois elementos do ARIS sejam interpretados para a mesma entidade da UFO-C teríamos um problema de **construtor em excesso**. Sendo assim, propomos a retirada do relacionamento *performs* entre *Location* e *Person Type* para evitar o problema de construtor em excesso e a não interpretação do elemento *Location* como um Institutional Agent para resolver o problema de sobrecarga semântica.

Por fim, o relacionamento *is organization manager for* indica que um pessoal (*Person*) e uma posição (*Position*) podem ser gerenciados por uma determinada localização. Porém, esta interpretação indica que o elemento *Location* novamente está sendo interpretado como uma unidade organizacional, por exemplo, Paulo *is organization manager for* Vitória, onde Vitória é uma unidade organizacional. Tomando como base esta interpretação, o elemento *Location* novamente é interpretado como um Institutional Agent quando for necessário realizar este relacionamento. Desta forma, encontramos o mesmo problema de sobrecarga de construtor que ocorreu no relacionamento *performs* entre *Location* e *Person Type*. Sendo assim, propomos a retirada do relacionamento *is organization manager for* entre *Person* e *Position* com o elemento *Location* para evitar o problema de sobrecarga de construtor.

4.3.9 Discussão Sobre as Diferenças entre Papéis no Nível Organizacional e no Nível de Processo de Negócio.

A análise ontológica do Metamodelo Organizacional identificou que os elementos *Position*, *Person Type* e *Position Description* são utilizados para representar papéis sociais (Social Role) que são reconhecidos na organização. Do ponto de vista ontológico, retirando as particularidades, tais elementos do ARIS Method são idênticos, ou seja, ambos são interpretados como Social Roles que são instanciados por agentes.

De acordo com Scheer (1999, p. 57), alguns papéis sociais estão associados, somente, no nível de processo de negócio. Alguns papéis sociais, entretanto, são raramente associados ao nível de processo de negócio, pois caso estes papéis deixem de existir será necessário atualizar o processo de negócio.

Este trabalho acredita que o elemento *Position* é utilizado para representar papéis sociais que estão ligados diretamente no nível organizacional e que são raramente alocados no nível de processo. O elemento Position é utilizado para representar postos de trabalhos que as pessoas assumem ao entrar em uma organização.

Por outro lado, os elementos *Person Type* e *Position Description* são papéis sociais que estão associados diretamente ao nível de Processo de Negócio. A eliminação de algum papel social que seja representando por um destes dois elementos não acarreta em impactos graves na estrutura organizacional (SCHEER, 1999).

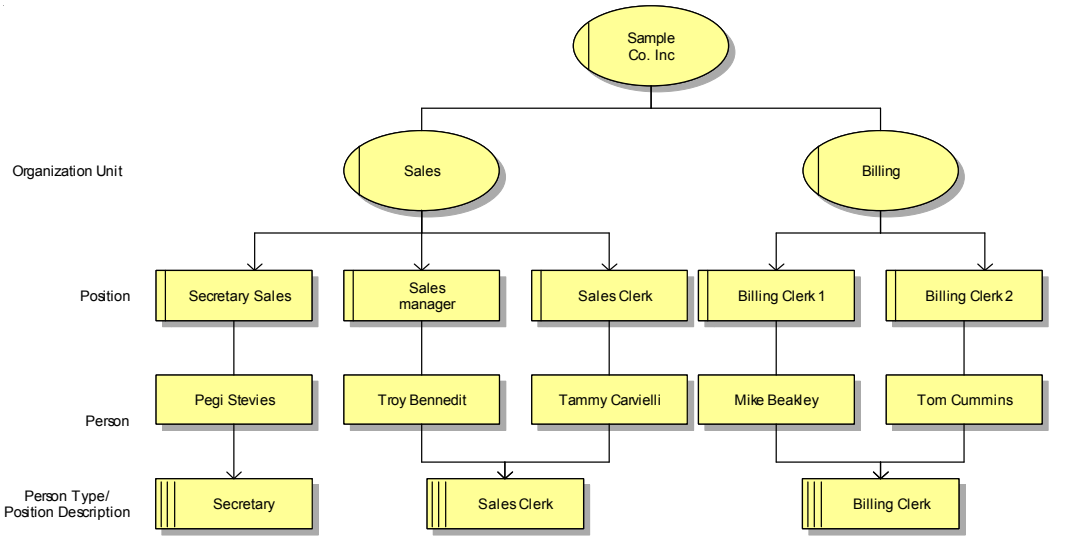


Figura 4.26 – Fragmento de um modelo organizacional apresentado em (SCHEER, 1999, p. 187).

A Figura 4.26 apresenta um fragmento do modelo organizacional retirado de (ARIS, 1999). Neste modelo, é possível observar que o elemento *Position* (*Secretary Sales*, *Sales Manager*, *Sales Clerk, Billing Clerk 1* e *Billing Clerk 2*) é utilizado para representar os papéis sociais que estão associados diretamente a estrutura organizacional, enquanto os elementos *Person Type* e *Position Description* (*Billing Clerk*) estão associados ao nível de processo de negócio.

Pode-se concluir que o elemento *Position* refere-se aos cargos assumidos quando uma pessoa (*Person*) entra na organização (através do processo de socialização) e o elemento *Person Type* são os papéis sociais que a pessoa assume na execução de um ou mais atividades organizacionais que estão presente nos diversos processos de negócio da organização.

4.4 ANÁLISE ONTOLÓGICA DO METAMODELO DA LINGUAGEM DE PROCESSO DE NEGÓCIO

De acordo com (SHARP e MCDERMOTT, 2000) e (OUYANG *et al.*, 2008) um processo de negócio pode ser definido como uma coleção de tarefas de trabalho inter-relacionadas, iniciadas em resposta a um evento, que visão alcançar algum objetivo organizacional. Em outras palavras, um processo de negócio é uma especificação de uma funcionalidade organizacional que deve ser realizada para alcançar um determinado objetivo da organização. Tal funcionalidade pode ser realizada em diversas atividades organizacionais inter-relacionadas.

Um processo de negócio pode ser interpretado Complex Action Universal da UFO-C, ou seja, uma especificação de um conjunto de atividades, que estão sob a responsabilidade de um ou mais agentes.

Para realizar a análise ontológica dos elementos do metamodelo de processo de negócio a etapa "*Analisar semântica do elemento e seus relacionamentos*" tomará como base a semântica desses elementos na seção 3.3. Também será utilizada a interpretação ontológica dos elementos organizacionais, apresentada na seção 4.3.

A seção 4.4.1 apresenta alguns conceitos da UFO-A, UFO-B e UFO-C com maior nível de detalhe que serão utilizados para a interpretação ontológica.

4.4.1 Base Ontológica para a Análise do Metamodelo de Processo de Negócio e do Metamodelo de Detalhamento de Atividades

Para realizar a análise ontológica serão utilizados os conceitos das ontologias de fundamentação UFO-A, UFO-B e UFO-C. A Figura 4.27 apresenta um fragmento com os conceitos da UFO-A, UFO-B e UFO-C inter-relacionados.

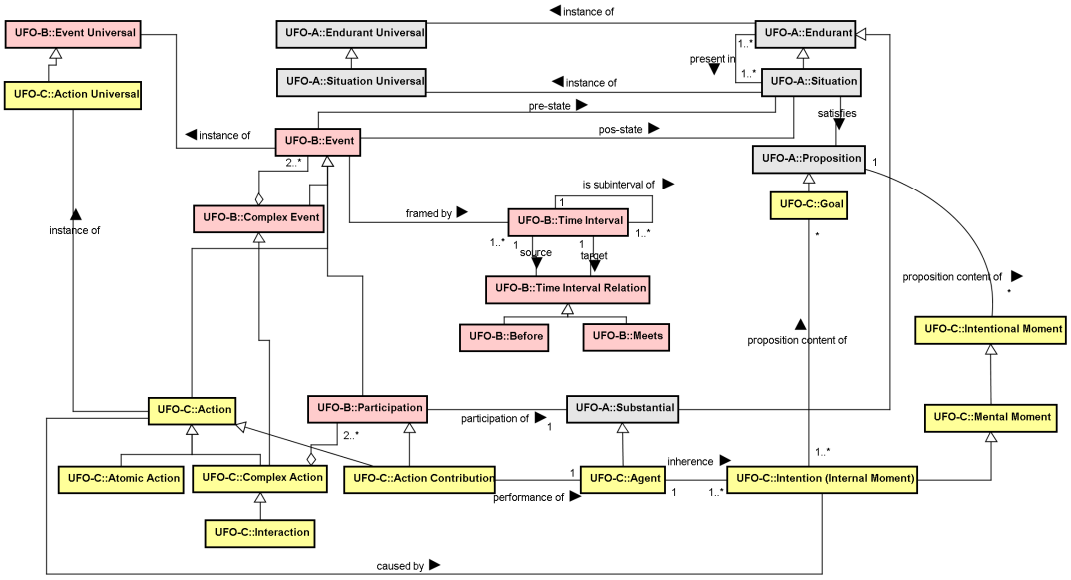


Figura 4.27 – Fragmento da UFO-A, UFO-B e UFO-C.

A UFO-B apresenta o conceito de Events. Os Events acontecem no tempo, no sentido de que eles se estendem no tempo acumulando partes temporais. Um exemplo de um Event é um processo de negócio. Sempre que um Event está presente, não é o caso que todas as suas partes temporais estão presentes.

Eventos (Events) podem ser especializados em dois tipos: **(i)** atômicos (Atomic Event) ou **(ii)** complexos (Complex Event). O primeiro não possui subpartes, enquanto o segundo é composto de agregações de pelo menos dois outros eventos – que por sua vez podem ser atômicos ou complexos. Eventos produzem possíveis transformações de uma porção da realidade para outra, ou seja, eles podem mudar a realidade mudando o estado de uma (*pre-state*) Situation para uma (*post-state*) Situation.

Eventos são entidades ontologicamente dependentes uma vez que eles dependem existencialmente de outros participantes (*Participant*s) para que existam. Um exemplo de Event é a "*Conversa entre médico e paciente*". Neste evento há a participação de dois indivíduos: o médico e o paciente. Cada uma destas participações é por si só um evento, que pode ser atômico ou complexo, mas existencialmente dependente de um individual.

As Situations (situações) são tipos especiais de Endurants**.** Situations são entidades complexas que são constituídas possivelmente por muitos endurants (incluindo outras Situations). Situations são assumidas aqui como sinônimos do que é denominado na literatura de *state of affairs*, isto é, uma parcela da realidade que possa ser compreendida como um todo. Os exemplos de situações incluem "*Paulo possui um carro*" e "*Paula está acessando o computador de Paulo enquanto ele está assistindo televisão*". Um Situation Universal (DOCKHORN COSTA *et al.*, 2006) é usado para definir situações com propriedades em comum.

A UFO-C adiciona distinções relativas à intencionalidade de eventos de UFO-B. UFO-C introduz os conceitos de Action, Action Universal, Action Contribution, Agent e outros. Uma Action é um evento intencional, ou seja, um evento que instância um plano (Action Universal) com o propósito específico de satisfazer algum compromisso interno (Internal Commitment) de um agente. Somente agentes – entidades capazes de ter intentional moments – podem executar ações. Uma Action pode ser composta de outras Actions. Uma Action composta de duas ou mais Actions é chamada de ação complexa (Complex Action), enquanto uma Action que não possui partes é chamada de ação atômica (Atomic Action). Uma ação complexa pode ser composta tanto por outras ações complexas quanto por ações atômicas.

Uma ação complexa é composta de duas ou mais participações (Participation). As participações podem ser eventos intencionais ou eventos não-intencionais. Um exemplo de uma ação complexa é "o ato de escrever este artigo no computador". Esta ação é composta pelas participações intencionais dos autores e uma participação não-intencional do computador. Quando um agente (entidades capazes de ter Intentional Moments) participa de uma ação, esta ação é chamada de Action Contribution. Uma ação complexa que é composta de Action Contribution de diferentes agentes é chamada de Interaction (Interação).

Caso um determinado objeto participe de uma ação ele é chamado de recurso (Resource). Existem diferentes modos de participação de recursos: criação (Creation), finalização (Termination), mudança (Change) e uso (Usage). A seção 2.3.1.3 apresenta a definição de cada tipo de participação de recursos.

Na UFO-B define-se o espaço conceitual de tempo como sendo uma estrutura "composta de" Time Intervals (intervalos de tempo). Um intervalo de tempo, por sua vez, é "composto de" Time points (pontos no tempo). Para representar especificamente uma estrutura de tempo linear e ordenada, tais pontos no tempo podem ser representados como números reais e os intervalos de tempo como sendo um conjunto de números reais. Neste modelo de tempo podem ser utilizadas as chamadas Allen's Relations. As diversas relações possíveis entre eventos a e b são representadas esquematicamente na Figura 2.5, na qual a direção horizontal representa a passagem do tempo.

4.4.2 Análise ontológica do Elemento Function

O elemento *Function* é responsável por descrever atividades que são desempenhadas na organização por um ou mais participantes (pessoas ou aplicativos computacionais), com a finalidade de atingir um ou mais objetivos organizacionais. Isso pode ser comprovado na Figura 4.28 através do relacionamento *carries out* entre *Participant* e *Function*.

Outra característica importante do elemento *Function* é que ele possui vários níveis de abstração e refinamento, ou seja, o elemento *Function* pode representar uma atividade de alto nível de abstração que é refinada em outras atividades de menor nível de abstração (SCHEER, 1999).

Através da semântica do elemento *Function* é possível interpretá-lo como uma **(i)** *Atomic Action Universal*, **(ii)** *Complex Action Universal* ou **(iii)** Interaction Universal.

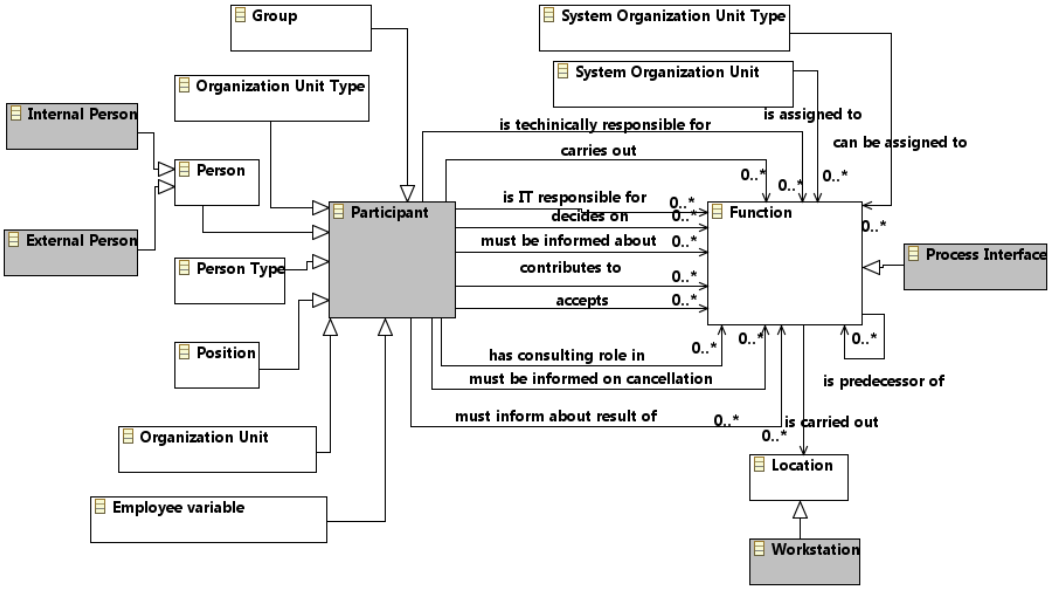


Figura 4.28 – Fragmento do Metamodelo de Processo de Negócio com os relacionamento entre Particpant e Function.

A interpretação do elemento *Function* como um *Atomic Action Universal* ocorre quando o elemento *Function* representa uma ação atômica (não possui refinamentos) e é executada por somente um agente (*Participant*). A interpretação do elemento *Function* como um *Complex Action Universal* ocorre quando o elemento *Function* representa uma ação não-atômica, composta de dois ou mais Actions Universais, e é executada por somente um agente (*Participant*). A interpretação do elemento *Function* como Interaction Universal ocorre quando o elemento *Function* representa uma Action (Atomic Action ou Complex Action) que é executada por dois ou mais agentes. A Figura 4.29 apresenta a interpretação ontológica do elemento *Function*.

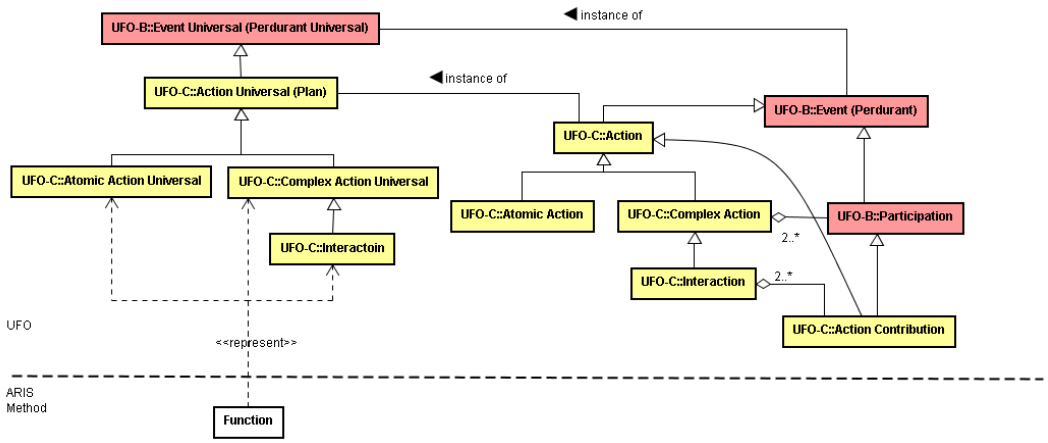


Figura 4.29 - Interpretação ontológica do elemento Function.

Devido às diferentes interpretações ontológicas do elemento *Function* é possível identificar um problema de *incompleteness* na linguagem, i.e., existem entidades na ontologia de referência que não são representadas na linguagem. Desta forma, há uma sobrecarga semântica no elemento *Function*, gerando, assim, o problema de não-lucidez na linguagem de modelagem. Uma possível solução deste problema é a criação de novos elementos notacionais que representem as diferenças ontológicas do elemento *Function*.

A existência de um relacionamento *carries out* entre um *Participant p* e uma *Function f* indica que um agente participará intencionalmente da atividade organizacional representada por *f*. Em outras palavras, existe um compromisso social (Social Commitment) por parte do agente com a organização, no qual, o agente se compromete em ser responsável pela execução e conclusão da atividade organizacional. Este agente contribuirá para a atividade organizacional representada por *f* intencionalmente através de uma Action Contribution.

O relacionamento *contributes to* entre um *Participant p* e uma *Function f* indica que um agente *a* contribui intencionalmente para a conclusão de uma atividade organizacional representada por *f*. Ou seja, o agente *a* possui um compromisso social (Social Commitment) de auxiliar na execução e conclusão da atividade organizacional *f*, porém, não possui nenhuma responsabilidade social com a organização para a conclusão da atividade organizacional *f*.

Dependendo da metaclasse que especializa *p* nos relacionamentos *carries out* e *contributes to*, o agente será uma instância do universal representado por *p* (nos casos de *Position*, *Person Type, Position Description, Organizational Unit Type*) ou será o próprio agente (*Person*, *Organization Unit, Group*).

Os relacionamentos *must be informed about*, *must be informed on cancellation* e *must be informed about result of* são interpretados como Interações (Interaction) que são realizadas entre dois agentes, ou entre um agente e um objeto ou, até mesmo, entre dois objetos. Neste caso, objetos são sistemas computacionais que não possuem características de agentes artificiais (Artificial Agent).

O auto-relacionamento *is predecessor of* do elemento *Function* representa um relacionamento temporal entre duas ou mais atividades do processo de negócio, ou seja, este relacionamento define que para que uma ou mais atividades sejam executadas é necessário que a atividade, antecessora a essas, tenha sido concluída. Esse relacionamento também descreve que uma atividade "produz" um pré-estado (Situation) de uma atividade sucessora a essa. O relacionamento *is predecessor of* é interpretado como uma Time Relation da UFO-B, mais especificamente, uma "before" relation ou uma "meets" relation.

Conforme apresentado na análise ontológica do metamodelo organizacional do ARIS Method (seção 4.3), o elemento *Location* representa uma localização geográfica, sendo assim, não pode ser responsável pela conclusão de uma atividade de processo de negócio. Logo, o relacionamento *is carried out* entre *Location* e *Function* não é valido e, portanto, não será interpretado.

Infelizmente não foi possível definir uma semântica clara dos demais relacionamentos entre *Function* e *Participant.* Desta forma, esses relacionamentos não serão interpretados para a UFO-A, UFO-B ou UFO-C.

4.4.3 Interpretação do Elemento Event

O elemento *Event* é utilizado de duas formas distintas em EPCs: para representar uma situação de interesse (informalmente, um "estado" de interesse) ou para representar uma ação arbitrária que influencia a execução do processo (informalmente um "evento").

No primeiro caso, o elemento *Event* do ARIS Method deve ser interpretado como um Situation Universal. Durante a execução do processo, instâncias deste Universal serão os pré-estados (situações) que habilitam atividades organizacionais ou pós-estados (situações) que resultam destas. Como eventos no ARIS podem envolver uma descrição textual arbitrária, não há regras sobre que tipos de entidades podem estar presentes (present in) na situação. Um evento inicial é denotado pela ausência de atividades do processo que a antecedam (ou seja, não há *Function* relacionada através de *creates* com um evento inicial). Os eventos iniciais denotam portanto as situações que habilitam o processo como um todo. Atividades que habilitem esta situação não fazem parte do

processo. Similarmente, um evento final é denotado pela ausência de atividades do processo que o sucedam.

No segundo caso, o elemento *Event* do ARIS Method é utilizado para representar ocorrências eventos (intencionais ou não intencionais) que dependem de um Substantials.Por exemplo, "o clicar de mouse" ou "mudança de temperatura". Desta forma, o elemento *Event* do ARIS Method é interpretado como um Event Universal da UFO. Como eventos no ARIS podem envolver uma descrição textual arbitrária, não há regras sobre que tipos de entidades podem participar do Event (ou seja, que tipos de Participations são permitidas). A Figura 4.30 apresenta a interpretação do elemento *Event* do ARIS Method como Situation Universal e Event Universal.

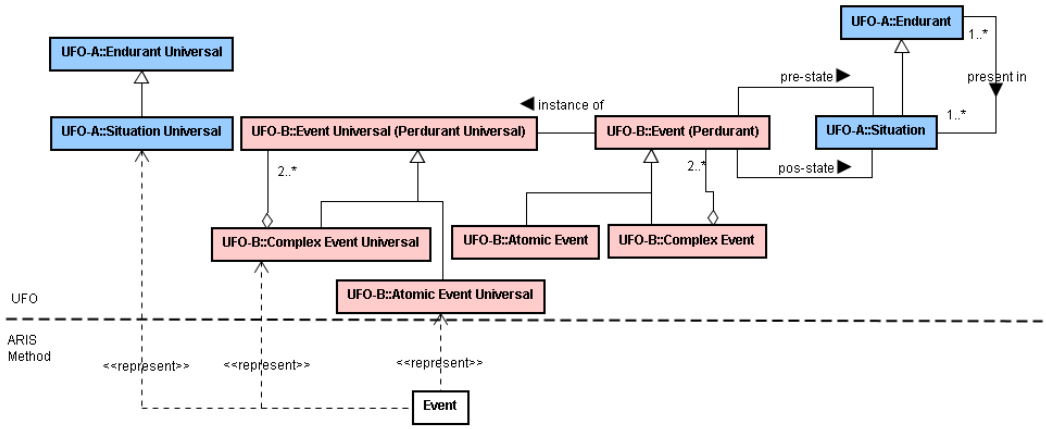


Figura 4.30 – Interpretação do elemento Event.

Para o uso do elemento *Event* do ARIS Method como um Situation Universal, o relacionamento *activates* representa que um ou mais eventos são pré-estados (pre-state Situation) necessários para uma ou mais atividades. Em outras palavras, para que a atividade inicie sua execução é necessário que a situação exista. O relacionamento *creates* entre *Function* e *Event* representa que um ou mais atividades geram situações (pos-state da *Function*) que podem ser pré-estados para outras atividades do processo de negócio.

Caso seja considerada a interpretação que o elemento *Event* do ARIS Method é um Event Universal, o relacionamento *activates representa* um relacionamento temporal, mais especificamente um "Before" ou "Meets", entre os Events representados por *Event* e *Function.* Ou seja, este relacionamento define que para que

uma ou mais atividades sejam executadas é necessário que os eventos antecessores a essas atividades tenham ocorrido. *O relacionamento creates* representa o inverso de activates (também uma relação entre Events).

Devido às diferentes interpretações ontológicas, o elemento *Event* do ARIS Method apresenta o problema de **sobrecarga semântica**. Uma possível solução deste problema é a criação de novos elementos notacionais que representem as diferenças ontológicas do elemento *Event.* Uma outra recomendação possível, é o uso exclusivo de *Events* para representar "estados" (como sugerido por (DAVIS, 2001)). Desta forma, não há sobrecarga semântica.

4.4.4 Análise Ontológica do Elemento Rule

O conjunto de *Rules* presente em um processo de negócio especifica quais atividades devem ser executadas, tomando como base o estado das coisas (do inglês, *state of affairs*). O estado das coisas para o processo de negócio são os eventos internos (produzidos pelas atividades) e os eventos externos (produzidos por outros processos ou por outras entidades) ao processo de negócio.

Através do elemento *Rule* é possível compor eventos complexos a partir da união de eventos internos do processo de negócio e/ou eventos externos ao processo de negócio. Estes eventos complexos podem ser pré-requisitos para que uma determinada atividade do processo ocorra. Por exemplo, o elemento notacional *AND* é utilizado para compor um evento complexo, a partir de dois ou mais eventos do processo de negócio, que pode ser o pré-estado de uma ou mais atividade do processo de negócio. O auto-relacionamento *Links* do elemento *Rule* também pode ser utilizado para criar eventos complexos através de combinações de *Rules.*

Diferentemente dos demais elementos do metamodelo de processo de negócio do ARIS Method, o elemento *Rule* não apresenta um interpretação direta para uma entidade ontológica da UFO. Isto ocorre, porque os *Rule* descrevem relações entre universais que restringem possíveis relações entre instâncias. Por exemplo, assumindo que o elemento *Event* represente uma Situation, temos as seguintes interpretações: **(i) X, Y *AND* Z**: significa que existe um Situation Universal W tal que toda instância de Z tem como

pré-situação uma instância w de W e toda instância de w é a soma mereológica de uma instância de X e uma instância de Y (ambas situações); **(ii) X,Y *OR* Z**: significa que toda instância de Z tem como pré-situação uma instância w tal que w é uma instância de X ou de Y ou a soma mereológica ambas instâncias; e **(iii) X,Y XOR Z** é igual a **(ii)** com o requisito adicional de que X e Y sejam Situation Universals que são disjuntos.

4.4.5 Análise Ontológica do Application System, Application System Type e Application System Class

O elemento *Application System* representa um sistema computacional que auxiliam na execução do processo de negócio. Por exemplo, o editor de texto instalado na máquina de um empregado da organização.

Desta forma, o elemento *Application System* pode ser interpretado na UFO-C como duas entidades: **(i)** Object Particular ou **(ii)** um Agent Artificial Particular. A diferença de interpretação do elemento *Application System* é devido ao uso do elemento. A Figura 4.31 apresenta a interpretação do elemento *Application System*.

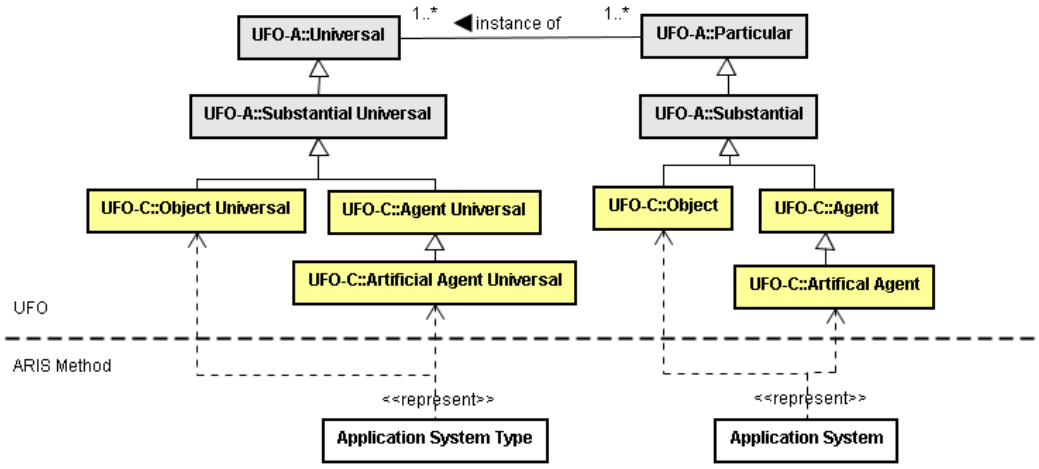


Figura 4.31 – Interpretação dos elementos Application System e Application System Type.

A primeira interpretação ocorre quando o sistema computacional não possui características de agentes (desejos, crenças e intenções) como, por exemplo, o editor de texto instalado na máquina de um empregado da organização.

A segunda interpretação ocorre quando o sistema computacional possui características de agentes como, por exemplo, um sistema orientado a agentes *A* que está em operação em um computador *Z*.

O elemento *Application System Type* representa uma tipificação de um sistema computacional (*Application System*). Em outras palavras, esse elemento representa o Universal que foi instanciado para criar um determinado *Application System*. Desta forma o elemento *Application System Type* é interpretado como um Object Universal e como um Agent Artificial Universal. A figura abaixo apresenta a interpretação do elemento *Application System Type*.

Como pode ser observado na figura acima, os elementos *Application System* e *Application System Type* possuem duas interpretações ontológicas, desta forma caracterizando um problema de **sobrecarga semântica**. Para corrigir este problema, é necessário criar mais um elemento no metamodelo que seja utilizado para indicar que um sistema é um objetivo ou um agente.

Os elementos *Application System* e *Application System Type* também podem ser interpretados, respectivamente, como Resource Particular e Resource Universal da UFO-C quando representam objetos (*Object Particular* e *Object Universal*) que são utilizados pelas entidades organizacionais na execução de uma atividade (Function). Isso pode ser comprovado pela semântica dos relacionamentos *can be user* e *is user* entre os elementos organizacionais e o *Application System* e *Application System Type*. Esses relacionamentos nesse contexto são interpretados como Resource participation, mais especificamente, Usage e Usage Universal.

O elemento *Application System Class* representa uma classificação realizada ao tipo de aplicações computacionais (*Application System Type*). Sendo que um mesmo tipo de aplicação computacional pode pertencer a mais de uma classificação de aplicações. Assim, o elemento *Application System Class* é interpretado como a entidade Category da UFO-A. Uma Category é um tipo de Mixin, que classifica as entidades que pertencem a tipos diferentes, mas que compartilham uma propriedade comum essencial (ou seja, uma propriedade que eles não podem perder) (GUIZZARDI, 2005a). A Figura 4.32 apresenta a interpretação do elemento *Application System Class*.

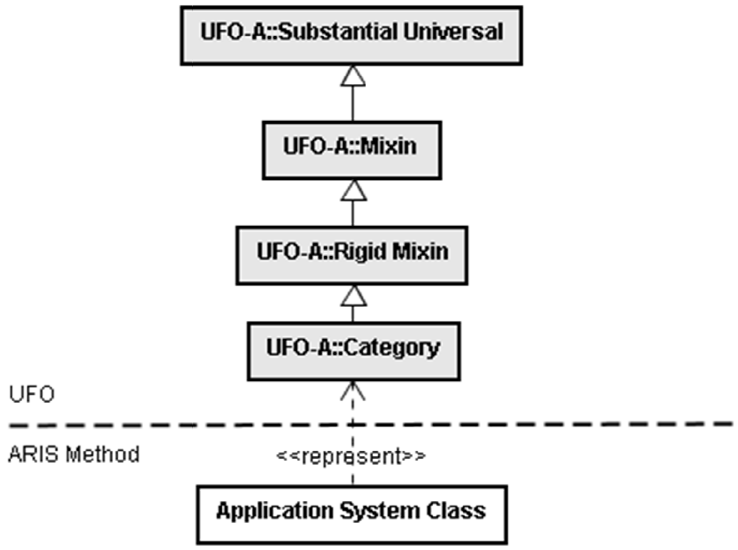


Figura 4.32 – Interpretação do elemento Application System Class.

O relacionamento *supports* entre *Application System* e *Application System Type* e o elemento Function é interpretado ontologicamente de duas maneiras distintas. Caso os elementos *Application System* e *Application System Type* sejam interpretados, respectivamente, como Object Particular e Object Universal, o relacionamento *supports* é interpretado como um Resource Participation, mais especificamente, Usage e Usage Universal.

A segunda interpretação do relacionamento *supports* leva em consideração que os elementos *Application System* e *Application System Type* são interpretados como Artificial Agent e Artificial Agent Universal. Desta forma, tal relacionamento é interpretado como Action Contribution.

O elemento *Application System Class* também possui o relacionamento *supports* com o elemento *Function.* Não Há conceitos na UFO-A, UFO-B e UFO-C que descreva que uma Category possa executar um atividade ou ser usada em uma atividade. Porém, é possível expressar situações do tipo "A categoria de aplicação X auxilia na execução de tais tipos de atividades" ou "A categoria de aplicação Y é responsável por executar tais tipos de atividades". Desta forma, não há um problema ontológico nesse relacionamento.

Os demais relacionamentos (*is technically responsible for*, *is resposible for development of*) não possuem uma interpretação ontológica de devido a falta de documentação destes relacionamentos.

4.4.5.1 Problemas Ontológicos com a Semântica do Elemento Function Utilizando o Elemento Application System.

Segundo o conceito de Function apresentada na seção, uma atividade do processo de negócio pode ser desempenhada por um sistema computacional, ou seja, uma atividade pode ser de responsabilidade e/ou executada por uma aplicação computacional.

Porém, não existe um relacionamento entre uma aplicação computacional (*Application System, Application System Type* e *Application System Class*) e um *Function* que descreva tal situação. Desta forma, é identificado um problema de **não-solidez** na linguagem de processos de negócio do ARIS Method, ou seja, existe uma conceitualização do domínio de Processos de Negócio que não pode ser expresso pela linguagem.

Para resolver tal problema, será necessário adicionar um novo relacionamento com a mesma semântica do *carries out* entre os elementos que representam sistemas computacionais (*Application System*, *Application System Type* e *Application System Class)* e o elemento *Function*.

4.5 ANÁLISE ONTOLÓGICA DO METAMODELO DE DETALHAMENTO DE ATIVIDADES

Para realizar a análise ontológica dos elementos do Metamodelo de Detalhamento de Atividades a etapa “Analisar semântica do elemento e seus relacionamentos” tomará como base a semântica apresentada na 0. Também será utilizada as interpretações ontológicas dos elementos Organizacionais e de Processo de Negócio apresentadas, respectivamente, nas seções 4.3 e 4.4.

4.5.1 Análise Ontológica do Cluster, Technical Term e Information Carrier

Os elementos *Cluster* e *Technical Term* representam informações e o elemento *Information Carrier* representa veículos de armazenamento de informações. Levando em consideração apenas esses conceitos, os elementos *Cluster*, *Technical Term* e *Information Carrier* são interpretados com Object Universal. A Figura 4.33 apresenta a interpretação dos elementos *Cluster*, *Technical Term* e *Information Carrier*.

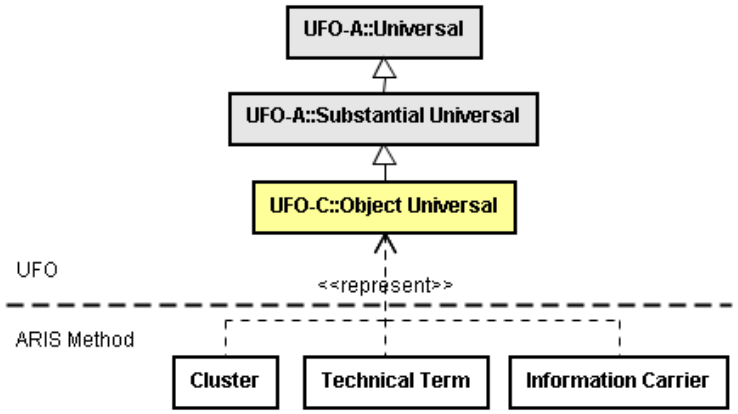


Figura 4.33 – Interpretação dos elementos Cluster, Technical Term e Information Carrier.

No entanto, no contexto do processo de negócio, mais especificamente no contexto da execução da atividade, os elementos *Cluster*, *Technical Term* e *Information Carrier* representam recursos que são criados, utilizados, modificados e destruídos em uma ou mais atividades de um processo de negócio. Isto pode ser comprovado ao analisar a semântica dos relacionamentos *creates*, *deletes*, *change* e *read* que os elementos *Cluster* e *Technical Term* possuem com o elemento *Function* e ao analisar a semântica dos relacionamentos *create output to*, *reads*, *provides input for* e *is used by* que ocorre entre o elemento *Information Carrier* e os elementos organizacionais e *Function*. Logo, os elementos *Cluster*, *Technical Term* e *Information Carrier*, no contexto de processo de negócio, também podem ser interpretados como Resource Universal.

Os relacionamentos *creates*, *deletes*, *change* e *read* que ocorrem entre os elementos *Cluster* e *Technical Term* com o elemento *Function* representam, respectivamente, as seguintes *Resource Participation Universal*: *Creation Universal*, *Termination Universal*, *Change Universal* e *Usage Universal*.

O relacionamento *lies on* entre *Information* (*Cluster* e *Technical Term*) e *Information Carrier* é interpretado como um relacionamento formal, utilizado para descrever o relacionamento entre a informação e meio de armazenamento.

O relacionamento *use* entre *Cluster* e *Application System* representa que uma informação é utilizada por um sistema computacional é interpretado como um Usage.

O relacionamento *can use* entre *Cluster* e *Application System* representa que uma informação é utilizada por um tipo de sistema computacional é interpretado como um Usage.

O relacionamento *is owner of* entre *Participant* e *Information* representa que uma ou mais entidades organizacionais são os responsáveis legais pelas informações. Desta forma, esse relacionamento interpretado com um *social moment* que uma ou mais entidades organizacionais assumem ao criarem uma ou mais informações dentro da organização.

O relacionamento *is specimen owner of* entre *Organization Unit* e *Cluster* possui a mesma interpretação que o relacionamento *is owner of* entre *Participant* e *Information*.

O relacionamento *reads* entre *Function* e *Information Carrier* representa que a atividade "ler" as informações contidas em uma veiculo de armazenamento. Desta forma, pode-se interpretar esse relacionamento com um Usage Universal (Resouce Participation) indireto, pois o recurso que está sendo utilizado é a informação contida no veiculo de armazenamento.

Por fim, é importante destacar que o elemento *Information Carrier* não possui nenhum relacionamento com os demais elementos do metamodelo que descreva os Resource Participation: Create, Terminate e Change. Desta forma, não é possível expressar situações de criação, exclusão ou modificação do meio de armazenamento. Assim, propomos que seja adicionado no metamodelo de detalhamento de atividade relacionamentos que possibilitem expressar as situações citadas.

4.5.2 Análise Ontológica do Elemento Business Rule

Infelizmente a documentação on-line da ferramenta, (SCHEER, 1999) e (DAVIS, 2001)não apresentaram a semântica para o elemento *Business Rule*. Desta forma, será necessário analisar o metamodelo de detalhamento de atividades na tentativa de encontrar a semântica deste elemento e assim realizar a análise ontológica.

No metamodelo de detalhamento de atividade o elemento *Business Rule* possui um relacionamento com o elemento *Function* chamado *describes*. Assim, é possível concluir, que o elemento *Business Rule* é utilizado para descrever as regras que ditam o comportamento de uma ou mais atividades do processo de negócio ou até mesmo de um processo de negócio. De acordo com Guide Business Rule (2008 *apud* AZEVEDO et al., 2009), as regras de negócio são declarações de políticas ou condições que devem ser satisfeitas pelos processos da organização. As regras de negócio também e definem ou restringem alguns aspectos da organização e são utilizadas para explicitar a dependência entre as atividades organizacionais definindo o fluxo do processo de negócio. Além disto, as regras de negócio definem regras especificas de uma atividade do processo de negócio.

Tendo com base o conceito apresentado no parágrafo acima, o elemento *Business Rule* é interpretado como um Technical Norms. A figura abaixo apresenta a interpretação do elemento *Business Rule*.

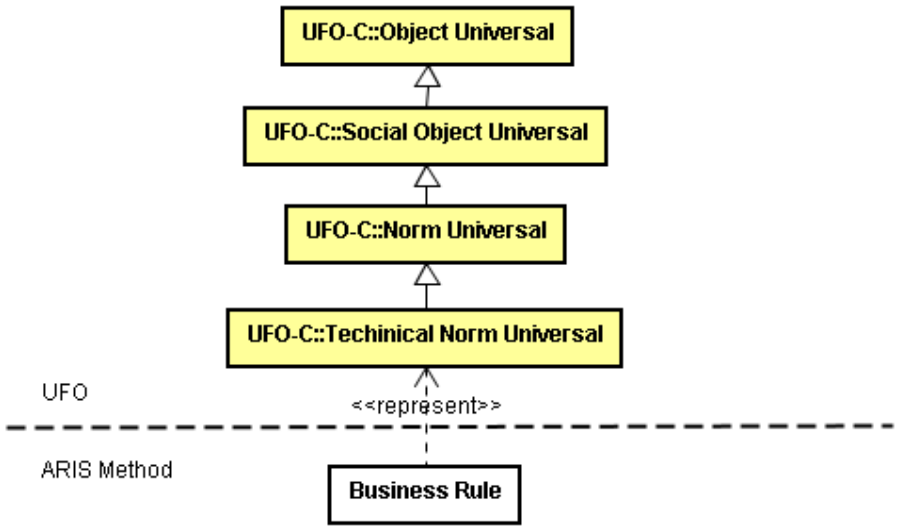


Figura 4.34 – Interpretação do elemento Business Rule.

4.6 DISCUSSÃO SOBRE ANÁLISE ONTOLÓGICA E TRABALHOS CORRELATOS

Um dos trabalhos que mais se assemelham a este é o apresentado por Green e Rosemann em (GREEN *et al.*, 2004)(GREEN *et al.*, 2004b)(GREEN e ROSEMANN, 2005). Green e Rosemann discutem a avaliação ontológica das linguagens presentes no ARIS Method com base na ontologia BWW(WEBER, 1997). A análise ontológica proposta neste trabalho difere do trabalho de Green e Rosemann em três aspectos principais.

Primeiramente, utilizamos como ponto de partida para a nossa avaliação fragmentos dos metamodelos que representam fielmente as linguagens de modelagem que estão implementadas na ferramenta ARIS Toolset. Desta forma, a avaliação se aplica aos modelos produzidos nesta ferramenta. Em sua vez, Green e Rosemann consideraram o metamodelo apresentado por Scheer (SCHEER, 1999) que já não retrata a linguagem como adotada nas organizações que utilizam a ferramenta ARIS Toolset.

Adicionalmente, Green e Rosemann interpretam apenas os elementos da linguagem isoladamente. Neste trabalho, avaliamos também a interpretação dos relacionamentos entre os elementos da linguagem (meta-associações no metamodelo).

Por fim, este trabalho utilizou uma ontologia de fundamentação diferente na análise ontológica. Isto nos permitiu chegar a diferentes conclusões com relação à interpretação de alguns elementos de modelagem. Por exemplo, nossa interpretação de *Function* difere significativamente daquela apresentada em(GREEN *et al.*, 2004). Naquele trabalho, uma *Function* é interpretada como um mapeamento formal entre dois valores em um espaço conceitual de estados. Desta forma, Green e Rosemann não fornecem uma explicação ontológica, por exemplo, para: **(i)** participação de agentes em atividades organizacionais; **(ii)** noção de tempo associada à execução de uma atividade organizacional; **(iii)** definição de funções complexas através de relações mereológicas entre funções e sub-funções; dentre outras relações que envolvem funções, eventos, agentes e compromissos.

Outros autores, usando o termo "Semantic Business Process Management" [39], propõem realizar anotações nos elementos dos modelos de processo de negócio para adicionar informações ("anotação semântica") nesses modelos. Tais anotações, normalmente estão se referindo a entidades de uma ontologia do domínio de negócio com o objetivo de definir objetivos, pré-condições e efeitos das atividades no modelo de processo de negócio. Este trabalho reconhece que a análise ontológica proposta neste trabalho é um pré-requisito para fundamentar as "anotações semânticas", quando os elementos do modelo anotado e os elementos da anotação são interpretados em termos de entidades de uma ontologia de fundamentação.

A abordagem inicialmente esboçada em (THOMAS e FELLMANN, 2006) foi desenvolvida em (THOMAS e FELLMANN, 2007) onde os autores propõem uma abordagem de anotação semântica em EPC baseada em uma ontologia que espelha um pequeno fragmento do metamodelo EPC. Infelizmente, tal abordagem proposta não clarifica a semântica dos elementos de modelagem do EPC.

O projeto SUPER tem investigado a semântica do EPC (sEPC)(FILIPOWSKA *et al.*, 2009a)(FILIPOWSKA *et al.*, 2009b). Essa abordagem é baseada na transformação do sEPCs na Ontologia de Modelagem de Processo de Negócio (Business Process Modeling Ontology – BPMO)(SUPER, 2008)(SUPER, 2009). O BPMO, por sua vez, especializa conceitos de uma ontologia chamada UPO. A UPO é inspirada no DOLCE(BOTTAZI e FERRARIO, 2006). A relação entre sEPCs a sua fundamentação

semântica não é direta, e consequentemente este projeto não apresenta a interpretação semântica do EPCs.

Por fim, outros trabalhos na literatura visam especificamente a análise de propriedades formais do metamodelo de processo de negócio do ARIS utilizando Rede de Petris (KINDLER, 2006) (ROSEMANN e VAN DER AALST, 2007)(VAN DER AALST, 1998)(VAN DONGEN *et al.*, 2005). Este trabalhos são complementares ao descrito aqui, pois definem semânticas de execução para EPCs.

4.7 CONCLUSÃO

A abordagem de análise ontológica apresentada neste capítulo possibilitou um melhor entendimento dos conceitos presentes nas linguagens de domínio do ARIS Method com base em uma ontologia de fundamentação.

Um benefício direto da análise ontológica dos metamodelos do ARIS Method está relacionado ao desenvolvimento de modelos organizacionais (organizacional, processo de negócio e detalhamento de atividades) com semântica bem definida. Afinal, não haverá elementos nas linguagens de domínio com semântica duvidosa e elementos em excesso (elementos sem nenhuma relação semântica com o domínio da linguagem). Com as linguagens de modelagem sem problemas ontológicos as técnicas de desenvolvimento de aplicativo computacionais poderão tirar um maior proveito da semântica dos elementos destas linguagens.

A avaliação ontológica permitiu também revelar problemas de uso (por exemplo, caso do uso do elemento *Event*). A análise permitiu identificar de forma sistemática, um problema já identificado informalmente na literatura (DAVIS, 2001, p. 111).

CAPÍTULO 5 UMA ABORDAGEM DE DESENVOLVIMENTO ORIENTADO A MODELOS UTILIZANDO ARTEFATOS DE ARQUITETURA ORGANIZACIONAL DE TI

5.1 INTRODUÇÃO

Os modelos produzidos pelas Arquiteturas Organizacionais de TI são artefatos ricos em informação que podem ser utilizados no desenvolvimento de sistemas computacionais através de técnicas de Desenvolvimento Orientado a Modelos.

Desta forma, diversos autores propuseram técnicas de Desenvolvimento Orientado a Modelos que utilizam os modelos das Arquiteturas Organizacionais de TI no desenvolvimento de sistemas computacionais, dentre eles:

i. Lubke *et al* (2008) propõe uma abordagem que permite utilizar os modelos de processos de negócio, modelados em EPC, para o desenvolvimento de sistemas orientado a serviços para empresas de pequeno e médio porte;

ii. Specht *et al* (2005) apresentam uma abordagem que permite utilizar os modelos de processos, modelados em EPC, e os modelos de detalhamento de atividade, modelados em FADs para o desenvolvimento de um sistema orientado a serviços;

iii. Ouyang *et al* (2008) propõe uma abordagem de automatização de modelos BPMN (BPMN, 2009). Tal automatização é feita através de uma transformação que traduz o modelo BPMN em um modelo BPEL (OASIS, 2006) pronto para execução em uma BPEL *engine*;

iv. Ziemann e Mendling (2005) apresentam uma abordagem de transformação de modelos de processos de negócio modelados na linguagem EPC para modelos em BPEL.

Apesar das abordagens supracitadas utilizarem os modelos de processo de negócio para o desenvolvimento de sistemas computacionais orientados a processos, tais abordagens descartam quase que completamente os outros modelos que estão presentes nas Arquiteturas de TI. Por exemplo, grande parte das abordagens supracitas utiliza apenas o modelo EPC para a construção dos sistemas computacionais. Desta forma, tais abordagens não garantem sistematicamente o alinhamento dos diferentes aspectos da organização (organizacional, detalhamento de atividades de processos de negócio, entre outros) com os sistemas computacionais desenvolvidos. Logo, a tarefa de alinhamento entre os diversos aspectos deve ser conduzida manualmente, aumentando o esforço necessário durante o processo de desenvolvimento e perdendo-se oportunidades de automação.

Adicionalmente, grande parte das abordagens apresentadas acima não permite que o projetista manipule por completo as informações contidas nos modelos das Arquiteturas Organizacionais de TI. Em outras palavras, as abordagens atuais não permitem ao projetista utilizar por completo a semântica dos elementos presentes nos modelos das Arquiteturas Organizacionais de TI no desenvolvimento do sistema computacional. Por exemplo, a abordagem proposta por Lubke (*et al*,. 2008) não diferencia atividades executadas por humanos e atividades executadas por aplicativos computacionais.

As abordagens supracitadas também não permitem que o projetista introduza decisões de projeto nas transformações que manipulam os modelos, com a finalidade de alcançar o sistema computacional desejado. Isso ocorre, pois grande parte dessas abordagens consistem em traduções diretas entre *espaços tecnológicos* (um termo definido em (KURTEV *et al.*, 2002)), ou seja, transformam os modelos de processo de negócio diretamente em um modelo de realização (linguagem de programação, XML, entre outros). Desta forma, os modelos produzidos nas Arquiteturas Organizacionais de TI são considerados artefatos finais, ou seja, assume-se que os modelos das Arquiteturas Organizacionais de TI estão prontos para serem automatizados. Esta abordagem colapsa o nível de modelagem organizacional (conceitual) e o nível detalhado de projeto, o que é inadequado, uma vez que, na maioria dos casos os modelos das Arquiteturas Organizacionais de TI foram criados para representar uma realidade organizacional e não para produção de sistemas computacionais. Logo, é necessário analisar os modelos das Arquiteturas de TI para identificar como os mesmos podem ser automatizados ou

como as informações contidas nos modelos podem ser úteis para construir sistemas computacionais que automatizem por completo ou parcialmente certos aspectos da organização (AZEVEDO *et al.*, 2009)(GONZÁLEZ e DÍAZ, 2007) (KOHLBORN *et al.*, 2009).

Como discutido em(CARDOSO *et al.*, 2008), a "união" das decisões de projeto com as informações contidas nos modelos das Arquiteturas Organizacionais de TI pode proporcionar a construção de diversos sistemas computacionais, a partir do mesmo modelo de Arquitetura Organizacional de TI. A maioria das abordagens de Desenvolvimento Orientado a Modelos com base em modelos de Arquiteturas Organizacionais de TI ou modelos de processos de ignora esta possibilidade. Desta forma, tais abordagens se baseiam na representação direta de soluções tecnológicas nos Modelos de Arquiteturas Organizacionais de TI, perdendo as vantagens da separação entre interesses (do inglês, *concerns*) organizacionais e tecnológicos.

Para suprir as deficiências apresentadas acima, é proposta uma nova abordagem de Desenvolvimento Orientado a Modelos para o desenvolvimento de sistemas computacionais orientados a processo. Tal abordagem permite que o Projetista utilize as informações contidas nos diversos modelos das Arquiteturas Organizacionais de TI e aplique as decisões de projeto nas transformações, com a finalidade de desenvolver um sistema computacional mais adequado às suas necessidades. A Figura 5.1 apresenta a abordagem de Desenvolvimento Orientado a Modelos proposta.

A abordagem é composta de três transformações parametrizadas como apresentado na Figura 5.1: **(i)** Transformação de Abstração (TA), **(ii)** Transformação de Refinamento (TR) e **(iii)** Transformação de Detalhamento de Atividade (TDA). Cada uma dessas transformações gera resultados que estão de acordo (em conformidade) com as informações contidas nos modelos das Arquiteturas Organizacionais de TI e não modificam decisões de projeto já tomadas em outras transformações. (De acordo com a teoria sobre conformidade e transformações apresentada na seção 2.4.3.)

Uma característica importante da abordagem de Desenvolvimento Orientado a Modelos proposta é a sua independência de plataforma (veja a seção 2.4.2), ou seja, as transformações TA (Transformação de Abstração), TR (Transformação de

Refinamento) e TDA (Transformação de Detalhamento de Atividade) não estão relacionadas com nenhuma tecnologia ou plataforma tecnológica. Desta forma, os modelos de sistemas computacionais gerados podem ser utilizados para gerar sistemas computacionais em diferentes tecnologias e plataformas tecnológicas (BPEL, JBOSS jBPM entre outras).

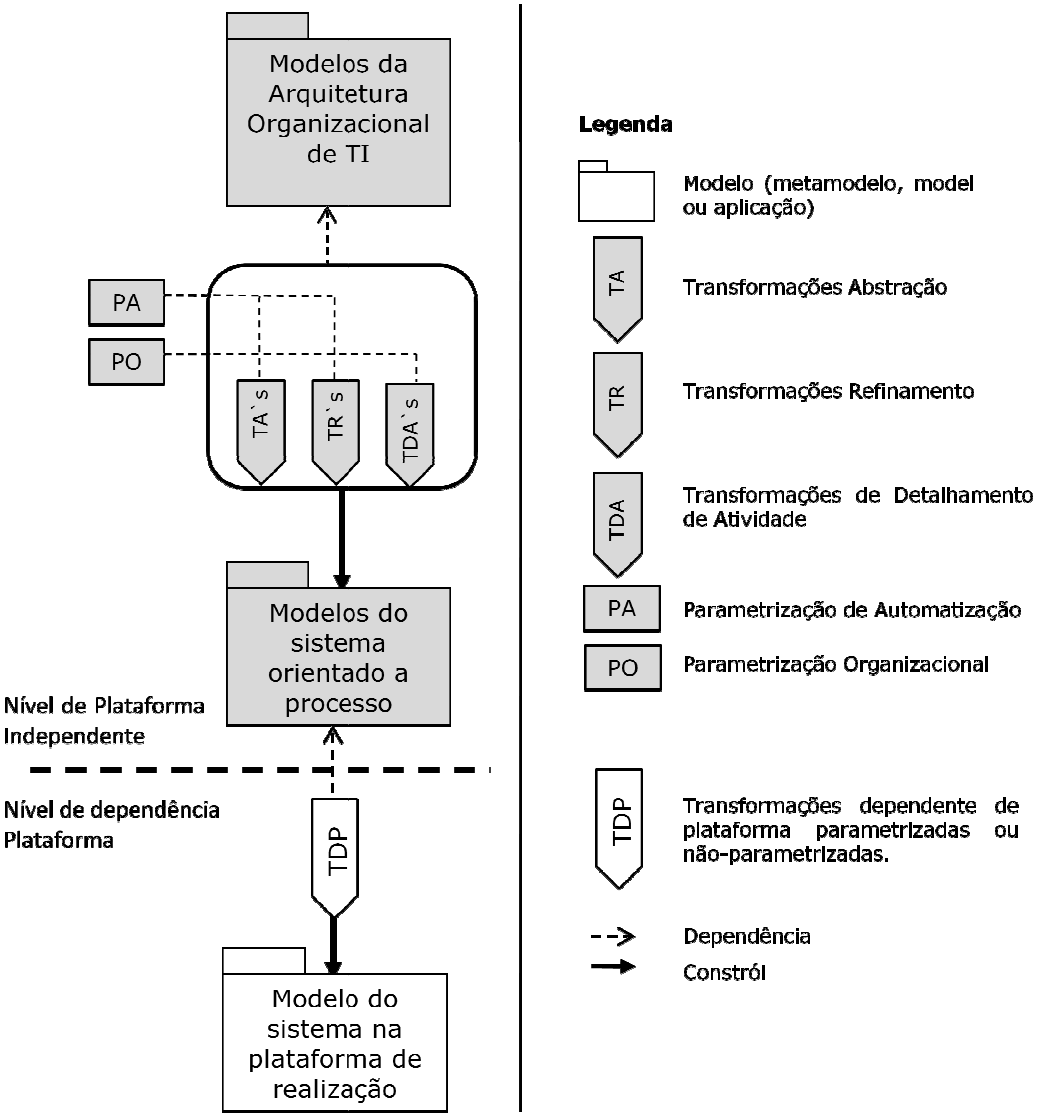


Figura 5.1 – Abordagem de Desenvolvimento Orientado a Modelos utilizando artefatos de Arquitetura Organizacional de TI.

As Transformações de Abstração (TA) e de Refinamento (TR) são responsáveis por alterar seletivamente a granularidade de um modelo de processos para construir um novo modelo que será utilizado para ditar as regras do comportamento do sistema computacional. Em outras palavras, essas transformações permitem que o Projetista

aplique decisões de projeto sobre o modelo de processo (modelo conceitual), transformando esse em um modelo comportamental (modelo de projeto) adequado para o sistema computacional em desenvolvimento.

As Transformações de Abstração e Refinamento, neste trabalho, serão chamadas de Transformações no nível comportamental. Tanto a transformação de Abstração (TA), quanto a transformação de Refinamento (TR) utilizam o conceito chamado de Componente como base para as suas etapas. De forma simplória, um componente é a maneira de representar um conjunto de atividade de menor nível de abstração em uma atividade de maior nível de abstração. (A teoria sobre componentes de processo de negócio é apresentada na seção 5.2.)

A Transformação de Detalhamento da Atividade (TDA) permite ao Projetista informar (selecionar) quais elementos do diagrama de detalhamento de atividades são necessárias, baseado em um ou mais decisões de projeto, para o desenvolvimento do sistema de computacional Como apresentado na seção 0, os modelos de detalhamento de atividade são utilizados para descrever os detalhes e, assim, possuem grandes informações (por exemplo, informações necessárias e produzidas na atividade) que podem ser úteis para o desenvolvimento de atividade que será automatizada. O resultado da TDA é um modelo de detalhamento de atividade que contem apenas os elementos necessários para o desenvolvimento do sistema computacional.

A abordagem de Desenvolvimento Orientado a Modelos proposta é composta de três tipos de parametrizações que guiam as transformações supracitadas: Parametrização de Automatização (PA) e Parametrização Organizacional (PO). Como demonstrado na Figura 5.1, a parametrização PA é utilizada exclusivamente nas Transformações de Abstração e Refinamento e a parametrização PO é utilizada em todas as transformações.

A Parametrização de Automatização (PA) permite ao projetista decidir o grau de automatização de cada atividade do modelo comportamental do sistema computacional em realização. A seção 5.5.1 apresenta maiores detalhes sobre esta parametrização.

A Parametrização Organizacional (PO) possibilita ao projetista manipular as entidades organizacionais presentes nos modelos organizacionais para a identificação dos atores

do sistema computacional em realização. Por exemplo, através dessa parametrização é possível ao projetista informar quais entidades organizacionais irão executar um determinado papel no sistema computacional em desenvolvimento.

No final da execução das transformações do nível independente de plataforma será gerado um conjunto de modelos do sistema orientado a modelos. Esse conjunto é composto por modelos que descrevem a estrutura comportamental, detalhes de cada componente de processo de negócio integrante da estrutura comportamental e as entidades organizacionais que fazem parte do sistema computacional em realização, como demonstrado na Figura 5.2.

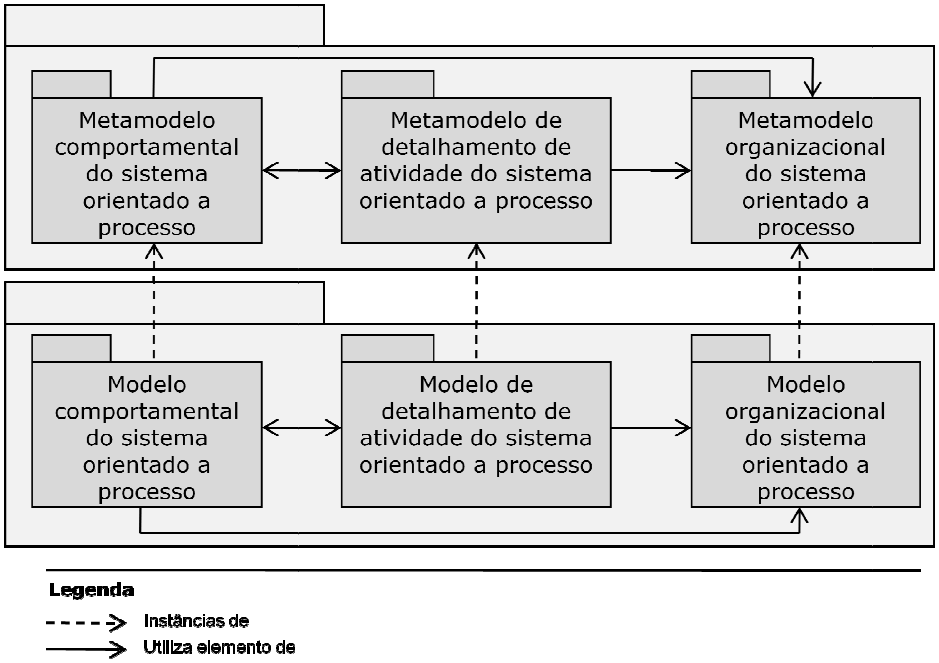


Figura 5.2 – Modelo representando as relações dos metamodelos do sistema orientado a processo de negócio

A descrição do sistema orientado a processos é realizada através de instâncias dos seguintes metamodelos:

i. **Metamodelo Comportamental do Sistema Orientado a Processos**: utilizado para criar modelos comportamentais que descrevam o comportamento funcional do sistema computacional em realização, e, assim, descrever quais componentes serão executados e quais entidades organizacionais serão responsáveis pelos mesmos no escopo do sistema

ii. **Metamodelo de Detalhamento de Atividade do Sistema Orientado a Processo**: utilizado para criar modelos que descrevam os detalhes de cada componente de processo de negócio que estão presentes no modelo comportamental; e

iii. **Metamodelo Organizacional do Sistema Orientado a Processo**: utilizado para criar modelos que descrevam as entidades organizacionais que estão no modelo comportamental e nos modelos de detalhamento.

A última etapa da abordagem proposta ocorre no nível dependente de plataforma. Nesse nível, o projetista utiliza os modelos criados no nível independente de plataforma para a concretização do sistema computacional através das Transformações Dependentes de Plataforma (TDP). As Transformações Dependentes de Plataforma são responsáveis por aplicar as decisões de projeto no nível tecnológico, gerando o sistema computacional em uma determinada tecnologia ou plataforma tecnológica escolhida pelo projetista.

A Figura 5.3 apresenta a comparação entre as abordagens de Desenvolvimento Orientado a Modelos atuais e a abordagem proposta neste trabalho. Como pode ser observado, a abordagem proposta permite ao projetista produzir diversas soluções tecnológicas enquanto as abordagens atuais permitem a produção de um tipo de solução tendo com base os modelos das Arquiteturas Organizacionais de TI.

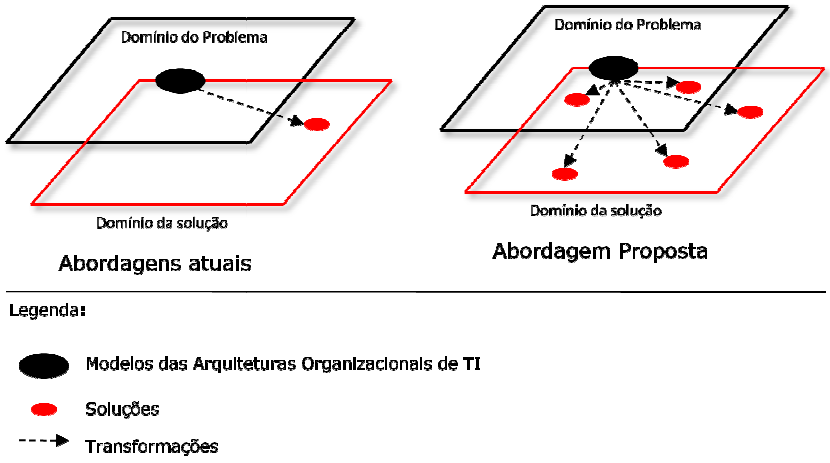


Figura 5.4 – Diferença entre a abordagem atuais e a abordagem proposta

Para comprovar que a abordagem proposta neste trabalho permite gerar diversos sistemas a partir de um conjunto de modelos de uma Arquitetura Organizacional de TI

imagine o seguinte exemplo: um modelo de processo com três atividades e cada atividade com apenas um responsável. Utilizando apenas as três possibilidades de automatização da Parametrização de Automatização (totalmente automatizada, gerenciada e manual) (apresentadas na seção 5.5.1) é possível gerar vinte e seis sistemas computacionais diferentes a partir desse modelo de processo de negócio. Caso sejam consideradas as Transformações de Abstração, Refinamento e Detalhamento de Atividade o número de possibilidades de sistemas gerados a partir de um processo de três atividades será ainda maior. Desta forma, é comprovado que a abordagem proposta permite o desenvolvimento de diversos sistemas computacionais tendo como base as decisões de projeto e os modelos das Arquiteturas Organizacionais de TI.

O capítulo está organizado da seguinte forma: a seção 5.2 apresenta o conceito de componentes, a seção 5.3 apresenta as transformações do nível comportamental, a seção 5.4 apresenta a transformação do nível de atividade e a seção 5.5 apresenta as parametrizações utilizadas nas transformações. Por fim, a seção 5.6 apresenta as conclusões e trabalhos futuros deste capítulo.

5.2 COMPONENTE DE PADRÕES DE PROCESSO

Um componente de padrões de processo é uma atividade de maior nível de abstração que engloba um ou mais padrões de processo de um determinado processo (OUYANG *et al.*, 2008). Em outras palavras, o componente é uma forma de representar que um conjunto de atividade (que possuem o mesmo objetivo) podem ser "agrupadas" em uma nova atividade de maior nível de abstração.

De acordo com Ouyang (*et al.,* 2008), um componente é uma atividade que possui somente uma entrada e uma saída. A entrada do componente (do inglês, *source place*) é responsável por receber as informações a serem manipuladas pelo componente e a saída da componente (no inglês, *sink place*) é a responsável por enviar as informações manipuladas para o próximo componente (caso exista).

Os pontos de entrada e saída do componente são os responsáveis pelas fronteiras de controle de fluxo (comportamento) e da informação do componente (OUYANG *et al.*, 2008).

Segundo Ouyang (*et al.,* 2008), o componente possui as seguintes características:

i. um componente é composto por pelo menos um objeto elementar do BPMN(BPMN, 2009) como, por exemplo, o elemento fork. (OUYANG *et al.*, 2008);

ii. um componente possui um, e somente um, ponto de entrada responsável por receber a informação a ser manipulado pelo componente;

iii. um componente possui um, e somente um, ponto de saída responsável por enviar para fora do componente a informação manipulada pelo componente;

iv. um componente é o caminho entre dois pontos do modelo de processo de negócio que estão entre os eventos: inicial e final; e

v. um componente possui apenas eventos intermediários em sua definição.

Propomos uma extensão do conceito de componente com as seguintes modificações:

i. um componente é composto por pelo menos um padrão de processo (ao invés de componentes de BPMN) e;

ii. um componente pode ser composto de outros componentes.

A Figura 5.5 apresenta um exemplo de componente que combina os padrões de processo *Parallel Split* (RUSSELL e HOFSTEDE, 2008) e *Multi-Merge* (RUSSELL e HOFSTEDE, 2008).

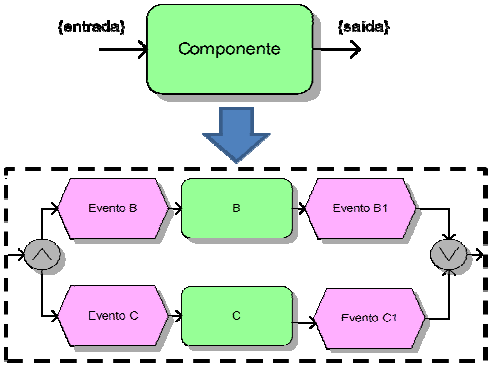


Figura 5.5 – Exemplo de Componente.

Como pode ser observado na Figura 5.5, o componente foi criado porque a combinação dos dois padrões de projeto formou um "componente" com apenas uma entrada e uma saída.

Para finalizar, diferente de como o conceito de componente é utilizado em (OUYANG *et al.*, 2008) e em (VANHATALO *et al.*, 2007)[4], neste trabalho o conceito de componente é utilizado para definir as fronteiras, no modelo de processo, onde as transformações serão aplicadas. O conceito de componente também é utilizado para definir a fronteira do resultado das transformações. Por exemplo, é possível definir que um componente é composto de dois padrões de processo contidos em um processo de negócio e uma determinada transformação será aplicada sobre esse componente; ou que através de uma transformação uma determinada atividade será substituída por dois padrões de processo.

5.3 TRANSFORMAÇÕES NO NÍVEL COMPORTAMENTAL

As transformações no nível comportamental abrangem a Transformação de Abstração e a Transformação de Refinamento. Essas transformações são aplicadas seletivamente pelo projetista para transformar os modelos de processo de negócio (modelo conceitual) em modelos comportamentais (modelo de projeto) que descrevam a seqüência de atividades que o sistema computacional deve realizar. A Figura 5.6 apresenta uma exemplificação da aplicação da transformação de abstração e refinamento sobre um modelo de processo de negócio.

---

[4] Em (VANHATALO *et al.*, 2007) o conceito de componente é chamado de *Single Entry Single Exit* (SESE)

Como discutido anteriormente, os modelos de processo de negócio são utilizados para representar uma realidade organizacional e não para o desenvolvimento de sistemas computacionais. Devido a esse fato, é necessário que o projetista modifique o modelo de processo de negócio gerando um modelo comportamental do sistema computacional que atenda as decisões de projeto. Por exemplo, na Figura 5.6 as atividades *A*, *B* e *C* foram abstraídas para o componente (atividade) *ABC* através da transformação de abstração (TA) tomando como base uma ou mais decisões de projeto, enquanto a atividade *D* foi refinada em um componente, que contém as atividades *D1* e *D2*, através da transformação de refinamento (TR) tomando como base uma decisão de projeto.

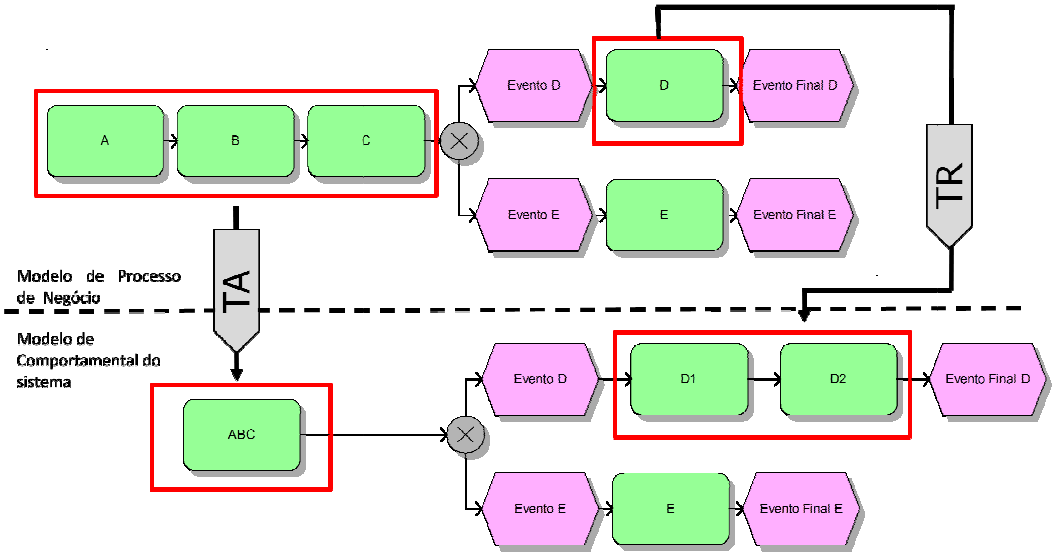


Figura 5.6 – Exemplo de utilização das transformações no nível comportamental.

5.3.1 Transformação de Abstração (TA)

A Transformação de Abstração (TA) permite que o projetista una duas ou mais atividades, dois ou mais componentes ou dois ou mais padrões de processo[5] em um único componente de mais alto nível de abstração. Em outras palavras, pode-se dizer que a Transformação de Abstração constrói um componente de maior nível de abstração a partir de dois ou mais componentes de menor nível de abstração.

[5] Um padrão de processo (do inglês, *workflow pattern*) é uma abstração de uma forma concreta que ocorre repetidamente em contextos específicos do processo de negócio (AZEVEDO *et al.*, 2009).

Através desta transformação, o projetista pode unir atividades do processo de negócio que têm um mesmo objetivo a ser alcançado em uma única atividade no modelo comportamental do sistema computacional em desenvolvimento. Desta forma, diminuindo a complexidade do sistema computacional.

A Figura 5.7 apresenta o modelo de processo de negócio que será utilizado para demonstrar a Transformação de Abstração.

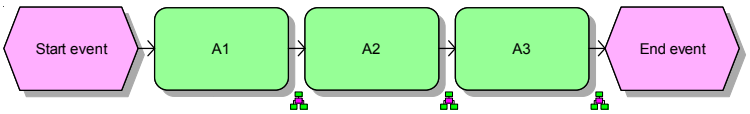


Figura 5.7 – Exemplo de aplicação da Transformação de abstração.

A Tabela 5.1 apresenta as possíveis abstrações que podem ser realizadas com as atividades A1, A2 e A3 da figura acima.

Tabela 5.1 – Abstrações das atividades do Participante A.

Número	Combinação	Resultado
1	A1 e A2	A1A2
2	A2 e A3	A2A3
3	A1, A2 e A3	A1A2A3

Como apresentado na Tabela 5.1, foi possível desenvolver três possíveis componentes utilizando a combinação das atividades e do padrão de processo *sequence.* A Figura 5.8 apresenta um exemplo de processo de negócio após a aplicação da transformação de abstração de atividades utilizando a terceira opção da Tabela 5.1.

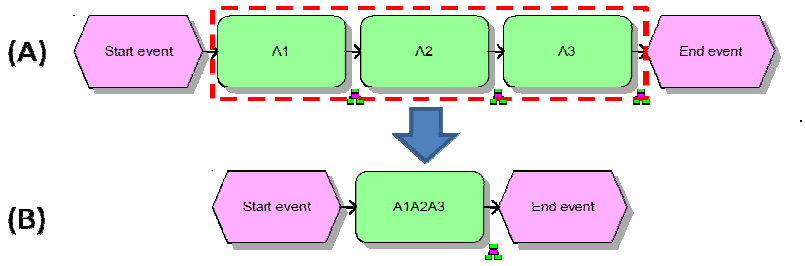


Figura 5.8 - Exemplo da transformação de abstração.

A Transformação de Abstração substituiu as atividades A1, A2 e A3 e em um novo componente chamado A1A2A3 de maior nível de abstração. (Vale notar que a Tabela 5.1 omite a combinação do elemento A1 com o elemento A3; á abstração de A1 e A3 é

inválida pois o novo modelo comportamental não estaria em conformidade com o modelo de processo original, uma vez que a atividade A2 seria eliminada sem ser incorporada à sua antecessora ou sucessora em um novo componente de um nível mais alto de abstração.)

Quando a transformação de abstração é aplicada sobre um conjunto de padrões de processo, a transformação leva à abstração de todo o conjunto de padrões de processo em um novo componente. É importante ressaltar que o projetista somente pode abstrair um ou mais padrões de processo que obedeçam as regras que definem um componente na seção 5.2. O processo de negócio ilustrado na Figura 5.9 será utilizado para exemplificar a Transformação de Abstração com um ou mais padrões de processo.

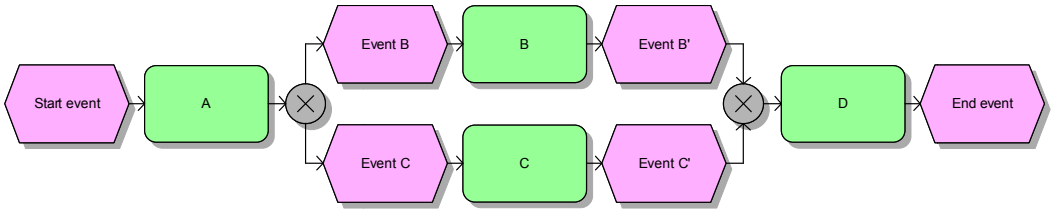


Figura 5.9 – Processo de utilizado para exemplificar abstração de padrões de processo.

Através do conceito de componente o Projetista pode abstrair os padrões de processo *Exclusive Choice*(RUSSELL e HOFSTEDE, 2008) e *Simple Merge* (RUSSELL e HOFSTEDE, 2008), apresentados na Figura 5.10 (i), e transformá-los no componente *Exclusive Choice/Simple Merge* que contempla esses dois padrões de processo, conforme é ilustrado na Figura 5.12 (ii).

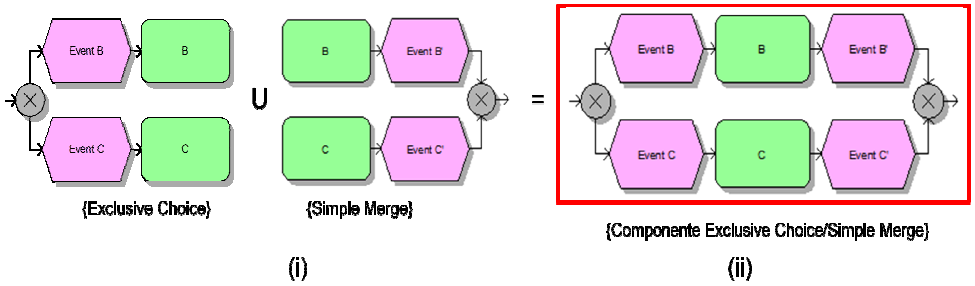


Figura 5.10 – Criando componente a partir de padrões de processo.

É importante observar na Figura 5.10, que somente foi possível abstrair os padrões de processo *Exclusive Choice* e *Simple Merge* e criar o componente *Exclusive Choice/Simple Merge* porque a combinação dos padrões de processo obedecem às

regras para criar um componente: **(i)** possui apenas um ponto de entrada e um ponto de saída e **(ii)** somente existe eventos intermediários dentro do componente. O componente *Exclusive Choice/Simple Merge* foi nomeado como BC no modelo, como demonstrado na Figura 5.11.

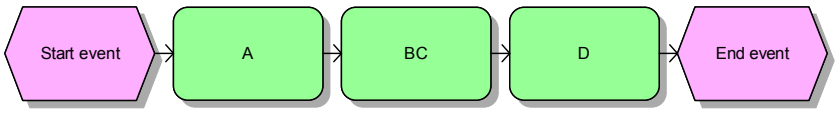


Figura 5.11 – Processo de negócio após a abstração dos padrões Excluise Choice e Simple Merge.

O caso extremo da utilização da transformação de abstração em padrões de processo ocorre quando todo o modelo de processo de negócio é abstraído em apenas um componente, ou seja, todo o modelo de processo de negócio é transformado em um componente de mais alto nível de abstração no modelo comportamental do sistema em realização. A Figura 5.12 apresenta o caso extremo.

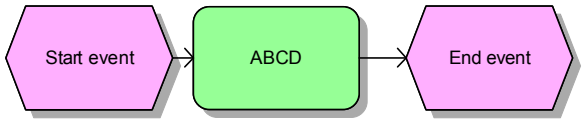


Figura 5.12 – Caso extremo da transformação de abstração em padrões de processo.

O componente ABCD é composto das atividades A e D e do componente BC do modelo apresentado na Figura 5.12. Desta forma, foi possível criar um modelo comportamental para o sistema computacional de mais alto nível de abstração que o modelo de processo apresentado na Figura 5.9.

5.3.1.1 Impactos da Transformação de Abstração sobre a Estrutura Organizacional

As regras da Transformação de Abstração (TA) apresentadas até o momento podem ser aplicadas para atividades que estão sob a responsabilidade de um mesmo participante ou que estão sob a responsabilidade de participantes distintos (cada atividade possui somente um participante responsável) do modelo do processo de negócio utilizado para o desenvolvimento do sistema computacional.

Sempre que a Transformação de Abstração (TA) é aplicada em atividades que estão sob a responsabilidade de participantes distintos é necessário definir uma entidade organizacional (participante) que será responsável pela nova atividade criada pela transformação. Desta forma, é estabelecido um relacionamento *carries out* entre a entidade organizacional e a nova atividade, ou seja, a entidade organizacional escolhida é a responsável "legal" pela execução da atividade. Isso não implica que a entidade organizacional irá executar a nova atividade, mas que é de sua responsabilidade a conclusão da mesma.

É importante ressaltar, que a nova atividade "concentra" as responsabilidades de cada um dos participantes das atividades envolvidas, e devido a isso, ao definir um responsável para essa nova atividade, esse irá assumir as responsabilidades de todos os participantes das atividades abstraídas. Desta forma, a escolha de um responsável para a nova atividade trás um grande impacto sobre a conformidade entre o modelo de processo e modelo organizacional da Arquitetura Organizacional de TI com os modelos de comportamento e organizacional do sistema orientado a processo. Afinal, é muito provável que nenhuma das entidades organizacionais das entidades abstraídas tenha Social Moments para ser responsável pela nova atividade abstraída.

Para solucionar o problema da conformidade, é necessário que o responsável pela nova atividade seja **(i)** uma das entidades organizacionais que eram responsáveis pelas atividades que foram abstraídos; ou **(ii)** uma nova entidade organizacional formadas pelas entidades organizacionais das atividades abstraídas (porque tais entidades tinham um relacionamento de *carries out* com as atividades abstraídas.)

No caso do projetista escolher a opção **(i)**, é importante enfatizar que a entidade organizacional escolhida pode não possuir todos os Social Moments necessários para que a atividade seja concluída com sucesso. Desta forma, é necessário que as outras entidades organizacionais que não foram escolhidas auxiliem a entidade organizacional escolhida para que a nova atividade seja realizada com sucesso. Por exemplo, naFigura 5.13, o projetista escolheu que o responsável pela atividade A3BC (abstração das atividades A3, B e C) será o Participante B, mas o Participante A e Grupo C contribuíram para a conclusão da atividade.

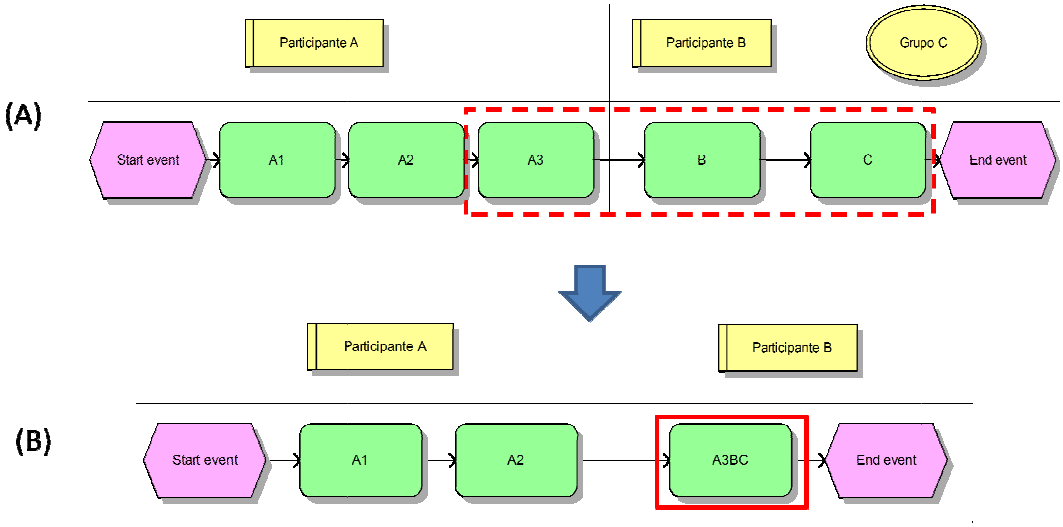


Figura 5.13 – Transformação de abstração em atividades de responsáveis distintos.

Outra característica da Transformação de Abstração quando envolve atividade com responsáveis distintos é a eliminação dos responsáveis sem atividades no modelo comportamental. Isso ocorre quando o responsável não possui nenhum relacionamento *carries out* com alguma atividade do modelo de processo. Por exemplo, ao abstrair a atividade *B,* do *Participante B*, e a atividade *C*, do *Grupo C*, esse último ficou sem nenhuma atividade sob a sua responsabilidade e, assim, o mesmo foi eliminado do modelo comportamental.

Caso o projetista descida pela a opção **(ii)** será necessário criar nova entidade organizacional que será composta ou instanciada pelas entidades organizacionais responsáveis pelas atividades abstraídas. Essa nova entidade organizacional terá os Social Moments necessários para se responsabilizar pela atividade.

No caso do ARIS, a Transformação de Abstração permite que o projetista crie uma das seguintes entidades organizacionais que será responsável pela nova atividade:

i. um novo papel social (*Person Type ou Position Description*): somente poderá ser instanciado por um ou mais participantes responsáveis pelas atividades abstraídas; ou

ii. ou um novo grupo (*Group*): somente pode ser composto por um ou mais participantes que eram responsáveis pelas atividades abstraídas.

Caso o Projetista decida criar um novo papel social (*Person Type ou Position Description*) para o novo componente, é necessário informar o nome do novo papel e quais entidades organizacionais irão instanciar esse papel. Como apresentado no Capítulo 4 o relacionamento de instanciação entre as entidades as organizacionais e o elemento *Person Type* é realizada pelo relacionamento *performs*. A Figura 5.14(A) apresenta o modelo comportamental com o Papel X.

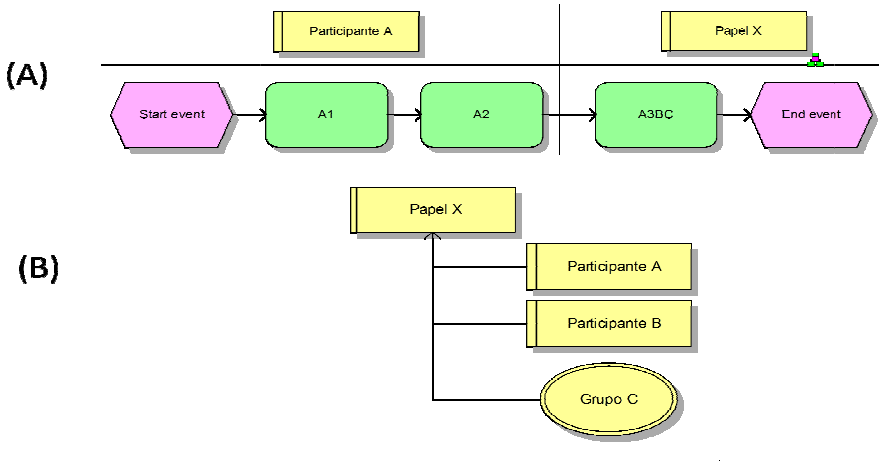


Figura 5.14 – Indicando um papel para nova atividade abstraída.

A Figura 5.14(B) apresenta que os elementos organizacionais *Participante A*, *Participante B* e *Grupo C* podem instanciar o Papel X e, desta forma, todos estes elementos podem ser responsáveis pelo componente A3BC.

Devido a uma limitação da linguagem organizacional do ARIS Method não é possível existir relacionamento *peforms* entre dois *Person Types*, assim, o projetista não pode informar que o novo papel (*Person Type*) é instanciado por um outro papel (*Person Type*) existente. Para resolver tal problema, no caso do ARIS Method, será necessário "transferir" para o novo papel (*Person Type*) todos os elementos (ou somente aqueles que o projetista deseja) que possuem o relacionamento *performs* com o elemento *Person Type* da atividade que participa da transformação abstração. A Figura 5.15 apresenta um exemplo de como o projetista pode escolher os elementos organizacionais de um papel (*Person Type*) que serão "transferidos".

O tipo de problema que ocorre com o relacionamento *performs* é o que torna importante ao projetista da transformação ter o conhecimento profundo das linguagens de modelagem de uma Arquitetura Organizacional de TI, pois sem tal conhecimento erros

dessa natureza passam despercebidos e implicando em problemas nas transformações que visam automatizar modelos das Arquiteturas Organizacionais de TI.

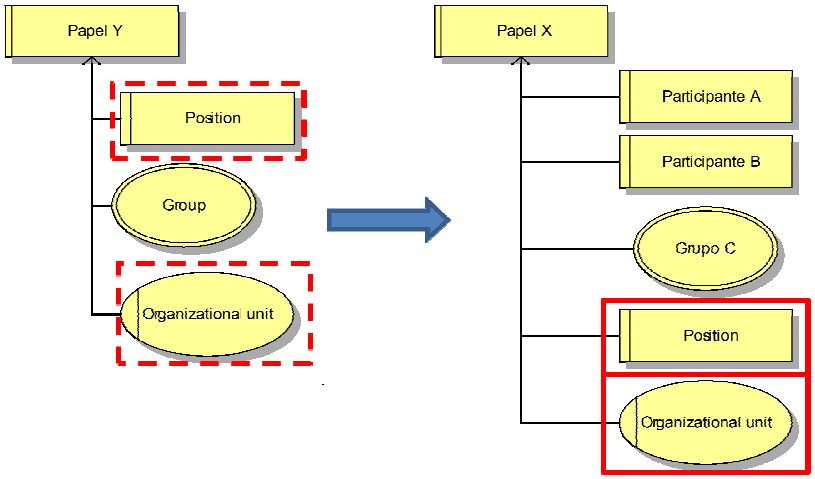


Figura 5.15 - Exemplo de escolha dos elementos organizacionais pertencente a um papel (Person Type) do processo de negócio.

Por fim, caso o Projetista decida criar um novo grupo (*Group*) para a nova atividade, é necessário informar quais entidades organizacionais irão compor esse grupo. A Figura 5.16 apresenta o novo modelo comportamental contendo o novo grupo que será responsável pela atividade A3BC e a Figura 5.17 apresenta a estrutura do novo elemento organizacional.

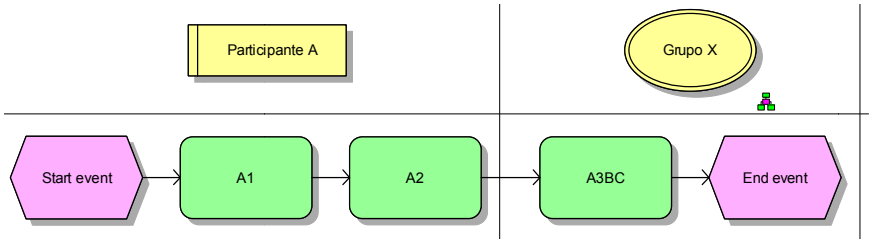


Figura 5.16 - Indicando um grupo para nova atividade abstraída.

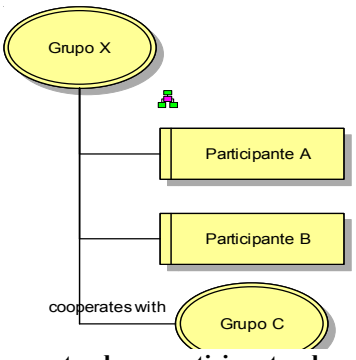


Figura 5.17 – Grupo contendo os participantes das atividades abstraídas

5.3.2 Transformação de Refinamento (TR)

A Transformação de Refinamento permite ao projetista transformar uma atividade do processo de negócio em um componente de processo de negócio. O componente criado pela Transformação de Refinamento é composto de um ou mais padrões de processo.

A Figura 5.18 apresenta exemplo da aplicação da transformação de refinamento sobre uma atividade do processo de negócio.

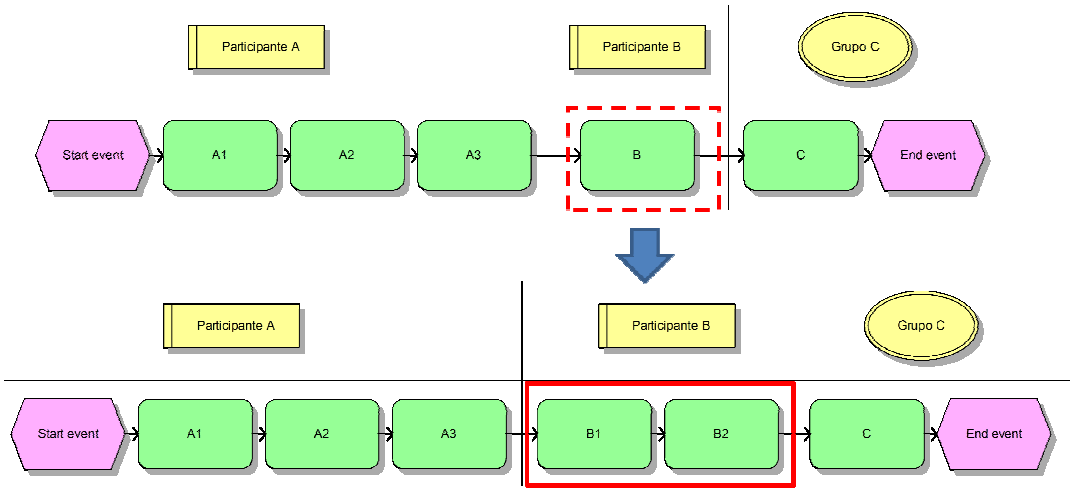


Figura 5.18 – Exemplo de aplicação da transformação refinamento.

A atividade B (dentro do retângulo pontilhado) do processo de negócio, apresentado na Figura 5.18, foi refinada no padrão de processo *sequence* (detalhes em (RUSSELL e HOFSTEDE, 2008)) composto por dois componentes de processo chamados de B1 e B2 (dentro do retângulo continuo). Observe que o padrão *sequence* obedece as regras que define um componente.

Um outro exemplo de aplicação da transformação de refinamento é apresentado na Figura 5.19. Nessa figura a atividade B (dentro do retângulo pontilhado) é refinada em um componente de processo (dentro do retângulo contínuo) que é composto por dois padrões de processo: *Exclusive Choice* (RUSSELL e HOFSTEDE, 2008) e *Simple Merge* (RUSSELL e HOFSTEDE, 2008).

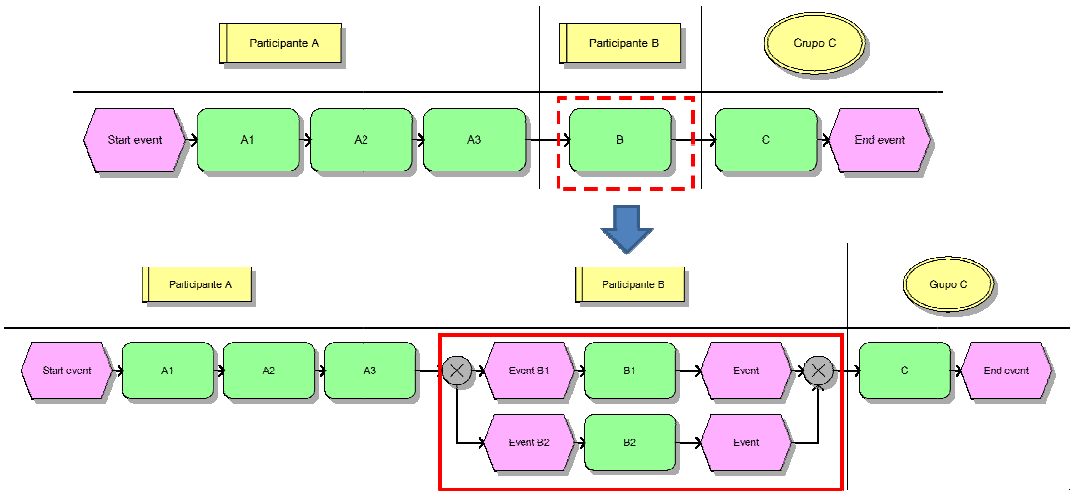


Figura 5.19 – Aplicando a transformação refinamento e criando um componente formado por mais de um padrão de processo.

Observe que o componente de processo formado pela união dos padrões de processo *Exclusive Choice* e *Simple Merge* obedece às regras de um componente.

5.3.2.1 Impacto das Entidades Organizacionais na Transformação de Refinamento

Ao aplicar a Transformação de Refinamento em uma atividade do processo de negócio é necessário que o projetista verifique qual o tipo da entidade organizacional é responsável por tal atividade. Como discutido no Capítulo 4, existem entidade organizacionais compostas de outras entidades organizacionais (por exemplo, unidades organizacionais e grupos) e entidades organizacionais que não são compostas de outras entidades organizacionais (papeis sociais e agentes que não são Function Complex). Para fins didáticos, neste trabalho, as entidades organizacionais compostas de outras entidades organizacionais serão chamadas de Entidades Organizacionais Complexas (EOC) e as demais de Entidades Organizacionais não-complexas (EONC).

Caso a Transformação de Refinamento seja aplicada sobre uma atividade que é da responsabilidade de uma EOC, o projetista necessitará de analisar a relação entre as entidades que compõe a EOC com a atividade a ser refinada. Isso é necessário, pois em muitos casos as entidades que compõe a EOC executam tarefas contidas na atividade a ser refinada.

Portanto, ao aplicar a Transformação de Refinamento em uma atividade que é da responsabilidade de uma EOC será necessário relacionar as entidades da EOC com as novas atividades contidas no componente gerado pela Transformação de Refinamento, e assim adicionando raias no modelo comportamental do sistema computacional. Isso trará um grande impacto para o sistema computacional em desenvolvimento, visto que cada entidade organizacional dentro de uma raia é considerada um ator no sistema computacional.

Para exemplificar essa afirmação acima, imagine que uma determinada atividade *A* do processo de negócio *X* seja da responsabilidade de um grupo *G* (*Group*). O grupo *G* é formado por três papéis sociais e cada uma desses papéis sociais executa uma ou mais tarefas da atividade *A*. Caso seja aplicada a Transformação de Refinamento na atividade *A* o projetista terá que executar os seguintes passos: **(i)** refinar o grupo G, e definir raias para cada um dos integrantes; **(ii)** criar atividades no processo de negócio que represente as tarefas executadas por cada integrante do grupo; e **(iii)** associar os integrantes do grupo com as atividades criada através do relacionamento *carries out*.

Caso a Transformação de Refinamento seja aplicada sobre uma atividade de responsabilidade de uma EONC, a transformação torna-se mais simples. Pois, cada atividade criada pela transformação pertencerá a EONC. Um exemplo de Transformação de Refinamento que envolva uma entidade do tipo EONC é apresentado na Figura 5.19.

Por fim, para simplificar a aplicação da Transformação de Refinamento neste trabalho foi definido que a transformação não tratará a complexidade da entidade organizacional responsável pela atividade, ou seja, a Transformação de Refinamento não irá refinar a entidade organizacional. Portanto, caso um grupo ou unidade organizacional seja

responsável por uma atividade, esses tipos de entidade serão responsáveis por todas as novas atividades criadas pela Transformação de Refinamento.

5.4 TRANSFORMAÇÃO NO NÍVEL DE ATIVIDADE (TDA)

Os modelos de detalhamento de atividades de processo de negócio das Arquiteturas Organizacionais de TI possuem informações importantes que podem ser utilizadas no desenvolvimento do sistema computacionais. Por exemplo, tais modelos possuem informações que descrevem: **(i)** as informações necessárias para a execução de uma atividade; **(ii)** as informações produzidas pela atividade; **(iii)** as regras de negócios da atividades; e **(iv)** as entidades organizacionais que devem ser notificadas sobre a execução da atividade, dentre outras informações.

A Transformação de Detalhamento de Atividade permite que o projetista utilize as informações contidas nos modelos de detalhamento das Arquiteturas Organizacionais de TI para construir modelos de detalhamento de componentes do sistema em realização, conforme apresentado na Figura 5.20. Desta forma, o projetista consegue descrever os detalhes (informações necessárias e produzidas pelo componente, sistemas de suporte entre outras informações) para cada componente do sistema computacional em desenvolvimento.

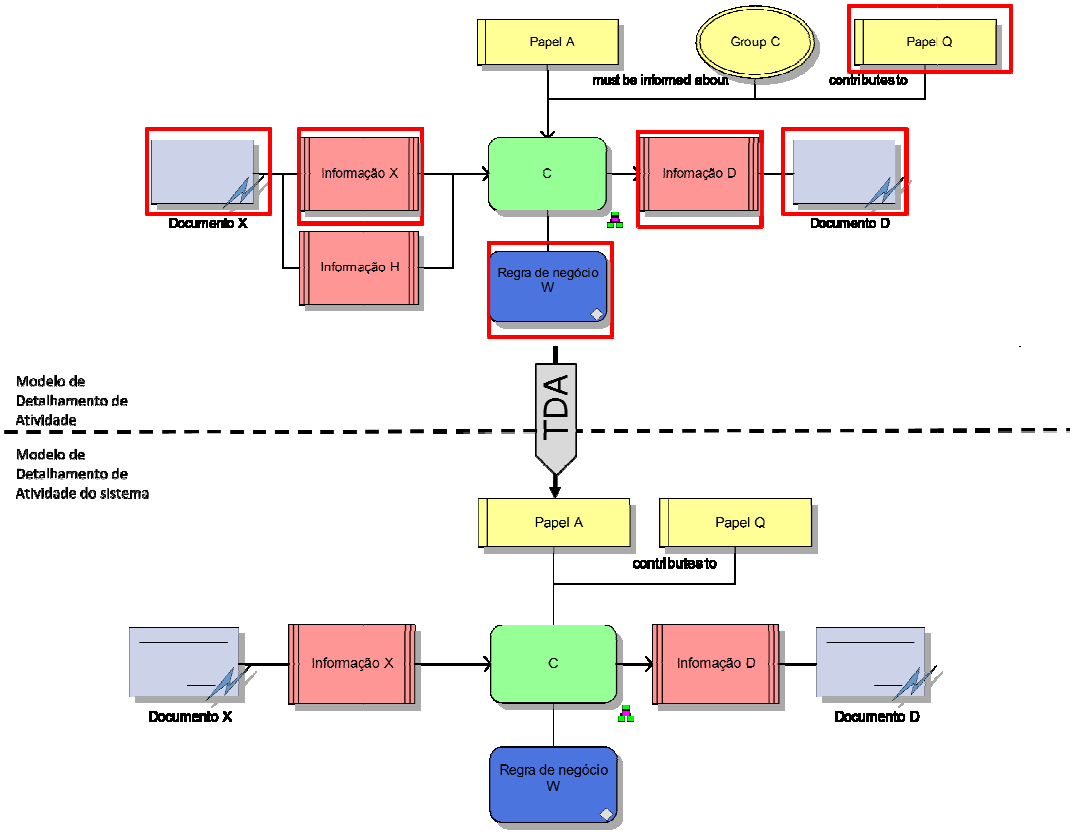


Figura 5.20 – Transformação no nível de componente.

Os retângulos descrevem a escolha feita pelo Projetista através da Transformação de Detalhamento de Atividade, i.e, o Projetista definiu que a atividade C do sistema computacional necessita como informação de entrada a Informação X (*Cluster*) do documento X (*Eletronic Document*) e produz como produto a Informação D (*Cluster*) e o documento Documento D (*Eletronic Document*). Além disto, o projetista também definiu que é o papel Q (*Person Type*) irá contribuir para a conclusão da atividade C e que o sistema computacional deve utilizar as regras de negócio contidas no elemento Regra de Negócio W (*Business Rule*).

5.4.1 Regras para Construção do Modelo de Detalhamento de Atividade do Sistema Computacional

A Transformação de Detalhamento de Atividade (TDA) possui um conjunto de regras para a construção do Modelo de Detalhamento de Atividade do Sistema Computacional. Tais regras garantem que conformidade entre os modelos de Detalhamento de Atividade

da Arquitetura Organizacional de TI seja alcançada e, também, garantem que as decisões de projeto sejam aplicadas sobre.

A primeira regra da Transformação de Detalhamento de Atividade (TDA) permite que o projetista adicione novos elementos no modelo de Detalhamento de Atividade do Sistema Computacional que não estejam presentes no Modelo de Detalhamento de Atividades, e assim, satisfazendo um ou mais decisões de projeto. Afinal, como em outros modelos da Arquitetura Organizacional de TI, os modelos de detalhamento não foram criados com a finalidade de desenvolver um sistemas computacionais em particular.

A segunda regra refere-se à construção dos modelos de Detalhamento de Atividades do Sistema Computacional para as atividades que foram criadas pelas Transformações de Refinamento e de Abstração. Tais atividades não possuem modelos de detalhamento nas Arquiteturas Organizacionais de TI. Portanto, é necessário que o projetista construa os modelos de detalhamento dessas atividades a partir das informações contidas nos modelos de Detalhamento de Atividade das atividades que foram abstraídas ou refinadas. Para fins didáticos, os novos modelos de Detalhamento de Atividades do Sistema Computacional criados a partir da Transformação de Abstração e da Transformação de Refinamento serão, respectivamente, chamados de modelos abstratos e modelos refinados.

A Figura 5.21 apresenta como as informações contidas em outros modelos de detalhamento de atividades da Arquitetura Organizacional de TI podem ser utilizadas para a construção de um modelo detalhado de componente, para um componente que foi criado pela Transformação de Abstração.

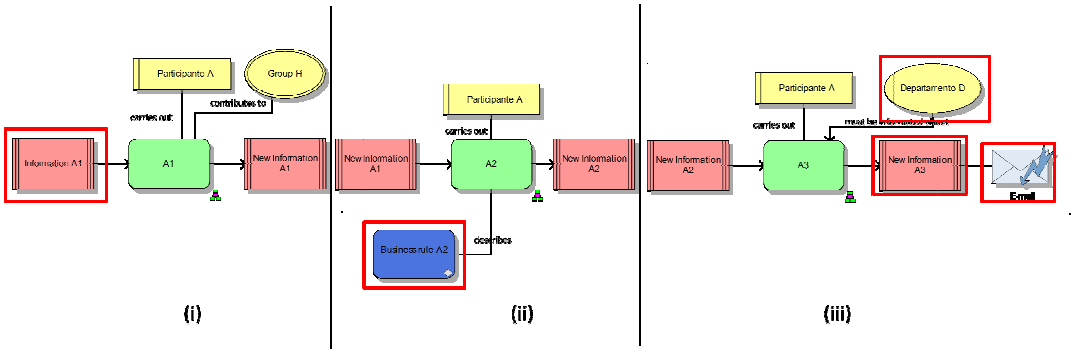


Figura 5.21 – Selecionando os elementos dos modelos de detalhamento de componente

Os retângulos na Figura 5.21 representam as seleções dos elementos que o projetista considerou necessário para a construção do modelo de detalhamento do componente A1A2A3, apresentado na Figura 5.22. O retângulo na Figura 5.22 representa o elemento adicionado no modelo de Detalhamento de Atividade através da primeira regra. O Sistema X auxilia a execução da atividade A1A2A3.

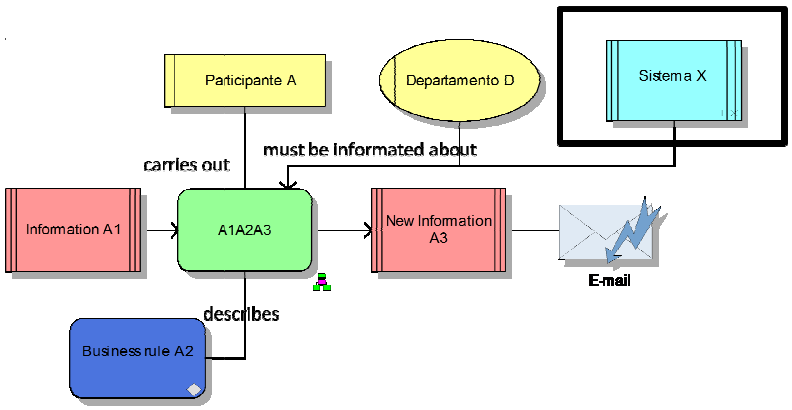


Figura 5.22 – Modelo de Detalhamento de uma atividade utilizando a primeira e segunda regra.

Os passos realizados para a construção do modelo detalhado do componente abstrato são os mesmos para a construção do modelo detalhado do componente refinado.

Para garantir a conformidade entres os modelos criados através da primeira e a segunda regra é necessário que os elementos adicionais e elementos escolhidos não interfiram nas decisões de projeto tomadas em outras transformações de Detalhamento de Atividade de outras atividades e nas Transformações de Abstração e Refinamento aplicadas sobre o processo de negócio no qual a atividade pertence. Por exemplo, não é permitido que o projetista **(i)** adicione um elemento de informação H (*Cluster*) como entrada para uma atividade X se a sua antecessora não produz tal informação e não há

nenhum sistema computacional associado à atividade X que produza a informação H e; **(ii)** selecione uma entidade organizacional como responsável pela atividade X sendo que tal entidade foi eliminada na Transformação de Abstração.

Por fim, vale ressaltar, que a Transformação de Detalhamento de Atividade é aplicada somente nas atividades que compõe o modelo comportamental do sistema computacional. Logo, as atividades que foram consideradas **Totalmente Manuais** na parametrização de automatização não serão considerados na Transformação de Detalhamento de Atividade. Os detalhes sobre a parametrização de automatização são apresentados na seção 5.5.1

5.5 Parametrizações das Transformações

5.5.1 Parametrização de Automatização

A Parametrização de Automatização (PA) permite ao projetista definir se uma atividade é totalmente automatizada, gerenciada e totalmente manual.

O projetista define uma atividade como **totalmente automatizada** quando não há a necessidade da intervenção humana para a execução da atividade, ou seja, a atividade é executado de forma automática pelo sistema computacional. Ao fim da execução da atividade o fluxo do processo é encaminhado para a próxima atividade. Desta forma, por exemplo, o projetista pode definir que a atividade será executado por um serviço ou por um sistema de workflow. A Figura 5.23 apresenta um exemplo de definição de uma atividade como sendo totalmente automatizado.

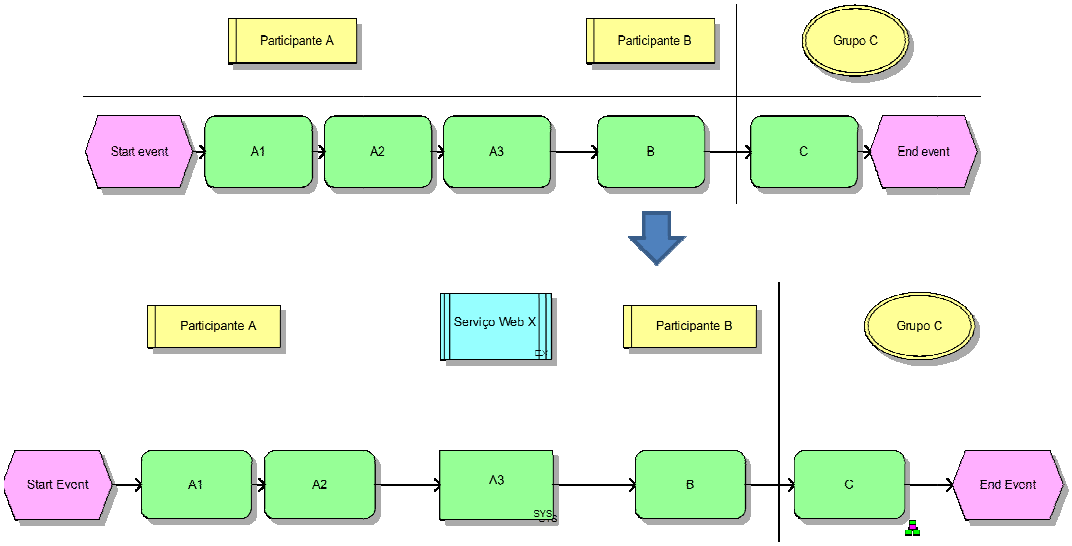


Figura 5.23 –Transformação de automatização: Atividade totalmente automatizada.

Quando uma atividade é totalmente automatizada é necessário que o projetista defina o sistema computacional[6] que será o novo responsável pela conclusão da atividade automatizada. Após essa etapa, será criada uma raia no Modelo Comportamental do Sistema Computacional para representar que o sistema computacional escolhido é o responsável pela atividade automatizada, como demonstrado na Figura 5.23.

Caso o projetista escolha automatizar a atividade de um responsável que não possua nenhuma outra atividade, a não ser a que está sendo automatizada, a transformação exclui a raia do responsável no modelo Comportamental do Sistema Computacional. Como demonstrado na Figura 5.24 , o *Position Participante B* foi excluído do modelo Comportamental do Sistema Computacional e foi criada a raia para o *Application System* "Sistema de Workflow X" como responsável pela atividade automatizada B.

[6] Em relação ao ARIS Method, o Projetista poderia definir que a atividade ou componente é de responsabilidade de um tipo de sistema computacional (*Application System Type*) ou de uma classe de aplicações (*Application System Class*).

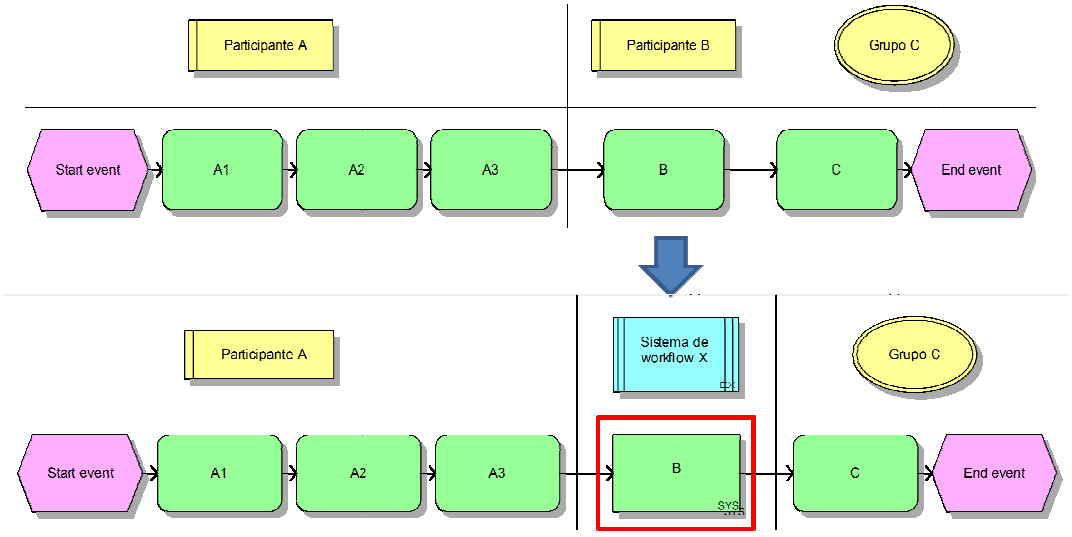


Figura 5.24 – Transformação de automatização: automatização de atividades para responsáveis com mais de uma atividade.

O projetista define uma atividade como **gerenciada** quando há a necessidade de uma ação humana para informar que o atividade foi executado, e assim, o fluxo do processo é encaminhado para o próximo componente do modelo comportamental. A Figura 5.25 apresenta um exemplo da definição do nível acompanhada/gerenciada para uma atividade do processo de negócio.

A atividade B da Figura 5.25 foi definida como gerenciada pelo projetista através da parametrização de automatização. Desta forma, o projetista definiu que a atividade (atividade C) será executada quando o Participante B confirmar no sistema que a atividade foi concluída. Atividade em amarelo é considerada como gerenciada.

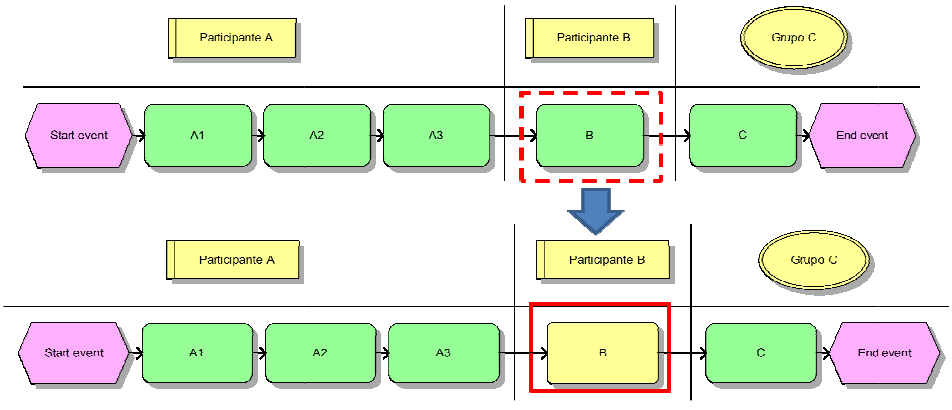


Figura 5.25 - Transformação de automatização nível: Acompanhada/gerenciada.

O projetista define uma atividade como **totalmente manual** quando essa não pertencerá ao sistema computacional em desenvolvimento, ou seja, não é necessário nem mesmo

que a sua execução seja registrada no sistema em desenvolvimento. A Figura 5.26 apresenta um exemplo da definição do nível totalmente manual para uma atividade do processo de negócio. Atividade em branco é considerada como totalmente gerenciada.

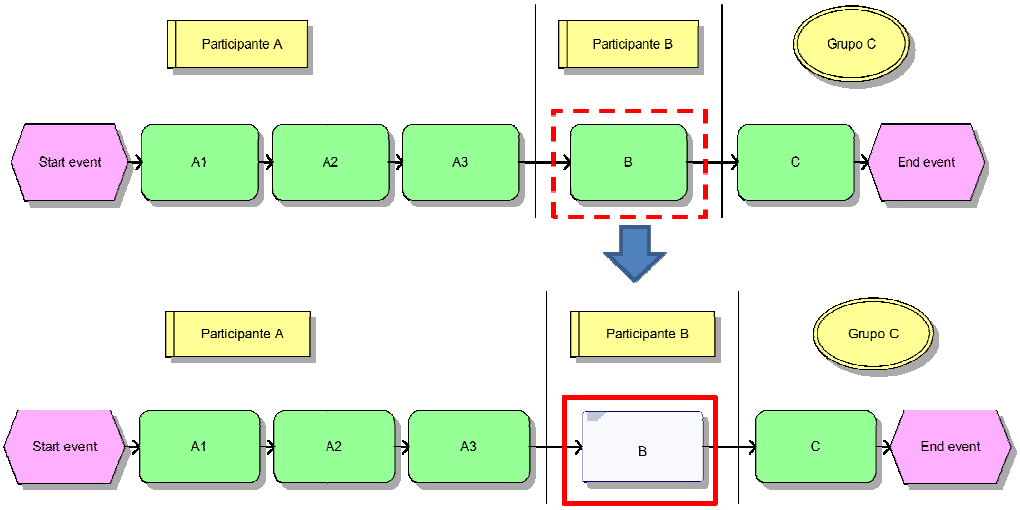


Figura 5.26 – Transformação de automatização nível: Totalmente manual.

Quando uma atividade é definida como totalmente manual o projetista tem que ter em mente que o fluxo das informações do sistema computacional não irá passar por esta atividade e, sim, para a próxima atividade que não seja totalmente manual. A Figura 5.27 demonstra o desvio do fluxo de informação devido a atividade B ser uma totalmente manual.

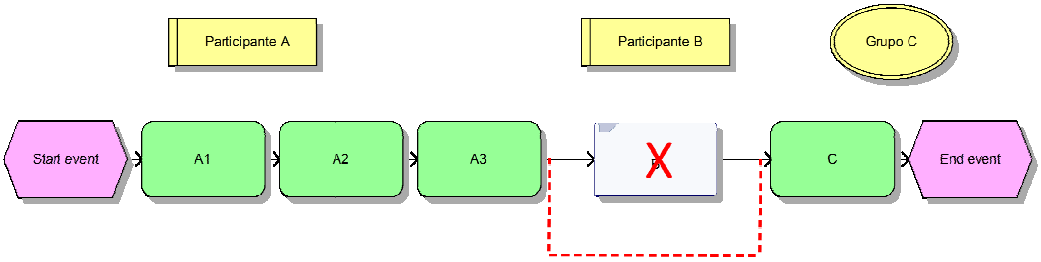


Figura 5.27 - Desvio do fluxo de informação.

Ao definir que uma atividade será totalmente manual, o projetista deve garantir que as informações produzidas nessa atividade estarão no sistema computacional em desenvolvimento, caso tais informações sejam úteis. Desta forma, o projetista deve produzir tais informações nas atividades que precedem a atividade definida como totalmente manual.

5.5.2 Parametrização Organizacional

A Parametrização Organizacional possibilita ao projetista manipular as entidades organizacionais presentes nos modelos organizacionais e assim identificar os atores que estarão presentes no sistema computacional em realização. Cada entidade organizacional manipulada pela parametrização organizacional está associada a um ou mais atividades do modelo comportamental do sistema em realização.

Os modelos organizacionais são utilizados para descrever as entidades organizacionais e seus relacionamentos. Da mesma forma que os demais modelos das Arquiteturas Organizacionais de TI, os modelos organizacionais, na sua maioria, não são criados para serem utilizados em desenvolvimento de software. Desta forma, é necessário que o projetista aplique decisões de projetos sobre tais modelos, para adequá-los ao sistema computacional em desenvolvimento.

Os elementos organizacionais são divididos em duas classes: Papéis Sociais (Social Agent) e Agentes (Human Agent, Social Agent e Artificial Agent). Em relação ao ARIS os papéis sociais são representados pelos elementos *Position*, *Person Type* e *Position Description* e os Agentes são representados por *Organization Unit*, *Organization Unit Type*, *Group*, *External Person* e *Internal Person*.

Para exemplificar a utilização da Parametrização Organizacional imagine que exista um modelo de processo de negócio no qual o papel social Cliente (representado pelo elemento *Person Type*) é um de seus participantes. Conforme apresentado na Figura 5.28, O Cliente é instanciado pela unidade organizacional Recursos Humanos (*Organization Unit*) e pelo Gerente (*Position*) da unidade organizacional Produção (*Organization Unit*).

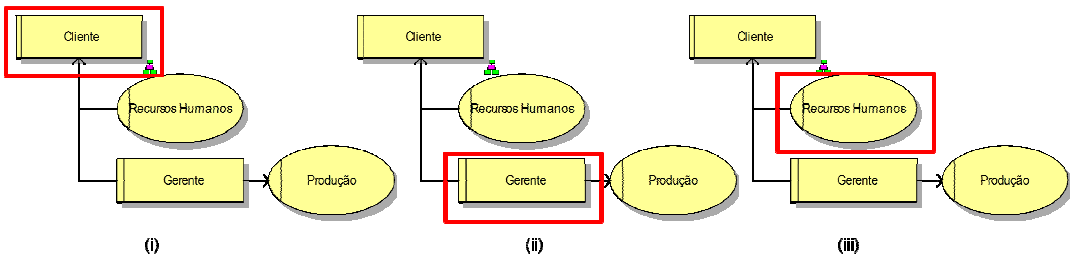


Figura 5.28 - Exemplo de utilização da transformação organizacional com papel social

Através da Transformação Organizacional o projetista pode definir: **(i)** que todos os elementos organizacionais que possuem o relacionamento de *performs* com papel social Cliente no modelo serão atores no sistema, conforme apresentado Figura 5.28 (i); **(ii)** que apenas o Gerente da unidade organizacional de Produção terá permissão de instanciar o papel de Cliente no sistema, conforme apresentado na Figura 5.28 (ii); ou **(iii)** apenas a unidade organizacional de Recursos Humanos irá instanciar o papel de Cliente no sistema, conforme apresentado na Figura 5.28 (iii).

Continuando com a exemplificação, caso o projetista escolha a opção **(iii)** da Figura 5.28, a parametrização organizacional permite ao projetista definir quais entidades organizacionais da unidade organizacional Recursos Humanos irão instanciar o papel Cliente no sistema computacional. Por exemplo, o projetista pode definir que apenas o Analista de Recursos Humanos desta unidade organizacional poderá instanciar o papel Cliente no sistema computacional, como apresentado na Figura 5.29.

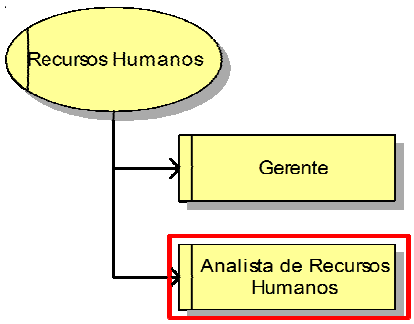


Figura 5.29 - Exemplo de definição de papéis do sistema.

5.6 CONCLUSÃO

Este capítulo apresentou a nova abordagem de Desenvolvimento Orientada a Modelos que utiliza modelos das Arquiteturas Organizacionais de TI para desenvolvimento de sistemas computacionais orientados a processos de negócio.

A primeira característica da abordagem proposta que difere da grande parte das abordagens atuais de Desenvolvimento Orientado a Modelos é a utilização adequada da semântica dos modelos utilizados, afinal parte das abordagens atuais considera apenas os aspectos comportamentais de controle de fluxo em EPCs.

Outra característica fundamental da abordagem apresentada neste capítulo é que todas as transformações pré-definidas na abordagem são parametrizadas. Essa é uma forma de permitir que o projetista aplique as decisões de projeto sobre os modelos utilizados no desenvolvimento do sistema computacional.

A terceira, e última, característica particular desta abordagem é a divisão clara do processo de desenvolvimento em etapas independente e dependente de plataforma. Essa divisão no processo de desenvolvimento possibilita ao projetista separar a decisões de projeto no nível independente e dependente de plataforma, evitando a contaminação dos modelos de processos de negócio com interesses de plataformas de realização.

No nível independente de plataforma o projetista pode produzir um conjunto de modelos que visa retratar diversos aspectos que o sistema computacional deve possuir. No nível dependente de plataforma o projetista utiliza os modelos produzidos no nível independente de plataforma e aplica as transformações para realizar os sistemas computacionais na plataforma tecnológica escolhida.

Em relação a futuros trabalhos para o aperfeiçoamento da abordagem proposta, é vislumbrado o desenvolvimento de transformações que permitam utilizar outros modelos das Arquiteturas Organizacionais de TI como, por exemplo, modelos da UML, modelos que representam regras organizacionais e outros modelos que sejam passíveis de automatização e que venham a contribuir para o desenvolvimento de sistemas computacionais.

Outro trabalho futuro relevante é o desenvolvimento de uma ou mais transformações que permitam o refinamento de entidades organizacionais, i.e., transformações que permitam o refinamento de uma determina entidade organizacional e de forma direta permitam refinar as atividades das qual essas entidades são responsáveis. Neste cenário, a preocupação com a conformidade entre os modelos de origem e destino é ainda mais significativa. Pretende-se então estudar de forma mais sistemática as relações de conformidade, definido-se noções de conformidade rigorosas como apresentado na seção 2.4.3.

Ainda relação aos trabalhos futuros, serão realizados estudos que possibilitem a construção de uma ou mais transformações que permitam ao projetista construir um modelo comportamental do sistema computacional a partir de dois ou mais processos de negócios que estejam interligados por eventos iniciais e finais ou por interfaces de processo, por exemplo.

Por fim, é necessário realizar estudos de transformações que permitam ao projetista reconstruir um sistema computacional, desenvolvido a partir da abordagem proposta neste capítulo, tendo como base as mudanças nos modelos arquiteturais (modelos de processo de negócio, organogramas, modelos de detalhamento de atividades e entre outros modelos) e/ou mudanças nas decisões de projeto. Estas transformações serão relevantes frente ao surgimento de novos requisitos em um processo de manutenção do sistema computacional.

CAPÍTULO 6 PROVA DE CONCEITOS

6.1 INTRODUÇÃO

Esse capítulo apresenta a prova de conceitos da abordagem de Desenvolvimento Orientado a Modelos proposta no Capítulo 5. Para a prova de conceitos será desenvolvido parte de um sistema de workflow utilizando a abordagem proposta neste trabalho.

O sistema de workflow visa automatiza o processo denominado "*Triar pacientes para admissão no setor*", que ocorre em um determinado hospital público da região da Grande Vitória (Vitória, Espírito Santo)(CARDOSO, 2009). De forma sucinta, este processo busca selecionar pacientes com doenças reumatológicas para que os mesmos ingressem no setor de reumatologia do hospital de maneira a receber tratamento adequado para sua enfermidade. Os detalhes sobre esse processo estão apresentados em (CARDOSO, 2009). A Figura 6.1 e a Figura 6.2 apresentam o processo "*Triar pacientes para admissão no setor*".

Obedecendo aos princípios da abordagem proposta de Desenvolvimento Orientado a Modelos o sistema de workflow será dividido em etapas do nível independente e dependente de plataforma.

No nível independente de plataforma, será realizada uma série de transformações que adéqua os modelos produzidos, para descrever o processo de negócio "*Triar pacientes para admissão no setor*", às decisões de projeto tomadas pelo projetista do sistema computacional. Para a seleção e parametrização das transformações desse nível, foi desenvolvido um metamodelo que permite ao projetista construir modelos que descrevam as transformações e suas parametrizações; e em quais modelos as transformações serão aplicadas. A seção 6.2.2 apresenta o metamodelo de seleção e parametrização de transformações do nível independente de plataforma.

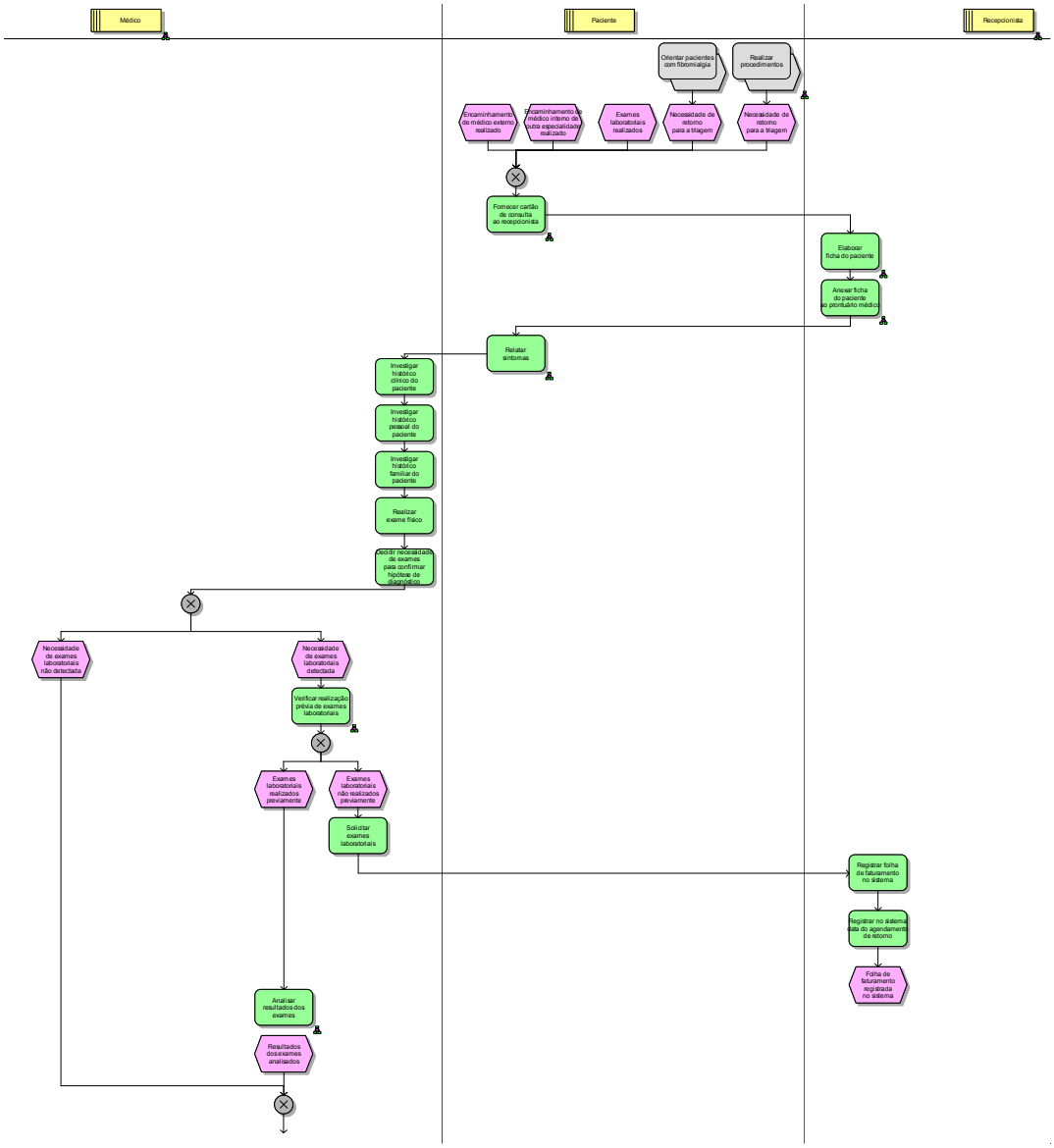


Figura 6.1 – Triar pacientes para admissão no setor (Parte 1).

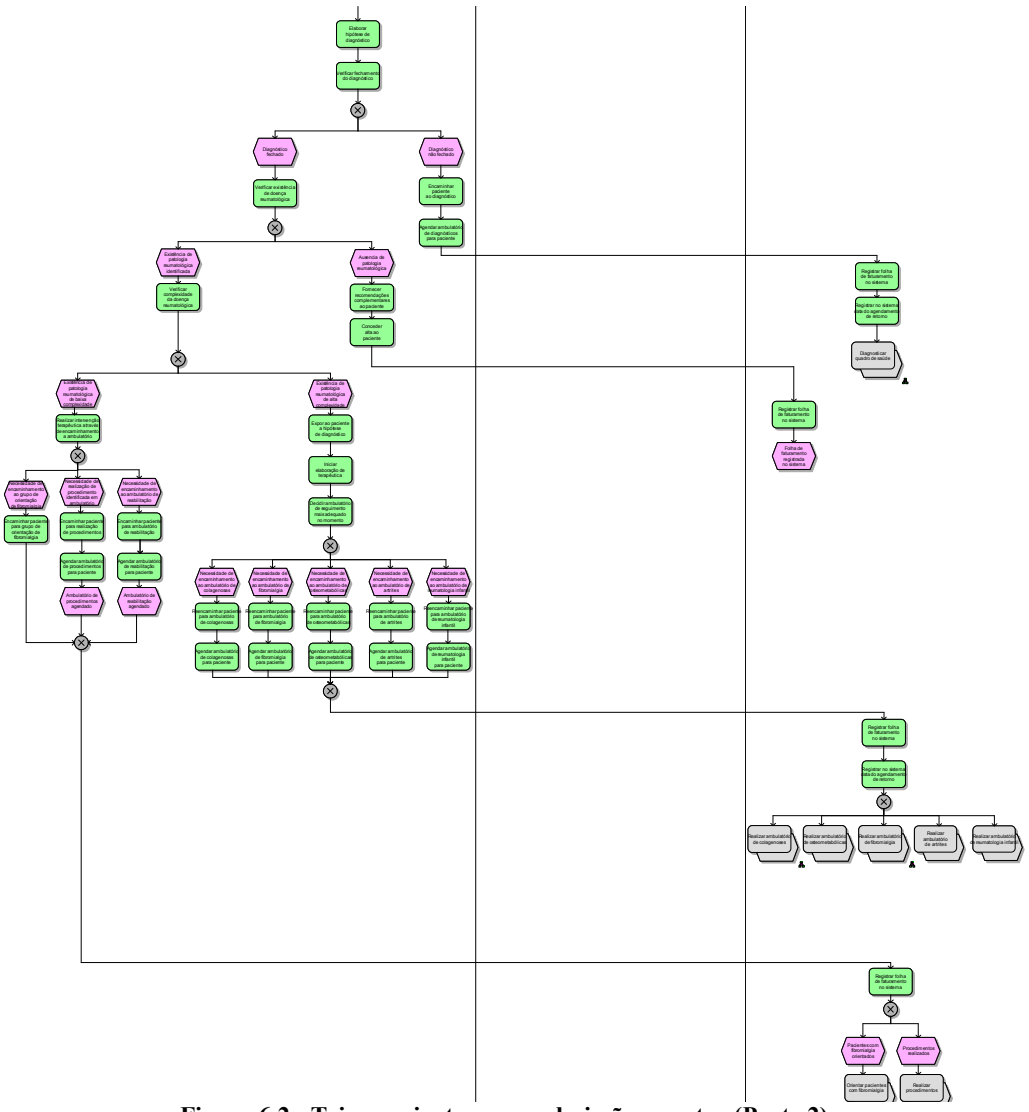


Figura 6.2 - Triar pacientes para admissão no setor (Parte 2).

No nível dependente de plataforma, as transformações transformarão os modelos gerados no nível independente de plataforma em artefatos tecnológicos. Neste trabalho, tais transformações produziram um modelo jPDL. Um modelo jPDL é uma especificação de um processo para a execução no framework JBOSS jBPM (CUMBERLIDGE, 2007). O JBOSS jBPM é um sistema de gerenciado de fluxo de trabalho (workflow). O modelo jPDL é um documento XML que permite expressar tarefas, eventos e outros elementos de um processo de negócio executável.

O capítulo está organizado da seguinte forma: a seção 6.2 apresenta a implementação da abordagem de Desenvolvimento Orientado a Modelos proposta neste trabalho, a seção 6.3 apresenta a implementação das transformações, a seção 6.4 apresenta a aplicação

das transformações no nível independente de plataforma, enquanto a seção 6.5 apresenta a aplicação das transformações no nível dependente de plataforma. A seção 6.6 apresenta uma discussão sobre as etapas manuais que serão necessárias para finalizar o sistema computacional e a seção 6.7 apresenta o sistema computacional criado a partir da abordagem. Por fim, a seção 6.8 apresenta as conclusões do capítulo.

6.2 IMPLEMENTAÇÃO DA ABORDAGEM PROPOSTA DE DESENVOLVIMENTO ORIENTADO A MODELOS

A implementação da abordagem proposta é composta de transformações que estão no nível dependente de plataforma (em cinza na Figura 6.3) e no nível dependente de plataforma (em branco na Figura 6.3).

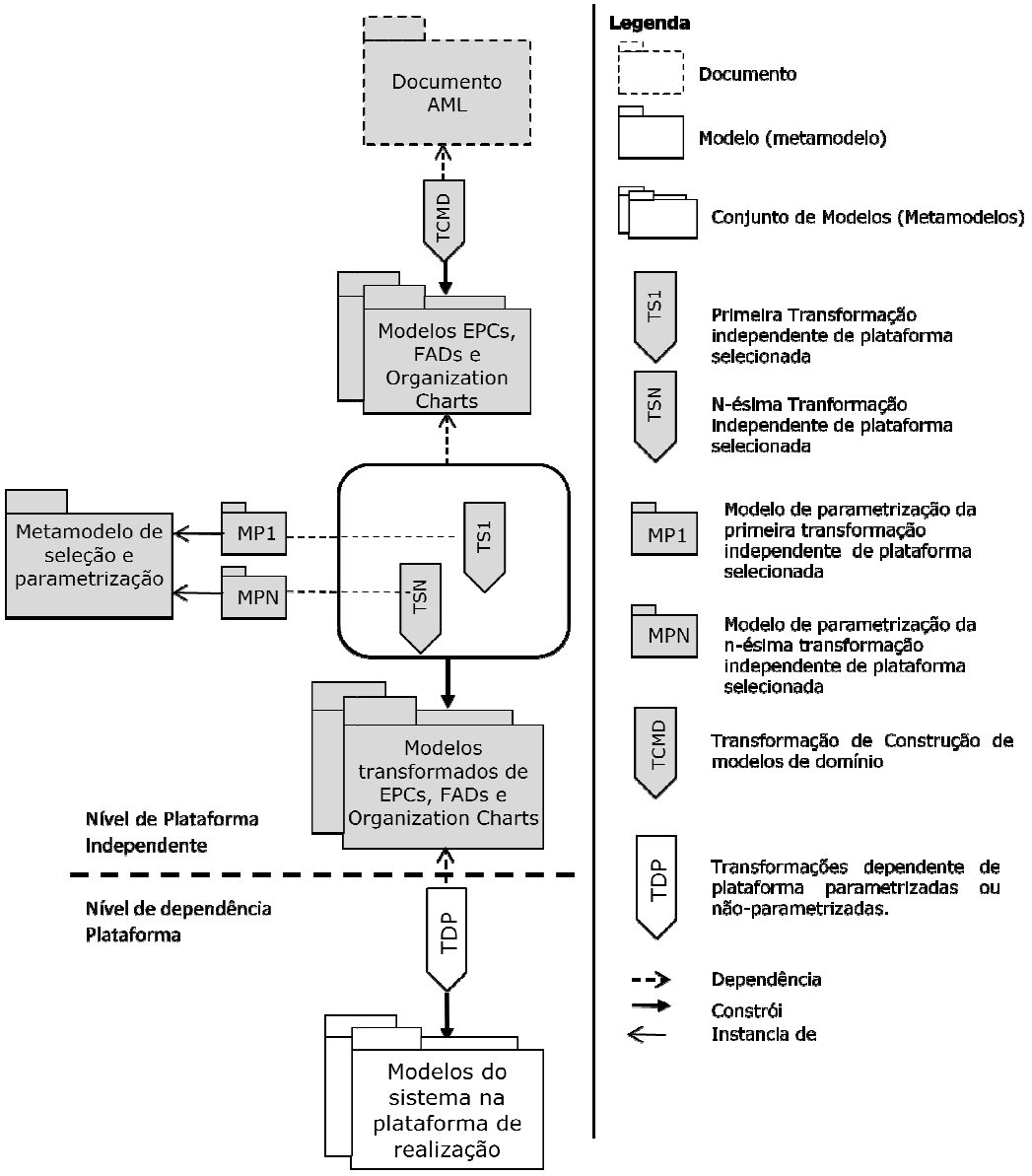


Figura 6.3 – Implementação da Abordagem de Desenvolvimento Orientado a Modelos.

A abordagem inicia com a **Transformação de Construção de Modelos de Domínio** (TCMD). Essa transformação é responsável por construir os modelos do domínio Organizacional, Processo de Negócio e Detalhe de atividade a partir de um documento AML (*ARIS Markup Language*). Um documento AML é um documento XML utilizado para serializar os modelos desenvolvidos dentro da ferramenta de modelagem ARIS Toolset e exportado para que seja utilizado em ferramentas externas. Detalhes sobre o formato AML são apresentado em(SANTOS JR. *et al.*, 2008). A seção 6.2.1 apresenta os detalhes da Transformação de Construção de Modelos de Domínio (TCMD).

Após a construção dos modelos de domínio, o próximo passo é aplicar as decisões de projeto através das Transformações de Abstração, Refinamento e Detalhamento de Atividade apresentadas no Capítulo 5. Desta forma, após a aplicação dessas transformações é gerado o conjunto de modelos que serão usados nas transformações dependentes de plataforma.

Para aplicar as decisões de projeto através das transformações de Abstração, Refinamento e Detalhamento de Atividades foi desenvolvido um metamodelo que permite construir modelos que expressam a seleção e parametrização das transformações no nível independente de plataforma. Por exemplo, o projetista pode desenvolver um modelo que retrate a transformação de abstração "Criando o componente Buscar ficha do paciente" apresentada na seção 6.4.1. Assim, é possível a manipulação dos elementos dos modelos da Arquitetura Organizacional de TI no nível de modelo e de forma automática, sem a necessidade de intervenção manual. A seção 6.2.2 apresenta o metamodelo das transformações independentes de plataforma.

Por fim, após a construção dos modelos independentes de plataforma, são executadas as transformações do nível dependente de plataforma sobre tais modelos. No caso deste trabalho, as transformação do nível dependente de plataforma gera um modelo jPDL a partir das informações contidas nos modelos independentes de plataforma. A seção 6.2.4 apresenta as transformações dependentes de plataforma.

6.2.1 Transformação de Construção de Modelos de Domínio

A Transformação de Construção de Modelos de Domínio é composta das seguintes transformações: **Construir AML Refatorado**, **Construir Modelo Organizacional**, **Construir Modelo de Processo de Negócio** e **Construir Modelo de Detalhamento de Atividades**. A Figura 6.4 apresenta o conjunto de transformações que são parte da Transformação de Construção de Modelos de Domínio.

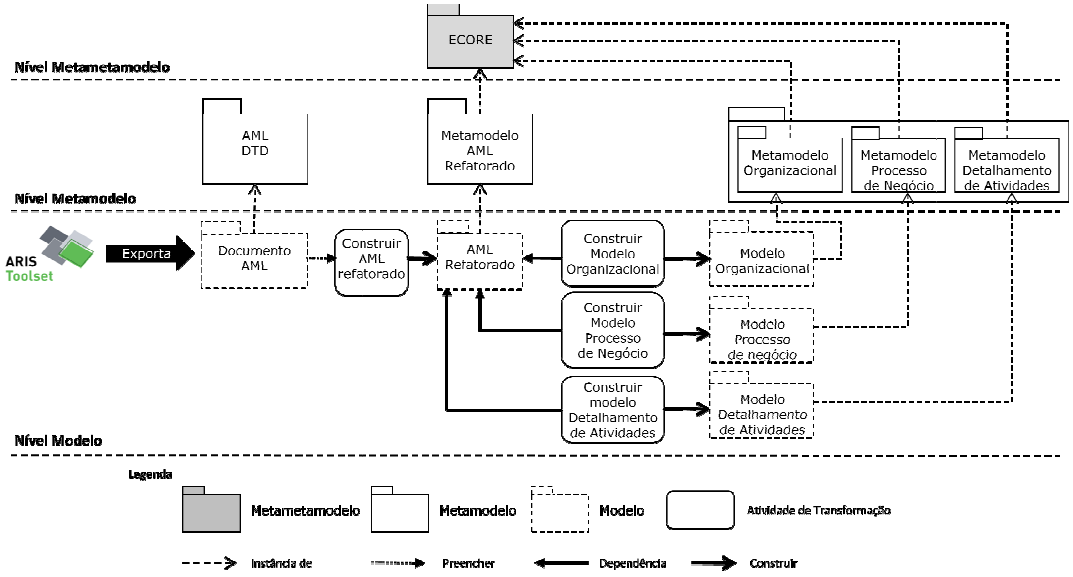


Figura 6.4 – Conjunto de transformações para criar os modelos de domínio organizacional, processo de negócio e detalhamento de atividades do ARIS Method.

A transformação **Construir AML Refatorado** é responsável por criar uma instância do metamodelo AML Refatorado[7] a partir de informações de um documento AML exportado da ferramenta ARIS Toolset. A transformação Construir AML Refatorado é uma transformação entre espaços tecnológicos, porque transforma um documento XML em um modelo ECORE preservando o isomorfismo entre documento de origem e modelo de destino. Apesar do modelo AML Refatorado conter informações sobre os modelos desenvolvidos na ferramenta ARIS Toolset, a manipulação dessas informações é onerosa pois não permite uma distinção clara das entidades contidas nos modelos de domínios do ARIS Method. A distinção entre as entidades de domínio é realizada através de codificações dos valores do atributo de cada classe presente no AML Refatorado. Por exemplo, para identificar se um determinado modelo é do tipo organizacional é necessário verificar se o valor do atributo *Type* da classe *Model* é MT_ORG_CHT. Para que a manipulação das informações seja menos onerosa, é necessário que tal procedimento seja feito no nível de modelo, ou seja, é necessária a construção dos modelos de domínio de processo de negócio, organizacional e detalhamento de atividade a partir das informações contidas no documento AML.

[7] O Metamodelo AML Refatorado é um metamodelo que descreve as entidades e os relacionamentos que estão presentes em um documento AML (SANTOS JR. *et al.*, 2008)..

As transformações **Construir Modelo Organizacional**, **Construir Modelo de Processo de Negócio** e **Construir Modelo de Detalhamento de Atividades** têm por objetivo, respectivamente, criar instâncias de modelos organizacionais, processo de negócio e detalhamento de atividades através de informações contidas no modelo AML Refatorado. A partir dos modelos gerados por essas transformações é possível a manipulação das informações dos modelos utilizando conceitos do domínio de maneira mais direta do que realizar a mesma operação sobre um documento AML. Em (PIUMBINI, 2009) é apresentado em detalhes a implementação do conjunto de transformações que forma a Transformação de Construção de Modelos de Domínio (TCMD).

6.2.2 Metamodelo de Seleção e Parametrização das Transformações Independentes de Plataforma

O Metamodelo das Transformações Parametrizadas Independentes de Plataforma permite que o Projetista crie modelos para representar as transformações de Abstração, Refinamento e Detalhamento de Componentes apresentadas no Capítulo 5. A Figura 6.5 apresenta o Metamodelo das Transformações Parametrizadas Independentes de Plataforma.

A metaclasse TransformationModel é utilizada para representar um *container*, ou seja, essa metaclasse é utilizada para conter as metaclasses que representam as transformações.

A metaclasse abstrata Transformation representa uma transformação parametrizada que será utilizada para aplicar as decisões de projeto sobre algum modelo de domínio da Arquitetura Organizacional de TI. O meta-atributo name e o meta-atributo modelDomain são utilizados para especificar um nome para a transformação e o nome do modelo sobre o qual a transformação será aplicada. Por exemplo, o projetista pode nomear a transformação de "transformação de Abstração em X" atribuindo um valor ao parâmetro name e informar que a transformação será aplicada sobre o modelo X atribuindo um valor no atributo modelDomain.

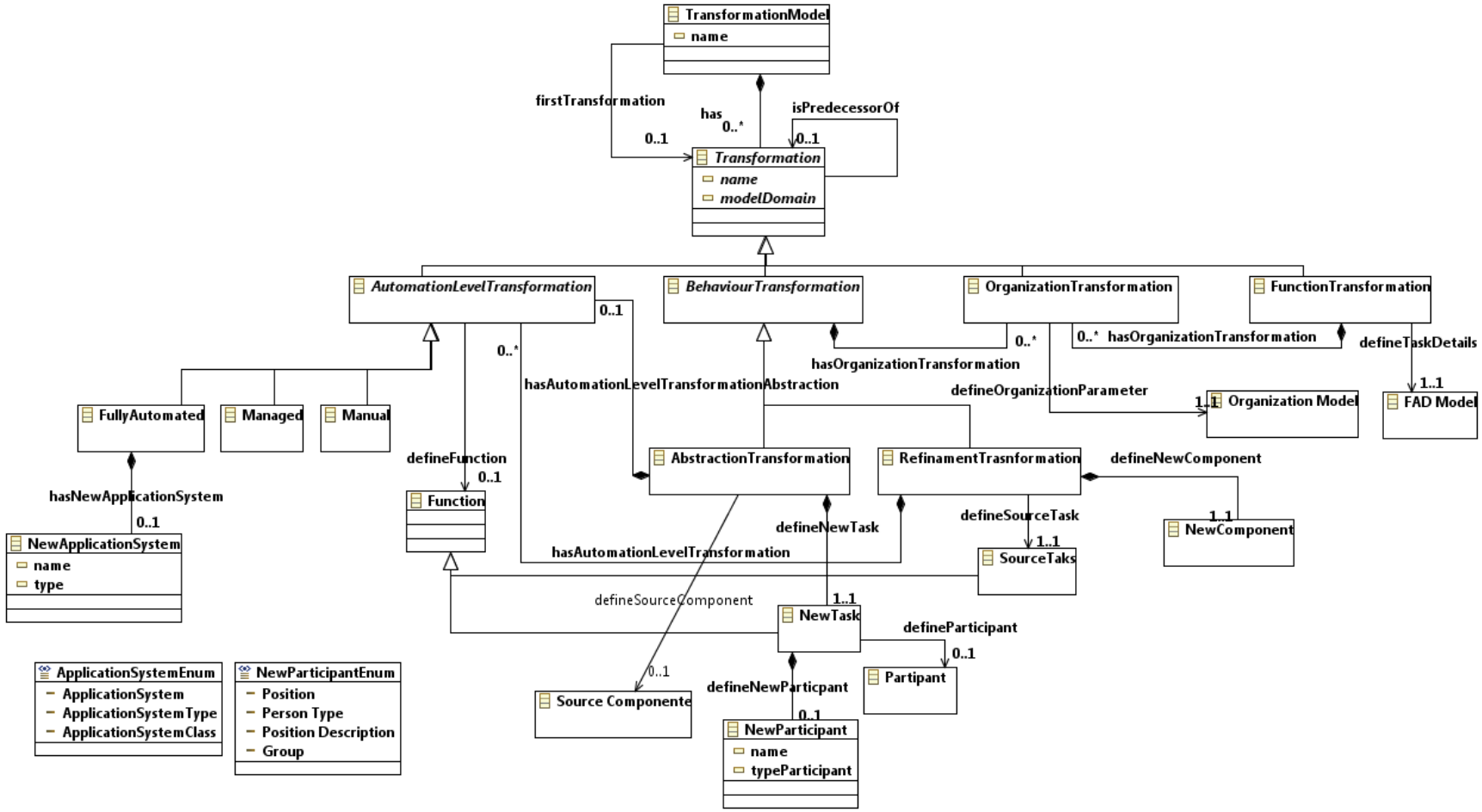


Figura 6.5 – Metamodelo de Seleção e Parametrização de Transformações Independentes de Plataforma.

O meta-relacionamento firstTransformation que ocorre entre Transformation Model e Transformation é utilizada para representar qual a primeira transformação será executada de um conjunto de transformações.

O meta-auto-relacionamento is predecessor of é utilizada para representar a sequência de transformações que serão executas durante o processo de construção dos modelos independentes de plataforma.

As metaclasses AbstractionTransformation, RefinementTransformation e FunctionTransformation representam as Transformações de Abstração, Refinamento e Detalhamento de Atividade, respectivamente.

Com relação a metaclasse AbstractionTransformation, o meta-relacionamento defineNewTask é utilizado para representar a nova atividade de alto nível (NewTask) que será criada após aplicar a transformação de abstração sobre um componente de processo (SourceComponent). A metaclasse NewParticipant, apresentada na, Figura 6.5, representa o novo participante que pode ser criado pela transformação de abstração e a metaclasse Participant é utilizada para representar um participante do processo de negócio que será o responsável pela nova atividade abstrata.

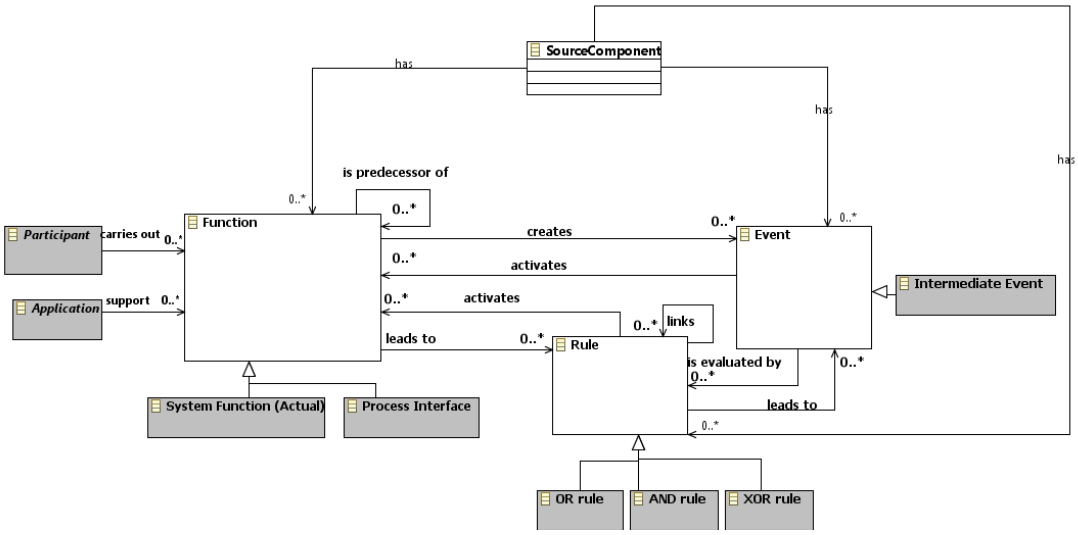


Figura 6.6 – Metamodelo da metaclasse SourceComponent.

A metaclasses abstrata AutomationLevelTransformation é utilizada para representar os tipos de transformações de automatização. O meta-relacionamento defineFunction especifica a atividade (instância de *Function*) que será sujeita a um nível de automação específico através desta transformação.

As especializações FullyAutomated, Managed e Manual desta metaclasse representam, respectivamente, os seguintes níveis de automatização: Totalmente Automatizada, Gerenciada e Totalmente Manual.

A metaclasse SourceTask, associada à metaclasse RefinementTransformation pelo relacionamento defineSourceTask, representa a atividade do processo de negócio que será refinada. A metaclasse NewComponent, apresentada na Figura 6.7, representa o componente que será criado pela transformação de refinamento.

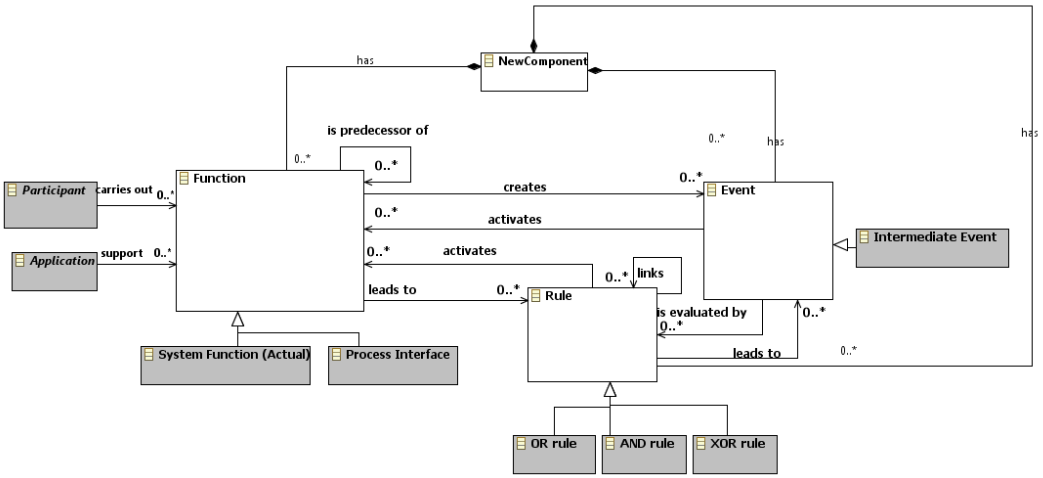


Figura 6.7 – Metamodelo da metaclasse NewComponent.

As metaclasses OrganizationModel e FADModel representam subconjuntos do metamodelo Organizacional e do metamodelo de Detalhamento de Atividade do ARIS Method apresentado nas seções 3.2 e 0, respectivamente. Esses subconjuntos são formados por apenas elementos e relacionamentos que tiveram a sua semântica definida na análise ontológica do Capítulo 4. Por exemplo, no caso do metamodelo Organizacional, o relacionamento *performs* entre *Person Type* e *Location* não está presente no subconjunto representado pela metaclasse OrganizationModel.

6.2.3 Construtor de Modelos de Seleção e Parametrização de Transformações Independentes de Plataforma

O protótipo inclui um *plugin* na plataforma Eclipse (ECLIPSE, 2008b) que permite ao projetista desenvolver modelos que selecionem e parametrizem as transformações independentes de plataforma. O *plugin* permite desenvolver os seguintes modelos: processo de negócio, organizacional e detalhamento *de* atividades do ARIS Method e os modelos de seleção e parametrização de transformações.

Para o desenvolvimento do *plugin*, foi necessário utilizar os metamodelos das linguagens de domínio do ARIS Method (apresentados no Capítulo 3) e os metamodelos de seleção e parametrização de transformações independentes de plataforma (apresentados na seção 6.2.2). Não foram considerados no desenvolvimento do plugin os relacionamentos das linguagens de domínio do ARIS Method que foram considerados desnecessários na análise ontológica, realizada no Capítulo 4.

A Figura 6.8 apresenta um exemplo de modelo de seleção e parametrização construído através do *plugin*.

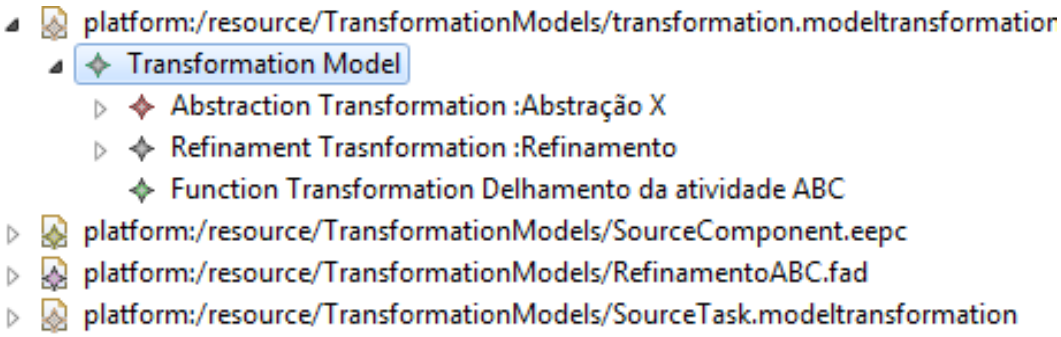


Figura 6.8 – Plugin para desenvolvimento de modelos de seleção e parametrização de transformações independentes de plataforma.

Como demonstrado na Figura 6.8, o projetista selecionou e parametrizou três transformações: **(i)** uma Transformação de Abstração, **(ii)** uma Transformação de Refinamento e **(iii)** uma Transformação de Detalhamento de Atividades.

A Transformação de Abstração, chamada de "Abstração X" cria uma nova atividade (NewTask) chamada de "Task X" e totalmente automatizada (FullyAutomated). O

responsável por essa atividade é o “Sistema X”. A Figura 6.9 apresenta a Transformação de Abstração “Abstração X” e a parametrização de automatização.

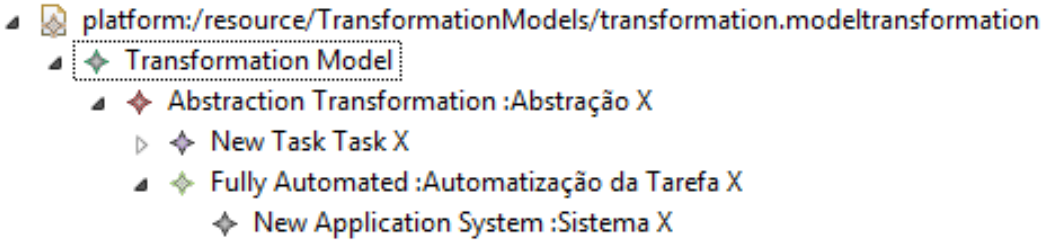


Figura 6.9 – Transformação de Abstração modelado no plugin.

Para indicar sobre qual componente a transformação será aplicada, é necessário que o projetista informe o modelo que representa o SourceComponent no campo “Define Source Component”, conforme demonstrado na Figura 6.10 .

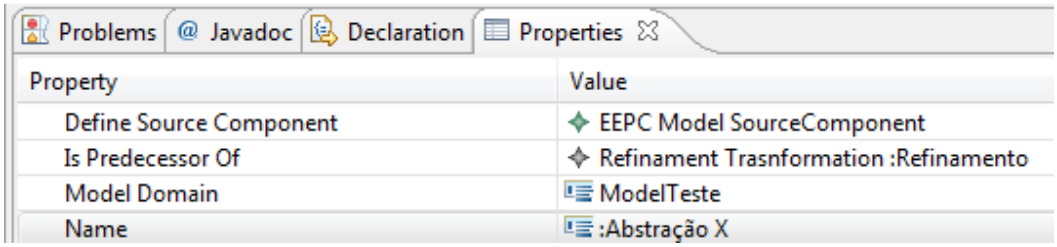


Figura 6.10 – Definindo o Source Component.

No caso do exemplo deste trabalho, o modelo que representa o Source Component é chamado de SourceComponent.eepc. Esse modelo contém duas atividades do processo de negócio: Atividade A e Atividade B. A Figura 6.11 apresenta o modelo do Source Component.

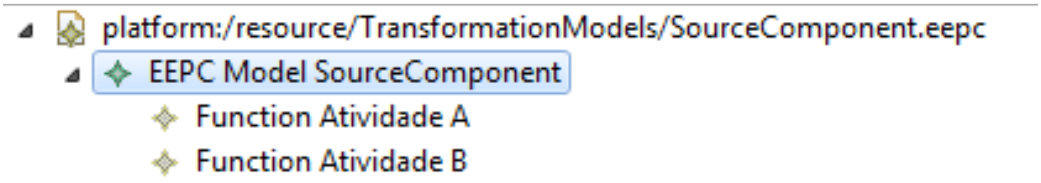


Figura 6.11 - Modelo SourceComponent.eepc construído no plugin.

A segunda transformação modelada pelo projetista é uma Transformação de Refinamento chamada de “Refinamento”. Como demonstrado na Figura 6.12, a Transformação de Refinamento modelada irá construir duas atividades automatizadas

(atividade Y e atividade Z) e uma gerenciada (atividade W). A atividade Y e Z serão da responsabilidade dos sistemas computacionais Sistema Y e Sistema Z, respectivamente.

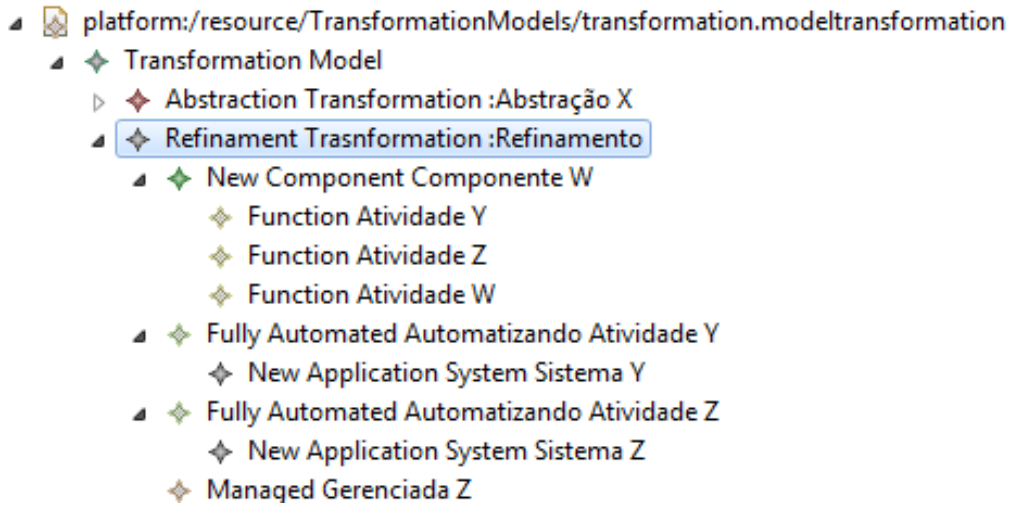


Figura 6.12 – Transformação de Refinamento modelado no plugin.

Para indicar sobre qual atividade a Transformação de Refinamento será aplicada é necessário que o projetista informe o nome da atividade no campo "Define Source Task". Como demonstrado na Figura 6.13, a Transformação de Refinamento será aplicada sobre a "Atividade X" do modelo de processo de negócio.

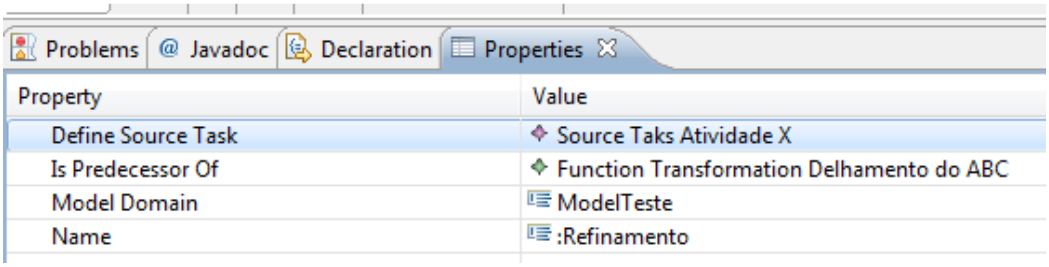


Figura 6.13 – Definindo o Source Task.

Como apresentado na Figura 6.8, a terceira transformação modelada pelo projetista é uma Transformação de Detalhamento de Atividade chamada de "Detalhamento da Atividade ABC". Para indicar quais elementos são necessários para uma atividade é necessário que o projetista crie um modelo de FAD com os tais. Na Figura 6.14. o projetista informou que para o sistema computacional somente são necessários os seguintes elementos: Notificação X (*E-mail - Information Carrier*) e Informação H (*Cluster*).

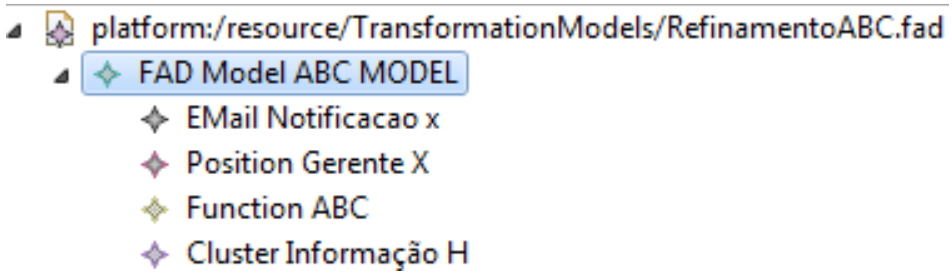


Figura 6.14 – Modelo listando os elemento necessários para a atividade ABC.

O campo "Define FAD Model" da Transformação de Detalhamento de Atividade associa o modelo FAD com a transformação, conforme demonstrado na Figura 6.15.

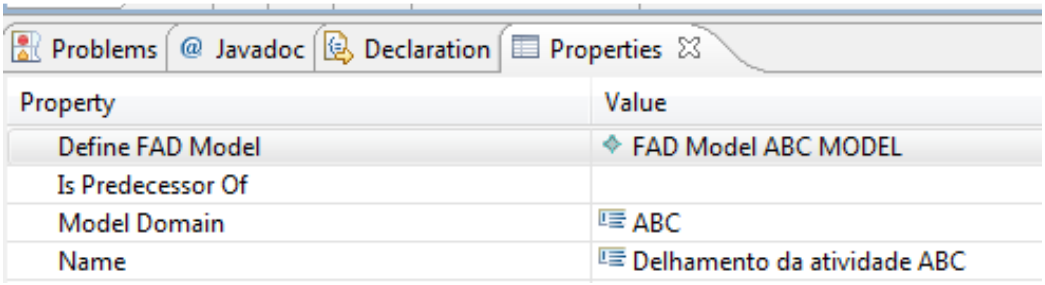


Figura 6.15 – Definindo o Modelo FAD na Transformação de Detalhamento de Atividade.

O *plugin* permite que o projetista defina a ordem na qual as transformações serão executadas. Primeiro é necessário definir a primeira transformação através do campo "First Transformation" do elemento Transformation Model. Como demonstrado na Figura 6.16, o projetista definiu que a primeira transformação a ser executada será a "Abstração X"

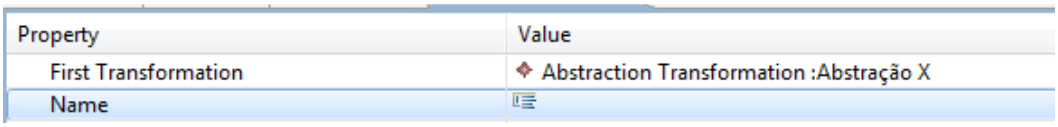


Figura 6.16 – Definindo a primeira transformação a ser executada.

Por fim, o segundo, e último passo, é a definição em cada uma das transformações sobre qual será a sua sucessora. Para isso, o projetista informa no campo "Is Predecessor Of" qual transformação será a sucessora da transformação atual. Por exemplo, como demonstrado na Figura 6.17, a sucessora da transformação Abstração X será a transformação Refinamento.

Define Source Component	EEPC Model SourceComponent
Is Predecessor Of	Refinament Trasnformation :Refinamento
Model Domain	ModelTeste
Name	:Abstração X

Figura 6.17 – Definindo a transformação sucessora.

6.2.4 Transformações Dependentes de Plataforma

Neste trabalho as transformações dependentes de plataforma são responsáveis por construir o modelo jPDL do sistema de workflow. Essa construção será feita através de transformações que utilizam as informações contidas nos modelos gerados pelas transformações do nível independente de plataforma. A Figura 6.18 apresenta as transformações utilizadas para o desenvolvimento do modelo jPDL.

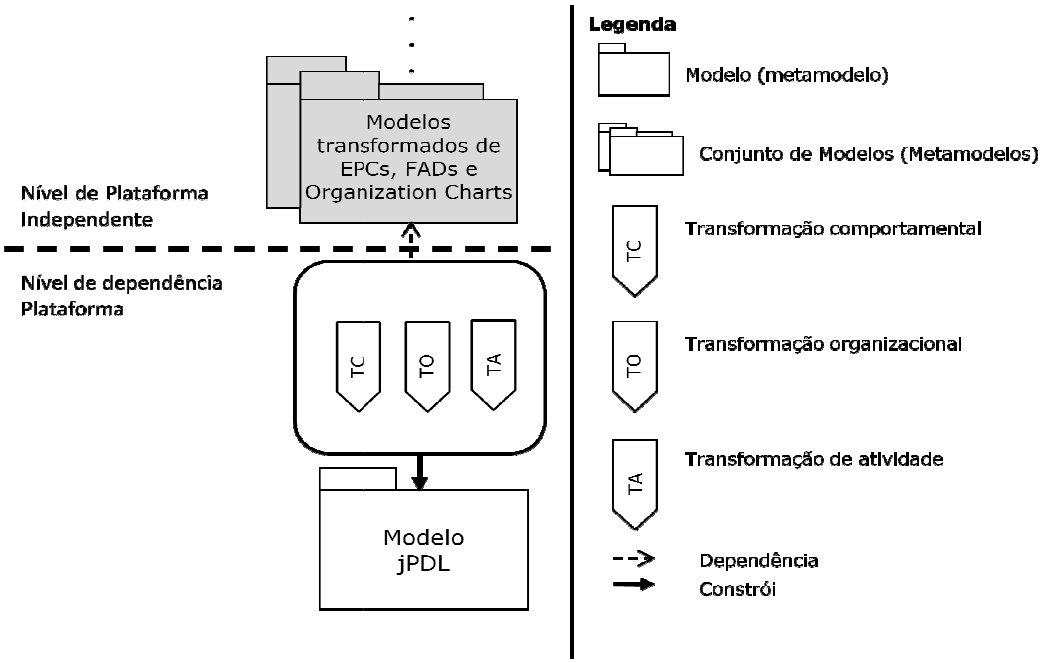


Figura 6.18 – Transformações no nível dependente de plataforma deste trabalho.

A Transformação Comportamental (TC) é responsável por definir no modelo jPDL as atividades (*Task* ou *Java*), as relações (*Transition*) entre as atividades e os principais atores (*Swimlanes*) do processo de negócio. Isso é feito através da análise das informações contidas no modelo EPC gerado pelas transformações do nível independente de plataforma. Por exemplo, através da identificação do elemento *Function* (e seus elementos notacionais) é possível identificar uma atividade (Task ou Java) que estará presente no modelo jPDL, como demonstrado na seção 6.5.1.

A Transformação de Atividade (TA) utiliza as informações contidas nos modelos de detalhamento de atividade para complementar o modelo jPDL construído pela Transformação Comportamental (TC). Por exemplo, esta transformação pode utilizar o relacionamento *must be informed about* para criar atividade de envio de e-mail no jPDL, conforme demonstrado na seção 6.5.2.

Por fim, a Transformação Organizacional (TO) utiliza as informações contidas nos modelos organizacionais gerados pelas transformações do nível independente de plataforma para complementar o modelo jPDL. Por exemplo, através dos modelos organizacionais criados no nível independente de plataforma, a transformação pode: **(i)** criar atividades de notificação para usuários do sistema no modelo jPDL, ao identificar o relacionamento *must be informed about*; ou **(ii)** adicionar os atores no `swimlanes` ao identificar o relacionamento *performs.*

6.3 IMPLEMENTAÇÃO DAS TRANSFORMAÇÕES

Para automatizar a abordagem proposta na Figura 6.3 foi desenvolvida uma aplicação em Java que permite o projetista produza um modelo jPDL a partir de um documento AML e de um conjunto de modelos de seleção e parametrização de transformações. A Figura 6.19 apresenta a arquitetura utilizada para desenvolver a aplicação que automatiza a abordagem proposta na Figura 6.3.

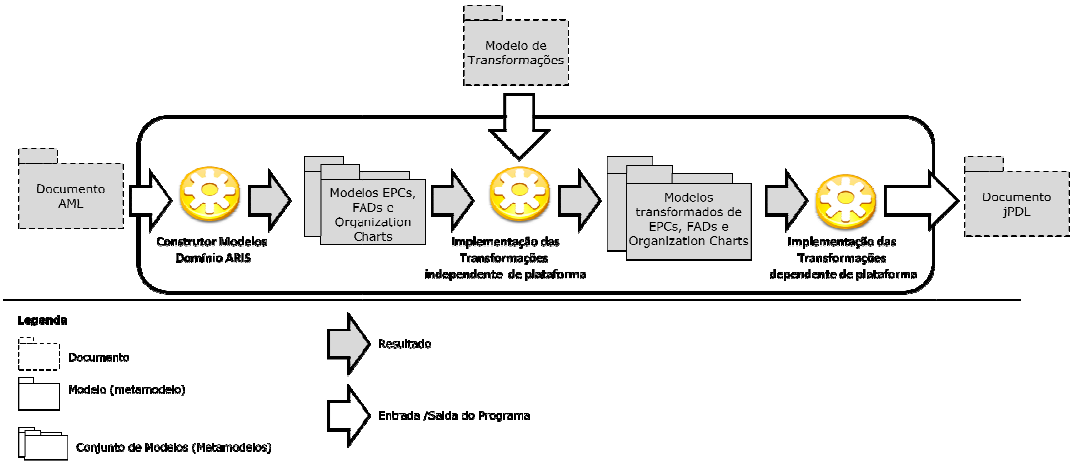


Figura 6.19 – Arquitetura de Automatização das Transformações.

A aplicação desenvolvida é composta por três implementações:

i. **Construtor de Modelos de Domínio ARIS:** é o responsável por transformar o documento AML, exportado da ferramenta de modelagem ARIS Toolset, em modelos de domínio do ARIS Method. Para o desenvolvimento dessa API foi utilizado a abordagem apresentada na seção 6.2.1.

ii. **Implementação das Transformações Independente de Plataforma:** é o responsável por aplicar as transformações descritas nos Modelos de Transformação sobre os modelos de domínio do ARIS Method. Após aplicar todas as transformações descritas nos Modelos de Transformações, a Implementação das Transformações Independente de Plataforma gera um conjunto de modelos transformados. Para o implementação das transformações independentes de plataforma foram utilizados os metamodelos de domínio do ARIS Method apresentados no Capítulo 3 e os metamodelo de transformações apresentado na seção 6.2.2.

iii. **Implementação das Transformações Dependente de Plataforma:** é o responsável por transformar o conjunto de modelos transformados, produzido pelas transformações independentes de plataforma, em um documento jPDL. Para isso, foram implementadas as transformação comportamental, organizacional e de atividades descritas na seção 6.2.4. Para facilitar o desenvolvimento do documento jPDL foi construído um metamodelo da linguagem jPDL, cujas principais metaclasses são apresentadas na Figura 6.20.

No metamodelo apresentado na Figura 6.20 a metaclasse `Process` representa o processo de negócio que será construído no modelo jPDL. As metaclasse `End` e `Start` representam, respectivamente, os eventos iniciais e finais do modelo de processo implementado no jPDL. A metaclasse `Java` representa atividades realizadas automaticamente e a metaclasse `Task` representa uma atividade realizada por um ator do sistema. A metaclasse `State` representa um estado do processo de negócio. As metaclasses `Fork`, `Join` e `Exclusive` são utilizadas para representar, respectivamente, uma divisão do fluxo em dois ou mais caminhos que devem executar as atividades em paralelo, uma união de dois ou caminhos do fluxo em apenas um caminho e uma escolha exclusiva entre dois ou mais caminhos do processo de negócio.

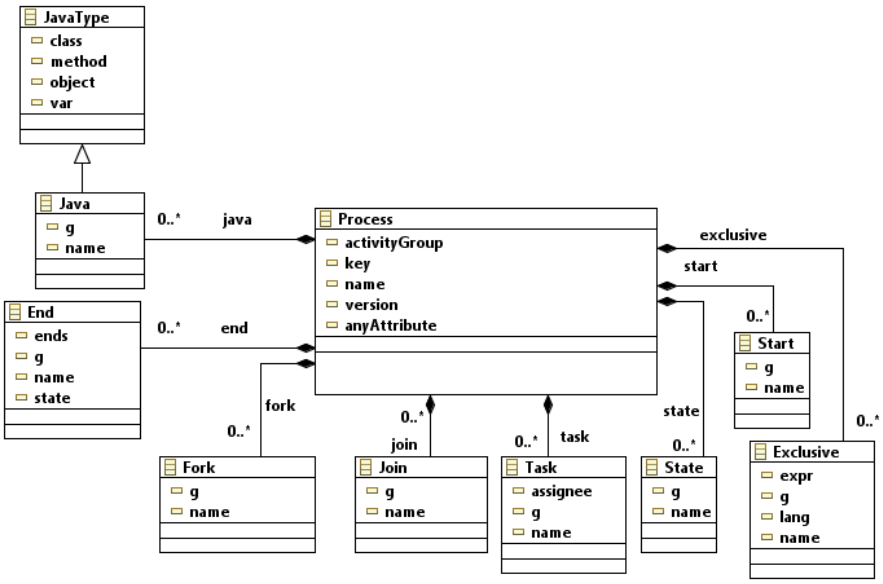


Figura 6.20 – Fragmento do metamodelo da linguagem jPDL.

Por fim, é importante ressaltar que o aplicativo que concretiza a arquitetura apresentada na Figura 6.19 foi desenvolvido utilizando objetos que representavam as entidades do ARIS Method, Metamodelo de Transformações e do metamodelo do jPDL . Em outras palavras, com exceção da transformação do documento AML para modelo AML e da transformação modelo jPDL para documento jPDL, a aplicação foi desenvolvida seguindo uma abordagem orientada a modelos. Isso possibilitou maior agilidade no desenvolvimento da aplicação, pois houve manipulação de informações no nível de modelos e não no nível de documento.

6.4 APLICAÇÃO DAS TRANSFORMAÇÕES NO NÍVEL INDEPENDENTE DE PLATAFORMA

Esta seção apresenta o conjunto de decisões de projeto e as transformações no nível independente de plataforma aplicadas sobre os modelos do processo “Triar pacientes para admissão no setor”.

6.4.1 Transformações Comportamentais

Esta seção apresenta as decisões de projeto e as transformações que permitiram definir quais atividades do processo de negócio “Triar pacientes para admissão no setor” serão incluídas no sistema de workflow.

A primeira decisão de projeto define que é necessário criar uma nova atividade de mais alto nível de abstração chamada "*Buscar ficha do paciente*", que abstrai a atividade "*Fornecer cartão de consulta ao recepcionista*", de responsabilidade do Paciente, e as atividades "*Elaborar ficha do paciente*" e "*Anexar ficha do paciente ao prontuário médico*", de responsabilidade do Recepcionista. A decisão de projeto também define que o responsável pela nova atividade é o Recepcionista. Para aplicar essa decisão de projeto foi construída uma Transformação de Abstração que possui uma Parametrização de Automatização que define a atividade como gerenciada. A Figura 6.21 apresenta o resultado da decisão de projeto após a transformação de abstração.

Como a Transformação de Abstração, apresentada na figura acima, envolveu atividade de responsáveis distintos (raias diferentes) foi necessária a escolha de um dos responsáveis para a nova atividade, respeitando as regras da Transformação de Abstração apresentadas na seção 5.3.1.

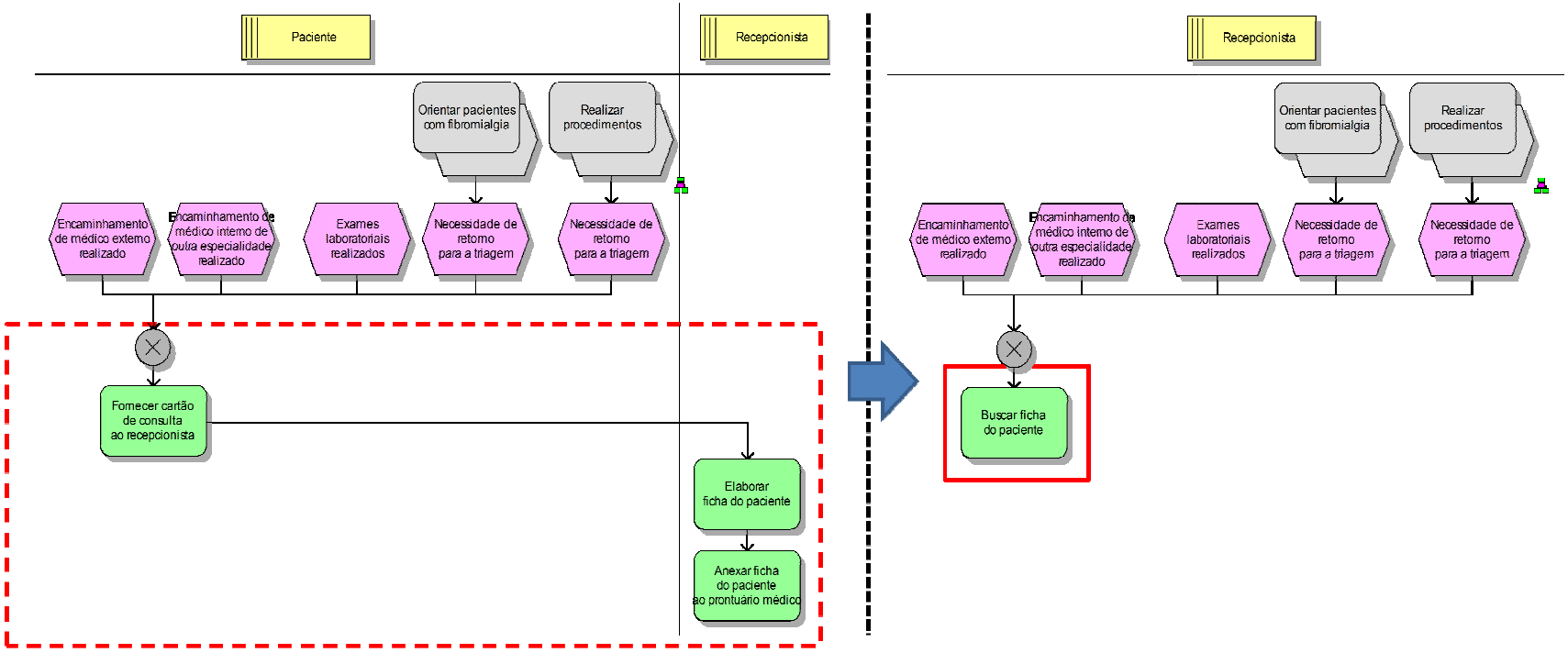


Figura 6.21 – Criando o componente “Buscar ficha do paciente”.

A segunda decisão de projeto define que é necessário abstrair a atividade "*Relatar sintomas*", da responsabilidade do Paciente, e o conjunto de atividades "*Investigar histórico clínico do paciente*", "*Investigar histórico pessoal do paciente*", "*Investigar histórico familiar do paciente*", "*Realizar exame físico*" e "*Decidir necessidade de exames para confirmar hipótese de diagnóstico*", da responsabilidade do Médico, para criar uma nova atividade de mais alto nível. Este a nova atividade é chamada de "*Examinar o paciente*" que será da responsabilidade do Médico. Para aplicar essa decisão de projeto foi construída uma Transformação de Abstração que possui uma Parametrização de Automatização que define que atividade será gerenciada. A Figura 6.22 o resultado da segunda decisão de projeto após aplicar a transformação de abstração. Na atividade "Examinar o paciente" o médico também informar se há necessidade de novos exames laboratoriais e, caso seja necessário, quais exames devem ser realizados pelo paciente.

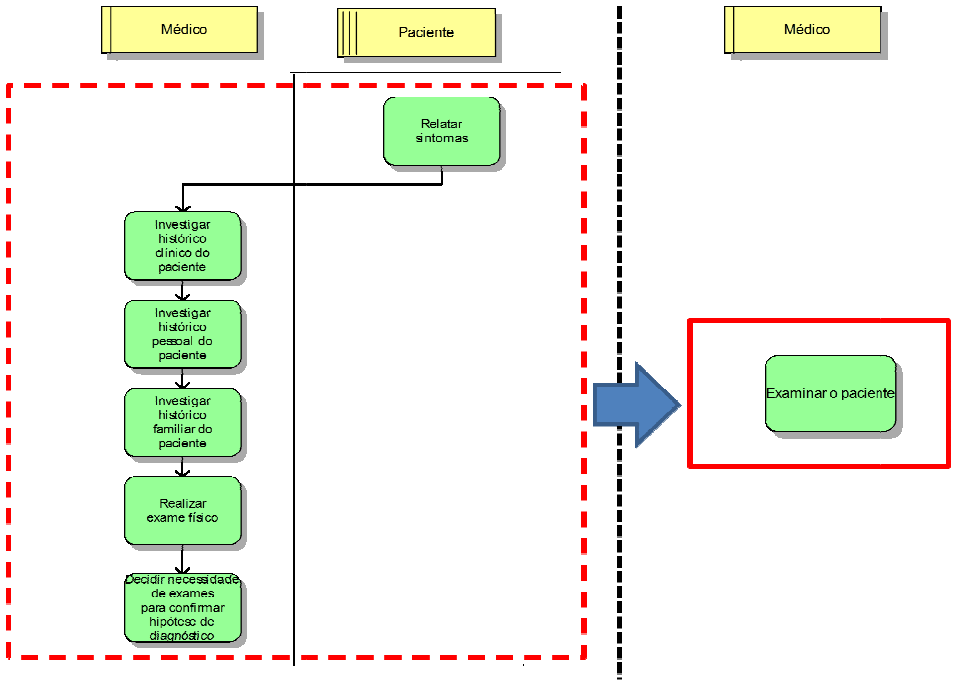


Figura 6.22 – Criando o componente "Examinar o paciente".

A próxima decisão de projeto define que a atividade "*Solicitar exames laboratoriais*", da responsabilidade do Médico, e as atividades "*Registrar folha de faturamento no sistema*" e "*Registrar no sistema data do agendamento de retorno*", de responsabilidade da Recepcionista, serão abstraídas em uma nova atividade de mais alto nível chamado de "*Solicitar exames laboratoriais*". A decisão de projeto também definiu que essa

atividade será totalmente automatizada e que será desempenhada pelo sistema computacional chamado "*Sistema de solicitação de exames*". Desta forma, para aplicar tal decisão de projeto foi selecionada uma Transformação de Abstração que transforma as atividades selecionadas em uma atividade de mais alto nível e que será desempenhada por um sistema. Em outras palavras, foi criada uma transformação de abstração com uma Parametrização de Automatização para transformar a atividade em totalmente automatizada. A Figura 6.23 apresenta a transformação da terceira decisão de projeto.

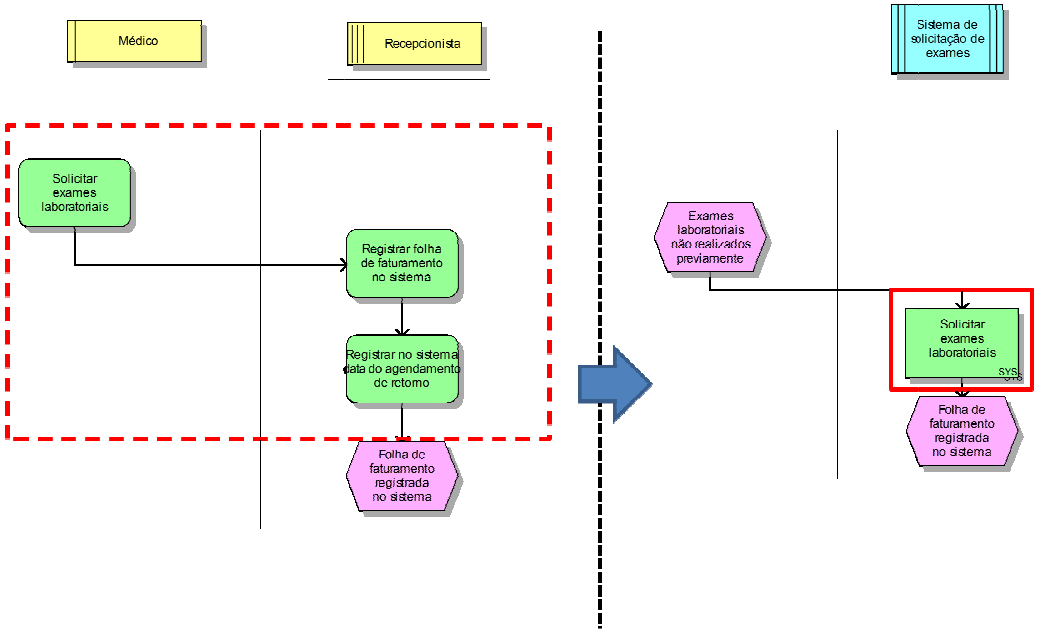


Figura 6.23 – Criando o componente totalmente automatizado "*Solicitar exames laboratoriais*".

Ao analisar o modelo do processo de negócio o projetista definiu que as atividades as atividades "*Elaborar hipótese de diagnóstico*" e "*Verificar fechamento do diagnóstico*", ambas de responsabilidade do Médico, podem ser abstraídas em apenas uma atividade de mais alto nível de abstração chamada "*Elaborar diagnóstico preliminar*". A nova atividade é de responsabilidade do Médico. Logo, foi criada uma transformação de abstração que abstrai as atividades citadas e cria a atividade de mais alto nível chamada "*Elaborar diagnóstico preliminar*" como gerenciada pelo Médico. A Figura 6.24 apresenta transformação de abstração utilizada para construir a nova atividade "*Elaborar diagnóstico preliminar*".

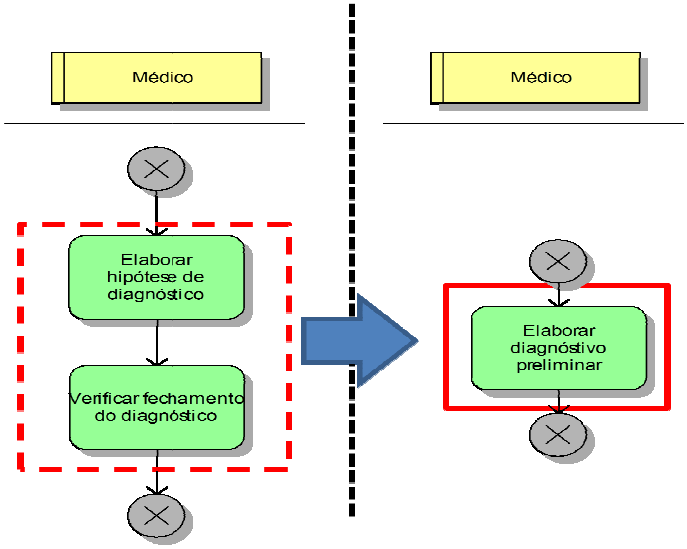


Figura 6.24 - Quinta transformação no nível de independente de plataforma.

Durante a análise do processo foi identificado no modelo de processo de negócio que as atividades "*Encaminhar paciente ao diagnóstico*", "*Agendar ambulatório de diagnóstico para paciente*", de responsabilidade do Médico, e as atividades "*Registrar folha de faturamento no sistema*" e "*Registrar no sistema data do agendamento de retorno*", de responsabilidade do Recepcionista, podem ser abstraídas, gerando uma nova atividade de mais alto nível. Desta forma, o projetista definiu que é necessário criar uma transformação de Abstração que una tais atividades e crie uma nova atividade de maior nível de abstração denominado "*Encaminhar paciente e agendar ambulatório*" que será da responsabilidade do Médico. A Figura 6.25 apresenta o resultado dessa decisão de projeto após a transformação.

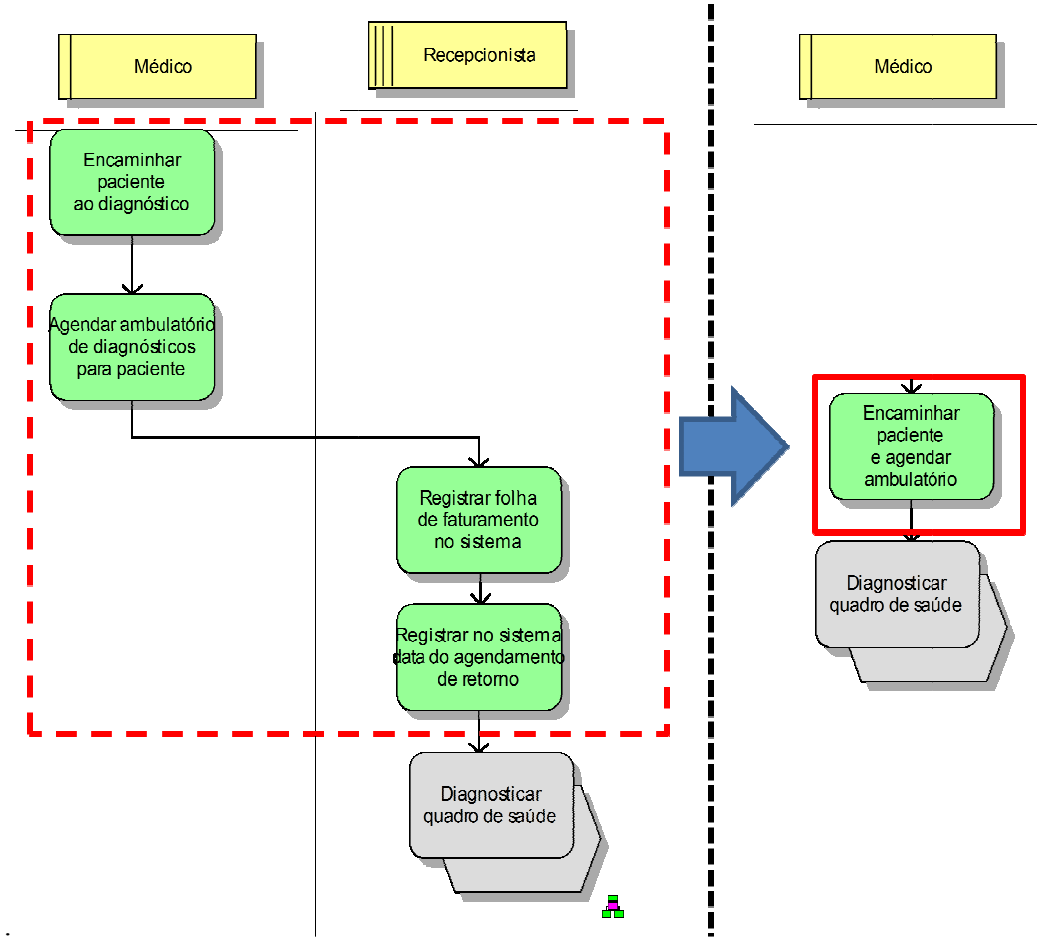


Figura 6.25 – Criando o componente "Encaminhar paciente e agendar ambulatório".

Dando continuidade à transformação do modelo de processo em um modelo estrutural do workflow, o projetista identificou que as atividades "*Fornecer recomendações complementares ao paciente*" e "*Conceder alta ao paciente*", de responsabilidade do Médico, e a atividade "*Registrar folha de faturamento no sistema*", de responsabilidade do Recepcionista, podem ser abstraídas em uma atividade de mais alto nível. O projetista definiu que a nova atividade será chamado de "*Conceder alta*" e será da responsabilidade do Médico. A Figura 6.26 apresenta a decisão de projeto que originou o componente "*Conceder alta*".

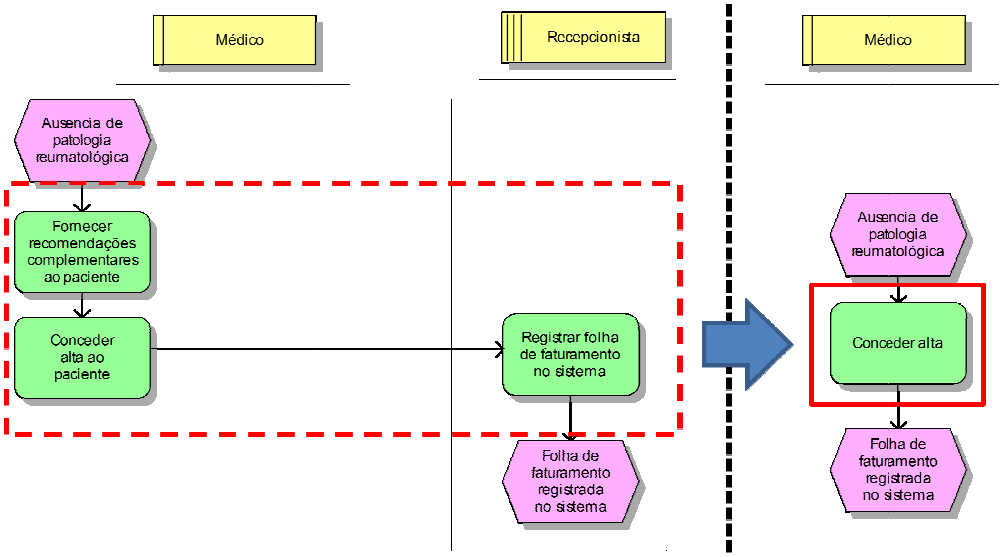


Figura 6.26- Criando o componente "Conceder alta".

Para aplicar a decisão de projeto acima, foi necessário criar uma transformação de abstração que abstraiu as atividades do Médico e Recepcionista e cria o nova atividade "*Conceder alta*" como gerenciada pelo usuário.

A análise do processo de negócio identificou que o conjunto de atividade presentes no retângulo pontilhado presente na Figura 6.27 poder ser abstraída em uma atividade do processo de maior nível de abstração. O projetista também definiu que a nova atividade será da responsabilidade do Médico. Para satisfazer essa decisão de projeto, foi criada uma transformação de abstração que abstrai as atividades contidas dentro do retângulo pontilhado e cria uma nova atividade de mais alto nível chamado de "*Encaminhar paciente para ambulatório*".

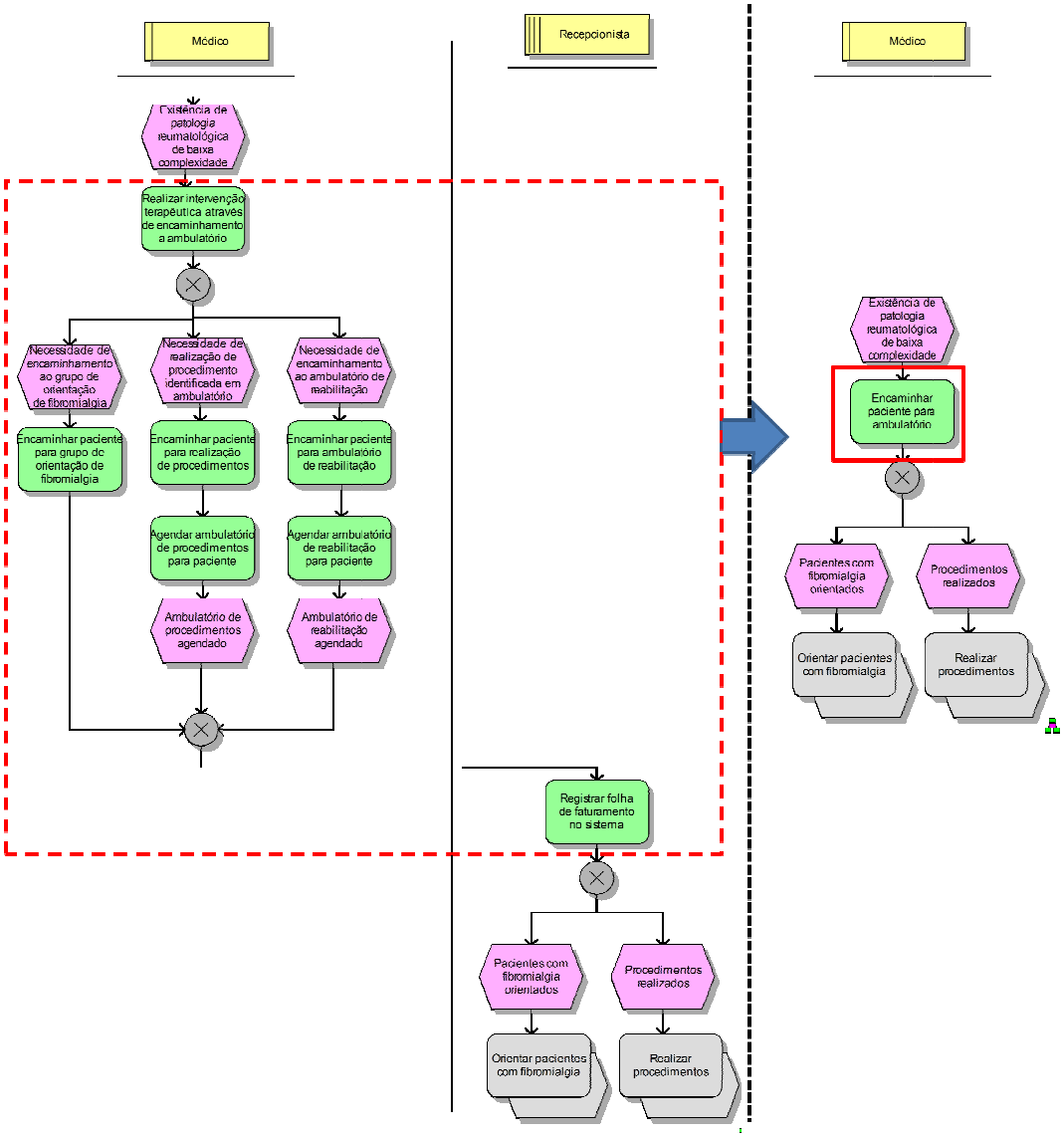


Figura 6.27 – Criando o componente "Encaminhar paciente para ambulatório".

A transformação que criou a atividade "*Encaminhar paciente para ambulatório*" é uma demonstração de como a Transformação de Abstração pode ser aplicada em um ou mais padrões de processo. Neste caso, a transformação foi aplicada sobre os padrões de processo *Sequence*, *XOR-Split* e *XOR-Join* que em conjunto formam um componente e, desta forma, permitindo que a Transformação de abstração crie uma atividade de mais alto nível.

A próxima decisão de projeto aplicada ao modelo de processo visa abstrair todo o conjunto de atividades presente no retângulo pontilhado da Figura 6.28 e construir uma atividade de maior nível de abstração, denominado de "*Explicar o diagnóstico ao*

paciente". O projetista também definiu que a nova atividade será da responsabilidade do Médico.

Para aplicar a decisão de projeto apresentado na figura acima, foi necessária a construção de uma Transformação Abstração que abstrai todo o conjunto de atividades presente no retângulo e, assim, criar a nova atividade "*Explicar o diagnóstico ao paciente*". O nível de automatização deste novo componente é gerenciada.

Para as atividades "*Verificar realização prévia de exames laboratoriais*", "*Analisar resultados dos exames*", "*Verificar existência de doença reumatológica*" e "*Verificar complexidade da doença reumatológica*" a decisão de projeto foi simplesmente definilas com atividades que serão gerenciadas por seus responsáveis. Neste caso, por coincidência, todas as atividades são da responsabilidade do Médico. Assim, para aplicar tal decisão de projeto no modelo de processo, foi criada uma transformação para cada atividade do processo citada acima que descreve que o nível de automatização delas é gerenciada.

Através das transformações foi possível aplicar as decisões de projeto no modelo de processo e, assim, construir o modelo estrutural do sistema de workflow. A Figura 6.29 e a Figura 6.30 apresentam o modelo estrutural do sistema de workflow

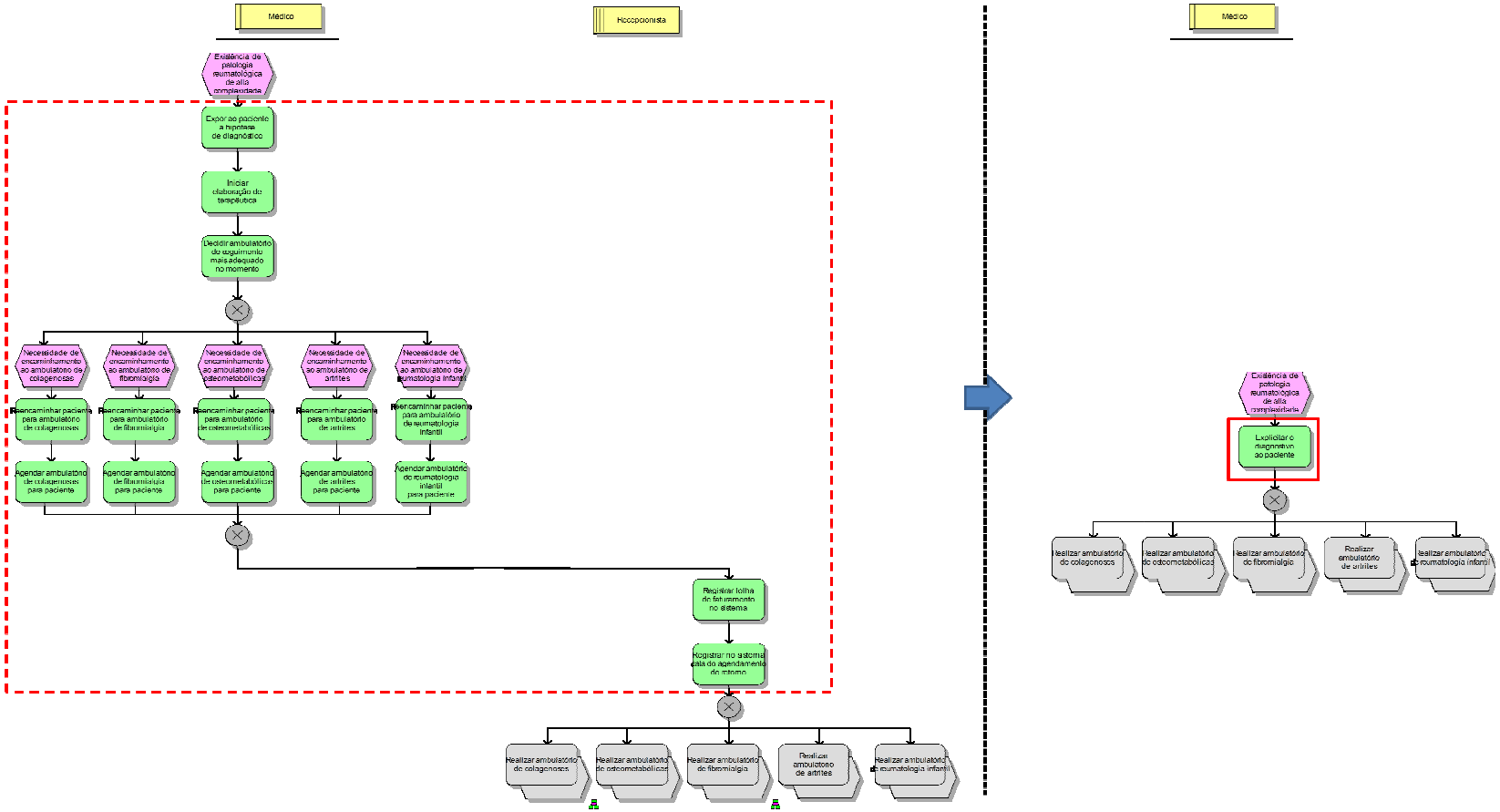


Figura 6.28 - Criando o componente "Explicitar o diagnóstico ao paciente".

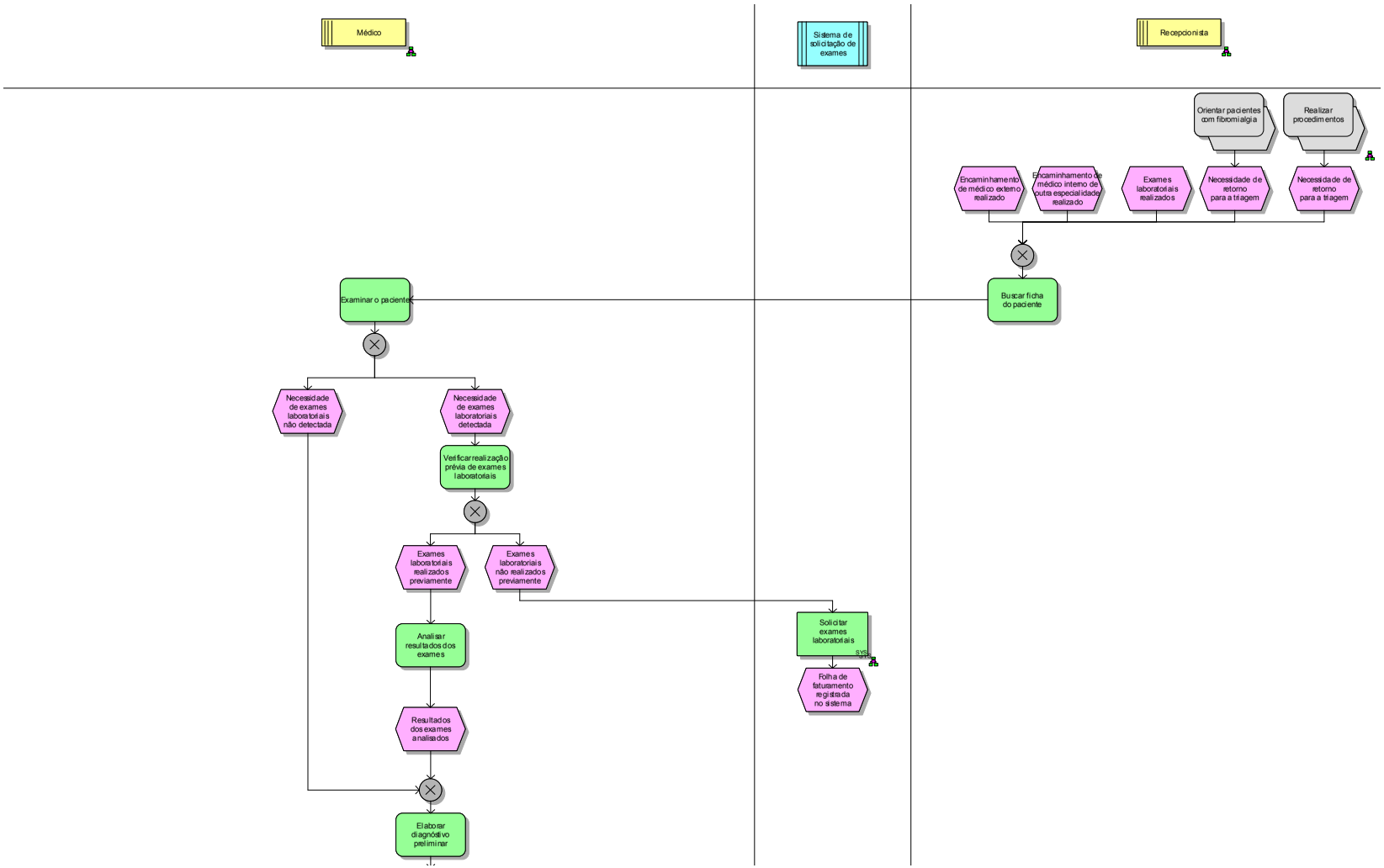


Figura 6.29 - Modelo de estrutural do sistema de workflow (Parte 1).

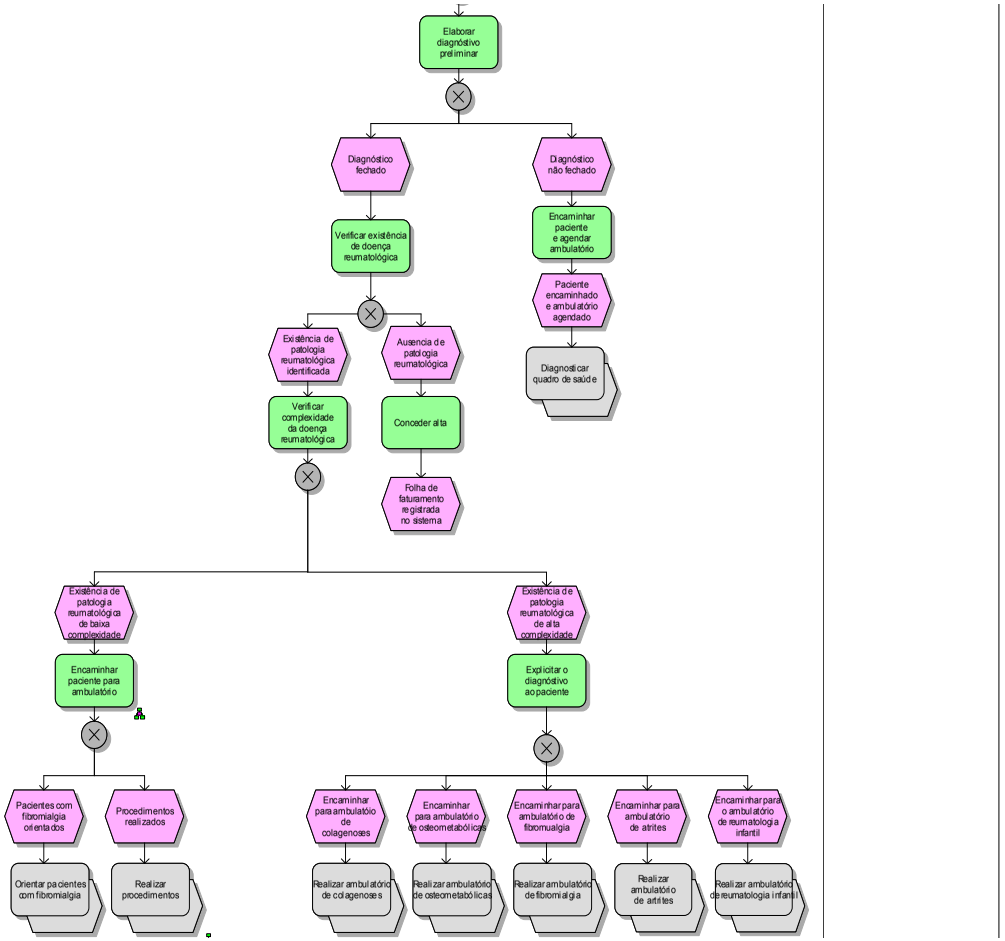


Figura 6.30 - Modelo de estrutural do sistema de workflow (Parte 2).

6.4.2 Transformações de Atividades

Nesta etapa do projeto, o projetista deve definir quais elementos são necessários para cada atividade do processo de negócio já existente ou para cada nova atividade do processo de negócio criado pelas Transformações de Abstração e de Refinamento. Em outras palavras, neste momento o projetista deve aplicar decisões de projeto que definem quais informações são necessárias e quais informações são produzidas por cada atividade, quais notificações que devem ser feitas pela atividade, quais regras de negócio devem ser consideradas e assim por diante.

Para que seja possível a definição dos detalhes de cada atividade, o projetista: **(i)** construir uma transformação de detalhamento de componente para os componentes que não foram construídos pelas transformações de Abstração e Refinamento e; **(ii)** construir modelos de detalhamento para os componentes que foram criados pelas transformações de Abstração e Refinamento.

Para exemplificar o primeiro caso, será utilizado o FAD da atividade "*Analisar resultados dos exames*". Nesta atividade, o projetista definiu que o documento "Exame do paciente" e suas informações (o cluster "Informações do exame") são necessários para a execução da atividade e que a atividade deve gerar a informação "Resultado da análise do exame". Por fim, a decisão de projeto define que o "Sistema do laboratório" irá auxiliar na execução da atividade no sistema computacional em desenvolvimento. Os retângulos pretos na Figura 6.31 demonstram a decisão de projeto sobre o FAD da atividade "*Analisar resultados dos exames*".

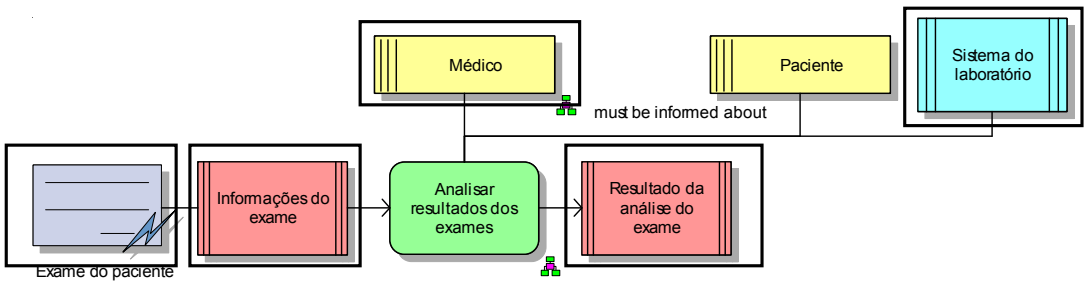


Figura 6.31 - Modelo Detalhado da atividade "Analisar resultados dos exames" parametrizado.

Para exemplificar o segundo caso, no qual é necessário criar os modelos de detalhamento de atividades, serão utilizadas como exemplos as novas atividades: "*Solicitar exames laboratoriais*" e "*Encaminhar paciente para ambulatório*".

Ao criar os modelos de detalhamento das atividades criadas pelas transformações de Abstração e Refinamento o projetista não pode infringir quaisquer decisões de projeto que já foram aplicadas. Por exemplo, no caso da atividade "*Solicitar exames laboratoriais*" o projetista deve modelar o "Sistema de solicitação de exames" como o responsável pela atividade, pois isso foi definido nas transformações comportamentais, como demonstrado na seção 5.3.

No caso da atividade "*Solicitar exames laboratoriais*" é importante observar que a transformação que o criou, definiu que tal atividade é totalmente automatizado e, assim, o responsável por tal atividade é um sistema computacional (Sistema solicitação de exames, conforme demonstrado na Figura 6.32). Logo, tal decisão de projeto deve está presente no modelo de detalhamento desta atividade.

O projetista definiu que a atividade "*Solicitar exames laboratoriais*" terá como entrada as informações do prontuário médico do paciente (representado pelo cluster Informações do prontuário médico), provido pelo Sistema de controle de paciente, e produzirá como saída a "Solicitação do exame laboratorial" e "Notificação sobre o atendimento do paciente". Estes elementos não estão presentes em nenhum dos FADs das atividades abstraídas que originaram a nova atividade "*Solicitar exames laboratoriais*", mas foram incluídos pela necessidade identificada.

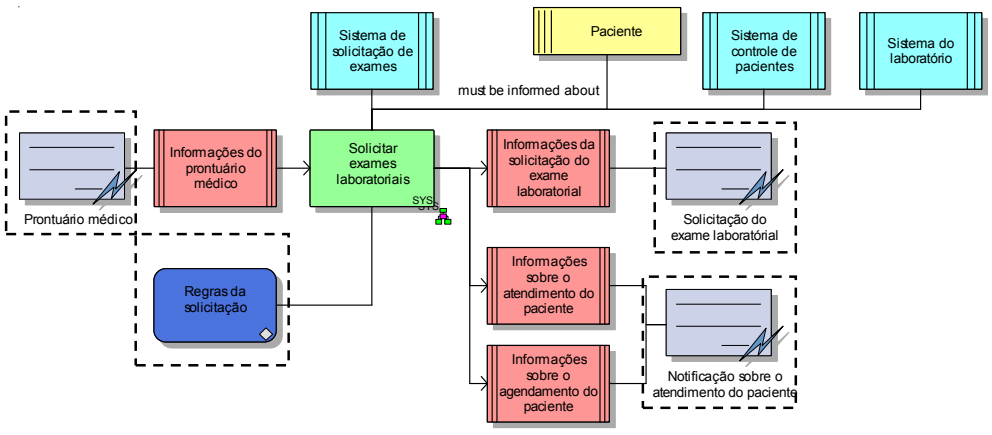


Figura 6.32 - FAD criado para o componente "Solicitar exames laboratoriais".

Observe que foi usado um objeto do tipo *Business Rule* denominado "Regras de solicitação". Este objeto foi incluído no modelo para definir as regras de negócio necessárias nesta atividade como, por exemplo, quando o Paciente deve ser notificado sobre a "Solicitação do exame laboratorial". Os demais elementos presentes no modelo acima, foram retirados das atividades abstraídas que originaram o componente.

A Figura 6.33 apresenta o modelo FAD construído para a nova atividade "Encaminhar paciente para ambulatório". Como pode ser observado, o projetista definiu que:

i. as informações contidas no prontuário médico do paciente são necessárias para a execução da atividade abstraída ;

ii. a atividade deve enviar uma ou mais notificações para os grupos de ambulatórios;

iii. os sistemas "Sistema de agendamento de ambulatório" e "Sistema de controle de paciente" auxiliam na execução do componente; e

iv. a atividade possui uma regra de negócio que descreve o seu comportamento.

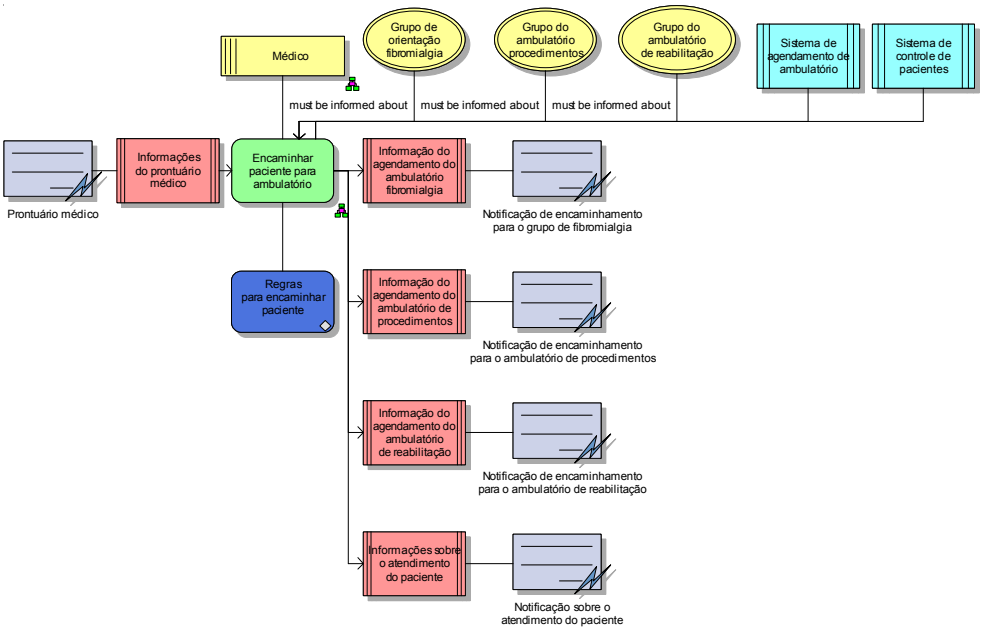


Figura 6.33 – FAD criado para o componente "Encaminhar paciente para ambulatório".

6.4.3 Transformações Organizacionais

As transformações organizacionais permitem que o projetista aplique as decisões de projeto nos modelos organizacionais com o objetivo de definir quais entidades organizacionais serão transformados em atores no sistema de workflow em desenvolvimento. Essas transformações também definem o relacionamento entre os atores do sistema computacional. As Transformações Organizacionais são transformações que possuem a Parametrização Organizacional.

A primeira transformação organizacional define quais entidades organizacionais poderão executar o papel de "Recepcionista" no sistema computacional em desenvolvimento. Conforme demonstrado na Figura 6.34, somente as pessoas que executam os *positions* "Auxiliar Administrativo" podem executar o papel de "Recepcionista" no sistema computacional em desenvolvimento.

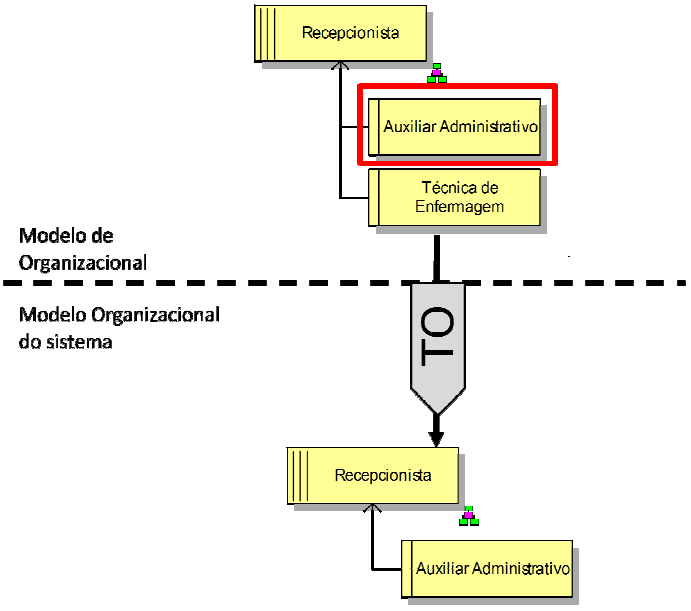


Figura 6.34 – Parametrização do modelo organizacional "Recepcionista".

O projetista definiu que dentre todas as especializações de médicos presentes no hospital, somente os médicos que executam o *position* "Reumatologista" podem instanciar o papel "Médico" no sistema de workflow que está sendo desenvolvido. A Figura 6.35 demonstra a decisão de projeto aplicada sobre o modelo organizacional do *Position Description* "Médico".

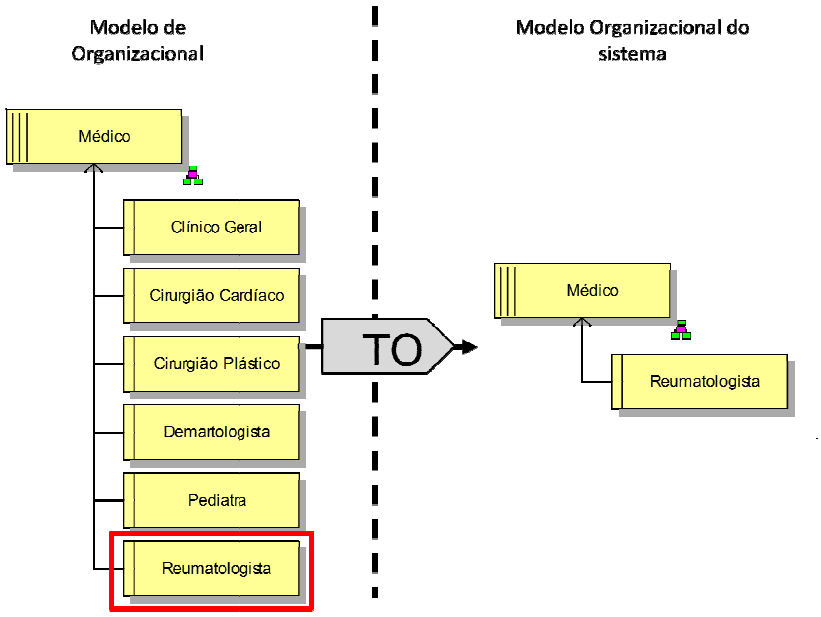


Figura 6.35 - Parametrização do modelo organizacional "Médico".

Por fim, o projetista decidiu que somente as pessoas que desempenharem o papel de "Enfermeira chefe" no "Grupo do ambulatório de reabilitação" receberão a notificação do agendamento de um paciente feito pelo Médico na atividade "Encaminhar paciente para ambulatório". A Figura 6.36 demonstra a decisão de projeto sobre o modelo organizacional "Grupo do ambulatório de reabilitação".

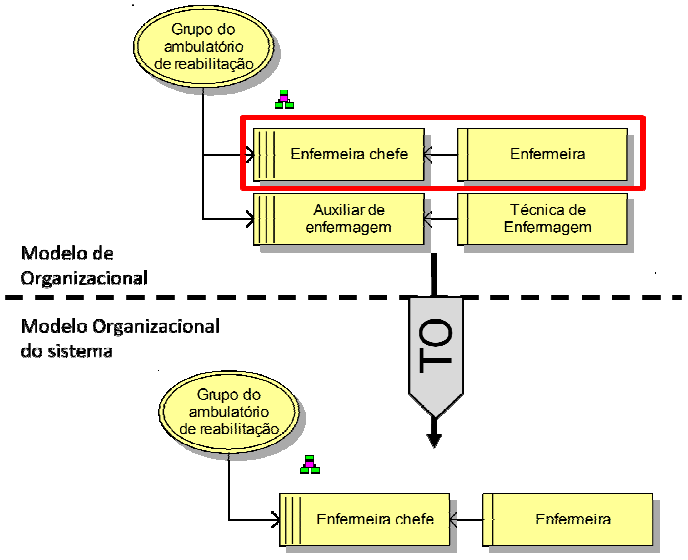


Figura 6.36 – Paramtrização do modelo organizacional "Grupo do ambulatório de reabilitação".

Por fim, observe na figura acima, que a decisão de projeto aplicada sobre o modelo organizacional também inclui a seguinte condição: Somente as pessoas que

desempenham o *position* “Enfermeira” na organização podem desempenhar o papel “Enfermeira chefe” no grupo do ambulatório de reabilitação. Isso é uma restrição imposta pela organização e que deve está presente no sistema computacional em desenvolvimento. Tais regras da decisão de projeto devem está presentes na transformação organizacional que auxilia no desenvolvimento do sistema computacional.

6.5 APLICAÇÃO DAS TRANSFORMAÇÕES NO NÍVEL DE DEPENDENTE DE PLATAFORMA

As transformações do nível dependente de plataforma possuem o mesmo princípio das transformações do nível independente de plataforma, ou seja, a partir de modelo de origem e das decisões de projeto é construído o modelo de destino. No caso das transformações do nível de plataforma de realização, os modelos de origem são os modelos produzidos nas transformações do nível independente de plataforma e o modelo de destino é um modelo jPDL do JBOSS jBPM.

Apesar das transformações do nível de dependência de plataforma, neste trabalho, serem transformações tecnológicas, o conhecimento da semântica dos elementos contidos nos modelos do domínio de origem e do domínio de destino é necessário. Afinal, através do conhecimento preciso dos conceitos presentes em cada domínio é possível a construção de transformações que utilizem de maneira mais adequada as informações contidas no modelo de origem e, assim, gerar modelos no domínio de destino que sejam compatíveis com os conceitos presentes no domínio de origem.

Para uma melhor utilização dos conceitos presentes nos modelos de origem, as transformações do nível dependente de plataforma foram divididas em 3 tipos: **(i)** Transformações comportamentais; **(ii)** Transformações de atividade; e **(iii)** Transformações organizacionais.

As transformações comportamentais utilizam as informações contidas no modelo comportamental, construído na seção 6.4.1, para produzir o modelo do processo em jPDL. Esse modelo é o que define a sequência de atividades e os responsáveis para as mesmas.

As transformações de atividade irão utilizar as informações contidas nos modelos de detalhamento de atividade para complementar o modelo jPDL desenvolvido na transformação comportamental.

As transformações organizacionais utilizam as informações contidas nos modelos organizacionais produzidos na seção 6.5.3, com a finalidade de informar as entidades organizacionais que serão atores no sistema.

Após a execução das transformações do nível dependente de plataforma o projetista possui um modelo jPDL que retrata toda as decisões de projeto que foram tomadas sobre os modelos da Arquitetura Organizacional de TI. Desta forma, o modelo produzido está de acordo com diversos aspectos organizacionais e com as decisões de projeto.

Por fim, o desenvolvedor deve complementar as "lacunas" deixadas pelas transformações. Por exemplo, a implementação das atividades totalmente automatizadas e das telas utilizadas para realizar a interface entre o usuário e o sistema computacional.

6.5.1 Transformação Comportamental

A transformação comportamental do nível de dependência de plataforma é composta de seis etapas especializadas. Cada uma dessas etapas trabalham em conjunto para o desenvolvimento do modelo jPDL do processo "Triar pacientes para admissão no setor".

A primeira etapa da transformação comportamental é utilizada para criar as raias das entidades organizacionais no jPDL. Para isso, é necessário identificar as entidades organizacionais que são responsáveis por componentes de processo com nível de gerenciada. A identificação dos responsáveis é feita verificando a existência do relacionamento *carries out* entre uma entidade organizacional com uma componente do processo (representado pela entidade *Function*).

Para cada entidade organizacional identificada na primeira etapa é criado o elemento `<swimlane>` no jPDL. Este elemento representa quais usuários ou conjunto de usuário

responsáveis por executar um ou mais atividades no JBPM. A Figura 6.37 apresenta o resultado do primeiro passo.

```
<swimlane name="Medico"/>

<swimlane name="Recepcionista"/>
```

Figura 6.37 – Interpretação do responsáveis pelas atividades no jDPL.

O jPDL não permite a criação de *<swimlane>* para entidades computacionais responsáveis por atividades automatizadas no processo de negócio. Todas as atividades que são da responsabilidade de uma entidade computacional serão executadas automaticamente pelo JBOSS jBPM.

A segunda etapa é responsável por criar os elementos que representam o evento inicial (*<start>*) e final (*<end>*) no modelo jPDL, através da identificação das entidades *Start Event* e *End Event* no modelo de processo. A linguagem jPDL possui uma restrição sob os eventos iniciais: não pode existir mais de um evento inicial no modelo. Não existem restrições desse tipo para os eventos finais.

Para se adequar a restrição do evento inicial, a transformação cria um evento inicial (*start*) seguindo de um *decision*. Para cada evento inicial contido no processo será criado um *state*. Desta forma, o modelo jPDL contem os seguintes *states*: "Encaminhamento de médico externo realizado", "Encaminhamento de médico interno de outra especialidade realizado", "Exames laboratoriais realizados" e "Necessidade de retorno para a triagem". A Figura 6.38 apresenta como os eventos iniciais do processo de negócio serão representados no jPDL.

Como pode ser observado na Figura 6.38, para cada evento inicial do modelo de processo foi criado um *state* e um *transition* no *decision* chamado decisionStartEvent. Cada *transition* do decisionStartEvent possui uma expressão (*expr*) que irá definir para qual estado (evento inicial) o processo irá seguir. Por exemplo, caso o valor da expressão seja "encaminhamentoMER", isso indicará que foi o evento inicial "Encaminhamento do medico externo realizado" que iniciou o processo de negócio.

```
<start name="inicio">
<transition to="decisionStartEvent"/>
</start>

<decision name="decisionStartEvent">
<transition to="Encaminhamento do medico externo realizado">
      <condition expr="#{conten==encaminhamentoMER}">
</transition>

<transition to="Encaminhamento do medico interno de outra especialidade
realizado">
      <condition expr="#{conten==encaminhamentoMIER}">
</transition>

<transition to="Exames laboratoriais realizados">
      <condition expr="#{conten==examesLR}">
</transition>

<transition to="Necessidade de retorno para a triagem">
      <condition expr="#{conten==necessidadeRT}">
</transition>

</decision>

<state name = "Encaminhamento do medico externo realizado"/>

<state name ="Encaminhamento do medico interno de outra especialidade
realizado"/>

<state name = "Exames laboratoriais realizados"/>

<state name = "Necessidade de retorno para a triagem"/>
```

Figura 6.38 – Interpretação do elemento Start Event no jPDL.

Em relação aos eventos finais, a segunda etapa cria um elemento `<end>` no modelo jPDL para cada evento final do modelo de processo de negócio. Conforme demonstrado na Figura 6.39 foram criados elementos `<end>` para os seguintes eventos finais: "Procedimentos realizados", "Pacientes com fibromialgia orientados", "Folha de faturamento registrada no sistema".

```
<end name="Procedimentos realizados"/>

<end name="Pacientes com fibromialgia"/>

<end name="Folha de faturamento registrada no sistema"/>
```

Figura 6.39 – Interpretação do elemento End Event no jPDL.

A terceira etapa é responsável por criar as entidades no jPDL que representam uma *Function*, *System Function Actual* e *Process Interface* do ARIS Method no modelo em desenvolvimento. O elemento *Function* do ARIS é interpretado como um elemento

Task do jPDL e os elementos *System Function (Actual)* e *Process Interfaces* são interpretados com o elemento *Java* do jPDL[8].

Levando em consideração a semântica do elemento notacional *Process Interface*, apresentada na seção 3.3, sempre que esse for interpretado como um elemento *Java* será necessário adicionar um elemento *end* para representar o fim da instância atual do processo.

Para exemplificar a terceira etapa da transformação será utilizada a *Function* "Examinar o paciente", a *System Function (Actual)* "Solicitar exames laboratoriais" e a *Process Interface* "Diagnosticar quadro de saúde". A Figura 6.40 apresenta o resultado da terceira etapa.

```
//Tarefa executada por humanos
<task name="Examinar paciente"/>

//Tarefa executada automaticamente por sistemas computacionais
<java name="Solicitar exames laboratoriais" class="TODO" method="TODO"
var="TODO" />

//Process Interface
<java name="Diagnosticar quadro de saúde" class="TODO" method="TODO"
var="TODO">
    <trasition name="toENDDiagosticar" to="ENDDiagnosticar"/>
</java>

<end name="ENDDiagnosticar" />
```

Figura 6.40 - Interpretação do elementos Function, System Function (Actual), Process Interfaces no jPDL.

Como pode ser observado na figura acima, o elemento *Java* é possui um conjunto de parâmetros que necessitam ser preenchidos. Por causa disto, foi atribuído aos mesmos o valor "*TODO*". O valor "*TODO*" descreve que deve ser preenchido pelo desenvolvedor após a conclusão das transformações.

Para representar que após a execução de um *Process Interface* é o fim do processo de negócio, foi adicionado o elemento *transition* e o elemento *end* no modelo jPDL.O

[8] No jPDL o elemento *Task* representa uma atividade que é desempenhada por humanos e o elemento *Java* representa uma tarefa automatizada.

elemento *transition* define um possível caminho que o fluxo do processo poderá seguir .

A quarta etapa é responsável por relacionar as entidades organizacionais, identificadas na primeira etapa, com as atividades realizadas por humanos, identificados na terceira etapa, no modelo jPDL em construção. O relacionamento entre essas duas entidades é feito através do relacionamento *carries out*. A Figura 6.41 apresenta o resultado da aplicação da quarta etapa sob a atividade "Examinar Paciente" (desempenhada pelo Médico).

```
//Tarefa executada por humanos
<task name="Examinar paciente" swimlane="Médico"/>
```

Figura 6.41 - Interpretação do elementos Function e System Function Actual no jPDL.

A quinta etapa é responsável por interpretar as regras de processo, representadas pelo elemento *Rule* do ARIS Method, em termos das entidades do jPDL. A Tabela 6.1 apresenta a interpretação para cada *Rule* que existe no ARIS Method na linguagem jPDL.

Tabela 6.1 – Interpretação das regras de processo no jPDL.

Elemento notacional do ARIS Method	Elemento no jPDL
OR –Split	*Decision*
OR-Join	*Decision*
XOR-Split	*Decision*
XOR-Join	*Decision*
AND-Split	*Fork*
AND-Join	*Join*

Para exemplificar a quinta etapa, esta será aplicada sobre o elemento *XOR* que ocorre após a atividade "Examinar paciente". A Figura 6.42 apresenta a representação deste elemento *Rule* no jPDL.

```
<decision name="DECISION_Examinar paciente" expr="#{content}">

   <transition name="Necessidadede exames laboratoriais
          Não detectada"
                to="DECISION_Analisar_resultados_dos_exames"/>
```

```
    <transition name="Necessidade de exames laboratoriais detectada"
                to="Verificar realização prévia de exames
                laboratoriais"/>
</decision>
```

Figura 6.42 - Interpretação do elemento XOR do ARIS Method no jPDL.

Para o nome do elemento *decision* foi definida a seguinte regra: DECISION + nome do elemento *Function* anterior ao elemento *Rule*. Caso exista um conjunto de elementos *Rule* interligados através do relacionamento *link* a regra é modificada para: DECISION + nome do elemento *Function* anterior ao elemento *Rule* mais um identificador.

O parâmetro *expr* do elemento *decision* descreve qual expressão ou variável define a regra de escolha de um ou mais caminhos que o fluxo do processo poderá seguir. Como dito anteriormente, cada elemento *transition* define um possível caminho que o fluxo do processo poderá seguir. Para indicar qual caminho o fluxo de processo irá seguir, o resultado da *expr* deve ser igual ao valor associado ao parâmetro *name* do elemento *transition*. O parâmetro *to* do elemento *transition* define o componente que será ativado.

Para a construção dos *transitions* que estão presente no elemento *decision* é necessário que a quinta etapa analise no modelo de processo os relacionamentos *leads to* (ocorrem entre os elementos *Rule* e *Event*), *activates* (ocorre entre os elementos *Event* e *Function*) e *is evaluted by* (ocorre entre os elementos *Event* e *Rule*). A análise destes relacionamentos permite identificar quais são os possíveis caminhos que o processo poderá seguir.

Como pode ser observado na Figura 6.42, o valor do atributo *name* do transition é o nome do *Event* que está associado ao elemento *Rule*, pelo relacionamento *leads to*. O parâmetro *to* é utilizado para definir o elemento que será ativado pela transação. O valor do parâmetro *to* é o nome do elemento que pode ser alcançado pelos relacionamentos *activates* e *is evaluted by*.

A identificação de um *AND-Split* pela quinta etapa é realizada através da análise dos seguintes casos:

i. identificação do relacionamento *leads to* entre o elemento *Function* e o elemento notacional *AND Rule*;

ii. identificação de um ou mais relacionamentos do tipo *link* saindo do elemento notacional *AND Rule*; ou

iii. identificação do relacionamento *is evaluated by* entre um *Event* e o elemento notacional *AND Rule*.

A identificação de um *AND-Join* pela quinta etapa é realizada quando:

i. existem dois ou mais relacionamentos *is evaluated by* entre dois ou mais elementos *Event* e o elemento notacional *AND Rule*;

ii. existe um ou mais relacionamentos do tipo *link* chegando em um elemento notacional *AND Rule*; ou

iii. existe um relacionamento do tipo activates entre o elemento *AND Rule* e um elemento *Function*.

A sexta, e última, etapa é responsável por relacionar os elementos construídos nas outras etapas das transformações no modelo jPDL em desenvolvimento, através do elemento `transition` do jPDL. Para isto, é necessário identificar os seguintes relacionamentos no modelo: **(i)** is *predecessor of*, **(ii)** *activates*, e **(iii)** *leads to*. O primeiro relacionamento ocorre entre os elementos do tipo *Function*, o segundo relacionamento ocorre entre o conjunto de elementos formado por *Event* e *Rule* e o elemento *Function*, enquanto o último relacionamento ocorre entre o elemento *Function* e o elemento *Rule*.

Para exemplificar a sexta etapa, essa foi aplicada no componente de processo "Examinar paciente" com o objetivo de relacioná-la com o elemento o *XOR-Rule* (ocorre após essa atividade). A figura abaixo demonstra o resultado desta etapa.

```
<task name="Examinar paciente" swimlane="Médico">
```

```
<transition name="Paciente Examinado" to="DECISION_Examinar paciente">

</task>
```

Figura 6.43 – Criando as inter-relações entre os elementos do processo de negócio.

A partir destas etapas é possui desenvolver uma transformação que produza o modelo comportamental de qualquer modelo de processo de negócio gerado pelas transformações do nível de independente de plataforma. A Figura 6.44 apresenta um fragmento do modelo jPDL produzido pela transformação comportamental.

```
<?xml version="1.0" encoding="UTF-8"?>

<process name="JPDL4" xmlns="http://jbpm.org/4.0/jpdl">

      <!-- Raias do processo  -->
      <swimlane name="Medico"/>
      <swimlane name="Recepcionista"/>

      <!-- Evento inicial do processo  -->
      <start name="start">
            <transition name="toBuscar_ficha" to="Buscar ficha"/>
      </start>

      <!-- Atividades do processo -->
      <task name="Buscar ficha " swimlane="Recepcionista">
            <transition  name="toExaminar_o_Paciente"  to="Examinar  o
paciente"/>
      </task>

      <task name="Examinar o paciente" swimlane="Medico">
            <transition          name="toDECISION_Examinar_o_paciente"
to="DECISION_Examinar_o_paciente"/>
      </task>

      <task name="Verificar realizacao previa de exames laboratoriais"
swimlane="Medico">
            <transition
name="toDECISIONVerificar_realizacao_previa_de_exames_laboratoriais"
to="DECISION_Verificar_realizacao_previa_de_exames_laboratoriais"/>
      </task>
      <decision name="DECISION_Examinar_o_paciente">
            <transition  name="Necessidade  de  exames  laboratoriais
detectada" to="Verificar realizacao previa de exames laboratoriais"/>
            <transition name="Necessidade de exames laboratoriais nao
detectada" to="DECISION_Examinar_o_paciente_1"/>

      </decision>
```

Figura 6.44 - Fragmento do modelo em jPDL do processo de negócio "Triar pacientes para admissão no setor.

6.5.2 Transformação de Atividade

Nesta etapa do desenvolvimento do sistema de workflow os modelos de Detalhamento de Atividade ajudarão a obter informações sobre cada atividade do processo presente no sistema.

Para exemplificar a utilização das informações contidas em tais modelos será apresentada uma transformação que é utilizada para "complementar" o modelo jPDL desenvolvido na seção 6.5.1. O complemento será a adição do componente de envio de e-mail no jPDL denominados `mail`.

A transformação é composta de cinco etapas. Será utilizado o modelo de detalhamento da atividade "Encaminhar paciente para ambulatório" (Figura 6.33) para exemplificar as etapas.

A primeira etapa da transformação é a identificação dos elementos organizacionais que devem ser notificados ao fim da atividade. Essa identificação é realizada através da análise do relacionamento *must be informed about* entre os elementos *Function* e *Participant*. O resultado dessa etapa sobre o componente de processo selecionado foi: "Grupo de Orientação de fibromialgia", "Grupo do ambulatório de procedimentos" e "Grupo do ambulatório de reabilitação".

A segunda etapa identifica as notificações que cada uma das entidades organizacionais receberá. A identificação das notificações é feita através do relacionamento *lies on* entre os elementos *Cluster* e *Eletronic Document*. Para capturar o nome de cada *Eletronic Document* é lido o valor do campo *name* desse elemento. É importante ressaltar que somente os *Clusters* que possuem o relacionamento *has output of* com o elemento *Function* são analisados.

Através da segunda etapa foi possível identificar os seguintes documentos: "Notificação de encaminhamento para o grupo de fibromialgia", "Notificação de encaminhamento para o ambulatório de procedimentos" e "Notificação de encaminhamento para o ambulatório de reabilitação".

A terceira etapa é a responsável por construir e adicionar o elemento *mail* no modelo jPDL utilizando as informações contidas nas duas etapas anteriores. Desta forma, serão adicionados três componentes *mail* no modelo jPDL.

A quarta etapa é responsável por modificar a tarefas "Encaminhar paciente para ambulatório", para que se adéqüe as modificações feitas pela terceira etapa. A modificação se resume em: **(i)** remover o *transition* existente e **(ii)** adicionar os *transitions* que levam o fluxo do processo para cada uma das notificações. A Figura 6.45 apresenta o resultado da quarta etapa.

```
<task name="Encaminhar paciente para o ambulatório" swimlane="Médico">
 <transition name="toNotificação_de_encaminhamento_para_o_grupo_de
fibromialgia"
to="Notificação de encaminhamento para o grupo de fibromialgia">

<transition name="toNotificação_de_encaminhamento_para_o_ambulatório
de procedimentos"
to="Notificação de encaminhamento para o ambulatório de
procedimentos">

<transition
name="toNotificação_de_encaminhamento_para_o_ambulatório_de_reabilitaç
ão"
to="Notificação de encaminhamento para o ambulatório de reabilitação">

</task>
```

Figura 6.45 – Modificação da atividade "Encaminhar paciente para ambulatório".

A quinta, e última, etapa é a responsável por adicionar um *transition* em cada uma das notificações. O valor do parâmetro *to* do *transition* será o mesmo do *transition* que foi removido na quarta etapa. Essa etapa é importante para manter a conformidade entre as transformações. A Figura 6.46 apresenta o resultado da terceira etapa e quinta-etapa.

```
//Notificação para o grupo de fibromialgia

<mail name="Notificação de encaminhamento para o grupo de
fibromialgia">

<to addresses=#{Grupo de orientação fibromialgia} />

<subject> Notificação de encaminhamento para o grupo de fibromialgia
</subject>

<text>TODO</text>

<transition to=" DECISIONEncaminhar_paciente_para_ambulatorio"/>
```

```
</mail>

// Notificaçao para o grupo do ambulatório de procedimentos

<mail name="Notificação de encaminhamento para o ambulatório de procedimentos">

<to addresses=#{Grupo do ambulatório procedimentos} />

<subject> Notificação de encaminhamento para o ambulatório de procedimentos </subject>

<text>TODO</text>

<transition to=" DECISIONEncaminhar_paciente_para_ambulatorio"/>

</mail>

//Notificação de encaminhamento para o ambulatório de reabilitação

<mail name="Notificação de encaminhamento para o ambulatório de reabilitação">

<to addresses=#{Grupo do ambulatório de reabilitação}/>
<subject>Notificação de encaminhamento para o ambulatório de reabilitação </subject>

<text>TODO</text>

<transition to=" DECISIONEncaminhar_paciente_para_ambulatorio"/>

</mail>
```

Figura 6.46 - Notificações da atividade "Encaminhar paciente para ambulatório".

6.5.3 Transformação Organizacional

As transformações organizacionais têm por objetivo complementar o modelo jPDL construído nas seções 6.5.1 e 6.5.2, utilizando as informações contidas nos modelos organizacionais do sistema computacional construído pelas transformações independentes de plataforma.

Para exemplificar o uso das transformações organizacionais do nível dependente de plataforma, serão apresentadas duas transformações. A primeira transformação complementa o modelo jPDL descrevendo quais entidades organizacionais que irão instanciar os papéis definidos no processo de negócio. A segunda transformação complementa o modelo jPDL adicionando atividades de notificação.

Para exemplificar a utilização da primeira transformação organizacional, essa será aplicada sobre os modelos organizacionais de sistema computacional do Médico (Figura 6.35) e da Recepcionista (Figura 6.34).

No caso desses dois modelos, a transformação organizacional identifica qual entidade irá assumir o papel de Médico e o papel de Recepcionista no sistema computacional através dos relacionamentos *performs* ou *occupies.* Como explicado na seção 4.3, esses relacionamentos devem existir entre o elemento *Person Type* (e suas especializações) e as entidades organizacionais que irão instanciá-lo. Porém, em outros modelos talvez seja necessário analisar outros relacionamentos.

Como demonstrado nas Figura 6.34 e Figura 6.35 somente, o Reumatologista poderá assumir o papel de Médico e, somente, o Auxiliar de Administrativo poderão assumir o papel de Recepcionista no processo. Desta forma, a Transformação Organizacional irá adicionar associar o *position* "Reumatologista" ao papel médico e o *position* "Auxiliar Administrativo" ao papel "Recepcionista". A Figura 6.47 apresenta o resultado da primeira transformação

```
<swimlane name= "Medico" assignee="reumatologista"/>
<swimlane name="Recepcionista" assignee="auxiliar Administrativo"/>
```

Figura 6.47 - Transformação organizacional

Para exemplificar a utilização da segunda transformação organizacional, essa será aplicada sobre o modelo organizacional de sistema computacional demonstrado na Figura 6.36.

No caso da Figura 6.36 o projetista definiu que apenas a "Enfermeira chefe" deve receber a notificação que será encaminhada para o "Grupo do ambulatório de reabilitação". Desta forma, a segunda transformação atualiza o modelo jPDL com as informações do modelo, conforme apresentado na Figura 6.48.

```
//Notificação de encaminhamento para o ambulatório de reabilitação
<mail name="Notificação de encaminhamento para o ambulatório de
reabilitação">
<to addresses=Enfermeira chefe/>
<subject>Notificação de encaminhamento para o ambulatório de
reabilitação </subject>
<text>TODO</text>
<transition to=" DECISIONEncaminhar_paciente_para_ambulatorio"/></mail>
```

Figura 6.48 – Adicionando a entidade organizacional no elemento notificação.

6.6 ETAPAS MANUAIS PARA O DESENVOLVIMENTO DO SISTEMA DE WORKFLOW

As implementações das transformações dependentes de plataforma, apresentadas neste capitulo, produziram o documento jPDL necessário para descrever o processo que será automatizado. No entanto, isso não é suficiente para a conclusão do desenvolvimento do sistema de workflow.

Afinal ainda é necessário: **(i)** desenvolver as telas que serão utilizadas pelos usuários, **(ii)** criar a base de dados e **(iii)** implementar as regras de negócio no sistema computacional em desenvolvimento. Desta forma, a equipe, responsável por desenvolver o sistema computacional, terá que implementar manualmente os itens **(i)**, **(ii)** e **(iii)** citados anteriormente.

Para complementar o restante do sistema computacional que visa automatizar o processo de negócio "Triar Paciente para Admissão no Setor" foi necessário criar todas as telas das atividades que serão gerenciadas por usuários, criar a base de dados, desenvolver as classes em Java que representam as atividades automatizadas e atualizar o documento jPDL com informações diversas (caminho da classe que representa as atividades automatizadas, caminhos das páginas que representam as telas dos usuários, entre outras).

É importante ressaltar que as implementações das transformações dependente de plataforma apresentadas neste trabalho apenas exemplificaram o uso da semântica contida nos modelos arquiteturais e como elas podem ser usadas para o desenvolvimento de sistemas computacionais. É possível o desenvolvimento de diversas outras transformações que explorem de forma diferente as informações contidas nos modelos arquiteturais.

Por exemplo, é possível o desenvolver uma transformação no nível dependente de plataforma que produza códigos que descrevam uma ou mais regras de negócio, que devem estar contidos na aplicação, tendo como fonte as informações contidas dentro dos elementos *Business Rule* associados as atividades (*Function*) do processo de negócio.

6.7 WORKFLOW DO PROCESSO “TRIAR PACIENTES PARA ADMISSÃO NO SETOR”

Esta seção é dedicada à demonstração da utilização do sistema de workflow desenvolvido através das transformações independentes, dependentes de plataforma e com o complemento manual do desenvolvedor. A Tabela 6.2 apresenta as atividades utilizadas na demonstração.

Tabela 6.2 - Atividades Utilizadas na Demonstração.

Atividade	Responsável
Buscar Ficha do Paciente	Recepcionista
Examinar o Paciente	Médico
Verificar realização prévia de exames laboratoriais	Médico
Solicitar exames laboratoriais	Sistema de solicitação de exames

O sistema de workflow inicia quando o Recepcionista informa o identificador do paciente, conforme demonstrado na Figura 6.49. Após o Recepcionista clicar no botão “Enviar Prontuário” o sistema automaticamente busca o prontuário médico do paciente na base de dados e envia para o médico que irá atender o paciente. Esses passos são realizados na atividade “Buscar Ficha do Paciente”.

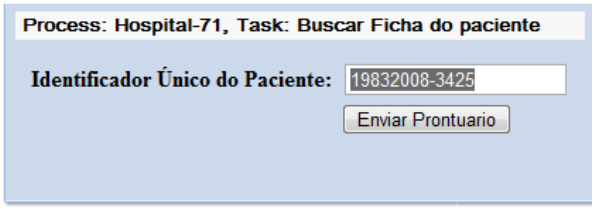


Figura 6.49 – Tela da atividade “Buscar ficha do paciente”.

Na atividade “Examinar o paciente", o sistema apresenta ao Médico todo o histórico do paciente que está sendo examinado. Nessa atividade o Médico cadastra no sistema os novos sintomas do paciente e informa se existe a necessidade de realizar exames laboratoriais por parte do paciente. A Figura 6.50 apresenta a tela da atividade “Examinar paciente”.

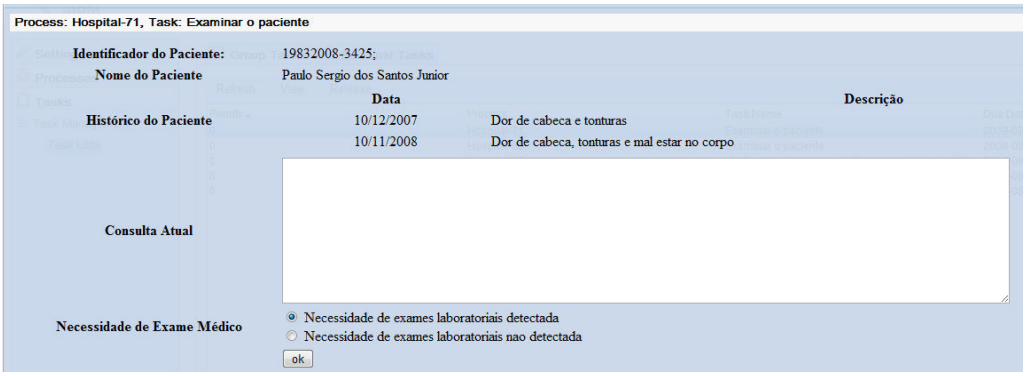


Figura 6.50 – Tela da atividade "Examinar o paciente".

Caso o Médico verifique a necessidade da realização de exames laboratoriais o sistema de workflow encaminha o Médico para a atividade "Verificar realização prévia de exames laboratoriais". Nessa atividade o Médico visualiza se o paciente já realizou algum exame laboratorial previamente. A Figura 6.51 apresenta a tela da atividade "Verificar realização previa de exames laboratoriais".

Caso o Médico queira que o Paciente realize algum exame que não foi realizado previamente, ele informa o nome do exame no campo "Novo Exame", seleciona a opção "Exames laboratoriais não realizados previamente" e clica no botão "Ok".

Por fim, o sistema de workflow executa a atividade automatizada "Solicitar exames laboratoriais" que solicita automaticamente o exame no laboratório para o paciente, através do "Sistema de Exames Laboratoriais" e registrar a consulta no "Sistema de Controle de Paciente". A Figura 6.52 apresenta log da execução da atividade "Solicitar exames laboratoriais".

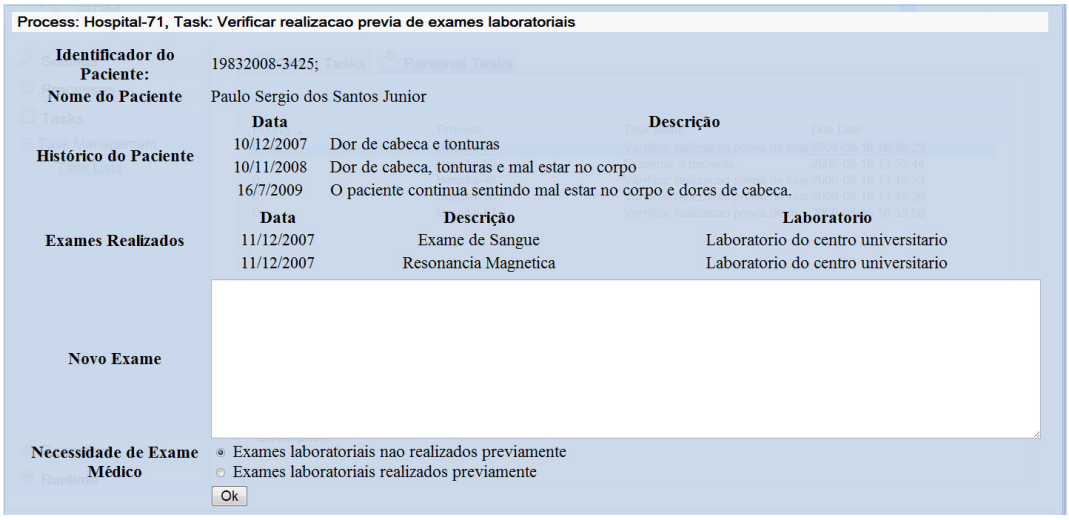


Figura 6.51 – Tela da atividade "Verificar realização previa de exames laboratoriais".

```
16:41:21,955 INFO  [STDOUT] ##########################################################
#####
16:41:21,957 INFO  [STDOUT] #Estabelecendo comunicacao com o sistema do laborat
rio
16:41:21,959 INFO  [STDOUT] #Comunicacao estabelecida
16:41:21,960 INFO  [STDOUT] #Enviando solicitacao de exame do paciente
16:41:21,961 INFO  [STDOUT] #Solicitacao enviada!
16:41:21,962 INFO  [STDOUT] #Estabelecendo comunicacao com o sistema do hospita

16:41:21,964 INFO  [STDOUT] #Registrando folha de faturamento no sistema do pac
ente
16:41:21,965 INFO  [STDOUT] #Registro realizado!
16:41:21,966 INFO  [STDOUT] ##########################################################
#######
```

Figura 6.52 – Log de execução da atividade "Solicitar exames laboratoriais".

Após a execução da atividade "Solicitar exames laboratoriais" a instância do processo "Triar pacientes para admissão no setor" é finalizado. A Figura 6.53 apresenta o fluxo de execução para essa instância do processo. As setas pretas indicam as atividades que foram executadas ao longo do sistema de workflow.

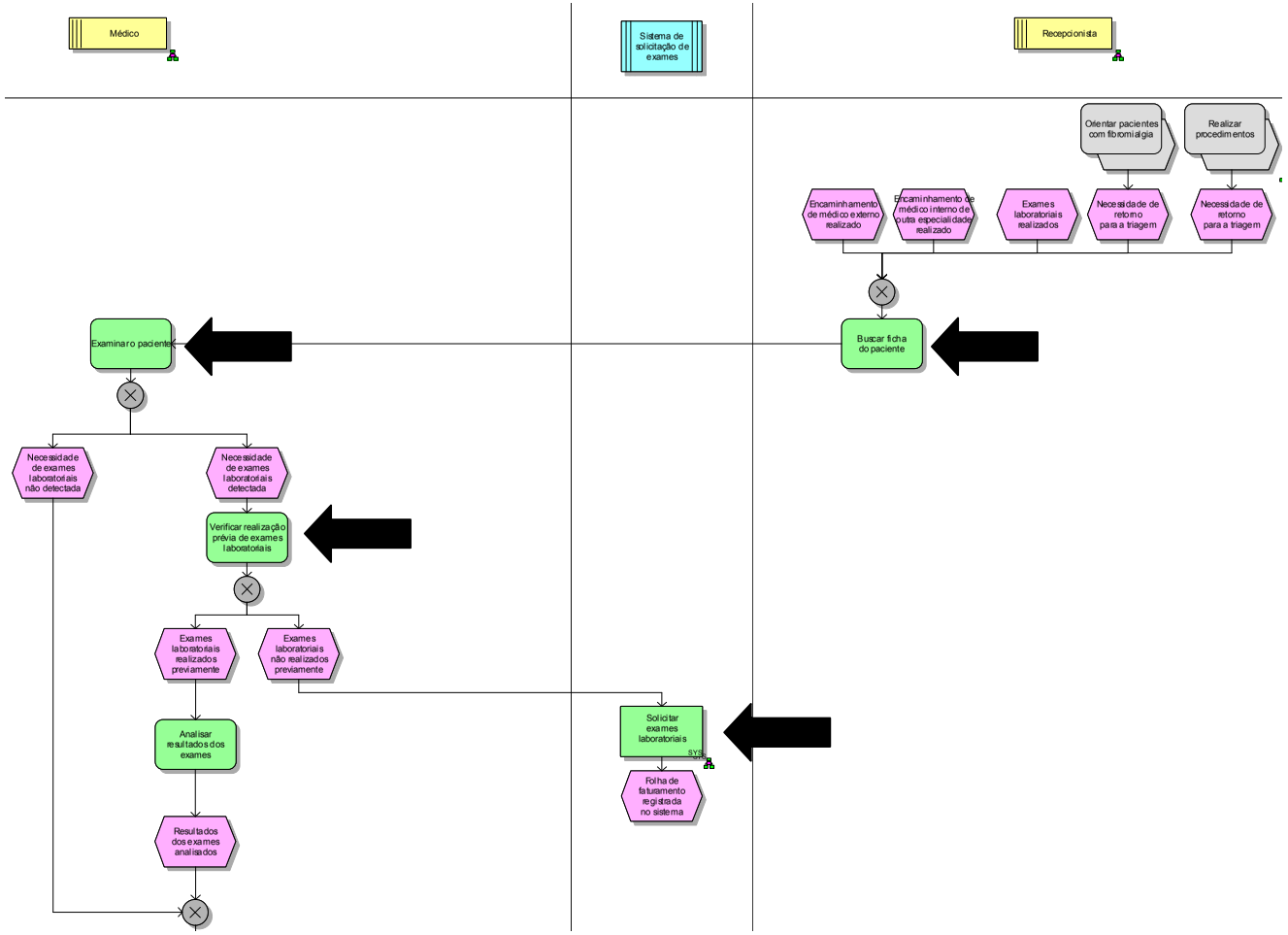


Figura 6.53 - Fragmento do processo de negócio.

6.8 CONCLUSÃO

Este capítulo apresentou como a abordagem de Desenvolvimento Orientado a Modelos proposta (Capítulo 5) pode ser utilizada para o desenvolvimento de sistemas computacionais orientados a processo de negócio.

Para a demonstração da abordagem proposta, as transformações (independente e dependente de plataforma) foram implementadas e aplicadas sobre os modelos de Processo de Negócio, Organizacional e Detalhamento de Processo de Atividade que descrevem o processo "*Triar pacientes para admissão no setor*", com o objetivo de desenvolver um sistema computacional que automatize tal processo de negócio. Desta forma, é comprovada a viabilidade da abordagem e demonstrado como os diferentes modelos podem ser utilizados nas etapas de desenvolvimento de um sistema computacional.

Ao implementar as transformações também foi demonstrado como o entendimento da semântica dos elementos de modelagem pode ser útil na construção e no resultado esperado destas.

Também foi demonstrado no capítulo, como as decisões de projeto são aplicadas nas transformações através das parametrizações, e assim, gerando o sistema computacional conforme requerido.

Trabalhos futuros poderão se concentrar na definição de transformações adicionais que sejam baseadas na utilização de modelos que descrevam as interfaces entre os atores e o sistema computacional ou em modelos conceituais de dados para gerar a base de dados utilizada pelo sistema computacional.

CAPÍTULO 7 CONCLUSÕES E TRABALHOS FUTUROS

7.1 CONCLUSÕES

Este trabalho apresentou uma abordagem de Desenvolvimento Orientado a Modelos que utiliza modelos das Arquiteturas Organizacionais de TI para desenvolvimento de sistemas computacionais orientados a processo. Diferentemente da maioria das abordagens atuais de Desenvolvimento Orientado a Modelos, a abordagem proposta neste trabalho permite ao projetista: **(i)** utilizar a semântica dos vários modelos que representam uma Arquitetura Organizacional de TI; **(ii)** aplicar as decisões de projeto sobre os modelos utilizados, através da definição de parametrizações nas transformações pré-definidas; e **(iii)** dividir o processo de desenvolvimento do sistema computacional em etapas independente e dependente de plataforma.

A utilização da semântica (bem definida) dos elementos das linguagens de modelagem possibilitou um melhor uso das informações contidas nos modelos. Através dessas transformações o projetista pode explorar a riqueza das informações contidas nos modelos e aplicar decisões de projeto sobre tais informações. Por exemplo, foram utilizadas informações relativas ao comportamento de processos, à estrutura organizacional e ao detalhamento de atividades (quem podem incluir a descrição das informações necessárias e produzidas pela atividade, os responsáveis pela execução da atividade).

As decisões de projeto são aplicadas sobre os modelos através das parametrizações das transformações pré-definidas. Desta forma, a abordagem proposta permite ao projetista uma maior flexibilidade nas transformações. Para que o desenvolvimento do sistema computacional seja inteiramente orientado a modelos, foi desenvolvimento um metamodelo (seção 6.2.2) que permite ao projetista desenvolver modelos que selecionam e parametrizam as transformações definidas na abordagem.

A divisão do processo de Desenvolvimento Orientado a Modelos nos níveis independente e dependente de plataforma possibilita ao projetista introduzir decisões de projeto em perspectivas diferentes.

No nível independente de plataforma, o projetista utiliza as decisões de projeto com a finalidade de construir um conjunto de modelos que descrevam o comportamento, os atores e os detalhes de cada atividade do sistema computacional sem levar em consideração as tecnologias e a infra-estrutura envolvida no desenvolvimento do sistema. Desta forma, a partir dos modelos gerados é possível descrever as partes essenciais de um sistema computacional orientado a processo e, assim, possibilitar o desenvolvimento de diversos sistemas computacionais baseado nesses modelos.

No nível dependente de plataforma, o projetista utiliza as decisões de projeto para transformar os modelos gerados no nível independente de plataforma em modelos que concretizem o sistema computacional orientado a modelos. Por exemplo, neste trabalho as transformações no nível dependente de plataforma possibilitaram a construção de um modelo jPDL a partir dos modelos gerados no nível independente de plataforma, conforme demonstrado no Capítulo 6.

A construção de modelos corretos e transformações corretas durante o processo de desenvolvimento orientado a modelos proposto neste trabalho, somente foi possível devido ao estudo aprofundado da semântica dos elementos e relacionamentos presentes nos modelos utilizados no processo de desenvolvimento. Como demonstrado no Capítulo 4, o estudo aprofundado da semântica dos modelos do ARIS possibilitou a identificação de problemas na linguagem que poderiam interferir em uma transformação automatizada. O estudo aprofundado da semântica dos elementos de modelagem também teve impacto na área da modelagem organizacional, afinal, com a semântica dos elementos bem definida tornou-se possível a construção de modelos corretos, ou seja, que representam as diversas situações do mundo real de forma correta e sem ambigüidade. O estudo semântico realizado neste trabalho é complementar aos estudos que concentram-se na semântica formal (de execução) de EPCs (como apresentado em(VAN DER AALST, 1998)(KINDLER, 2006)).

Por fim, de forma sucinta, pode-se listar as seguintes contribuições deste trabalho:

i. **Entendimento dos conceitos dos domínios Organizacional, de Processo de Negócio e de Detalhamento de Atividades presentes no ARIS**: Através dos metamodelos das linguagens dos domínios organizacionais, de processo de

negócio e de detalhamento de atividades, que estão implementadas nas ferramentas ARIS, foi possível uma melhor compreensão dos conceitos de cada um desses domínios proposto pelo ARIS Method. Também vale ressaltar, que os metamodelos de domínio propostos neste trabalho são os mais atuais publicamente conhecidos; e esses modelos também podem ser utilizados para desenvolvimento de aplicações que manipulem as informações dos modelos de domínio organizacional, processo de negócio e detalhamento de atividades, como demonstrado em (PIUMBINI, 2009).

ii. **Análise Ontológica das linguagens dos domínios de Processo de Negócio, Organizacional e Detalhamento de Atividades:** Através da análise ontológica dos elementos dos metamodelos, das linguagens de domínio escavadas, foi possível estabelecer uma relação entre esses elementos e as entidades do mundo real com base em uma ontologia de fundamentação. A análise ontológica também possibilitou identificar erros de semântica nas linguagens de domínio do ARIS Method. Além disso, também foi possível a eliminação de erros nas linguagens de domínio que afetavam a clareza, expressividade, completude, parcimônia, lucidez, correção e consistência dos modelos construídos como ARIS Method.

iii. **Maior detalhamento dos conceitos sociais da UFO-C:** Durante a elaboração da análise ontológica foi necessário realizar um estudo profundo dos conceitos sociais presentes na UFO-C. O estudo possibilitou um melhor detalhamento sobre os conceitos de organização, agentes sociais, papéis sociais, contratos sociais e de outros conceitos que estão envolvidos no contexto de uma organização;

iv. **Abordagem de Desenvolvimento Orientado a Modelos utilizando modelos das Arquiteturas Organizacionais de TI:** Apresentamos uma abordagem de Desenvolvimento Orientado a Modelos que difere das demais abordagens atuais por permitir ao projetista aplicar as decisões de projeto, dividir o processo de desenvolvimento em duas etapas bem definidas (independente e dependente de plataforma) e utilizar a semântica dos modelos arquiteturais de forma mais adequada no processo de construção de um sistema computacional;

7.2 TRABALHOS FUTUROS

Os trabalhos futuros se concentrarão principalmente: **(i)** na aplicação e no aperfeiçoamento da abordagem de Desenvolvimento Orientada a Modelos proposta neste trabalho; **(ii)** no desenvolvimento de um ambiente computacional que implemente completamente a abordagem proposta e; **(iii)** no desenvolvimento de técnicas que permitam evoluir o sistema computacional tendo como base as mudanças nos processos de negócio e/ou Arquitetura Organizacional de TI da organização .

O ambiente computacional permitiria ao projetista:

i. aplicar as transformações da abordagem de forma automática, aumentando a produtividade no desenvolvimento de sistemas computacionais;

ii. visualizar todo o processo de desenvolvimento do sistema computacional e, desta forma, prever o resultado das transformações e suas conseqüências no sistema computacional em desenvolvimento;

iii. permitir que o projetista desenvolva o sistema computacional em diversas plataformas de realização, apenas tendo como base os modelos gerados pelas transformações no nível independente de plataforma. Por exemplo, a partir de um mesmo conjunto de modelos gerar diversos sistemas computacionais em BPEL (OASIS, 2006), Java e jBPM.

Em relação ao aperfeiçoamento da abordagem de Desenvolvimento Orientado a Modelos, proposta neste trabalho, vislumbramos o desenvolvimento de:

i. Transformações que utilizem outros modelos das Arquiteturas Organizacionais de TI como, por exemplo, diagramas da UML, diagramas de banco de dados, modelos de interface com usuário, modelos de segurança de informação e outros modelos que sejam passíveis de transformação automática e venham contribuir para o desenvolvimento de sistemas computacionais;

ii. Estudos de transformações (e regras para transformações) que permitam ao projetista:

a. aplicar conceitos de abstração e refinamento nas entidades organizacionais que estão presente nos modelos de processos de negócio e vislumbrar quais serão as modificações nos modelos comportamentais do sistema computacional;

b. construir um modelo comportamental do sistema **c**omputacional a partir de dois ou mais processos de negócios;

c. aplicar conceitos nos modelos comportamentais que não estejam presentes nas linguagens de processo de negócio nativas das Arquiteturas Organizacionais de TI. Por exemplo, aplicar o conceito de multi-instância que está presente em BPMN (BPMN, 2009) nos modelos EPC (SCHEER, 1999); e

d. verificar os impactos de mudanças no processo de negócio, de requisitos e/ou decisões de projeto sobre um sistema computacional desenvolvido ou em desenvolvimento.

REFERÊNCIAS

ALLEN, J. F. **Maintaining Knowledge About Temporal Intervals**. Communications of the ACM, Vol.26, n. 11, 1983.

ALMEIDA, J. P. A. **Model-Driven Design of Distributed Applications**. Ph.D. Thesis in Computer Science. Enschede: Telematica Instituut Fundamental Research Series, 2006. ISBN 90-75176-422.

ALMEIDA, J. P. A. et al. **Model Driven Design, Refinement and Transformation of Abstract Interactions**. International Journal of Cooperative Information Systems (IJCIS). Vol.15. n. 4. 2006b. 599-632.

ALMEIDA, J. P. A.; GUIZZARDI, G.; SANTOS JR., P. S. **Applying and Extending a Semantic Foundation for Role-Related Concepts in Enterprise Modelling**. Enterprise Information Systems.Vol. 3. 2009. pag. 253-277.

ALMEIDA, J. P. A.; VAN ECK, P.; IACOB, M. E. **Requirements Traceability and Transformation Conformance in Model-Driven Development**. Tenth IEEE International EDOC Conference (EDOC 2006). 2006a. pag.355-366.

ALTER, S. **Information Systems: A Management Perspective**. Addison-Wesley, 1999.

ARCHIMATE CONSORTIUM. **Archimate Resource Tree**, 2009. Disponível em: <http://www.telin.nl/NetworkedBusiness/Archimate/ART>. Acesso em: 20 janeiro 2009.

AZEVEDO, L. G. et al. **Identificação de Serviços a partir da Modelagem de Processos de Negócio**. V Simpósio Brasileiro de Sistema de Informação (SBSI). 2009.

BALDINGER, K. **Semantic Theory: Towards a Modern Semantics**. Palgrave Macmillan. 1980.

BOTTAZZI, E.; FERRARIO, R. **Preliminaries to a DOLCE Ontology of Organizations**. J. Business Process Integration and Management. Vol.1. 2006. pag.64–74.

BPMN. BPMN - **Business Process Modeling Notation**, 2009. Disponível em: <http://www.omg.org/spec/BPMN/1.1/PDF>. Acesso em: 2009 Janeiro 8.

CARDOSO, E. C. S. **Uma Comparação entre Requisitos de Sistemas Gerados por Técnicas de Modelagem de Processos com Requisitos de Sistemas Gerados por**

Técnicas Convencionais de Engenharia de Requisitos. 2007. Monografia de Conclusão de Curso, Universidade Federal do Espírito Santo.

CARDOSO, E. C. S. **Towards the Alignment of Enterprise Architecture Models and Goal Models**. Vitória: 2009. Dissertação de Mestrado. Universidade Federal do Espírito Santo.

CARDOSO, E. C. S.; ALMEIDA, J. P. A.; GUIZZARDI, G. **A Requirements Engineering Experiment based on Process Models (Uma Experiência com Engenharia de Requisitos baseada em Modelos de Processos – in portuguese)**. XI Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS'2008), Recife, 2008.

CONTE, R.; CASTELFRANCHI, C. **Cognitive and Social Action. Londres**: UCL Press, 1995.

CUMBERLIDGE, M. **Business Process Management with JBoss jBPM: A Practical Guide for Business Analysts**. Olton,Birmingham,UK: Packt Publishing, 2007.

DAVIS, R. **Business Process Modeling with ARIS - A Pratical Guide.** Springer, 2001.

DIGNUM, V. **A Model for Organization Interaction: Based on Agents, Founded in Logic**. The Netherland, 2004. Ph.D. Thesis.

DOCKHORN COSTA, P.; GUIZZARDI, G.; ALMEIDA, J. P. A. **Situations in Conceptual Modeling of Context**. Proceedings of the IEEE EDOC 2nd International.Ontologies and Rules for The Enterprise (VORTE'06). Hong Kong, China. Outubro. 2006.

DODAF. **DODAF - DoD Architecture Framework Version 1.5**. Vol. II: Product Descriptions, 2007.

DUMAS, M.; VAN DER AASLT, W. M. P.; TER HOFSTEDE, A. H. M. **Process-Aware Information Systems: Bridging people and software through process technology**. Book Reviews. ed. 2007. DOI= http://dx.doi.org/10.1002/asi.v58:3.

ECLIPSE, F. **Generating an EMF Model using XML Schema (XSD)**. 2008b. Disponível em: <http://eclipse.org/modeling/emf/docs/2.x/tutorials/xlibmod/xlibmod_emf2.0.html>. Acesso em: 15 maio 2008.

FELHBERG, G. **Interoperabilidade Semântica de Aplicações Comercialmente Disponíveis Usando Ontologias de Domínio**. Monografia de Conclusão de Curso. Universidade Federal do Espírito Santo, 2007.

FERDINAND, M.; ZIRPINS, C.; TRASTOUR, D. **Lifting XML Schema to OWL**. Web Engineering. 4th International Conference, ICWE 2004, LNCS 3140, 2004. pag. 26-30.

FILIPOWSKA, A. et al. **Organisational Ontology Framework for Semantic Business Process Management**. Proceedings of the 12th International Business Information Systems Conference (BIS 2009), Poznan, Polonia. 2009a. pag. 1-12.

FILIPOWSKA, A. et al. **Organizational Ontologies to Support Semantic Business Process Management**. SBPM, 2009b.

GÄRDENFORS, P. **Conceptual Spaces: the Geometry of Thought**. USA: MIT Press , 2000.

GONZÁLEZ, J.; DÍAZ, J. **Business process-driven requirements engineering: a goal-based approach**. In Proc. of 8th Workshop on Business Process Modelling.Development and Support (BPMDS'07), 2007.

GREEN, P. F.; ROSEMANN, M. **Ontological Analysis of Business Systems Analysis Techniques: Experiences and Proposals for and Enhanced Methodology**. Business Systems Analysis with Ontologies, 2005. 1-27.

GREEN, P.; DAVIES, I. G.; ROSEMANN, M. **Exploring proposed ontological issues of ARIS with different categories of modellers**. Proceedings of the Australasian Conference on Information Systems, Hobart, Australia, 2004.

GREEN, P.; INDULSKA, M.; AND ROSEMANN, M. **A Reference Methodology for Conducting Ontological Analyses**. A Reference Methodology for Conducting Ontological Analyses. In Proceedings of the 23rd International Conference on Conceptual Modelling,ER 2004, Shanghai, China, 2004b.

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

GUARINO, N. **Formal Ontology and Information Systems**. Formal Ontologyin Information Systems, Proceedings of the International Conference on Formal Ontology and Information Systems (FOIS), Trento, Italia. 1998. pag.3-15.

GUARINO, N.; GIARETTA, P. **Ontologies and Knowledge Bases: Towards a Terminological Clarification**. Amsterdam: IOS Press, 1995. pag. 25-32.

GUIDE BUSINESS RULE. Business Rules Group, 2008. Disponível em: <http://www.businessrulesgroup.org/home-brg.shtml>.

GUIZZARDI, G. **Ontological Foundations for Structural Conceptual Models**. Ph.D. Thesis, University of Twente,The Netherlands, 2005a.

GUIZZARDI, G. **On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models**. Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV, Amsterdam, 2007.

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.; FERREIRA PIRES, L.; VAN SINDEREN, M. **An Ontology-Based Approach for Evaluating the Domain Appropriateness and Comprehensibility Appropriateness of Modeling Languages**. ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Vol. 3713, 2005a.

GUIZZARDI, G.; WAGNER, G. **Towards Ontological Foundations for Agent Modeling Concepts using UFO**. Agent-Oriented Information Systems II, Lecture Notes on Artificial Intelligence (LNAI), 3508, 2005b.

GUIZZARDI, R. S. S. et al. **Towards an Ontological Account of Agent Oriented Goals,Software Engineering for Multi-Agent Systems**, Berlin, Vol. V, 2007a.

HAREL, D.; RUMPE, B. **Modeling Languages: Syntax, Semantics and All That Stuff, Part1: The Basic Stuff.** The Weizmann Institute of Science, Israel, 2000.

HELLER, B.; HERRE, H. **Ontological categories in gol: Axiomathes**. Kluwer Academic Publishers, 2004a. pag. 57-76.

HELLER, B.; HERRE, H. **General ontological language GOL: A formal framework for building and representing ontologies**, Leipzig, Alemanha, 2004b.

HERRE, H. et al. **General Formal Ontology (GFO): a foundational ontology integrating objects and processes, Part 1: Basic Principals**. Onto-Med Report 8, University of Leipzig, 2006..

HSI, I. **Analyzing the Conceptual Integrity of Computing Applications Through Ontological: Excavation and Analysis**. Georgia Institute of Technology, 2005. Dissertação de Doutorado.

IBM. XML **Import tool from ARIS to IBM WebSphere Business Modeler**, 2008. Disponível em: <http://www.alphaworks.ibm.com/tech/arisimport/requirements>. Acesso em: 23 Julho 2008.

JABLONSKI, S.; BUSSLER, C. **Workflow Management: Modeling Concepts, Architecture, and Implementation**. Londres, Inglaterra. International Thomson Computer Press, 1996. ISBN 1850322228/978-1850322221.

JONKERS, H. et al. **Concepts for modeling enterprise architectures**. Special issue on Architecture in IT of the International Journal of Cooperative Information Systems, setembro 2004. 257--287.

KERN, H.; KÜHNE, S. **Model Interchange between ARIS and Eclipse EMF**. The 7th OOPSLA Workshop Domain-Specific Modeling, 2007.

KINDLER, E. **On the Semantics of EPCs: A Framework for Resolving the Vicious Circle**. Technical Report. Computer Science Department, University of Paderborn, Alemanha, 2006.

KOHLBORN, T. et al. **Identification and Analysis of Business and Software Services**: **A Consolidated Approach**. IEEE Transactions on Services Computing, vol. 2, 2009. pp. 50-64. doi:10.1109/TSC.2009.6.

KURTEV, I.; BÉZIVIN, J.; AKSIT, M. **Technological spaces: An initial appraisal**. CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, USA, 2002.

LANKHORST, M. **Enterprise Architecture at Work - Modelling, Communication and Analysis**. Springer, 2005.

LANKHORST, M.; VAN DRUNEN, H. **Enterprise Architetcture Development and Modelling**. Via-nova-architectura, Março 2007. Disponível em: <www.via-nova-architectura.org>. Acesso em: 2008 ago. 20.

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

LEYMANN, F.; ROLLER, D. **Production Workflow: Concepts and Techniques.** NJ: Prentice- Hall PTR, Upper Saddle River, 1999.

LUBKE, D. et al. **Using event-driven process chains for model-driven development of business application**. International Journal of Business Process Integration and Management, Vol. 3, N. 2, pag. 109-117, 2008.

MASOLO, C. et al. **Ontology Library**. WonderWeb Deliverable D18, 2008.

MENDLING. **EPML**, 2009. Disponível em: <http://www.mendling.com/EPML/index2.html>. Acesso em: 12 setembro 2009.

MENDLING, J.; NEUMANN, G.; NÜTTGENS, M. **Towards Workflow Pattern Support of Event-Driven Process Chains (EPC)**. Proceedings of the 2nd GIWorkshop XML4BPM at the 11th GI Conference BTW, 2005a.

MENDLING, J.; NEUMANN, G.; NÜTTGENS, M. **Yet Another Event-Driven Process Chain (Extended Version): Technical Report**. Enterprise Modelling and Information Systems Architectures - International Journal (EMISA Journal), Outubro 2005b. pag. 3-13.

MENDLING, J.; ZIEMANN, J. **Transformation of BPEL Processes to EPCs**. Proc. of the 4th GI Workshop on Event-Driven Process Chains, Hamburg, 167 , Dezembro 2005. 41-53.

MENDLINGY, J.; NÜTTGEN, M. T**ransformation of ARIS Markup Language to EPML**. Proc. of the 3rd GI Workshop on Event-Driven Process Chains, outrubro 2004.

MORRIS, C. W. **Foundations of a theory of signs. International Encyclopedia of Unified**, Chicago, 1938. 77-138.

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

OASIS. **Web Services Business Process Execution Language**, 2006. Disponível em: <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html>. Acesso em: 21 agosto 2009.

OMG. **MDA Guide Version 1.0.1**, junho 2003. Disponível em: <http://www.omg.org/docs/omg/03-06-01.pdf>. Acesso em: 10 maio 2008.

OMG. **Object Constraint Language**, 2006. Disponível em: <http://www.omg.org/docs/ptc/05-06-06.pdf>. Acesso em: 10 abril 2008.

OUYANG, C. et al. **From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way**. ACM Transactions on Software Engineering and Methodology, 2008.

PIUMBINI, M. T. **Acesso a Modelos do ARIS Toolset: Uma abordagem Orientada a Modelos para Arquiteturas Organizacionais de TI**. Espírito Santo: Monografia. Universidade Federal do Espírito Santos, 2009.

RM-ODP-ISO-ISO/ITU-T. **Open Distributed Processing - Reference Model**. I. S. I. Open Distributed Processing - Reference Model., 1995.

ROSEMANN, M.; VAN DER AALST, W. M. P. **A Configurable reference modelling language. Inf. Syst.**, Março 2007. pag. 1-23. DOI= http://dx.doi.org/10.1016/j.is.2005.05.003.

RUSSELL, N.; HOFSTEDE, A. H. M. **WORKFLOW CONTROL-FLOW PATTERNS A Revised View**. Workflow Patterns, 2008. Disponível em: <http://www.workflowpatterns.com/>. Acesso em: 10 outubro 2008.

SANTOS JR., P. S.; ALMEIDA, J. P. A. **Escavando as Linguagens de Modelagem Organizacional e Modelagem de Processos de Negócio do ARIS Method**. Workshop em Gerência de Processos de Negócio (WBPM).WebMedia 2008 XIV Simpósio Brasileiro de Sistemas Multimídia e Web, Anais Artigos Resumidos e Artigos de Workshops., Vila Velha, ES, v.2, 2008. 225-232.

SANTOS JR., P. S.; ALMEIDA, J. P. A.; GUIZZARDI, G. **An Ontology-Based Semantic Foundation for ARIS EPCs**. 7 Enterprise Engineering, ACM SAC, Sierre, Suiça, 22-26 março 2010.

SANTOS JR., P. S.; ALMEIDA, J. P. A.; PIANISSOLLA, T. L. **Escavação, Refatoração e Análise de um Metamodelo para o ARIS Method**. 3rd Workshop on Ontologies and Metamodels in Software and Data Engineering, Campinas, São Paulo, 2008. 107-118.

SCHEER, A. W. **ARIS – Business Process Modeling**. Terceira Edição. ed. Springer, 1999.

SCHEER, I. **ARIS Platform: XML Export and Import**,White Paper, 2006b.

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

SHARP, A.; MCDERMOTT, P. **Workflow modeling: Tools for Process Improvement and Application Development**. 2000. ISBN 1-58053-021-4.

SPECHT, T. et al. **Modeling cooperative business processes and transformation to a service oriented architecture**. E-Commerce Technology, CEC 2005.Seventh IEEE International Conference, 2005. pag. 249- 256.

SUN. **Sun Microsystems**. Java.com, 2009. Disponível em: <http://www.java.com>. Acesso em: 10 abril 2009.

SUPER. **sBPMN and sEPC to BPMO Translation**. Version 1.0. 2008. http://www.ip-super.org/res/Deliverables/M24/D4.5.pdf.

SUPER. **Process Ontology Stack: Evolved Version**. 2009. http://www.ip-super.org/res/Deliverables/M36/D1.5.pdf.

THOMAS, O.; FELLMANN, M. **Semantic Event-Driven Process Chains**. Workshop SBPM, Budva, Montenegro, 2006.

THOMAS, O.; FELLMANN, M. **Semantic EPC: Enhancing Process Modeling Using Ontology Languages**. Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management, Innsbruck, Austria, 7 Junho 2007.

TOGAF, T. O. G. **The Open Group Architectural Framework (TOGAF 8.1.1 'The Book')**. 2008. Van Haren Publishing, 2007.

UML. **Unified Modeling Language (UML)**, Version 2. OMG, 2009. Disponível em: <http://www.omg.org/technology/documents/formal/uml.htm>. Acesso em: 20 agosto 2009.

VAN DER AALST, W. M. P. **Formalization and Verification of Event-driven Process Chains**. Computing Science Reports 98/01, Eindhoven University of Technology,Eindhoven, 1998.

VAN DER AALST, W. M. P.; JABLONSKI, S. **Dealing with Workflow Change: Identification of Issues and Solutions**. International Journal of Computer Systems, Science, and Engineering, pag. 267–276, 2000.

VAN DER AALST, W. M. P.; VAN HEE, K. M. **Workflow Management: Models, Methods, and Systems**. The MIT Press, 2002. ISBN 0-262-01189-1/978-0-262-01189-1.

VAN DONGEN, B. F.; VAN DER AALST, W. M. P.; VERBEEK, H. M. W. **Verification of EPCs: Using Reduction Rules and Petri Nets.** Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05).Lecture Notes in Computer Science, Berllin, Vol. 3529, 2005. pag. 372-386.

VANHATALO, J.; VÖLZER, H.; LEYMANN, F. **Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition**. Proceedings of the 5th international Conference on Service-Oriented Computing, Vienna, Austria, 17-20 Setembro 2007.

VOLZ, R. et al. **OntoLiFT Prototype - WonderWeb: Ontology Infrastructure for the Semantic Web**, 2002.

W3SCHOOLS. **DTD Tutorial**, 2008. Disponível em: <http://www.w3schools.com/dtd/default.asp>. Acesso em: 2008 abril 10.

W3SCHOOLS. **Introduction to XML Schema**. W3Schools, 2008b. Disponível em: <http://www.w3schools.com/Schema/schema_intro.asp.>.

WEBER, R. **Ontological Foundations of Information System**. Coopers & Lybrand and the Accounting Association of Australia and New Zealand, Melboune, Australia, 1997.

WEBSTER. **Merriam-Webster Dictionary**, 2004. Disponível em: <www.m-w.com>.

ZACHMAN, J. A. **A Framework for Information Systems Architecture**. IBM Systems Journal, 26, 1987. 276–292.

ZIEMANN, J.; MENDLING, J. **EPC-Based Modelling of BPEL Processes: a Pragmatic Transformation Approach**. Proceedings of the 7th International Conference "Modern Information Technology in the Innovation Processes of the Industrial Enterprises" (MITIP 2005), Genova,Italia, Setembro 2005.

INDÍCE DE FIGURA

Figura 2.1 – Tipos de Ontologias de acordo com o nível de generalização (GUARINO, 1998).....17
Figura 2.2 – Fragmento da UFO-A. 20
Figura 2.3 – Fragmento da UFO-A com foco nos Universais. 24
Figura 2.4 – Fragmento da UFO-B. 26
Figura 2.5 - Exemplificando os Allen's Relations: relações temporais entre intervalos de tempo. 28
Figura 2.6 - Fragmento da UFO-C. 29
Figura 2.7 – Fragmento da UFO-C com foco na entidade Action. 31
Figura 2.8 - Fragmento da UFO-C que distingue as relações de Dependência, Delegação e Aquisição, (Guizzardi et al., 2007a). 33
Figura 2.9 – Redução do espaço de soluções para um projeto com diferente níveis de abstração (ALMEIDA et al., 2006a). 36
Figura 2.10 – Relação entre conformidade e transformações (ALMEIDA et al., 2006).37
Figura 2.11 – Conformidade e transformação (ALMEIDA *et al.*, 2006a). 38
Figura 2.12 – Gerenciador de fluxo de trabalho como link entre Pessoas e Sistemas computacionais (DUMAS *et al.*, 2007). 43
Figura 3.1 - Fragmento do metamodelo organizacional contendo Organization Unit, Organization Unit Type e seus relacionamentos. 47
Figura 3.2 – Símbolos do elemento Organization Unit. 48
Figura 3.3- Símbolos do elemento Organization Unit Type. 48
Figura 3.4 – Símbolo do elemento Position. 49
Figura 3.5 - Fragmento do Metamodelo Organizacional contendo os elemento Position, Position Type Organizacional Unit, Organizational Unit Type e seus relacionamentos. 49
Figura 3.6 – Símbolo dos elementos notacionais do elemento Person. 50
Figura 3.7 – Fragmento do Metamodelo Organizacional com foco no elemento Person. 51
Figura 3.8 – Símbolo do elemento Person Type. 52
Figura 3.9 - Fragmento do metamodelo organizacional com foco no elemento Person Type e seus relacionamentos. 53
Figura 3.10 – Símbolo do elemento Location. 54
Figura 3.11 – Fragmento do metamodelo organizacional com foco no elemento Location e em seus relacionamentos. 55
Figura 3.12 – Símbolo do elemento Group. 55
Figura 3.13 – Fragmento do metamodelo organizacional com foco no elemento Group e em seus relacionamentos. 56
Figura 3.14 – Fragmento do metamodelo organizacional com foco nos elementos System Organization Unit e System Organization Unit Type. 57
Figura 3.15 – Exemplo de Modelo Organizacional Construído Utilizando as Metaclasses do Metamodelo Organizacional 58
Figura 3.16 - Fragmento do metamodelo de processo com foco no elemento Function e seus relacionamentos com o elemento Participant e com o elemento Application. 59
Figura 3.17 - Símbolo do elemento Function. 59
Figura 3.18 – Símbolo do elemento Process Interface. 59
Figura 3.19 – Símbolo do elemento System function (actual). 60
Figura 3.20 – Símbolo do elemento Application System. 61
Figura 3.21 – Símbolo do elemento Application System Type. 61

Figura 3.22- Símbolo do elemento Application System Class. 62
Figura 3.23 – Símbolo do elemento Event. 62
Figura 3.24 – Fragmento do metamodelo de processo contendo os relacionamentos entre Function, Event e Rule. 63
Figura 3.25 – Exemplo o relacionamento is evaluated by entre Event e Rule. 65
Figura 3.26 – Exemplo do relacionamento activates. 65
Figura 3.27 – Exemplo do relacionamento links. 66
Figura 3.28 – Exemplo do relacionamento leads to. 66
Figura 3.29 – Mapeamento entre as entidades Participant e Function do metamodelo de processo de negócio e em suas instancias em um modelo de processo de negócio. 67
Figura 3.30 - Mapeamento entre as entidades Rule e Event do metamodelo de processo de negócio e suas instâncias em um modelo de processo de negócio. 67
Figura 3.31 - Fragmento do Metamodelo de Detalhamento de Atividade contendo Function, Cluster e Technical Term e seus relacionamentos. 68
Figura 3.32 - Fragmento do Metamodelo de Detalhamento de Atividade contendo Cluster, Participant, User e seus relacionamentos 69
Figura 3.33 – Símbolo do elemento Cluster. 69
Figura 3.34 – Símbolo do elemento Technical Term. 70
Figura 3.35 – Símbolo do elemento Information Carrier. 71
Figura 3.36 – Fragmento do metamodelo de detalhamento de atividade com foco no elemento Information Carrier. 72
Figura 3.37 – Fragmento do metamodelo de detalhamento de atividades com foco nos elementos Application System, Application System Type e Application System Class. 73
Figura 3.38 – Símbolo do elemento Business Rule. 74
Figura 3.39 – Fragmento do metamodelo de detalhamento de atividades apresentado o relacionamento do elemento Business rule. 74
Figura 3.40 - Mapeamento entre as entidades do metamodelo de Detalhamento de Atividades e suas instâncias em um modelo de detalhamento de atividades. 75
Figura 3.41 – Fragmento do elemento ARIS Object e o Elemento Assignments 76
Figura 3.42 – Fragmento do Metamodelo Organizacional. 77
Figura 4.1 – Abordagem da avaliação ontológica 81
Figura 4.2 – Fragmento da UFO-C contendo a relação entre Social Agent e Normative Description. 84
Figura 4.3 – Fragmento da UFO-C com Norm. 85
Figura 4.4 – Fragmento da UFO-C contendo o relacionamentos entre Normative Description e Social Role. 85
Figura 4.5 – Fragmento da UFO-C contendo o relacionamento entre Normative Description e Social Relator. 86
Figura 4.6 – Fragmento da UFO-C contendo Social Contract. 87
Figura 4.7 - Interpretação dos elementos Organization Unit Type, Organization Unit e Position seguindo os conceitos do ARIS Method. 89
Figura 4.8 - Fragmento do metamodelo organizacional com os elementos Position e Person. 89
Figura 4.9 - Fragmento da UFO-C contendo Institutional Agent, Human Agent e Artificial Agent (GUIZZARDI et al., 2007). 90
Figura 4.10 - Nova interpretação do elemento Position. 91
Figura 4.11 - Fragmento do Metamodelo Organizacional contendo somente Position Type, Organization Unit Type e Position 92
Figura 4.12 - Interpretação do elemento Position Type como um Social Role Mixin. 93

Figura 4.13 - Interpretação do Position Type como um High Order Universal. 93
Figura 4.14 - Fragmento do metamodelo organizacional contendo alguns relacionamentos do elemento Person, Position, Organization Unit e Organization Unit Type.....94
Figura 4.15 - Exemplo do relacionamento *is composed of*.....96
Figura 4.16 - Exemplo de utilização do relacionamento belongs to. 98
Figura 4.17 - Exemplo do relacionamento *is organization manager for*. 99
Figura 4.18 – Fragmento do metamodelo contendo os relacionamentos do elemento Person Type. 101
Figura 4.19 – Exemplo de relacionamento performs. 102
Figura 4.20 – Exemplo de relacionamento belongs to entre Person Type e Organization Unit. 102
Figura 4.21 – Interpretação do elemento Person Type e dos relacionamentos performs e belongs to. 104
Figura 4.22 – Fragmento do Metamodelo Organizacional com elemento Person Type e Position Description 106
Figura 4.23 - Interpretação do elemento Position Description. 107
Figura 4.24 - Interpretação do elemento Group. 111
Figura 4.25 - Interpretação do Location e dos relacionamentos at location, is located at e encompasses. 113
Figura 4.26 – Fragmento de um modelo organizacional apresentado em (SCHEER, 1999, p. 187). 115
Figura 4.27 – Fragmento da UFO-A, UFO-B e UFO-C. 117
Figura 4.28 – Fragmento do Metamodelo de Processo de Negócio com os relacionamento entre Particpant e Function. 120
Figura 4.29 - Interpretação ontológica do elemento Function. 120
Figura 4.30 – Interpretação do elemento Event. 123
Figura 4.31 – Interpretação dos elementos Application System e Application System Type. 125
Figura 4.32 – Interpretação do elemento Application System Class. 127
Figura 4.33 – Interpretação dos elementos Cluster, Technical Term e Information Carrier. 129
Figura 4.34 – Interpretação do elemento Business Rule. 132
Figura 5.1 – Abordagem de Desenvolvimento Orientado a Modelos utilizando artefatos de Arquitetura Organizacional de TI. 138
Figura 5.2 – Modelo representando as relações dos metamodelos do sistema orientado a processo de negócio 140
A Figura 5.3 apresenta a comparação entre as abordagens de Desenvolvimento Orientado a Modelos atuais e a abordagem proposta neste trabalho. Como pode ser observado, a abordagem proposta permite ao projetista produzir diversas soluções tecnológicas enquanto as abordagens atuais permitem a produção de um tipo de solução tendo com base os modelos das Arquiteturas Organizacionais de TI. 141
Figura 5.4 – Diferença entre a abordagem atuais e a abordagem proposta 141
Figura 5.5 – Exemplo de Componente. 144
Figura 5.6 – Exemplo de utilização das transformações no nível comportamental. 145
Figura 5.7 – Exemplo de aplicação da Transformação de abstração. 146
Figura 5.8 - Exemplo da transformação de abstração. 146
Figura 5.9 – Processo de utilizado para exemplificar abstração de padrões de processo. 147
Figura 5.10 – Criando componente a partir de padrões de processo. 147

Figura 5.11 – Processo de negócio após a abstração dos padrões Excluise Choice e Simple Merge. 148
Figura 5.12 – Caso extremo da transformação de abstração em padrões de processo. 148
Figura 5.13 – Transformação de abstração em atividades de responsáveis distintos. 150
Figura 5.14 – Indicando um papel para nova atividade abstraída. 151
Figura 5.15 - Exemplo de escolha dos elementos organizacionais pertencente a um papel (Person Type) do processo de negócio. 152
Figura 5.16 - Indicando um grupo para nova atividade abstraída. 152
Figura 5.17 – Grupo contendo os participantes das atividades abstraídas 153
Figura 5.18 – Exemplo de aplicação da transformação refinamento. 153
Figura 5.19 – Aplicando a transformação refinamento e criando um componente formado por mais de um padrão de processo. 154
Figura 5.20 – Transformação no nível de componente. 157
Figura 5.21 – Selecionando os elementos dos modelos de detalhamento de componente 159
Figura 5.22 – Modelo de Detalhamento de uma atividade utilizando a primeira e segunda regra. 159
Figura 5.23 –Transformação de automatização: Atividade totalmente automatizada. 161
Figura 5.24 – Transformação de automatização: automatização de atividades para responsáveis com mais de uma atividade. 162
Figura 5.25 - Transformação de automatização nível: Acompanhada/gerenciada. 162
Figura 5.26 – Transformação de automatização nível: Totalmente manual. 163
Figura 5.27 - Desvio do fluxo de informação. 163
Figura 5.28 - Exemplo de utilização da transformação organizacional com papel social 164
Figura 5.29 - Exemplo de definição de papéis do sistema. 165
Figura 6.1 – Triar pacientes para admissão no setor (Parte 1). 169
Figura 6.2 - Triar pacientes para admissão no setor (Parte 2). 170
Figura 6.3 – Implementação da Abordagem de Desenvolvimento Orientado a Modelos. 172
Figura 6.4 – Conjunto de transformações para criar os modelos de domínio organizacional, processo de negócio e detalhamento de atividades do ARIS Method. 174
Figura 6.5 – Metamodelo de Seleção e Parametrização de Transformações Independentes de Plataforma. 176
Figura 6.6 – Metamodelo da metaclasse SourceComponent. 177
Figura 6.7 – Metamodelo da metaclasse NewComponent. 178
Figura 6.8 – Plugin para desenvolvimento de modelos de seleção e parametrização de transformações independentes de plataforma. 179
Figura 6.9 – Transformação de Abstração modelado no plugin. 180
Figura 6.10 – Definindo o Source Component. 180
Figura 6.11 - Modelo SourceComponent.eepc construído no plugin. 180
Figura 6.12 – Transformação de Refinamento modelado no plugin. 181
Figura 6.13 – Definindo o Source Task. 181
Figura 6.14 – Modelo listando os elemento necessários para a atividade ABC. 182
Figura 6.15 – Definindo o Modelo FAD na Transformação de Detalhamento de Atividade. 182
Figura 6.16 – Definindo a primeira transformação a ser executada. 182
Figura 6.17 – Definindo a transformação sucessora. 183
Figura 6.18 – Transformações no nível dependente de plataforma deste trabalho. 183
Figura 6.19 – Arquitetura de Automatização das Transformações. 184

Figura 6.20 – Fragmento do metamodelo da linguagem jPDL.....186
Figura 6.21 – Criando o componente “Buscar ficha do paciente”.....188
Figura 6.22 – Criando o componente “Examinar o paciente”.....189
Figura 6.23 – Criando o componente totalmente automatizado “*Solicitar exames laboratoriais*”.....190
Figura 6.24 - Quinta transformação no nível de independente de plataforma.....191
Figura 6.25 – Criando o componente “Encaminhar paciente e agendar ambulatório”. 192
Figura 6.26- Criando o componente “Conceder alta”.....193
Figura 6.27 – Criando o componente “Encaminhar paciente para ambulatório”. 194
Figura 6.28 - Criando o componente “Explicitar o diagnóstico ao paciente”. 196
Figura 6.29 - Modelo de estrutural do sistema de workflow (Parte 1).....197
Figura 6.30 - Modelo de estrutural do sistema de workflow (Parte 2).....198
Figura 6.31 - Modelo Detalhado da atividade “Analisar resultados dos exames” parametrizado.....199
Figura 6.32 - FAD criado para o componente “Solicitar exames laboratoriais”. 200
Figura 6.33 – FAD criado para o componente “Encaminhar paciente para ambulatório”. 201
Figura 6.34 – Parametrização do modelo organizacional “Recepcionista”.....202
Figura 6.35 - Parametrização do modelo organizacional “Médico”. 203
Figura 6.36 – Paramtrização do modelo organizacional “Grupo do ambulatório de reabilitação”. 203
Figura 6.37 – Interpretação do responsáveis pelas atividades no jDPL. 206
Figura 6.38 – Interpretação do elemento Start Event no jPDL. 207
Figura 6.39 – Interpretação do elemento End Event no jPDL. 207
Figura 6.40 - Interpretação do elementos Function, System Function (Actual), Process Interfaces no jPDL. 208
Figura 6.41 - Interpretação do elementos Function e System Function Actual no jPDL. 209
Figura 6.42 - Interpretação do elemento XOR do ARIS Method no jPDL. 210
Figura 6.43 – Criando as inter-relações entre os elementos do processo de negócio. ..212
Figura 6.44 - Fragmento do modelo em jPDL do processo de negócio “Triar pacientes para admissão no setor. 212
Figura 6.45 – Modificação da atividade “Encaminhar paciente para ambulatório”. 214
Figura 6.46 - Notificações da atividade “Encaminhar paciente para ambulatório”.....215
Figura 6.47 - Transformação organizacional.....216
Figura 6.48 – Adicionando a entidade organizacional no elemento notificação. 216
Figura 6.49 – Tela da atividade “Buscar ficha do paciente”. 218
Figura 6.50 – Tela da atividade “Examinar o paciente”.....219
Figura 6.51 – Tela da atividade “Verificar realização previa de exames laboratoriais”. 220
Figura 6.52 – Log de execução da atividade “Solicitar exames laboratoriais”. 220
Figura 6.53 - Fragmento do processo de negócio. 221

ÍNDICE DE TABELA

Tabela 3.1 - Regras básicas do ARIS. 63
Tabela 3.2 – Outras regras do ARIS. 64
Tabela 3.3 – Elementos notacionais do Information Carrier. 71
Tabela 5.1 – Abstrações das atividades do Participante A. 146
Tabela 6.1 – Interpretação das regras de processo no jPDL. 209
Tabela 6.2 - Atividades Utilizadas na Demonstração. 218