UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ANA VITORIA PIAGGIO DE FREITAS

APSEE-Global: um Modelo de Gerência de Processos Distribuídos de Software

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Daltro José Nunes
Orientador

Porto Alegre, dezembro de 2005

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Freitas, Ana Vitoria Piaggio de

APSEE-Global: um Modelo de Gerência de Processos Distribuídos de Software / Ana Vitoria Piaggio de Freitas – Porto Alegre: Programa de Pós-Graduação em Computação, 2005.

155 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2005. Orientador: Daltro José Nunes

1. Processos de software. 2. Gerência de processos distribuídos. 3. Desenvolvimento distribuído de software. 4. Ambientes de engenharia de software centrados no processo. I. Nunes, Daltro José. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof . José Carlos Ferraz Hennemann

Vice-Reitor: Prof. Pedro Cezar Dutra Fonseca

Pró-Reitora de Pós-Graduação: Profa. Valquiria Linck Bassani

Diretor do Instituto de informática: Prof. Philippe Olivier Alexandre Navaux

Coordenador do PPGC: Flávio Rech Wagner

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

A Deus e a meu anjo da guarda, presentes em todos os momentos. Fontes de força e de ânimo.

Meu muitíssimo obrigado à minha mãe, Nelcy, e ao meu pai, Gildasio. Sem o amor, a dedicação e a confiança deles não seria o que sou, e não teria chegado até aqui. Agradeço imensamente à Carla, minha irmã e grande amiga, pelo apoio contínuo. Aos meus parentes, fica também o muito obrigado pelo incentivo.

A Lucas Albuquerque, meu amor, por estar sempre ao meu lado: na alegria e na tristeza, na riqueza e na pobreza, na saúde e na doença.

Agradeço ao meu orientador, Dr. Daltro Nunes, pelos ensinamentos e pelo incentivo. Aos colegas do grupo Prosoft (Guilherme Rangel, Lincoln Rabelo, Alessandra Dahmer, Antonio Terceiro e Paulo Júnior), agradeço as valiosas contribuições para este trabalho. Destaco aqui o papel essencial de Anderson Maia, companheiro de mestrado e amigo de verdade para todas as horas. Devo muito a ele pela conclusão deste trabalho. Meus agradecimentos também ao Sr. Heribert Schlebbe e ao Paulo Mello, pela implementação do trabalho. Estendo os meus agradecimentos a Rodrigo e Carla Reis, pela atenção e pelas valiosas informações.

Ao Laboratório de Sistemas Distribuídos da UFBa, meu berço acadêmico. Em especial, agradeço ao Dr. Flávio Assis, pela confiança, pelas lições e pela amizade.

Aos muitos amigos que fiz durante a minha estadia em Porto Alegre. Serei eternamente grata aos amigos do Hotel Ritz, da Turma Mística e do Centro de Tradição Nordestina. A Izi Sena e Anderson Spiger, com quem dividi apartamento. Deixo aqui um muito obrigado especial à Deise Côrtes e Andreia Tres, por terem tão generosamente me acolhido em casa durante o percurso. Aos "colegas da sala ao lado": Igor Steinmacher, Tiago Telecken, Níccholas Vidal, Maximira André, Leila Rossi, Montgomery França. Agradeço também aos muitos amigos que estiveram ao meu lado, apesar da distância física. Em especial, agradeço a Nivea Ferreira, que mesmo em Liverpool sempre foi amiga próxima e presente.

A Carla e Martha Echenique, pelo suporte durante os meus altos e baixos. Minha gratidão a Mônica Latorre, Júlia Magdalena, Leonel Vagner e Marcelo Camara, pelo crescimento pessoal que me proporcionaram.

Ao CNPq, pelo suporte financeiro, e ao Instituto de Informática da UFRGS, pela estrutura oferecida.

E, por fim, agradeço a mim mesma, pela dedicação na realização deste trabalho.

A vida é assim: esquenta e esfria,
aperta e daí afrouxa,
sossega e depois desinquieta.
O que ela quer da gente é coragem.
Guimarães Rosa

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ATO	Ambiente de Tratamento de Objetos
CASE	*Computer-Aided Software Engineering* – Engenharia de Software Auxiliada por Computador
CMMI^{SM}	*Capability Maturity Model Integration* – Integração de Modelos de Maturidade da Capacidade
FIPS	*Federal Information Processing Standards* – Padrões federais para processamento de informação
GG	Gramática de Grafos
GSI	Gerência de Sistemas de Informação do Programa Nacional de Crédito Fundiário
ICS	Interface de Comunicação do Sistema
IDEF0	*Integration Definition for Function Modeling* – Definição de integração para modelagem de funções
MSF	*Microsoft Solutions Framework* – *Framework* de Soluções Microsoft
NAC	*Negative Application Condition* – Condição Negativa de Aplicação
PMI	*Project Management Institute* – Instituto de Gerenciamento de Projetos
PNCF	Programa Nacional de Crédito Fundiário
PSEE	*Process-centered Software Engineering Environment* – Ambiente de Engenharia de Software Centrado no Processo
RUP	*Rational Unified Process* – Processo Unificado Rational
SADT	*Structured Analysis and Design Technique* – Técnica estruturada de design e análise
SIG/CF	Sistema de Informações Gerenciais do Programa Nacional de Crédito Fundiário
TAD	Tipo Abstrato de Dado
UML	*Unified Modelling Language* – Linguagem de Modelagem Unificada

UTE	Unidade Técnica Estadual
WfMC	Workflow Management Coalition

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Desde o início dos anos 90, uma tendência no desenvolvimento de software tem despertado a atenção dos pesquisadores: a distribuição do desenvolvimento. Esse fenômeno é um reflexo de mudanças sociais e econômicas, que têm levado organizações a distribuírem geograficamente seus recursos e investimentos, visando aumento de produtividade, melhorias na qualidade e redução de custos no desenvolvimento de software. Em virtude dessa distribuição, equipes geograficamente dispersas cooperam para a obtenção de um produto final de software.

A distribuição física das equipes agrava problemas já inerentes à gerência do processo de software. O desenvolvimento de ambientes, modelos e ferramentas para gerenciar processos conduzidos nesse contexto é um desafio cada vez mais importante nos estudos em Tecnologia de Processos de Software. Os ambientes de suporte a processos devem prover infra-estrutura para processos distribuídos.

Este trabalho propõe um modelo de gerência de processos distribuídos, denominado APSEE-*Global*, que estende o APSEE, um ambiente de engenharia de software centrado no processo desenvolvido no contexto do grupo de pesquisa Prosoft, provendo um conjunto de funcionalidades para suporte a processos distribuídos. O APSEE-*Global* viabiliza o aumento da autonomia das equipes que participam do projeto, pela possibilidade de adotarem modelos de processo distintos e pela gerência descentralizada do processo de desenvolvimento; permite a documentação e facilita a análise e a gerência das relações entre as equipes; e provê um canal de comunicação formal para acompanhamento da execução do processo distribuído.

Os diferentes componentes do APSEE-*Global* foram especificados formalmente, o que constitui uma base semântica de alto nível de abstração que deu origem à implementação de um protótipo integrado ao ambiente de desenvolvimento de software Prosoft. A especificação do modelo foi realizada pela combinação dos formalismos Prosoft-Algébrico e Gramática de Grafos.

Palavras-chave: Processos de software, Gerência de processos distribuídos, Desenvolvimento distribuído de software, Ambientes de engenharia de software centrados no processo.

APSEE-*Global* : A Model of Distributed Software Process Management

ABSTRACT

Since the beginning of 90's, a trend on software development has awakened the attention of the researchers: the distribution of the software development. This phenomenon is an economics and socials changes reflex, which has taken organizations to distribute geographically their investments and resources, seeking an increase in the productivity, a quality improvement and cost reduction on software development. By virtue of this distribution, geographically dispersed teams cooperate to obtaining a final software product.

The team physical distribution aggravates problems already inherent to software process management. The development of environments, models and tools to manage processes led in this context is a challenge that has increased its importance on Software Processes Technology researches. The process support environments must provide the infrastructure to distributed processes.

This research proposes a model of distributed software process management, named APSEE-Global, that extends the APSEE, a process-centered software engineering environment developed in the context of the Prosoft research group, providing a set of functionalities to support distributed processes. The APSEE-Global makes possible the increase of project's team autonomy, by the possibility of adopting distinct process models and by the decentralized process management; allows documentation and facilitates the analysis and management of the relationship among teams; and provide a formal communication channel to monitoring the execution of distributed process.

The different components of APSEE-Global were formally specified, which constitutes a semantic base with high level of abstraction that originates the implementation of a prototype integrated to the Prosoft software development environment. The model specification was made by the combination of Algebraic-Prosoft and Graph Grammar formalisms.

Keywords: Software process, Distributed process management, Distributed software development, Process-centered Software Engineering Environments (PSEEs).

1 INTRODUÇÃO

Segundo o *Project Management Institute* (PMI, 2004), a gerência de projetos consiste na aplicação de conhecimentos, habilidades, ferramentas e técnicas de maneira a atingir os objetivos estabelecidos de escopo, tempo, custos e qualidade do produto a ser desenvolvido. Em projetos de desenvolvimento de software, a gerência desses objetivos é uma atividade complexa, pois, na maioria dos casos, os requisitos fornecidos no início do projeto são informais, confusos, incompletos e até contraditórios, nem sempre refletindo as necessidades reais dos usuários (CUGOLA; GHEZZI, 1998). Isso acarreta mudanças durante o projeto, afetando prazos e custos. Outro fator que agrava essa complexidade é a participação de profissionais que possuem diferentes papéis, conhecimentos, objetivos e visões do projeto, de modo que interpretações pessoais a respeito das atividades a serem realizadas podem desviar o foco do trabalho ou introduzir erros durante o desenvolvimento, comprometendo a qualidade do resultado (BANDINELLI et al, 1994). Quanto maior o tamanho e a complexidade do software a ser produzido, maior a quantidade de requisitos e pessoas envolvidas, o que dificulta ainda mais o gerenciamento (FALBO, 1996).

Uma das premissas que guiam os estudos relacionados ao desenvolvimento de software é a afirmação de que a qualidade do software produzido é fortemente dependente da qualidade do processo usado no seu desenvolvimento (PRIKLADNICKI; AUDY; EVARISTO, 2003). Fuggetta (2000) define o processo de desenvolvimento de software (ou, simplesmente, processo de software) como o conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver e manter um produto de software. Um processo eficaz deve, claramente, considerar as relações entre as atividades, os artefatos produzidos no desenvolvimento, as ferramentas e os procedimentos necessários, a habilidade, o treinamento e a motivação do pessoal envolvido. Segundo Falbo (1996), um passo importante no tratamento dos problemas de gerenciamento de projetos é entender o processo de software adotado, de modo a controlá-lo, medi-lo e gerenciá-lo. Nesse contexto, modelos de certificação do nível de maturidade do processo de desenvolvimento de software, como o *Capability Maturity Model Integration* – CMMI^{SM} (SEI, 2002) e metodologias que guiem os processos de desenvolvimento, como o MSF (*Microsoft Solutions Framework*) *Process Model* (MICROSOFT, 2004) e o RUP (*Rational Unified Process*) (IBM, 2004) têm se tornado cada vez mais importantes.

Os estudos em Engenharia de Software acerca de mecanismos para gerenciar processos de software constituem uma sub-área denominada Tecnologia de Processos de Software. A Tecnologia de Processos de Software envolve um conjunto de linguagens, métodos, arquiteturas e ferramentas utilizadas para apoiar

a gerência de processos de software (TOTLAND; CONRADI, 1995). Os principais objetivos da Tecnologia de Processos de Software são (REIS, 2002): construir ferramentas para modelar e executar os processos, guiando a realização de atividades e registrando desvios no processo; monitorar eventos ocorridos durante o desenvolvimento, com o intuito de facilitar a melhoria da maturidade dos processos da organização desenvolvedora de software; registrar o conhecimento adquirido durante a execução de processos, visando a reutilização futura; controlar a alocação e o consumo de recursos durante o processo; coletar e disponibilizar métricas do processo.

1.1 Motivação

Desde o início dos anos 90, uma tendência no desenvolvimento de software tem despertado a atenção dos pesquisadores: a distribuição do desenvolvimento (CARMEL, 2003). Esse fenômeno é um reflexo de mudanças sociais e econômicas, que têm levado organizações a distribuírem geograficamente seus recursos e investimentos, visando aumento de produtividade, melhorias na qualidade e redução de custos no desenvolvimento de software (PRIKLADNICKI; AUDY, 2002). Em virtude dessa distribuição, equipes geograficamente dispersas cooperam para a obtenção de um produto final de software.

A distribuição física das equipes agrava problemas já inerentes à gerência do processo de desenvolvimento. Diferenças culturais, de linguagem, de fuso-horário, entre outros aspectos, aumentam a complexidade na comunicação, coordenação e controle durante o processo de desenvolvimento (LANUBILE; DAMIAN; OPPENHEIMER, 2003). Embora vários dos problemas encontrados no desenvolvimento distribuído de software sejam causados por razões sociais e culturais, o uso de recursos tecnológicos pode contribuir para a solução dos problemas (MAIDANTCHICK; DA ROCHA , 2002).

Na maioria dos casos, a distribuição física do desenvolvimento implica em distribuição na gerência do processo adotado. Equipes possuindo práticas de desenvolvimento distintas e usando diferentes ferramentas cooperam para o desenvolvimento de um produto de software comum. Os processos de software que guiam esse tipo de desenvolvimento são geralmente bastante extensos, consistindo de vários sub-processos, que podem ser modelados e gerenciados de forma independente (MASLOVA, 2002).

O desenvolvimento de ambientes, modelos e ferramentas para gerenciar processos conduzidos nesse contexto é um desafio cada vez mais importante nos estudos em Tecnologia de Processos de Software (ZANONI, 2002). Os ambientes de gerência de processos de software precisam evoluir para atender aos novos requisitos de processos. Em ambientes distribuídos, cada equipe conduz internamente o desenvolvimento das atividades sob sua responsabilidade, e o processo final deve ser coerente e gerenciável (BECKER et al, 2001).

Este trabalho visa contribuir com o crescimento da Tecnologia de Processos de Software, propondo soluções para auxiliar a gerência distribuída de processos de software.

1.2 Objetivo

O objetivo geral deste trabalho é propor um modelo de gerência de processos distribuídos de software. Para atender a esse objetivo, os seguintes passos devem ser seguidos:

- Identificar as principais características da gerência de processos distribuídos de desenvolvimento de software;
- Definir e especificar formalmente um modelo de gerência de processos distribuídos;
- Prototipar o modelo proposto, isto é, desenvolver uma implementação limitada que forneça as funcionalidades básicas descritas pela especificação formal;
- Apresentar um exemplo de processo distribuído, a fim de demonstrar a utilização do modelo proposto.

1.3 Estrutura do texto

Os capítulos restantes deste trabalho são estruturados como segue. No capítulo 2 são apresentados conceitos e abordagens relacionadas à gerência de processos distribuídos. No capítulo 3 é apresentada uma descrição informal do modelo de gerência de processos distribuídos proposto. No capítulo 4 o modelo é formalizado. No capítulo 5 é apresentado o protótipo construído para validar a proposta. No capítulo 6 é apresentado um cenário que exemplifica o uso do modelo. Por fim, o capítulo 7 apresenta algumas considerações finais em relação ao trabalho desenvolvido.

Além dos capítulos citados, um anexo foi incluído neste trabalho, onde são apresentados os tipos de dados do ambiente APSEE de gerência de processos, o qual o modelo proposto estende.

2 GERÊNCIA DE PROCESSOS DISTRIBUÍDOS

> Desenvolvimento de software envolve trabalho de equipe e muita comunicação. Parece racional pôr todos os envolvidos em um mesmo lugar, encorajá-los para compartilharem objetivos, e deixar o projeto se desenrolar. Por que usar locais distribuídos quando é mais fácil trabalhar em um local sem a sobrecarga de comunicação remota e planejamento? Como é possível a sobrevivência (e o sucesso) de projetos globalmente dispersos? (EBERT; DE NEVE, 2001)

Além dos problemas inerentes à gerência de processos, o gerenciamento de processos distribuídos apresenta alguns desafios decorrentes da distribuição do processo. A adoção de ambientes de suporte a processos distribuídos tende a facilitar bastante a solução de diversos desses desafios. Com esse intuito, uma série de ambientes têm sido propostos na literatura. Este capítulo introduz os conceitos de gerência de processos distribuídos, apresentando alguns dos ambientes encontrados na literatura.

O capítulo está organizado em sete seções: a seção 2.1 apresenta os principais conceitos relacionados à gerência de processos de software; os conceitos, vantagens e características de processos distribuídos são apresentados na seção 2.2; a seção 2.3 traz os principais desafios encontrados na gerência de processos distribuídos. A seção 2.4 apresenta as características dos ambientes de suporte a processos distribuídos e os principais requisitos desse tipo de ambiente. A seção 2.5 apresenta algumas das principais abordagens encontradas na literatura, e a seção 2.6 faz uma comparação entre as abordagens apresentadas. Por fim, a seção 2.7 traz algumas considerações sobre o capítulo.

2.1 Gerência de processos de software

Como observado por Totland e Conradi (1995), há um consenso na comunidade de Engenharia de Software de que a melhor maneira de gerenciar os processos de desenvolvimento de software é representando-os por meio de modelos de processo. Um modelo de processo de software descreve a seqüência de atividades que devem ser realizadas, as ferramentas a serem utilizadas, os artefatos (produtos e documentos que serão criados ou manipulados) e os papéis das pessoas envolvidas no desenvolvimento de software (GRUHN, 2002). Se o processo de software é definido explicitamente, o desenvolvimento pode ser realizado de forma sistemática e ordenada. Isso previne a introdução de erros e permite o controle da qualidade do produto de software. A análise do processo de software permite a detecção e correção de falhas durante o desenvolvimento, além de melhoria na

comunicação entre os profissionais envolvidos no projeto (BANDINELLI et al, 1994).

Para representar um modelo de processo é freqüentemente adotada uma linguagem de modelagem de processos, que deve oferecer recursos para descrever e manipular os passos do processo. As linguagens de modelagem de processos possuem características específicas, que as diferem das linguagens de programação de propósito geral, como a necessidade de permitir a descrição de atividades que necessitam da interação humana para serem executadas. Diversas classificações para essas linguagens são encontradas na literatura, envolvendo aspectos como o paradigma computacional utilizado para modelagem e a adequação da linguagem às diferentes etapas do processo de software (REIS, 2002). Como exposto por Reis (2002), a maioria das linguagens de modelagem de processo existentes adota o paradigma orientado a atividades, que fornece um estilo intuitivo para ordenação dos eventos, facilitando a execução e o acompanhamento dos processos.

As linguagens de modelagem de processos são construídas em função de meta-modelos de processos. Um meta-modelo fornece uma série de construtores sintáticos que definem os tipos de objetos e relacionamentos que podem existir em um modelo de processo (ENGELS et al, 2001). Por exemplo, atividades e artefatos são dois tipos de objetos presentes em diversos meta-modelos. Um relacionamento possível entre eles indica que um artefato é produzido ou consumido por uma atividade. A linguagem de modelagem de processos é capaz de representar todos os modelos de processo expressos em termos do meta-modelo no qual a linguagem se baseia.

Modelos de processo devem ser executados[1], a fim de automatizar a gerência do processo (REIS, 2002). Na execução do modelo de processo, as atividades modeladas são realizadas tanto pelos desenvolvedores quanto automaticamente. Um dos objetivos da execução é manter o estado do modelo de processo consistente com o estado real de execução das tarefas (LIMA REIS, 2003). A execução dos modelos de processo provê um meio de integrar os profissionais, o processo de desenvolvimento e as ferramentas que o suportam. Segundo Dowson, citado por Arbaoui et al. (2002), a execução de modelos de processo deve ser suportada por algum ambiente, com base nas informações contidas no modelo de processo. Ambientes de Engenharia de Software Centrados no Processo (PSEEs – *Process-centered Software Engineering Environments*[2]) são uma classe de ambientes que provêem suporte ao desenvolvimento de software através da execução de modelos de processo (GARG; JAZAYERI, 1996). PSEEs geralmente são responsáveis por registrar o progresso das atividades, habilitando novas

[1] O termo *process enactment* é empregado na literatura para distinguir a execução de processos, que não é completamente automática, da execução de programas.

[2] Na literatura, encontram-se diversas denominações para ambientes de desenvolvimento de software orientados ao processo: PCE - *Process-Centered Environments* (ARBAOUI et al, 2002); PCSDE - *Process-Centered Software Development Environments* (BARTHELMESS, 2003); PSS - *Process Sensitive Support System* (ESTUBLIER, 1998), entre outras. Neste texto adotaremos a terminologia PSEE (GARG; JAZAYERI, 1996) (FUGGETTA, 2000) (LIMA REIS, 2003), por ser mais largamente utilizada na literatura.

atividades tão logo as pré-condições sejam válidas. Além disso, devem prover suporte ao gerenciamento do processo, através da monitoração e dos ajustes necessários em razão de mudanças no projeto (BARTHELMESS, 2003).

Segundo o *Software Engineering Institute* (1994), a execução de modelos de processo permite:

- Controle da seqüência existente entre atividades que compõem o processo;
- Gerência dos produtos em desenvolvimento;
- Invocação das ferramentas necessárias ao cumprimento das tarefas;
- Execução automática de ações que não necessitem de intervenção humana;
- Suporte à comunicação entre as pessoas envolvidas no projeto;
- Coleta automática de dados e de métricas;
- Suporte ao gerenciamento de tarefas pessoais;
- Redução de erros;
- Gerência do processo com informação precisa do estado corrente.

O componente central de um PSEE é a máquina de processo (ou mecanismo de execução de processos), que executa o modelo de processo (CUGOLA; GHEZZI, 1998). A máquina de processo pode ser vista como o interpretador da linguagem de modelagem de processos, e se sujeita à mesma classificação adotada para as linguagens de modelagem de processos. Assim, pode-se afirmar que um PSEE é caracterizado pela linguagem que adota para modelagem de processos (CUGOLA; GHEZZI, 1998).

Diversos PSEEs podem ser encontrados na literatura internacional, como MARVEL (KAISER; FEILER; POPOVICH, 1988), PROSYT (CUGOLA; GHEZZI, 1999), EPOS (JACCHERI; LARSEN; CONRADI, 1992), Merlin (JUNKERMANN et al, 1994), Oz (BEN-SHAUL; KAISER, 1996), SPADE (BANDINELLI et al, 1994). Comparações entre eles podem ser encontradas, por exemplo, em (CUGOLA; GHEZZI, 1998), (ARBAOUI et al, 2002) e (BARTHELMESS, 2003). Na literatura nacional, algumas propostas foram desenvolvidas, como por exemplo, APSEE (LIMA REIS, 2003), ExPSEE (GIMENES, 2000) e Estação TABA (TRAVASSOS, 1994).

2.2 Desenvolvimento distribuído de software

Segundo Zanoni (2002), o desenvolvimento distribuído de software (também chamado desenvolvimento globalizado[3]) caracteriza-se pelo envolvimento de

[3] Na maioria dos casos, as terminologias *desenvolvimento distribuído* e *desenvolvimento globalizado* são utilizadas como sinônimos na literatura. Alguns trabalhos, no entanto, consideram o termo *desenvolvimento globalizado* como um caso especial de desenvolvimento distribuído, onde a distância física entre as equipes envolve necessariamente países distintos.

várias equipes no projeto, sendo que pelo menos uma delas está distante fisicamente das demais. A Figura 2.1 ilustra as principais configurações estruturais encontradas no desenvolvimento de software. O eixo horizontal representa a distância física, e o eixo vertical representa a relação entre as organizações envolvidas no desenvolvimento de software. Os quadrantes à esquerda representam casos em que o desenvolvimento é centralizado fisicamente: no quadrante superior, todo o desenvolvimento é realizado internamente por uma única organização (*in-house*); o quadrante inferior representa uma situação de *outsourcing*, ou seja, uma organização contrata outra organização externa para desenvolver um projeto (ou parte dele). No caso descrito, trata-se de *nearshore outsourcing*, onde os desenvolvedores contratados ficam alocados no mesmo espaço físico (na sede da organização contratante, por exemplo), formando uma única equipe.

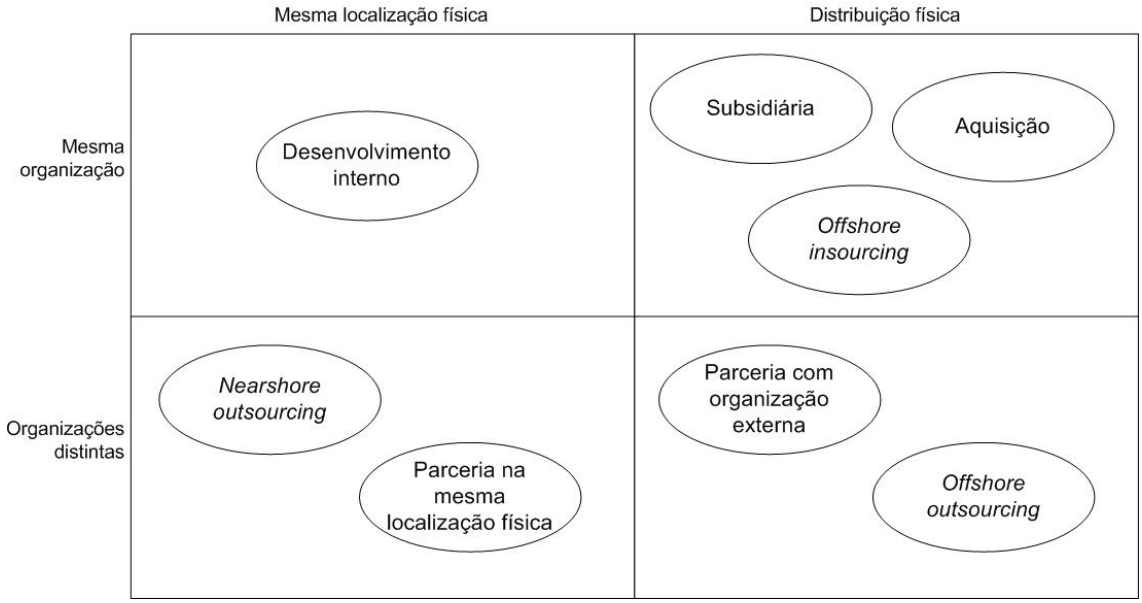

Figura 2.1: Configurações estruturais no desenvolvimento de software (adaptado de (CARMEL; AGARWAL, 2001))

O desenvolvimento fisicamente distribuído é representado nos quadrantes direitos, que ilustram o desenvolvimento composto por equipes independentes. No quadrante superior, todas as equipes pertencem à mesma organização. Três exemplos são apresentados: a existência de centros de desenvolvimento distribuídos (*offshore insourcing*); o envolvimento de subsidiárias; e a participação de outras organizações adquiridas pela organização principal. No quadrante inferior está representado o desenvolvimento distribuído envolvendo equipes pertencentes a organizações independentes. Esse cenário é encontrado, por exemplo, quando é estabelecida parceria com uma organização externa ou é firmado um contrato para desenvolvimento de parte do software com outra organização com localização física diferente da organização contratante (*offshore outsourcing*).

Diversas são as razões que motivam a adoção do desenvolvimento distribuído de software. Dentre as razões citadas na literatura, encontram-se as listadas a

seguir (DAMIAN, 2004) (HERBSLEB; GRINTER, 1999) (PRIKLADNICKI; AUDY, 2002) (CARMEL, 2003):

- Necessidade de profissionais qualificados em áreas especializadas, independente da localização física dos mesmos;
- Incentivos fiscais para o investimento em pesquisas em informática oferecidos por países como o Brasil, que atraem a instalação de centros de desenvolvimento de organizações multinacionais;
- Disponibilidade de mão-de-obra especializada e de custos reduzidos em países em desenvolvimento, como Índia, Brasil, Coréia, México e Filipinas;
- Realização de determinadas etapas do desenvolvimento de software perto dos clientes, obtendo mais informações sobre as condições locais;
- Redução dos prazos de entrega proporcionada pelo desenvolvimento *round-the-clock*, onde um fluxo quase ininterrupto de trabalho pode ser alcançado pela participação de equipes em diferentes fuso-horários;
- Formação de organizações e equipes virtuais para aproveitar oportunidades de mercado;
- Necessidade de integrar recursos resultantes de aquisições e fusões organizacionais.

Apesar de todas as motivações citadas para a adoção do desenvolvimento distribuído, a distância física também acarreta uma série de problemas, principalmente relacionados à redução da comunicação. A experiência prática mostra que a separação espacial e temporal das pessoas afeta profundamente o desenvolvimento. As características do desenvolvimento distribuído de software podem afetar negativamente o andamento das atividades. A seguir são apresentadas algumas das características que podem estar presentes nesse tipo de desenvolvimento, observando-se os principais problemas causados por elas (HERBSLEB; GRINTER, 1999) (SIQUEIRA; SILVA, 2004):

- Separação temporal: um problema que pode existir em projetos distribuídos de desenvolvimento é a diferença de fuso horário (ESPINOSA; CARMEL, 2004), que restringe os períodos em que as equipes trabalham simultaneamente, dificultando a comunicação síncrona e causando atrasos na resposta à comunicação assíncrona. A diferença no fuso-horário pode inclusive levar a confusões no que diz respeito a prazos. Além da questão do fuso horário, a separação temporal também pode existir por uma diferença de horários de trabalho entre pessoas de um mesmo local, por exemplo, quando existem diferentes turnos de trabalho;
- Cultura regional: o aspecto cultural pode causar problemas por diferenças de comportamento entre equipes de diferentes culturas. Diferenças podem ser detectadas, por exemplo, no planejamento do trabalho, no processo decisório, no estilo de argumentação, no fluxo da conversa (graus de

formalidade e objetividade distintos) e no senso de responsabilidade (OLSON; OLSON, 2003);

- Idioma: no desenvolvimento de software, além da comunicação em reuniões e diálogos em geral, ainda existem os diversos documentos do projeto. Isso faz com que a grande maioria dos projetos adote um idioma padrão que deve ser utilizado por todas as partes envolvidas. A falta de proficiência de alguns membros e diferentes interpretações semânticas pode ocasionar problemas (MOCKUS; HERBSLEB, 2001);
- Diferenças na localização física: os problemas que melhor evidenciam a diferença entre os locais talvez sejam os aspectos jurídicos. Grupos localizados em diferentes países estão sujeitos a diferentes leis comerciais, trabalhistas, civis, etc. Essas diferentes legislações podem afetar de diversas formas o desenvolvimento de software, como, por exemplo, ao dificultar a importação de um determinado hardware (HAYWOOD, 2000), ao impor diferenças de ano fiscal ou ao obrigar a retirada de vistos para viagens de negócios (KOBITZSCH; ROMBACH; FELDMANN, 2001);
- Cultura organizacional: diferenças na cultura organizacional podem causar problemas de entendimento do significado de atividades do processo, visão diferente sobre qualidade (KOBITZSCH; ROMBACH; FELDMANN, 2001) e estrutura hierárquica;
- Infra-estrutura das organizações: a existência de infra-estrutura adequada em todas as organizações envolvidas no desenvolvimento pode ser difícil de ser obtida. Além de adequada, a infra-estrutura deve ser compatível entre as organizações, no que diz respeito a sistemas operacionais, ambientes de desenvolvimento, etc;
- Relações comerciais: dependendo da natureza das relações comerciais entre as organizações (*outsourcing*, subsidiária, aquisição), o compartilhamento de informações e a estrutura da equipe, por exemplo, podem ser prejudicadas;
- Problemas estratégicos: gerados pela dificuldade em dividir entre as equipes o trabalho a ser realizado. Vários aspectos devem ser considerados, como o grau de conhecimento sobre as tecnologias a serem utilizadas em cada atividade, a infra-estrutura e os recursos disponíveis. O ideal é que a divisão do trabalho permita a maior autonomia possível a cada equipe, para que ela não dependa das demais para a realização do trabalho sob sua responsabilidade;
- Gerência do conhecimento: se as informações sobre o processo não estiverem disponíveis e atualizadas, gerentes de projeto podem ter dificuldades para acompanhar o andamento do processo. A ausência de documentação compromete a integração entre as equipes, levando ao retrabalho e à perda de oportunidades de reuso no projeto;

- Gerência do projeto e do processo de desenvolvimento: em projetos distribuídos, cada equipe gerencia seu processo de software. Se os processos não estiverem sincronizados, as equipes podem definir a mesma atividade de maneiras diferentes. Marcos para sincronização devem ser definidos, com pré e pós-condições bem claras.

2.3 Desafios na gerência de processos distribuídos

Além dos problemas inerentes ao processo de desenvolvimento, a gerência de processos distribuídos enfrenta os problemas causados pela distribuição do desenvolvimento. A partir dos problemas decorrentes do desenvolvimento distribuído de software apontados na seção anterior, pode-se traçar os principais desafios a serem considerados na gerência de processos distribuídos, citados a seguir:

- Definição clara da estrutura de dependências entre as equipes;
- Definição de critérios para a atribuição de atividades às equipes;
- Definição de estratégias para redução da distância cultural;
- Estabelecimento de interfaces bem definidas para comunicação formal;
- Criação de canais para comunicação informal;
- Garantia de consistência e disponibilidade dos artefatos e documentos do projeto;
- Acompanhamento do progresso das equipes.

Vários trabalhos têm sido desenvolvidos com o intuito de propor soluções para esses problemas. A estrutura de dependências entre equipes pode ser explicitada pela definição de relações entre os sub-processos. Mockus e Weiss (2001) sugerem que a divisão de atividades entre equipes seja realizada com base nas funcionalidades do sistema, nos estágios do desenvolvimento (análise, testes, etc) ou de acordo com a localização física das equipes, deixando para aquelas próximas ao cliente as atividades que requerem maior interação com o usuário, como levantamento de requisitos e implantação. Divisão de atividades de acordo com o componente de software a ser desenvolvido e distribuição apenas de atividades de customização do produto de software também são sugeridas na literatura (GRINTER; HERBSLEB; PERRY, 1999). Karolak (1998) também sugere uma série de critérios para a divisão de atividades, dentre eles: relações comerciais, arquitetura do sistema, conhecimento das equipes e disponibilidade de ferramentas e recursos.

Diversas estratégias têm sido propostas para diminuir a distância cultural: escolha estratégica de projetos, de modo a definir equipes com semelhanças culturais ou com experiência em trabalho conjunto; definição de equipes com 25% do trabalho realizado perto do cliente; utilização do papel do *liason*, um gerente de projeto que viaja freqüentemente entre as equipes; investimento no fortalecimento

de uma língua comum entre as equipes (CARMEL; AGARWAL, 2001) são algumas das estratégias apontadas pela literatura.

A definição de interfaces de comunicação formal pode ser obtida por meio de modelos de processo bem definidos, com marcos (*milestones*) e métricas bem estabelecidas. Os canais de comunicação informal, por sua vez, podem abranger videoconferência, espaços de trabalho compartilhado, programas de troca de mensagens (*instant messengers*), etc. A consistência e a disponibilidade de artefatos e documentos pode ser obtida pela manutenção de uma página do projeto na web que contenha descrição, métricas, planejamento e informações sobre as equipes do projeto e pela utilização de software de gerenciamento de configuração.

O acompanhamento do andamento das equipes pode ser realizado pela monitoração da execução dos processos de software, pela realização de reuniões (ou videoconferências) em intervalos regulares e pela definição da freqüência para sincronização do andamento dos processos locais.

Analisando os desafios mencionados, pode-se observar que a distribuição do processo entre equipes aumenta os problemas de coordenação e controle na gerência do processo, tanto diretamente quanto pelas dificuldades de comunicação geradas (LANUBILE; DAMIAN; OPPENHEIMER, 2003). Segundo a Teoria da Coordenação de Malone e Crowston (1990), a coordenação é o ato de gerenciar dependências entre atividades que visam a atingir um objetivo comum. No caso do desenvolvimento distribuído, pode-se definir coordenação como a integração entre atividades e equipes, de modo que as equipes contribuam para atingir o objetivo final (CARMEL; AGARWAL, 2001). O controle entre equipes pode ser definido como a aderência a objetivos, políticas, padrões e níveis de qualidade. Alguns dos problemas de coordenação e controle presentes em processos distribuídos são listados a seguir (MAIDANTCHICK; DA ROCHA, 2002):

- Coordenação das equipes: refere-se à supervisão e aos relacionamentos entre as equipes. Essa coordenação envolve a definição das habilidades de cada equipe e a delegação de atividades, de acordo com as características e restrições das equipes;
- Coordenação de atividades: refere-se ao controle do processo de desenvolvimento, envolvendo identificação de atividades concorrentes, atividades cooperativas e a sincronização na execução dos processos;
- Controle de artefatos: diz respeito à integração de componentes, disponibilidade de resultados, notificação de alterações e gerência de configuração;
- Suporte à comunicação: envolve troca de informações durante o processo de desenvolvimento, para garantir uma gerência efetiva do processo.

Os problemas de coordenação em processos distribuídos de desenvolvimento de software podem ser minimizados pela adoção de processos de software, definidos de acordo com as particularidades dos projetos distribuídos (MAIDANTCHICK; DA ROCHA, 2002).

2.4 Ambientes de suporte a processos distribuídos

Quando o desenvolvimento de software envolve organizações distintas, é inviável utilizar um único modelo centralizado para refletir todo o escopo do processo de software. O modelo de processo deve refletir a estrutura do projeto, sendo dividido em sub-processos autônomos (WANG, 2000-a). Embora autônomos, os sub-processos devem interagir, possuindo fluxos de controle e dados entre si. Atividades de um processo distribuído podem envolver diversas equipes durante a sua execução, e podem depender do resultado da execução de atividades executadas por equipes distintas. Para diferenciar os diferentes conceitos de processo presentes no desenvolvimento distribuído, a seguinte terminologia é definida neste trabalho:

- **Processo local**: é o processo de desenvolvimento que corresponde a um sub-processo autônomo;
- **Processo distribuído**: é definido por um conjunto de processos locais e pelas relações existentes entre os processos locais;

Segundo (GRUHN, 2002), ambientes de suporte a processos não devem assumir que processos de software são centralizados. Ao contrário, tais ambientes devem prover infra-estrutura para processos distribuídos. As características de ambientes de suporte a processos distribuídos de software diferem das apresentadas por ambientes de desenvolvimento que provêem suporte ao desenvolvimento centralizado. PSEEs que provêem suporte a processos distribuídos devem coordenar usuários de diversas equipes, notificando uma equipe sobre as atividades realizadas pelas outras e permitindo a troca de artefatos entre equipes (BEN-SHAUL; KAISER, 1998). Cada equipe deve ser capaz de especificar seu próprio modelo de processo local, e obter colaboração com outras equipes. Para prover essas funcionalidades, PSEEs que suportem processos distribuídos devem possuir um certo grau de acoplamento, para facilitar a coordenação entre as equipes, mas ao mesmo tempo devem possuir um grau de independência, para que as equipes possam responsabilizar-se pelos próprios processos.

2.4.1 Classificação dos ambientes

Várias abordagens podem ser encontradas na literatura para suportar processos de software distribuídos. Ben-Shaul e Kaiser (1998) propõem a seguinte classificação para as abordagens, de acordo com o grau de acoplamento e de heterogeneidade entre os PSEEs:

- Cada equipe escolhe o seu próprio PSEE, sem preocupação com a compatibilidade entre os PSEEs escolhidos;
- Cada equipe escolhe o seu próprio PSEE, mas é definido um formato padrão para troca de dados compartilhados;
- As equipes utilizam várias instâncias independentes do mesmo PSEE;

- As equipes utilizam um PSEE *multi-site*, ou seja, um único PSEE que distingue as equipes e provê facilidades para compartilhamento e colaboração dentro do ambiente;
- Utilização de *multi*-PSEEs, que são PSEEs distintos comunicando-se por meio de um modelo de notificação de eventos e com um formato padrão para troca de dados;
- Um único PSEE é utilizado, e embora seja executado de forma distribuída, não há distinção entre equipes – o processo é definido como sendo composto por uma única equipe.

Nos três primeiros casos da classificação proposta por Ben-Shaul e Kaiser, os PSEEs são completamente independentes, e a coordenação entre as equipes é realizada externamente ao ambiente, por meio de formatos comuns para troca de dados, como os propostos pela WfMC (*Workflow Management Coalition*) para troca de dados em modelos de *workflow* (WFMC, 1999). Esses formatos provêem suporte à representação de um modelo de processo em ambientes distintos, mas não provêem meios para colaboração durante a execução dos processos. Por outro lado, o último caso apresentado exige que um único modelo de processo seja adotado, o que limita a autonomia das equipes.

As abordagens de PSEEs *multi-site* e *multi*-PSEEs, também denominadas de federações[4] de PSEES, possuem as características de ambientes de coordenação de projetos distribuídos apresentadas por (BEN-SHAUL; KAISER, 1998). Um PSEE *multi*-site (ou federação de PSEEs homogêneos) oferece instâncias do mesmo PSEE para todas as equipes, mas suporta a execução de modelos de processos distintos. Já um *multi*-PSEE (ou federação heterogênea) provê uma estrutura que integra PSEEs distintos, que podem adotar diferentes formalismos de modelagem e modelos de processos, desde que os PSEEs possuam interface compatível à federação.

Ambas as abordagens de federações de PSEEs permitem que cada equipe especifique o seu próprio modelo de processo e as interações existentes com outras equipes. Uma federação é composta de instâncias locais, que executam o modelo de processo de cada equipe, e de um componente responsável pela comunicação entre os processos locais, denominado fundação. A comunicação entre os processos locais é guiada pela definição das relações existentes entre os processos de cada equipe.

2.4.2 Requisitos de ambientes de suporte a processos distribuídos

Os requisitos a serem atendidos por ambientes que suportem a gerência de processos distribuídos são encontrados em diversos trabalhos da literatura (BEN-SHAUL; KAISER, 1998) (MAIR; HAAG, 2001).

---

[4] Na área de banco de dados, o termo “gerenciadores de banco de dados federados” descreve uma coleção de sistemas de bancos de dados autônomos, que cooperam entre si

- **Requisito 1**: infra-estrutura de comunicação

  Uma federação deve prover infra-estrutura para que as instâncias locais troquem informações a respeito de atividades que envolvam mais de uma equipe;

- **Requisito 2**: definição e aceitação de um processo global

  Os ambientes devem prover suporte à modelagem e a execução de processos globais, que definem as dependências entre processos locais. Devem ser suportadas estratégias *top-down* e *bottom-up* para modelagem do processo global. A estratégia *top-down* refere-se ao detalhamento do processo global em vários níveis de granularidade, e é útil na especificação de processos que necessitam de um controle central, por exemplo, em relação a padrões de qualidade e de projeto. A estratégia *bottom-up* refere-se à definição da interoperabilidade entre processos locais sem definição uma visão global, e é usada quando processos já existentes precisam ser integrados;

- **Requisito 3:** integração e interoperabilidade entre modelos de processo existentes

  Ferramentas de suporte à gerência de processos distribuídos devem permitir o uso das diferentes abordagens adotadas pelas equipes para modelagem e execução do processo local. Embora federações homogêneas sejam formadas por várias instâncias de um único PSEE, deve ser suportada a definição da interoperabilidade entre os diferentes modelos de processo adotados pelas equipes;

- **Requisito 4**: autonomia local e operação independente

  Cada instância local do PSEE deve funcionar de forma autônoma e independente das demais durante a definição e a execução de atividades que envolvam apenas recursos e dados locais. Comunicação com outras instâncias não deve ser necessária durante a modelagem e a execução de atividades puramente locais. A autonomia dos processos locais tem sua importância ressaltada por Schwenkreis, como citam (BEN-SHAUL; KAISER, 1998): “Uma equipe com um nível mais baixo na hierarquia não deseja que a equipe que a gerencia tome conhecimento de como uma atividade é realizada”;

- **Requisito 5**: restrição das dependências globais aos processos afetados

  Esse requisito relaciona-se ao anterior, e indica que quando uma atividade envolvendo um conjunto de processos locais estiver sendo realizada, apenas os processos envolvidos na execução devem depender entre si. Os demais processos locais não devem ser envolvidos na execução da atividade;

- **Requisito 6**: percepção (*awareness*) da configuração

  Cada instância do PSEE deve ter conhecimento de quais são as demais instâncias com os quais ela interage;

- **Requisito 7**: configuração dinâmica

Os requisitos de um processo de desenvolvimento de software vão sendo conhecidos durante o decorrer do projeto. É possível que no início do desenvolvimento não se saiba ao certo quais equipes estarão envolvidas. Mesmo quando essa informação é conhecida, equipes podem participar do desenvolvimento apenas durante um período. Por exemplo, uma equipe pode participar do projeto somente durante a implantação do produto de software. Em decorrência disso, um requisito necessário é que instâncias locais do PSEE possam entrar ou sair da federação a qualquer momento, mesmo durante a execução do projeto distribuído;

- **Requisito 8**: estado do processo persistente

  A execução de processos de software costuma ter uma duração longa. Durante esse período, equipes podem sair do projeto, ou falhas podem ocorrer. Em virtude disso, o estado e os artefatos do projeto devem ser armazenados de forma persistente;

- **Requisito 9**: semântica transacional

  Uma atividade de um processo de software pode depender de outras atividades durante o projeto. A falha de uma atividade pode, portanto, ter efeito sobre a execução de outras. Além disso, no caso de projetos distribuídos, processos distintos podem executar simultaneamente partes de uma mesma atividade. Se uma das partes falhar, a atividade toda deve ser interrompida, e seus efeitos devem ser eliminados. Mecanismos para recuperação em caso de falhas devem ser fornecidos, para que os efeitos da falha na execução de uma atividade sejam propagados.

2.5 Abordagens encontradas na literatura

Processos distribuídos de software têm sido largamente estudados na literatura. Diversos ambientes de suporte a processos distribuídos têm sido propostos. Durante a realização deste trabalho foi realizado um estudo acerca dos principais PSEEs que provêem suporte à modelagem e a execução de processos distribuídos. Alguns dos PSEEs estudados são descritos nesta seção, de maneira resumida, abordando somente as principais características de cada PSEE, pois o estudo completo de cada PSEE está fora do escopo do presente trabalho[5].

As subseções a seguir apresentam os PSEEs homogêneos Oz, Serendipity-II e AHEAD e os PSEEs heterogêneos CAGIS, APEL e PIE.

2.5.1 Oz

Desenvolvido na *Columbia University* como um sucessor do Marvel (KAISER; FEILER, 1987), Oz (BEN-SHAUL; KAISER, 1995) (BEN-SHAUL; KAISER,

[5] Em (FREITAS, 2004) foi realizado um estudo mais detalhado de alguns ambientes apresentados neste trabalho.

1996) foi o primeiro PSEE distribuído. Oz é uma federação homogênea de PSEES, proposta para estender o Marvel para projetos distribuídos.

Cada instância local do Oz é denominada *SubEnv*. Cada *SubEnv* executa um processo local, especificado por meio de regras, que definem uma lista de tipos de objetos usados como parâmetros, uma pré-condição para que a regra seja inicializada, uma ação e uma ou mais pós-condições.

Oz utiliza a metáfora de "aliança internacional" (*international alliance*) para definir processos distribuídos. Segundo essa metáfora, países independentes (processos) operam de forma autônoma. Colaboração entre países (interoperabilidade entre processos) ocorre apenas com base em acordos pré-definidos. A colaboração é modelada como um encontro de ápice, onde os preparativos e as consequências do encontro são realizados de forma independente (por meio de sub-processos privados). Apenas o encontro de ápice em si é realizado de forma cooperativa (com base em um sub-processo compartilhado).

Um acordo (*treaty*) define um sub-processo comum, um sub-esquema comum e um conjunto de regras para acesso a dados. A definição de acordos envolve os seguintes aspectos:

- Para permitir a invocação de atividades envolvendo múltiplas equipes, deve ser definido um acordo entre as equipes. O acordo define um sub-processo comum e determina quais ações serão realizadas em cada *SubEnv*. O acordo definido fará parte de cada processo local que participará do sub-processo comum. No entanto, uma atividade descentralizada é executada em apenas um único *SubEnv*, denominado coordenador. Desse modo sub-processos comuns não são necessariamente recíprocos, no sentido de que nem todos os *SubEnv* terão os mesmos “privilégios” em atividades descentralizadas;
- Para habilitar a definição da execução de atividades descentralizadas, os *SubEnv* envolvidos no sub-processo devem possuir um sub-esquema comum, para que os tipos dos parâmetros especificados na atividade sejam conhecidos por todos os *SubEnv*. Um sub-esquema comum não implica necessariamente em compartilhamento de dados; apenas os tipos de dados são compartilhados, não as instâncias.
- É preciso definir e controlar quais dados podem ser acessíveis e por quais *SubEnvs*.

Por exemplo, um *SubEnv* 1 propõe um acordo para um *SubEnv* 2, definindo determinadas atividades. Se o *SubEnv* 2 concordar com o acordo proposto, então pode-se iniciar a execução do sub-processo comum entre *SubEnv* 1 e *SubEnv* 2.

A execução de um sub-processo comum é realizada através do protocolo de ápice (*summit*). As etapas do protocolo, ilustradas na Figura 2.2, são descritas a seguir:

- *Pre-summit*: essa fase começa quando uma atividade é invocada em um processo com dados de múltiplos processos. Os *SubEnvs* envolvidos são

notificados, e os pré-requisitos existentes para a execução da atividade são verificados em cada *SubEnv*, cada um de acordo com seu processo local;

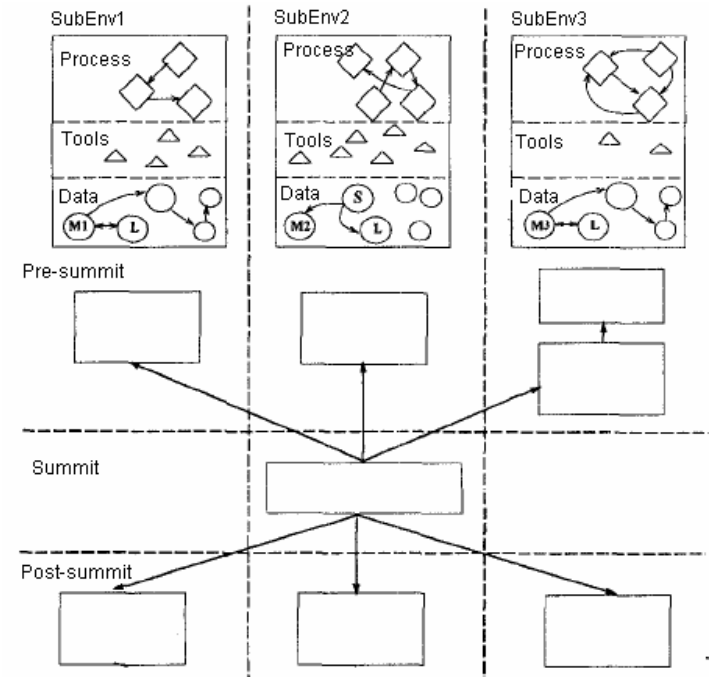

Figura 2.2: Interoperabilidade entre processos no Oz
(adaptado de (BEN-SHAUL; KAISER, 1996))

- *Summit*: se todos os *SubEnv* participantes concordam, então a atividade é realizada no processo coordenador (que foi responsável pela invocação), utilizando todos os argumentos locais e remotos necessários;
- *Post-summit*: quando o *summit* é finalizado, todos os *SubEnv* envolvidos são notificados pelo coordenador. Então, cada *SubEnv* realiza simultaneamente as ações necessárias, atualizando seu estado local com a informação da execução da atividade.

2.5.2 Serendipity-II

Serendipity-II (GRUNDY, 1999) (GRUNDY et al, 1998), desenvolvido em conjunto pela *University of Waikato* e pela *University of Auckland*, Nova Zelândia, é uma extensão do Serendipity, uma ferramenta centralizada de suporte a processos.

O ambiente Serendipity-II suporta modelagem e execução distribuída de processos. Cada usuário define o seu próprio modelo de processo, e pode compartilhar a modelagem de fragmentos do modelo com outros usuários. A edição colaborativa pode ocorrer de forma síncrona ou assíncrona. No caso de edição assíncrona, as alterações são combinadas posteriormente. A sincronização é realizada através da troca de informações acerca das alterações ocorridas. Na edição síncrona, alterações realizadas são imediatamente enviadas a todos os usuários e incorporadas aos respectivos modelos. Na edição assíncrona, cada usuário seleciona quais alterações devem ser aplicadas à sua visão do modelo.

Todas as informações de modelos de processo que devem ser compartilhadas são replicadas entre os ambientes locais dos usuários, e a consistência é obtida através da troca de eventos ou objetos de atualização. Desse modo, usuários podem trabalhar desconectados, sincronizando posteriormente os modelos.

Serendipity-II utiliza uma linguagem visual para suportar processos de software. Um processo é descrito através de estágios de processo, que definem tarefas específicas de desenvolvimento. Fluxos de execução conectam os estágios, transmitindo eventos de execução, que guiam a execução do processo. Estágios possuem papéis associados, e podem ser refinados em sub-estágios.

Cada usuário executa localmente o seu modelo de processo. A execução de um estágio de processo em um modelo gera eventos de execução, que são propagados para os demais usuários, podendo disparar a execução de estágios remotos. Como o Serendipity-II permite a edição assíncrona de modelos, é possível que diferentes máquinas de processo executem versões distintas de um modelo. Assim, um evento de execução recebido por determinada máquina pode não ser compatível com a versão do processo sendo executada. Neste caso, o evento é notificado ao usuário, de modo que este possa monitorar o estado de execução de outros usuários.

A arquitetura do Serendipity-II é ilustrada na Figura 2.3. Cada usuário possui um ambiente, que é responsável pela comunicação com outros ambientes e pela execução local do modelo de processo. uma coleção de objetos, que representam repositórios, modelos de processos e agentes de software locais[6].

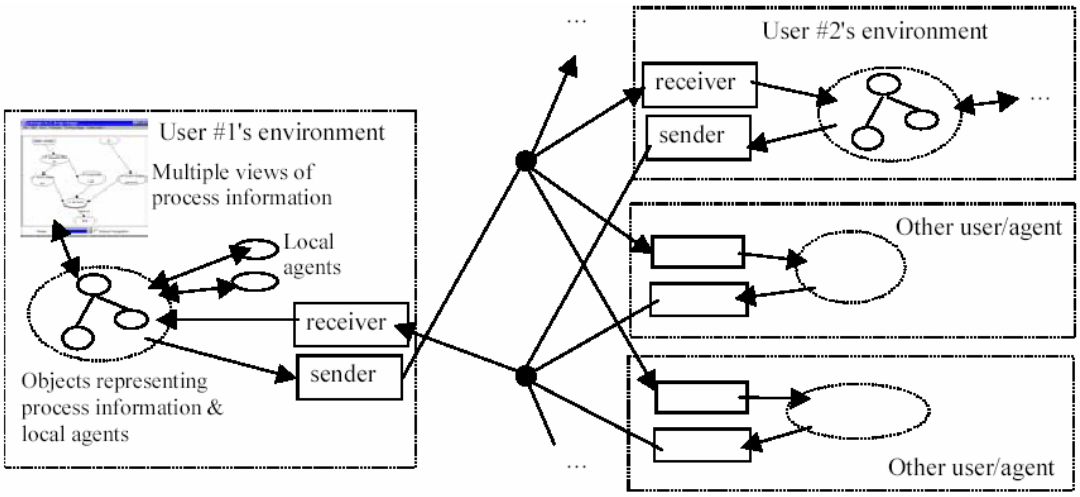

Figura 2.3: Arquitetura do Serendipity-II
(GRUNDY, 1999)

A arquitetura do Serendipity-II implementa a comunicação entre usuários e agentes de software por meio de comunicação ponto a ponto através da Internet, dispensando a existência de um servidor central. A rede pode ser modificada

[6] No Serendipity-II, agentes de software são unidades de funcionalidade definidas pelo usuário capazes de realizar tarefas de maneira autônoma, que podem ser configurados para automatizar tarefas, coordenar o trabalho de múltiplos usuários e realizar a interface com sistemas externos.

dinamicamente, porém cada ambiente precisa conhecer a localização dos demais ambientes.

2.5.3 AHEAD

O PSEE AHEAD (*Adaptable Human-centered Environment for the Administration of Development processes*) (BECKER et al, 2001), desenvolvido na *Aachen University of Technology*, Alemanha, utiliza o conceito de delegação de processos para representação de processos distribuídos. Uma abordagem *top-down* é empregada, onde uma organização atua como supervisora do processo distribuído. A organização supervisora é vista como cliente no processo, delegando a execução de partes do processo para as demais organizações, que atuam como contratadas. Uma contratada pode, por sua vez, delegar partes do seu processo, atuando nesse caso como cliente das organizações sub-contratadas.

Cada organização contratada possui autonomia para executar a parte do processo delegada, podendo refinar as atividades nela definidas, desde que não altere a estrutura original. As informações sobre refinamentos realizados podem ser ocultadas do cliente. Alterações na estrutura não podem ser realizadas pelo fato de que uma delegação é vista como um contrato: uma vez que uma parte de processo tenha sido delegada, ela só pode ser alterada em comum acordo por ambas as organizações envolvidas na delegação.

A alteração de uma parte de processo delegada somente pode ser realizada pelo cliente. Para realizar a alteração, ele deve modificar o estado das atividades delegadas para *Planning*. Se a alteração não causar inconsistências na cópia do processo utilizada pelo contratado, então ela será efetivada em ambos os lados.

A Figura 2.4 ilustra os passos para a delegação de processos. No passo 1, o gerente da organização cliente modela o processo distribuído, incluindo as atividades a serem delegadas. No passo 2, as partes a serem delegadas são selecionadas. As partes selecionadas são exportadas para um arquivo no passo 3, e continuam presentes no processo distribuído.

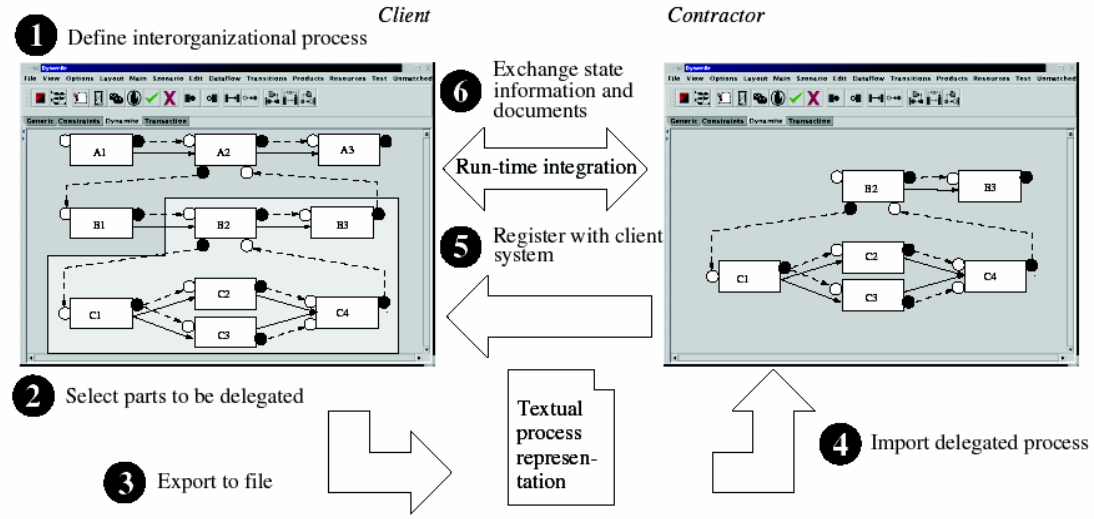

Figura 2.4: Passos da delegação de processos no AHEAD
(BECKER et al, 2001)

No passo 4, a organização contratada importa o arquivo contendo o processo delegado. A importação cria uma cópia desse processo na instância de PSEE utilizada pelo contratado. O AHEAD restringe-se a PSEEs homogêneos, ou seja, tanto o cliente quanto o contratado devem utilizar instâncias do mesmo PSEE (no caso, o AHEAD). A organização contratada registra-se junto ao cliente no passo 5. Esse registro define um canal de comunicação entre ambos, e habilita a execução do fragmento delegado. O passo 6 indica que o estado de execução das atividades delegadas é enviado ao cliente, que atualiza a sua cópia do fragmento delegado. Documentos de entrada e saída das atividades delegadas também são compartilhados.

2.5.4 CAGIS

O projeto CAGIS (*Cooperative Agents in Global Information Space*) (RAMAMPIARO; WANG; BRASETHVIK, 2000) (WANG, 2000-a) (WANG, 2000-b) (WANG, 2002), proposto na *Norwegian University of Science and Technology*, é o sucessor do EPOS (JACCHERI; LARSEN; CONRADI, 1992), e foi desenvolvido para modelar e suportar processos de software cooperativos entre equipes geograficamente dispersas. No contexto do projeto foi desenvolvido um ambiente de suporte à execução de processos, o CAGIS PCE, que consiste de três componentes principais:

- O CAGIS SimpleProcess é uma ferramenta de *workflow* que provê suporte à definição e à execução de atividades que envolvem apenas uma equipe;
- O CAGIS Distributed Intelligent Agent System (DIAS) utiliza agentes móveis para prover suporte a atividades cooperativas, envolvendo mais de uma equipe;
- O CAGIS GlueServer especifica regras de cooperação entre atividades individuais e cooperativas, possibilitando a interação entre o CAGIS SimpleProcess e o CAGIS DIAS.

A interação entre os três componentes do CAGIS PCE é ilustrada na Figura 2.5, e apresenta a seguinte seqüência:

- As instâncias do CAGIS SimpleProcess informam o estado de execução das atividades individuais ao CAGIS GlueServer;
- O CAGIS GlueServer procura regras de cooperação que definam a execução de atividades cooperativas, com base no estado de execução dos *workflows*. Se um relacionamento que mapeia o estado corrente for encontrado, o CAGIS GlueServer irá iniciar um ou mais agentes no CAGIS DIAS, para execução da atividade cooperativa correspondente;
- Quando a execução da atividade cooperativa for finalizada, o resultado será informado ao CAGIS GlueServer;
- O CAGIS GlueServer irá então ativar a reação definida no GlueModel para o resultado informado. Reações podem ser executadas nos *workflows* (por exemplo, a execução de atividades individuais), no CAGIS DIAS (por

exemplo, execução de outra atividade cooperativa) ou no próprio CAGIS GlueServer (por exemplo, modificação de uma regra de cooperação).

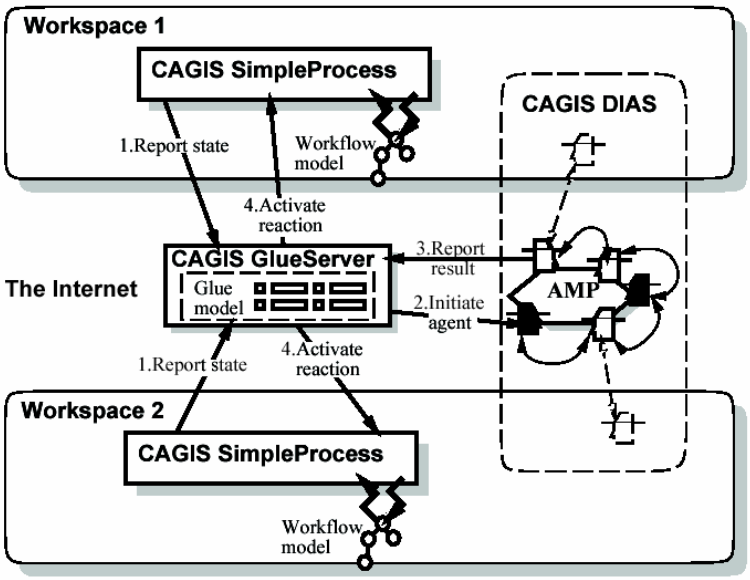

Figura 2.5: Arquitetura do CAGIS PCE
(WANG, 2002)

2.5.5 APEL

Desenvolvido no *Laboratoire Logiciels, Systèmes, Rèseaux,* na França, APEL (CUGOLA; GHEZZI, 1998) (ESTUBLIER; CUNIN; BELKHATIR, 1998) (ESTUBLIER; AMIOUR; DAMI, 1999) é um PSEE com o objetivo de suportar interoperabilidade entre PSEEs heterogêneos, permitindo a construção de uma federação de processos capaz de gerenciar processos distribuídos complexos.

Segundo (ESTUBLIER; AMIOUR; DAMI, 1999), a federação de componentes requer uma arquitetura onde a interoperabilidade seja suportada no nível conceitual (meta-nível), de modelo e de execução (máquina de processo). A arquitetura do APEL visa atingir estes três níveis de interoperabilidade, definindo um modelo de processo comum que obedece à semântica de um meta-modelo conceitual comum. O meta-modelo comum contém o conjunto de conceitos compartilhados por todos os componentes, ou seja, os conceitos básicos do processo.

O modelo de processo comum é composto pelas atividades que necessitam de interação entre os diferentes PSEEs. APEL define também um modelo de interoperabilidade, com foco nos detalhes técnicos de como aspectos mencionados no processo comum devem ser tratados. O modelo de interoperabilidade contém informações relacionadas ao controle de consistência do modelo de processo comum. Para prover interoperabilidade no nível de execução, o ambiente APEL inclui os seguintes componentes, ilustrados na Figura 2.6:

- *Servidor de Processo (Process Server):* contém o modelo comum e o estado comum, disponíveis para todos os PSEEs. A interface provida pelo servidor de processos é composta por métodos que modificam o modelo de processo.

A chamada de um método do servidor de processo gera um evento que contém a assinatura do método;

- *Servidor de eventos (Event Server):* o servidor de eventos recebe os eventos gerados no APEL e os dispara para os componentes registrados para recepção de cada evento. O servidor de eventos suporta a comunicação entre o servidor de processo, a máquina de processo comum, a máquina de processo de interoperabilidade e os PSEEs que compõem o ambiente. No APEL os componentes não se comunicam diretamente: toda a comunicação é realizada através do servidor de eventos;
- *Máquina de processo comum (CPE – Commom Process Engine):* executa o modelo de processo comum de acordo com a semântica do meta-modelo comum, e atualiza o estado comum no servidor de processo;
- *Máquina de processo de interoperabilidade (IPE – Interoperability Process Engine):* garante a consistência da federação através do controle do servidor de eventos, do servidor de processo e da máquina de processo comum.

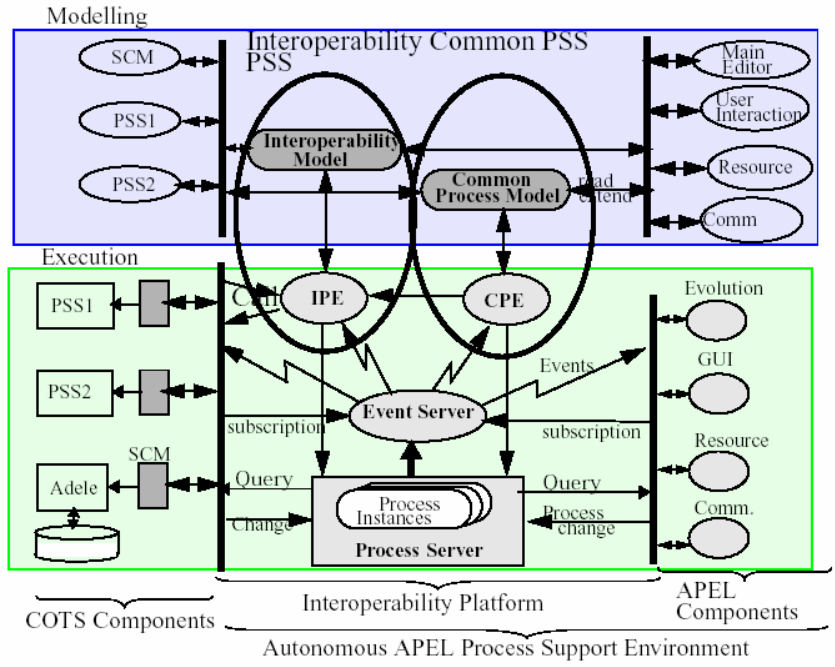

Figura 2.6: Arquitetura de uma federação APEL
(ESTUBLIER; CUNIN; BELKHATIR, 1998)

Além destes componentes, APEL possui componentes de suporte ao ambiente de processo: um editor gráfico, para modelagem de processos; gerenciador de recursos, para modelagem e alocação de recursos e aspectos organizacionais; gerenciador de comunicação, que permite modelagem e tratamento de fluxo de mensagens; um componente de interface com o usuário; e um gerenciador de evolução, que auxilia na realização de alterações no processo.

2.5.6 PIE

PIE (*Process Instance Evolution*) (CUGOLA et al, 2000) é parte do projeto europeu ESPRIT, desenvolvido com base no APEL. O ambiente PIE é uma federação de PSEEs heterogêneos, com o objetivo de suportar e gerenciar a evolução das instâncias de processo. Na definição da federação deve-se estabelecer um conjunto de conceitos comuns que abstraia e integre os conceitos dos PSEEs individuais. Estes conceitos são definidos através do Universo Comum, que contém instâncias dos conceitos comuns.

Os aspectos funcionais do comportamento do Universo Comum estão contidos no Modelo do Universo Comum. Deve-se definir também um Modelo Operacional, com o objetivo de definir como os aspectos mencionados no Modelo do Universo Comum devem ser tratados. O Modelo Operacional descreve a reação a mudanças no Universo Comum, indicando explicitamente as regras de consistência para cada reação, como ordem de invocação de componentes para determinada mudança no Universo Comum, controle transacional, etc.

O ambiente PIE é ilustrado na Figura 2.7. A fundação deve prover um repositório para o Universo Comum, um Modelo do Universo Comum (e o respectivo interpretador) e um Modelo Operacional (com o respectivo interpretador). O Modelo Operacional utiliza um *middleware* dedicado para prover facilidades básicas de comunicação e serviços específicos de controle. A federação possui dois tipos de componentes: componentes PIE, que provêem as funcionalidades do sistema, e componentes da aplicação, que atendem a requisitos da aplicação do usuário.

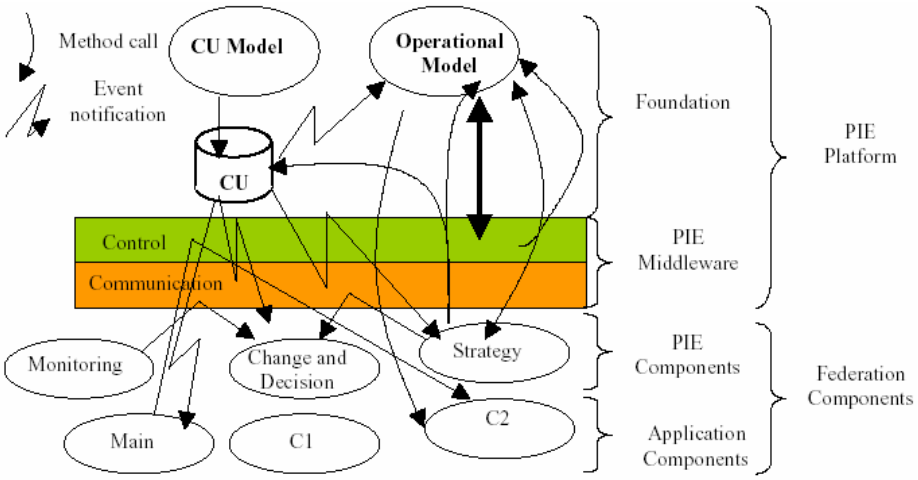

Figura 2.7: Arquitetura da plataforma PIE
(CUGOLA et al, 2000)

2.6 Comparação entre as abordagens

As diversas abordagens apresentadas podem ser comparadas em relação a diversos aspectos que envolvem o suporte a processos distribuídos. A Tabela 2.1 apresenta uma breve comparação entre os PSEEs discutidos, com base nos

requisitos para ambientes que provêem suporte a processos distribuídos apresentados na seção 2.4.

Tabela 2.1: Comparação entre as abordagens apresentadas

Requisito	Oz	Serendipity-II	AHEAD	CAGIS	APEL	PIE
R1: infra-estrutura de comunicação	Peer-to-peer	Peer-to-peer	Peer-to-peer	Centralizada	Centralizada	Centralizada
R2: definição e aceitação de um processo global	Bottom-up	Top-down	Top-down	Bottom-up	Bottom-up	Bottom-up
R3: integração e interoperabilidade entre modelos de processo existentes	Sim	Não	Não	Sim	Sim	Sim
R4: autonomia local e operação independente	Sim	Sim	Sim	Sim	Sim	Sim
R5: restrição das dependências globais aos processos afetados	Sim	Sim	Sim	Sim	Sim	Sim
R6: percepção (*awareness*) da configuração	Sim	Sim	Sim	Não	Não	Não
R7: configuração dinâmica	Sim	Sim	Sim	Sim	Sim	Sim
R8: estado do processo persistente	Sim	Sim	Sim	Sim	Sim	Sim
R9: semântica transacional	Sim	Sim	Sim	Sim	Sim	Sim

2.7 Considerações

Como discutido neste capítulo, a gerência de processos distribuídos deve tratar não apenas aspectos inerentes à gerência de processos, mas deve levar em consideração os desafios impostos pela distribuição física entre as equipes. Esses desafios tornam-se ainda mais evidentes quando diversas organizações estão envolvidas no desenvolvimento, pelas diferenças organizacionais decorrentes, ou quando a distância física envolve países com diferenças culturais significativas.

Ambientes de desenvolvimento de software devem prover suporte à modelagem e a execução de processos distribuídos. As abordagens centralizadas de PSEEs são suficientes para processos onde todas as equipes seguem um único modelo de desenvolvimento, e são guiadas pelo mesmo processo. Nos casos em que as equipes adotam diferentes modelos de processo, os PSEEs devem permitir que o processo distribuído seja representado por meio de um conjunto de modelos de processo com dependências entre si.

Este capítulo apresentou os aspectos teóricos relacionados à gerência de processos distribuídos, introduzindo os requisitos desejáveis em ambientes que suportem a modelagem e a execução de processos distribuídos definidos por diferentes modelos de processo. Algumas das abordagens de PSEEs com suporte a processos distribuídos encontradas na literatura foram apresentadas e comparadas entre si.

A partir da base teórica apresentada e das abordagens estudadas, foi proposto um modelo para gerência de processos distribuídos. O objetivo do modelo é prover suporte à modelagem e à execução de processos distribuídos compostos por um conjunto de processos locais. O próximo capítulo descreve o modelo proposto, apresentando os principais componentes e funcionalidades.

3 O MODELO APSEE-*GLOBAL*

Este capítulo apresenta a abordagem proposta para gerência de processos distribuídos de desenvolvimento de software. É proposto um modelo para distribuição de processos denominado APSEE-*Global*. Esse modelo está inserido no contexto do grupo de pesquisa Prosoft, no qual o ambiente APSEE, um PSEE que suporta processos de software centralizados foi desenvolvido (LIMA REIS, 2003). O APSEE-*Global* estende o APSEE, provendo um conjunto de funcionalidades para suportar processos distribuídos.

Este capítulo está estruturado como segue: a seção 3.1 apresenta os objetivos específicos do modelo proposto; a seção 3.2 delimita o escopo da solução proposta; a seção 3.3 contextualiza o APSEE-*Global* no grupo de pesquisa Prosoft; o ambiente APSEE é descrito na seção 3.4, enquanto a seção 3.5 apresenta uma visão dos principais aspectos do modelo APSEE-*Global*. A seção 3.6 descreve os componentes do modelo; as funcionalidades são apresentadas na seção 3.7. Por fim, a seção 3.8 algumas considerações sobre o capítulo.

3.1 Objetivos específicos

Com o intuito de propor soluções para os requisitos para ambientes de suporte a gerência de processos distribuídos de software levantados no capítulo 2, são objetivos específicos do APSEE-*Global*:

- Prover suporte à distribuição de modelos de processo, de modo que processos distribuídos sejam representados por um conjunto de processos locais. A interoperabilidade entre os processos locais deve ser definida por meio de modelos de processo globais, com o intuito de atender ao requisito 2, *Definição e aceitação de um processo global*;
- Permitir que os diversos modelos de processo que compõem o processo distribuído sejam definidos segundo a metodologia de processo de cada equipe, como sugere o requisito 3, *Integração e interoperabilidade entre modelos de processo existentes*;
- Prover autonomia às equipes na execução de atividades locais, com dependência entre as equipes apenas na execução de atividades distribuídas, como indicado nos requisitos 4, *Autonomia local e operação independente*, e 5, *Restrição das dependências globais aos processos afetados*;

- Permitir que cada equipe que participa do processo distribuído tenha conhecimento das equipes com as quais interage, e que equipes possam ingressar ou sair do projeto inclusive durante o andamento do processo, de acordo com os requisitos 6, *Percepção (awareness) da configuração*, e 7, *Configuração dinâmica*;
- Para atender ao requisito 1, *Infra-estrutura de comunicação*, o APSEE-*Global* irá utilizar a estrutura de comunicação distribuída proposta pelo Prosoft-Cooperativo (REIS, 1998), que permite a comunicação entre objetos localizados em diferentes máquinas (*hosts*). Sobre essa estrutura serão especificados métodos para comunicação entre instâncias do APSEE-*Global*;
- O requisito 8, *Estado do processo persistente*, será atendido utilizando a própria estrutura de armazenamento persistente de processos do APSEE;
- O APSEE provê um mecanismo de propagação de falhas na execução de atividades, que será estendido para o caso de processos distribuídos, com o objetivo de atender ao requisito 8, *Semântica transacional*.

3.2 Delimitação do escopo

Como definem os objetivos específicos do APSEE-*Global*, este trabalho enfoca a definição de um modelo para gerência de processos distribuídos de desenvolvimento de software. Dentro desse foco, é importante identificar claramente que aspectos são tratados pelo modelo proposto. É igualmente importante delinear quais aspectos estão fora do escopo da proposta. A seguir o escopo do APSEE-*Global* é definido em relação a alguns aspectos relacionados à gerência de processos distribuídos.

- **Distribuição do processo de desenvolvimento**

Como definido no capítulo 2, processos distribuídos são definidos como processos de desenvolvimento de software compostos por um conjunto de sub-processos autônomos (WANG, 2000-a). Os sub-processos são modelados e executados por equipes distintas, que podem adotar diferentes metodologias de desenvolvimento (MASLOVA, 2002). Embora autônomos, esses sub-processos possuem dependências entre si.

Na literatura, o conceito de gerência distribuída do processo aparece intimamente relacionado ao conceito de distribuição do desenvolvimento (DAMIAN, 2004) (HERBSLEB; GRINTER, 2001) (CARMEL, 2003). Processos distribuídos representam, geralmente, projetos com desenvolvimento distribuído. Assim, a gerência de processos distribuídos é fortemente influenciada pelas características encontradas no desenvolvimento fisicamente distribuído (MAIDANTCHICK; DA ROCHA, 2002).

No entanto, a distribuição do processo pressupõe que haja um grau de autonomia e independência entre as equipes, o que pode ser encontrado inclusive em situações nas quais não necessariamente existe distância física entre as equipes.

É o que ocorre, por exemplo, na terceirização de parte do desenvolvimento para outra equipe, responsável pela definição e execução do sub-processo a ela atribuído. Nesse caso, o processo apresenta as características de processos distribuídos, independentemente de as equipes estarem ou não distantes fisicamente.

Outro aspecto a ser considerado é que, em alguns casos, embora as equipes estejam fisicamente distribuídas, o processo de desenvolvimento é conduzido de maneira homogênea, com um único modelo de processo definindo todo o ciclo de vida do processo. Essa situação pode ser observada, por exemplo, no caso de Fábricas de Software, onde cada equipe concentra-se em apenas parte do ciclo de desenvolvimento, mas todas trabalham segundo a metodologia da Fábrica. A gerência do processo de desenvolvimento, nesse caso, é realizada de forma centralizada, com definição de um único modelo de processo, mesmo que as equipes estejam fisicamente distribuídas.

Com base no exposto, observa-se que, em relação à gerência do processo, a definição de desenvolvimento distribuído pode ser estendida com base no conceito de distribuição virtual, introduzido por (MASLOVA, 2002). Segundo esse conceito, o desenvolvimento é distribuído quando é definido por um conjunto de sub-processos desacoplados logicamente e com um grau de independência entre si, sem necessariamente existir distribuição física. Nesse contexto, pode-se definir que o desenvolvimento é distribuído quando as equipes que participam do desenvolvimento possuem um grau de autonomia entre si, cada uma adotando o seu próprio processo de desenvolvimento, sem que haja necessariamente distância física entre elas.

O APSEE-*Global* enfoca as características da gerência de processos distribuídos, e o conceito de desenvolvimento distribuído adotado é o de distribuição virtual.

- **Estratégia de definição do processo global**

Como apresentado no capítulo 2, a modelagem de um processo global pode ser realizada utilizando as estratégias *top-down* e *bottom-up*. Na estratégia *top-down* o processo global é definido em níveis mais altos de granularidade, e o seu detalhamento em níveis mais baixos de granularidade define os processos locais. Esse tipo de definição tem como vantagem prover uma visão global do processo distribuído. A estratégia *bottom-up* define os processos locais, e depois estabelece as relações entre eles. Uma vantagem dessa abordagem é permitir a integração de processos já existentes.

O APSEE-*Global* adota uma abordagem híbrida, com o intuito de obter as vantagens de ambos os modelos. Assim, uma visão geral do processo distribuído é definida, e processos já existentes podem ser integrados.

- **Distribuição x descentralização**

Um sistema distribuído provê uma perspectiva homogênea às suas aplicações, embora sua execução seja fisicamente distribuída entre várias máquinas. Ou seja, a

distribuição é transparente para as aplicações. Um sistema descentralizado, por sua vez, é composto por subsistemas relativamente independentes com algum grau de relacionamento entre eles, e não necessariamente é executado em máquinas distintas. Nesse tipo de sistema, a transparência intencionalmente não é suportada, entre outras razões, para manter a autonomia entre os subsistemas.

O APSEE-*Global* é definido como um sistema descentralizado, pois a independência entre as instâncias de PSEE executadas pelas diferentes equipes deve ser mantida e explicitada.

- **Heterogeneidade**

A heterogeneidade de PSEEs descentralizados pode ser classificada em três níveis: sistema, linguagem e aplicação. A heterogeneidade no nível de sistema diz respeito às diferenças na estrutura interna das instâncias do PSEE. PSEEs com heterogeneidade de linguagem provêem suporte à adoção de diferentes linguagens de modelagem de processos pelas suas instâncias. A heterogeneidade de aplicação diz respeito à definição de diferentes modelos de processo utilizando a mesma linguagem de modelagem de processos.

Suporte à heterogeneidade é, geralmente, um problema complexo. A solução proposta neste trabalho explora um aspecto limitado de heterogeneidade, no contexto da descentralização do ambiente. O modelo proposto é homogêneo no que diz respeito ao sistema e à linguagem de modelagem de processos, pois considera que instâncias do mesmo PSEE compõem o sistema, adotando uma única linguagem de modelagem. No entanto, a heterogeneidade de aplicação é suportada, uma vez que os processos locais podem ser definidos por modelos de processo distintos.

- **Definição e execução de processos**

PSEEs devem prover suporte à modelagem e à execução de processos de software. No caso de PSEEs que provêem suporte a processos distribuídos, suporte à modelagem da distribuição e à execução distribuída também deve ser provido.

O modelo proposto neste trabalho tem seu foco apenas na distribuição, e não contempla os demais aspectos do suporte a processos de software. Assim, a modelagem e a execução de processos está fora do escopo do APSEE-*Global*. A solução proposta será integrada a um PSEE centralizado já existente, incorporando a ele os requisitos para prover suporte à distribuição. Assim, o APSEE-*Global* propõe-se a gerenciar a distribuição de modelos de processo definidos segundo uma linguagem de modelagem de processos e executados por uma máquina de execução de processos existentes.

3.3 Contextualização

O trabalho apresentado nesta dissertação foi desenvolvido no contexto do grupo de pesquisa Prosoft, da Universidade Federal do Rio Grande do Sul, sob a coordenação do Prof. Dr. Daltro José Nunes. O principal objetivo do grupo de pesquisa é a construção de um ambiente de desenvolvimento de software para

apoiar o engenheiro de software desde a fase de análise do problema até a fase de construção do programa.

O ambiente desenvolvido pelo grupo, cuja implementação mais recente é denominada Prosoft-Java, permite que ferramentas de apoio ao desenvolvimento de software sejam especificadas, implementadas e integradas ao ambiente. Isto somente é possível devido à integração funcional[7], sintática[8] e semântica[9] que o ambiente provê para suas ferramentas.

Nos últimos anos, diversos trabalhos de mestrado e doutorado do Instituto de Informática da UFRGS foram desenvolvidos no contexto do grupo Prosoft, gerando ferramentas incorporadas ao ambiente Prosoft-Java. Dentre os trabalhos desenvolvidos no grupo Prosoft, pode-se citar: sistema especialista para o desenvolvimento de software (MORAES, 1997), suporte ao desenvolvimento cooperativo de software (REIS, 1998), abordagem para reutilização de especificações de requisitos (PIMENTA, 1998), modelo para decisões em grupo no desenvolvimento de software (ALVES, 2002), mecanismo para prototipação de software (RANGEL, 2003).

Além de suporte às etapas do desenvolvimento de software, uma das principais linhas de pesquisa do grupo Prosoft tem como foco a construção de mecanismos que auxiliem na gerência de processos de software. Essa linha surgiu a partir do gerenciador de processos de software proposto por Lima Reis (1998), e consiste de ferramentas que apóiam a modelagem, execução (LIMA REIS, 2003), reutilização (REIS, 2002), visualização (SOUSA, 2003) e simulação (SILVA, 2001) de processos de software. Esse conjunto de ferramentas constitui um ambiente para gerência de processos de software denominado APSEE (*A Process-centered Software Engineering Environment)* (LIMA REIS, 2003), também integrado ao ambiente Prosoft-Java.

Atualmente, as pesquisas acerca de processos de software no grupo Prosoft envolvem estudantes e pesquisadores no desenvolvimento de ferramentas integradas ao APSEE que atuam sobre o meta-modelo de processo proposto por Lima Reis (2003). Ferramentas para auxílio automatizado à adaptação de processos de software (MAIA; FREITAS; NUNES; STEINMACHER, 2004) (MAIA; FREITAS; NUNES, 2004) e para desenvolvimento e gerência de processos educacionais (DAHMER, 2005) estão sendo desenvolvidas dentro do contexto do APSEE.

O conjunto de ferramentas que compõem o APSEE tem como foco o suporte a processos centralizados de desenvolvimento de software. Como exposto no

---

[7] Por integração funcional, entende-se que cada ferramenta do ambiente deve ter uma função bem definida e interagir com outras através de sua interface.

[8] A integração sintática permite que dados gerados por uma ferramenta sejam lidos e usados por outras ferramentas.

[9] Graças à integração semântica, operações de uma ferramenta podem ser construídas a partir de outras ferramentas.

capítulo 2, ambientes de suporte à modelagem e à execução de processos distribuídos devem atender a requisitos adicionais aos especificados para a execução de processos centralizados. Este trabalho estende o ambiente APSEE para permitir também o suporte a processos distribuídos de desenvolvimento.

Existem na literatura diversos outros PSEEs que provêem suporte a processos de software centralizados, como EPOS (JACCHERI; LARSEN; CONRADI, 1992), SPADE (BANDINELLI et al, 1994), ExPSEE (GIMENES, 2000), Estação TABA (TRAVASSOS, 1994) e Merlin (JUNKERMANN et al, 1994). A escolha pelo APSEE como o PSEE a ser estendido nesse trabalho é justificada por uma série de razões, a saber:

- Embora a maioria dos PSEEs permita a integração de diferentes ferramentas (através da definição de uma interface pública para componentes escritos pelo usuário), muitos dos PSEEs – do ponto de vista do suporte à modelagem e execução de processos – são sistemas monolíticos, nos quais é muito difícil propor mudanças na infra-estrutura básica de modelagem e execução de processos;
- A documentação disponível acerca das implementações existentes muitas vezes é insuficiente, dificultando sobremaneira a extensão dos ambientes. O APSEE apresenta uma grande vantagem sob esse aspecto, por ser especificado formalmente;
- O fato deste trabalho ser desenvolvido dentro do grupo de pesquisa Prosoft, no qual o APSEE também foi concebido, provê maior facilidade na modificação da estrutura interna do APSEE, além de permitir o aproveitamento da base de conhecimento do grupo.

Para definir essa extensão do APSEE, denominada APSEE-*Global*, foi utilizada uma metodologia de acordo com os objetivos gerais traçados no capítulo 1. Com base na definição informal dos requisitos do APSEE-*Global*, obteve-se a especificação formal do modelo, construiu-se o protótipo e definiu-se um exemplo para demonstrar a utilização do protótipo construído. A especificação formal do APSEE-*Global* é composta pela definição dos dados e das transformações realizadas sobre esses dados, que representam as funcionalidades do modelo. Os dados são especificados segundo a notação de Tipos Abstratos de Dados (TADs) do Prosoft-Algébrico (NUNES, 1994), enquanto as transformações que podem ser realizadas sobre os dados são especificadas por meio de regras de Gramática de Grafos (DÉHARBE et al, 2000) (RIBEIRO, 2000).

O diagrama[10] da Figura 3.1 ilustra as atividades realizadas para definição do APSEE-*Global*. Esse diagrama, bem como os demais diagramas utilizados neste trabalho para representar fluxo de atividades, é descrito segundo a notação IDEF0

[10] Este trabalho está inserido em um projeto de cooperação internacional que envolve a Universidade Federal do Rio Grande do Sul e a Universität Stuttgart (Alemanha). Assim, os diagramas definidos neste trabalho (IDEF0, UML, definição de classes do Prosoft-Algébrico e a definição das regras de Gramática de Grafos) foram construídos em inglês.

(*Integration Definition for Function Modeling*), formalizada pelo *Federal Information Processing Standards* (FIPS, 1993). A linguagem de modelagem de sistemas proposta pelo padrão IDEF0 é baseada no padrão SADT (*Structured Analysis and Design Technique*) (ROSS, 1985). Um modelo IDEF0 é composto por uma série hierárquica de diagramas que apresentam, gradativamente, um nível maior de detalhe, descrevendo funções e suas interfaces no contexto de um sistema. Por causa da uniformidade e iteração limitadas para facilitar o processo, este método é largamente utilizado em modelagem de processos industriais, gerenciais, executivos, etc., e é suportado por um grande número de ferramentas comerciais de modelagem de sistemas.

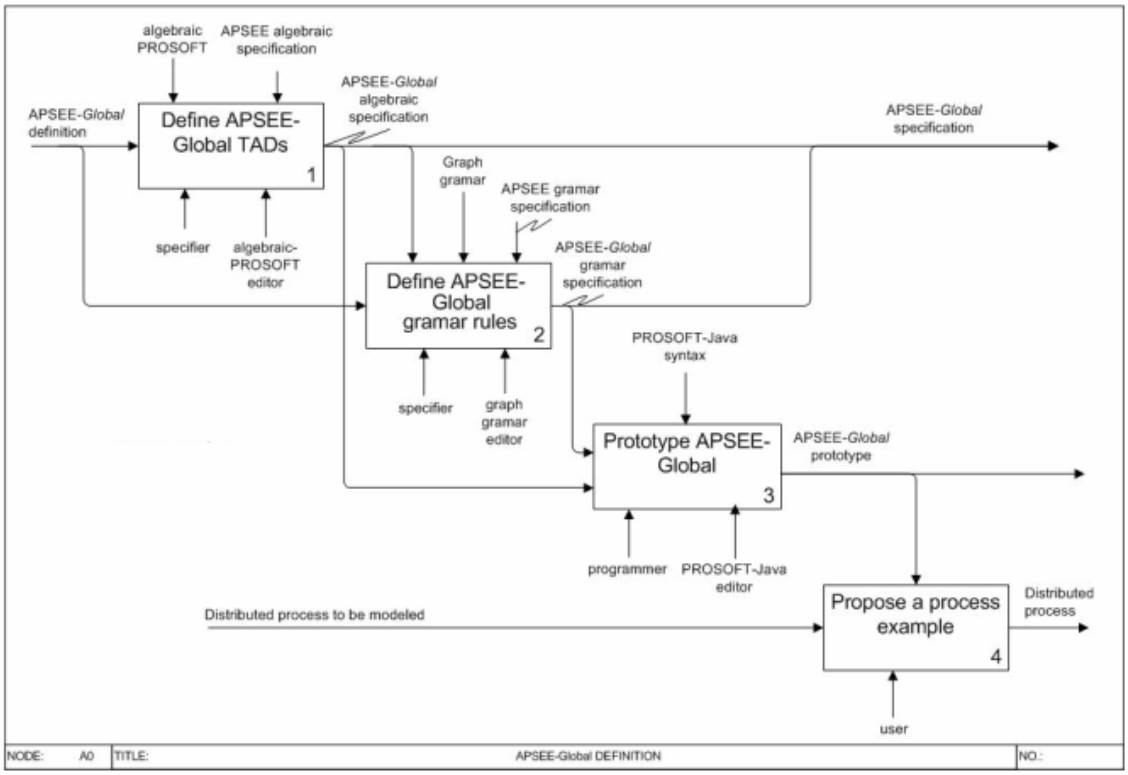

Figura 3.1: Processo de definição do APSEE-*Global*

A função A0.1 representa a definição dos TADs do APSEE-*Global*. Os tipos de dados foram definidos de forma a complementarem os TADs definidos por (LIMA REIS, 2003) para especificação dos dados do APSEE. Para especificar as funcionalidades do modelo, regras de transformação de grafos foram definidas (função A0.2). Com base na especificação dos TADs e das regras de transformação, o APSEE-*Global* foi prototipado no ambiente Prosoft (função A0.3). Posteriormente, um processo distribuído foi modelado e executado, para exemplificar o funcionamento do protótipo (função A0.4).

3.4 O ambiente APSEE

Como mencionado na seção anterior, o APSEE-*Global* estende o ambiente APSEE, incorporando funcionalidades para prover suporte à distribuição de processos de software. Esta seção apresenta uma visão geral do APSEE, por duas

razões: primeiro, para introduzir conceitos que serão utilizados ao longo do texto. Segundo, para criar uma distinção clara do trabalho realizado nesta dissertação do trabalho realizado anteriormente por outros autores para definição das funcionalidades existentes no APSEE.

A Figura 3.2 apresenta os principais componentes do APSEE, organizados em três níveis (interação com usuário, mecanismos para gerência de processos e meta-modelo), os quais são descritos nas subseções a seguir.

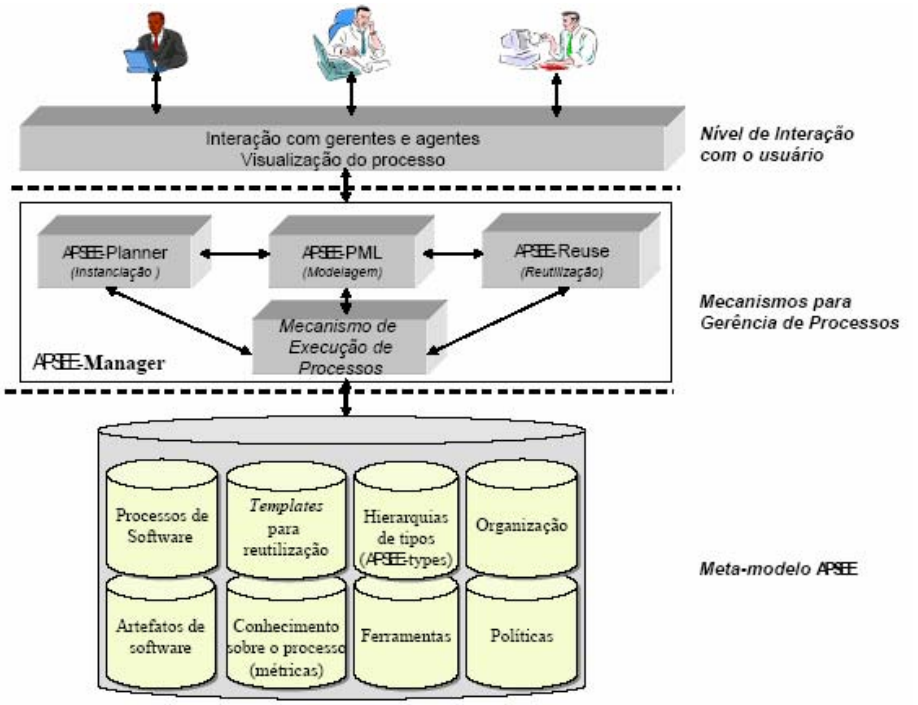

Figura 3.2: Visão geral dos principais componentes do modelo APSEE (LIMA REIS, 2003)

3.4.1 Interação com o usuário

Esse nível provê mecanismos de interação para os diferentes usuários por meio de interfaces gráficas disponibilizadas no ambiente. Por exemplo, a execução de um processo pode ser vista sob o ponto de vista do gerente, com as dependências entre atividades; ou do ponto de vista do desenvolvedor, com agenda de tarefas a serem realizadas.

3.4.2 Mecanismos para gerência de processos

O nível de mecanismos para gerência de processos descreve um conjunto de serviços para modelagem, instanciação[11], reutilização e execução de processos de

[11] A instanciação de modelos de processo de software associa a um modelo de processo abstrato (que descreve de forma abstrata o processo de software) características de um projeto de desenvolvimento específico, por meio da definição do cronograma, da alocação de desenvolvedores e recursos para as tarefas, dentre outros ajustes. Uma vez instanciado, o processo pode então ser executado.

software. A seguir são listados os mecanismos para gerência de processos disponibilizados pelo APSEE:

- Mecanismo de execução de processos: coordena as atividades do processo em execução, podendo executar processos incompletos e permitindo a alteração do processo durante a execução (LIMA REIS, 2003);
- APSEE-Planner: responsável por fornecer assistência automática para instanciação de recursos e agentes durante a execução de processos de software através de políticas programáveis pelo usuário (REIS; LIMA REIS; NUNES, 2001) (REIS et al, 2002);
- APSEE-PML: linguagem gráfica para modelagem de processos que permite a definição de processos de software e de seus relacionamentos com outros componentes do meta-modelo (LIMA REIS, 2003);
- APSEE-Reuse: componente de apoio à reutilização de processos responsável pela criação, recuperação e adaptação de *templates* reutilizáveis (REIS, 2002).

O mecanismo de execução de processos constitui a base que interpreta os modelos de processo (descritos na linguagem APSEE-PML) e exerce um papel central na organização dos mecanismos existentes. Desse modo, o mecanismo de execução é responsável por fornecer dados sobre a dinâmica da execução de processos que são úteis para o mecanismo de instanciação APSEE-Planner. Além disso, o mecanismo de execução fornece informação sobre processos abstratos, instanciados e executados para o APSEE-Reuse.

3.4.3 O meta-modelo APSEE

O meta-modelo APSEE facilita a integração dos serviços providos pelos mecanismos de gerência de processos do APSEE. Uma breve descrição dos componentes do meta-modelo é apresentada a seguir.

- Processos de software: descrevem os modelos de processo de software;
- *Templates* para reutilização: descrevem processos abstratos reutilizáveis;
- Hierarquias de tipos (*APSEETypes*): relacionadas aos componentes do APSEE e utilizadas na descrição de processos abstratos e de reutilização, além de permitir o raciocínio sobre elementos do processo de forma genérica;
- Organização: representa os recursos de apoio utilizados pelas atividades, as pessoas da organização (agentes), suas habilidades, afinidades, cargos e grupos de trabalho;
- Artefatos de software: correspondem aos itens de dados manipulados, criados e utilizados durante o desenvolvimento de software;
- Conhecimento sobre o processo (métricas): permite definir e armazenar métricas e estimativas para os componentes do processo, as quais podem ser consultadas dinamicamente durante a execução do mesmo;

- Ferramentas: armazena informações sobre as ferramentas disponíveis no ambiente para realização das atividades;
- Políticas: são regras definidas pelo usuário que permitem estabelecer quando um modelo de processos está correto (estáticas), como atividades devem ser instanciadas (de instanciação) e que ações realizar na ocorrência de eventos durante a execução (dinâmicas).

Dentre os componentes que formam o APSEE, as *Hierarquias de Tipos*, os *Processos de Software* e os *Artefatos de Software* são utilizados pelo APSEE-*Global*. Por essa razão, eles serão apresentados com mais detalhes.

3.4.3.1. Hierarquias de tipos

Os principais componentes da arquitetura APSEE são tipados, ou seja, são classificados através de tipos predefinidos. O pacote *APSEETypes* contém as hierarquias de tipos para os componentes do APSEE. São propostas as hierarquias de tipos principais, fornecidas juntamente com o modelo, mas o usuário do ambiente poderá criar novas hierarquias de tipos e instâncias para hierarquias existentes de acordo com as necessidades da organização ou do processo.

As hierarquias de tipos propostas pelo APSEE são: recursos, cargos, grupos, habilidades, métricas, atividades, conexões, políticas, artefatos e ferramentas. A Figura 3.3 apresenta a estrutura do componente *APSEETypes*.

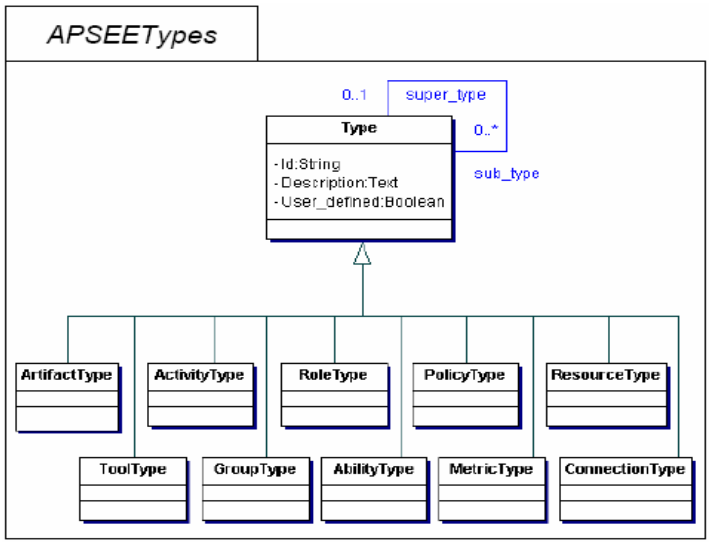

Figura 3.3: Hierarquias de tipos do APSEE
(LIMA REIS, 2003)

O pacote *APSEETypes* exerce um papel importante na descrição do modelo, visto que a maioria dos componentes do meta-modelo APSEE estão associados a um tipo (*Type*) que deve existir na hierarquia correspondente. As hierarquias de tipos relacionadas aos componentes do APSEE permitem que sejam descritos processos abstratos que podem ser refinados conforme necessário para execução ou usados para reutilização e ainda apóiam a descrição de mecanismos que lidam com elementos genéricos de processos. Essa estrutura de tipos aumenta a flexibilidade do modelo, o que permite a descrição de regras gerais (políticas) que se referem a

tipos e não a instâncias, e de uma abordagem para reutilização de processos (REIS, 2002).

3.4.3.2. Processos de software

Os processos de software são representados por modelos de processo, que têm como objetivo descrever as características de um processo de software e seu relacionamento com os outros componentes do meta-modelo. Como ilustra a Figura 3.4, um modelo de processo é composto basicamente de atividades e conexões entre atividades.

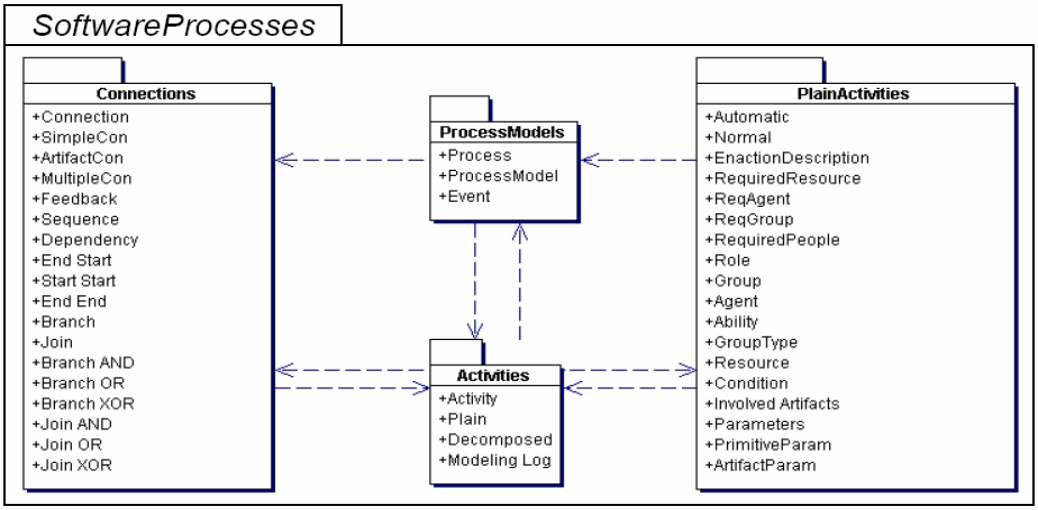

Figura 3.4: Estrutura do componente Processos de Software
(LIMA REIS, 2003)

Uma atividade pode ser simples ou decomposta. Se a atividade for decomposta, então é definida por um novo modelo de processo, de forma recursiva. Caso contrário, trata-se de uma atividade-folha na decomposição do modelo de processo (*PlainActivity*). Atividades simples podem ser classificadas como normais ou automáticas. Atividades automáticas não consomem recursos nem tempo e são realizadas através de chamadas a operações de ferramentas integradas no ambiente. Já as atividades normais necessitam de recursos, agentes e envolvem artefatos de entrada e saída. O estado da atividade armazena a situação de cada atividade e serve de referência para a transição de estados de todo o processo de software. Os possíveis estados de uma atividade simples são descritos a seguir:

- *Waiting*: as dependências da atividade ainda não estão satisfeitas;
- *Ready*: pronta para começar;
- *Active*: a atividade está sendo realizada pelos agentes responsáveis (pelo menos um agente está trabalhando na atividade);
- *Paused*: todos os agentes solicitaram pausa da atividade;
- *Finished*: a atividade foi concluída. Se a atividade for cooperativa, significa que todos os agentes concluíram;
- *Canceled*: a atividade foi cancelada antes de iniciar;

- *Failed*: a atividade falhou após o início por decisão dos agentes ou do gerente.

As conexões servem para interligar atividades, representando o fluxo de controle e de dados do processo. De acordo com o tipo de fluxo que expressam, as conexões podem ser classificadas em: conexões simples, conexões múltiplas e conexões de artefato.

As conexões simples podem ser de seqüência ou de *feedback* (retorno a uma atividade anterior), enquanto que as conexões múltiplas são *branch* e *join* combinados com operadores lógicos. As conexões simples e múltiplas também possuem um tipo de dependência (*end-start* – a atividade destino somente pode iniciar quando a origem terminar, *start-start* – a atividade destino pode iniciar somente após o início da origem, *end-end* – a atividade destino somente pode terminar quando a origem terminar) que influencia diretamente na execução das atividades e pode representar diferentes dependências encontradas em processos reais. As conexões de artefato não influenciam no fluxo de controle do processo; descrevem o fluxo de informações entre atividades, podendo ser ligadas a conexões múltiplas. Os tipos de conexão e dependências são detalhados a seguir.

- Conexão simples: define o fluxo de controle entre duas atividades (origem e destino). Este fluxo pode ser de seqüência ou de *feedback*:
  - Conexão simples de seqüência: indica que a atividade destino depende da atividade origem;
  - Conexão simples de *feedback*: é utilizada para indicar o retorno a uma atividade já realizada desde que uma condição seja satisfeita;
- Conexão múltipla: também define o fluxo de controle, porém pode envolver várias atividades e conexões múltiplas. Existem dois tipos de conexão múltipla, ambos subdivididos pela aplicação de operadores lógicos (AND, XOR, OR) e tipos de dependência (*end-start*, *start-start*, *end-end*):
  - *Branch*: possui uma atividade ou conexão múltipla origem e várias atividades ou conexões múltiplas destino. A semântica da conexão depende do operador lógico associado (AND – todas, XOR – somente uma ou OR – um subconjunto de) e do tipo de dependência utilizado (*end-start*, *start-start* ou *end-end*). A semântica de conexões *branch* associadas a operadores lógicos OR ou XOR é definida também por uma condição lógica associada à atividade destino. A atividade destino só pode ser executada ou finalizada se, além do tipo de dependência da conexão ser satisfeito, a condição lógica associada à atividade destino for avaliada como verdadeira;
  - *Join*: possui várias atividades ou conexões múltiplas origem e uma atividade ou conexão múltipla destino. Uma aplicação dessa conexão seria que assim que todas as atividades origem terminarem, então uma atividade destino pode começar (*join* AND *end-start*). Os operadores lógicos e tipos de dependências também se aplicam ao *join*;

- Conexão de artefato: define o fluxo de dados do processo através da passagem de artefatos entre atividades. Um artefato produzido por uma atividade pode servir de entrada para outras e essa informação é explicitada através da conexão de artefato.

O estado do modelo de processo é determinado dinamicamente em função de seu conteúdo e pode receber os seguintes valores:

- *Requirements:* o modelo de processo não possui nenhuma atividade, apenas uma descrição textual de seus requisitos, ou ainda, possui somente atividades decompostas que estão no estado *requirements*;
- *Abstract*: o modelo de processo possui atividades que não estão relacionadas ao contexto da organização, ou seja, são genéricas. Neste estado as atividades indicam apenas os cargos necessários, os tipos de recurso e tipos de artefato;
- *Instantiated*: neste estado o modelo de processo já possui atividades instanciadas e pode ser executado;
- *Enacting*: pelo menos uma das atividades do modelo de processo está sendo executada;
- *Finished*: todas as atividades do modelo de processo foram concluídas;
- *Failed*: todas as atividades do modelo de processo falharam. Esta situação é possível através da solicitação de um gerente. As conseqüências da falha são tratadas pelo mecanismo de execução;
- *Canceled*: o modelo de processo foi cancelado e todas as atividades estão canceladas. Também é necessária a solicitação de um gerente para que o processo assuma este estado. A diferença com relação à falha é que somente processos não iniciados podem ser cancelados;
- *Mixed*: este estado é caracterizado quando os componentes do processo estão em diferentes estados sem caracterizar nenhum dos estados anteriores. Por exemplo, um processo que possui uma atividade falhada, outra concluída e uma outra atividade decomposta no estado *requirements* está no estado *mixed*.

3.4.3.3. Artefatos de software

O ambiente ao qual o APSEE está integrado é o ambiente PROSOFT, que possui um componente para tratar de gerência de objetos, chamado PROSOFT Cooperativo (REIS, 1998). Este componente trata objetos de qualquer tipo no ambiente, guardando seu estado, gerenciando o seu acesso e suas versões. O componente PROSOFT Cooperativo foi integrado na arquitetura do APSEE dentro do componente *Artifacts*, apresentado na Figura 3.5. Esse componente acrescenta mais atributos que se mostraram importantes do ponto de vista de gerência de processos de software, e possui as seguintes características:

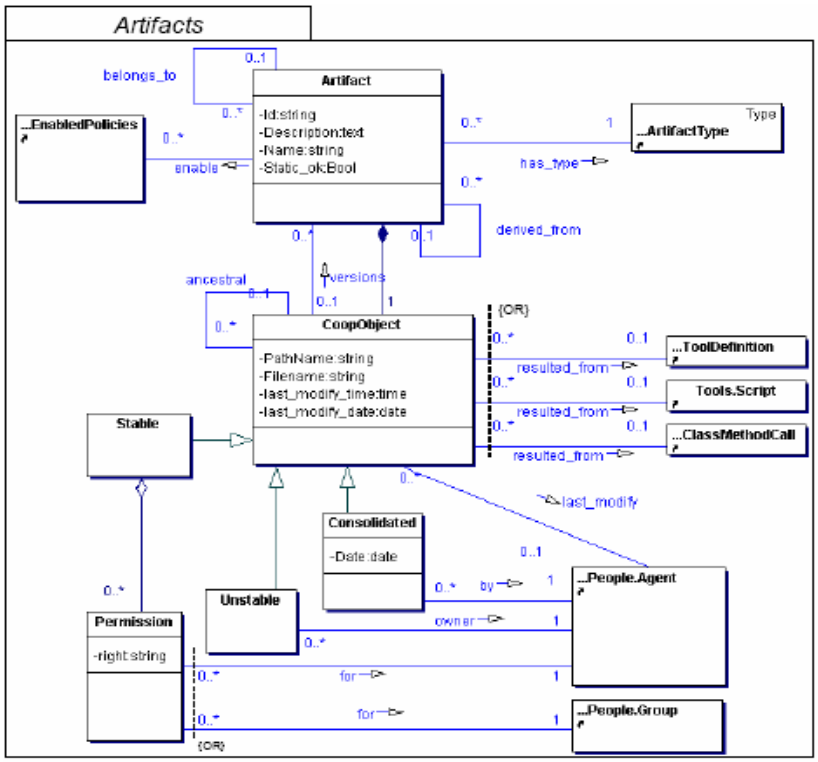

Figura 3.5: Definição de artefatos de software

- Um identificador, um tipo na hierarquia de tipos de artefatos do APSEE-Types e uma descrição;
- Políticas habilitadas (*EnabledPolicies*) associadas ao artefato;
- O objeto cooperativo em si (*CoopObject*), que representa o uso do componente Objeto Cooperativo definido em (REIS, 1998), com as mesmas características;
- Uma relação de derivação entre artefatos (associação *derived_from*), indicando qual artefato serviu de base para a construção deste. Neste caso, pode-se representar, por exemplo, a informação de que um diagrama de classes em UML derivou de uma especificação de requisitos do sistema;
- Uma relação de composição entre artefatos (associação *belongs_to*). Por exemplo, um artefato que representa uma especificação de requisitos pode conter vários diagramas e textos.

3.5 Visão geral do modelo APSEE-*Global*

Esta seção introduz o APSEE-*Global*, um modelo de gerência de processos distribuídos de desenvolvimento de software proposto com o intuito de atender aos objetivos específicos mencionados na seção 3.1. O APSEE-*Global* estende o ambiente APSEE, provendo funcionalidades para a gerência de modelos de processo distribuídos.

A distribuição de processos do APSEE-*Global* utiliza a abordagem de delegação de atividades (VAN DER AALST, 1999) (BECKER et al, 2001).

Segundo essa abordagem, uma equipe é responsável pela modelagem e execução do processo distribuído, podendo contratar outras equipes e delegar a elas a execução de uma ou mais atividades do processo. Quando uma equipe delega atividades, ela está estabelecendo o que deve ser executado. A definição de como a execução será conduzida cabe à equipe contratada.

Equipes podem assumir, no mesmo projeto de desenvolvimento, os papéis de contratante e contratada. Ou seja, uma equipe que executa atividades delegadas pode também delegar atividades a outras equipes. Nesse caso, além do papel de contratada, a equipe também assume o papel de contratante em relação às equipes às quais ela delega atividades. Embora a relação contratante-contratada seja evidenciada nessa abordagem, todas as equipes têm o objetivo comum de desenvolver o mesmo produto de software. Assim, equipes que firmam contratos para execução de atividades são vistas entre si como parceiros durante um projeto de desenvolvimento de software.

Em situações reais, cada equipe pode adotar uma metodologia própria de desenvolvimento de software, o que se reflete na definição de modelos de processo distintos pelos parceiros de projeto. Embora utilizem os seus próprios modelos de processo, equipes contratadas devem executar atividades a elas delegadas, definidas em um modelo de processo possivelmente distinto do adotado para condução de suas atividades. Para que as atividades delegadas à equipe sejam compatíveis com as atividades locais (definidas no modelo de processo local adotado pela equipe), é estabelecida uma correspondência entre atividades delegadas e atividades locais, de modo que as atividades delegadas sejam executadas internamente de acordo com o modelo de processo local.

A correspondência entre atividades delegadas e atividades locais é estabelecida por meio do mapeamento entre as atividades delegadas e atividades definidas no processo de software local da equipe. A equipe pode, por exemplo, refinar ou fragmentar atividades sob sua responsabilidade, definindo que uma atividade delegada a ela corresponde a um conjunto de atividades definidas no seu modelo de processo. Atividades delegadas, então, são definidas localmente pela combinação das atividades que as representam.

A Figura 3.6 ilustra um exemplo de correspondência entre atividades delegadas e locais. No exemplo, são exibidos dois fragmentos de modelos de processo[12]: o fragmento na parte superior da figura representa as atividades delegadas à equipe; o fragmento na parte inferior representa atividades do modelo de processo local. No exemplo, as atividades *Prototipação* e *Desenvolvimento* foram delegadas. As setas pontilhadas indicam que a atividade delegada de *Prototipação* é mapeada pela atividade de *Prototipação* definida no modelo de processo local da equipe. Já

[12] Os fragmentos estão definidos segundo a notação para modelagem de processos do APSEE, onde elipses representam atividades, retângulos representam conexões de controle e caixas tridimensionais representam conexões de artefato. As conexões são associadas a atividades por meio de setas contínuas, que indicam atividades de origem (setas de entrada) e atividades destino (setas de saída).

a atividade de *Desenvolvimento* é representada no modelo de processo local pelas atividades de *Implementação* e *Finalização da documentação do sistema*.

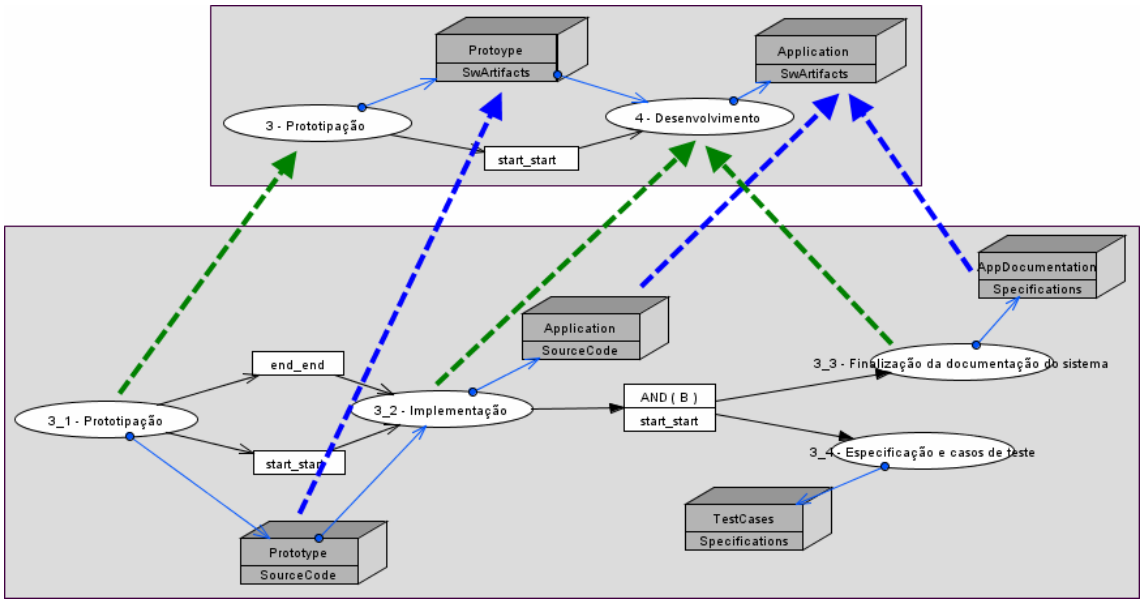

Figura 3.6: Correspondência entre atividades e artefatos delegados e locais

A correspondência entre os artefatos produzidos pelas atividades delegadas também precisa ser realizada. No exemplo apresentado, a correspondência entre o artefato da atividade de prototipação é direta. Já o artefato gerado pela atividade de desenvolvimento é composto pelos artefatos gerados pelas atividades de implementação e de finalização da documentação do sistema.

Além de prover suporte à modelagem da distribuição de processos de software, o APSEE-*Global* também provê suporte à sincronização durante a execução dos modelos de processo. Informações sobre o andamento da execução de atividades delegadas são disponibilizadas à equipe contratante, para que o seu modelo de processo reflita a execução das atividades delegadas. Informações também são trocadas entre os modelos de processo dos parceiros de projeto para garantir a consistência do processo. Verificações de consistência são realizadas durante a modelagem e a execução dos processos, com o intuito de garantir que as dependências entre atividades não sejam violadas.

O diagrama apresentado na Figura 3.7 ilustra os pacotes de classes que compõem o APSEE-*Global*, bem como as dependências em relação aos pacotes do meta-modelo APSEE. O diagrama é definido segundo a notação UML (OMG, 2004). Essa escolha foi realizada levando-se em conta a larga utilização de UML na literatura e o alto nível de abstração da notação, o que facilita o entendimento.

Os principais componentes propostos pelo modelo APSEE-*Global* são apresentados brevemente a seguir.

- **Parceiros** (*Partners*): representam as equipes participantes de um projeto de desenvolvimento distribuído;
- **Processos distribuídos** (*DistributedProcesses*): armazenam informações sobre a delegação de modelos de processo a equipes contratadas;

- **Processos remotos** (*RemoteProcesses*): contêm informações sobre atividades delegadas por equipes contratantes.

Os pacotes *DistributedProcesses* e *RemoteProcesses* relacionam-se com os pacotes *SoftwareProcesses* e *Artifacts* do APSEE, que contêm, respectivamente, os modelos de processo e os artefatos de software. Já o pacote *Partners* depende da hierarquia de tipos do APSEE (*APSEETypes*). Os pacotes *SoftwareProcesses*, *Artifacts* e *APSEETypes* foram apresentados na seção 3.4.

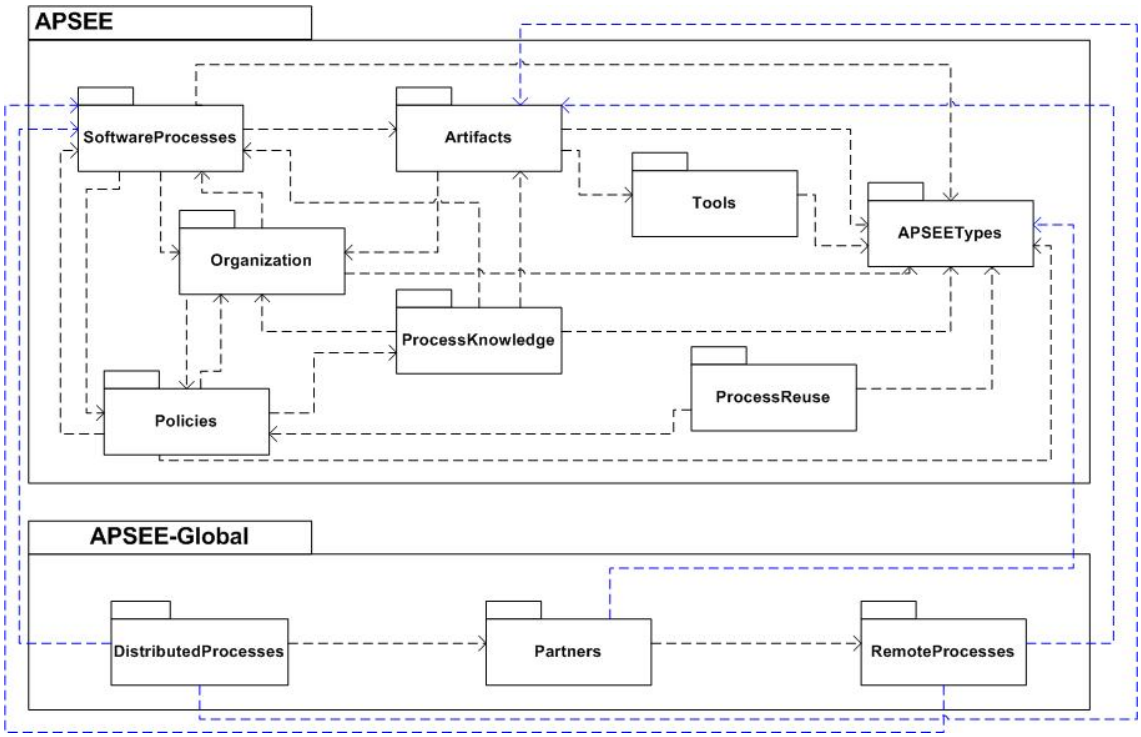

Figura 3.7: Componentes do APSEE-*Global*

Para gerenciar os processos distribuídos, o modelo APSEE-*Global* propõe as seguintes funcionalidades:

- **Definição de parceiros**, que permite a cada equipe definir quais os parceiros com quem ela interage nos seus processos distribuídos de software.
- **Modelagem da distribuição de processos**:
  - *Distribuição de processos*, que associa a um modelo de processo um conjunto de parceiros para os quais as atividades do processo podem ser delegadas;
  - *Delegação de atividades*, que atribui a um ou mais parceiros a responsabilidade pela execução de determinadas atividades de um modelo de processo distribuído;
  - *Mapeamento de processos*, que relaciona, a cada processo delegado à equipe, o modelo de processo local que executará as atividades delegadas;

- *Mapeamento de atividades*, que define a correspondência entre atividades definidas no modelo de processo local e atividades delegadas à equipe;
- *Mapeamento de artefatos*, que relaciona os artefatos de saída das atividades delegadas a artefatos produzidos pelas atividades que a representam.

- **Sincronização da execução dos processos:**
  - *Atualização de estado de atividades delegadas*, que atualiza os modelos de processo dos parceiros para refletir a execução de atividades delegadas;
  - *Propagação de falhas*, que propaga os efeitos de falhas na execução de atividades delegadas;
  - *Propagação de cancelamento*, que propaga os efeitos do cancelamento de atividades delegadas;

As seções a seguir discutem com mais detalhes os componentes e as funcionalidades propostas no modelo APSEE-*Global*.

3.6 Componentes

Como ilustrado pelo diagrama de pacotes da na Figura 3.7, os principais componentes do modelo APSEE-*Global* são parceiros, processos distribuídos e processos remotos. A seguir são apresentadas as classes UML que compõem cada um desses componentes.

3.6.1 Parceiros

O pacote *Partners*, apresentado na Figura 3.8, é composto por uma única classe, *Partner*. Ela armazena as informações sobre os parceiros com os quais a equipe interage em processos de desenvolvimento de software. A cada parceiro estão associados os processos remotos que ele delegou à equipe. Os atributos *APSEEId* e *Host* identificam unicamente o parceiro, fazendo referência à instância do APSEE-*Global* que ele utiliza.

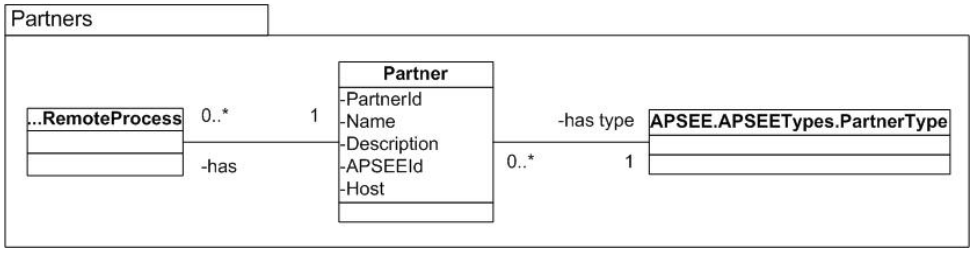

Figura 3.8: O pacote *Partners*

Os parceiros possuem tipos, definidos de acordo com a hierarquia de tipos do APSEE (*APSEETypes*). Por exemplo, pode-se definir um parceiro como interno (pertence à mesma organização desenvolvedora de software) ou externo (pertence a outra organização de software).

3.6.2 Processos distribuídos

Processos distribuídos representam modelos de processo que possuem atividades delegadas a parceiros. O pacote *DistributedProcesses* (Figura 3.9) apresenta as classes envolvidas na distribuição de processos. Os processos distribuídos são definidos pela classe *Process*, do APSEE.

Atividades decompostas ou normais podem ser delegadas. A delegação de uma atividade decomposta é equivalente à delegação de cada uma das suas sub-atividades, e serve para simplificar o processo de delegação. Atividades automáticas não podem ser delegadas pelo fato de que, quando uma atividade é automática, a sua execução é gerenciada pela máquina de execução, sem intervenção humana. Assim, não é necessário delegá-la a outra equipe, já que ela independe do controle manual.

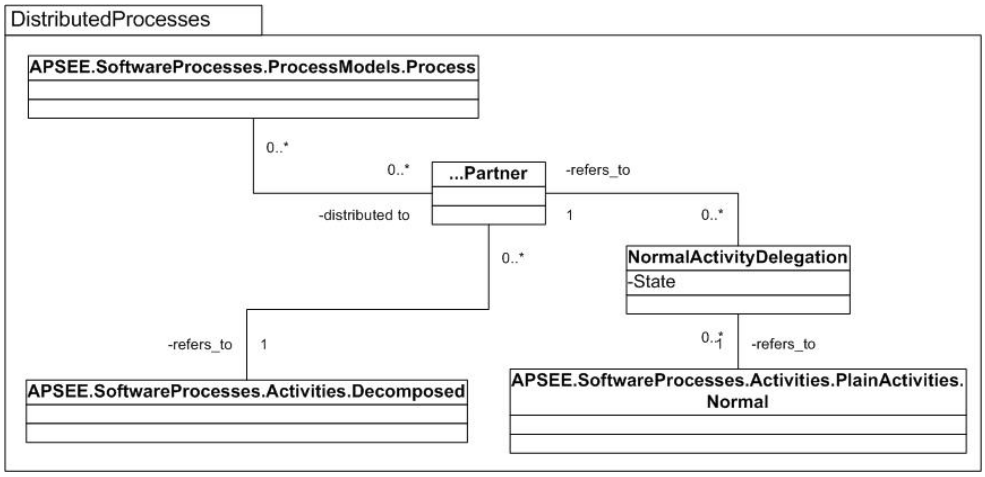

Figura 3.9: O pacote *DistributedProcesses*

O estado de uma atividade normal delegada é definido pelo andamento da sua execução pelas equipes contratadas, bem como pelas dependências existentes no modelo de processo local da equipe contratante. O estado de uma atividade normal delegada é definido da seguinte forma:

- *Waiting*: existem dependências para o inicio da atividade não satisfeitas no modelo de processo da equipe contratante;
- *Ready*: pronta para começar, ou seja, as dependências definidas modelo de processo da equipe contratante estão satisfeitas;
- *Active*: pelo menos um dos parceiros contratados está executando a atividade;
- *Paused*: todos os parceiros solicitaram pausa da atividade;
- *Finished*: a atividade foi concluída. Isso significa que todos os parceiros concluíram a execução da atividade;
- *Canceled*: a atividade foi cancelada antes de iniciar, por decisão da equipe contratante. Esse cancelamento é propagado às equipes contratadas;

- *Failed*: a execução da atividade falhou após o início em todos os parceiros que a executam, ou o gerente da equipe contratante solicitou a falha. No primeiro caso, a falha é propagada ao modelo de processo da equipe contratante. Já no segundo caso, as equipes contratadas são informadas da falha.

O estado de execução de atividades decompostas delegadas é definido pela combinação de estados das atividades-filha, de modo análogo ao que acontece com atividades decompostas executadas localmente.

3.6.3 Processos remotos

O pacote *RemoteProcesses*, ilustrado na Figura 3.10, representa os processos delegados por parceiros à equipe. Processos remotos são mapeados para processos locais. O mapeamento entre um processo local e um processo remoto indica que as atividades do processo remoto podem ser mapeadas por atividades do processo local.

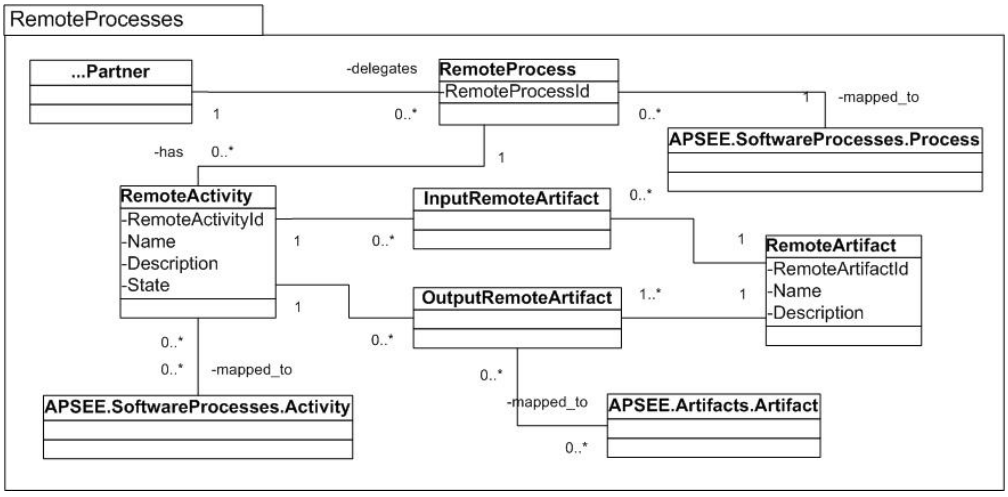

Figura 3.10: O pacote *RemoteProcesses*

Processos remotos são compostos de atividades remotas. Cada atividade remota é mapeada para uma ou mais atividades locais. O mapeamento influencia na definição do estado da atividade remota, como descrito a seguir:

- *Waiting*: existem dependências não satisfeitas para o início da execução da atividade no modelo de processo da equipe contratante;
- *Ready*: pronta para começar, ou seja, as dependências para o início da execução no modelo de processo da equipe contratante foram satisfeitas;
- *Active*: a atividade remota é mapeada para pelo menos uma atividade local que está em execução;
- *Paused*: todas as atividades locais para as quais a atividade remota é mapeada foram pausadas;
- *Finished*: todas as atividades locais para as quais a atividade remota é mapeada foram concluídas;

- *Canceled*: a atividade remota foi cancelada antes de iniciar, por decisão da equipe contratante;
- *Failed*: todas as atividades locais para as quais a atividade remota é mapeada falharam após o início, ou a equipe contratante solicitou a falha.

Atividades remotas podem consumir e produzir artefatos. Os artefatos consumidos pela atividade remota são representados pela classe *InputRemoteArtifact*, e devem servir de base para que a equipe contratada execute a atividade. Os artefatos produzidos pela atividade remota, representados pela classe *RemoteOutputArtifact*, devem ser mapeados para artefatos produzidos pelas atividades locais para as quais a atividade remota é mapeada.

3.7 Funcionalidades

Esta seção detalha as funcionalidades do modelo APSEE-*Global* introduzidas na seção 3.5. A Figura 3.11 ilustra, utilizando a notação IDEF0, as principais funções relacionadas à gerência de processos de software distribuídos. É importante destacar que as funções A0.2 (modelagem de processos de software) e A0.4 (execução de processos de software) são providas pelo ambiente APSEE, e foram inseridas no diagrama com o intuito de ressaltar a integração entre o APSEE-*Global* e o APSEE. A seguir são descritas as funções que compõem o diagrama.

3.7.1 Definição de parceiros

A definição dos parceiros com os quais a equipe interage é representada pela função A0.1. O gerente de projeto da equipe realiza essa definição com o auxílio de um editor de parceiros fornecido pelo APSEE-*Global*. A interação entre a equipe e os seus parceiros pode ocorrer em diferentes projetos de desenvolvimento, tanto pela delegação de atividades ao parceiro quanto pelo mapeamento de atividades a ele delegadas.

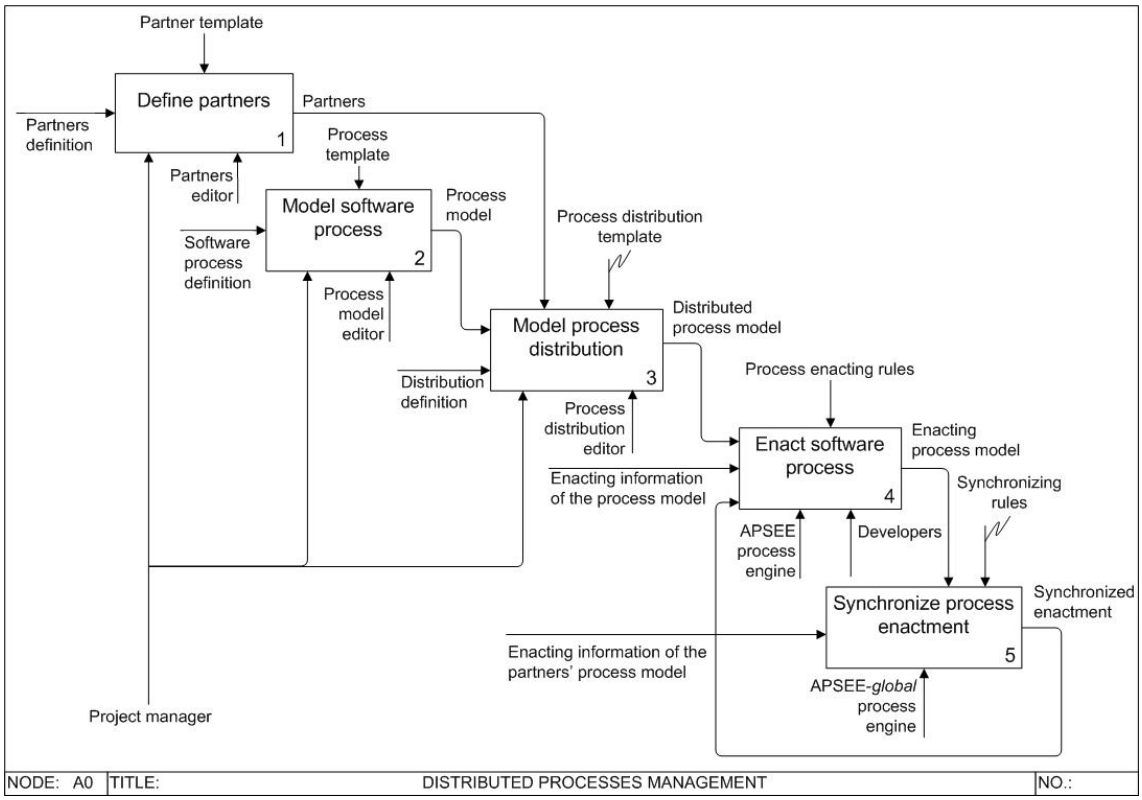

Figura 3.11: Principais funções relacionadas à gerência de processos distribuídos

3.7.2 Modelagem de processos de software

A função A0.2 corresponde à modelagem de processos de software. Essa função é suportada pelo ambiente APSEE, segundo a notação da APSEE-PML, linguagem visual de modelagem de processos integrada ao ambiente. Para auxiliar o gerente de projeto na modelagem, o APSEE provê um editor de processos. A Figura 3.12 apresenta um exemplo de fragmento de processo construído segundo a notação da APSEE-PML. Na APSEE-PML, elipses representam atividades, retângulos representam conexões de controle entre atividades e caixas tridimensionais indicam conexões de artefato.

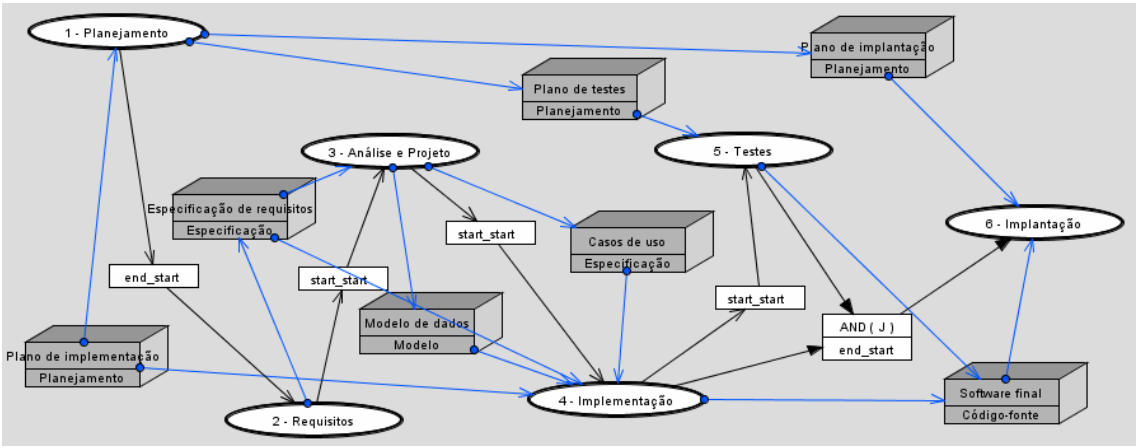

Figura 3.12: Fragmento de processo de desenvolvimento de software

3.7.3 Modelagem da distribuição de processos

A modelagem da distribuição de processos de software definidos no ambiente APSEE (representada pela função A0.3) é realizada por um conjunto de funções do APSEE-*Global*: distribuição de processos, delegação de atividades, mapeamento de processos, mapeamento de atividades e mapeamento de artefatos. O diagrama IDEF0 da Figura 3.13 detalha a função de modelagem de distribuição de processos.

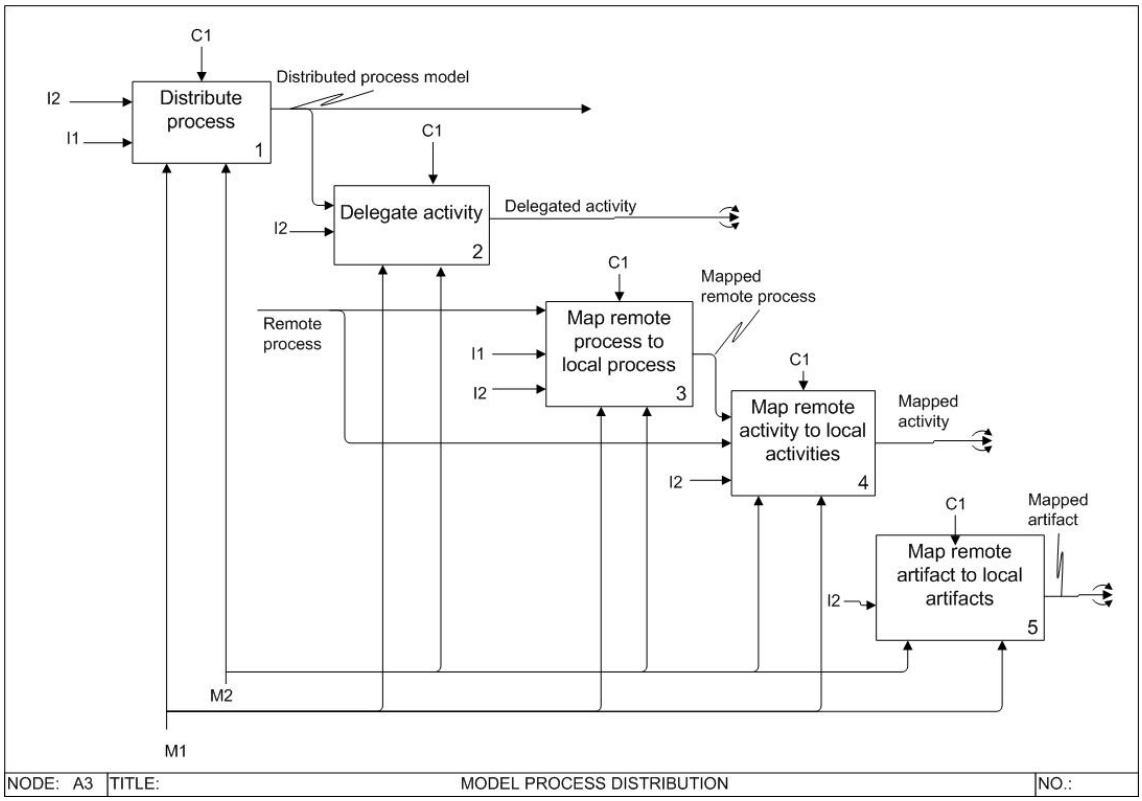

Figura 3.13: Detalhamento da função de modelagem da distribuição de processos

A distribuição de processos, definida pela função A3.1 consiste em associar a um modelo de processo um ou mais parceiros de projeto, contratados para execução de parte do processo. A instância do APSEE-*Global* do parceiro contratado é notificada da contratação e define um processo remoto correspondente ao processo distribuído.

A função A3.2 ilustra a delegação de atividades a um ou mais parceiros contratados para o projeto. Quando uma atividade é delegada, a responsabilidade pela execução passa a ser dos parceiros. Isso significa que a atividade não pode ser executada localmente, e o estado da atividade será definido de acordo com o estado da execução nos parceiros.

A delegação da execução de uma atividade simples normal corresponde à inclusão, na instância do APSEE-*Global* do parceiro contratado, de uma atividade remota que representa a atividade delegada. Uma atividade decomposta não é delegada na mesma maneira que uma atividade simples, pelo fato de que não é a execução da atividade decomposta que é delegada, e sim a execução de suas sub-atividades. Assim, delegar uma atividade decomposta tem o mesmo efeito que delegar uma a uma todas as sub-atividades que a compõem. Quando uma atividade

decomposta tem como filha outra atividade decomposta, o processo é repetido recursivamente, de modo que todas as atividades-folha da estrutura sejam delegadas ao parceiro.

Apenas atividades abstratas podem ser delegadas. Se uma atividade possui informações sobre agentes e recursos envolvidos na sua execução, então ela não poderá ser delegada.

Além de contratar parceiros para a execução de suas atividades, uma equipe também pode ser contratada para executar processos remotos. Para isso, é preciso definir qual o modelo de processo local que mapeia o processo remoto. O mapeamento de processos remotos para processos locais, definido pela função A3.3, é responsável por essa definição.

Um modelo de processo local pode mapear no máximo um processo remoto. Essa restrição se faz necessária porque um processo local poderia mapear dois processos remotos que possuem dependência entre si (por exemplo, o primeiro processo remoto mapeia atividades do segundo processo remoto). Essa dependência poderia causar um ciclo no processo distribuído.

O mapeamento de atividades remotas para atividades locais (função A3.4) define como atividades delegadas por um parceiro serão executadas pela equipe. Com o recurso do mapeamento, pode-se executar atividades remotas sem necessidade de modificar o modelo de processo local. O mapeamento consiste na associação de uma ou mais atividades locais à atividade remota. Essa associação indica que a execução da atividade remota corresponde à execução das atividades locais relacionadas.

A abordagem de mapeamento de atividades permite maior autonomia à equipe, pois ela pode aumentar o detalhamento da atividade remota, informando que a atividade corresponde a um conjunto de atividades definidas em seu modelo de processo local, ou pode manter o mesmo nível de abstração, associando a atividade remota a uma única atividade local.

Como o mapeamento interfere diretamente no estado de execução da atividade delegada no modelo de processo no qual ela foi definida, algumas regras devem ser observadas em relação ao mapeamento de atividades remotas, como as exemplificadas a seguir:

- O mapeamento entre atividades pode ser desfeito, desde que a execução da atividade local ainda não tenha sido finalizada;
- Uma atividade local que já tenha iniciado a sua execução não pode mapear atividades remotas com dependências que impeçam o início da sua execução;
- Se duas atividades remotas possuem dependências entre si, elas não podem ser mapeadas para atividades locais que possuam fluxo de controle no sentido oposto. Essa condição previne a inserção de ciclos no processo distribuído.

A função A3.5 define o mapeamento de artefatos remotos para artefatos locais. Esse mapeamento consiste em associar aos artefatos de saída de atividades remotas a artefatos produzidos pelas atividades para as quais ela é mapeada.

3.7.4 Execução de modelos de processo

Durante a execução de modelos de processo, definida pela função A0.4, as atividades são realizadas tanto pelos desenvolvedores quanto automaticamente. O mecanismo de execução de processos do APSEE interpreta o modelo de processo e gerencia as informações do ambiente e desenvolvedores de acordo com esse modelo.

3.7.5 Sincronização da execução dos processos

A sincronização da execução dos processos permite que as equipes contratantes tenham conhecimento do estado de execução das atividades delegadas. Essa sincronização também realiza a propagação dos efeitos de falhas e cancelamentos.

3.7.6 Alterações nas funcionalidades do APSEE

Para que as funcionalidades do APSEE-*Global* não insiram inconsistências nos modelos de processo, a semântica de algumas funcionalidades do APSEE foi modificada. A seguir são descritas as modificações que se fizeram necessárias:

- A exclusão de atividades delegadas deve ser propagada para os parceiros a ela associados;
- Se duas atividades locais mapeiam atividades remotas que possuem algum tipo de dependência entre si, então nenhum fluxo de controle deve ser inserido entre as atividades locais;
- Atividades só podem instanciar cargos, agentes, grupos e recursos se não estiverem delegadas;
- Atividades automáticas só podem ser executadas quando não estão mapeando atividades remotas que se encontram nos estados *null* ou *waiting*;
- Atividades normais só podem ser definidas como prontas para execução (estado *ready*) quando não estão mapeando atividades remotas que se encontram nos estados *null* ou *waiting*.

3.8 Considerações

Este capítulo apresentou informalmente o modelo APSEE-*Global*, para gerência de projetos distribuídos de desenvolvimento de software. O modelo de gerência adotado permite que o mecanismo APSEE-*Global* guie as equipes durante o decorrer do projeto. Cada equipe trabalha seguindo a sua própria metodologia, tomando conhecimento das alterações relevantes ocorridas nos demais processos de software. O próximo capítulo formaliza os componentes e as funcionalidades descritas neste capítulo.

4 FORMALIZAÇÃO DO MODELO APSEE-*GLOBAL*

> Métodos formais, usados no desenvolvimento de sistemas de computador, são técnicas baseadas em matemática para descrição de propriedades de um sistema. Tais métodos fornecem arcabouços, dentro dos quais as pessoas podem especificar, desenvolver e verificar sistemas de maneira sistemática ao invés da usual.
>
> O método é formal se tem uma sólida base matemática, dada tipicamente por uma linguagem formal de especificação. Essa base fornece um meio de definir precisamente noções como consistência e completude, e mais relevantemente, especificação, implementação e correção. (MARCINIAK apud PRESSMAN, 2002).

Uma especificação de sistema que utiliza métodos formais é inerentemente menos ambígua que uma especificação expressa em linguagem natural, devido à notação matemática utilizada. Desta forma, os métodos formais reduzem consideravelmente os erros de especificação de sistemas. Métodos formais também facilitam o processo de validação, pois se pode provar matematicamente que um programa desenvolvido atende a uma dada especificação.

As vantagens advindas da especificação formal de sistemas motivaram a adoção de métodos formais para definição do modelo APSEE-*Global*. Foram utilizados na especificação do modelo, de maneira complementar, os métodos formais Prosoft-Algébrico e Gramática de Grafos. O Prosoft-Algébrico foi utilizado para especificar os tipos de dados do modelo, e as funcionalidades foram especificadas utilizando Gramática de Grafos.

Este capítulo contém a especificação formal do APSEE-*Global* e está organizado como segue. A seção 4.1 apresenta a especificação dos componentes do APSEE-*Global*, segundo o formalismo Prosoft-Algébrico. A seção 4.2 introduz a especificação das funcionalidades do modelo, utilizando Gramática de Grafos. Por fim, a seção 4.3 traz algumas considerações sobre o capítulo.

4.1 Especificação dos componentes

A adoção do Prosoft-Algébrico para especificação da estrutura dos componentes do APSEE-*Global* proporcionou uma derivação direta para implementação no ambiente Prosoft-Java[13], visto que há uma correspondência semântica entre os elementos usados na especificação e a implementação dos

[13] Prosoft-Java é a denominação do ambiente escolhido para prototipação do modelo proposto.

componentes de software descritos nesse paradigma. Além disso, o grupo de pesquisa no qual o trabalho foi desenvolvido possui uma vasta experiência no uso desse formalismo.

Esta seção descreve o formalismo Prosoft-Algébrico, e apresenta a especificação da estrutura dos componentes do APSEE-*Global*.

4.1.1 Prosoft-Algébrico

O Prosoft-Algébrico (NUNES, 1992) (NUNES, 1994) é um formalismo que permite a descrição de tipos abstratos de dados através de um paradigma algébrico baseado em objetos. Esse paradigma adota uma abordagem *data-driven* (NUNES, 1994) para o desenvolvimento de software, isto é, estimula o desenvolvimento de software inicialmente através da composição dos tipos de dados necessários.

O Prosoft-Algébrico é, portanto, uma técnica orientada a propriedades[14] que define um objeto matemático baseado nas relações entre as operações desse objeto (COHEN; HARWOOD; JACKSON, 1986). Cada tipo de dado é definido a partir de um ATO – Ambiente de Tratamento de Objetos. Cada ATO especifica algebricamente um tipo abstrato de dado. Qualquer termo desse tipo é chamado de objeto e, segundo Nunes (1994), não é similar com o conceito de objeto das linguagens de programação orientadas a objetos. Os objetos algébricos são entidades passivas, que não têm capacidade de responder a estímulos (mensagens) (DAUDT, 1992). Portanto, os termos Prosoft armazenam dados mas não podem manipulá-los: toda manipulação de dados é descrita através de funções definidas no escopo de um ATO.

A Figura 4.1 apresenta o esquema gráfico de um software desenvolvido sob o paradigma Prosoft, sendo composto por um conjunto de ATOs que comunicam-se através da ICS (Interface de Comunicação do Sistema). Cada ATO é dividido em cinco partes, como mostrado à direita da Figura 4.1. A primeira parte tem como objetivo definir uma instanciação do tipo através de uma linguagem gráfica (**classe**[15]). A segunda parte (***include***) especifica as importações de tipos de dados primitivos. A terceira parte (**interface**) especifica a funcionalidade (assinatura) das novas operações. A quarta parte define as **variáveis formais** utilizadas nas operações. A quinta parte define a semântica das novas operações (**axiomas**) que atuam sobre o objeto.

A parte esquerda da Figura 4.1 mostra a Interface de Comunicação do Sistema (ICS), que tem a função de integrar os diversos ATOs. Qualquer termo só pode ser

---

[14] Os formalismos orientados a propriedades definem o comportamento do sistema por meio de propriedades a serem satisfeitas. São duas as abordagens de especificação orientada a propriedades: especificação algébrica, onde os tipos abstratos de dados são definidos em termos de álgebras heterogêneas; e especificação axiomática, que inclui o uso de lógicas (de primeira ordem, de predicados, etc).

[15] Vale ressaltar que o conceito de classe empregado no Prosoft-Algébrico difere do conceito de classes em linguagens de programação orientadas a objetos. Uma classe Prosoft é uma estrutura de dados que corresponde ao que normalmente é chamado de atributos nas linguagens orientadas a objetos.

criado ou modificado pelo ATO que contém sua classe. Assim, quando um ATO necessita processar termos que não pertencem à sua classe, ele deve requisitar operações, via ICS, dos ATOs que definem estes termos. A ICS é também uma operação algébrica, e sua sintaxe é:

ICS (<ATO>,<operação>,<seletor>,<argumentos*>)

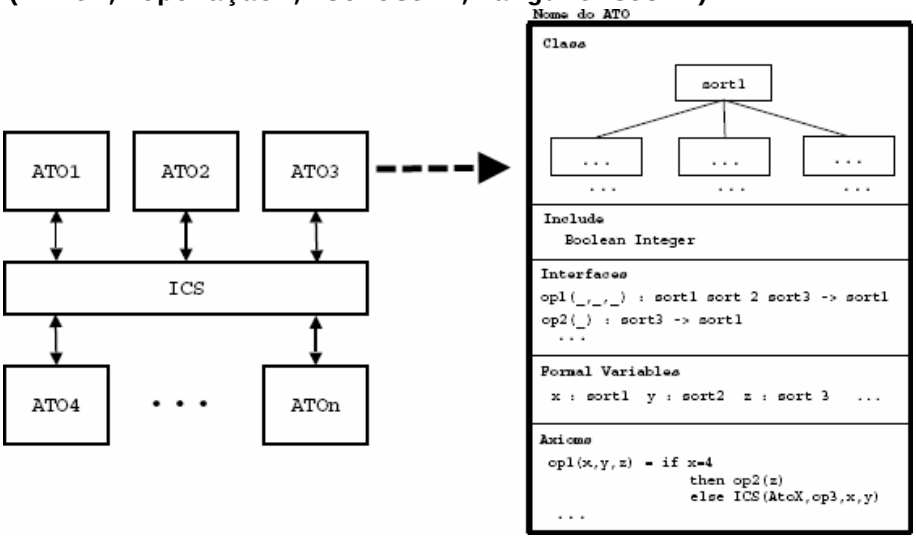

Figura 4.1: Composição de ATOs no Prosoft-Algébrico

A pesquisa da operação é feita de cima para baixo dentro do ATO: se o nome do ATO coincidir e o nome da operação for uma operação pertencente ao ATO, os argumentos serão ligados aos parâmetros formais e a operação será executada. Como uma operação pode possuir várias definições, um dos argumentos do tipo do *sort* definido pelo ATO – denominado seletor – é usado para encontrar a operação.

Os tipos de dados Prosoft são classificados como primitivos (*Integer*, *String*, *Real*, *Date* e *Boolean*), compostos (Conjunto, Lista, Mapeamento, Registro e União Disjunta) ou definidos pelo usuário. Para facilitar a utilização do método Prosoft-algébrico, foi definida uma representação gráfica, inspirada nos diagramas de Jackson (JACKSON, 1983), para definição das classes. Cada tipo composto possui uma representação gráfica na forma de árvore. Na instanciação, a raiz da árvore descreve o *sort* definido pelo ATO, os nodos são tipos de dados compostos e as folhas são referências a outros tipos de dados. Os tipos compostos e suas representações gráficas são mostrados na Figura 4.2, através de um exemplo.

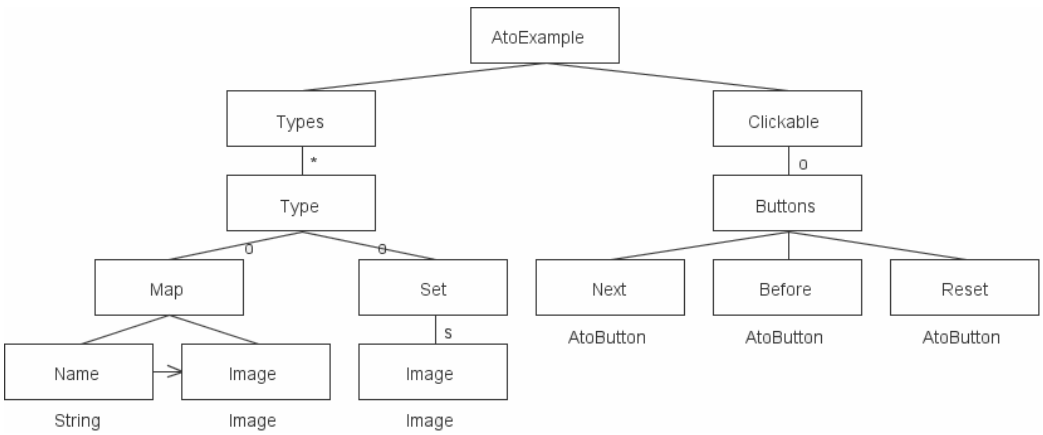

Figura 4.2: Representação gráfica dos tipos compostos do Prosoft-Algébrico

O TAD definido na raiz da árvore (*Example*) representa um registro com dois campos: *Types* e *Clickable*. *Types* é uma lista de elementos com *sort Type*. Cada elemento *Type* corresponde à União Disjunta entre *Map* e *Set*, onde *Map* é um mapeamento que tem *Name* como domínio e *Image* como contradomínio, e *Set* é um conjunto de elementos com *sort Image*. *Name* é uma instância do TAD primitivo *String*, e *Image* é um TAD definido em outro ATO. O outro elemento de *Example* é *Clickable*, uma União Disjunta composta por um único elemento, *Buttons*. *Buttons*, por sua vez, é um registro com três campos, todos pertencentes ao TAD *Button*, definido em outro ATO.

A instanciação de uma classe importa automaticamente todas as operações pré-definidas dos tipos compostos e primitivos que compõem a classe. Por exemplo, se a classe é uma lista de *strings*, todas as operações de criação e manipulação de listas e de *strings* são automaticamente incluídas na definição do ATO.

Para a definição de novas operações no ATO, se utiliza a interface para declarar a assinatura de cada operação. A assinatura é composta por uma *string* que representa o nome da operação, seguida de seu respectivo *rank*[16], como ilustrado no exemplo abaixo:

```
include-type : Example Type -> Example
```

Para cada nova operação definida num ATO, deve-se criar pelo menos um axioma para lhe dar semântica. Um axioma geralmente usa variáveis formais para generalizar sua definição. Abaixo são definidas as variáveis *types* e *type*, com seus respectivos *sorts*.

```
types : Types
type : Type
```

A semântica da operação *include-type* é dada pelo axioma:

```
include-type(reg-Example(types,_), type)=
   reg-Example(cons(type,types),_)
```

Nos métodos algébricos convencionais, variáveis são utilizadas nos axiomas. O Prosoft-Algébrico conta com o artifício "_", que facilita a construção de axiomas e conseqüentemente aumenta a legibilidade dos mesmos. Utiliza-se o símbolo "_" no lugar de variáveis que representam termos do lado esquerdo de um axioma que não são modificados no lado direito. Esse recurso foi utilizado no exemplo de axioma apresentado, para omissão da variável de *sort Clickable*.

4.1.2 Tipos de dados do APSEE-*Global*

Os tipos de dados que compõem o APSEE-*Global* foram especificados em Prosoft-Algébrico. Como as operações do modelo foram definidas em Gramática de Grafos, a especificação em Prosoft-Algébrico limitou-se à definição das classes que compõem cada tipo de dado. A Figura 4.3 ilustra os ATOs correspondentes às principais classes definidas, e a comunicação entre eles via ICS.

---

[16] O *rank* de uma operação é composto pela aridade da operação seguido do *sort* do resultado da mesma.

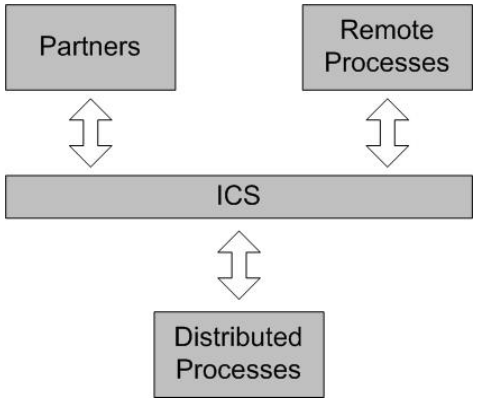

Figura 4.3: Tipos Abstratos de Dados do APSEE-*Global*

O ATO *Partners* define os parceiros de projeto, que cooperam com a equipe na execução de processos, por meio de delegação de atividades e pela execução de atividades delegadas; O ATO *DistributedProcesses* define os processos distribuídos, enquanto o ATO *RemoteProcesses* define os processos de software que foram delegados por parceiros.

A seguir são apresentadas as classes que definem cada um dos tipos de dados definidos no APSEE-*Global*. Como mencionado no capítulo 3, o APSEE-*Global* faz referência aos componentes do APSEE que definem a hierarquia de tipos, os processos de software e os artefatos. Assim, os tipos de dados apresentados nessa seção fazem referência às classes do APSEE *Processes*, *Artifacts* e *Types*. Essas classes serão apresentadas no Anexo.

- ***Partners***

Esse tipo de dado define os parceiros de projeto de uma instância do APSEE-*Global*. Os atributos *APSEEId* e *Host* identificam unicamente um parceiro, e permitem a comunicação entre diferentes instâncias do APSEE-*Global*. O tipo de parceiro é definido segundo a hierarquia de tipos do APSEE (*APSEETypes*), onde foi adicionado o tipo *PartnerType*. Cada objeto que representa um parceiro referencia o conjunto de processos remotos delegados por ele à equipe.

A Figura 4.4 apresenta a classe que define o tipo de dados *Partners*. Essa classe é um mapeamento que relaciona o identificador do parceiro a um registro contendo os demais atributos. Os processos remotos são referenciados por um conjunto de identificadores de processos remotos.

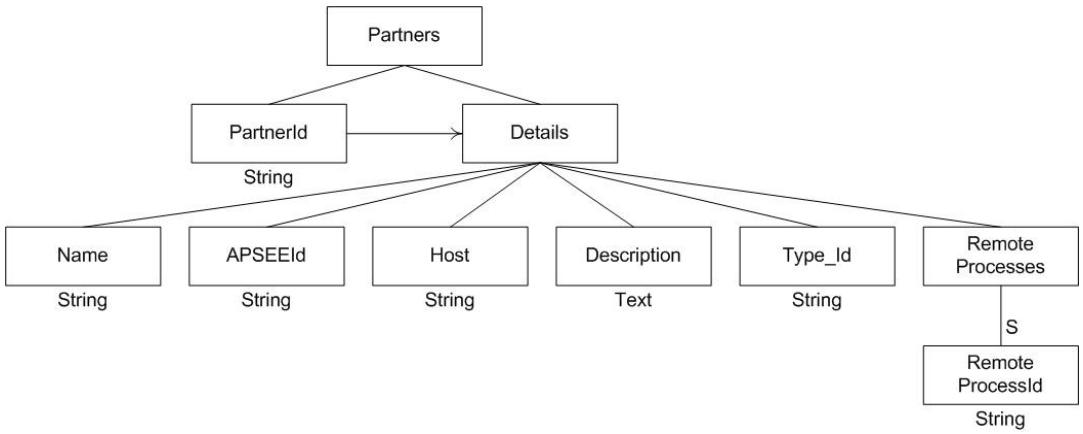

Figura 4.4: Classe *Partners*

- ***DistributedProcesses***

A Figura 4.5 ilustra a classe *DistributedProcesses*, que define os processos distribuídos. Os processos de software distribuídos são definidos segundo a APSEE-PML, e posteriormente são delegados a parceiros. O atributo *ProcessId* é o identificador do processo definido no APSEE: esse atributo corresponde ao atributo *ProcessId* da classe *Process*. A classe *DistributedProcesses* é um mapeamento que relaciona cada identificador de processo APSEE a um registro que contém os dois tipos de delegação que são realizados: delegação do processo e delegação das atividades do processo. A delegação do processo é representada pelo conjunto *ProcessDelegation*, que armazena o identificador de todos os parceiros para os quais o processo foi delegado.

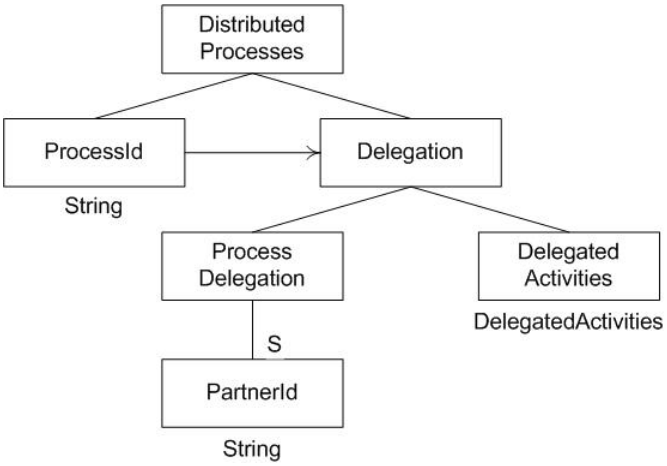

Figura 4.5: Classe *DistributedProcesses*

A delegação de atividades é apresentada na Figura 4.6, pela classe *DelegatedActivities*. Essa classe mapeia, para cada atividade, definida no processo APSEE, as delegações realizadas. Uma atividade delegada pode ser decomposta ou normal, como expressa a união disjunta *ActivityDelegation*. A delegação de atividades decompostas define o conjunto de parceiros a ela associados.

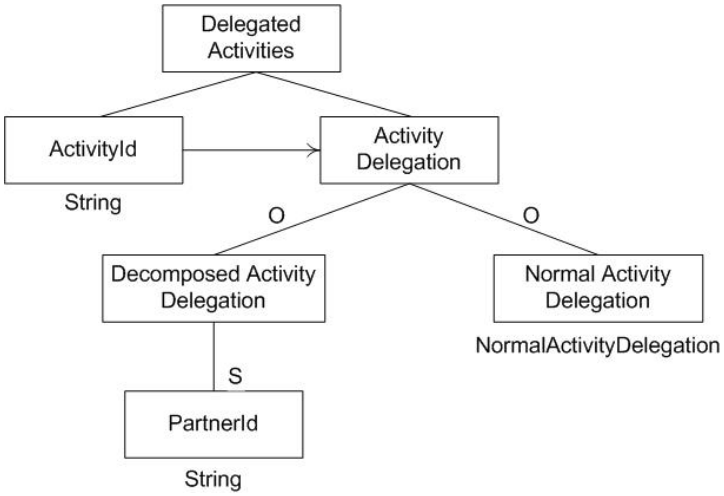

Figura 4.6: Classe *DelegatedActivities*

A delegação de atividades normais é definida pela classe *NormalActivityDelegation*, que, para cada delegação de atividade normal a um

parceiro atribui o estado da atividade, que representa o estado da execução da atividade pelo parceiro e os artefatos de entrada e de saída.

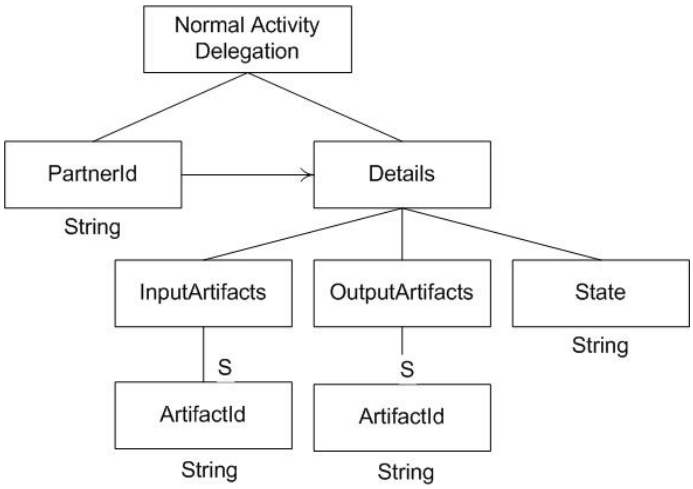

Figura 4.7: Classe *NormalActivityDelegation*

- ***RemoteProcesses***

Um processo remoto (Figura 4.8) é composto por um identificador (que é o identificador do processo delegado), pelas atividades remotas, que são as atividades do processo remoto delegadas à equipe e pelo identificador do processo local que o instancia.

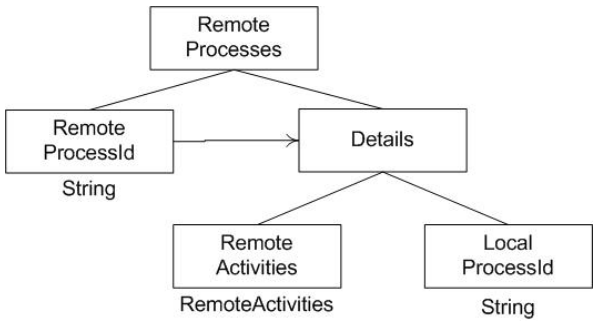

Figura 4.8: Classe *RemoteProcesses*

Atividades remotas, definidas pela classe *RemoteActivities* (Figura 4.9), são compostas pelo identificador da atividade delegada e pelos atributos de nome, descrição e estado. Como apenas atividades normais originam atividades remotas, o estado de uma atividade remota é definido como sendo um dos estados possíveis para atividades normais: *waiting*, *ready*, *active*, *paused*, *finished*, *canceled* e *failed*. Uma atividade remota está associada a um conjunto de atividades locais que a representam.

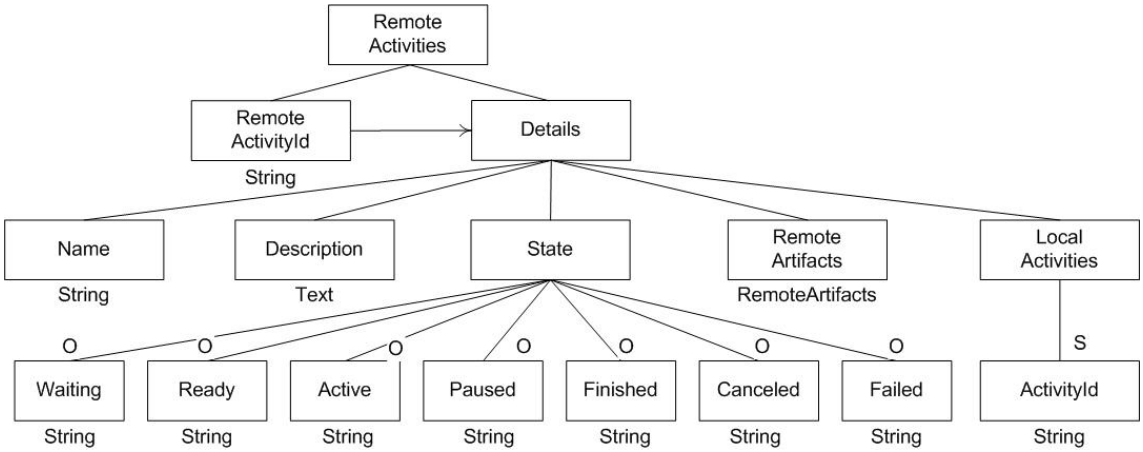

Figura 4.9: Classe *RemoteActivities*

Uma atividade pode consumir e produzir artefatos. A classe *RemoteArtifacts*, ilustrada na Figura 4.10, representa os artefatos de entrada (consumidos) e de saída (produzidos) pela atividade remota. Artefatos remotos de saída são mapeados para artefatos locais.

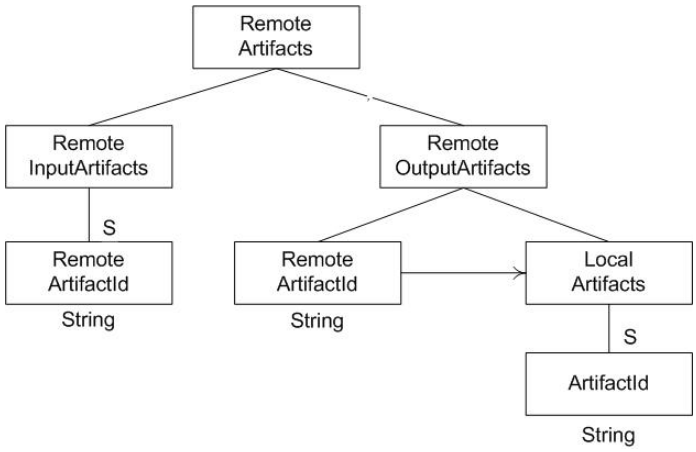

Figura 4.10: Classe *RemoteArtifacts*

4.2 Especificação das funcionalidades

A escolha de se usar Gramática de Grafos para especificar as funcionalidades do modelo proposto foi influenciada pelo fato de Gramáticas de Grafos serem formais e intuitivas ao mesmo tempo, e de poderem tratar com simplicidade aspectos de concorrência e distribuição de sistemas.

Essa seção introduz o formalismo de Gramática de Grafos, e apresenta as regras que definem as funcionalidades do APSEE-*Global*.

4.2.1 Gramática de Grafos

Gramáticas de Grafos baseiam-se no processo de transformação que um grafo pode sofrer em função de um conjunto finito de regras de derivação. Um sistema é especificado em termos de estados, que são modelados por grafos, e mudanças de estados, modeladas por regras de derivação. A linguagem da Gramática de Grafos é o conjunto de todos os grafos que podem ser derivados do grafo original em uma

seqüência finita de aplicações das regras de derivação. Cada regra consiste de um grafo esquerdo (E) e um grafo direito (D), e descreve como uma ocorrência do grafo esquerdo pode ser substituída por uma ocorrência do grafo direito em um grafo G. A aplicação de uma regra ao grafo G é chamada passo de derivação, e só é possível se existe uma ocorrência do lado esquerdo (E) da regra em G. O lado direito (D) da regra define o grafo resultante da aplicação desta regra. A interpretação de uma regra r : E → D é feita da seguinte forma:

- Itens em E que não tem imagem em D são eliminados;
- Itens em E que são mapeados para D são preservados;
- Itens em D que não tem uma pré-imagem em E são criados.

O grafo resultante é dito derivado do grafo esquerdo por um passo de derivação. Para garantir a consistência do grafo derivado, após cada passo de derivação é necessário excluir os arcos pendentes: arcos que estavam conectados a vértices que tenham sido excluídos durante a derivação.

Uma Gramática de Grafos consiste de um conjunto finito de rótulos para nodos e arcos, um axioma (isto é, um grafo inicial) e um conjunto finito de regras de derivação. Cada nodo ou arco pode ser associado a um elemento do conjunto de rótulos. Neste trabalho será apresentado um grafo-tipo representando o conjunto de rótulos (tipos) de nodos e arcos do sistema. Os nodos do grafo-tipo correspondem aos tipos de dados definidos em Prosoft-Algébrico. O grafo inicial e os grafos derivados das transformações devem ser compatíveis com o grafo-tipo. O conjunto de regras de derivação do grafo inicial que determinam a semântica das funcionalidades do APSEE-*Global* também será apresentado.

4.2.2 Grafo-tipo

O grafo-tipo para distribuição de processos, apresentado na Figura 4.11, é uma extensão do grafo-tipo do APSEE. Cada nodo está associado a um nome e um conjunto de atributos (separados por uma linha do nome do nodo). Atributos de dados são elementos tipados usados para representar e raciocinar sobre o estado do processo. As setas do diagrama representam os arcos do grafo, significando relacionamentos (setas pontilhadas) e chamadas de operações (setas sólidas) entre os nodos. Os símbolos representados com fundo branco estão presentes também no grafo-tipo do APSEE. Os símbolos com fundo cinza foram inseridos exclusivamente para tratar a distribuição de processos.

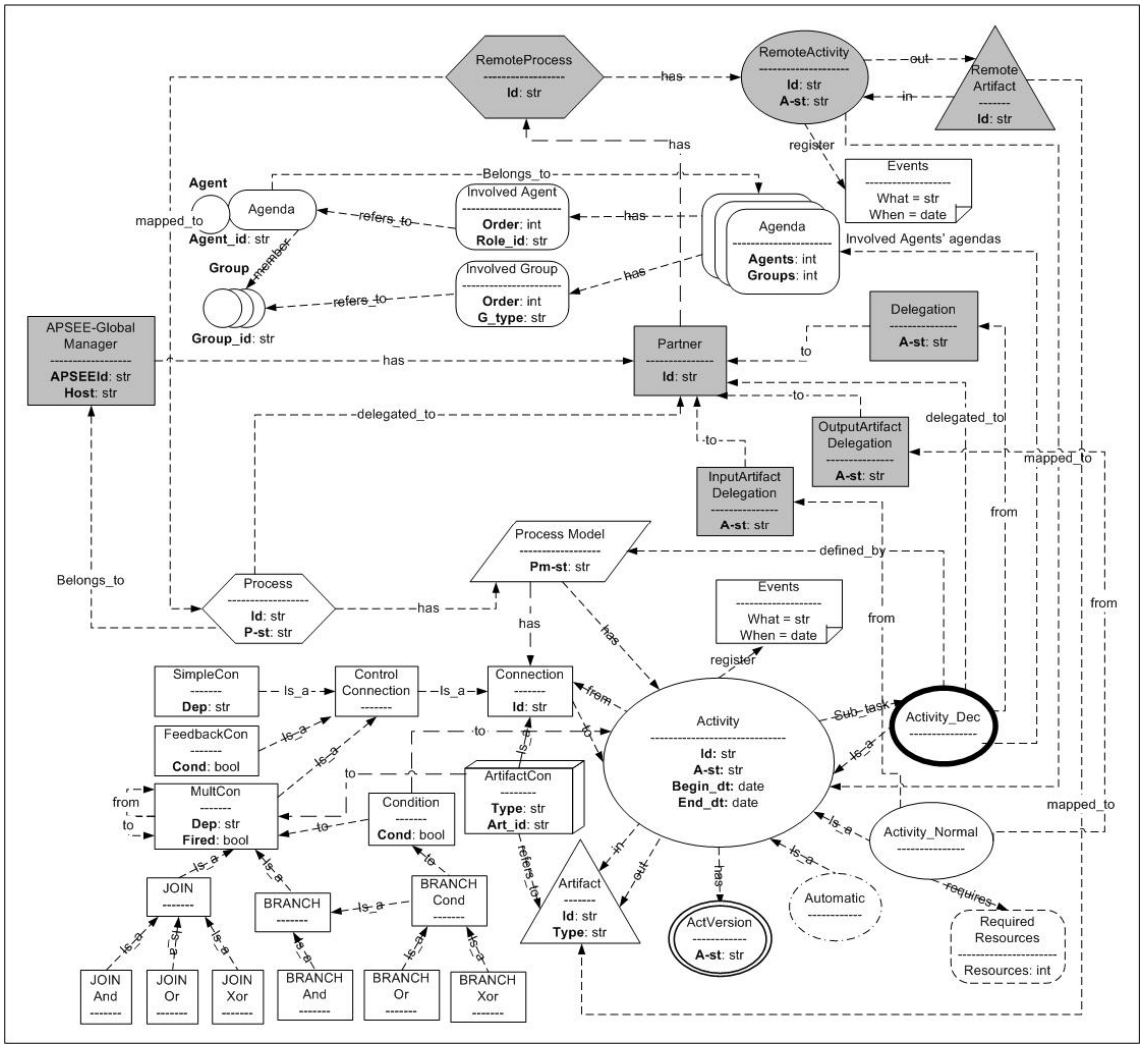

Figura 4.11: Grafo-tipo do modelo *APSEE-Global*

A parte principal do grafo-tipo é ilustrada na Figura 4.12. Essa parte indica que uma instância do APSEE-*Global* possui processos de software. Cada processo de software possui um modelo de processo composto por atividades e conexões. Os atributos *APSEEId* e *host* identificam unicamente uma instância do APSEE-*Global*, e são parâmetros utilizados na comunicação entre diferentes instâncias.

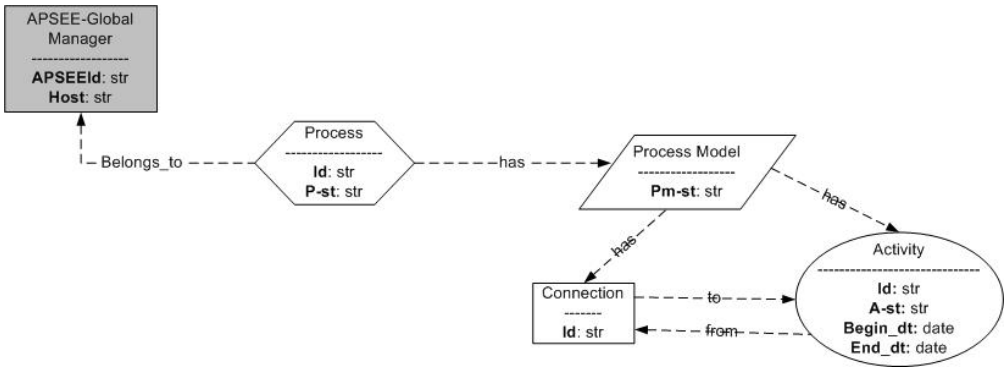

Figura 4.12: Parte do grafo-tipo relacionada à estrutura de processos de software

O atributo *P-st* corresponde ao estado da execução do processo e pode assumir os seguintes valores: *not-started* (não iniciado), *enacting* (executando) ou *finished*

(concluído). O atributo *Pm-st* (do componente *Process Model*) está associado ao estado do processo no ciclo de vida de processos: *Null* (durante a fase de modelagem), *Requirements* (quando existe apenas uma descrição textual para o processo), *Abstract* (o processo não está instanciado), *Instantiated* (o processo está pronto para execução porque possui componentes instanciados), *Enacting (o processo está sendo executado)*, *Finished (a execução foi finalizada)* ou *Mixed* (quando o processo é composto por fragmentos em diferentes estados). Os estados possíveis para atividades (atributo *A-st*) são: *Null* (na fase de modelagem), *Waiting* (quando as dependências não estão satisfeitas), *Ready* (pronta para executar), *Active* (em execução), *Paused (a execução foi suspensa)*, *Finished (a atividade foi finalizada)*, *Canceled* (a atividade foi cancelada antes de iniciar) e *Failed* (a atividade falhou após início da execução). As setas de relacionamento entre o tipo *Activity* e *Connection* indicam que uma conexão pode ter origem e destino em instâncias do tipo atividade.

A Figura 4.13 ilustra a parte relacionada a conexões do grafo-tipo. O relacionamento *is_a* pode ser compreendido como herança, similar ao uso de herança em linguagens de programação orientadas a objetos. Como indica esse tipo de relacionamento, conexões podem ser de controle (*ControlConnection*) ou de artefato (*ArtifactCon*). Conexões de controle, por sua vez, podem ser: conexões simples (*SimpleCon*), conexões de *feedback* (*FeedbackCon*) e conexões múltiplas (*MultCon*). As conexões múltiplas subdividem-se em *join* e *branch*, cada uma delas com definição de tipo (AND, OR ou XOR).

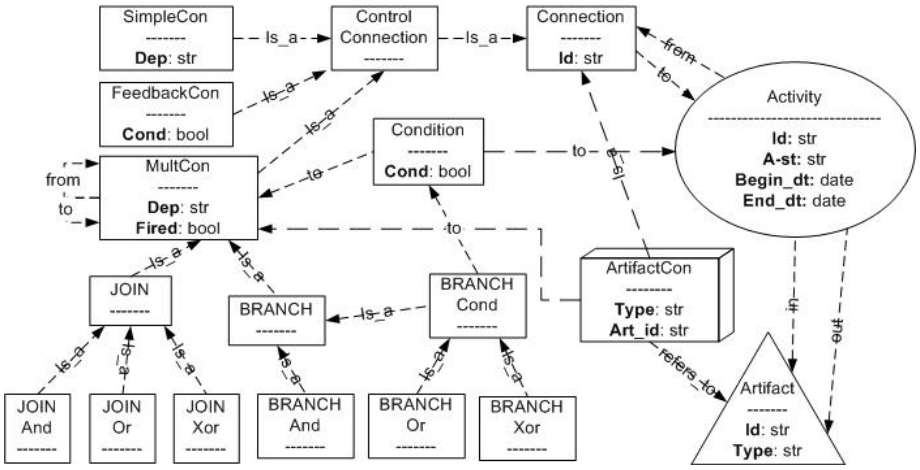

Figura 4.13: Parte do grafo-tipo relacionada a conexões

O relacionamento *is_a* simplifica bastante o grafo-tipo e as regras de transformação correspondentes, já que atributos e relacionamentos são herdados aos subtipos. Por exemplo, quando o lado esquerdo de uma regra contiver uma referência para o tipo principal *ControlConnection*, então essa referência pode ser substituída por quaisquer dos subtipos associados a *ControlConnection* (*SimpleCon*, *FeedbackCon* e *MultCon*).

Os arcos *from* e *to* descrevem a origem e o destino da conexão. O relacionamento *to* entre conexão e atividade é redefinido para conexões múltiplas condicionais do tipo *branch* (*branch* OR e *branch* XOR), já que essas conexões

requerem uma condição lógica adicional para cada destino. Outros atributos importantes são: o tipo de dependência para conexões simples e múltiplas (*end-start*, *start-start*, *end-end*) e a condição lógica para conexões de *feedback* e *BranchCond*.

A parte referente à definição de atividades no grafo-tipo é apresentada na Figura 4.14. O relacionamento *is_a* é usado para herdar atributos e relacionamentos do vértice *Activity*. No grafo-tipo, os símbolos que representam os tipos de atividades são diferenciados para facilitar o desenvolvimento das regras de gramática de grafos. A atividade decomposta (*Activity_Dec*) é definida por um modelo de processo que por sua vez possui conexões e atividades; esse relacionamento é refinado transitivamente pelo relacionamento *Sub_task*. O nodo *ActVersion*, por sua vez, representa as versões de atividades criadas pelo sistema quando a atividade tem que ser re-executada ou quando o usuário solicita a criação da versão.

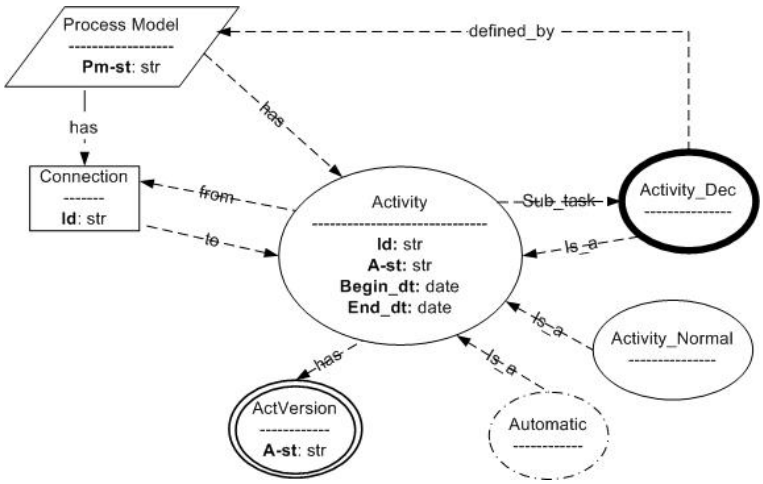

Figura 4.14: Parte do grafo-tipo relacionada a atividades

Somente as atividades normais (*Activity_Normal*) possuem uma ligação com agentes e recursos, como mostra a Figura 4.15. Cada atividade possui uma ligação com o nodo *Agenda*. Este, por sua vez, possui atributos que identificam a quantidade de pessoas e grupos trabalhando na atividade. A agenda referencia agentes e grupos. Através desses relacionamentos é possível identificar os agentes que trabalham para uma atividade.

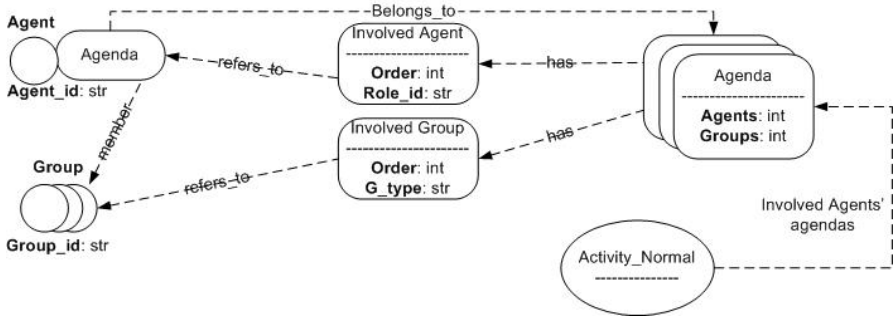

Figura 4.15: Parte do grafo-tipo relacionada a envolvimento com agentes

A parte da linguagem de modelagem referente à ligação de atividades normais com recursos foi omitida no grafo-tipo da Figura 4.11, pelo fato de que o APSEE-*Global* não trata a alocação de recursos. Apenas atividades normais requerem recursos, que podem ser exclusivos, compartilháveis ou consumíveis.

A delegação de processos a parceiros é representada na Figura 4.16. Processos podem ser delegados a parceiros, e o relacionamento entre esses dois tipos de nodos demonstra essa situação. Atividades normais e decompostas que pertençam a processos delegados podem também ser delegadas a parceiros. O relacionamento entre atividades normais delegadas e parceiros é estabelecido por meio do nodo *Delegation*. Esse nodo indica o estado de execução da atividade em cada um dos parceiros, e serve de base para a determinação do estado geral de execução da atividade. A relação entre atividades decompostas e parceiros é estabelecida pelo relacionamento *delegated_to*. Os artefatos de entrada e saída das atividades normais delegadas também são delegados aos parceiros.

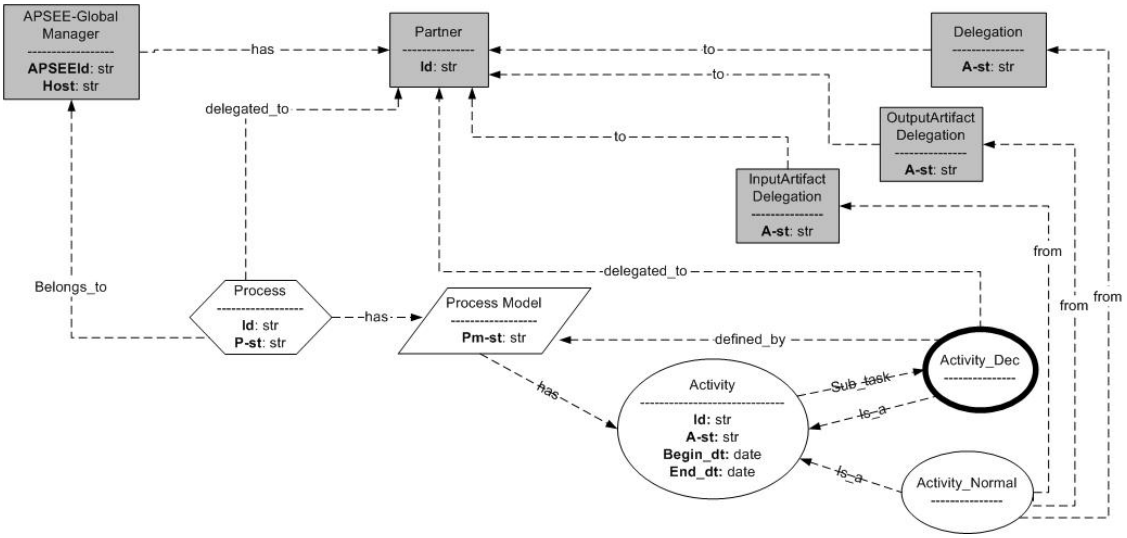

Figura 4.16: Parte do grafo-tipo relacionada a delegação de processos

Já o mapeamento de processos e atividades remotos é definido na Figura 4.17. Um parceiro pode estar associado a processos remotos, que são os processos delegados por ele à equipe representada na instância do APSEE-*Global*. Um processo remoto é mapeado para um processo local, e possui um conjunto de atividades remotas. O mapeamento entre atividades remotas e atividades locais é representado pelo relacionamento entre a atividade remota e atividades locais. O mapeamento dos artefatos remotos é representado pelo relacionamento *mapped_to*, entre artefatos remotos e locais.

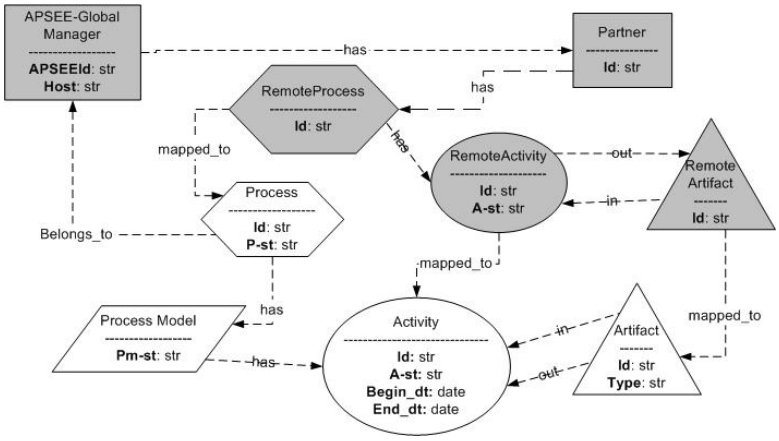

Figura 4.17: Parte do grafo-tipo relacionada a mapeamento de processos

4.2.3 Regras para definição das funcionalidades do APSEE-*Global*

Para definir as funcionalidades do APSEE-*Global*, foram descritas regras de transformação para especificar a semântica das operações que podem ser aplicadas aos tipos de dados do modelo. Em muitos casos, foram usadas condições negativas de aplicação (NAC – *Negative Application Conditions*) para descrever condições que devem ser negativas para que a regra seja habilitada. Neste caso as partes cortadas no lado esquerdo da regra formam as condições negativas e não devem estar presentes na instância do grafo para que a regra seja aplicada.

A Figura 4.18 apresenta um exemplo de regra, com o objetivo de ilustrar os elementos usados na sua descrição. No exemplo, o nome da operação (*DelegateProcess*), assim como seus parâmetros (*process_id*, *partner_id*), são incluídos acima do lado esquerdo da regra. O lado esquerdo da regra descreve a condição esperada para executar a operação solicitada: no caso, a instância do APSEE-*Global* que recebe a solicitação deve possuir um parceiro com identificador *partner_id*. A NAC contida na regra indica que o processo com identificador *process_id* não pode estar delegado ao parceiro *partner_id*. O lado direito da regra apresenta o resultado obtido após a realização do passo de derivação: o processo é delegado e uma operação da instância do APSEE-*Global* do parceiro é chamada.

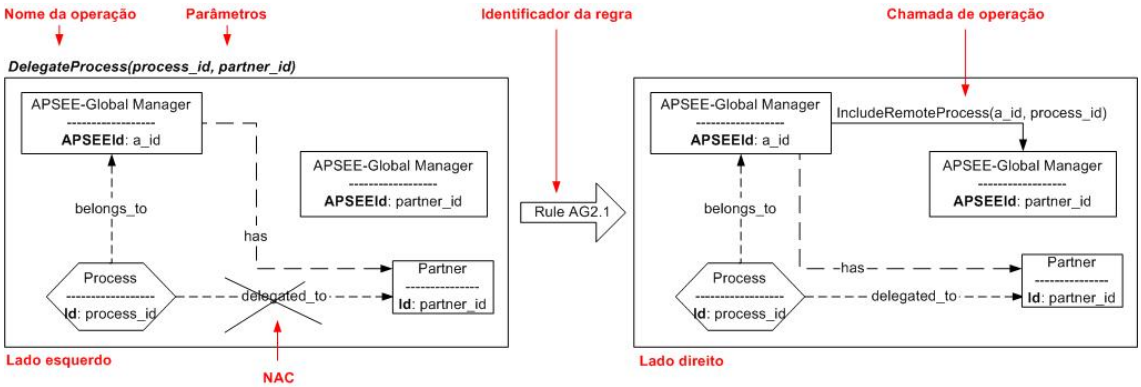

Figura 4.18: Exemplo de regra em Gramática de Grafos

Outra maneira de definir NACs é apresentar, em separado, as condições que não podem estar presentes na instância do grafo. A Figura 4.19 ilustra a mesma regra da Figura 4.18, definida com NAC em separado.

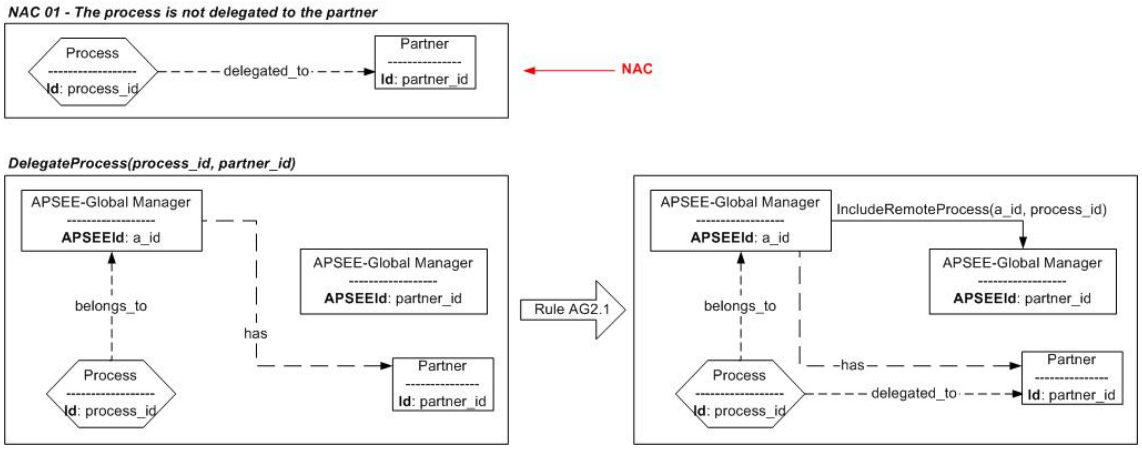

Figura 4.19: Exemplo de regra em Gramática de Grafos com NAC em separado

As subseções a seguir apresentam as regras definidas, agrupadas de acordo com a funcionalidade do APSEE-*Global* que representam.

4.2.3.1. Definição de parceiros

A definição de parceiros é definida por duas operações:

- `NewPartner(partner_id)`, que inclui um novo parceiro em uma instância do APSEE-*Global*;
- `RemovePartner(partner_id)`, que remove um parceiro.

A figura Figura 4.20 apresenta a regra que define a operação de inclusão de um novo parceiro. Segundo essa regra, um parceiro pode ser incluído se ainda não estiver associado à instância do APSEE-*Global*.

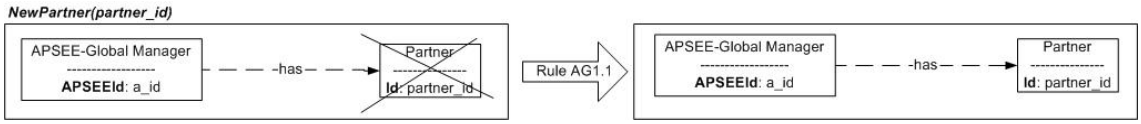

Figura 4.20: Regra para inclusão de parceiro

A operação de exclusão de parceiros é definida pela regra ilustrada na Figura 4.21. As condições para exclusão são que o parceiro não possua nenhum processo remoto associado e que nenhum processo local esteja delegado ao parceiro.

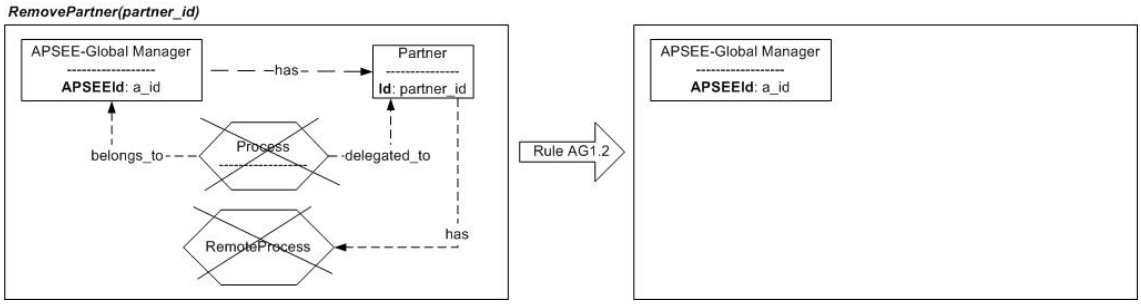

Figura 4.21: Regra para exclusão de parceiro

4.2.3.2. Modelagem da distribuição de processos

A modelagem da distribuição de processo engloba cinco funcionalidades: distribuição de processos, delegação de atividades, mapeamento de processos, mapeamento de atividades e mapeamento de artefatos. A seguir serão apresentadas as operações que definem cada funcionalidade, e as regras que especificam a semântica de cada uma das operações.

- **Distribuição de processos**

A distribuição de processos envolve as seguintes operações:

- *DelegateProcess(process_id,partner_id)*, que associa o parceiro *partner_id* ao processo *process_id*;
- *IncludeRemoteProcess(a_id,process_id)*, que define o processo remoto que representa um processo delegado;
- *RemoveProcessDelegation(process_id,partner_id)*, que remove o parceiro *partner_id* do conjunto de parceiros do processo *process_id*;
- *RemoveRemoteProcess(a_id,process_id)*, que remove um processo remoto.

A seguir são detalhadas as regras que definem cada uma das operações.

a) Operação *DelegateProcess*

A operação *DelegateProcess* é definida na pela regra AG2.1. Essa regra associa um novo parceiro a um processo. A condição é que o parceiro ainda não esteja associado ao processo. A operação *IncludeRemoteProcess* é chamada na instância APSEE-*Global* do parceiro, tendo como parâmetros o identificador da equipe contratante e o identificador do processo delegado.

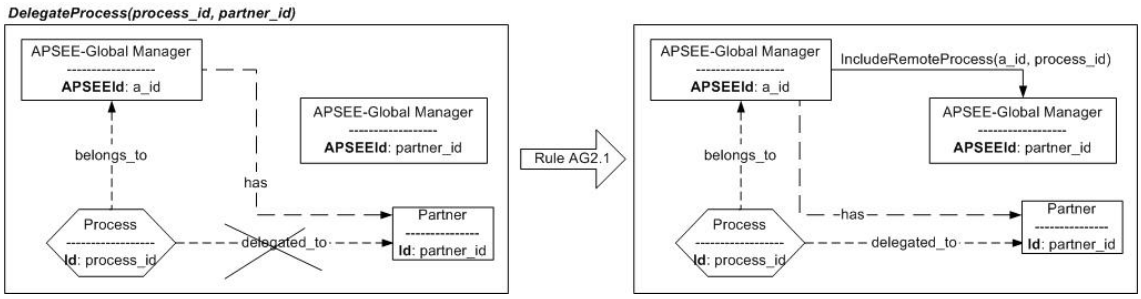

Figura 4.22: Regra para atribuição de parceiro a um processo

b) Operação *IncludeRemoteProcess*

A operação *IncludeRemoteProcess* insere, no modelo de processo do parceiro contratado, um processo remoto que corresponde ao processo a ele delegado.

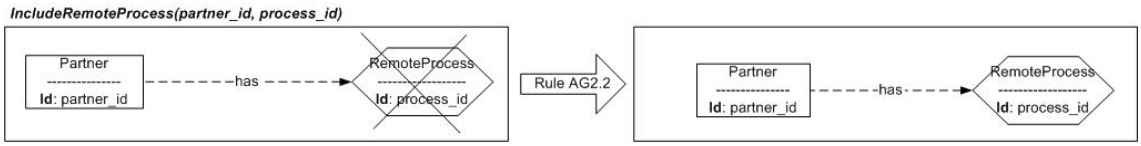

Figura 4.23: Regra para inclusão de processo remoto

c) Operação *RemoveProcessDelegation*

A operação *RemoveProcessDelegation* é definida pela regra AG2.3. Essa regra remove a associação entre o parceiro e o processo, e chama a operação de exclusão de processo remoto na instância APSEE-*Global* do parceiro. Para que a associação seja removida, o processo não pode possuir nenhuma atividade (normal ou decomposta) delegada ao parceiro.

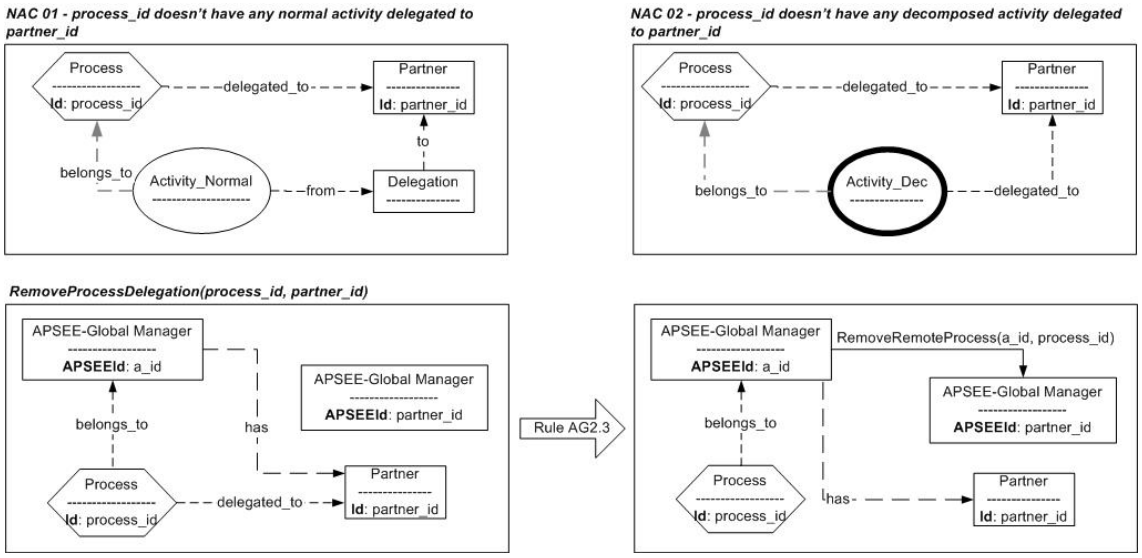

Figura 4.24: Regra para remoção de delegação de processo

d) Operação *RemoveRemoteProcess*

A regra AG2.4 define a operação *RemoveRemoteProcess*, que remove um processo remoto. A condição para remoção é que o processo remoto não possua nenhuma atividade remota associada.

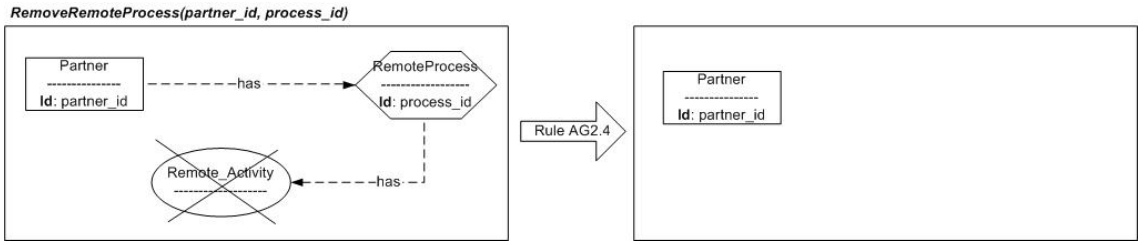

Figura 4.25: Regra para exclusão de processo remoto

- **Delegação de atividades**

A delegação de atividades envolve as seguintes operações:

- *DelegateActivity(process_id,act_id,partner_id)*, que delega a atividade *act_id* ao parceiro *partner_id*;
- *IncludeRemoteActivity(partner_id,process_id,act_id,state)*, que define a atividade remota *act_id*;
- *RemoveActivityDelegation(process_id,act_id,partner_id),* que remove a delegação de *act_id* a *partner_id;*

- *RemoveRemoteActivity(partner_id,process_id,act_id),* que remove a atividade remota *act_id;*
- *IncludeRemoteInputActivity(partner_id,process_id,act_id,art_id),*que define o artefato remoto *art_id* como artefato de entrada da atividade *act_id.*
- *IncludeRemoteOutputActivity(partner_id,process_id,act_id,art_id),* que define o artefato remoto *art_id* como artefato de saída da atividade *act_id.*
- *RemoveRemoteInputActivity(partner_id,process_id,act_id,art_id),* que dessassocia o artefato *art_id* da atividade *act_id;*
- *RemoveRemoteOutputActivity(partner_id,process_id,act_id,art_i*d), que dessassocia o artefato *art_id* da atividade *act_id.*

Além das operações citadas, foram definidas três operações auxiliares para manter a integridade do modelo e para realização de testes necessários para execução de outras operações:

- *ActivityDelegationConsistency*, que mantém a integridade do modelo em relação à delegação de atividades.
- *ArtifactDelegationConsistency*, que mantém a integridade do modelo em relação aos artefatos delegados.
- *RemoteArtifactConsistency*, que mantém a integridade do modelo em relação aos artefatos remotos.
- *Test Subtask*, que verifica a existência da relação *Subtask* entre atividades;
- *Test ActivityBelongsToProcess*, que verifica a existência da relação *belongs_to* entre uma atividade e um processo.

A seguir são apresentadas as regras que definem cada operação.

a) Operação *DelegateActivity*

Existem três regras que definem a operação de delegação de atividades. As regras AG3.1 e AG3.2 tratam da delegação de atividades normais, e a regra AG3.3 trata da delegação de atividades decompostas.

A regra AG3.1 estabelece que a delegação é realizada caso a atividade não aloque recursos ou agentes e não tenha iniciado a execução. A operação de inclusão de atividades remotas é chamada na instância APSEE-*Global* do parceiro, passando o estado da atividade como um dos parâmetros.

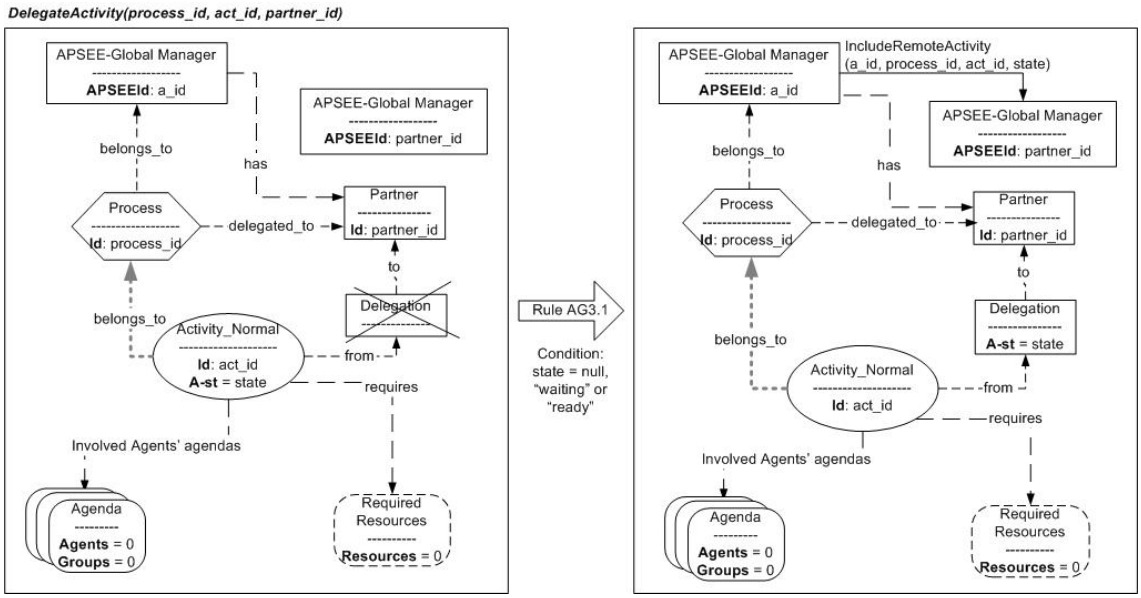

Figura 4.26: Primeira regra para delegação de atividade normal

A regra AG3.2 define a delegação de atividades que estão sendo executadas. Nesse caso, o estado da atividade passado como parâmetro para o parceiro é *ready*, indicando que o parceiro pode iniciar a execução da atividade remota.

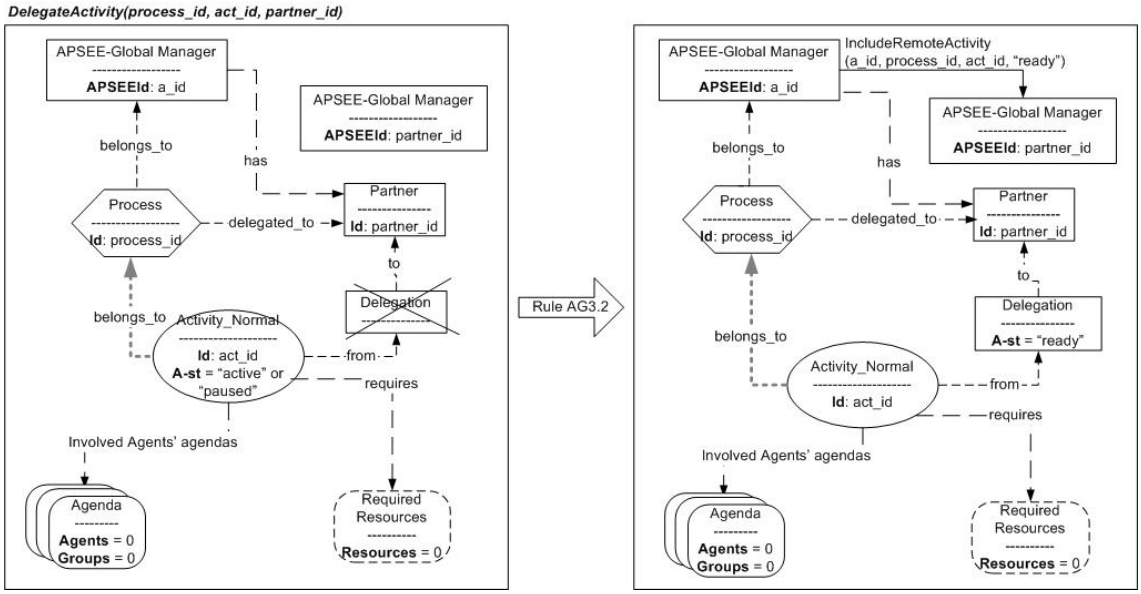

Figura 4.27: Segunda regra para delegação de atividade normal

A regra AG3.3 define a delegação de atividades decompostas. Essa delegação difere das anteriores, no sentido de que nenhuma chamada de operação na instância APSEE-*Global* é disparada. Na realidade, a delegação de uma atividade decomposta significa que todas as suas atividades-filha serão delegadas, mas a execução da atividade decomposta em nenhum momento é atribuída ao parceiro. A condição para que uma atividade decomposta seja delegada é que todas as suas sub-atividades normais sejam abstratas e não tenham finalizado a execução. Atividades decompostas que possuam atividades automáticas como atividades-filha não podem ser delegadas.

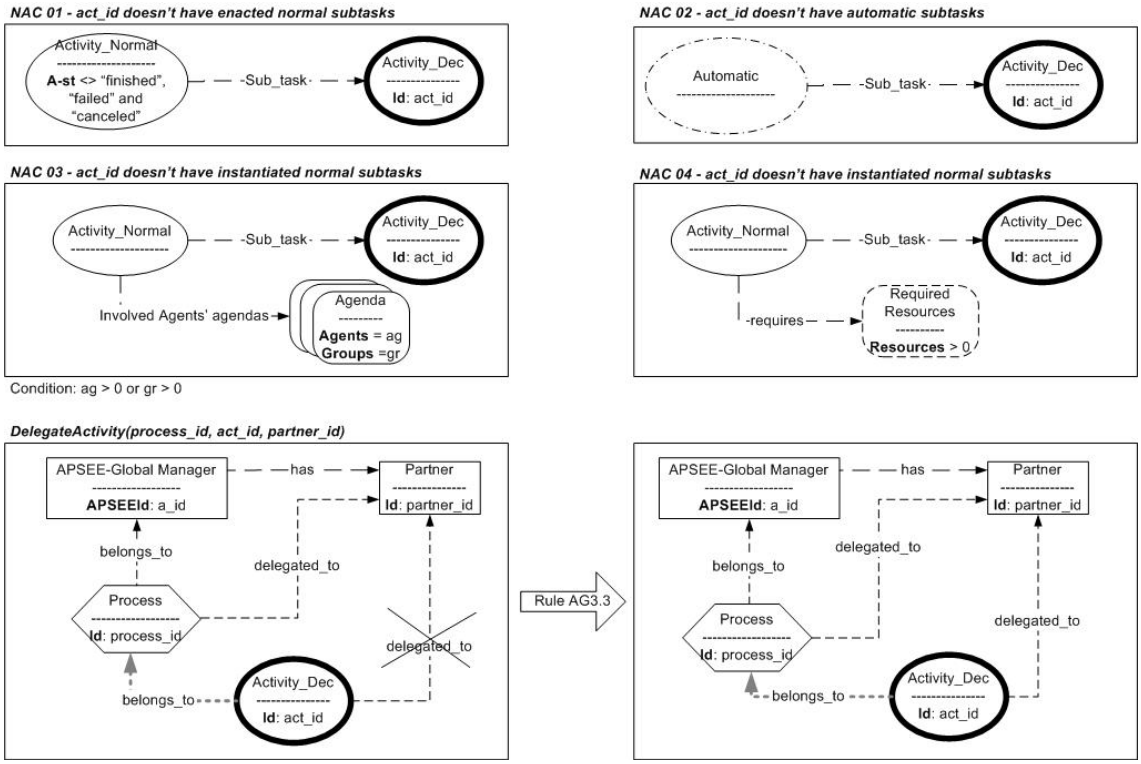

Figura 4.28: Regra para delegação de atividade decomposta

b) Operação *IncludeRemoteActivity*

A definição de atividades remotas é definida pela regra AG3.4. Essa regra associa uma atividade a um processo remoto, desde que a atividade ainda não esteja associada ao processo.

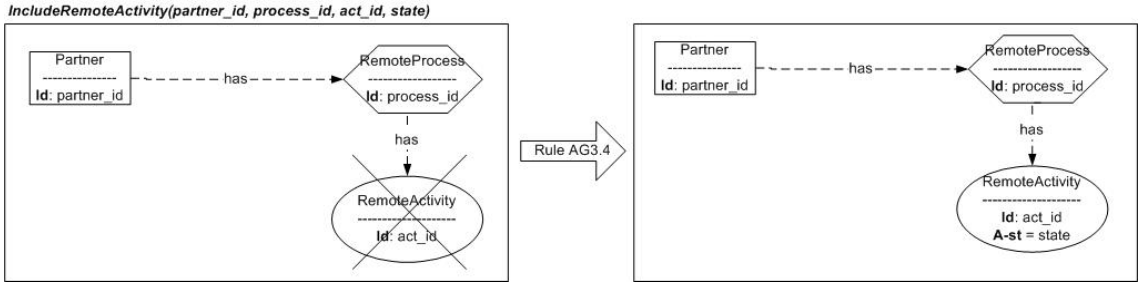

Figura 4.29: Regra para inclusão de atividade remota

c) Operação *RemoveActivityDelegation*

A remoção da delegação de uma atividade a um processo é definida por duas regras: a primeira remove a delegação de atividades normais, e a segunda remove a delegação de atividades decompostas.

A regra AG3.5 define a remoção da delegação de atividades normais. Para que a delegação seja removida, o parceiro não pode ter finalizado a execução da atividade. Quando a delegação é removida, o parceiro para o qual a atividade estava delegada é notificado, pela chamada do método *RemoveRemoteActivity*.

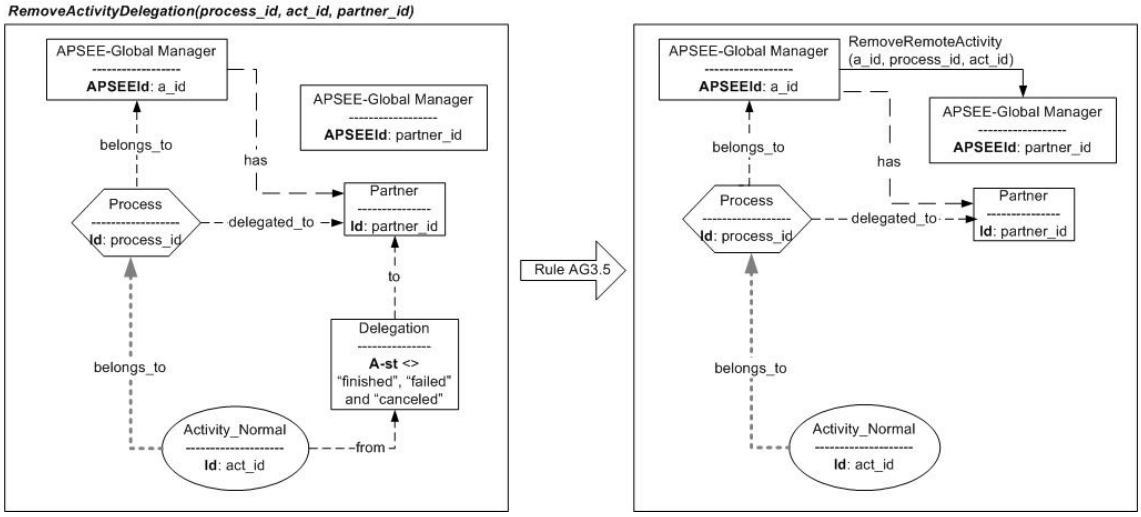

Figura 4.30: Regra para remoção de delegação de atividade normal

A regra AG3.6 define a remoção de delegação de atividades decompostas. A remoção da delegação não possui efeito sobre as atividades já delegadas; apenas indica que novas sub-atividades definidas na atividade decomposta não serão delegadas automaticamente ao parceiro.

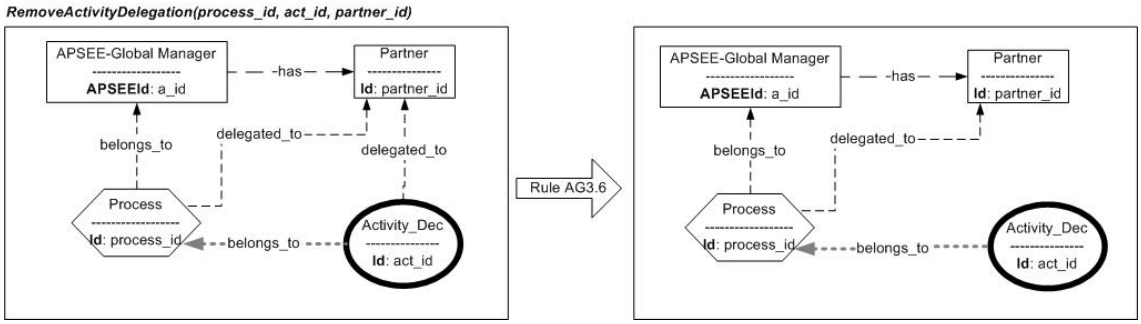

Figura 4.31: Regra para remoção de delegação de atividade decomposta

d) Operação *RemoveRemoteActivity*

A exclusão de atividades remotas de um processo é definida pela regra AG3.7. A exclusão ocorre desde que a execução da atividade remota não tenha sido finalizada.

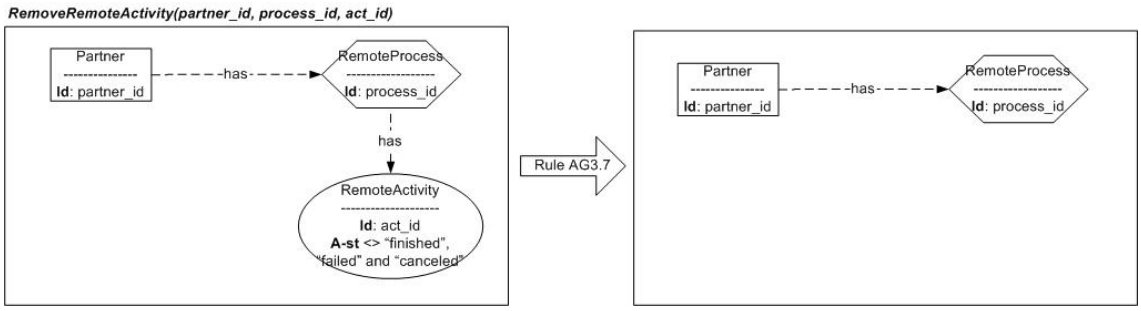

Figura 4.32: Regra para exclusão de atividade remota

e) Operação *ActivityDelegationConsistency*

A operação *ActivityDelegationConsistency* é definida por regras que verificam a consistência na delegação de atividades. Essas regras propagam a delegação de atividades decompostas para as suas sub-atividades.

As regras AG3.8 e AG3.9 procuram atividades decompostas delegadas com sub-atividades normais não delegadas, e realiza a delegação das mesmas.

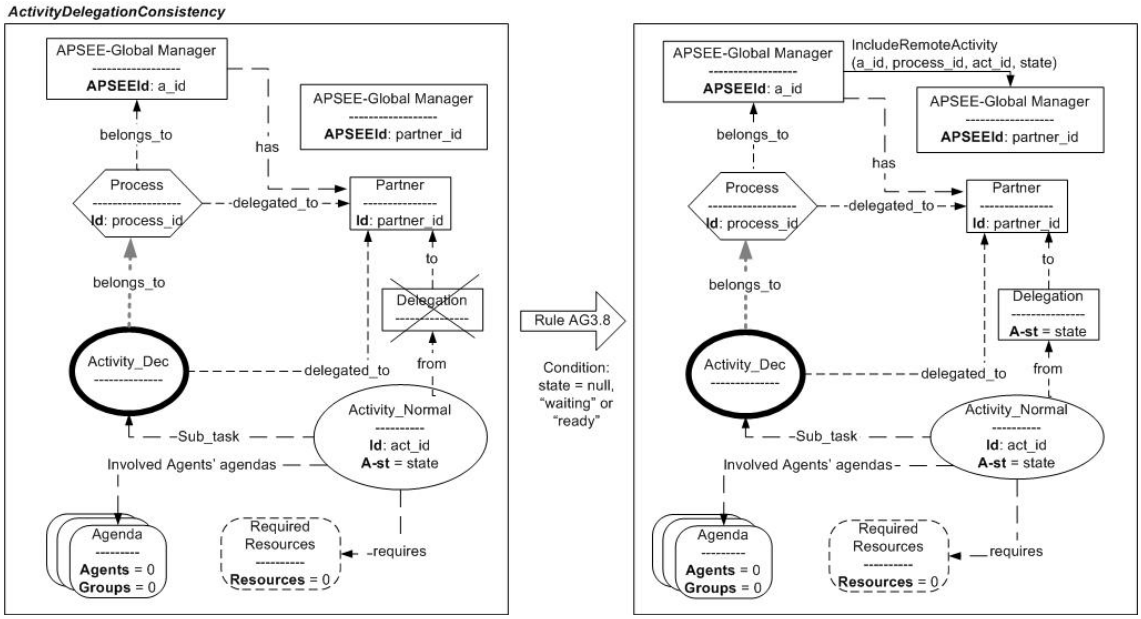

Figura 4.33: Primeira regra de consistência na delegação de atividades

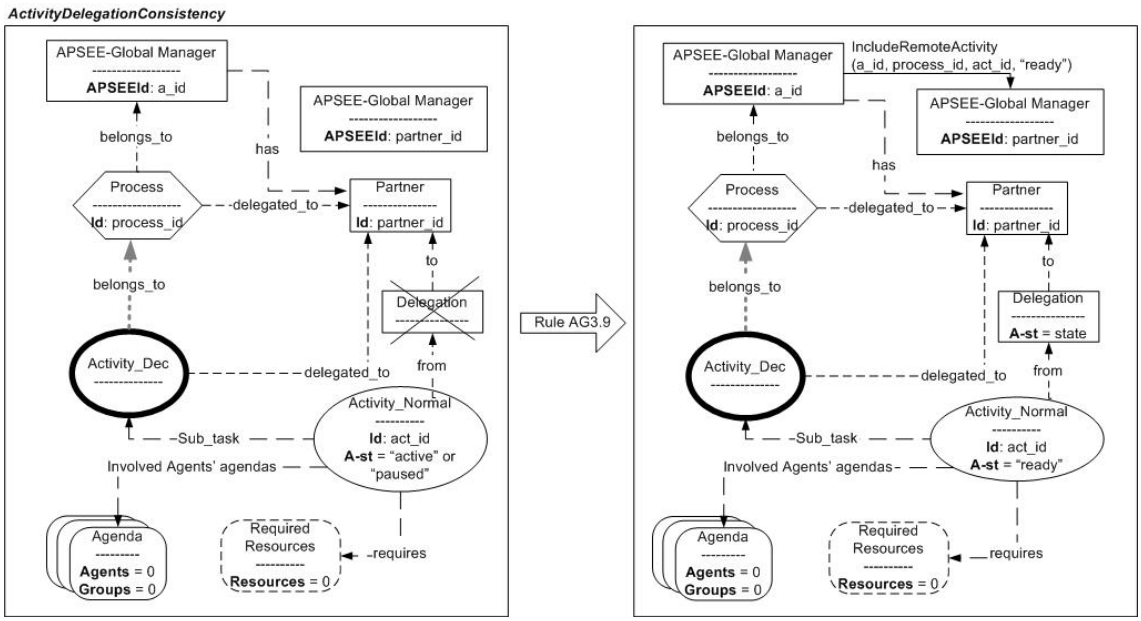

Figura 4.34: Segunda regra de consistência na delegação de atividades

A propagação para atividades decompostas é definida pela regra AG3.10. Essa regra procura atividades decompostas delegadas que possuam sub-atividades decompostas não delegadas ao mesmo parceiro. A regra então determina que a sub-atividade decomposta será também delegada ao parceiro.

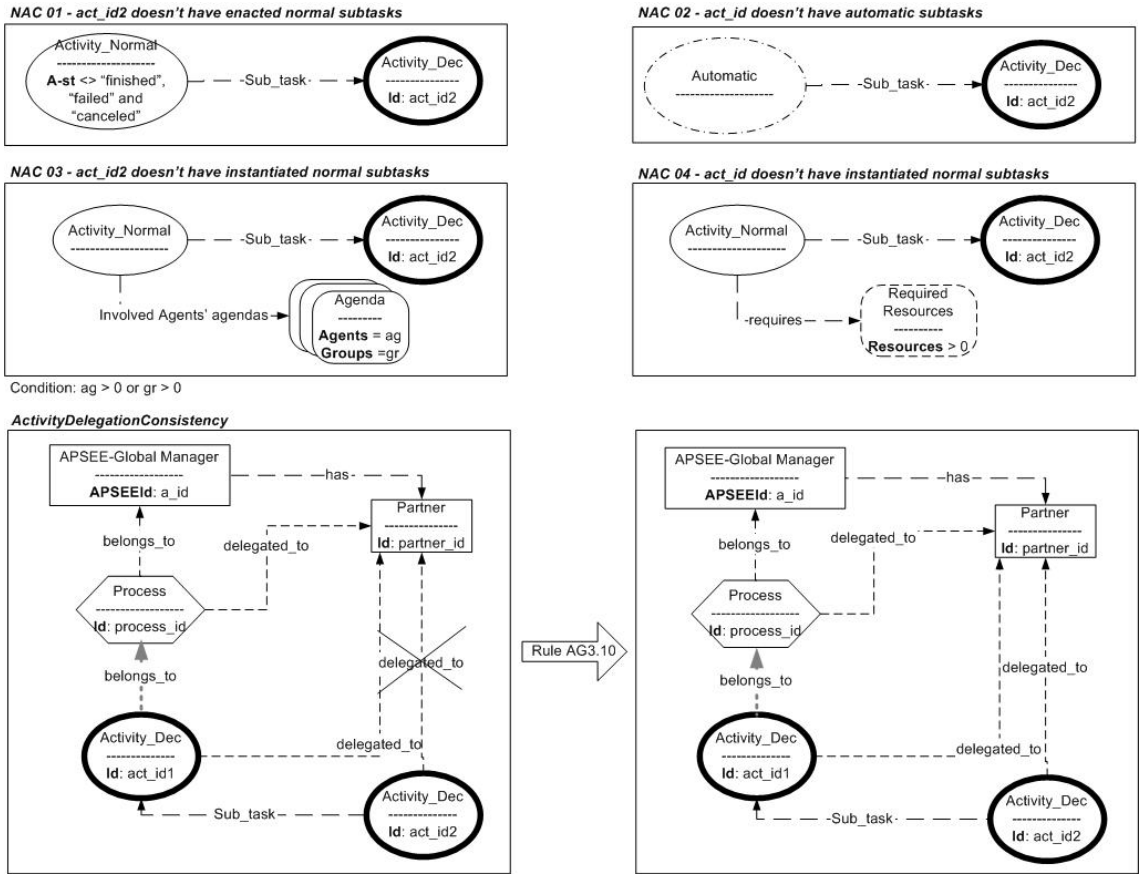

Figura 4.35: Terceira regra de consistência na delegação de atividades

f) Operação *ArtifactDelegationConsistency*

A operação *ArtifactDelegationConsistency* é definida por regras que verificam a consistência na delegação de artefatos. Essas regras definem que todos os artefatos de entrada e saída de atividades delegadas também devem estar delegados aos parceiros que executam a atividade. As regras AG3.11 e AG3.12 delegam os artefatos associados a atividades normais delegadas.

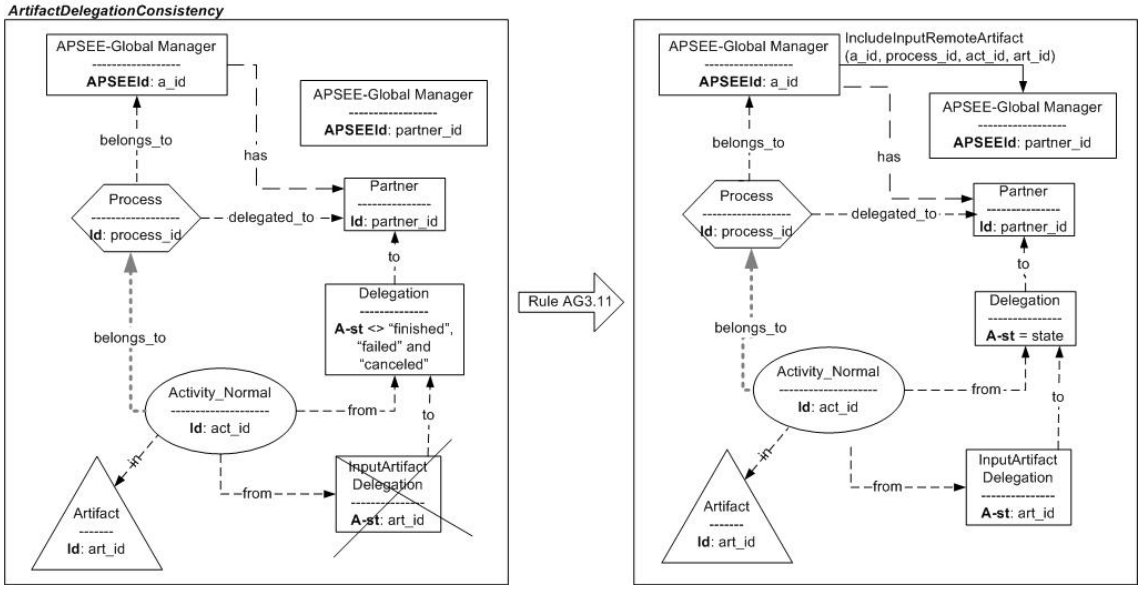

Figura 4.36: Primeira regra de consistência da delegação de artefatos

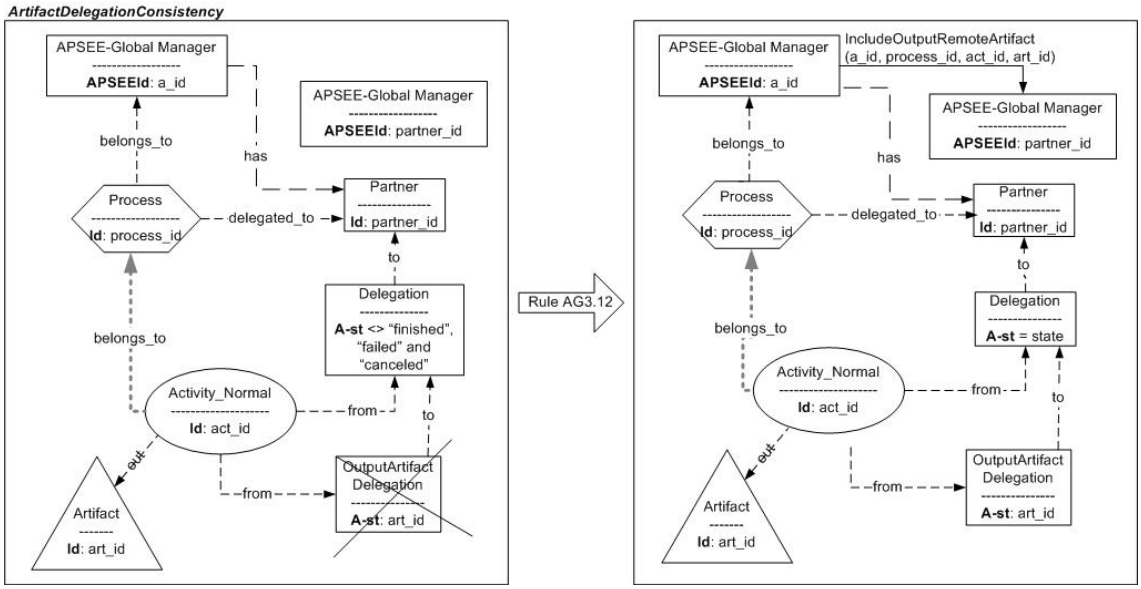

Figura 4.37: Segunda regra de consistência da delegação de artefatos

As regras AG3.13 e A3.14 removem a delegação de artefatos que não estão associados à atividade delegada.

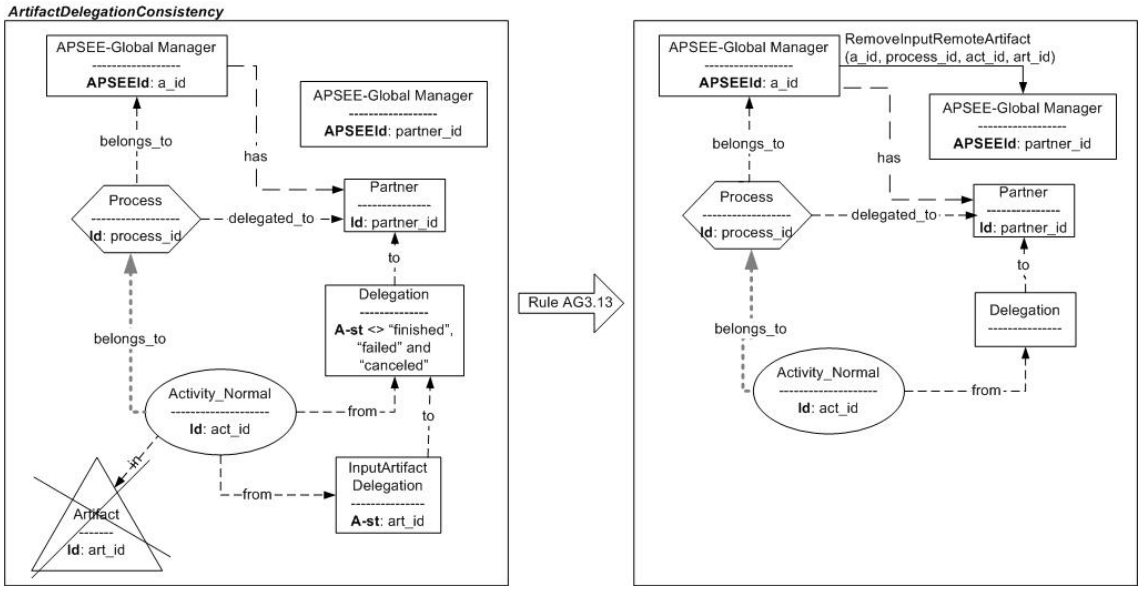

Figura 4.38: Terceira regra de consistência da delegação de artefatos

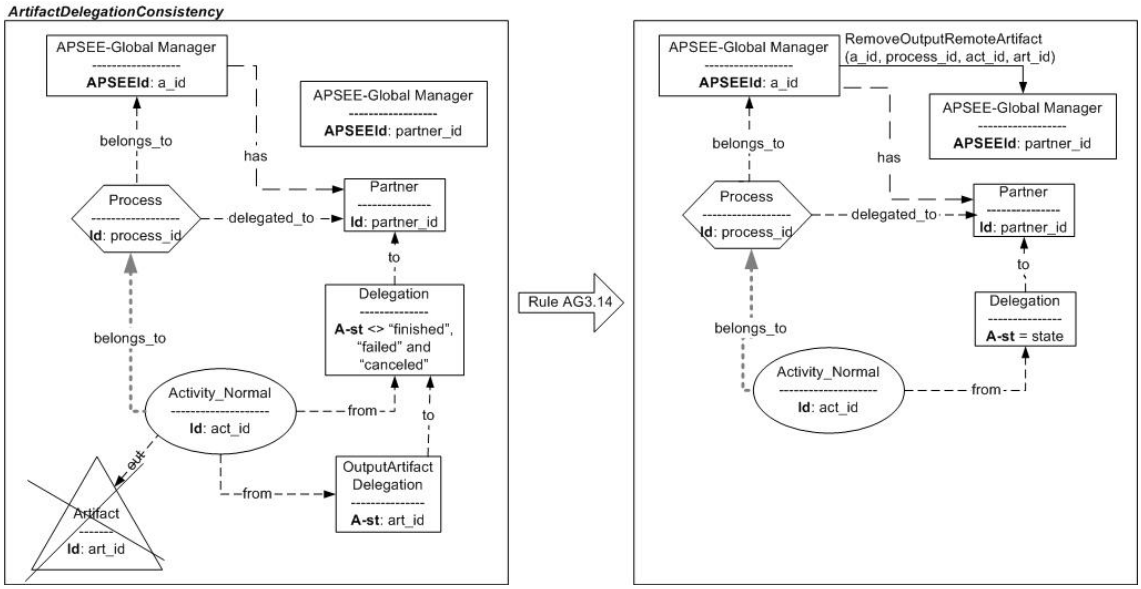

Figura 4.39: Quarta regra de consistência da delegação de artefatos

g) Operação *IncludeRemoteInputActivity*

A regra AG3.15 define a operação de inclusão de artefatos remotos de entrada.

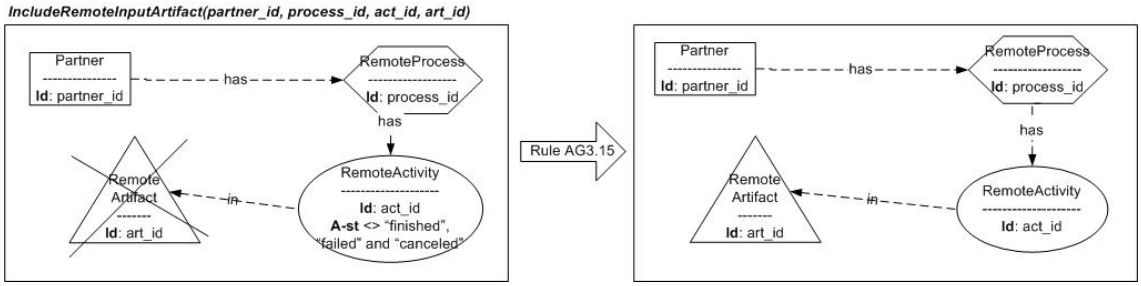

Figura 4.40: Regra para inclusão de artefato remoto de entrada

h) Operação *IncludeRemoteOutputActivity*

A inclusão de artefatos remotos de saída é definida pela regra AG3.16.

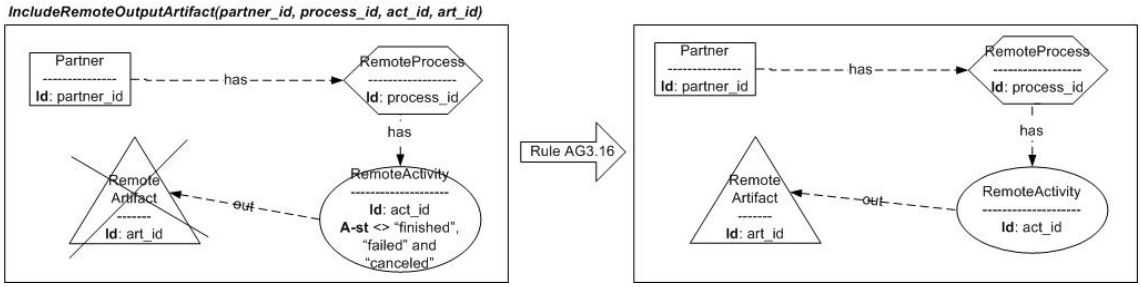

Figura 4.41: Regra para inclusão de artefato remoto de saída

i) Operação *RemoveRemoteInputActivity*

A exclusão de artefatos de entrada de atividades remotas é definida pela regra AG3.17.

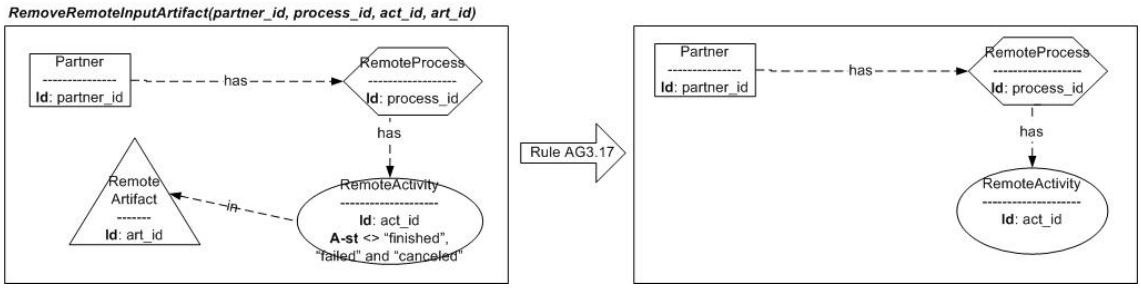

Figura 4.42: Regra para exclusão de artefato remoto de entrada

j) Operação *RemoveRemoteOutputActivity*

A exclusão de artefatos de saída de atividades remotas é definida pela regra AG3.18.

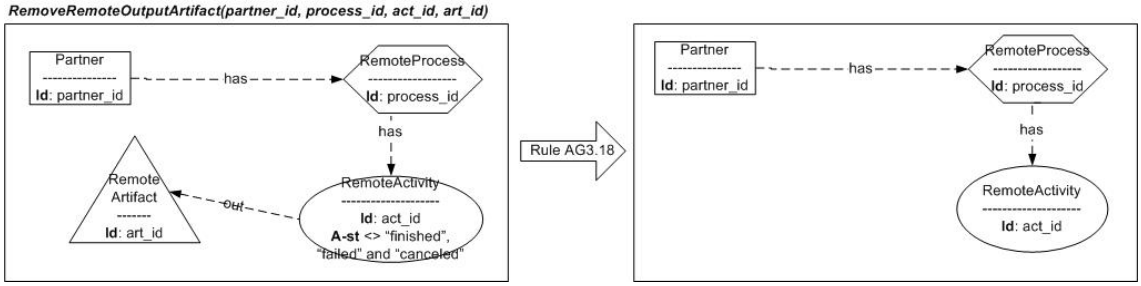

Figura 4.43: Regra para exclusão de artefato remoto de saída

k) Operação *Test Subtask*

As duas regras a seguir, AG3.19 e AG3.20, definem a relação *sub_task* entre uma atividade (normal, automática ou decomposta) e uma atividade decomposta. Essa relação existe quando: a atividade decomposta é definida por um modelo de processo no qual a outra atividade está definida (regra AG3.19); a atividade decomposta é definida por um modelo de processo no qual está definida uma atividade decomposta que possui relação de *sub_task* com a segunda atividade da relação (regra AG3.20).

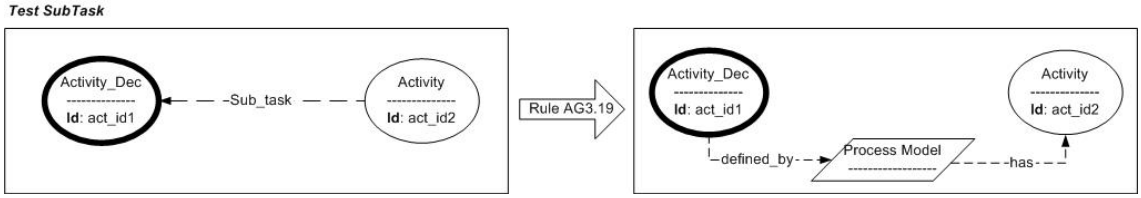

Figura 4.44: Primeira regra de definição da relação *Sub_task*

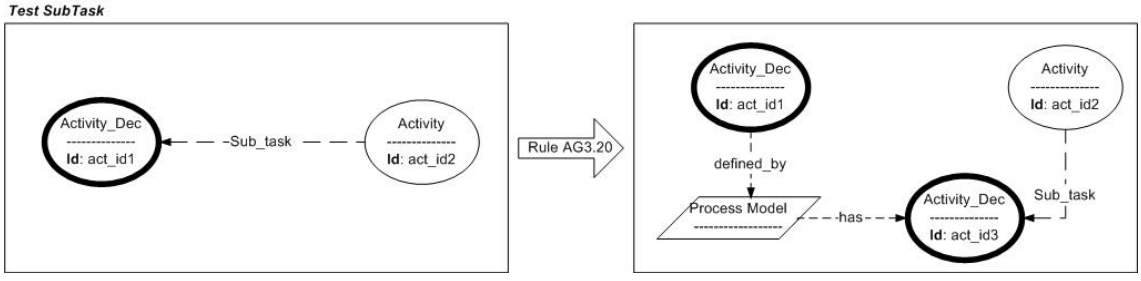

Figura 4.45: Segunda regra de definição da relação *Sub_task*

l) Operação *Test ActivityBelongsToProcess*

A relação *belongs_to* entre um processo e uma atividade é definida pelas regras AG3.21 e AG3.22. A primeira regra define que a relação *belongs_to* existe se a atividade é definida no modelo de processo.

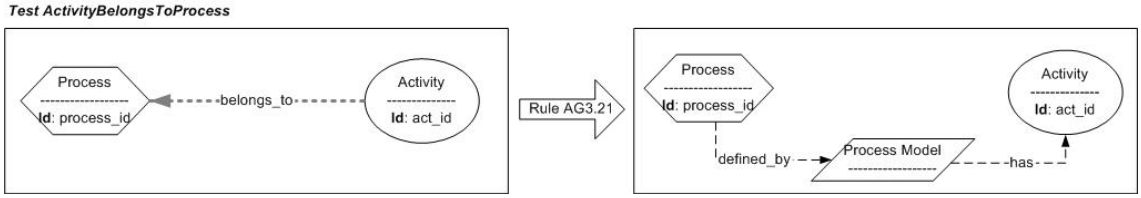

Figura 4.46: Primeira regra de definição de relação *belongs_to*

A segunda regra define a existência da relação *belongs_to* entre o modelo de processo e as sub-atividades das atividades definidas no modelo de processo.

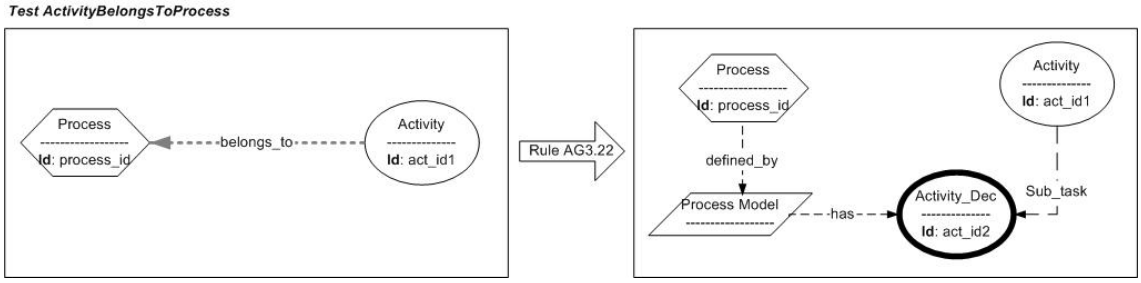

Figura 4.47: Segunda regra de definição de relação *belongs_to*

m) Operação *RemoteArtifactConsistency*

A consistência de artefatos remotos é mantida quando arterfatos que não são mais referenciados por atividades remotas são eliminados.

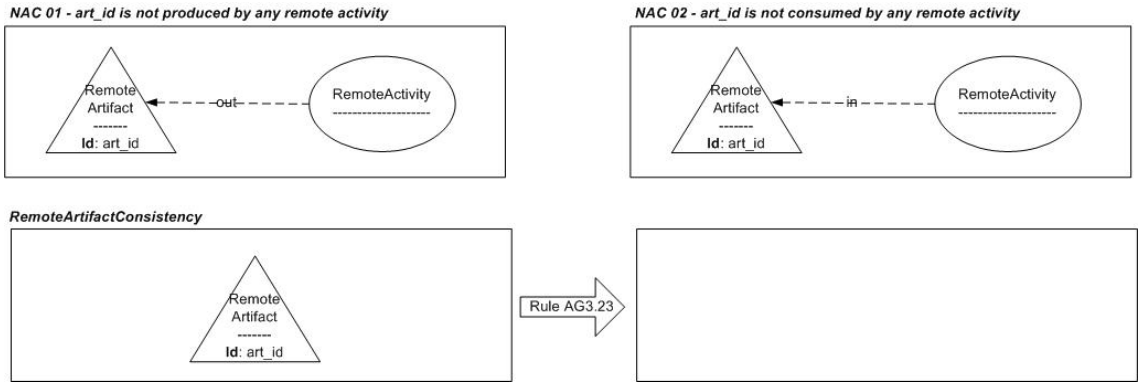

Figura 4.48: Regra de consistência de artefatos remotos

- Mapeamento de processos

O mapeamento de processos é definido pelas seguintes operações:

- *MapRemoteProcess(rem_proc_id, loc_proc_id)*, que mapeia o processo remoto *rem_proc_id* ao processo local *loc_proc_id;*
- *RemoveRemoteProcessMap(rem_proc_id)*, que desfaz o mapeamento do processo remoto *rem_proc_id.*

As duas operações são descritas a seguir.

a) Operação *MapRemoteProcess*

Duas regras definem a operação *MapRemoteProcess*, que mapeia um processo remoto a um processo local. A primeira delas, AG4.1, determina que o mapeamento só ocorre se o processo remoto não estiver mapeado para nenhum outro processo local, e se o processo local não mapear nenhum outro processo remoto.

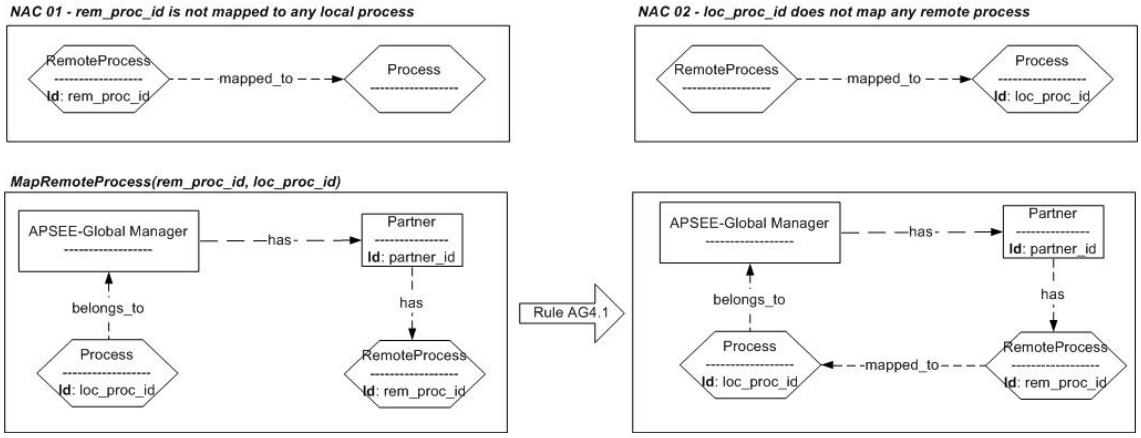

Figura 4.49: Primeira regra para mapeamento de processo remoto

A regra AG4.2 altera o mapeamento de um processo remoto. Ela define que um processo remoto *rem_proc_id*, já mapeado para um processo local *proc_id_2*, pode passar a ser mapeado para um processo local *loc_proc_id*. As restrições são que *rem_proc_id* não pode possuir nenhuma atividade mapeada, e *loc_proc_id* não pode mapear nenhum outro processo remoto.

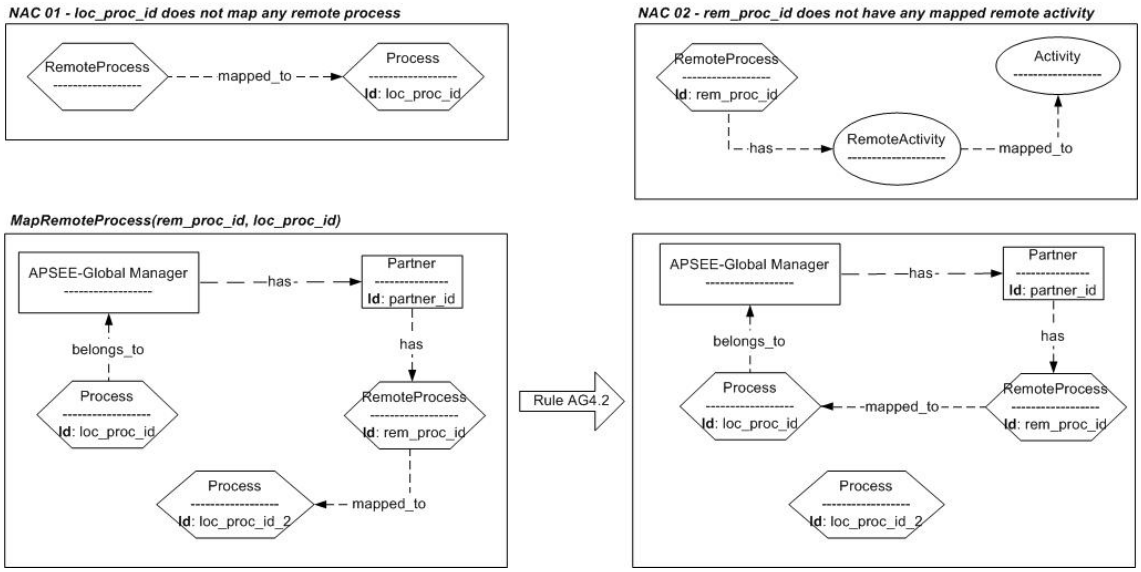

Figura 4.50: Segunda regra para mapeamento de processo remoto

b) Operação *RemoveRemoteProcessMap*

A regra AG4.3 desfaz o mapeamento de um processo remoto para um processo local, desde que nenhuma atividade do processo remoto esteja mapeada para atividades locais.

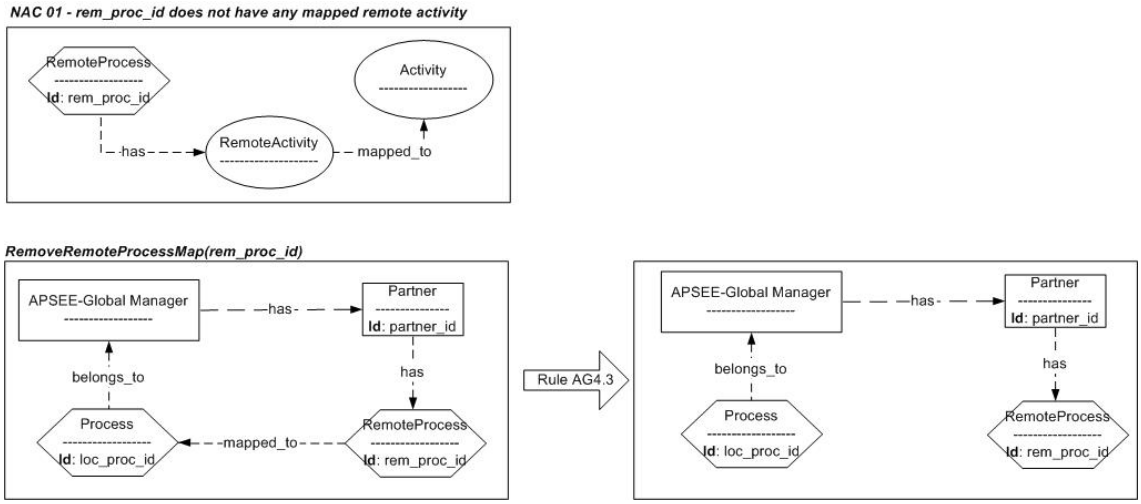

Figura 4.51: Regra para remoção de mapeamento de processo remoto

- Mapeamento de atividades

O mapeamento de atividades remotas para atividades locais envolve as seguintes operações:

- *MapRemoteActivity(remote_act_id, local_act_id),* que mapeia a atividade remota *remote_act_id* para a atividade local *local_act_id;*
- *RemoveRemoteActivityMapping(remote_act_id, local_act_id),* que remove o mapeamento da atividade remota *remote_act_id* para a atividade local *local_act_id.*

A seguir, as operações são descritas:

a) Operação *MapRemoteActivity*

O mapeamento de atividades remotas é guiado por duas regras: AG5.1, que trata do mapeamento de atividades que não estão prontas para execução, e AG5.2, que define o mapeamento de atividades em execução ou prontas para executar.

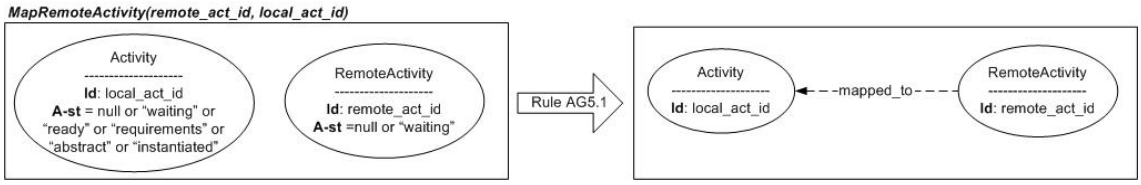

Figura 4.52: Primeira regra para mapeamento de atividade remota

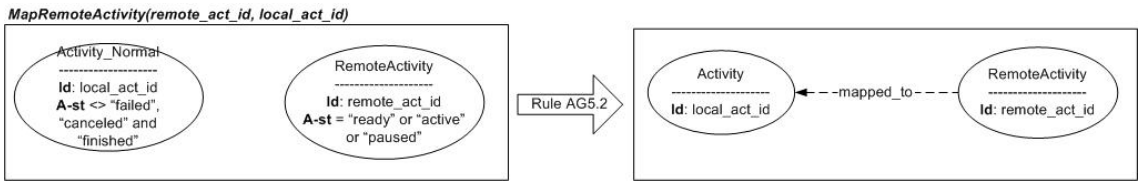

Figura 4.53: Segunda regra para mapeamento de atividade remota

b) Operação *RemoveRemoteActivityMapping*

A remoção do mapeamento de atividades remotas é definida pela regra AG5.3. A condição é que tanto a atividade remota quanto a atividade local não tenham finalizado a execução.

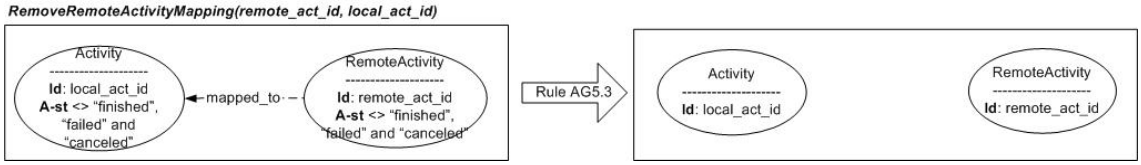

Figura 4.54: Regra de remoção de mapeamento de atividade remota

- Mapeamento de artefatos

O mapeamento de artefatos remotos para artefatos locais envolve as seguintes operações:

- *MapRemoteArtifact(remote_art_id, local_art_id),* que mapeia o artefato remoto *remote_art_id* para o artefato local *local_art_id;*
- *RemoveRemoteArtifactMapping(remote_art_id, local_art_id),* que remove o mapeamento do artefato remoto *remote_art_id* para o artefato local *local_art_id*

A seguir, as operações são descritas:

a) Operação *MapRemoteArtifact*

O mapeamento de artefatos remotos é guiado pela regra AG6.1. Essa regra define que artefatos de saída de atividades remotas são mapeados para artefatos locais.

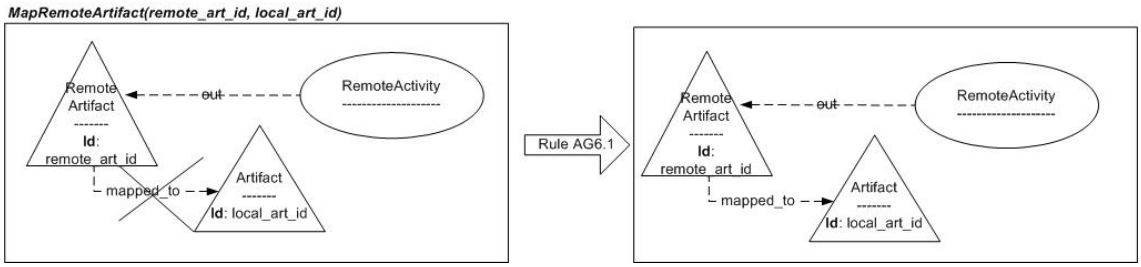

Figura 4.55: Regra para mapeamento de artefato remoto

b) Operação *RemoveRemoteArtifactMapping*

A remoção do mapeamento de artefatos remotos é definida pela regra AG6.2.

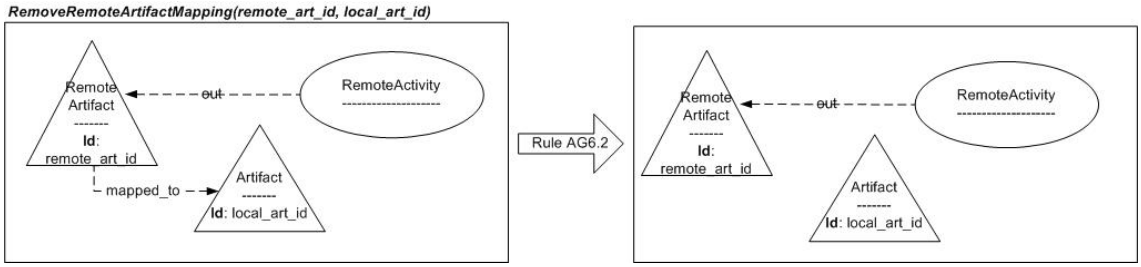

Figura 4.56: Regra para remoção de mapeamento de artefato remoto

4.2.3.3. Sincronização na execução dos processos

Como o gerenciamento da distribuição de processos é completamente automatizado, as regras de sincronização não são ativadas por estímulos do usuário; são executadas automaticamente, com o objetivo de atualizar o estado do modelo de processo para que a execução possa continuar. Essas regras visam, portanto, manter o estado do modelo de processo consistente. A cada evento ocorrido no ambiente, o APSEE-*Global* busca atualizar o estado de todas as atividades. As regras que definem a sincronização na execução de processos podem ser divididas nos seguintes grupos:

- Regras de início da execução de atividades remotas
- Regras que atualizam o estado da atividade delegada
- Regras que propagam falhas
- Regras que propagam cancelamento

A seguir são apresentadas as operações e regras que compõem cada um desses grupos.

- **Início da execução de atividades remotas**

O início da execução de uma atividade remota só é possível quando as condições para início de execução da atividade delegada que ela representa estão satisfeitas. Assim, as mudanças de estado de atividades remotas para *waiting* ou *ready* são realizadas unicamente a partir do modelo de processo no qual a atividade está definida. Os parceiros responsáveis pela execução das atividades remotas não podem iniciar a execução das atividades mapeadas pela atividade remota antes que esta possa ser executada, sob pena de violarem as dependências do processo distribuído.

A regra AG7.1 define a alteração do estado de delegações para *waiting*.

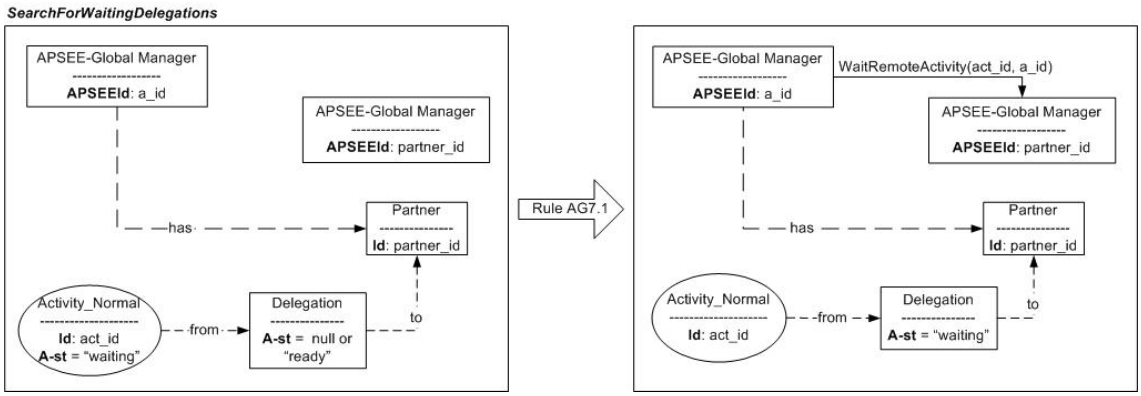

Figura 4.57: Regra para alteração do estado de delegação para *waiting*

A regra AG7.2 define a alteração do estado da delegação quando esta não está pronta para executar (estados *null* ou *waiting*), embora a atividade delegada esteja no estado *ready*.

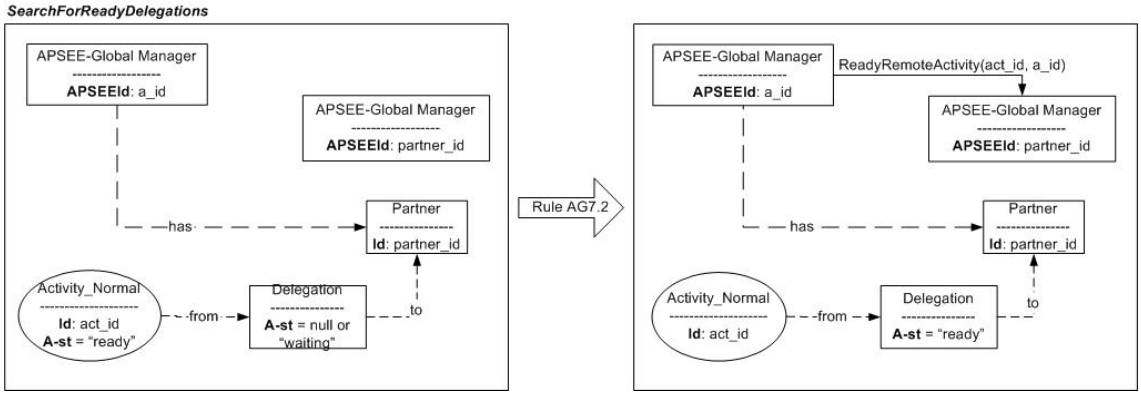

Figura 4.58: Regra para alteração do estado de delegação para *ready*

Ambas as regras apresentadas fazem chamada de operação da instância APSEE-*Global* do parceiro. As operações chamadas alteram o estado da atividade remota no parceiro, como definem as regras AG7.3 e AG7.4.

A regra AG7.3 define a função *WaitRemoteActivity*, que altera o estado da atividade remota para *waiting*.

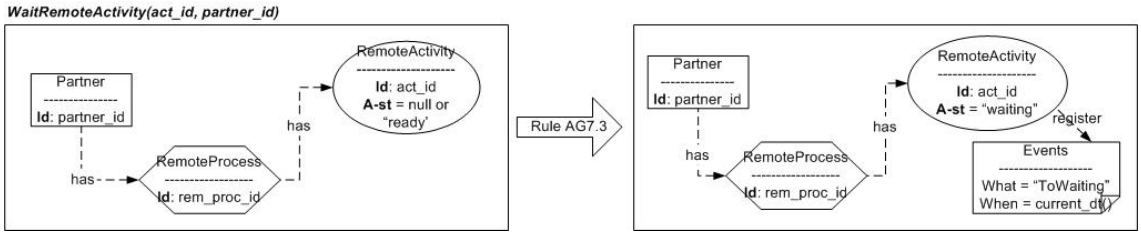

Figura 4.59: Regra de alteração de estado da atividade remota para *waiting*

A função *ReadyRemoteActivity*, que altera o estado da atividade remota para *ready*, é definida pela regra AG7.4.

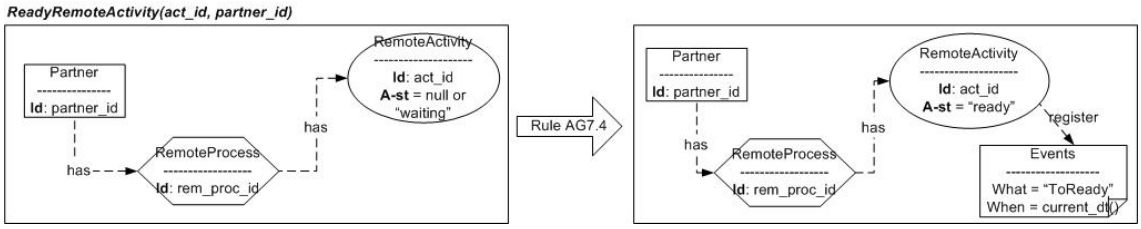

Figura 4.60: Regra de alteração de estado da atividade remota para *ready*

- **Atualização de estado de atividades delegadas**

O estado de execução de atividades delegadas é definido pela combinação dos estados das atividades remotas em cada um dos parceiros. O estado das atividades remotas, por sua vez, é definido pela combinação dos estados das atividades para as quais ela está mapeada. Para definir as possíveis mudanças de estado de atividades remotas foram definidas operações que analisam o estado das atividades para as quais elas estão mapeadas.

A operação que inicia a execução de atividades remotas é definda pela regra AG7.5, que inicia a atividade remota se ela estiver pronta para execução (*ready*) ou pausada (*paused*) e estiver mapeada para alguma atividade em execução. Essa

operação chama a operação *StartDelegatedActivity*, que atualiza o modelo de processo no qual a atividade foi definida, informando que a atividade remota foi iniciada.

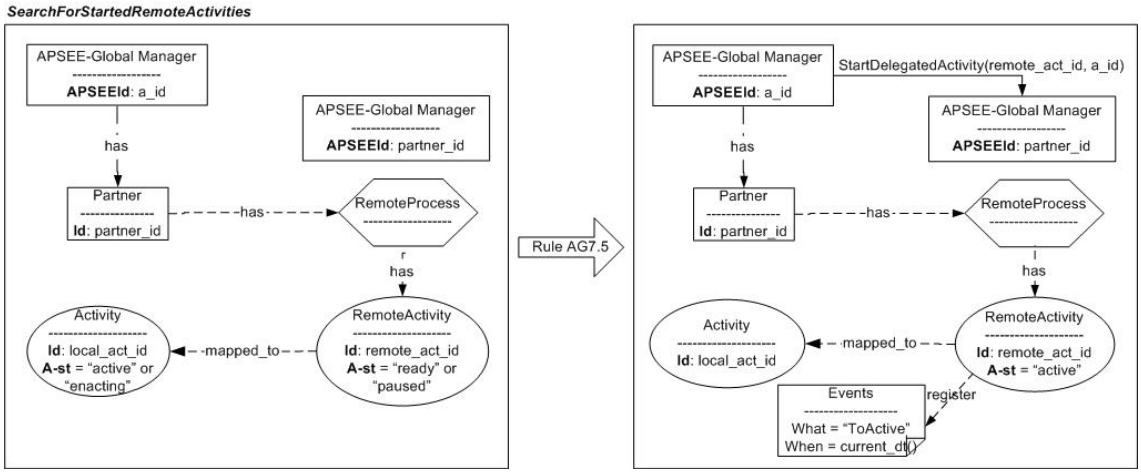

Figura 4.61: Regra de início de execução de atividade remota

A operação que finaliza a execução de atividades remotas é definida pela regra AG7.6. Essa regra finaliza a atividade remota se ela estiver executando, e estiver mapeada apenas para atividades já finalizadas. O término da execução é informado ao modelo de processo onde a atividade está definida por meio da operação *FinishDelegatedActivity*.

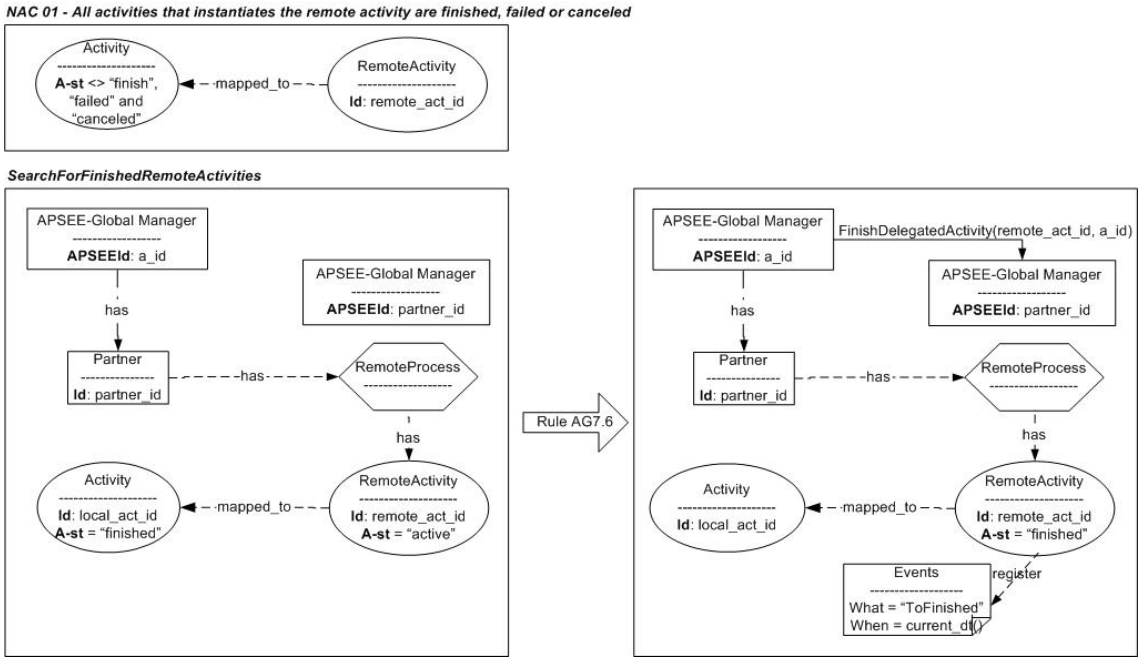

Figura 4.62: Regra de término de execução de atividade remota

A regra AG7.7 define a pausa na execução de atividades remotas mapeadas apenas para atividades pausadas.

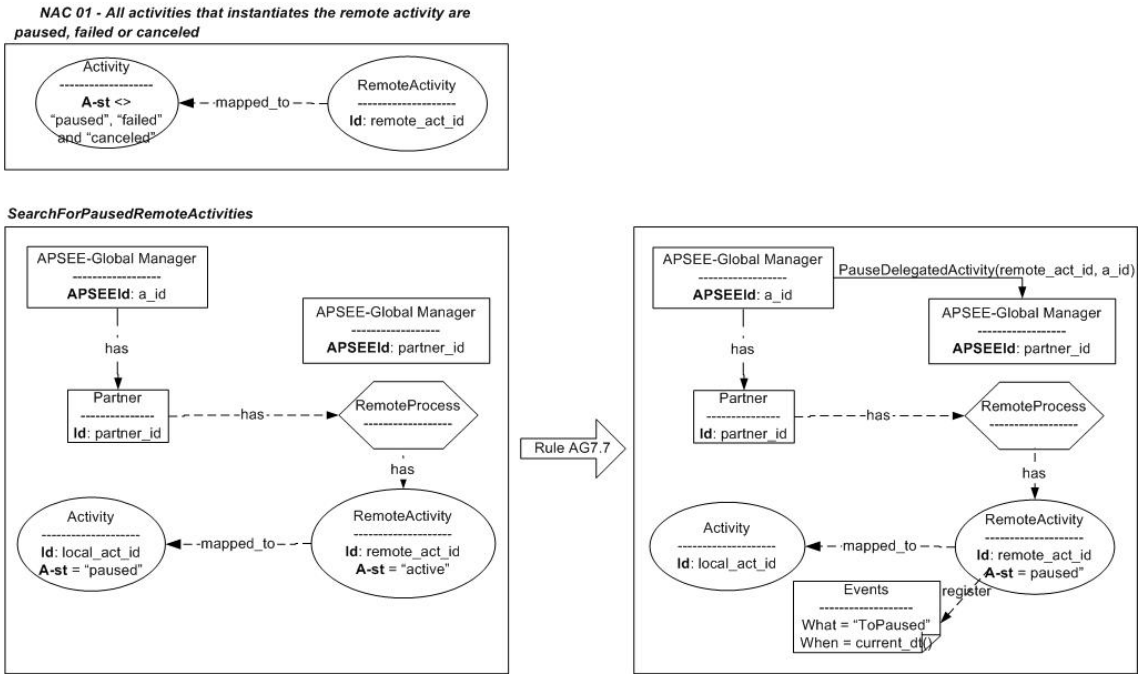

Figura 4.63: Regra de pausa na execução de atividade remota

O estado de atividades delegadas é definido pela combinação dos estados das suas delegações. As operações *StartDelegatedActivity*, *FinishDelegatedActivity* e *PauseDelegatedActivity*, todas tendo como parâmetro o identificador de uma atividade e de um parceiro ao qual ela está delegada, alteram o estado de execução da atividade pelo parceiro. Quando essas operações são executadas, o estado das demais delegações é avaliado, e o resultado da avaliação determina se o estado da atividade será modificado.

A operação *StartDelegatedActivity* é formada por duas regras. A regra AG7.8 informa o início da execução da atividade por um dos parceiros. Segundo a regra, a atividade encontra-se pronta para a execução, e terá seu estado alterado para *active*, indicando o início da execução.

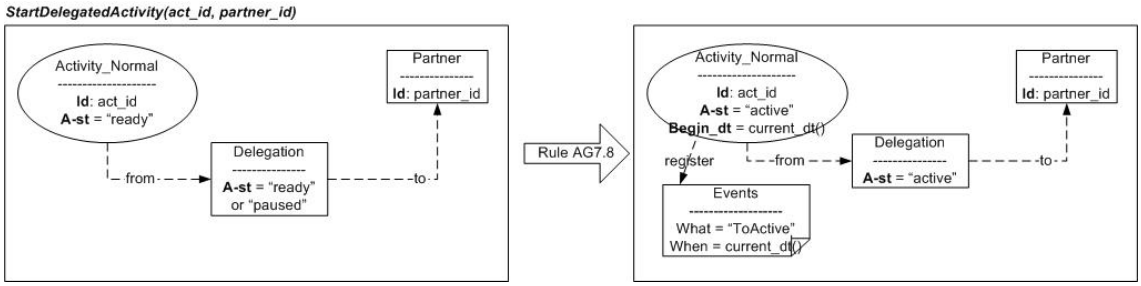

Figura 4.64: Primeira regra de início de execução de atividade delegada

Como mostra a regra AG7.8, para que uma atividade delegada esteja em execução (estado *active*), é necessário que pelo menos um dos parceiros tenha iniciado a execução da atividade remota correspondente. Assim, existem casos em que uma delegação é alterada para o estado *active*, sem que essa alteração modifique o estado da atividade delegada, pois a mesma já se encontra em execução. Esses casos são tratados pela regra AG7.9. Nesse caso, apenas o estado

da delegação é atualizado, sem nenhuma alteração do estado de execução da atividade.

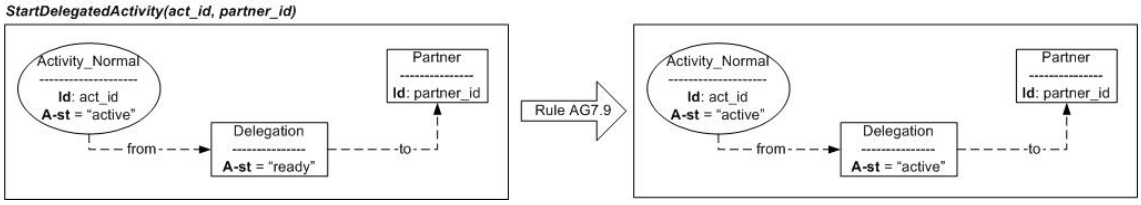

Figura 4.65: Segunda regra de início de execução de atividade delegada

Para que uma atividade delegada seja considerada finalizada, é necessário que todos os parceiros tenham finalizado a execução das atividades remotas correspondentes. Se a execução da atividade delegada por um parceiro é finalizada, e outros parceiros ainda não finalizaram a execução, apenas o estado da delegação é alterado, sem modificação do estado da atividade, como define a regra AG7.10.

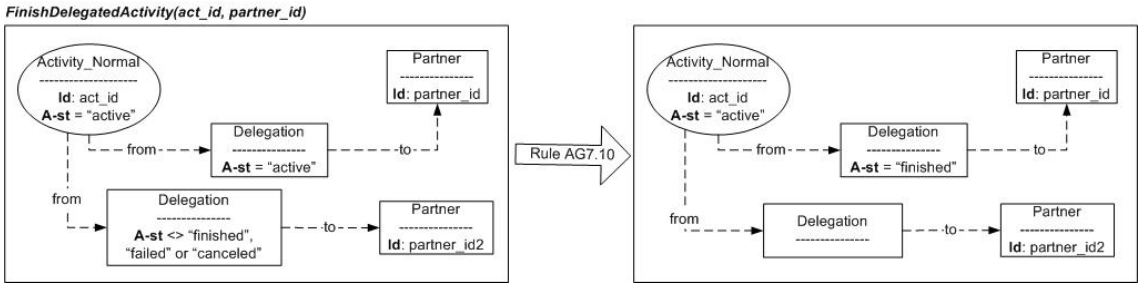

Figura 4.66: Primeira regra de término de execução de atividade delegada

A regra AG7.11 trata do término da execução, por todos os parceiros, de uma atividade delegada. As NACs indicam que todas as condições para término da atividade devem estar satisfeitas. Nesse caso, além de alterar o estado da delegação, a atividade também é finalizada.

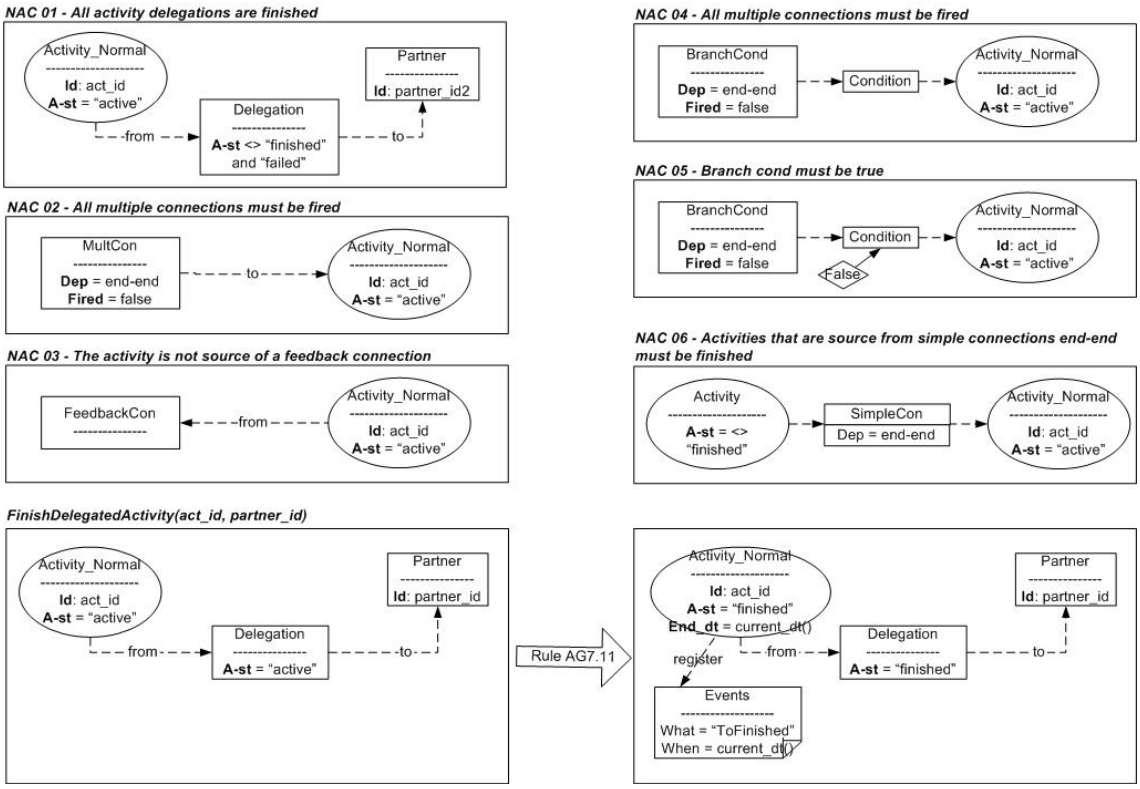

Figura 4.67: Segunda regra de término de execução de atividade delegada

A pausa na execução de atividades delegadas é tratada pelas regras AG7.12 e AG7.13. A primeira diz respeito ao caso em que todas as delegações pausaram. Nesse caso, a execução da atividade delegada é pausada.

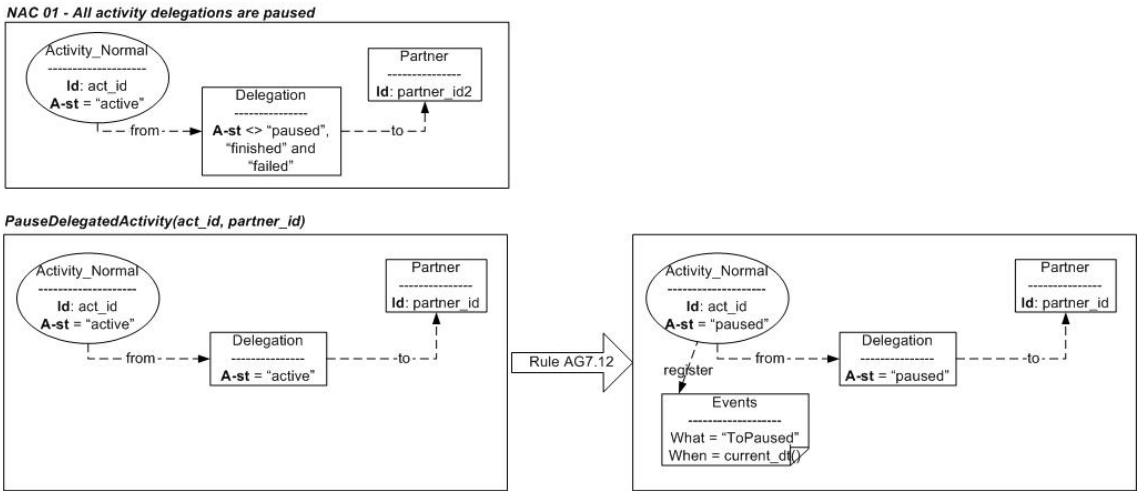

Figura 4.68: Primeira regra de pausa na execução de atividade delegada

A segunda regra trata o caso em que existem delegações em execução ou finalizadas. Nesse caso, a atividade delegada continua em execução, apesar da pausa da delegação.

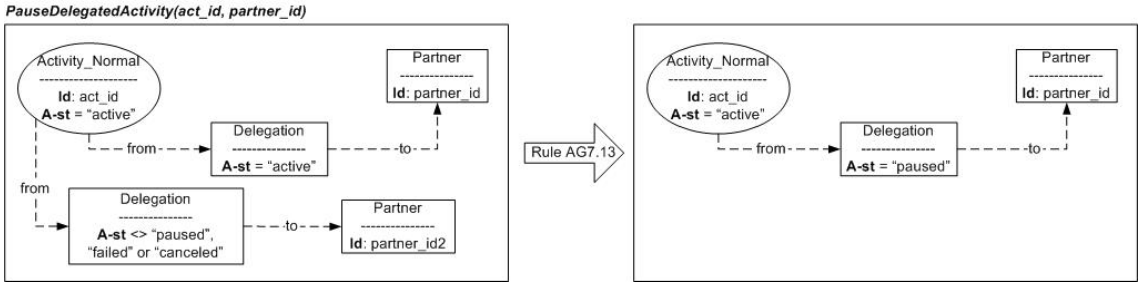

Figura 4.69: Segunda regra de pausa na execução de atividade delegada

- **Propagação de falhas**

Uma atividade remota é definida como falha por duas razões: a primeira é quando todas as atividades locais que a mapeiam falham. A segunda é quando a atividade delegada à qual a atividade remota está associada falha.

A regra AG7.14 define a falha da execução de atividades remotas em razão da falha na execução de todas as atividades locais por ela mapeadas. Essa falha tem efeito direto no estado da atividade delegada; a falha de pelo menos uma atividade remota implica na falha da atividade delegada, com propagação da falha para os demais parceiros.

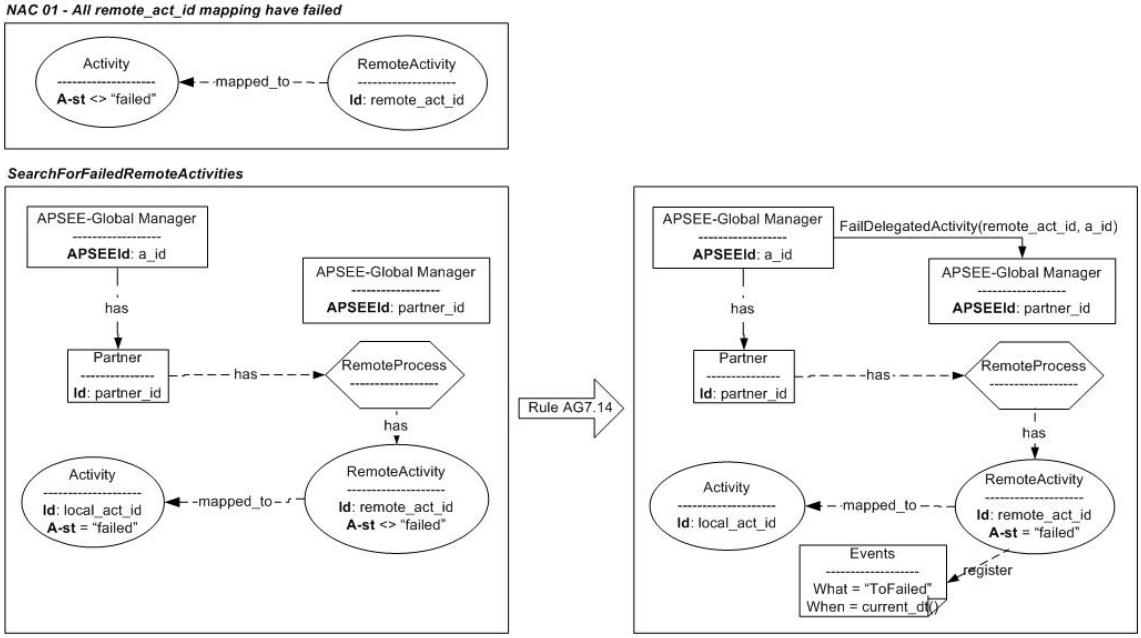

Figura 4.70: Primeira regra de falha de execução de atividade remota

O segundo caso de falha de atividade remota é causado por falha da atividade delegada no modelo de processo da equipe contratante. A regra AG7.15 define esse caso, onde a atividade remota tem seu estado alterado para *failed* quando a operação *FailRemoteActivity* é chamada.

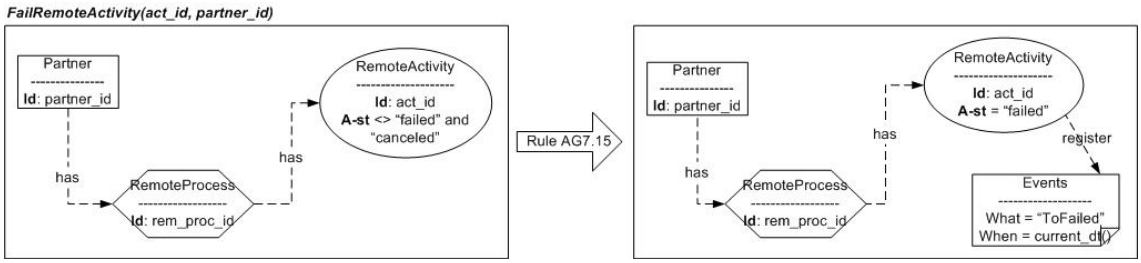

Figura 4.71: Segunda regra de falha de execução de atividade remota

No caso exposto na regra AG7.15, a falha deve ser propagada às atividades locais para as quais a atividade remota está mapeada. Essa propagação é definida pela regra AG7.16. Essa regra define a operação *SearchForFailedActivities*, determinando a falha das atividade mapeadas por atividades remotas que falharam.

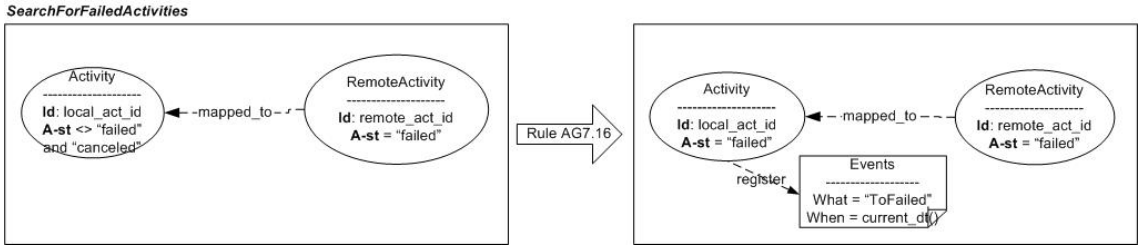

Figura 4.72: Regra para propagação de falha na execução de atividade remota

Como definiu a regra AG7.14, a falha na execução de atividades remotas em decorrência da falha das atividades locais por ela mapeadas implica em chamada à operação *FailDelegatedActivity*. Essa operação propaga a falha da atividade remota, causando a falha da atividade delegada, como mostra a regra AG7.17.

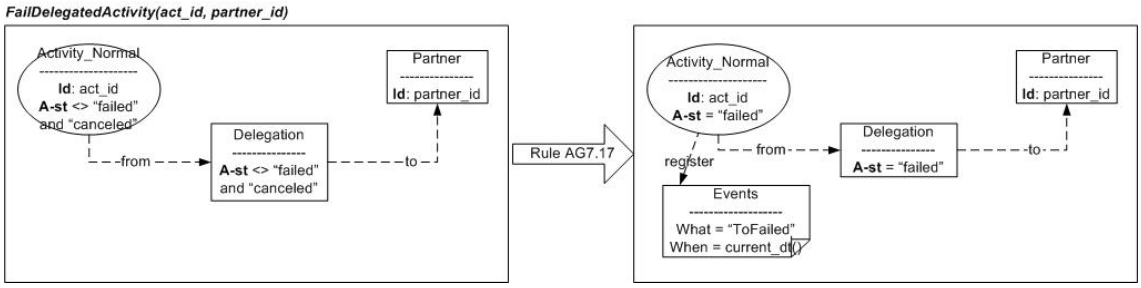

Figura 4.73: Regra de propagação de falha para atividade delegada

A falha de atividades delegadas é propagada para todos os parceiros que a executam. A regra AG7.18 define o comportamento da operação *SearchForFailedDelegations*, responsável por esta propagação.

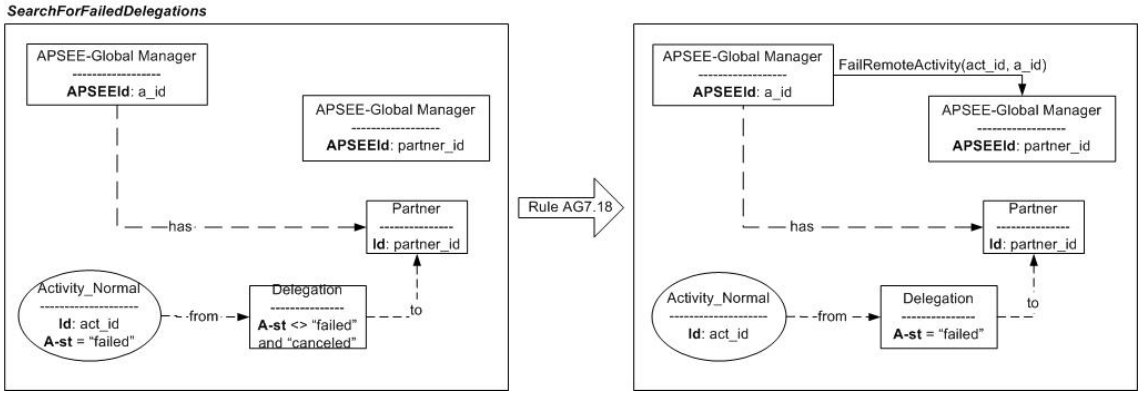

Figura 4.74: Regra de propagação de falha de atividades delegadas

- **Propagação de cancelamento**

O cancelamento de uma atividade remota só acontece por solicitação do gerente responsável pelo modelo de processo do qual ela está definida. Nenhum parceiro tem autonomia para cancelar uma atividade remota, já que esse cancelamento interfere no fluxo do modelo de processo da equipe contratante.

Quando a atividade delegada é cancelada pelo gerente do processo, esse cancelamento é propagado para todos os parceiros, de maneira semelhante ao que ocorre em caso de falha. A regra AG7.19 define essa propagação.

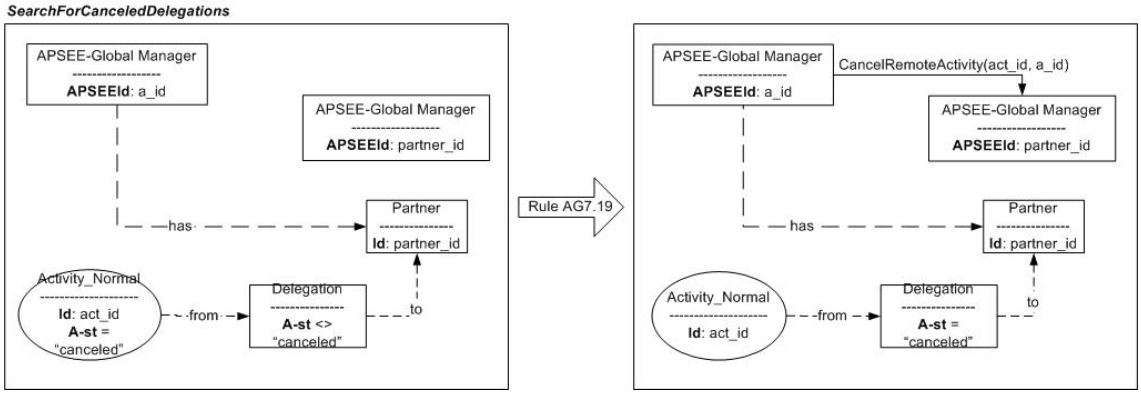

Figura 4.75: Regra de propagação de cancelamento de atividades delegadas

A operação *CancelRemoteActivity*, que tem como objetivo propagar o cancelamento para as atividades remotas, é definida pela regra AG7.20.

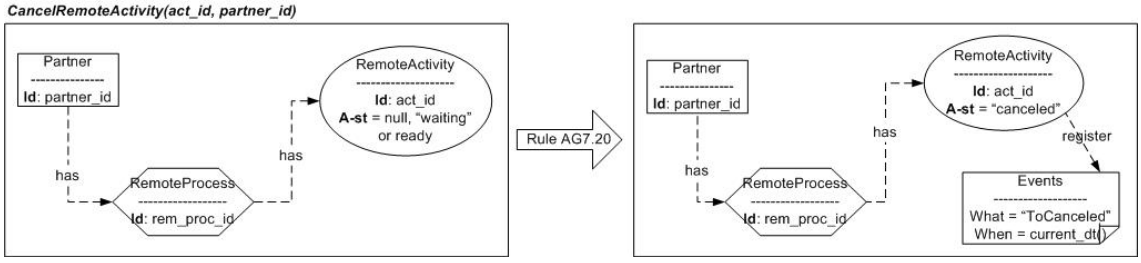

Figura 4.76: Regra de cancelamento de atividades remotas

Quando uma atividade remota é cancelada, todas as atividades locais para as quais ela está mapeada também devem ser. Essa propagação se dá pela regra AG7.21.

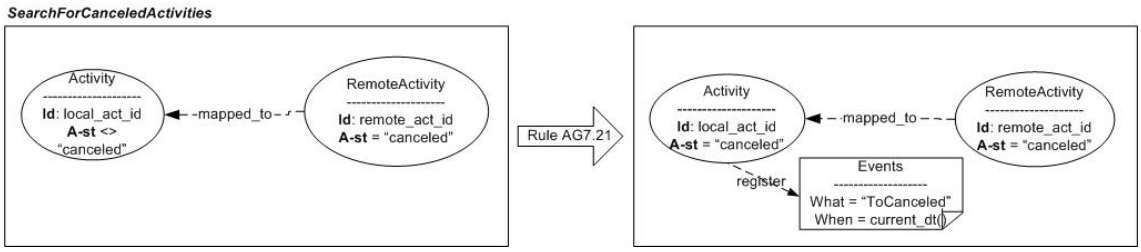

Figura 4.77: Regra de cancelamento de atividades locais

4.3 Considerações

Este capítulo apresentou os formalismos Prosoft-Algébrico e Gramática de Grafos, utilizados na especificação do APSEE-*Global*. A especificação formal dos tipos abstratos de dados segundo a notação do Prosoft-Algébrico foi apresentada, bem como as regras de Gramática de Grafos que definem as funcionalidades do APSEE-*Global*. A utilização dos dois formalismos combinados é possível porque os nodos do grafo-tipo correspondem aos tipos de dados especificados em Prosoft-Algébrico.

Vale ressaltar que, embora o uso de métodos formais possibilite a realização de provas e verificações matemáticas acerca de propriedades do modelo, tal característica não constitui objetivo do trabalho aqui apresentado sendo, entretanto, um recurso importante que pode ser aproveitado em trabalhos futuros.

O próximo capítulo aborda a prototipação do APSEE-*Global*. O protótipo construído é produto da implementação das especificações definidas neste capítulo.

5 PROTOTIPAÇÃO DO MODELO APSEE-*GLOBAL*

A especificação apresentada no capítulo anterior, combinando os formalismos Prosoft-Algébrico e Gramática de Grafos, serviu de base para a implementação do protótipo do modelo APSEE-*Global*. O protótipo está integrado ao ambiente Prosoft-Java e estende a implementação do ambiente APSEE. O objetivo da prototipação foi permitir a experimentação do modelo proposto.

A implementação do protótipo descrito neste capítulo contou com o auxílio do Sr.Heribert Schlebbe, da *Fakultät Informatik* da *Universität Stuttgart*. Esse auxílio foi possível graças à cooperação estabelecida entre o grupo Prosoft da Universidade Federal do Rio Grande do Sul e a *Universität Stuttgart*.

Este capítulo descreve o protótipo implementado, e está estruturado como segue: a seção 5.1 apresenta o ambiente Prosoft-Java, no qual o protótipo foi desenvolvido; a seção 5.2 introduz o ambiente APSEE. O protótipo do APSEE-*Global* é descrito na seção 5.3, e a seção 5.4 apresenta algumas considerações do capítulo.

5.1 O ambiente Prosoft-Java

O Prosoft, ambiente integrado de desenvolvimento de software, baseia-se na estratégia *data-driven* de desenvolvimento de software (NUNES, 1992). O principal objetivo do ambiente é estabelecer uma infra-estrutura que apóie o desenvolvimento de software de alta complexidade através da integração de ferramentas escritas segundo o paradigma Prosoft-Algébrico. O ambiente é resultado do esforço cooperativo de estudantes e pesquisadores do PPGC-UFRGS e da *Fakultät Informatik* da *Universität Stuttgart* (Alemanha), sob coordenação do Prof. Dr. Daltro José Nunes.

O ambiente Prosoft foi implementado em diferentes versões, até chegar à sua versão atual, denominada Prosoft-Java (SCHLEBBE, 1997). A primeira versão, monousuária, foi desenvolvida em Solaris-Pascal, como descrito por Nunes (1992). Posteriormente, foi desenvolvido o Prosoft-Distribuído, utilizando as linguagens Pascal e C, segundo descrito em (SCHLEBBE, 1994) e (GRANVILE; SCHLEBBE, 1996). A implementação mais recente do ambiente, desenvolvida em

Java, resultou em um ambiente portável[17], distribuído[18] e cooperativo[19] que integra ferramentas CASE para auxiliar diferentes fases do processo de software.

A Figura 5.1 apresenta a tela principal do ambiente Prosoft-Java. A partir dos menus disponíveis nessa tela, as diversas ferramentas CASE que compõem o ambiente podem ser acessadas. O menu ATO Configuration, que está expandido na Figura, mostra o acesso a ferramentas como o ambiente APSEE (*Apsee*), ferramentas de definição de classes segundo o paradigma Prosoft-Algébrico (*Classe*), trabalho cooperativo (*Cooperation*) e a ferramenta de prototipação algébrica (*ProTool*).

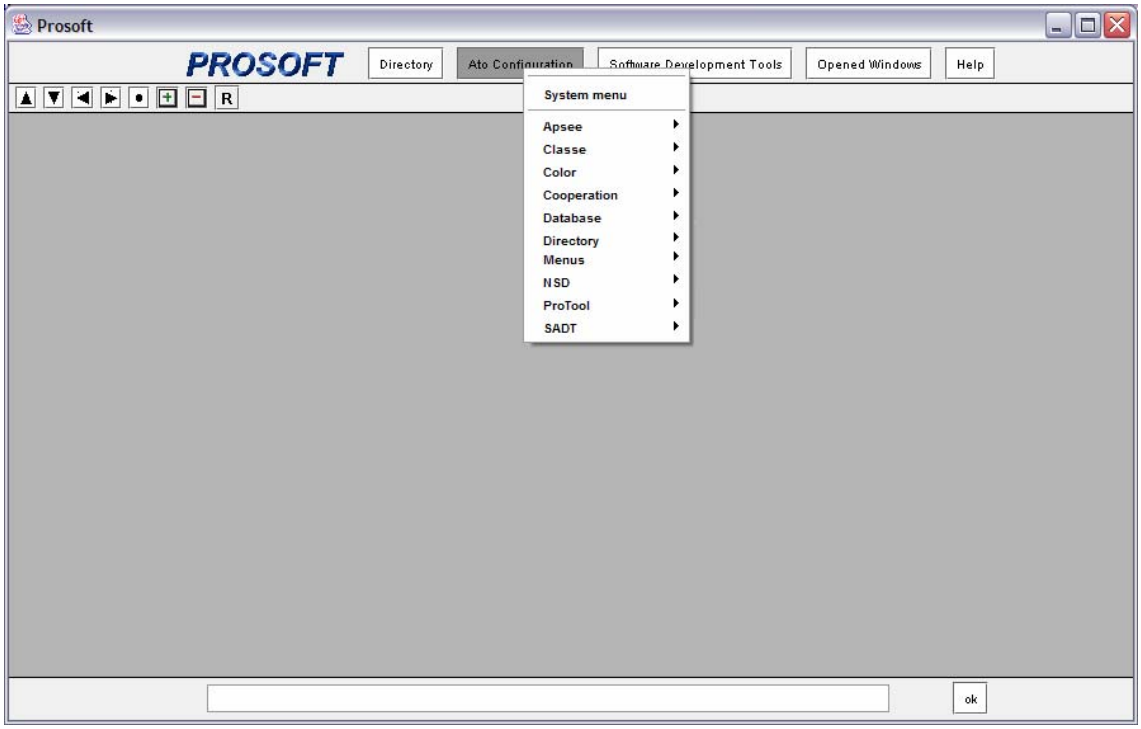

Figura 5.1 – Tela principal do ambiente Prosoft-Java

O paradigma Prosoft sugere que o desenvolvimento completo de um módulo de software siga duas etapas distintas: uma de especificação e outra de implementação de acordo com a especificação (NUNES, 1994). Esta separação é muito útil do ponto de vista organizacional, pois qualquer processo que precise usar um módulo de software (ATO) pode fazê-lo examinando somente sua definição. Não é necessário esperar que o ATO esteja totalmente implementado, nem é necessário compreender sua implementação.

---

[17] A portabilidade deve-se ao fato de que o sistema é executável nas plataformas para as quais está disponível o *Java Runtime Environment* (SUN, 2005-a).

[18] A distribuição do ambiente, que permite que os objetos estejam armazenados em diferentes *hosts*, é fornecida pelo mecanismo *ad-hoc* de comunicação denominado ICS-Distribuído que está implementado sobre o protocolo *Java Remote Method Invocation* (SUN, 2005-b).

[19] O suporte ao trabalho cooperativo foi proposto por (REIS, 1998), por meio de uma extensão denominada ProsoftCooperativo, que permite o gerenciamento de versões de artefatos compartilhados.

Existe uma correspondência entre ATOs algébricos e ATOs Java, como ilustra o exemplo da Figura 5.2. Esta figura mostra que o conceito de Tipo Abstrato de Dado do Prosoft-Algébrico é diretamente mapeado para a implementação em Prosoft-Java, onde os axiomas algébricos correspondem aos métodos Java.

O ambiente Prosoft possui uma ferramenta, denominada ATO-Classe, para edição das classes algébricas que também gera código-fonte Java para implementá-las. Na versão atual desse gerador de código, as operações básicas de inclusão e exclusão são geradas automaticamente. As demais operações, especificadas algebricamente no ATO, devem ser traduzidas manualmente para métodos escritos de acordo com a sintaxe da linguagem Java (SUN, 2005-c), podendo utilizar os construtores disponíveis pelo pacote *prosoft.kernel*. Informações acerca do desenvolvimento de ATOs-Java podem ser encontradas em (SCHLEBBE; SCHIMPF, 1997) (SCHLEBBE, 2005).

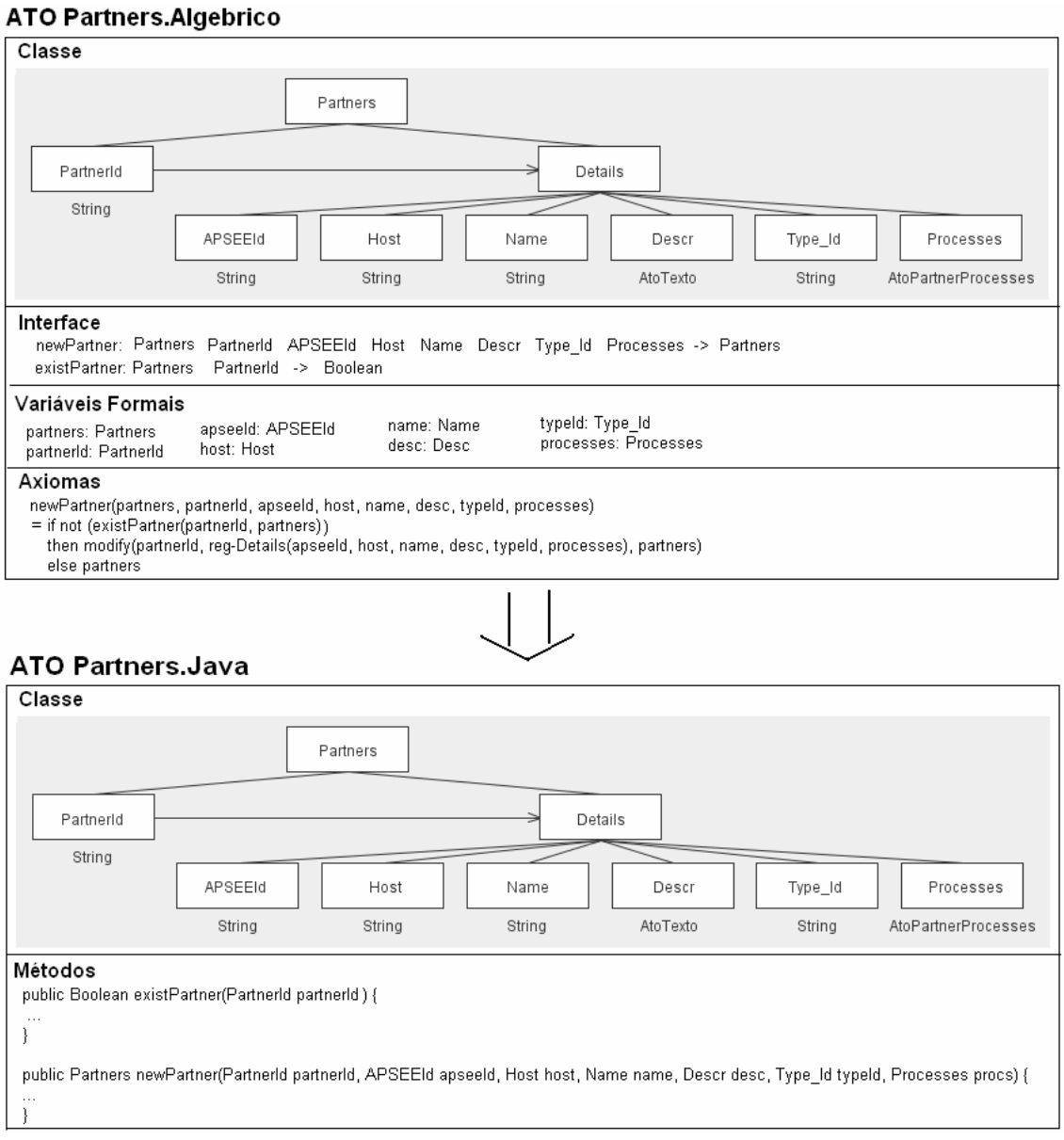

Figura 5.2 – Correspondência entre ATOs algébricos e ATOs Java

5.2 O ambiente APSEE

O ambiente APSEE reúne um conjunto de ferramentas desenvolvidas segundo o paradigma Prosoft, que disponibilizam serviços de definição, visualização, execução e reutilização de processos de software. Os principais componentes do APSEE foram apresentados no capítulo 3, que define também a relação entre os componentes. O APSEE está completamente integrado ao ambiente Prosoft-Java, e pode ser acessado a partir do menu principal do Prosoft-Java, como pode ser observado na Figura 5.1.

A definição de processos de software no APSEE é realizada no editor de processos, ilustrado na Figura 5.3. O editor permite a definição gráfica de atividades e conexões que compõem o processo de software. No exemplo apresentado, o fragmento de processo *Planejamento* é composto por quatro atividades decompostas (*Planejamento do projeto*, *Planejamento da gerência de riscos*, *Planejamento da gerência de configuração* e *Definição de estimativas*) e duas atividades normais (*Definição do cronograma* e *Criação do website do projeto*). A conexão *branch* AND *start-start* indica que as atividades *Planejamento da gerência de riscos* e *Planejamento da gerência de configuração* somente podem ser iniciadas após o início da execução da atividade *Planejamento do projeto*.

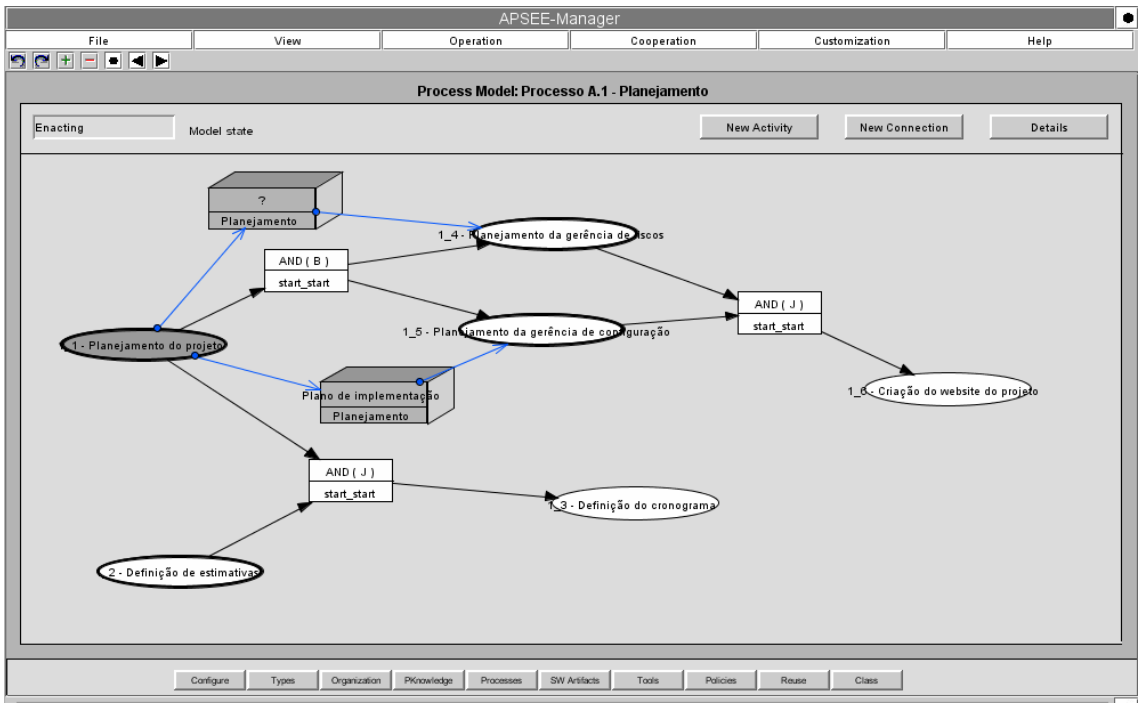

Figura 5.3 – O editor de processos do APSEE

A ferramenta de edição de processos também fornece informações sobre a dinâmica de execução de processos. Uma das maneiras de visualizar o andamento da execução é por meio da mudança de cor das atividades de acordo com o seu estado. Por exemplo, na Figura 5.3 a execução da atividade *Planejamento do projeto* foi finalizada. Outras informações sobre a execução, como data de início e término, podem ser visualizadas na tela de detalhes das atividades. Essa tela é

acessível a partir de menu exibido ao clicar-se com o botão direito do mouse sobre a atividade.

Do ponto de vista dos desenvolvedores, as informações são disponibilizadas por meio na agenda de atividades. Nessa agenda o desenvolvedor pode visualizar o estado e os detalhes das atividades sob sua responsabilidade, além de poder interagir com o APSEE, solicitando mudanças de estado. Como ilustra o exemplo da Figura 5.4, o desenvolvedor pode informar o início, o término ou a pausa na execução de uma atividade, bem como pode delegá-la para outro desenvolvedor da equipe. As mudanças de estado informadas na agenda serão refletidas no editor de processos, com alteração nas cores das atividades.

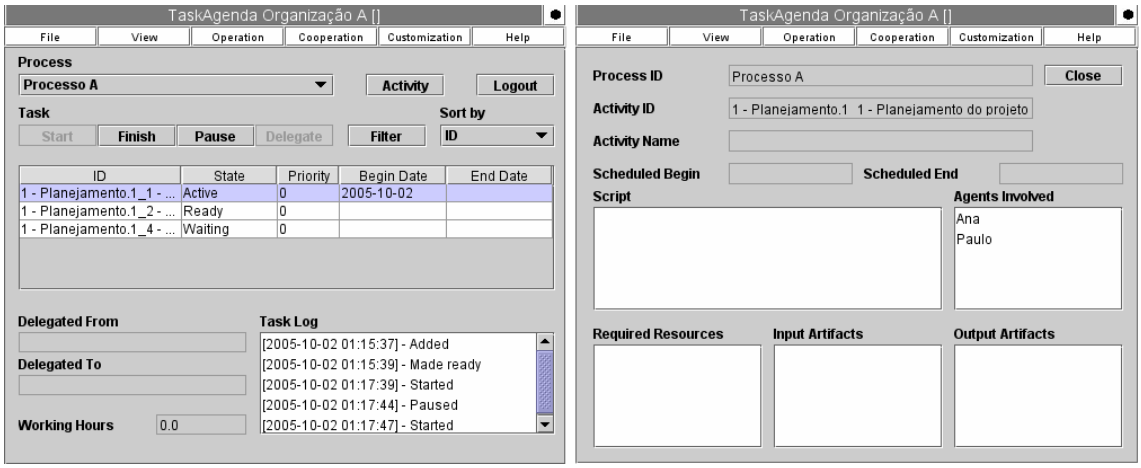

Figura 5.4 – Agenda de atividades do APSEE

5.3 O protótipo APSEE-*Global*

Essa seção descreve o protótipo do APSEE-*Global*. As subseções 5.3.1 e 5.3.2 mostram a integração com os ambientes Prosoft-Java e APSEE, respectivamente. Telas que implementam as funcionalidades do APSEE-*Global* são apresentadas nas subseções 5.3.3 a 5.3.6.

5.3.1 Integração do APSEE-*Global* com o Prosoft-Java

Como ilustra a Figura 5.5, o APSEE-*Global* está completamente integrado ao Prosoft-Java, e é acessado a partir do menu principal do ambiente. Conforme apresentado no Capítulo 4, os TADs do APSEE-*Global* foram definidos segundo a notação do Prosoft-Algébrico. Esses TADs deram origem às interfaces do protótipo.

Por estar integrado ao Prosoft-Java, o APSEE-*Global* faz uso de serviços oferecidos pelo ambiente. Por exemplo, o recurso de distribuição de ATOs é utilizado para que instâncias do APSEE-*Global* executando em diferentes servidores possam comunicar-se.

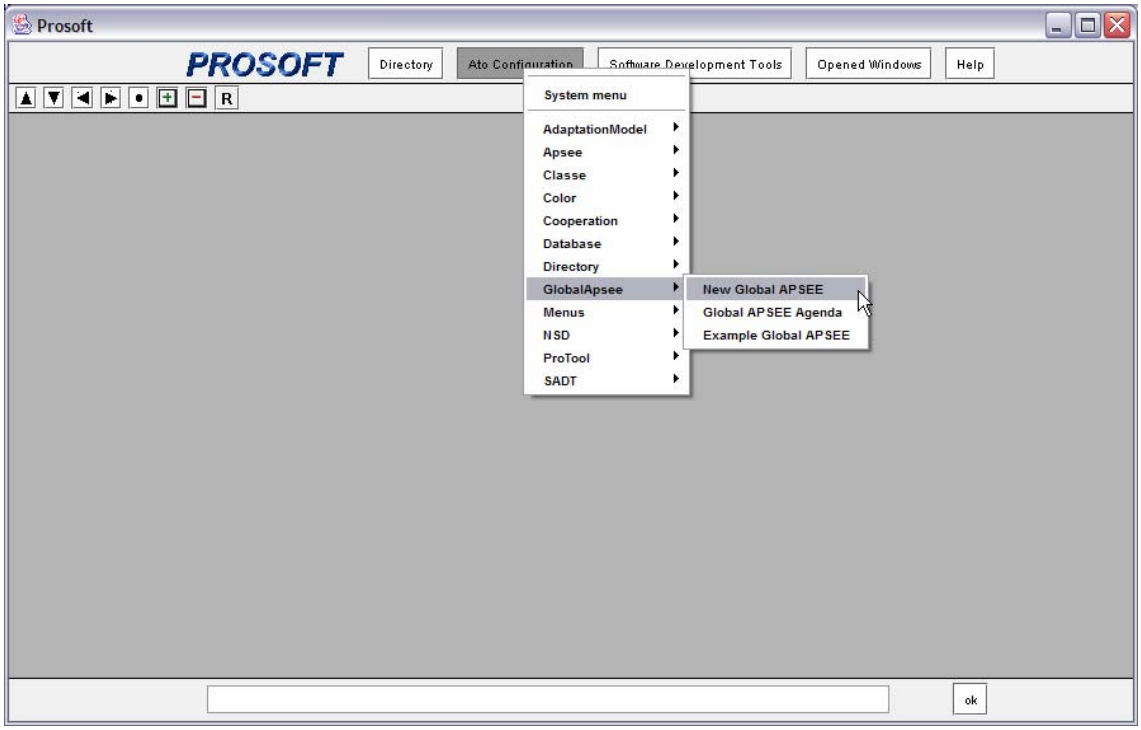

Figura 5.5 – Chamada ao APSEE-*Global* a partir do ambiente Prosoft-Java

5.3.2 Integração do APSEE-*Global* com o ambiente APSEE

O APSEE-*Global* estende o ambiente APSEE, incluindo funcionalidades para distribuição de processos. Como mostra a Figura 5.6, o APSEE-*Global* contém, além de todas as funcionalidades do APSEE, o item *Distribution*, que dá acesso às opções de distribuição (*Partners* e *Processes Distribution*).

Figura 5.6 – Tela inicial do ambiente APSEE-*Global*

5.3.3 Definição de parceiros de projeto

A definição de parceiros de projeto na instância do APSEE-*Global* executada por uma equipe permite que processos e atividades definidos localmente sejam delegados posteriormente. Em contrapartida, parceiros de projeto podem delegar atividades à equipe.

A Figura 5.7 ilustra a tela de definição de parceiros de projeto do APSEE-*Global*, acessível a partir da opção *Partners*, no menu *Distribution* do APSEE-*Global* (Figura 5.6). As informações de *APSEE ID* e *Host* do parceiro permitem que a comunicação entre as instâncias do APSEE-*Global* se estabeleça. Essa comunicação é definida por meio do serviço oferecido pela ICS-Distribuída de comunicação entre ATOs hospedados em diferentes servidores.

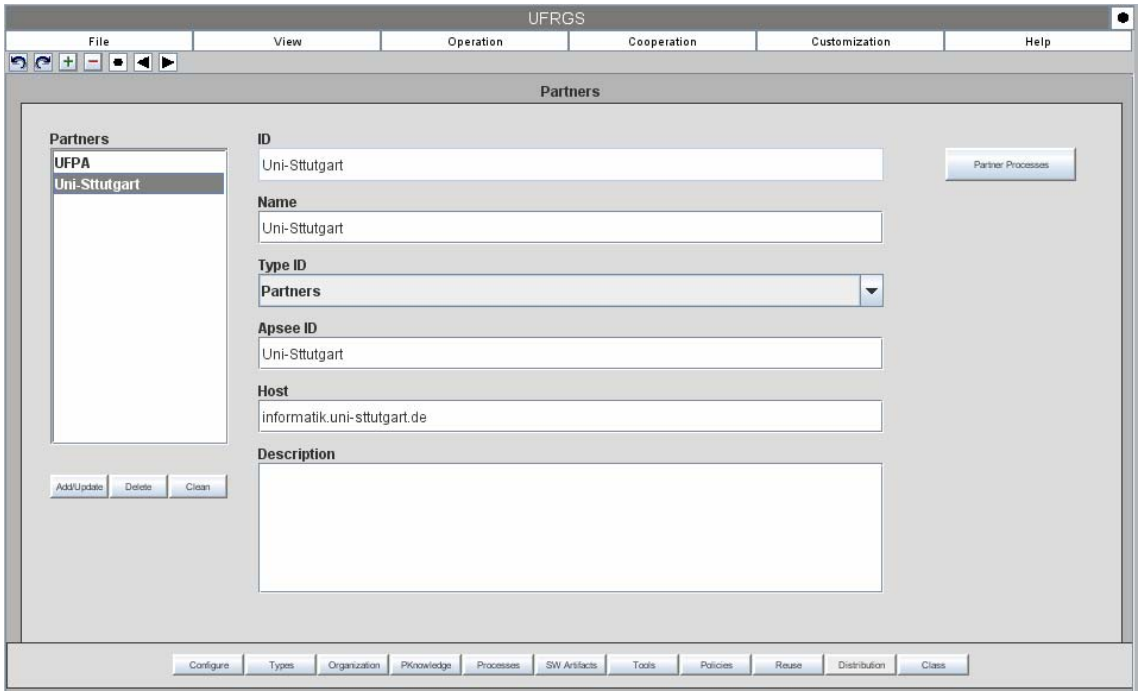

Figura 5.7 – Tela de definição de parceiros de projeto

5.3.4 Distribuição de processos

A segunda opção do menu *Distribution* do APSEE-*Global* (Figura 5.6) é *Process Distribution*. Essa opção permite a delegação de processos a parceiros de projeto. A tela de delegação de processos, ilustrada na Figura 5.8, lista à esquerda todos os processos da organização. Ao selecionar um dos processos, pode-se atribuí-lo a um ou mais parceiros de projeto. A condição para que a atribuição ocorra é que os parceiros selecionados também tenham a organização como parceira.

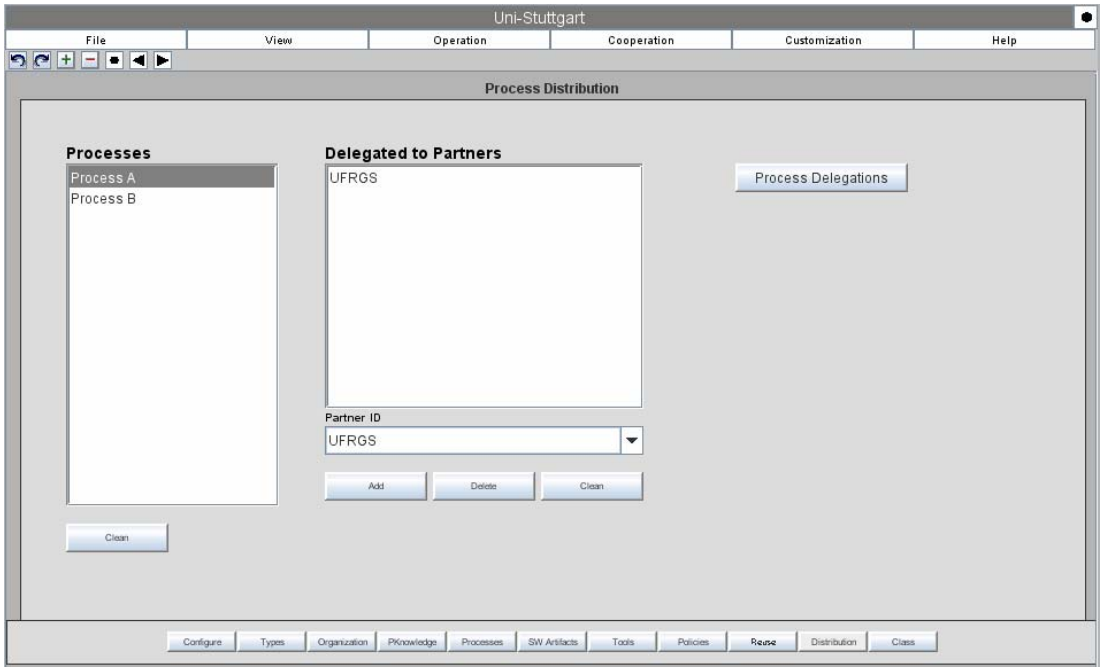

Figura 5.8 – Tela de delegação de processos a parceiros de projeto

5.3.5 Delegação de atividades

A definição de quais atividades serão delegadas a cada parceiro pode ser feita ao selecionar-se um processo local e um parceiro, clicando em seguida no botão *Process Delegations*, localizado à direita da tela de delegação de processos (Figura 5.8). A tela de delegação de atividades, ilustrada na Figura 5.9, é então exibida. No lado direito, as atividades que pertencem ao processo selecionado são listadas. Ao selecionar uma das atividades, pode-se atribuí-la a um ou mais parceiros de projeto. Essa atribuição é realizada na parte inferior da tela. A parte superior exibe o modelo de processo que está sendo delegado.

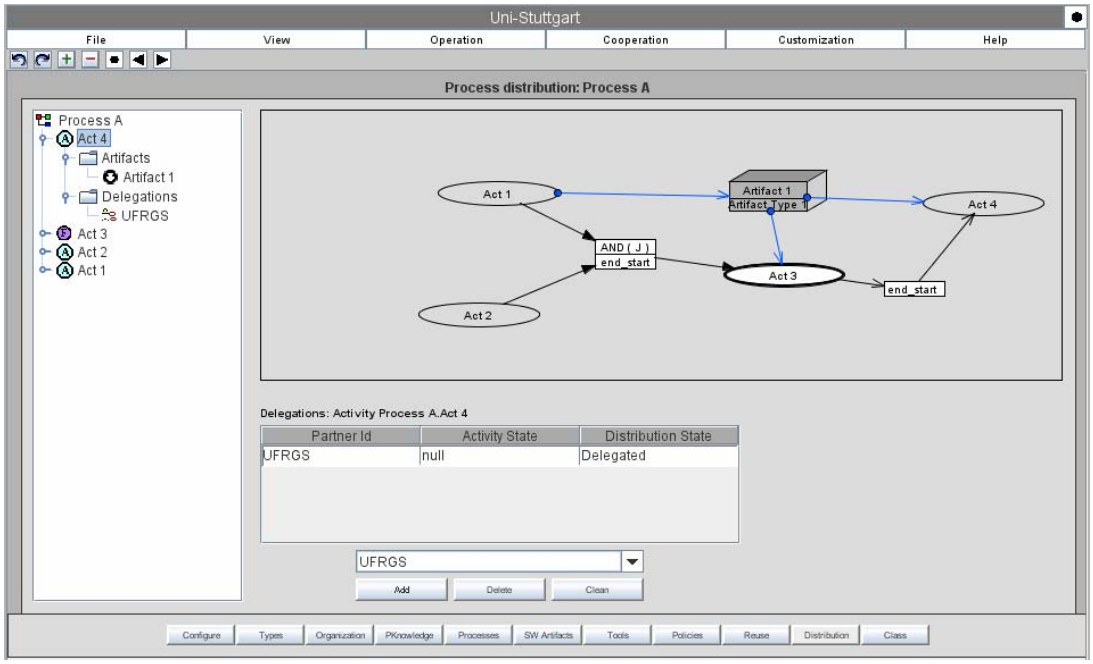

Figura 5.9 – Tela de delegação de atividades a parceiros de projeto

5.3.6 Mapeamento de processos e atividades remotas

Parceiros de projeto podem delegar processos e atividades. Ao clicar no botão *Partner Processes* da tela de definição de parceiros (Figura 5.7) a tela de mapeamento de processos e atividades remotas é exibida. No lado esquerdo da tela, mostrada na Figura 5.10, uma árvore contendo todos os processos e atividades delegadas pelo parceiro é exibida. Ao selecionar um processo, a parte superior direita da tela exibe uma representação gráfica do fragmento do processo delegado que contém as atividades delegadas à organização. Logo abaixo, pode-se selecionar, dentre os processos da organização, qual será mapeado pelo processo remoto.

Uma vez que o processo remoto é mapeado para um processo local, as atividades remotas também podem ser mapeadas. Ao selecionar uma atividade na árvore de delegações, pode-se selecionar quais as atividades locais que a representam. Essa seleção é feita no canto inferior direito da tela, onde as atividades pertencentes ao processo local são listadas.

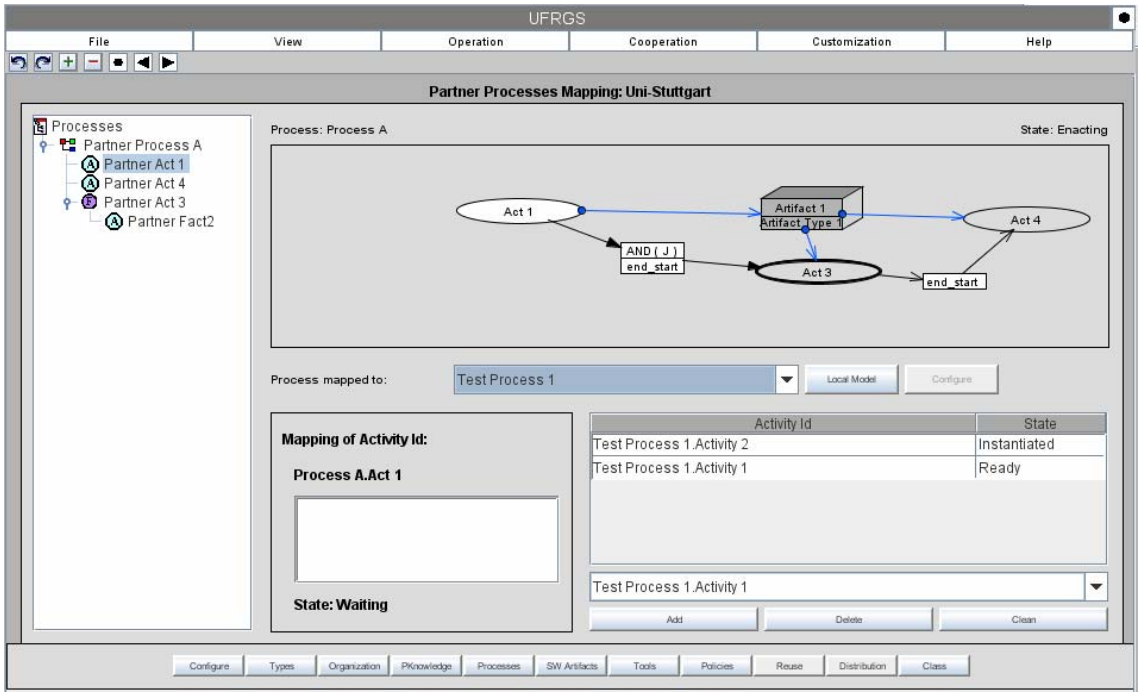

Figura 5.10 – Tela de mapeamento de atividades delegadas

5.4 Considerações

Este capítulo apresentou a prototipação das principais funcionalidades do APSEE-*Global*. O intuito da prototipação foi construir uma ferramenta que viabilize a avaliação do modelo proposto.

No próximo capítulo é apresentado um exemplo que ilustra a utilização da ferramenta construída.

6 EXEMPLO DE APLICAÇÃO DO MODELO

Esse capítulo descreve o uso prático do protótipo desenvolvido em um cenário real de processo distribuído. O objetivo da elaboração desse exemplo é demonstrar o uso do protótipo pra gerência de um processo distribuído. O cenário escolhido reflete a experiência no desenvolvimento do Sistema de Informações Gerenciais do Programa Nacional de Crédito Fundiário (SIG/CF).

O projeto de desenvolvimento foi acompanhado durante os meses de fevereiro a setembro de 2005. O desenvolvimento do SIG/CF envolveu quatro organizações de software distintas, distribuídas entre Recife (PE) e Brasília (DF).

A estrutura do capítulo é como segue: a seção 1 caracteriza o cenário escolhido. A seção 2 apresenta a justificativa para escolha do cenário. A definição do modelo de processo distribuído é apresentada na seção 3. A seção 4 apresenta a modelagem da distribuição do processo. Aspectos da sincronização na execução dos processos são tratados na seção 5. A seção 6 apresenta uma análise dos resultados obtidos com a utilização do protótipo. E, por fim, a seção 7 apresenta as considerações do capítulo.

6.1 Contextualização

O projeto utilizado como exemplo é o desenvolvimento do Sistema de Informações Gerenciais do Programa Nacional de Crédito Fundiário. No início do estudo, o Programa Nacional de Crédito Fundiário já possuía alguns sistemas de gerenciamento de informações implantados. No entanto, esses sistemas eram independentes, o que dificultava a análise gerencial das informações. O projeto consistiu na reestruturação dos sistemas existentes, para integrá-los como módulos do SIG/CF, e no desenvolvimento de novos módulos. As subseções a seguir contextualizam o estudo de caso realizado, apresentando uma visão geral do Programa Nacional de Crédito Fundiário e do SIG/CF.

6.1.1 O Programa Nacional de Crédito Fundiário

O PNCF – Programa Nacional de Crédito Fundiário – tem como objetivo financiar a aquisição de imóveis rurais a associações de agricultores sem terra. O PNCF faz parte do Plano Nacional de Reforma Agrária do Ministério do Desenvolvimento Agrário e está vinculado à Secretaria de Reordenamento Agrário.

O PNCF é executado de forma descentralizada, em parceria com os governos estaduais e com o movimento sindical de trabalhadores rurais e da agricultura familiar e conta com a participação dos Conselhos Municipais e Estaduais de Desenvolvimento Rural Sustentável. A função dos órgãos parceiros é identificar beneficiários potenciais do Programa, auxiliando-os inclusive a encontrar terra para aquisição, a elaborar a proposta de financiamento e o projeto produtivo. Em cada Estado existe um órgão executor do PNCF, denominado Unidade Técnica Estadual (UTE). A UTE apóia todo o trabalho, intermedia a negociação de preço entre a associação e o proprietário do imóvel, concede o financiamento e monitora a execução dos projetos produtivos.

Além do financiamento para aquisição de imóveis, são também financiados os investimentos em infra-estrutura básica (casas, energia elétrica, rede de abastecimento de água, estradas), e comunitária (assistência técnica, investimentos iniciais na produção). Esses recursos são liberados em parcelas, ao longo da execução dos projetos produtivos.

6.1.2 Sistema de Informações Gerenciais do PNCF

O objetivo do Sistema de Informações Gerenciais do PNCF é acompanhar todas as etapas dos financiamentos, provendo meios para captura, armazenamento e gerenciamento das informações. Quando a análise do cenário começou a ser realizada, o Programa Nacional de Crédito Fundiário contava com um conjunto de sistemas, descritos brevemente a seguir:

- **Sistema de Qualificação da Demanda**: Esse sistema permite que os órgãos parceiros do Programa realizem o cadastro inicial das propostas de financiamento. As possíveis demandas de financiamento são então identificadas, e enviadas posteriormente para análise da Unidade Técnica Estadual;
- **Sistema de Análise e Contratação**: Nesse sistema a análise das propostas é realizada pela UTE. Após aprovação, as propostas são encaminhadas aos agentes financeiros (bancos) para contratação;
- **Sistema de Registro de Contratação**: Esse sistema é utilizado pelos agentes financeiros, para registro das informações sobre a assinatura do contrato de financiamento;
- **Sistema Financeiro**: Esse sistema registra a liberação das parcelas relativas aos financiamentos para investimentos de infra-estrutura básica e comunitária.

O nível de integração entre os sistemas era bastante reduzido. De fato, os quatro sistemas foram concebidos de forma independente. A integração deu-se, portanto, de forma pontual, na medida em que as necessidades iam sendo detectadas.

O projeto estudado consistiu no desenvolvimento de um sistema que contemple as funcionalidades existentes nos sistemas já implantados e inclua um novo módulo, de **monitoramento de projetos**. O foco desse módulo é acompanhar a

evolução dos agricultores após a aquisição do imóvel. Ele permite o planejamento do desenvolvimento da comunidade e a concretização do projeto produtivo, envolvendo a elaboração de sub-projetos, que visam a liberação de parcelas do financiamento para infra-estrutura básica e comunitária.

6.1.3 Equipes envolvidas

O projeto contou com a participação de quatro equipes: uma equipe do PNCF e três equipes pertencem a organizações desenvolvedoras de software. A coordenação do projeto foi realizada pela equipe do PNCF, representada pela Gerência de Sistemas de Informação da Secretaria do Reordenamento Agrário (GSI), com sede em Brasília.

A relação da GSI com as demais equipes baseou-se no conceito de *inshore outsourcing* (CARMEL; AGARWAL, 2001): por meio de processo licitatório, foram contratadas organizações desenvolvedoras de software para realização das atividades do projeto. As seguintes organizações participaram do projeto:

- Organização A[20]: Organização com sede em Recife e experiência no desenvolvimento de sistemas de acompanhamento de projetos. A Organização A é uma empresa com CMM nível 2, e possui metodologia já validada por projetos anteriores.
- Organização B: A principal área de atuação da organização B é o desenvolvimento de sistemas financeiros, contando inclusive com profissionais que participaram do desenvolvimento do sistema financeiro do SIG/CF. A organização é sediada em Brasília. A metodologia adotada baseia-se no *Unified Process*.
- Organização C: Sediada em Brasília, possui experiência no desenvolvimento de sistemas com bases de dados estatísticas. A metodologia adotada baseia-se no *MSF Process Model*.

6.2 Justificativa para escolha do cenário

Diversos fatores motivaram a escolha do cenário:

- A distribuição do cenário, que envolve equipes distintas e autônomas no desenvolvimento do sistema;
- A complexidade do processo, uma vez que o Sistema de Informações Gerenciais envolve uma série de módulos interdependentes, alguns deles tendo como base sistemas legados;
- O fato de cada organização envolvida adotar uma metodologia de desenvolvimento de sistemas distinta;

[20] O nome real das organizações foi omitido neste trabalho.

- Os ganhos obtidos na definição do exemplo pelo acompanhamento de um cenário real, onde os processos estão em andamento e se tem condição de observar o impacto real da adoção do modelo.

6.3 O processo de desenvolvimento do SIG/CF

Com base no contexto apresentado, foram identificadas as principais atividades do projeto, para que fosse possível definir o modelo de processo global e para que fossem estabelecidos os critérios adotados para distribuição das atividades.

O desenvolvimento do Sistema de Informações Gerenciais do PNCF envolve a construção de três módulos: Contratação de Propostas (CONTRATO), Financeiro (FINANCEIRO) e Monitoramento de Projetos (PROJETOS).

A definição do modelo de processo do SIG/CF reflete a metodologia de desenvolvimento de sistemas adotada pela Gerência de Sistemas de Informação do da Secretaria do Reordenamento Agrário (GSI). Essa metodologia é composta pelas seguintes fases: Levantamento de Requisitos, Modelagem, Prototipação, Desenvolvimento, Testes, Homologação e Implantação.

A Figura 6.1 mostra o modelo de processos do SIG/CF, definido segundo a notação da APSEE-PML. Vale salientar que o objetivo desse modelo é definir *o que* deve ser executado, sem detalhar *como*. Assim, as atividades de Prototipação, Desenvolvimento e Homologação não foram detalhadas.

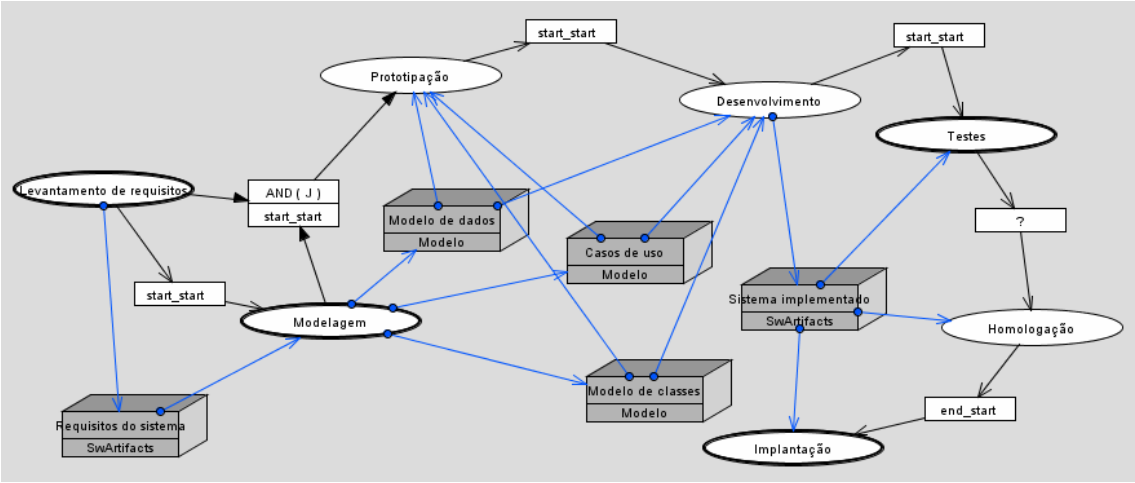

Figura 6.1: Modelo de processo da GSI

O detalhamento das atividades Levantamento de requisitos, Modelagem, Testes e Implantação é apresentado a seguir.

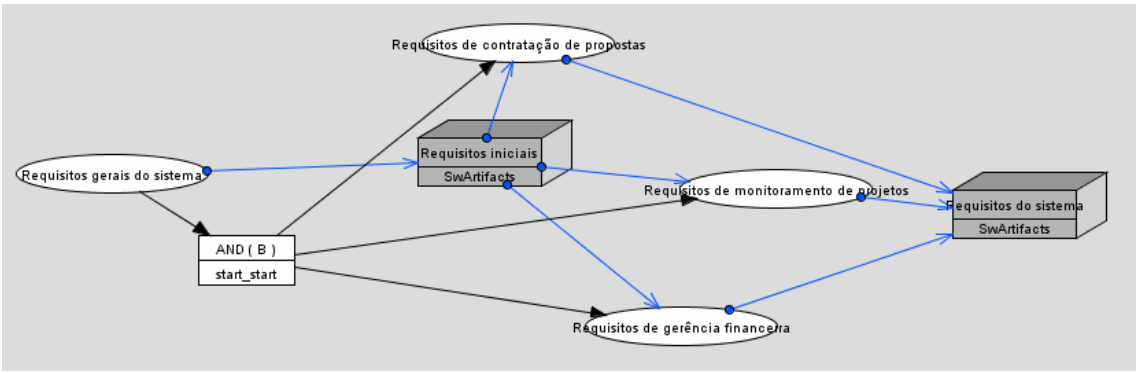

Figura 6.2: Detalhamento da atividade Levantamento de requisitos

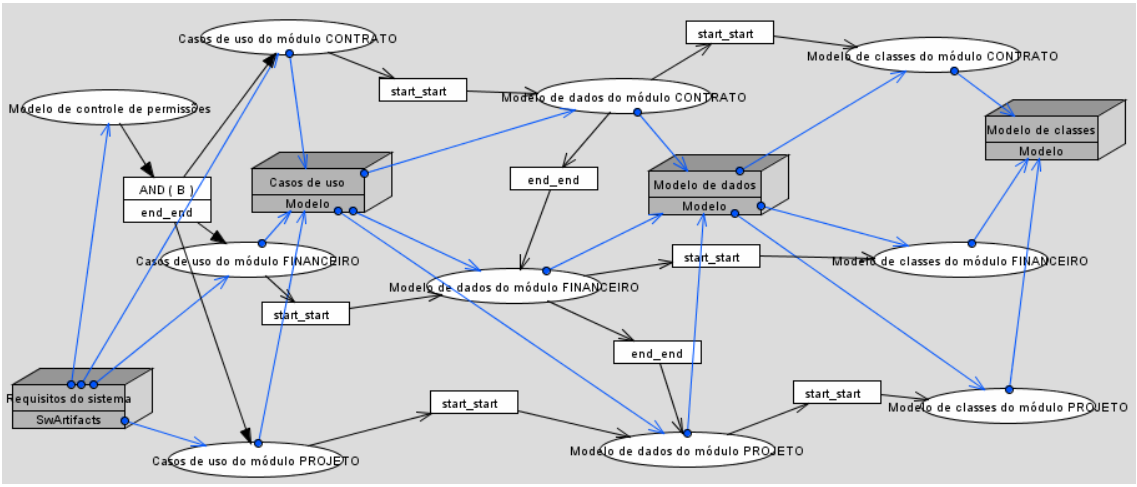

Figura 6.3: Detalhamento da atividade Modelagem

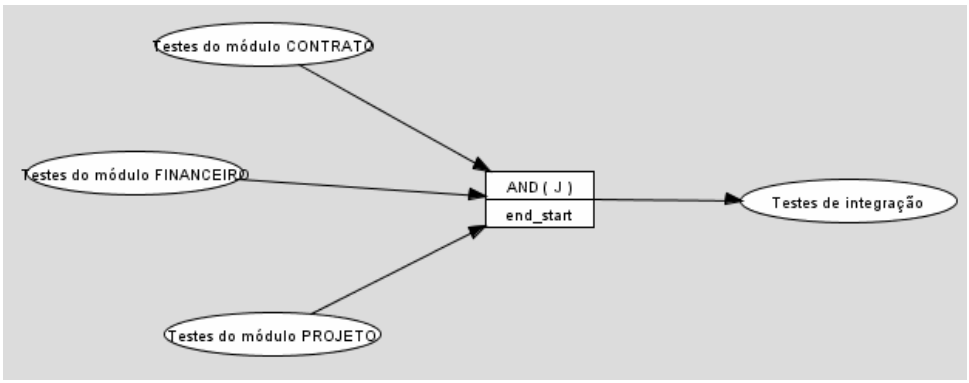

Figura 6.4: Detalhamento da atividade Testes

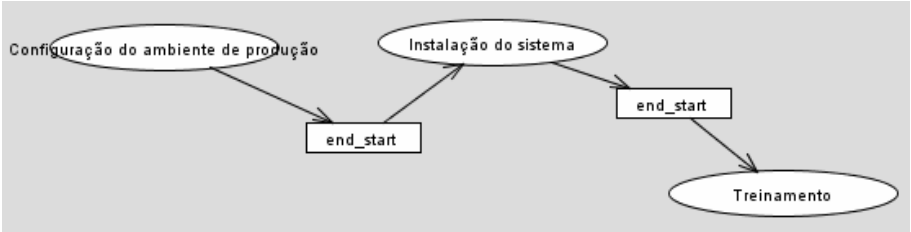

Figura 6.5: Detalhamento da atividade Implantação

6.4 Modelagem da distribuição do processo

O desenvolvimento do SIG/CF envolveu a participação de um conjunto de organizações desenvolvedoras de software. Essa seção apresenta os critérios adotados para a distribuição das atividades. A seção também apresenta a distribuição do modelo de processo do SIG/CF, utilizando o ambiente APSEE-*Global*.

6.4.1 Critérios para distribuição do processo

A divisão das atividades foi obtida pela combinação de um conjunto de abordagens diferentes de distribuição. A principal abordagem adotada foi a divisão com base nas funcionalidades do sistema, como sugerem Mockus e Weiss (2001). Essa abordagem levou à divisão do SIG/CF em módulos, cada um desenvolvido por uma organização distinta. A base de conhecimento de cada organização foi considerada na divisão, distribuindo os módulos de acordo com o portfólio das organizações. O resultado da divisão é apresentado na Tabela 6.1.

Tabela 6.1: Divisão dos módulos do SIG/CF entre os parceiros

Módulo	Parceiros relacionados
Contratação de Propostas	Organização C
Gestão Financeira	Organização B
Monitoramento de Projetos	Organização A

As relações comerciais, mencionadas por Karolak (1998), também foram levadas em consideração: a GSI assumiu os papéis de cliente e de gerente do projeto, com a função de coordenar as atividades das diversas organizações.

6.4.2 Distribuição do processo

O processo de desenvolvimento do SIG/CF envolveu a participação da GSI e das organizações A, B e C. A GSI atuou como mediadora e coordenadora do processo: foi responsável pela distribuição das atividades e por intermediar a troca de informações entre as organizações. Por esta razão, a GSI definiu como parceiros de projeto todas as demais organizações envolvidas, como mostra a Figura 6.6.

Embora o trabalho cooperativo entre as organizações se tenha feito necessário em alguns momentos, a relação direta entre elas não foi definida no modelo de processo: o trabalho cooperativo aparece pela delegação da mesma atividade a mais de um parceiro. Assim, cada organização definiu apenas a GSI como parceira de projeto.

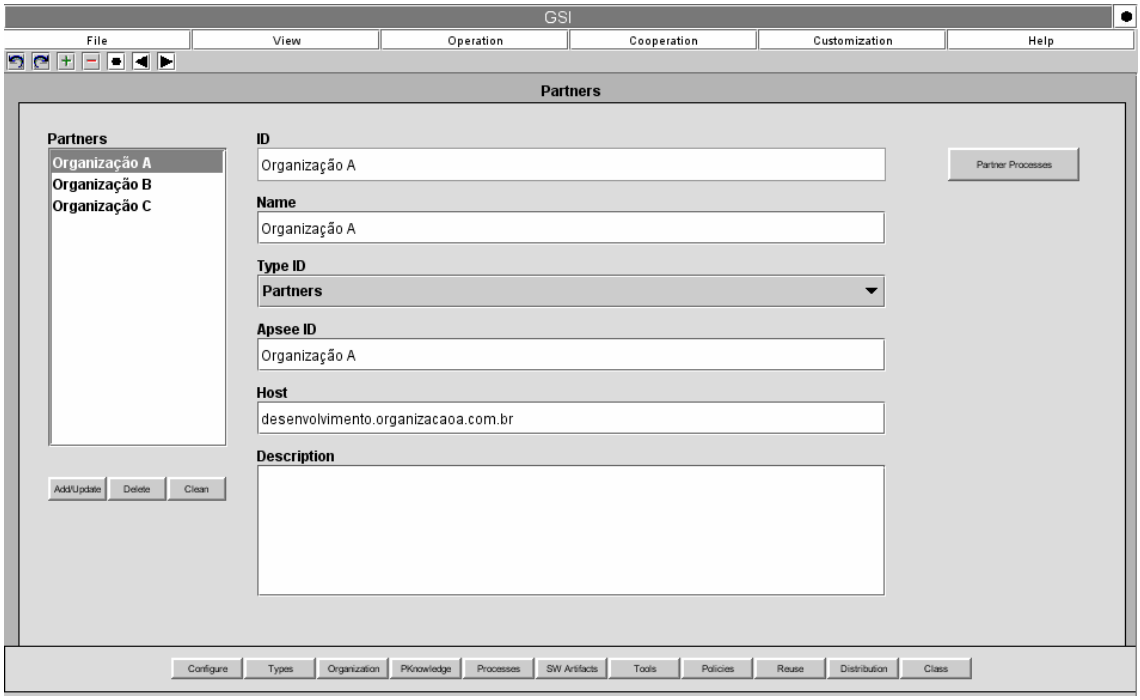

Figura 6.6: Definição dos parceiros de projeto da GSI

Uma vez tendo definido as organizações como parceiras de projeto, a GSI pôde delegar o processo de desenvolvimento do SIG/CF ao conjunto de parceiros, como mostra a Figura 6.7.

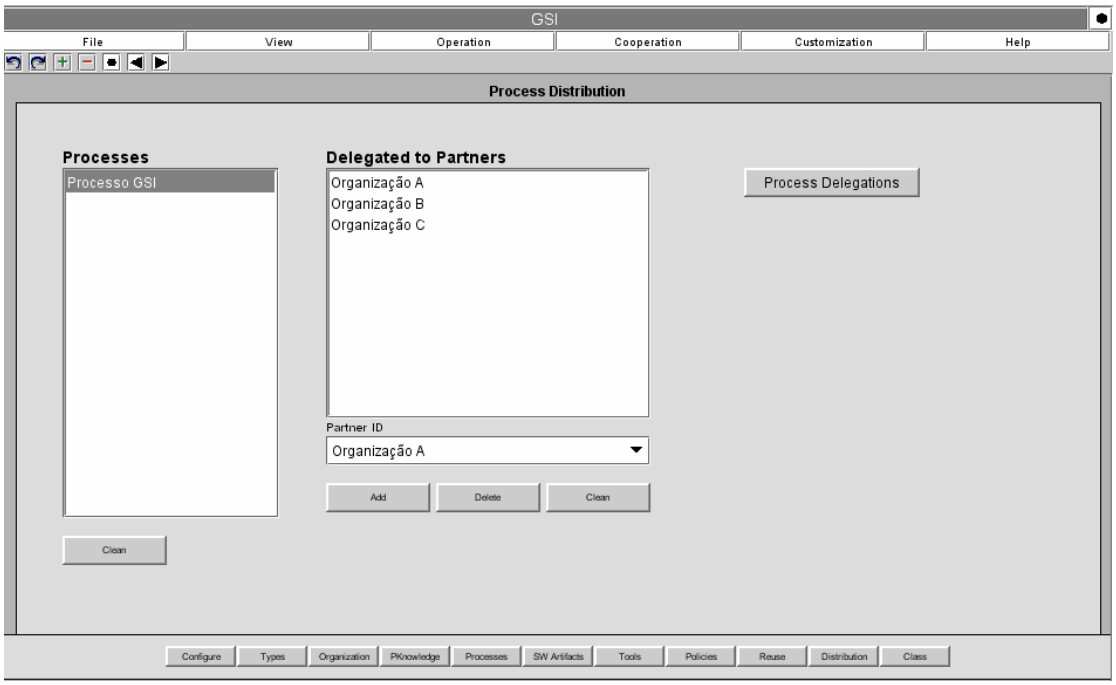

Figura 6.7: Delegação do processo SIG/CF aos parceiros de projeto

6.4.3 Delegação de atividades

A delegação de atividades aos parceiros define as interações entre as organizações participantes do projeto. A Figura 6.8 apresenta um exemplo: a

atividade Prototipação é delegada aos parceiros Organização A, Organização B e Organização C.

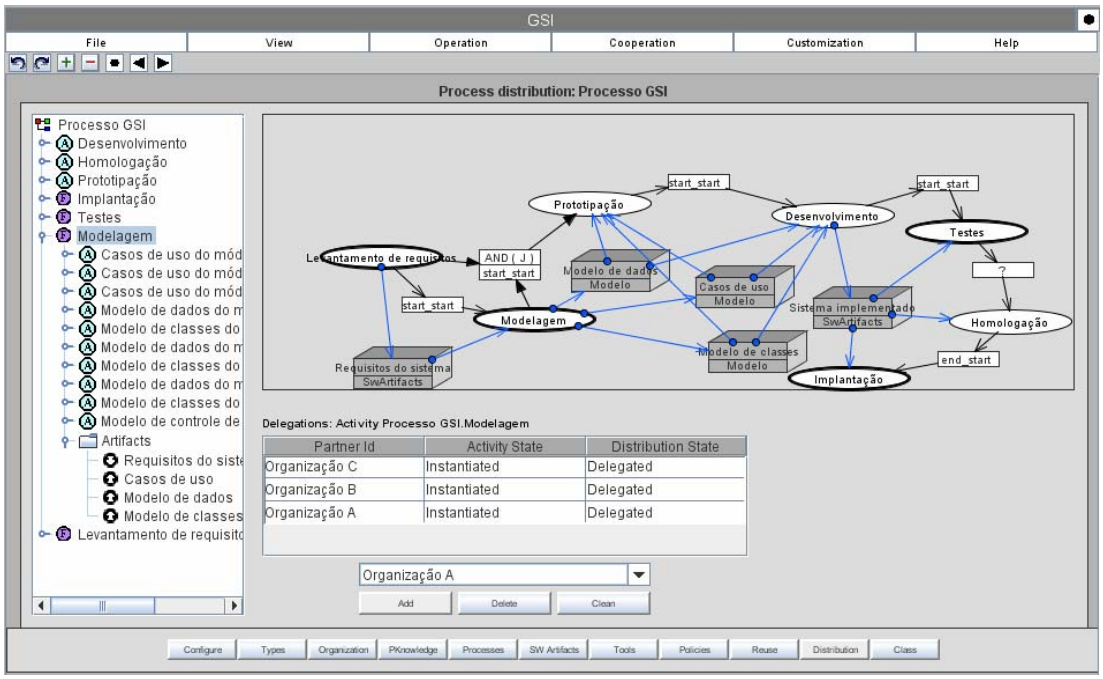

Figura 6.8: Delegação da atividade Modelagem

A Tabela 6.2 apresenta as delegações das atividades do modelo de processo da GSI, detalhando quais os responsáveis pela execução de cada atividade.

Tabela 6.2: Delegação das atividades do modelo de processo da GSI

Atividade delegada	Parceiros
1 Levantamento de Requisitos	
1.1 Requisitos gerais do sistema	GSI
1.2 Requisitos de contratação de propostas	Organização C
1.3 Requisitos de gerência financeira	Organização B
1.4 Requisitos de monitoramento de projetos	Organização A
2 Modelagem	
2.1 Modelo de controle de permissões	GSI
2.2 Casos de uso do módulo CONTRATO	Organização C
2.3 Casos de uso do módulo FINANCEIRO	Organização B
2.4 Casos de uso do módulo PROJETO	Organização A
2.5 Modelo de dados do módulo CONTRATO	Organização C
2.6 Modelo de dados do módulo FINANCEIRO	Organização B
2.7 Modelo de dados do módulo PROJETO	Organização A
2.8 Modelo de classes do módulo CONTRATO	Organização C

2.9	Modelo de classes do módulo FINANCEIRO	Organização B
2.10	Modelo de classes do módulo PROJETO	Organização A
3	Prototipação	Organização A Organização B Organização C
4	Desenvolvimento	Organização A Organização B Organização C
5	*Testes*	
5.1	Testes do módulo CONTRATO	Organização C
5.2	Testes do módulo FINANCEIRO	Organização B
5.3	Testes do módulo PROJETO	Organização A
5.4	Testes de integração	GSI
6	Homologação	GSI
7	*Implantação*	
7.1	Configuração do ambiente de produção	GSI
7.2	Instalação do sistema	GSI
7.3	Treinamento	GSI

6.4.4 Mapeamento do processo remoto

A partir da distribuição do processo de desenvolvimento da GSI, as organizações contratadas realizaram o mapeamento entre os seus processos de desenvolvimento e o processo remoto. A seguir são apresentados os processos de desenvolvimento adotados pelas organizações parceiras, para os quais o processo distribuído foi mapeado.

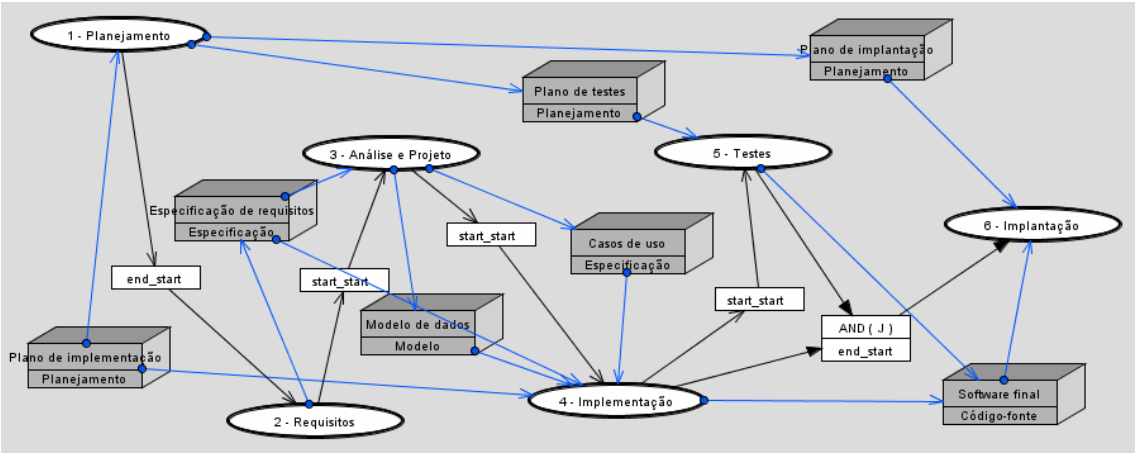

Figura 6.9: Modelo de processo da Organização A

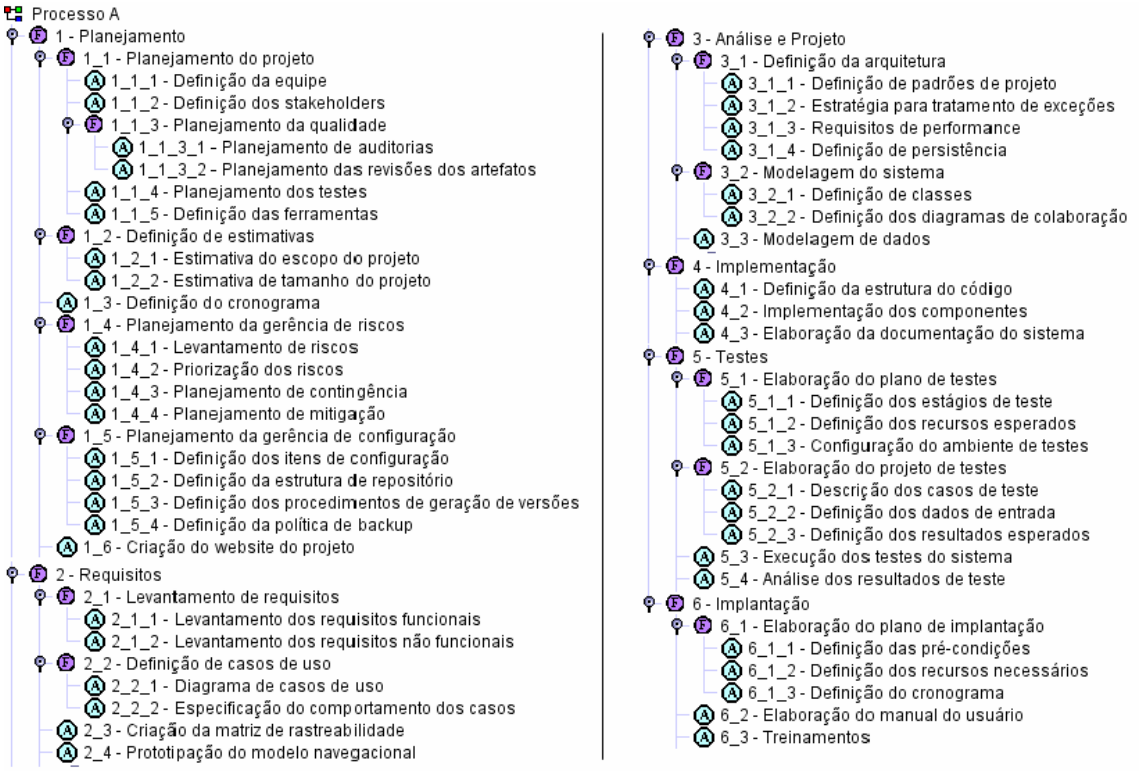

Figura 6.10: Estrutura hierárquica de atividades do processo da Organização A

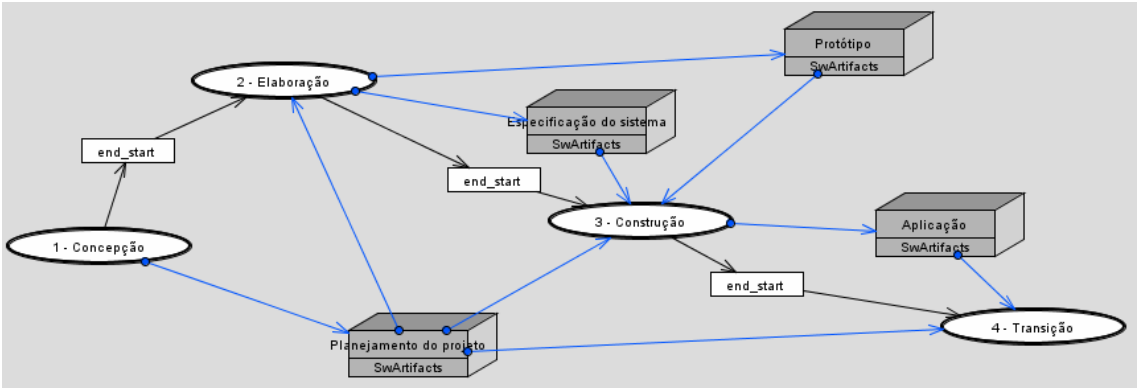

Figura 6.11: Modelo de processo da Organização B

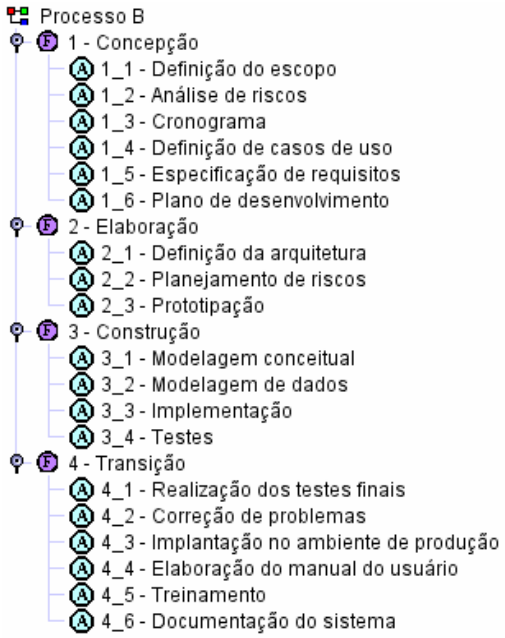

Figura 6.12: Estrutura hierárquica de atividades do processo da Organização B

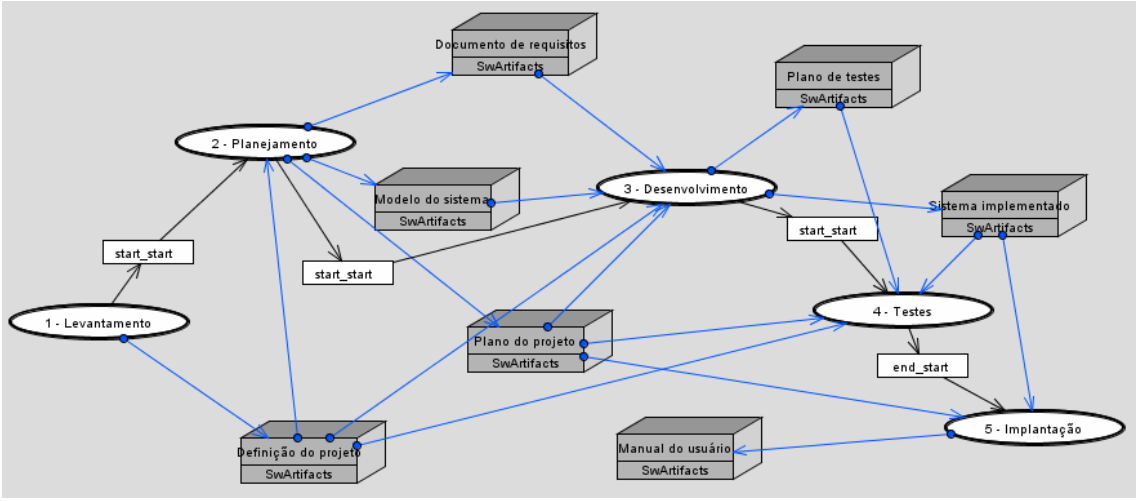

Figura 6.13: Modelo de processo da Organização C

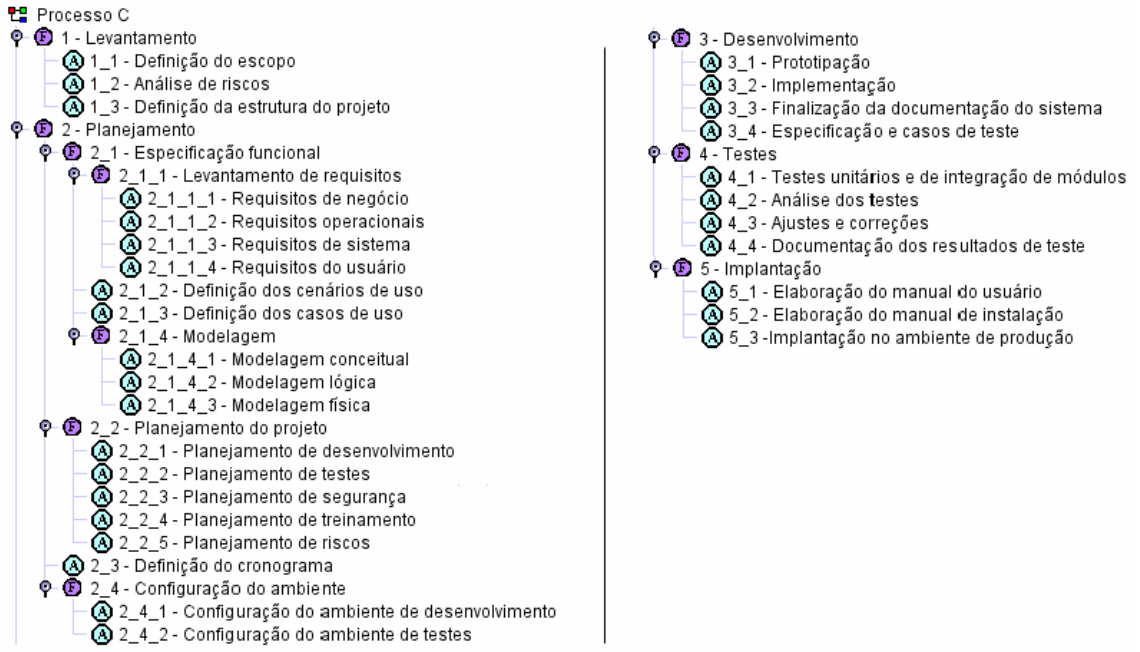

Figura 6.14: Estrutura hierárquica de atividades do processo da Organização C

A Figura 6.15 ilustra um caso de mapeamento do processo definido pela GSI por um dos parceiros ao qual o processo foi delegado. No exemplo apresentado, o parceiro Organização A mapeia o processo da GSI delegado a ele ao Processo A. Após a mapeamento do processo remoto, o parceiro está apto a mapear as atividades a ele delegadas.

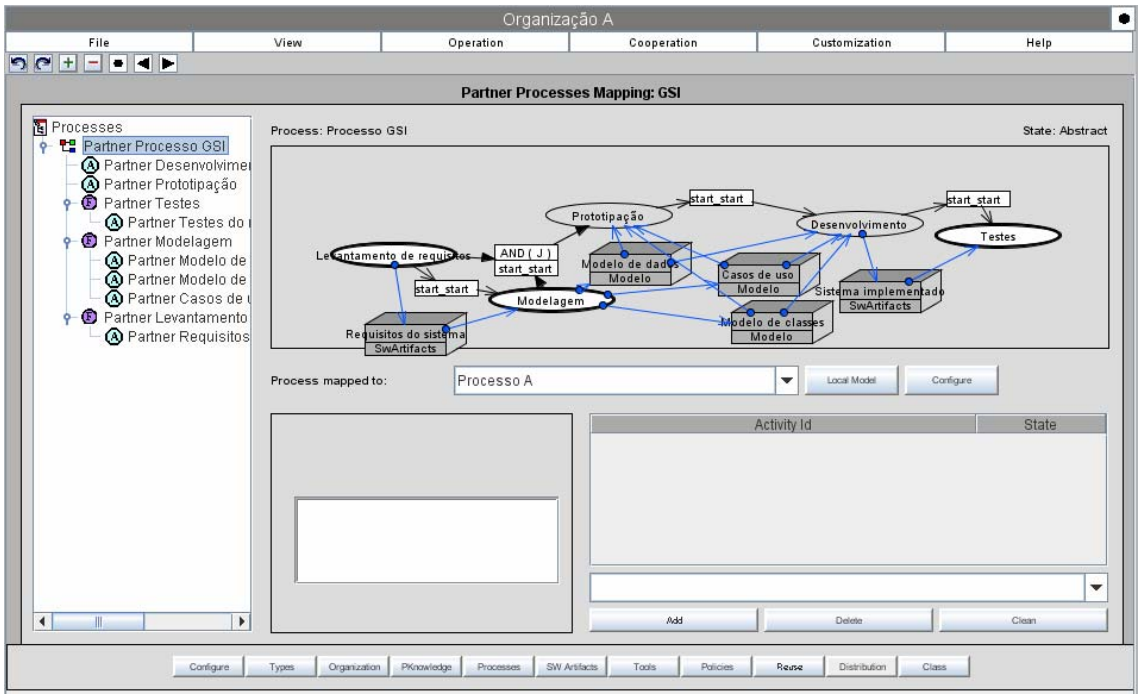

Figura 6.15: Mapeamento do processo remoto GSI pela Organização A

6.4.5 Mapeamento de atividades remotas

O mapeamento de atividades permite que o parceiro associe atividades a ele delegadas a atividades existentes em seus modelos de processo. No exemplo da

Figura 6.16, a Organização A, que já mapeou o modelo de processo, mapeia a atividade remota Desenvolvimento à atividade local Implementação.

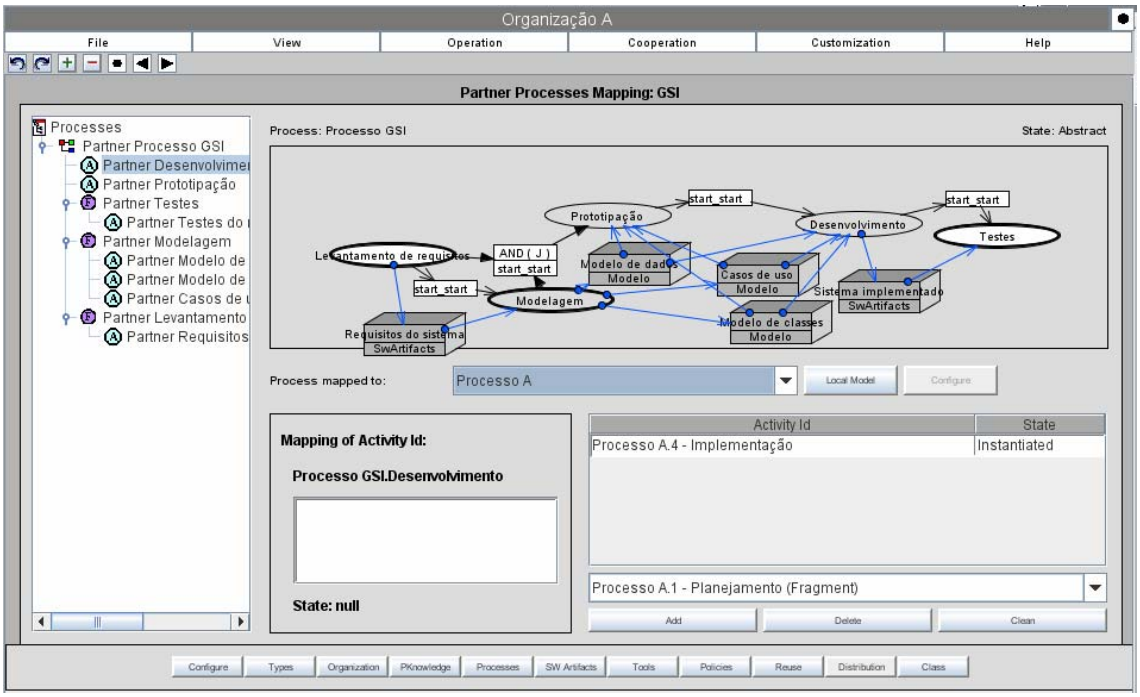

Figura 6.16: Mapeamento de atividade remota Desenvolvimento

As tabelas a seguir apresentam o mapeamento das atividades remotas realizado pelas organizações contratadas.

Tabela 6.3: Mapeamento de atividades remotas para atividades da Organização A

Atividades remotas	Atividades mapeadas
1.4 Requisitos de monitoramento de projetos	Levantamento de requisitos
2.4 Casos de uso do módulo PROJETO	Definição de casos de uso Criação da matriz de rastreabilidade
2.7 Modelo de dados do módulo PROJETO	Modelagem de dados
2.10 Modelo de classes do módulo PROJETO	Modelagem do sistema
3 Prototipação	Prototipação do modelo navegacional
4 Desenvolvimento	Implementação
5.3 Testes do módulo PROJETO	Testes

Tabela 6.4: Mapeamento de atividades remotas para atividades da Organização B

Atividades remotas	Atividades mapeadas
1.4 Requisitos de gerência financeira	Especificação de requisitos

2.4	Casos de uso do módulo FINANCEIRO	Definição de casos de uso
2.7	Modelo de dados do módulo FINANCEIRO	Modelagem de dados
2.10	Modelo de classes do módulo FINANCEIRO	Modelagem conceitual
3	Prototipação	Prototipação
4	Desenvolvimento	Implementação
		Testes
5.3	Testes do módulo FINANCEIRO	Realização dos testes finais
		Correção de problemas
		Documentação do sistema

Tabela 6.5: Mapeamento de atividades remotas para atividades da Organização C

	Atividades remotas	**Atividades mapeadas**
1.4	Requisitos de gerência financeira	Levantamento de requisitos
2.4	Casos de uso do módulo CONTRATO	Definição dos cenários de uso
		Definição dos casos de uso
2.7	Modelo de dados do módulo CONTRATO	Modelagem lógica
		Modelagem física
2.10	Modelo de classes do módulo CONTRATO	Modelagem conceitual
3	Prototipação	Prototipação
4	Desenvolvimento	Configuração do ambiente de desenvolvimento
		Implementação
		Finalização da documentação do sistema
5.3	Testes do módulo CONTRATO	Especificação de casos de teste
		Testes

6.5 Sincronização na execução dos processos

As informações geradas pela sincronização podem ser visualizadas tanto pela GSI quanto pelas organizações parceiras. A GSI acompanha o andamento da execução das atividades delegadas tanto pela atualização do estado da atividade no editor de processos do APSEE tanto pela tela de delegação de atividades. Nesse

caso, informações sobre a evolução da execução em cada uma das organizações é exibida, como ilustra a Figura 6.17.

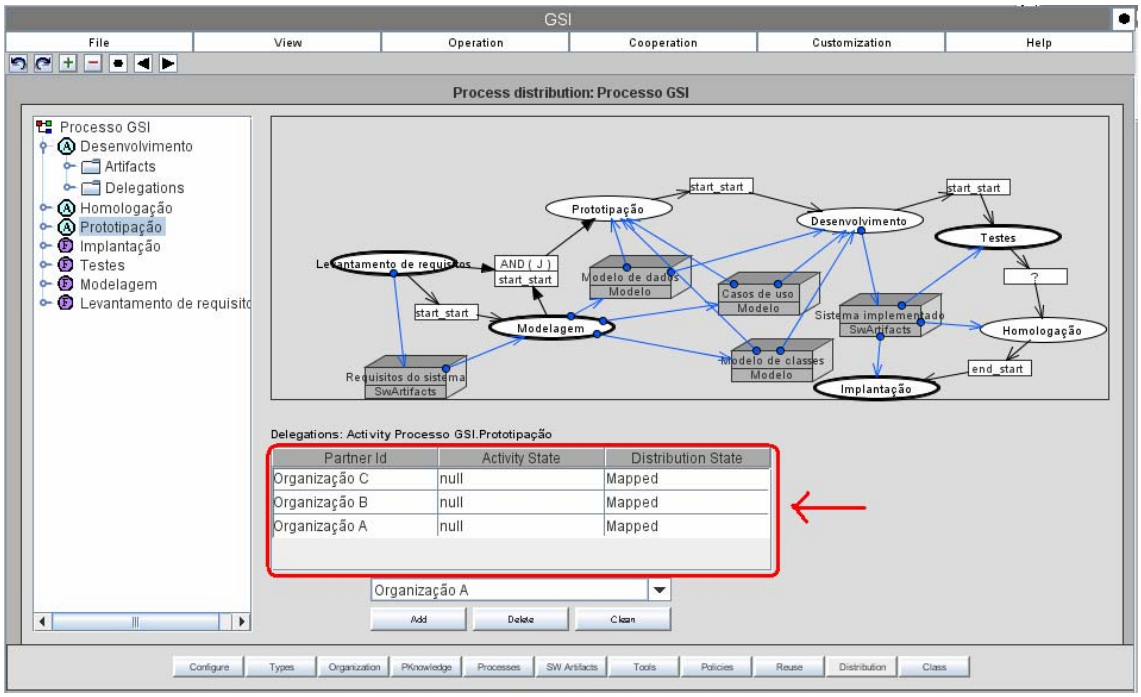

Figura 6.17: Informação sobre o andamento de execução de atividades delegadas

As organizações, por sua vez, acompanham a evolução local da execução de atividades remotas pela tela de mapeamento de atividades. Nessa tela, duas informações estão disponíveis, como ilustra a Figura 6.18: o estado de execução de cada atividade que mapeia a atividade remota, e o estado da atividade remota, definido pela combinação dos estados das atividades locais.

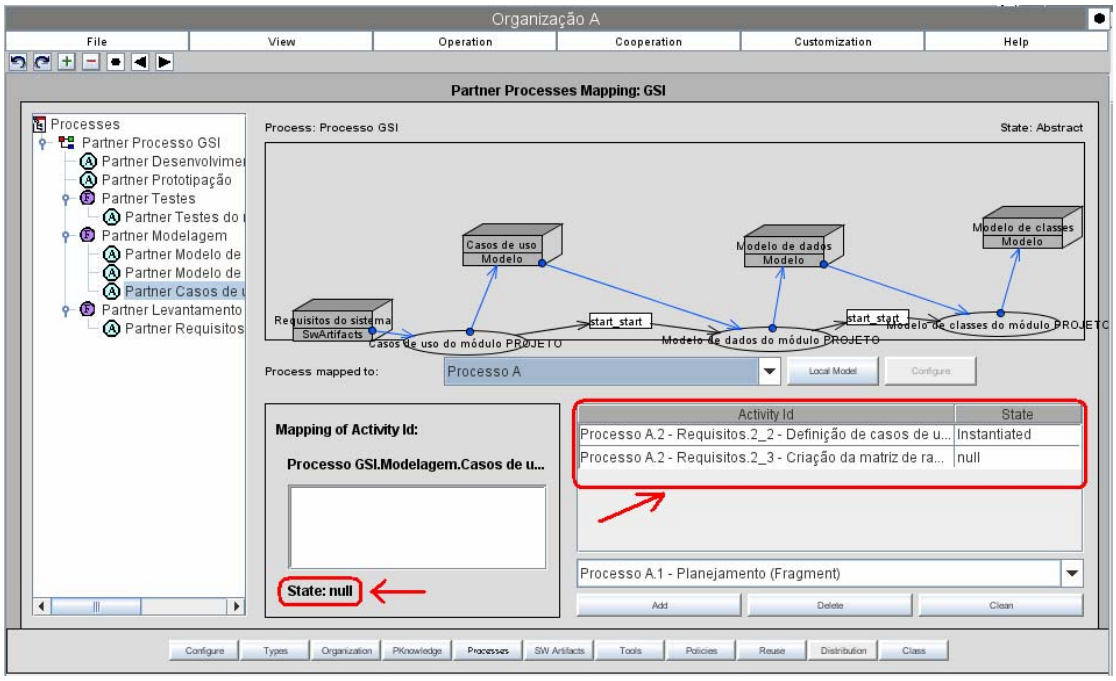

Figura 6.18: Informação sobre a evolução de atividades remotas

6.6 Análise dos resultados obtidos

O projeto utilizado como exemplo ainda encontra-se em andamento: assim não foi possível realizar uma avaliação final sobre o impacto da utilização do APSEE-*Global* na gerência de processos distribuídos. No entanto, alguns ganhos obtidos com a adoção do APSEE-*Global* já podem ser observados.

A combinação das abordagens *top-down* e *bottom-up* na definição dos processos globais mostrou-se um ganho importante da utilização do modelo. De um lado, a equipe da Gerência de Sistemas de Informação da SRA conseguiu ter uma visão global da distribuição do processo, detectando de maneira clara as relações entre atividades realizadas pelos diferentes parceiros. Isso facilita, por exemplo, a análise do impacto no andamento do projeto causado pelo atraso no cronograma de cada uma das equipes. Por outro lado, a utilização de modelos de processo já existentes facilitou bastante a condução do projeto: cada equipe pôde adotar o seu próprio modelo de processo, e ainda assim o acompanhamento do projeto tornou-se viável.

6.7 Considerações

Esse capítulo apresentou um exemplo prático de utilização do protótipo do APSEE-*Global*. Foi descrito o cenário utilizado para distribuição e mostrada a relação estabelecida entre os modelos de processo envolvidos. O desenvolvimento deste exemplo permitiu que se observasse o comportamento do protótipo durante a utilização, possibilitando a realização de ajustes na implementação e na especificação do sistema. Desse modo, a prototipação permitiu a validação da especificação formal.

A elaboração deste exemplo teve o intuito de ilustrar a utilização do protótipo. A avaliação e a validação do modelo proposto somente podem ser alcançadas pela realização de um conjunto de estudos de caso reais. Ainda assim, o desenvolvimento do protótipo é o passo inicial no caminho da experimentação prática do modelo APSEE-*Global*.

7 CONCLUSÕES

Este trabalho apresentou um modelo de gerência de processos distribuídos, que utiliza uma abordagem de delegação de atividades. O modelo proposto, denominado APSEE-*Global*, está integrado ao ambiente APSEE de modelagem e execução de processos, e fornece mecanismos para a distribuição dos modelos de processo definidos no APSEE. A distribuição de modelos de processo tem como objetivo facilitar a coordenação em projetos de desenvolvimento de software envolvendo equipes que trabalham de forma distribuída.

O modelo APSEE-*Global* foi especificado formalmente, com a utilização dos formalismos Prosoft-Algébrico e Gramática de Grafos. A especificação formal deu origem à implementação de um protótipo, no qual foi desenvolvido um exemplo. Embora o trabalho aqui apresentado não forneça uma solução completa para o problema investigado, acredita-se que o modelo proposto constitui uma contribuição importante para a área de desenvolvimento distribuído de software, em virtude da especificação formal apresentada e da prototipação realizada.

Este capítulo contém as conclusões obtidas com a realização do trabalho, e está estruturado como segue: a seção 7.1 apresenta as contribuições do trabalho; na seção 7.2 são abordadas as limitações do modelo proposto. A seção 7.3 compara este trabalho com outras abordagens encontradas na literatura; e a seção 7.4 identifica alguns trabalhos futuros. Por fim, a seção 7.5 apresenta as considerações finais do capítulo.

7.1 Contribuições

As principais contribuições apresentadas pela presente Dissertação são listadas a seguir:

- O texto apresenta, no capítulo 2, um estudo sobre diversos aspectos relacionados à gerência de processos distribuídos de software. Acredita-se que tais aspectos fornecem uma base importante para avaliação do trabalho aqui apresentado, permitindo a comparação com abordagens similares. Tais requisitos foram levantados em vários trabalhos encontrados na literatura;
- Os conceitos que serviram de base ao trabalho proposto neste texto foram publicados na 30a Conferencia Latinoamericana de Informática, em Arequipa (FREITAS; MAIA; NUNES, 2004-a), e no IV Congresso Brasileiro de Computação, em Itajaí, (FREITAS; MAIA; NUNES, 2004-b).

Essas publicações trouxeram como resultado contribuições importantes para a definição e a avaliação da proposta, tanto pela análise do comitê de programa quanto durante a apresentação dos artigos;

- O modelo para gerência de processos distribuídos proposto viabiliza o aumento da autonomia das equipes, pela possibilidade de adotarem modelos de processo distintos, e pela gerência descentralizada do processo de desenvolvimento;
- O mecanismo de modelagem da distribuição de processos de software permite a documentação e facilita a análise e a gerência das relações entre as equipes. A distribuição dos processos e o mapeamento das atividades delegadas podem ser realizados inclusive durante a execução do processo, de modo que a definição da distribuição do processo seja realizada de acordo com as necessidades identificadas no decorrer do desenvolvimento;
- O mecanismo de sincronização da execução dos modelos de processo provê um canal de comunicação formal para atualização do estado do processo distribuído, e permite que as equipes acompanhem a evolução dos processos distribuídos durante a execução;
- Os diferentes componentes envolvidos na definição do modelo foram especificados formalmente, constituindo uma base semântica de alto nível de abstração, o que traz muitos benefícios quando se considera a obtenção de uma especificação independente de implementação, tendo também o potencial de permitir que as idéias do trabalho possam ser implementadas em outros PSEEs;
- Um protótipo que implementa o modelo de distribuição de processos foi construído no ambiente Prosoft-Java e integrado ao ambiente APSEE. Essa integração estende o APSEE, e permite que o APSEE-*Global* utilize os recursos disponíveis no ambiente. Um exemplo de utilização do protótipo foi apresentado com base em um processo de desenvolvimento real, visando demonstrar as características do modelo proposto.

7.2 Limitações

Devido à complexidade do problema tratado, alguns tópicos não foram abordados no modelo proposto, por apresentarem, por si só, um grau de complexidade elevado, ou por não estarem contidos no escopo do trabalho. Esses tópicos constituem, em alguns casos, limitações na abrangência do modelo proposto, e podem ser alvo de melhorias futuras a serem incorporadas ao modelo. Dentre as limitações encontradas, é importante mencionar as seguintes:

- Por ser uma abordagem homogênea, o APSEE-*Global* provê suporte à distribuição de processos definidos segundo o meta-modelo de processos do APSEE. A gerência de processos modelados e/ou executados em outros PSEEs não é contemplada pelo APSEE-*Global*. Vale salientar que essa limitação foi assumida como premissa do trabalho, e que o fato de utilizar

uma abordagem homogênea permite que o modelo proposto possua forte integração com o APSEE, refletindo nos modelos de processo as alterações ocorridas na execução remota de atividades. Assim, essa limitação permite ganhos em contrapartida;

- Nenhum serviço de nomes é fornecido pelo APSEE-*Global*. Assim, a identificação das instâncias do ambiente envolvidas em um projeto distribuído é realizada pelos identificadores da instância e do *host*, e cada equipe informa ao ambiente a identificação dos seus parceiros;
- Embora o APSEE-*Global* permita a inclusão e a remoção de delegações de processos e atividades mesmo em tempo de execução, o histórico de delegações não é mantido. Tem-se apenas a informação das delegações vigentes, sem registro de delegações que tenham sido desfeitas. Possíveis relações entre delegações (por exemplo, transferência de delegação de uma equipe para outra), também não são registradas;
- Outro aspecto que merece um estudo mais detalhado é o de propagação de falhas entre atividades. A incorporação do conceito de atividades críticas e não-críticas pode ser interessante no comportamento do mecanismo em caso de falha de atividades, para definição de para quais atividades a falha deve ser propagada.

7.3 Comparação com trabalhos relacionados

No capítulo 2 foram apresentadas algumas abordagens encontradas na literatura de ambientes que provêem suporte a processos distribuídos. Os trabalhos Oz, Serendipity-II, AHEAD, CAGIS, APEL e PIE foram descritos brevemente, e uma comparação entre eles foi realizada. Essa sessão compara o APSEE-*Global* aos trabalhos apresentados, ressaltando como o trabalho proposto está relacionado com outras soluções encontradas no contexto da gerência de processos distribuídos de software.

Uma vantagem do APSEE-Global em relação a todos os trabalhos apresentados é a definição de processos globais usando uma abordagem que combina as estratégias *top-down* e *bottom-up*. Essa abordagem permite a integração de modelos de processo já existentes, e ao mesmo tempo provê uma visão geral do processo de desenvolvimento, de modo que a distribuição seja definida sem necessidade de conhecer os sub-processos envolvidos no desenvolvimento. A integração entre modelos existentes é suportada pelo Oz, CAGIS, APEL e PIE. Já os trabalhos Serendipity-II e AHEAD provêem a visão geral do modelo. No entanto, nenhuma das abordagens combina as duas características.

A hierarquia na distribuição dos processos é uma característica importante do APSEE-*Global*. A maioria dos trabalhos estudados considera que todas as equipes estão no mesmo nível. Apenas o AHEAD define uma hierarquização; mas não permite, por exemplo, que uma equipe re-delegue uma atividade a ela delegada. No APSEE-*Global* isso é possível pela delegação de atividades que mapeiam atividades remotas.

Uma diferença entre o APSEE-*Global* e os trabalhos citados é o tipo de relação entre os processos. Oz, CAGIS, APEL e PIE têm foco na execução cooperativa de atividades, mas não explicitam relações de dependências entre atividades de processos distintos. Apesar do foco na cooperação, atividades cooperativas no Oz compartilham dados, mas são executadas por uma única equipe, que atualiza os dados das demais. A relação de dependência entre atividades é encontrada no AHEAD, pelo fato de que atividades delegadas possuem dependência em relação as demais atividades do processo da organização contratante. No entanto, o AHEAD não suporta o conceito de atividades cooperativas, pois só permite a delegação de atividades a uma única equipe. O único trabalho estudado que reúne os dois focos é o Serendipity-II. A abordagem de delegação do APSEE-*Global* define dependências entre atividades, e a execução cooperativa é suportada pela delegação de atividades a mais de uma equipe. A limitação é que atividades delegadas não podem ser executadas cooperativamente pela equipe contratante e pela contratada.

Das abordagens estudadas, apenas o AHEAD utiliza o conceito de delegação de atividades. Além dos já citados, o APSEE-*Global* apresenta outros ganhos em relação ao AHEAD, como o fato de que, no AHEAD, o fragmento delegado não pode ser alterado durante a execução. Assim, se novas atividades tiverem que ser delegadas, com dependência direta às já delegadas, essa relação não é explicitada, pois dois fragmentos distintos serão criados na equipe contratada.

7.4 Trabalhos futuros

O trabalho aqui apresentado apresenta importantes soluções para a gerência de processos distribuídos. No entanto, vários tópicos ainda estão em aberto, servindo de base para o desenvolvimento de outros trabalhos que abordem a mesma linha:

- Embora o uso de métodos de especificação formal possibilite a realização de provas e verificações matemáticas acerca de propriedades do modelo, tal aspecto não foi considerado como objetivo do trabalho aqui apresentado. A realização de provas, entretanto, é um recurso importante que pode ser aproveitado em trabalhos futuros. Um ponto crítico do modelo que necessita ser provado é que as atividades sempre estarão em estados consistentes, sem que a execução remota viole as dependências entre atividades definidas nos modelos de processo;
- Um dos tópicos que necessita de exploração é a avaliação empírica da influência do modelo proposto na gerência de processos distribuídos. A realização desse trabalho, por si só, não garante um aumento da qualidade, produtividade ou melhoria na gerência de processos distribuídos ou para o software resultante. Assim, uma importante atividade a ser realizada é a condução de uma avaliação empírica que determine, através de estudos de casos aplicados em projetos reais de desenvolvimento distribuído, os benefícios reais obtidos quando o modelo proposto é utilizado, comparando

com situações similares quando nenhum apoio automatizado para a distribuição está disponível;

- Um aspecto relacionado à execução de processos que não foi considerado neste trabalho é a avaliação de métricas globais. Estudos podem ser realizados para permitir a coleta de métricas relativas à execução de atividades por parceiros. Devem ser levadas em consideração as relações entre equipes, que podem restringir o acesso a informações sobre os modelos de processo. Um mecanismo para definição, em comum acordo, de quais informações podem ser coletadas pode ser necessário;
- Como apresentado no capítulo 2, diversos critérios a serem adotados para divisão das atividades entre equipes podem ser encontrados na literatura, como por exemplo a partir das funcionalidades do sistema, dos estágios de desenvolvimento, com a disponibilidade de ferramentas e recursos. Estudos adicionais podem ser realizados para definir estratégias que auxiliem a distribuição das atividades, a partir da seleção de critérios e de informações sobre o processo e sobre os parceiros;
- Por se tratar de um aspecto ortogonal ao foco deste trabalho, o modelo proposto não trata a operação desconectada de uma ou mais instâncias do APSEE-*Global*. Embora a execução de atividades puramente locais não dependa da comunicação com outras instâncias, a desconexão pode levar a inconsistências no modelo, por exemplo, pelo atraso na propagação de falhas e cancelamento. É interessante, no entanto, a realização de estudos para análise do funcionamento do modelo no caso de operação desconectada de instâncias do APSEE-*Global;*
- Por fim, um outro tópico para trabalhos futuros é a análise da viabilidade de estender o modelo proposto para a operação em ambientes heterogêneos (envolvendo diferentes PSEEs), ou mesmo para operação homogênea tendo como base outro PSEE diferente do APSEE.

7.5 Considerações finais

Este trabalho foi proposto com o objetivo de contribuir com a evolução da Tecnologia de Processos de Software, propondo um modelo de gerência de processos de software distribuídos. As contribuições do trabalho, limitações e sugestões de trabalhos futuros foram apresentadas neste capítulo.

As idéias contidas neste trabalho foram embasadas, em muitos casos, por entrevista realizada com o Sr. Franco Vieira, *liason* da HP (*Hewlett-Packard*®), que forneceu informações sobre problemas práticos encontrados em situações reais de desenvolvimento distribuído.

Graças ao projeto de cooperação internacional Brasil-Alemanha foi possível contar com o auxílio do pesquisador Heribert Schlebbe da *Universität Stuttgart* que muito contribuiu para a prototipação do trabalho.

REFERÊNCIAS

ALVES, R. **Uma Proposta de apoio para decisões de grupo no ambiente Prosoft.** 2002. 173 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

ARBAOUI, S. et al. A Comparative Review of Process-Centered Software Engineering Environments. **Annals of Software Engineering**, The Netherlands, v.14, n.1-4, p. 311–340, Dec. 2002.

BANDINELLI, S. et al. SPADE: an environment for software process analysis, design, and enactment. In: FINKELSTEIN, A.; KRAMER, J.; NUSEIBEH, B. (Ed.). **Software process modeling and technology**. Taunton, UK: Research Studies Press, 1994. p.223–248. (Research Studies Press Advanced Software Development Series).

BARTHELMESS, P. Collaboration and Coordination in Process-Centered Software Development Environments: a review of the literature. **Information & Software Technology**, The Netherlands, v.45, n.13, p.911–928, Oct. 2003.

BECKER, S. et al. A Delegation Based Model for Distributed Software Process Management. In: EUROPEAN WORKSHOP ON SOFTWARE PROCESS TECHNOLOGY, EWSP, 8., 2001, Witten, Germany. **Software process technology**: proceedings. Berlin: Springer, 2001. p. 130–144. (Lecture Notes in Computer Science, 2077).

BEN-SHAUL, I.; KAISER, G. **An Interoperability Model for Process-Centered Software Engineering Environments and its Implementation in Oz**. [S.l.]: Columbia University, Department of Computer Science, 1995. (Technical Report CUCS-034-95).

BEN-SHAUL, I.; KAISER, G. A Paradigm for Decentralized Process Modeling and its Realization in the Oz Environment. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 18., 1996, Sorrento, Italy. **Proceedings...** [S.l.]: IEEE CD Press, 1996. p. 179-188.

BEN-SHAUL, I.; KAISER, G. Federating Process-Centered Environments: the Oz Experience. In: AUTOMATED SOFTWARE ENGINEERING, 5., 1998. **Proceedings…** The Netherlands: Kluwer Academic Publishers, 1998. p. 97–132.

CARMEL, E. Taxonomy of New Software Exporting Nations. **The Electronic Journal on Information Systems in Developing Countries**, Hong Kong, v. 13, p. 1-6, May 2003.

CARMEL, E.; AGARWAL, R. Tactical Approaches for Alleviating Distance in Global Software Development. **IEEE Software**, [S.l.], p. 22-29, Mar. 2001.

COHEN, B.; HARWOOD, W.; JACKSON, M. **The specification of complex systems**. Great Britain: Addison-Wesley Publishing Company, 1986. 143p.

CUGOLA, G.; GHEZZI, C. Software Processes: a retrospective and a path to the future. **Software Process: Improvement and Practice**, [S.l.], v.4, n.3, p.101–123, 1998.

CUGOLA, G.; GHEZZI, C. Design and Implementation of PROSYT: a distributed process support system. In: WORKSHOP ON ENABLING TECHNOLOGIES ON INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISES, WETICE, 8., 1999, Washington, DC, USA. **Proceedings...** [S.l.]**:** IEEE Computer Society, 1999. p.32–39.

CUGOLA G. et al. Support for Software Federations: the PIE Platform. In: EUROPEAN WORKSHOP ON SOFTWARE PROCESS TECHNOLOGY, EWSPT, 7., 2000, Kaprun, Áustria. **Proceedings**… [S.l.:s.n.], 2000.

DAHMER, A. **Um Ambiente para Desenvolvimento e Gerência de Cursos em Educação a Distância.** 2005. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre. Tese a ser defendida.

DAMIAN, D. Global Software Development. In: WORKSHOP DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE, 1., 2004, Porto Alegre. **Anais...** Porto Alegre: PUC-RS, 2004.

DAUDT, R. **Uma Proposta de Extensão do Prosoft Básico**. 1992. Trabalho de Diplomação (Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

DÉHARBE, D.; MOREIRA, A.; RIBEIRO, L.; RODRIGUES, V. Introdução a Métodos Formais: Especificação, Semântica e Verificação de Sistemas Concorrentes. **Revista de Informática Teórica e Aplicada**, Porto Alegre, v.6, n.1, set. 2000.

EBERT, C.; DE NEVE, P. Surviving Global Software Development. **IEEE SOFTWARE**. [S.l.], p. 62 – 69, Mar. / Apr. 2001.

ENGELS, G. et al. Process-centered software engineering environments: academic and industrial perspectives. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 23., 2001, Toronto, Ontario, Canada. **Proceedings...** [S.l.]: IEEE Computer Society, 2001. p.671–673.

ESPINOSA, J.; CARMEL, E. The Impact of Time Separation on Coordination in Global Software Teams: a Conceptual Foundation. In: SOFTWARE PROCESS IMPROVEMENT AND PRACTICE, SPIP, 8., 2004. **Proceedings…** [S.l.]: John Wiley & Sons, 2004.

ESTUBLIER, J.; CUNIN, P.; BELKHATIR, N. Architectures for Process Support System Interoperability. In: INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS, ICSP, 5., 1998, Lisle. **Proceedings...** [S.l.:s.n.], 1998.

ESTUBLIER, J.; AMIOUR, M.; DAMI, S. Building a Federation of Process Support Systems. In: WORK, ACTIVITY COORDINATION AND COOPERATION, WACC, 1999, San Francisco, USA. **Proceedings…** [S.l.]: Siplan, Sigmod, Sigsoft, 1999. p. 22 – 26.

FALBO, A. Automatização do Processo de Desenvolvimento de Software. In: CONFERÊNCIA INTERNACIONAL DE TECNOLOGIA DE SOFTWARE, CITS, 7., 1996, Curitiba. **Anais...** [S.l.:s.n], 1996. p. 71 – 88.

FIPS (Federal Information Processing Standards). **Integration definition for function modeling: IDEF0**. Gaithersburg, MD, USA, 1993.

FREITAS, A. **Execução descentralizada de processos de software**. 2004. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

FREITAS, A; MAIA, A; NUNES, D. Gerenciamento da Integração entre Processos de Software no APSEE-Integrate. In: CONFERENCIA LATINOAMERICANA DE INFORMÁTICA, CLEI, 30., 2004, Arequipa, Peru. **Artículos...** La Paz: Universidad Mayor de San Andrés, 2004. p. 620 – 631.

FREITAS, A; MAIA, A; NUNES, D. Um Modelo para Interação entre Processos de Software. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, CBCOMP, 4., 2004, Itajaí. **Anais....** Itajaí: Univali, 2004. p. 149 – 154.

FUGGETTA, A. Software Process: a roadmap. In: CONFERENCE ON THE FUTURE OF SOFTWARE ENGINEERING, 2000, Limerick, Ireland. **Proceedings...** New York: ACM Press, 2000. p.25 – 34.

GARG, P.; JAZAYERI, M. Process-Centered Software Engineering Environments: a grand tour. In: FUGGETTA, A.; WOLF, A. (Ed.). **Trends in Software**: software process. Chichester, UK: John Wiley & Son, 1996. v.4, p.25–52.

GIMENES, I. **ExPSEE**: um ambiente experimental de engenharia de software orientado a processos. Maringá: Departamento de Informática da Universidade Estadual de Maringá, 2000. Relatório de Projeto.

GRANVILLE, L.; SCHLEBBE, H. **Distributed Prosoft**: management of tools and memory. UFRGS e Uni-Stuttgart. 1996. 26 p. Relatório Técnico.

GRINTER, R.; HERBSLEB, J.; PERRY, D. The Geography of Coordination: Dealing with Distance in R&D Work. In: INTERNATIONAL CONFERENCE ON SUPPORTING GROUP WORK, GROUP, 1999, Phoenix, Arizona, USA. **Proceedings…** New York: ACM SIG GROUP, 1999. p. 306 – 315.

GRUNDY, J. Visual specification and monitoring of software agents in decentralized process-centred environments. **International Journal of Software Engineering and Knowledge Engineering** , Skokie, USA, v.9, n.1, Feb. 1999.

GRUNDY, J. et al. A decentralized architecture for process modeling and enactment. **IEEE Internet Computing**, New York, v.2, n.5, p. 53 – 62, Sept. / Oct. 1998.

GRUHN, V. Process-Centered Software Engineering Environments: a brief history and future chalenges. **Annals of Software Engineering**, The Netherlands, v.14, n.1–4, p.363–382, Dec. 2002.

HAYWOOD, M. Working in Virtual Teams: a tale of two projects and many cities. **IT Professional**, New York, v.2, n.2, p. 58– 60, Mar. / Apr. 2000.

HERBSLEB, J.; GRINTER, R. Splitting the Organization and Integrating the Code: Conway's Law Revisited. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 1999, Los Angeles, CA. **Proceedings…** [S.l.]: IEEE CD Press, 1999. p. 85 – 95.

JACCHERI, L.; LARSEN, J.-O.; CONRADI, R. Software Process Modeling and Evolution in EPOS. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, SEKE, 4., 1992, Capri, Italy. **Proceedings. . .** [S.l.]: IEEE Computer Society Press, 1992. p.574–581.

JACKSON, M. **System Development.** New York: Prentice-Hall International, 1983.

JUNKERMANN, G. et al. Merlin: supporting cooperation in software development through a knowledge-based environment. In: FINKELSTEIN, A.; KRAMER, J.; NUSEIBEH, B. (Ed.). **Software process modelling and technology**. Taunton, UK: Research Studies Press, 1994. p.103–129. (Research Studies Press Advanced Software Development Series).

KAISER, G.; FEILER, P. An architecture for intelligent assistance in software development. In: INTERNATIONAL CONFERENCE ON SOFTWARE

ENGINEERING, ICSE, 9., Monterey C. **Proceedings…** [S.l.]: IEEE Computer Society Press, 1987. p. 180–188.

KAISER, G.; FEILER, P.; POPOVICH, S. Intelligent Assistance for Software Development and Maintenance. **IEEE Software**, [S.l.], v.5, n.3, p.40–49, May / June 1988.

KAROLAK, D. **Global Software Development:** managing virtual teams and environment. New Brunswick: The Computer Society, 1998.

KOBITZSCH, W.; ROMBACH, D.; FELDMANN, R. Outsourcing in India. **IEEE Software**, [S.l.], v.18, n.2, p. 78–86, Mar. / Apr. 2001.

LANUBILE, F.; DAMIAN, D.; OPPENHEIMER, H. Global Software Development: Technical, Organizational, and Social Challenges. **ACM SIGSOFT Software Engineering Notes**, New York, v.28, n.6, Nov. 2003.

LIMA REIS, C. **Um Gerenciador de Processos de Software para o ambiente Prosoft**. 1998. 197 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

LIMA REIS, C. **Uma abordagem flexível para execução de processos de software evolutivos**. 2003. 267 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

MAIA, A.; FREITAS, A.; NUNES, D. Um Modelo para Auxiliar a Adaptação de Processos de Software. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, CBCOMP, 4., 2004, Itajaí. **Anais...** Itajaí: Univali, 2004. p.155–160.

MAIA, A.; FREITAS, A.; NUNES, D.; STEINMACHER, I. Componentes de um Modelo para Adaptação de Processos de Software. In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, CACIC, 10., 2004, Buenos Aires. **Anales...** [S.l.:s.n.], 2004. p. 42-53.

MAIDANTCHICK, C.; DA ROCHA, A. Managing a worldwide software process. In: INTERNATIONAL WORKSHOP ON GLOBAL SOFTWARE DEVELOPMENT, ICSE, 2002, Orlando, Florida. **Proceedings…** [S.l.:s.n.], 2002.

MAIR, Q.; HAAG, Z. Process Model Integration in Internet-based Virtual Software Corporations. In: INTERNATIONAL WORKSHOP ON GLOBAL SOFTWARE DEVELOPMENT, ICSE, 2001, Toronto, Canadá. **Proceedings…** [S.l.:s.n.], 2001.

MALONE, T.; CROWSTON, K. What is coordination theory and how can it help design cooperative work systems? In: COMPUTER-SUPPORTED COOPERATIVE WORK, CSCW, 1990, Los Angeles, California, United States. **Proceedings…** New York, ACM Press, 1990. p. 357- 370.

MICROSOFT. **Microsoft Solution Framework White Paper**: MSF Process Model v. 3.1. USA, 2004.

MOCKUS, A.; WEISS, D. Globalization by Chunking: a quantitative approach. **IEEE Software**, [S.l.], v.18, n.2, p. 30 – 37, Mar. / Apr. 2001.

MOCKUS, A.; HERBSLEB, J. Challenges of Global Software Development. In: INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, METRICS, 7., 2001. London. **Proceedings…** [S.l.:s.n.], 2001. p. 182 – 184.

MORAES, S. **Um Ambiente Expert para Apoio ao Desenvolvimento de Software**. 1997. 139 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

NUNES, D. Estratégia Data-Driven no Desenvolvimento de Software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 6., 1992, Porto Alegre. **Anais. . .** Porto Alegre: Instituto de Informática – Universidade Federal do Rio Grande do Sul, 1992. p.81–95.

NUNES, D. **Prosoft**: um ambiente de desenvolvimento de software baseado no método algébrico. Porto Alegre: Instituto de Informática – Universidade Federal do Rio Grande do Sul, 1994. (Relatório Técnico)

OLSON, J.; OLSON, G. Culture Surprises in Remote Software Development Teams. **Queue Focus: Distributed Development**, [S.l.], v.1, n.9, p. 52-59, Dec. / Jan. 2003-2004.

OMG (Object Management Group). **Unified Modeling Language:** UML 2 Superstructure. Needham, USA, 2004.

PIMENTA, A. **Especificação formal de uma ferramenta de reutilização de especificações de requisitos**. 1998. 120 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

PMI (Project Management Institute). **A Guide to Project Management Body of Knowledge** (PMBOK Guide). 3rd ed. [S.l.]: Project Management Institute, 2004.

PRESSMAN, R. **Engenharia de Software**. 5. ed. Rio de Janeiro: McGraw-Hill, 2002. p. 657-681.

PRIKLADNICKI, R.; AUDY, J. Towards a Model of software development process for a physically distributed environment In: CONGRESSO ARGENTINO DE CIÊNCIAS DE LA COMPUTACION, CACIC, 8., 2002, Buenos Aires. **Anales...** Buenos Aires: Del Dpto. Computacion FCET – UBA, 2002. p.798 – 809.

PRIKLADNICKI, P.; AUDY, J.; EVARISTO, R. Distributed Software Development: toward an understanding of the relationship between project team, users and customer. In: INTERNATIONAL CONFERENCE ON ENTERPRISE

INFORMATION SYSTEMS, ICEIS, 5., 2003, Angers, France. **Proceedings…** [S.l.:s.n.], 2003.

RAMAMPIARO, H.; WANG, A.; BRASETHVIK, T. Supporting Distributed Cooperative Work in CAGIS. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND APPLICATIONS, SEA, 4., 2000, Las Vegas, Nevada, USA. **Proceedings…** [S.l.:s.n.], 2000.

RANGEL, G. S. **ProTool**: uma ferramenta de prototipação de software para o ambiente Prosoft. 2003. 223 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

IBM. **Rational Unified Process**: Best Practices for Software Development Teams. [S.l.], 2004.

REIS, R. **Uma proposta de suporte ao desenvolvimento cooperativo de software no ambiente Prosoft**. 1998. 177 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

REIS, R. **APSEE-Reuse**: um meta-modelo para apoiar a reutilização de processos de software. 2002. 214 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

REIS, R.; LIMA REIS, C.; NUNES, D. APSEE-StaticPolicy: verificação de políticas estáticas em modelos de processos de software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 15., 2001, Rio de Janeiro. **Anais. . .** Rio de Janeiro: UFRJ, 2001.

REIS, R. Q.; LIMA REIS, C. A.; SCHLEBBE, H.; NUNES, D. J. Automatic Verification of Static Policies on Software Process Models. **Annals of Software Engineering**, Boston, MA, v.14, n.1-4, 2002.

RIBEIRO, Leila. Métodos Formais de Especificação: Gramáticas de Grafos. In: ESCOLA DE INFORMÁTICA DA SBC – SUL, 8., 2000, Santa Maria. **Anais...** Santa Maria: UFSM, 2000. p. 1 – 34.

ROSS, D. Applications and Extensions of SADT. **IEEE Computer**, New York, USA, v.18, n.4, p.25–35, 1985.

SCHLEBBE, H. **Distributed Prosoft**: report on a working stay at the institute of computer science of the state university of Rio Grande do Sul (UFRGS) at Porto Alegre, Brazil from May 1 to June 15, 1994. Stuttgart: Universität Stuttgart. Fakultät Informatik, 1994.

SCHLEBBE, H. **Java Prosoft Manual.** Disponível em: <http://www.informatik.uni-stuttgart.de/ifi/bs/schlebbe/prosoft_doc/guide> Acesso em: 01 set. 2005.

SCHLEBBE, H.; SCHIMPF, S. **Reenginering of Prosoft in Java.** Stuttgart: Universität Stuttgart. Fakultät Informatik, 1997. Technical Report.

SEI (Software Engineering Institute). **CMU/SEI-94-TR-007**: a Practical Guide to the Technology and Adoption of Software Process Automation. [S.l.], 1994.

SEI (Software Engineering Institute). **CMMI-SE/SW/IPPD/SS, V1.1**: CMMI^{SM} for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing. [S.l.], 2002.

SILVA, F. **Um modelo de simulação de processos de software baseado em conhecimento para o ambiente Prosoft**. 2001. 123 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SIQUEIRA, F.; SILVA, P. As Características do Desenvolvimento Distribuído de Software. In: SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO, SBSI, 1., 2004, Porto Alegre. **Anais...** Porto Alegre: PUCRS, 2004.

SOUSA, A. L. R. de. **APSEE-Monitor**: um mecanismo de apoio à visualização de modelos de processos de software. 2003. 112 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SUN. **Java Runtime Environment (JRE)**. Disponível em: <http://java.sun.com/j2se/desktopjava/jre/index.jsp>. Acesso em: 07 set. 2005.

SUN. J**ava Remote Method Invocation (Java RMI).** Disponível em: <http://java.sun.com/products/jdk/rmi/>. Acesso em: 07 set. 2005.

SUN. **Java™ 2 Platform Standard Edition 5.0 API Specification**. Disponível em: <http://java.sun.com/j2se/1.5.0/docs/api/>. Acesso em: 07 set. 2005.

TOTLAND, T.; CONRADI, R. A Survey and Comparison of Some Research Areas Relevant to Software Process Modeling. In: EUROPEAN WORKSHOP ON SOFTWARE PROCESS TECHNOLOGY, EWSPT, 4., 1995, London, UK. **Proceedings...** [S.l.]: Springer–Verlag, 1995. p.65–69.

TRAVASSOS, G. H. **O modelo de integração de ferramentas da estação TABA**. 1994. Tese (Doutorado em Ciência da Computação) – Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro.

VAN DER AALST, W. Process-oriented architectures for eletronic commerce and interorganizational workflow. **Information Systems**, [S.l.], v.24, n.8, p. 639-671, 1999.

WANG, A. Support for Mobile Software Processes in CAGIS. In: EUROPEAN WORKSHOP ON SOFTWARE PROCESS TECHNOLOGY, EWSPT, 7., 2000, Kaprun near Salzburg, Austria. **Proceedings...** [S.l.:s.n.], 2000.

WANG, A. Using Software Agents to Support Evolution of Distributed Workflow Models. In: INTERNATIONAL ICSC SYMPOSIUM ON INTERACTIVE AND COLLABORATIVE COMPUTING, ICC, 2000, Wollongong (near Sydney), Austrália. **Proceedings**... [S.l.:s.n.], 2000.

WANG, A. A Process Centred Environment for Cooperative Software Engineering. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, SEKE, 4., 2002, Ischia, Italy. **Proceedings**... [S.l.:s.n.], 2002.

WFMC. ***Workflow*** **Management Coalition Interface 1:** process definition interchange process model. [S.l.], 1999.(Technical Report WFMC-TC-1016).

ZANONI; R. **Modelo de Gerência de Projeto Baseado no PMI para Ambiente de Desenvolvimento de Software Fisicamente Distribuído**. 2002. 130 f. Dissertação (Mestrado em Ciência da Computação) – CPGCC, PUC-RS, Porto Alegre.

ANEXO TIPOS DE DADOS DO APSEE

Neste anexo são apresentadas as principais classes que compõem os tipos de dados hierarquias de tipos, processos de software e artefatos de software, definidas por Lima Reis (2003). Esses tipos de dados foram utilizados na especificação do APSEE-*Global*. Uma descrição detalhada das classes apresentadas foge ao escopo deste trabalho, podendo ser encontrada em (LIMA REIS, 2003).

- **Tipo Hierarquia de tipos**

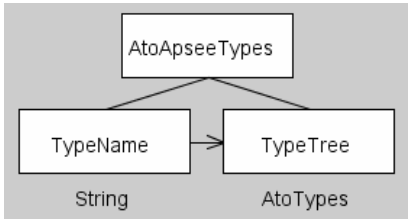

Figura A.1: Classe *APSEETypes*

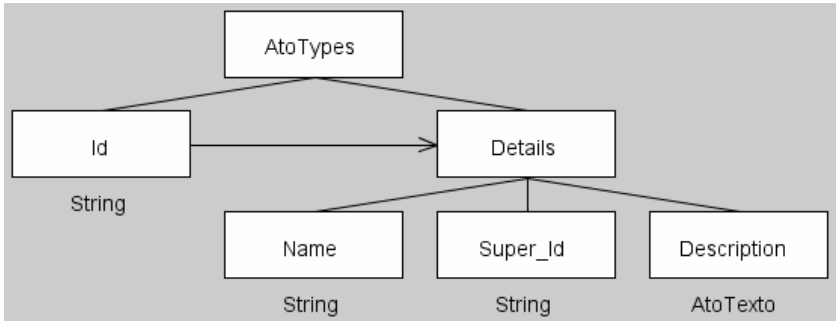

Figura A.2: Classe *Types*

- **Tipo Processos de software**

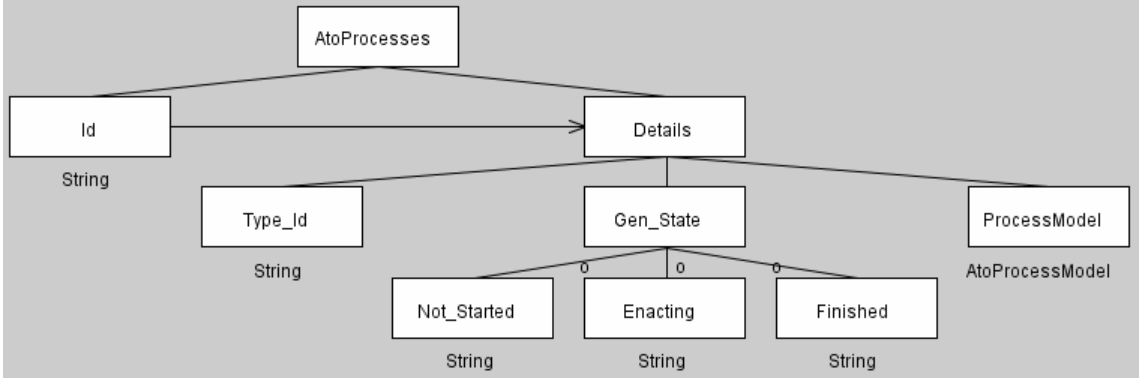

Figura A.3: Classe *Processes*

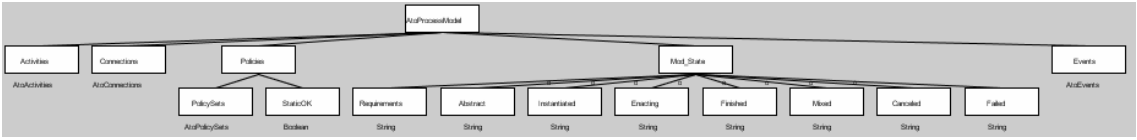

Figura A.4: Classe *ProcessModel*

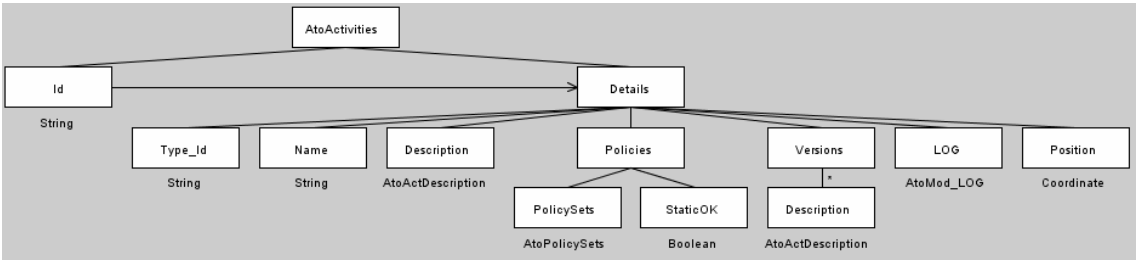

Figura A.5: Classe *Activities*

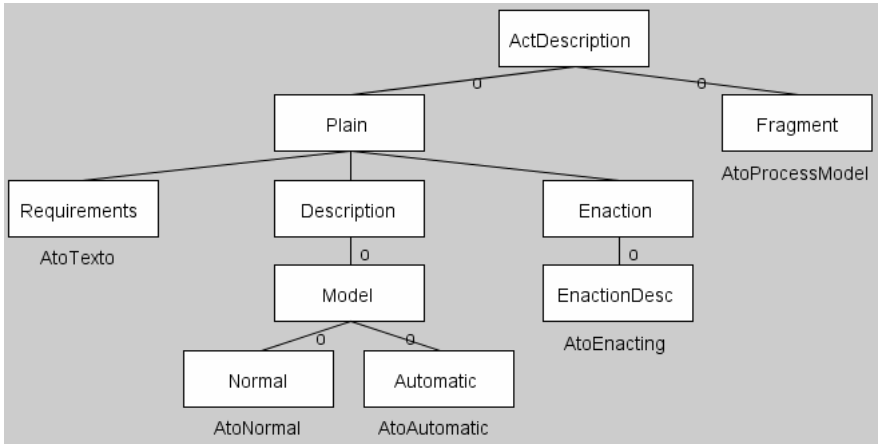

Figura A.6: Classe *ActDescription*

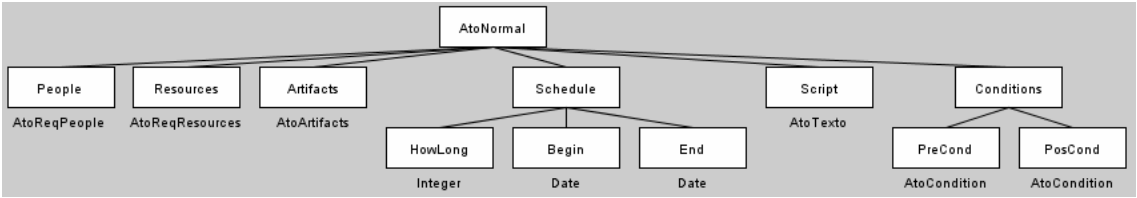

Figura A.7: Classe *Normal*

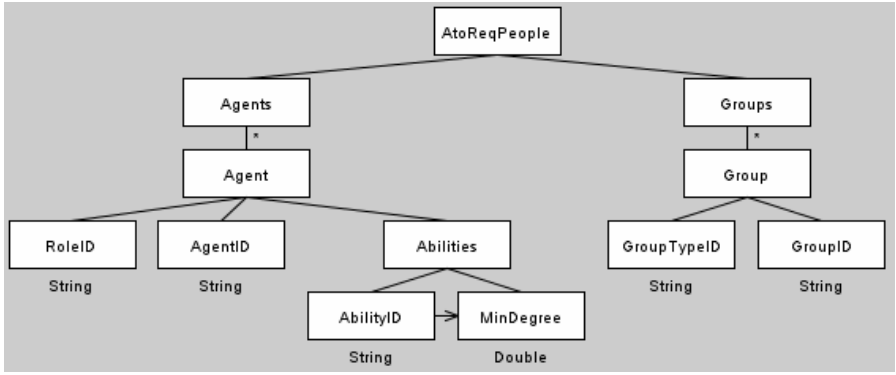

Figura A.8: Classe *ReqPeople*

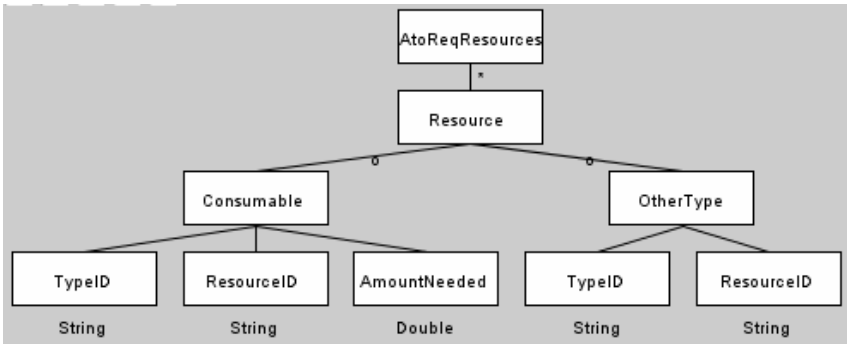

Figura A.9: Classe *ReqResources*

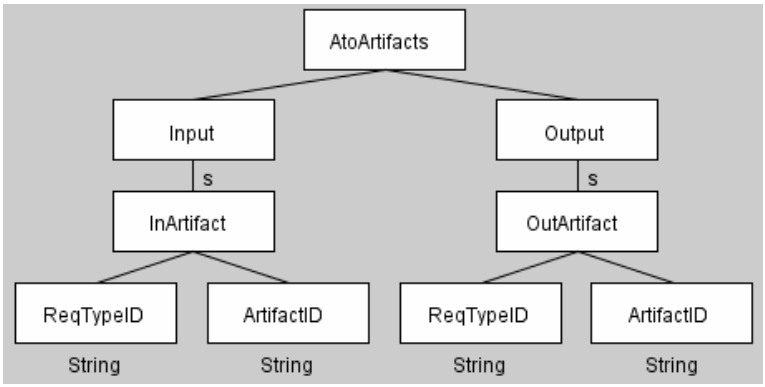

Figura A.10: Classe *Artifacts*

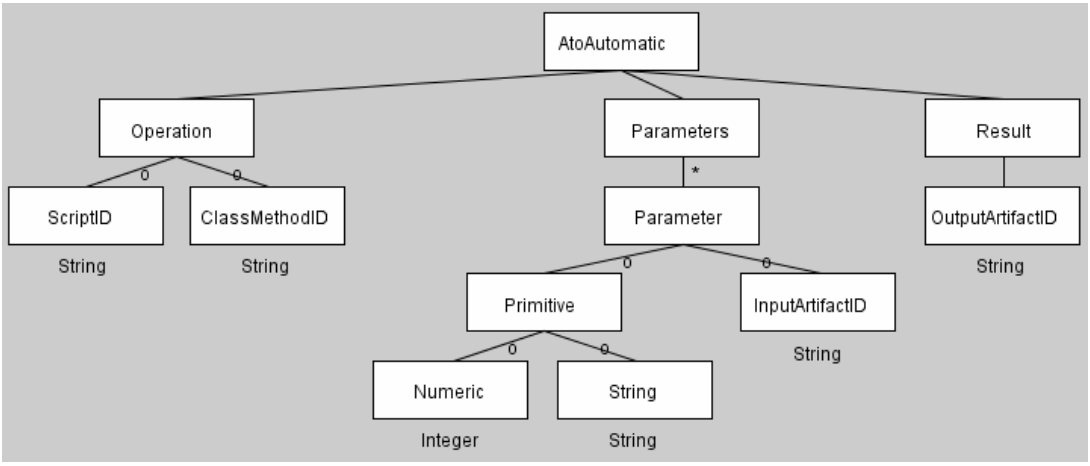

Figura A.11: Classe *Automatic*

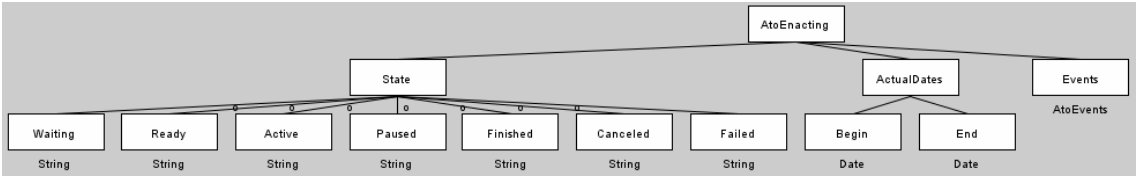

Figura A.12: Classe *Enacting*

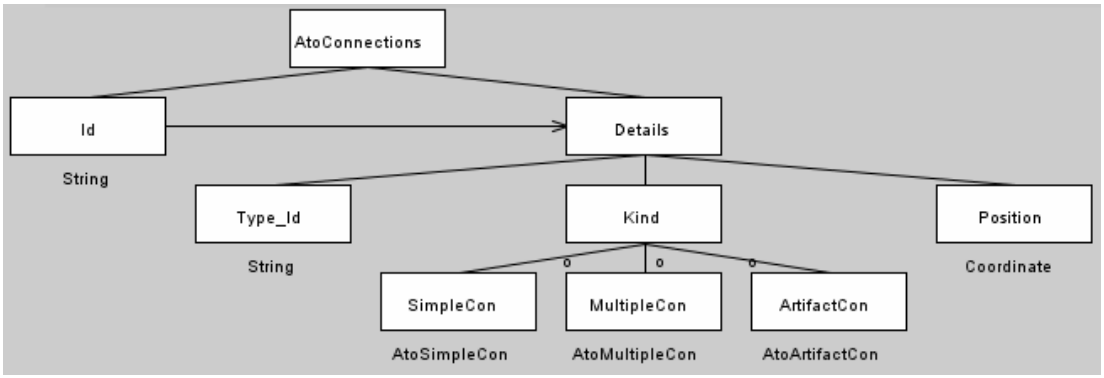

Figura A.13: Classe *Connections*

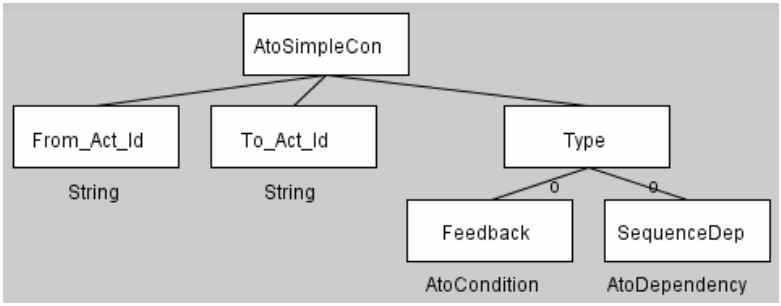

Figura A.14: Classe *SimpleCon*

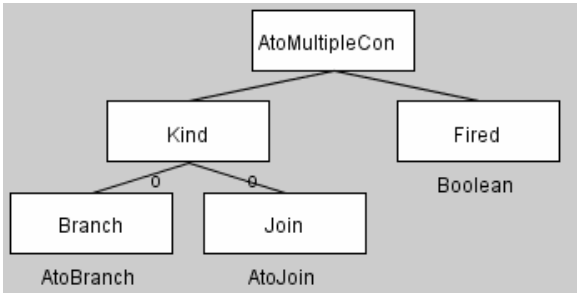

Figura A.15: Classe *MultipleCon*

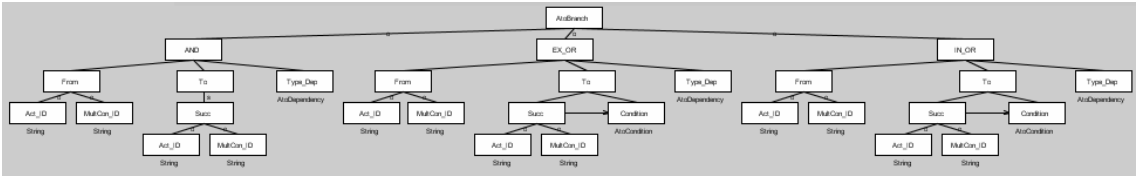

Figura A.16: Classe *Branch*

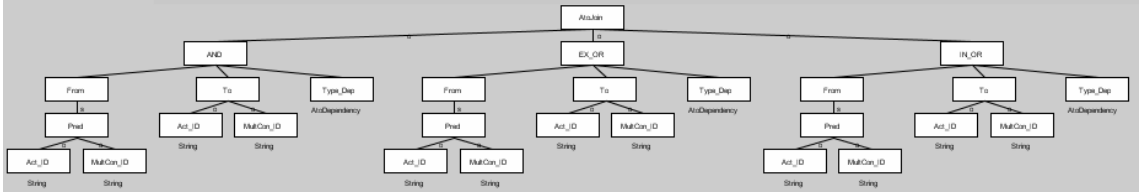

Figura A.17: Classe *Join*

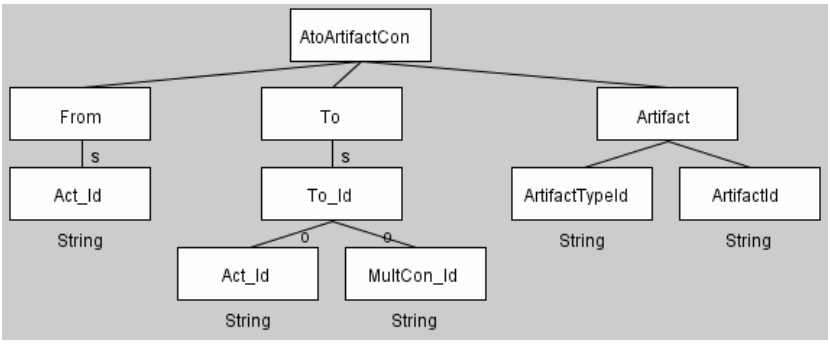

Figura A.18: Classe *ArtifactCon*

- **Tipo Artefatos de software**

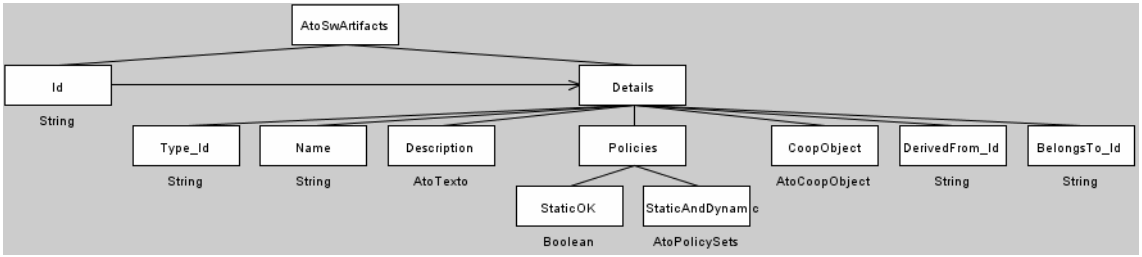

Figura A.19: Classe *SwArtifacts*