EDNA TOMIE TAKANO

UM MODELO DE GERENCIAMENTO DE PROCESSO DE *SOFTWARE* PARA O AMBIENTE DiSEN

MARINGÁ

2006

EDNA TOMIE TAKANO

UM MODELO DE GERENCIAMENTO DE PROCESSO DE *SOFTWARE* PARA O AMBIENTE DiSEN

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Itana Maria de Souza Gimenes

MARINGÁ

2006

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

T136m

Takano, Edna Tomie
Um modelo de gerenciamento de processo de software para o ambiente DiSEN / Edna Tomie Takano. -- Maringá : [s.n.], 2006.
93 f. : il. figs., tabs.

Orientador : Profª. Drª. Itana Maria de Souza Gimenes.
Dissertação (mestrado) - Universidade Estadual de Maringá. Programa de Pós-Graduação em Ciência da Computação, 2006.

1. Engenharia de software. 2. Processo de software. 3. Workflows interorganizacionais. 4. Serviços Web. 5. DiSEN (Ambiente de Desenvolvimento de Software Distribuído). I. Universidade Estadual de Maringá. Programa de Pós-Graduação em Ciência da Computação. II. Título.

005.1 CDD 21.ed.

EDNA TOMIE TAKANO

UM MODELO DE GERENCIAMENTO DE PROCESSO DE *SOFTWARE* PARA O AMBIENTE DiSEN

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Aprovado em 18/12/2006.

BANCA EXAMINADORA

Profa. Dra. Itana Maria de Souza Gimenes
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Elisa Hatsue Moriya Huzita
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Edmundo Sérgio Spoto
Centro Universitário Eurípides de Marília – PPGCC/UNIVEM

Dedico este trabalho

ao meu esposo, Edson, e

ao meu filho, Felipe.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que sempre esteve presente, por me dar àquilo que muitas vezes duvidei alcançar e que um dia irei confirmar que se tratava da graça divina. Obrigada pela força e proteção que me dedicou para que eu pudesse cumprir mais um objetivo em minha vida.

Agradeço ao meu esposo, Edson Yanaga, por todo o amor, carinho, compreensão e apoio, principalmente nos momentos mais difíceis; ao meu filho, Felipe Noriyo Yanaga, por ser a força que me permite continuar lutando pela vida; e ao meu cunhado Fabio Yanaga por ser um grande amigo para o meu filho.

Agradeço a todos os professores, que transmitiram seus conhecimentos e, portanto, tornaram possível a realização deste trabalho. Um agradecimento em especial à professora Elisa Hatsue Moriya Huzita, pela amizade, força e incentivo a mim demonstrados.

Reservo um especial agradecimento à minha orientadora, Itana Maria de Souza Gimenes. Obrigada pelo profissionalismo, dedicação, e principalmente, pela compreensão nestes últimos meses.

Agradeço à minha querida amiga, Lúcia Norie Matsueda Enami, pela amizade conquistada durante esta caminhada.

Agradeço a todos os funcionários do Departamento de Informática, em especial à Maria Inês Davanço, pela força, incentivo e paciência.

Não menos importante, agradeço, finalmente, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro oferecido a mim durante a realização deste trabalho.

RESUMO

Nas últimas décadas, as melhorias nas ferramentas e métodos para o desenvolvimento de *software* têm permitido que grupos de diferentes localidades e culturas, com diferentes expectativas possam formar uma equipe para trabalhar em projetos distribuídos. Neste contexto, surge o Ambiente de Desenvolvimento de *Software* Distribuído denominado DiSEN (*Distributed Software Engineering Environment*) que tem a finalidade de apoiar o desenvolvimento distribuído de *software*. Seu processo de *software* é constituído por atividades que estão dispersas em locais geográficos distintos. Porém, o ambiente DiSEN ainda não possui um mecanismo de gerenciamento de processo de *software* que permita a definição e execução das atividades de seus processos. Desse modo, este trabalho de mestrado tem como objetivo propor um modelo de gerenciamento de processo de *software* para o ambiente DiSEN. O modelo proposto consiste de um conjunto de componentes necessários para apoiar a definição e execução de um processo de *software* em um ambiente distribuído, como o DiSEN. Este modelo também possibilita a execução de algumas atividades do processo como serviços *Web*. Além do modelo proposto, foi realizado um estudo de caso ilustrando como um processo de *software* no ambiente DiSEN pode ser especificado utilizando a Linguagem de Execução de Processo de Negócio para Serviços *Web*, BPEL4WS (*Business Process Execution Language for Web Service*).

Palavras-Chaves: processo de *software*, *workflows* interorganizacionais, serviços *Web*, BPEL4WS, BPEL.

ABSTRAT

In the last decades software development tools and methods have been increasingly enhanced. This scenario enabled working teams from different places, cultures and expectations to work in distributed projects. The DiSEN (Distributed Software Engineering Environment) is an environment whose main objective is to support distributed software development. Its processes are composed of activities which are geographically distributed. However, DiSEN does not provide a mechanism for process management, as yet, that allows the definition and execution of the activities of its processes. Therefore, the objective of this work is to propose a model of software process management to DiSEN. The model consists of a set of components necessary to support process definition and execution in a distributed environment, such as DISEN. The proposed model allows the execution of some of the process activities as Web services. In addition, a case study was carried out to illustrate how a DiSEN software process could be specified with BPEL4WS (Business Process Execution Language for Web Services).

Keywords: software process, interorganizational workflow, Web Services, BPEL4WS, BPEL.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

BPEL4WS	*Business Process Execution Language for Web Service*
DDS	*Desenvolvimento Distribuído de Software*
DiSEN	*Ambiente de Desenvolvimento Distribuído de Software*
ExPSEE	*An Experimental Process Centred Software Engineering Environment*
SADT	*Structured Analisys and Desgn Technique*
SGPN	*Sistema Gerenciadores de Processos de Negócios*
SGWf-I	*Sistemas de Gerenciamento de Workflows Interorganizacionais*
SOAP	*Simple Object Access Protocol*
UDDI	*Universal Description, Discovery, and Integration*
UML	*Unified Modelling Language*
WfMC	*Workflow Management Coalition*
WSDL	*Web Services Description Language*
WSFL	*Web Services Flow Language*
XLANG	*Web Services for Business Process Design*
XML	*eXtendible Markup Language*

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

Há mais de quatro séculos atrás, Nicolau Maquiavel, o mais importante historiador, filósofo, dramaturgo, diplomata e cientista político italiano do Renascimento disse: "*...não há nada mais difícil de se dominar, não há nada mais perigoso de se conduzir ou mais incerto em seu sucesso, do que tomar a iniciativa de se introduzir uma nova ordem de coisas...*". Mesmo com as mudanças tecnológicas e o avanço da computação desde que Maquiavel emitiu esse pensamento, suas palavras continuam a soar verdadeiras.

Com o objetivo de obter custos menores e maior qualidade no processo de desenvolvimento de *software* (HERBSLEB, 1999), muitas organizações tomaram a iniciativa de introduzir uma "*nova ordem de coisas*" por meio do Desenvolvimento Distribuído de *Software* (DDS).

Ao introduzir o DDS, uma organização começa a enfrentar diversos desafios de adaptação, diferenças culturais, planejamento do trabalho e treinamento, além de todas as dificuldades inerentes ao processo de desenvolvimento centralizado. Essas mudanças estão causando um grande impacto não apenas no mercado de *software* propriamente dito, mas na maneira como os produtos de *software* estão sendo criados, modelados, construídos, testados e entregues aos clientes (HERBSLEB, 2001) (KAROLAK, 1998) (KIEL, 2003).

Os engenheiros de *software* têm reconhecido a grande vantagem da introdução do DDS no seu dia-a-dia e estão em busca de modelos que facilitem o desenvolvimento de *software* com equipes geograficamente distantes.

Dentro deste contexto, surge o Ambiente de Desenvolvimento de *Software* Distribuído – DiSEN (PASCUTTI, 2002). O ambiente DiSEN tem a finalidade de oferecer o suporte

necessário ao desenvolvimento distribuído de *software*. Seu processo de *software* é constituído por atividades que podem ser realizadas em locais geograficamente distintos. Neste processo, os desenvolvedores trabalham de forma cooperativa de acordo com uma metodologia para desenvolvimento distribuído de *software*.

Porém, nota-se que, o ambiente DiSEN ainda não possui um mecanismo de gerenciamento de processo distribuído de *software* que permita a definição e execução das atividades de seus processos. Esta é a principal motivação para a elaboração deste trabalho.

Desse modo, este trabalho tem como objetivo propor um modelo de gerenciamento de processo de *software* para o ambiente DiSEN. O modelo proposto consiste de um conjunto de componentes necessários para apoiar a definição e execução de um processo de *software* no ambiente DiSEN. Além disso, o modelo permite a execução de algumas atividades do processo como serviços *Web*, dada a atual importância desses serviços em ambientes computacionais. Neste contexto, foi também realizado um estudo de caso ilustrando como um processo de *software* no ambiente DiSEN pode ser especificado utilizando a Linguagem de Execução de Processo de Negócio para Serviços *Web* (BPEL4WS)[1], ou simplesmente, BPEL. Atualmente, não existe uma linguagem de especificação de processo de software para ambiente DISEN, e conceber uma dessas linguagens está fora do contexto deste trabalho.

Esta dissertação está organizada da seguinte maneira: o Capítulo 2 apresenta a fundamentação teórica necessária à compreensão e contextualização do trabalho desenvolvido; o Capítulo 3 apresenta o modelo de gerenciamento de processo de *software* proposto para o ambiente DiSEN; o Capítulo 4 apresenta a especificação de um processo de *software* no ambiente DiSEN utilizando BPEL; e por fim, o Capítulo 5 apresenta as conclusões e trabalhos futuros.

[1] *Business Process Execution Language for Web Service*

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo é apresentada a fundamentação teórica relevante para o desenvolvimento deste trabalho que inclui o Ambiente de Desenvolvimento Distribuído de *Software* – DiSEN, Processo de *Software* e *Workflows* Interorganizacionais. Também são apresentados os conceitos sobre serviços *Web* e BPEL.

Uma análise feita em Tanaka (1999) apresenta uma comparação entre os elementos do modelo da WfMC (*Workflow Management Coalition*) (WfMC, 1995) e do padrão *Process Manager* (GIMENES, 1999). A semelhança do modelo da WfMC com o padrão *Process Manager* indica que existem muitos elementos comuns nestes sistemas. Desse modo, a tecnologia de gerência do processo de *software* pode ser vista como um caso especial de *workflow*, especializado para tratar o desenvolvimento de *software* (OCAMPO e BOTELLA, 1998).

Ao invés de considerar o processo de *software* como diferenciado de outros processos de negócios, pode-se abordar este como uma especialização de processo de negócio, em que o negócio é o de desenvolvimento de *software*. Esta evidência apresenta o pressuposto de que um processo de *software* pode ser gerenciado pelo mesmo conjunto de mecanismos definidos para gerenciar *workflow*. Pode-se então especializar as funcionalidades de certos componentes do modelo de *workflow* de modo a tratar as particularidades inerentes ao processo distribuído de *software*, e ao mesmo tempo potencializar os benefícios proporcionados ao processo de *software* pelas técnicas já estabelecidas de *workflow*.

Durante os últimos anos, muitas pesquisas têm sido realizadas na área de gerenciamento de *workflow* interorganizacionais e em assuntos correlatos. A maior parte dos trabalhos realizados, especificamente, na área de *workflow* interorganizacionais foi realizada no final da década de 90 e início desta década. A partir do início desta década, muitos grupos de pesquisa passaram a adotar o termo “processos de negócio” de um modo mais geral, englobando tanto *workflow* intraorganizacionais como interorganizacionais (FANTINATO, 2005). Nesta perspectiva, os *workflows* interorganizacionais são inerentemente distribuídos e os sistemas de gerenciamento de processo de negócios atuais apoiam esta característica.

O desafio nessa área de aplicação é construir um sistema computacional capaz de oferecer apoio ao ciclo de vida inteiro de um processo de negócio. A tecnologia de serviços *Web* tem sido vista como uma solução para oferecer a infra-estrutura tecnológica necessária para que organizações interajam durante a realização de um determinado processo de negócio (PAPAZOGLOU e GEORGAKOPOULOS, 2003).

Nesse sentido, diversas linguagens para especificação de processos de negócio baseadas em serviços *Web* têm sido propostas, tais como a WSFL (*Web Service Flow Language*) proposta pela IBM, XLANG (*Web Service for Business Process Design*) proposta pela Microsoft, BPEL4WS proposta pela IBM, dentre outras. BPEL4WS foi a linguagem escolhida para a especificação de um processo de *software* no ambiente DiSEN.

As argumentações acima indicam que existe uma semelhança entre os propósitos do DiSEN e os sistemas de gerenciamento de processos de negócio, e que a tecnologia de serviços *Web* pode ser incorporada ao ambiente.

Este Capítulo apresenta a evolução desses conceitos. Do ponto de vista deste trabalho, processo é tratado como um caso particular de processo de negócio.

2.1 AMBIENTE DE DESENVOLVIMENTO DISTRIBUÍDO DE *SOFTWARE* - DISEN

Nas ultimas décadas, as melhorias nas ferramentas e métodos têm permitido que grupos de diferentes localidades e culturas, com diferentes expectativas possam formar uma equipe para trabalhar em projetos distribuídos.

Neste sentido, o Ambiente de Desenvolvimento Distribuído de *Software* - DiSEN (PASCUTTI, 2002) foi projetado para apoiar metodologias de desenvolvimento de *software* distribuído. Baseia-se em agentes e tanto a plataforma quanto as ferramentas que fazem parte deste ambiente levam em consideração aspectos de trabalho cooperativo, permitindo assim a interação de profissionais geograficamente dispersos.

2.1.1 ARQUITETURA DO AMBIENTE DISEN

A arquitetura proposta para o ambiente DiSEN segue o modelo em camadas, em que as camadas inferiores fornecem serviços para as camadas superiores, conforme mostra a Figura 2.1.

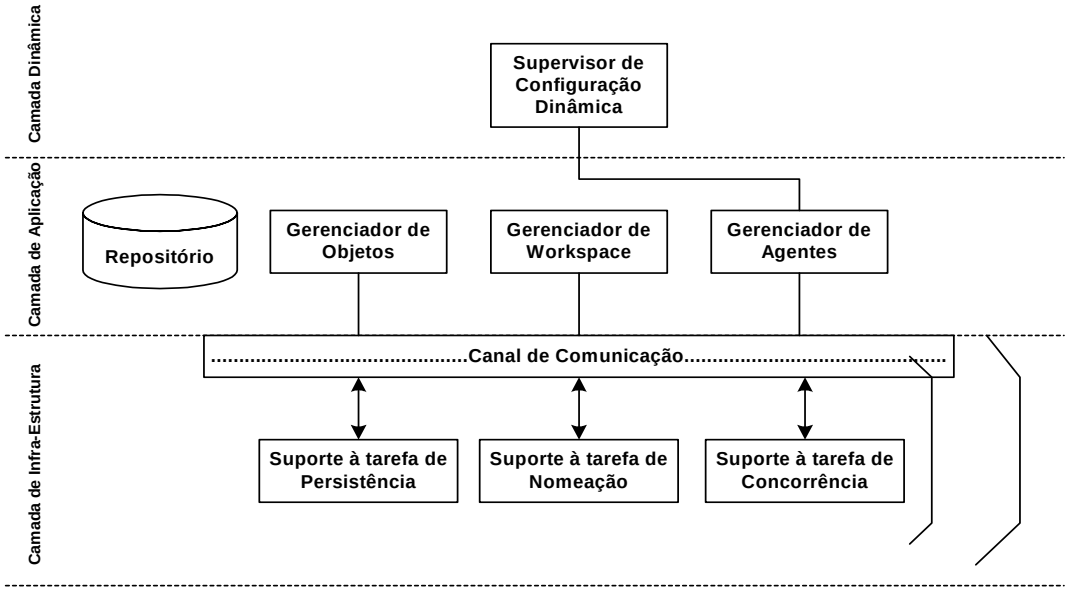

Figura 2.1: Arquitetura proposta para o ambiente DiSEN (Pascutti, 2002).

A Camada Dinâmica, composta pelo Supervisor de Configuração Dinâmica, é responsável pelo controle e gerenciamento da configuração do ambiente, bem como dos serviços que podem ser acrescentados ao ambiente em tempo de execução.

A Camada de Aplicação possui um:

- Gerenciador de Objetos, responsável pelo controle e gerenciamento do ciclo de vida dos artefatos (por exemplos: diagramas, modelos, manuais, códigos fonte e códigos objeto);
- Gerenciador de *Workspace*, responsável pelo controle e gerenciamento da edição cooperativa de documentos e itens de *software*;
- Gerenciador de Agentes, responsável pela criação, registro, localização, migração e destruição de agentes;
- Repositório, responsável pelo armazenamento dos artefatos, dados das aplicações geradas pelo Ambiente de Desenvolvimento de *software* e, também, do conhecimento necessário para a comunicação dos agentes;

A Camada de Infra-Estrutura oferece funcionalidades e suporte para as tarefas de nomeação, de persistência e de concorrência e, também, incorpora o canal de comunicação. O Canal de Comunicação é responsável pela comunicação entre os elementos da arquitetura.

2.2 PROCESSO DE *SOFTWARE*

A tecnologia de processo de *software* surgiu no final da década de 80 e representou um importante passo em direção à melhoria da qualidade de *software*, pois introduz mecanismos que permitem o gerenciamento automatizado do desenvolvimento de *software* (FEILER e HUMPHREY, 1993). Diversas teorias, conceitos, formalismos, metodologias e ferramentas surgiram nesse contexto, enfatizando a descrição de um modelo de processo de *software* que pode ser automatizado por um ambiente integrado de desenvolvimento de *software*.

O processo de *software* pode ser definido como sendo o conjunto de todas as atividades relacionadas ao desenvolvimento, controle, validação e manutenção de um *software* operacional. Estas atividades incluem: a determinação e especificação dos requisitos do *software*, a elaboração de protótipos, o projeto do *software*, a implementação, a verificação e validação do *software*, o controle de qualidade, a integração dos componentes, a documentação, a gestão de configurações e versões, a gerência do projeto e a avaliação do *software* (GIMENES, 1994).

2.2.1 MODELAGEM DO PROCESSO DE SOFTWARE

O modelo de processo de *software* é uma representação formal (abstração) dos elementos envolvidos no processo de *software*, sendo que esses elementos podem ser executados por pessoas ou máquinas. A maioria das descrições de ciclo de vida representam um modelo extremamente abstrato de desenvolvimento de *software* e não fornecem orientação sobre como integrar os vários passos de processo que são realizados pelas pessoas.

Um dos objetivos da modelagem de processo de *software* é decompor estas descrições em detalhe suficiente para orientar a execução do processo. Os modelos de processo de *software* concentram-se na representação dos elementos do processo de *software* e sua dinâmica. Esses modelos incluem cooperação entre os profissionais, iterações de atividades, políticas, restrições e alterações no estado dos produtos (GIMENES, 1994).

Os modelos podem ser representados por meio de técnicas como diagramas SADT[2], diagramas de módulos, diagramas de transição de estados, diagramas de fluxo de dados e regras de produção. Porém, a principal forma de representar processos de *software* é por meio de uma linguagem de programação de processo, ou linguagem de modelagem de processo, conforme proposto por Osterweil em (OSTERWEIL, 1987).

[2] *Structured Analisys and Desgn Technique* – Técnica de Análise e Projeto Estruturado (ROSS, 1985).

Para que um processo de *software* seja descrito adequadamente, muitos tipos de informação devem estar integrados. Freqüentemente, as informações encontradas em um modelo de processo são: o que será feito (quais as atividades), quem o fará (quais os agentes), quando, onde, como e por que será feito e quem é dependente de que cada passo seja feito. As linguagens de modelagem de processos diferem nas respostas dadas a estas questões e apresentam uma ou mais perspectivas diferentes relacionadas a estas respostas.

2.2.1.1 Requisitos para Modelagem do Processo de *Software*

Gimenes (1994) apresenta requisitos de modelos do processo de *software* divididos em duas partes. A primeira envolve as características inerentes ao processo de *software*. A segunda envolve características importantes para que abordagens específicas ofereçam boa cobertura e suporte ao processo de *software*. Os requisitos apresentados a seguir incluem aqueles previamente propostos por (HUMPHEY e KELLNER, 1989) (DOWSON, 1991).

Requisitos Inerentes ao Processo de Software

- **Critérios de transição e relações de precedência**: um processo se caracteriza por uma série de ações e um conjunto de estados de seus componentes. Assim, torna-se necessário garantir que o estado de um componente só possa ser modificado se as condições para esta transição forem satisfeitas. Também é necessário que seja possível definir relações de precedência, para se estabelecer prioridades de ações e critérios gerenciais;
- **Iteração e *feedback loop***: o processo de *software* não é monotônico e, usualmente, requer repetição de tarefas. Iteração em processo de *software* significa a repetição de trabalho necessária para garantir a correta evolução das atividades e o acerto de inconsistências encontradas na avaliação destas. *Feedback loop* representa a repetição de trabalho, que deve ser realizada para acertar as inconsistências

detectadas entre diferentes atividades ou fases, tais como inconsistência entre especificação e implementação;

- **Concorrência e não-determinismo**: muitas atividades de desenvolvimento de *software* podem ser realizadas concorrentemente. Também um certo grau de não-determinismo é necessário para representar atividades que não demandam uma estrita ordem de execução;
- **Comunicação entre os elementos do processo**: o processo de *software*, usualmente, envolve grandes grupos de projeto. A cooperação entre os membros do projeto exige um alto grau de comunicação.

Requisitos Adicionais para Modelagem do Processo de Software

- **Oferecer uma representação sintática e semântica para o processo de *software***: processos de *software* são difíceis de se representar informalmente ou por meio de representação puramente visual. Detalhes sobre os tipos dos objetos, métodos e restrições precisam ser especificados. Dessa forma, uma linguagem de programação de processos é necessária para que se possa especificar, compilar, interpretar e analisar processos de *software*;
- **Permitir a construção de mecanismos para automação do processo**: um modelo de processo de *software* deve permitir a construção de mecanismos automatizados para definição, validação e simulação de processos. Idealmente, o modelo deve estar integrado em um ambiente de suporte, onde um maior controle da execução do processo possa ser exercido;
- **Oferecer mecanismos de análise do processo**: é importante que o modelo do processo de *software* seja baseado num formalismo matemático, que permita análise

das propriedades do processo, tais como conectividade, tempo de duração e dependência das tarefas;

- **Permitir integração com gestão de projetos**: existe uma necessidade de integração da gestão de projetos com a gestão de processos, pois as funções inerentes a essas áreas são fortemente relacionadas;
- **Suportar múltiplas visões do processo**: é necessário que se possa acessar as informações do processo em níveis diferentes de granularidade e pontos de vista. O modelo deve possibilitar diferentes visões do processo, tais como: visão da atividade, visão do produto e visão da organização;
- **Permitir registro da história do processo**: os componentes do processo de *software* evoluem ao longo da execução do processo. Dessa forma, é importante manter um registro dos objetos e atividades durante a execução do processo. As informações a serem registradas incluem tempo de duração das atividades, grupo real de desenvolvimento, métodos e ferramentas utilizadas. Estas informações permitem auditoria e determinação de responsabilidades.

2.2.1.2 Requisitos de Linguagens de Modelagem de Processos de *Software*

A maioria das linguagens de modelagem de processos de *software* aplicam abordagens existentes e herdam características de linguagens de diferentes áreas, adicionando características específicas para aplicações de processo de *software*. Das linguagens de propósito geral são herdadas características como: abstração, modularidade e generalização. A fim de modelar apropriadamente o processo de *software*, as linguagens de processo também utilizam características das linguagens de sistemas concorrentes, reativos e de tempo real, tais como: paralelismo, especificação de restrições de tempo e descrições de interações com o ambiente.

Prover um modelo de dados conceitual, lidar com objetos persistentes de diferentes granulosidades e suportar transações longas e gerenciamento de versões são características que as linguagens de modelagem de processos de *software* herdam da área de banco de dados (ARMENISE, 1992) (TOYOTA, 1995).

Algumas das capacidades que uma linguagem de modelagem de processo deve possuir a fim de suportar o domínio de processos de *software* incluem (ROSA,1994):

- Persistência. A linguagem deve permitir a especificação de como o armazenamento persistente será implementado e como os objetos derivados serão manipulados;
- Relacionamento entre objetos. A linguagem de programação de processos deve prover mecanismos para descrever tais relacionamentos;
- Concorrência. A linguagem de modelagem de processos deve prover primitivas de concorrência, já que muitas atividades do processo de *software* podem ser realizadas concorrentemente. Um mecanismo de controle também é necessário para evitar o acesso conflitante a objetos compartilhados;
- Controle de transição. A linguagem deve permitir a especificação dos estados dos componentes, de modo a garantir que o estado de um objeto só possa ser modificado sob certas condições;
- Mecanismo para o disparo de atividades (*triggers*). Algumas atividades no processo de *software* podem ser ativadas automaticamente após o término de uma atividade ou em reação a ocorrência de um evento;
- Relações de precedência de atividades. É necessário especificar a ordem das atividades para estabelecer a prioridade das várias atividades que podem ser executadas;

- Não-determinismo. Algumas atividades no desenvolvimento de *software* não exigem uma determinada ordem em sua execução. Dessa forma, a linguagem de modelagem de processos deve permitir a representação desse não-determinismo;
- Modelo flexível de consistência. Durante o processo de *software* várias restrições podem ser incluídas, excluídas ou modificadas, conduzindo a inconsistência em produtos de *software*. É necessário, então, que a linguagem e modelagem de processos não permita que alterações nas restrições gerem inconsistências;
- Interface com o usuário que forneça diferentes visões do processo de *software*. Cada participante do projeto, de acordo com sua função, possui disponível determinados objetos e atividades. A interface deve ser amigável e permitir personalização;
- Transações longas. Durante o processo de *software* atividades, tais como implementação de um programa fonte, podem durar horas ou dias, assim a linguagem deve prover meios para execução de tais atividades;
- Modificações dinâmicas. O processo de *software* pode durar um longo tempo e está sujeito a mudanças durante o seu andamento. Assim, a linguagem de modelagem de processo de *software* deve ser flexível o suficiente para permitir modificações dinâmicas no processo, isto é, enquanto estiver sendo executado;
- Auxílio ao conhecimento incompleto. Muitas atividades do processo de *software*, como a análise de requisitos, não podem ser prescritas. A linguagem de modelagem de processos deve, portanto, oferecer auxílio inteligente à realização de parte da atividade, por meio de uma base conhecimento.

Os requisitos apresentados acima representam um conjunto mínimo de requisitos, essenciais para que uma linguagem possa servir para modelagem de processos de *software*.

2.3 *WORKFLOW* INTERORGANIZACIONAL

O conjunto de atividades a serem executadas por uma ou mais organizações para atingir um objetivo de negócio pode ser chamado de *workflow*. Um *workflow* pode ser intraorganizacional em que apenas uma organização está envolvida na execução de suas atividades, ou interorganizacional em que várias organizações estão envolvidas (FANTINATO, 2004). De um modo geral, a partir do início desta década, muitos grupos de pesquisas passaram a adotar o termo “processo de negócio”, englobando tanto *workflow* intraorganizacional como interorganizacional.

Muitas organizações estão atuando de forma cooperativa para atingir objetivos comuns de negócio, por meio da integração e automação de seus processos de negócio. Companhias estão cada vez mais se concentrando nas atividades chaves de sua área de negócio e subcontratando a realização de atividades secundárias. Assim, a Internet tem apresentado um grande potencial para permitir a realização de cooperações interorganizacionais, principalmente por meio do comércio eletrônico entre organizações virtuais disponíveis em mercados eletrônicos (FANTINATO, 2005).

Para que os processos de negócio, envolvendo várias organizações possam ser realizados na Internet deve ser fornecida a infra-estrutura tecnológica necessária para que tais cooperações sejam realizadas. Com esse propósito, são desenvolvidos os Sistemas Gerenciadores de Processos de Negócios (SGPN) ou Sistemas de Gerenciamento de *Workflows* Interorganizacionais (SGWf-I) que são extensões dos Sistemas de Gerenciamento de *Workflows* Intraorganizacionais (SGWf) (FANTINATO, 2004).

Nesta Seção, são apresentadas as principais características específicas de um SGPN, necessárias para apoiar a definição dos elementos do modelo de gerenciamento de processo de

software proposto. Uma das características fundamentais requeridas por SGPNs é a habilidade para apoiar as políticas e estratégicas das organizações participantes para mudanças dinâmicas nos processos de negócio cooperativos. As duas principais propriedades de um SGPN dinâmico são:

- Flexibilidade: o sistema deve permitir que sejam criadas e desfeitas cooperações entre organizações para a execução de um determinado processo de negócio, em tempo de execução;
- Adaptabilidade: o sistema deve permitir que os modelos de processo de negócios sejam facilmente modificáveis em tempo de execução para se adaptar a mudanças no ambiente de negócios.

Além das propriedades dinâmicas, outros requisitos funcionais que um SGPN deve satisfazer são apresentados a seguir:

- Definição e Simulação de Processos: o sistema deve oferecer mecanismos para que modelos de processos de negócios possam ser criados. Além disso, o sistema deve oferecer mecanismos para que o processo definido seja analisado de simulações de processos;
- Estabelecimento de Contratos Eletrônicos: o sistema deve disponibilizar um mecanismo para permitir que as organizações participantes cheguem a acordos sobre os detalhes dos processos de negócio a serem realizados de forma cooperativa e sobre os dados a serem trocados durante a execução desses processos;
- Instanciação e Execução de Processos: o sistema deve oferecer mecanismos para a invocação das atividades que fazem parte do modelo de processos de negócio;

- Administração do Sistema: o sistema deve disponibilizar mecanismos para sua própria administração, incluindo funções para configurar o ambiente, gerenciar usuários e direitos de acesso e alocar recursos;
- Monitoramento e Auditoria de Processos: o sistema deve possibilitar o monitoramento e a auditoria da execução de partes do processo de negócio em uma determinada organização por uma outra organização envolvida no processo de negócio;
- Verificação e Otimização do Desempenho: o sistema deve disponibilizar operações para que as organizações envolvidas acessem informações sobre o desempenho de processos em execução; e
- Apoio à Inteligência de Processos: o sistema deve disponibilizar ferramentas analíticas que aplicam a tecnologia de *Data Warehousing* e *Data Mining* na análise e otimização dos processos de negócios.

2.3.1 GERENCIAMENTO DE PROCESSO DE NEGÓCIO

O escopo da colaboração para a realização de um processo de negócio pode ser intra ou interorganizacional. No primeiro caso, o gerenciamento de processos pode provavelmente lidar com um ambiente homogêneo. Entretanto, no caso de colaborações interorganizacionais o gerenciamento de processos deve estar preparado para lidar com ambientes heterogêneos. A natureza heterogênea das organizações envolvidas agrega uma complexidade muito maior às operações de gerenciamento de processos de negócio, causando impacto nas atividades de modelagem de processo, de execução e de monitoramento entre outras.

O gerenciamento de processos de negócio pode ser realizado de duas formas principais (FANTINATO, 2005):

- Gerenciamento centralizado: quando o gerenciamento do processo de negócio é realizado por uma única organização. Essa forma de gerenciamento pode ser

aplicada a processos cujo modelo segue a topologia hierárquica, em que o gerenciamento de processos é chamado de **orquestração**. Nesse caso, o processo de mais alto nível oferece um único ponto de controle, e detalhes adicionais podem ser obtidos pelos processos de mais baixo nível na estrutura hierárquica. Assim, o SGPN pode ser disponibilizado completamente apenas na organização que vai gerenciar os processos de negócio, sendo chamado normalmente de servidor do SGPN. As outras organizações podem possuir uma infra-estrutura básica capaz apenas de executar suas partes do processo, e comunicar-se com o servidor do SGPN.

- Gerenciamento descentralizado: quando o gerenciamento do processo de negócio é realizado por várias organizações. Essa forma de gerenciamento pode ser aplicada a processos cujo modelo segue a topologia ponto-a-ponto, em que o gerenciamento de processo é chamado de **coreografia**. Nesse caso, todos os processos estão em um mesmo nível, precisando comunicar-se entre si para que as atividades de gerenciamento possam ser executadas. Assim, o SGPN deve ser disponibilizado completamente em todas as organizações envolvidas no processo de negócio. O gerenciamento descentralizado é bem mais complexo do que o gerenciamento centralizado, entretanto a maioria dos processos de negócio se encaixa na topologia ponto-a-ponto.

2.3.2 SISTEMAS GERENCIADORES DE PROCESSOS DE NEGÓCIOS BASEADOS EM SERVIÇOS WEB

Atualmente, os serviços *Web* estão emergindo como uma tecnologia promissora para a efetiva automação das interações interorganizacionais, por facilitar a descoberta e a invocação automática de serviços relevantes. Um serviço *Web* é uma aplicação disponível via Internet que pode ser utilizada para desempenhar um determinado serviço em um processo

cooperativo. Eles são baseados em novos e simples padrões, tais como XML[3], SOAP[4], WSDL[5] e UDDI[6], que oferecem a infra-estrutura necessária para apoiar a descrição, descoberta e composição de serviços *Web*.

O benefício essencial associado a tecnologia de serviços *Web* é a ampla padronização utilizada, já que os padrões podem ser utilizados de forma simples em qualquer plataforma de *software* e *hardware*. Desta forma, a integração de aplicações se torna mais fácil, já que todos os envolvidos acabam usando o mesmo padrão.

A seguir é apresentada uma breve descrição dos principais padrões utilizados pelos serviços *Web*:

- WSDL – *Web Services Description Language*: Uma descrição WSDL oferece toda a informação relacionada ao serviço *Web* necessária para a sua publicação, descoberta e invocação, incluindo capacidades, interface, comportamento e qualidade de serviço (WSDL, 2001);
- UDDI – *Universal Description, Discovery, and Integration*: Padrão que define a estrutura e conteúdo dos diretórios de serviços que contém as descrições dos serviços oferecidos (UDDI, 2001);
- XML – *eXtendible Markup Language*: Linguagem de marcação de dados que oferece um padrão para descrever dados estruturados, de modo a facilitar a declaração mais precisa de conteúdo e resultados mais significativos de busca (XML, 2004);
- SOAP – *Simple Object Access Protocol*: Protocolo utilizado para a troca de mensagens por meio da Internet, as quais podem ser usadas para a invocação

[3] *eXtendible Markup Language*
[4] *Simple Object Access Protocol*
[5] *Web Services Description Language*
[6] *Universal Description, Discovery, and Integration*

de serviços *Web* de acordo com suas interfaces descritas em WSDL ou para o registro e a procura de serviços *Web* via UDDI (SOAP, 2004);

Algumas arquiteturas de SGPN baseados em serviços *Web* foram analisadas e serviram de base para definir o modelo de gerenciamento de processo proposto para o ambiente DiSEN:

- O sistema Conductor (BURDETT e CHEN e HSU, 2002), desenvolvido pela empresa *Commerce One*, é apresentado como uma plataforma de Colaboração de Negócios baseada nos padrões de serviços *Web*. Esse sistema lida com dois tipos de serviços: as atividades humanas, chamadas de Serviços de Portal; e as aplicações de negócio, chamadas de Serviços *Web*.
- Ganesarajah e Lupo (2002) apresentam uma abordagem para arquiteturas de SGPN baseados em serviços *Web* em que praticamente todo o sistema é tratado como serviços *Web*, sendo na maioria deles serviços *Web* compostos.
- *Web*Transact (PIRES e BENEVIDES e MATTOSO, 2002) é apresentado como um *framework* que oferece a infra-estrutura necessária para a construção de composições confiáveis de serviços *Web*. O *framework Web*Transact é composto por uma arquitetura multicamadas, uma linguagem baseada em XML, chamada de *Web Services Transaction Language* (WSTL), e um modelo de transações.
- Em (McGREGOR e KUMARAN, 2002) é apresentada a arquitetura do sistema SMWS (*Solution Management Web Service*) que oferece apoio à definição de serviços *Web* a serem disponibilizados por organizações fornecedoras, com atenção especial à análise de desempenho da execução desses serviços *Web*.
- Em (ZHAO e LIU e YANG, 2004) é proposta uma arquitetura de SGPN baseado em serviços *Web* com ênfase em interoperabilidade, flexibilidade,

confiabilidade e escalabilidade interorganizacional. Esse sistema oferece apoio à execução de processos de negócio que seguem tanto a topologia hierárquica quanto a topologia ponto-a-ponto.

A arquitetura de Zhao, Liu e Yang foi escolhida para servir de base para a elaboração do modelo de gerenciamento de processo de *software* proposto. Esta escolha teve como base o fato desta arquitetura apresentar elementos semelhantes ao da arquitetura genérica de um Sistema de *Workflow* proposto pela WfMC (*Workflow Management Coaliton*) (2005) e, com o ambiente ExPSEE (GIMENES, 1999), que também serviram de base para a elaboração do modelo proposto para o ambiente DiSEN. Na Figura 2.2 é apresentada a arquitetura proposta por Zhao, Liu e Yang. O principal componente é o Gerenciador de Domínio de *Workflow* que se comunica com todos os outros componentes presentes nessa arquitetura. Os outros componentes da arquitetura são formados por serviços *Web*. Os serviços *Web* são usados tanto para as funções de gerenciamento do processo de negócio, quanto para a execução das atividades do processo de negócio.

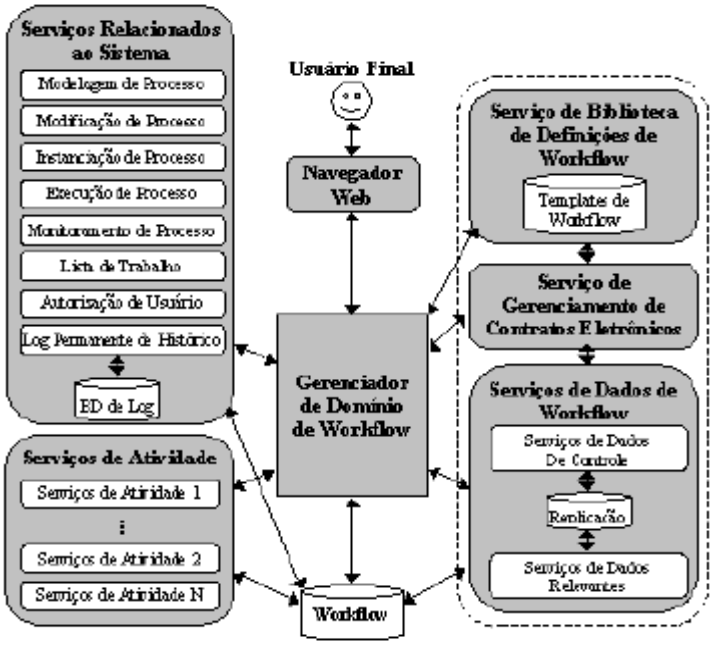

Figura 2.2: Arquitetura do SGPN de Zhao, Liu & Yang.

2.4 BPEL - LINGUAGEM DE EXECUÇÃO DE PROCESSOS DE NEGÓCIOS PARA SERVIÇOS *WEB*

Na última década surgiram numerosos formalismos e linguagens de modelagem, também chamadas de Linguagens de Modelagem de Processos (*Process Modeling Languages* – PMLs), que permitem representar, de maneira precisa e compreensível, várias das características ou elementos do processo de *software*. Com a introdução de processos de negócios, linguagens como BPEL demonstraram ser mais apropriadas para apoiar a modelagem de processos interorganizacionais.

BPEL é uma linguagem que permite compor e orquestrar diferentes serviços especificando toda a informação num documento XML. Com o conjunto de instruções XML de BPEL pode-se definir um processo com uma linguagem precisa e uma sintaxe estruturada, fornecendo um conjunto compreensivo e legível de informações que documentam um processo.

BPEL foi desenvolvida por meio da colaboração entre Microsoft, IBM e BEA, combinando XLANG – *Web Services for Business Process Design* (XLANG, 2006) uma linguagem de construções estruturais para processos proposta pela Microsoft, e WSFL – *Web Services Flow Language* (WSFL, 2006) uma linguagem de suporte para processos orientados à grafo proposta pela IBM.

Com BPEL é possível especificar processos de duas formas:

- Processos Executáveis: permite a definição completa de todas as partes de um processo e sua seqüência, que será controlada centralmente por uma máquina BPEL de orquestração.
- Processos Abstratos: permite a especificação pública da troca de mensagens, somente entre cada interveniente, sem a especificação do processo completo. Esta é a configuração para a coreografia.

Como uma linguagem de execução de processos, o papel de BPEL é definir um novo serviço *Web* por meio da composição de um conjunto de serviços existentes. Desta forma, BPEL4 é basicamente uma linguagem para descrever tal composição. A interface de um serviço composto é descrita como uma coleção de portTypes WSDL, assim como qualquer outro serviço *Web*. A composição (chamado de processo) indica como a interface de serviço se encaixa em uma execução completa de uma composição.

Um processo especificado em BPEL é composto por:

<*partnerLinks*>	Os serviços com que um processo de negócio interage é modelado como partnerLinks em BPEL. Cada partnerLinks é caracterizada por um partnerLinkType definindo os "papéis" executados por cada um dos serviços envolvidos e especificando o portType oferecido por cada serviço para receber mensagens.
<*variables*>	Define as variáveis de dados usadas pelo processo, oferecendo suas definições em termos de mensagem WSDL.
<*correlationsSets*>	No BPEL, o processo de encontrar uma instância adequada ou criar um, caso seja necessário, é chamado de correlação de mensagens. Quando mensagens são trocadas entre parceiros de negócio, elas tipicamente carregam dados que são usados para correlacionar a mensagem com o processo de negócio apropriado.
<*faultHandlers*>	Contém tratadores de falhas que definem as atividades que devem ser executadas quando elas acontecem.
<*compensationHandler*>	BPEL suporta a noção de compensação, que é uma técnica para permitir ao projetista do processo implementar ações de compensação para determinadas ações irreversíveis. Assim que uma reserva tenha sido confirmada, alguém precisa executar algumas operações explícitas para cancelar aquela reserva. Essas

	ações são chamadas "ações de compensação" para a ação original.
<*eventHandlers*>	Todo o processo como também cada extensão pode ser associada com um conjunto de tratadores de evento que são invocados simultaneamente se algum evento correspondente acontecer.

Um processo especificado em BPEL consiste basicamente de uma seqüência de atividades, como a expressão de um algoritmo. Existe uma coleção de atividades primitivas:

- invocar uma operação em algum serviço *Web* (<*invoke*>);
- aguardar uma mensagem de invocação (<*receive*>);
- gerar a resposta de uma operação de entrada/saída (<*reply*>);
- aguardar por algum tempo (<*wait*>);
- copiar dados de um lugar para outro (<*assign*>);
- indicar se houve uma exceção (<*throw*>);
- terminar a instância de serviço (<*terminate*>), ou
- realizar nada (<*empty*>).

As atividades primitivas podem ser combinadas em algoritmos mais complexos utilizando quaisquer das atividades estruturais fornecidas na linguagem. Estas atividades estruturais são:

- a habilidade de definir uma seqüência ordenada de atividades (<*sequence*>);
- a habilidade de possuir ramificações utilizando a expressão "caso-sentença" (<*switch*>);
- a habilidade de selecionar para execução um dentre vários caminhos alternativos (<*pick*>);

- a habilidade para indicar que uma atividade será repetida até que um certo critério de sucesso tenha sido encontrado (<*while*>);
- a habilidade para definir uma atividade aninhada com suas próprias variáveis associadas, tratadores de exceções e tratadores de compensações (<*scope*>); e por fim;
- a habilidade para indicar que uma coleção de atividades deve ser executada em paralelo (<*flow*>).

A linguagem BPEL permite combinar recursivamente as atividades estruturadas para expressar algoritmos complexos arbitrários que representam a implementação do processo. Estas atividades primitivas e estruturais podem ser vistas com detalhes em Anexo A.

2.5 CONSIDERAÇÕES FINAIS

Este Capítulo apresentou o Ambiente de Desenvolvimento de *Software* Distribuído - DiSEN como contexto em que o modelo de gerenciamento de processo de *software* é proposto; a tecnologia de processo de *software* como uma especialização de processo de negócio, o que permite tomar como base as abordagens de *workflow*s interorganizacionais para elaborar o modelo de gerenciamento de processo de *software* proposto para o ambiente DiSEN; e os conceitos básicos de BPEL, linguagem utilizada para especificar um processo de *software* no ambiente DiSEN.

CAPÍTULO 3

DESENVOLVIMENTO DE UM MODELO DE GERENCIAMENTO DE PROCESSO DE *SOFTWARE* PARA O AMBIENTE DiSEN

Este Capítulo apresenta o modelo de gerenciamento de processo de *software* proposto para o ambiente DiSEN, bem como as justificativas para a sua proposta e contribuições.

O modelo de gerenciamento de processo de *software* proposto para o ambiente DiSEN consiste de uma arquitetura composta por um conjunto de componentes e suas interconexões. Esta arquitetura toma como base a arquitetura genérica de um Sistema de *Workflow* proposto pela WfMC (*Workflow Management Coaliton*) (2005), os modelos de Sistemas de Gerenciamento de Processos de Negócios baseada em Serviços *Web* (FANTINATO, 2005), e a arquitetura do ambiente DiSEN proposta por Pascutti (2002).

Uma análise comparativa com o ambiente ExPSEE (GIMENES, 1999) também foi realizada para elaborar o modelo de gerenciamento proposto. O ExPSEE é uma ambiente de engenharia de *software* orientado a processos que visa oferecer mecanismos de apoio a modelagem e automação de processos. Os ambientes DiSEN e o ExPSEE possuem características semelhantes, o que contribuiu para a elaboração do modelo de gerenciamento de processo de *software* para o Ambiente DiSEN.

O modelo de gerenciamento de processo de *software* proposto foi elaborado a partir da especificação dos requisitos do ambiente DiSEN (ENAMI, 2006), tomando como base a abordagem de desenvolvimento baseado em componentes, a notação a linguagem UML (JACOBSON e BOOCH e RUMBAUGH, 1999). Em particular, foi seguido o processo de

desenvolvimento definido no método Catalysis (D'SOUZA, 1999). Como ferramenta de apoio à especificação foi utilizado Poseidon (POSEIDON, 2006).

Os quatro estágios do método Catalysis seguidos para elaboração do modelo proposto são:

- Especificação de Requisitos: visa entender e representar o problema definindo o contexto do sistema envolvido e produzindo um modelo do domínio;
- Especificação do Sistema: trata a solução de *software* extraída do modelo do domínio, representando-se os tipos e operações do sistema e seu comportamento exterior;
- Projeto da Arquitetura: vista em duas partes: a arquitetura da aplicação que representa a estrutura lógica do sistema como uma coleção de componentes cooperantes com os tipos e operações obtidos na especificação distribuídos por meio dos componentes, e a arquitetura técnica que inclui as partes do sistema independentes do domínio como a infra-estrutura de comunicação de componentes, a plataforma de hardware e a plataforma de *software*; e, por fim,
- Projeto Interno dos Componentes: envolve a definição das interfaces dos componentes.

As seções seguintes apresentam o desenvolvimento desses estágios para a obtenção do modelo proposto.

3.1 ESPECIFICAÇÃO DE REQUISITOS

Neste estágio é identificado o domínio da aplicação representando objetos e ações do domínio em questão. Os artefatos resultantes deste estágio são os Diagramas de Casos de Uso para cada um dos atores identificados, os quais são apresentados nas: Figura 3.1, Figura 3.2, Figura 3.3 e Figura 3.4. É interessante observar que, dada a complexidade em especificar o ambiente DiSEN, todos os diagramas de casos de uso são apresentados de forma detalhada.

O estágio de especificação de requisitos identificou quatro atores envolvidos no modelo de gerenciamento de processo de *software* para o ambiente DiSEN, que são: Gerente Geral, Gerente Local, Gerente de Projeto e Engenheiro de *Software*.

Os Gerentes Gerais são os responsáveis pela análise estratégica do ambiente e cuidam das atividades estratégicas.

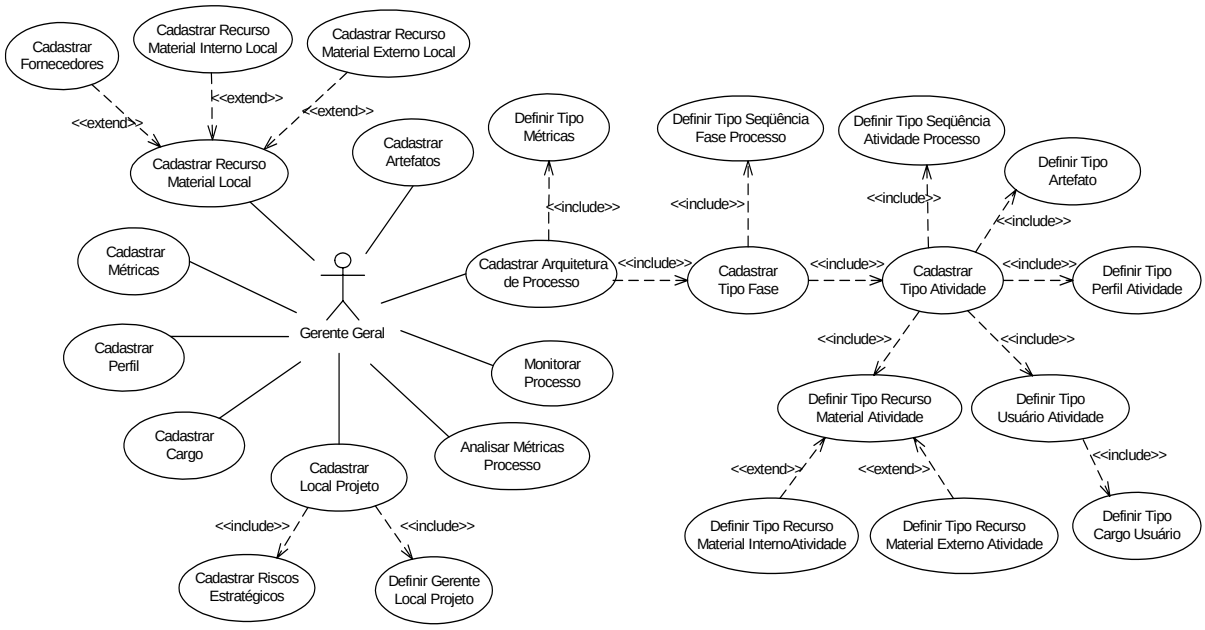

Figura 3.1: Diagrama de Casos de Uso para o ator Gerente Geral.

O diagrama de casos de uso para o ator Gerente Geral é composto pelos casos de uso descritos na Tabela 3.1.

Tabela 3.1: Descrição dos casos de uso para o ator Gerente Geral.

Caso de Uso	Descrição
Cadastrar Arquitetura de Processo	Permite inserir/atualizar/excluir as arquiteturas de processo criadas.
Cadastrar Tipo Fase	Permite inserir/atualizar/excluir as fases de uma arquitetura de processo.
Definir Tipo Seqüência Fase Processo	Permite determinar a seqüência das fases de uma arquitetura de processo.
Cadastrar Tipo Atividade	Permite inserir/atualizar/excluir as atividades de cada uma das fases de uma arquitetura de processo.

Definir Tipo Seqüência Atividade Processo	Permite determinar a seqüência das atividades de uma das fases de uma arquitetura de processo.
Definir Tipo Artefato	Permite determinar os artefatos a serem gerados pelas atividades de uma das fases de uma arquitetura de processo.
Definir Tipo Perfil Atividade	Permite determinar o perfil adequado para a realização de uma determinada atividade de uma das fases de uma arquitetura de processo.
Definir Tipo Usuário Atividade	Permite determinar os usuários necessários para a realização de uma determinada atividade de uma das fases de uma arquitetura de processo.
Definir Tipo Cargo Usuário	Permite determinar o cargo a serem ocupados pelos usuários, tais como analistas, engenheiros.
Definir Tipo Recurso Material Atividade	Permite determinar os recursos materiais necessários para a realização de uma determinada atividade de uma das fases de uma arquitetura de processo.
Definir Tipo Recurso Material Interno Atividade	Permite determinar os recursos materiais internos ao ambiente necessários para a realização de uma determinada atividade de uma das fases de uma arquitetura de processo.
Definir Tipo Recurso Material Externo Atividade	Permite determinar os recursos materiais externos ao ambiente necessários para a realização de uma determinada atividade de uma das fases de uma arquitetura de processo.
Definir Tipo Métricas	Permite determinar as métricas associadas a uma arquitetura de processo.
Cadastrar Local Projeto	Permite inserir/atualizar/excluir os projetos destinados a um determinado local.
Definir Gerente Local Projeto	Permite determinar os gerentes locais de cada unidade de trabalho (local).
Cadastrar Riscos Estratégicos	Permite inserir/atualizar/excluir e realizar uma análise dos riscos estratégicos de cada projeto local.
Cadastrar Recurso Material Local	Permite inserir/atualizar/excluir os recursos materiais a serem utilizados em uma determinada unidade (local).
Cadastrar Recurso Material Interno Local	Permite inserir/atualizar/excluir os recursos materiais internos ao ambiente para cada unidade (local).
Cadastrar Fornecedores	Permite inserir/atualizar/excluir os fornecedores dos recursos utilizados no ambiente.
Cadastrar Recurso Material Externo Local	Permite inserir/atualizar/excluir os recursos materiais externos ao ambiente para cada unidade (local).
Cadastrar Métricas	Permite inserir/atualizar/excluir as métricas associadas a um processo no ambiente

Analisar Métricas Processo	Permite realizar uma análise das métricas de um determinado processo.
Cadastrar Artefatos	Permite inserir/atualizar/excluir os artefatos gerados pelos projetos no ambiente.
Cadastrar Cargo	Permite inserir/atualizar/excluir os cargos existentes no ambiente a serem ocupados pelos usuários, tais como analistas, engenheiros.
Cadastrar Perfil	Permite inserir/atualizar/excluir o perfil de cada usuário do ambiente, que engloba a habilidade, o conhecimento e o treinamento que cada um possui.
Monitorar Processo	Permite monitorar todo o processo analisando possíveis problemas que venham a ocorrer durante a sua execução.

A elaboração do diagrama de caso de uso para o Gerente Geral propõe algumas sugestões de modificação dos nomes dos casos de uso apresentado por ENAMI (2006), conforme mostra a Tabela 3.2, e a inclusão de novos casos de uso, conforme mostra a Tabela 3.3.

Tabela 3.2: Sugestão de Modificação de Nomes de Casos de Uso para o Gerente Geral.

Caso de Uso (ENAMI, 2006)	Caso de Uso Modificado
Cadastrar Processo	*Cadastrar Arquitetura de Processo*
Cadastrar Fase	*Cadastrar Tipo Fase*
Definir Seqüência Fase Processo	*Definir Tipo Seqüência Fase Processo*
Cadastrar Atividade	*Cadastrar Tipo Atividade*
Definir Seqüência Atividade Processo	*Definir Tipo Seqüência Atividade Processo*
Associar Tipo Artefato	*Definir Tipo Artefato*
Associar Perfil Atividade	*Definir Tipo Perfil Atividade*
Associar Tipo Recurso Atividade	*Definir Tipo Recurso Material Atividade*
Associar Métricas	*Definir Tipo Métricas*
Definir Gerente Local	*Definir Gerente Local Projeto*
Cadastrar Recurso Material	*Cadastrar Recurso Material Local*
Cadastrar Fornecedor	*Cadastrar Fornecedores*

Cadastrar Tipo Artefato	*Cadastrar Artefatos*

Tabela 3.3: Sugestão de Inclusão de Casos de Uso para o Gerente Geral.

Caso de Uso (ENAMI, 2006)	Caso de Uso Incluso
Não contém caso de uso	*Definir Tipo Usuário Atividade*
Não contém caso de uso	*Definir Tipo Recurso Material Interno Atividade*
Não contém caso de uso	*Definir Tipo Recurso Material Externo Atividade*
Não contém caso de uso	*Cadastrar Local Projeto*
Não contém caso de uso	*Cadastrar Riscos Estratégicos*
Não contém caso de uso	*Cadastrar Recurso Material Interno Local*
Não contém caso de uso	*Cadastrar Recurso Material Externo Local*
Não contém caso de uso	*Analisar Métricas Processo*
Não contém caso de uso	*Cadastrar Cargo*
Não contém caso de uso	*Monitorar Processo*

Os Gerentes Locais, em nível tático, são responsáveis por cada local ou unidade geograficamente dispersa.

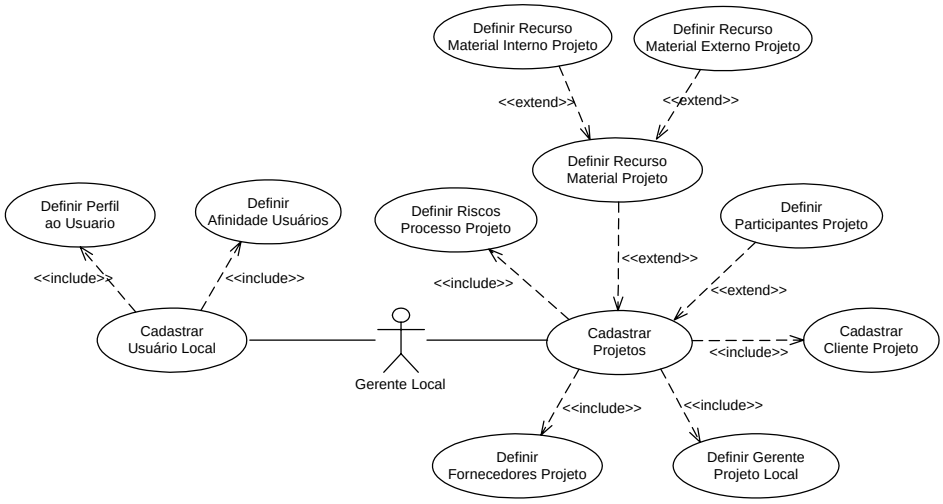

Figura 3.2: Diagrama de Casos de Uso para o ator Gerente Local.

O diagrama de casos de uso para o ator Gerente Local é composto pelos casos de uso descritos na Tabela 3.4.

Tabela 3.4: Descrição dos casos de uso para o ator Gerente Local.

Caso de Uso	Descrição
Cadastrar Projetos	Permite inserir/atualizar/excluir os projetos a serem desenvolvidos no local.
Definir Recurso Material Projeto	Permite determinar os recursos materiais a serem utilizados em projetos no local.
Definir Recurso Material Interno Projeto	Permite determinar os recursos materiais internos ao ambiente a serem utilizados em projetos no local.
Definir Recurso Material Externo Projeto	Permite determinar os recursos materiais externos ao ambiente a serem utilizados em projetos no local.
Definir Participantes Projeto	Permite determinar os participantes (usuários) a serem utilizados em projetos no local.
Definir Riscos Processo Projeto	Permite determinar os riscos preliminares que estão associados a não disponibilidade de recursos.
Cadastrar Cliente Projeto	Permite inserir/atualizar/excluir os clientes de um determinado projeto.
Definir Fornecedores Projeto	Permite determinar os fornecedores dos recursos utilizados em um determinado projeto.
Cadastrar Usuário Local	Permite inserir/atualizar/excluir cada usuário do local.
Definir Perfil Usuário	Permite determinar um perfil para cada usuário do local.
Definir Afinidade Usuário	Permite determinar afinidade entre usuários local.

A elaboração do diagrama de caso de uso para o Gerente Local propõe algumas sugestões de modificação dos nomes dos casos de uso apresentado por ENAMI (2006), conforme mostra a Tabela 3.5, e a inclusão de novos casos de uso, conforme mostra a Tabela 3.6.

Tabela 3.5: Sugestão de Modificação de Nomes de Casos de Uso para o Gerente Local.

Caso de Uso (ENAMI, 2006)	Caso de Uso Modificado
Cadastrar Projeto	*Cadastrar Projetos*
Associar Recurso Material Projeto	*Definir Recurso Material Projeto*
Associar Participante Projeto	*Definir Participantes Projeto*
Cadastrar Cliente	*Cadastrar Cliente Projeto*

Associar Fornecedor Projeto	*Definir Fornecedores Projeto*
Cadastrar Usuário	*Cadastrar Usuário Local*
Associar Perfil ao Usuário	*Definir Perfil Usuário*
Definir Afinidade	*Definir Afinidade Usuário*

Tabela 3.6: Sugestão de Inclusão de Casos de Uso para o Gerente Local.

Caso de Uso (ENAMI, 2006)	Caso de Uso Incluso
Não contém caso de uso	*Definir Recurso Material Interno Projeto*
Não contém caso de uso	*Definir Recurso Material Externo Projeto*

Os Gerentes de Projeto, também em nível tático, são responsáveis pelos projetos realizados em determinado local.

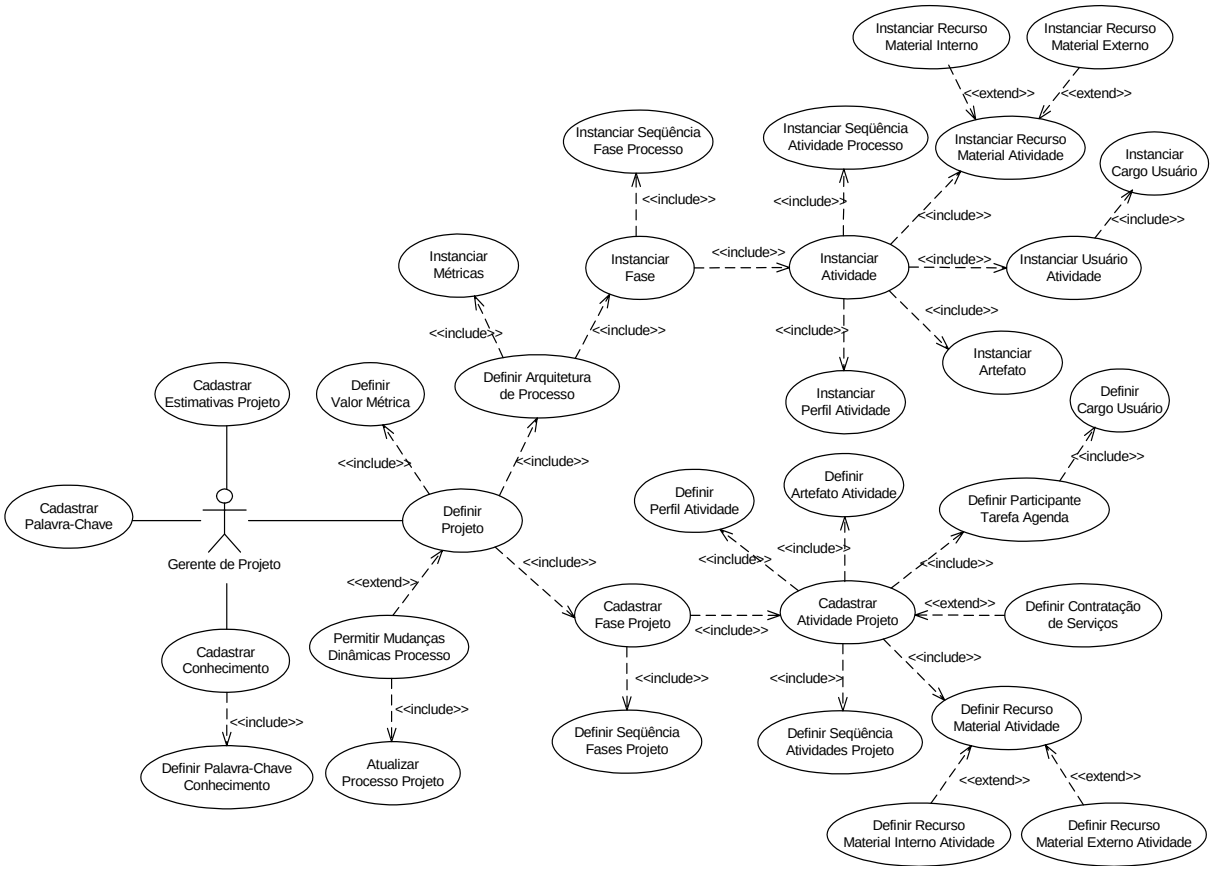

Figura 3.3: Diagrama de Casos de Uso para o ator Gerente de Projeto

O diagrama de casos de uso para o ator Gerente Projeto é composto pelos casos de uso descritos na Tabela 3.7.

Tabela 3.7: Descrição dos casos de uso para o ator Gerente Projeto.

Caso de Uso	Descrição
Definir Projeto	Permite determinar um projeto a ser desenvolvido em uma determinada unidade (local).
Definir Arquitetura de Processo	Permite determinar uma das arquiteturas de processo criadas.
Instanciar Fase	Permite criar as fases específicas de um processo definido.
Instanciar Seqüência Fase Processo	Permite criar uma seqüência específica das fases de um processo definido.
Instanciar Atividade	Permite criar as atividades específicas de cada uma das fases de um processo definido.
Instanciar Seqüência Atividade Processo	Permite criar uma seqüência específica das atividades de uma das fases de um processo definido.
Instanciar Artefato	Permite criar os artefatos específicos a serem gerados pelas atividades de uma das fases de um processo definido.
Instanciar Usuário Atividade	Permite criar os usuários específicos necessários para a realização de uma determinada atividade de uma das fases de um processo definido.
Instanciar Cargo Usuário	Permite criar os cargos específicos existentes no ambiente a serem ocupados pelos usuários, tais como analistas, engenheiros.
Instanciar Perfil Atividade	Permite criar o perfil específico adequado para a realização de uma determinada atividade de uma das fases de um processo definido.
Instanciar Recurso Material Atividade	Permite criar os recursos materiais específicos necessários para a realização de uma determinada atividade de uma das fases de um processo definido.
Instanciar Recurso Material Interno Atividade	Permite criar os recursos materiais internos específicos ao ambiente necessários para a realização de uma determinada atividade de uma das fases de um processo definido.
Instanciar Recurso Material Externo Atividade	Permite criar os recursos materiais externos específicos ao ambiente necessários para a realização de uma determinada atividade de uma das fases de um processo definido.
Instanciar Métricas	Permite criar as métricas específicas associadas a um processo definido.
Cadastrar Fases Projeto	Permite inserir/atualizar/excluir as fases de um processo instanciado para um determinado projeto.
Definir Seqüência Fases Projeto	Permite determinar a seqüência das fases de um processo

	instanciado para um determinado projeto.
Cadastrar Atividade Projeto	Permite inserir/atualizar/excluir as atividades de cada uma das fases de um processo instanciado para um determinado projeto.
Definir Seqüência Atividades Projeto	Permite determinar a seqüência das atividades de uma das fases de um processo instanciado para um determinado projeto.
Definir Artefato Atividade	Permite determinar os artefatos a serem gerados pelas atividades de uma das fases de um processo instanciado para um determinado projeto.
Definir Participante Tarefa Agenda	Permite determinar os participantes para a realização de uma determinada atividade de uma das fases de um processo instanciado para um determinado projeto.
Definir Cargo Usuário	Permite determinar os cargos existentes no ambiente a serem ocupados pelos usuários, tais como analistas, engenheiros, para um determinado projeto.
Definir Perfil Atividade	Permite determinar o perfil adequado para a realização de uma determinada atividade de uma das fases de um processo instanciado para um determinado projeto.
Definir Recurso Material Atividade	Permite determinar os recursos materiais necessários para a realização de uma determinada atividade de uma das fases de um processo instanciado para um determinado projeto.
Definir Recurso Material Interno Atividade	Permite determinar os recursos materiais internos ao ambiente necessários para a realização de uma determinada atividade de uma das fases de um processo instanciado para um determinado projeto.
Definir Recurso Material Externo Atividade	Permite determinar os recursos materiais externos ao ambiente necessários para a realização de uma determinada atividade de uma das fases de um processo instanciado para um determinado projeto.
Definir Contratação de Serviços	Permite determinar a contratação de serviços realizados por outras organizações para um determinado projeto.
Cadastrar Estimativas Projeto	Permite inserir/atualizar/excluir as estimativas do projeto.
Definir Valor Métrica	Permite determinar quais métricas além das que já existem para o processo devem ser realizadas.
Permitir Mudanças Dinâmicas Processo	Permite que alterações no processo e projeto sejam realizadas em tempo de execução.
Atualizar Processo Projeto	Permite atualizar o processo ou projeto em tempo de execução.
Cadastrar Conhecimento	Permite inserir/atualizar/excluir os conhecimentos adquiridos com o desenvolvimento de projetos, cadastrando os conhecimentos.
Cadastrar Palavra-Chave	Permite inserir/atualizar/excluir as palavras-chaves a serem

	associadas aos conhecimentos.
Definir Palavra-Chave Conhecimento	Permite determinar palavras-chaves aos conhecimentos.

A elaboração do diagrama de caso de uso para o Gerente Projeto propõe algumas sugestões de modificação dos nomes dos casos de uso apresentado por ENAMI (2006), conforme mostra a Tabela 3.8, e a inclusão de novos casos de uso, conforme mostra a Tabela 3.9.

Tabela 3.8: Sugestão de Modificação de Nomes de Casos de Uso para o Gerente Projeto.

Caso de Uso (ENAMI, 2006)	Caso de Uso Modificado
Definir Processo	*Definir Arquitetura de Processo*
Associar Participante Atividade	*Definir Participante Tarefa Agenda*
Associar Recurso Material Tarefa	*Definir Recurso Material Atividade*
Associar Tipo Artefato	*Definir Artefato Atividade*
Associar Perfil Atividade	*Definir Perfil Atividade*
Associar Palavra-Chave Conhecimento	*Definir Palavra-Chave Conhecimento*

Tabela 3.9: Sugestão de Inclusão de Casos de Uso para o Gerente Projeto.

Caso de Uso (ENAMI, 2006)	Caso de Uso Incluso
Não contém caso de uso	*Definir Projeto*
Não contém caso de uso	*Instanciar Fase*
Não contém caso de uso	*Instanciar Seqüência Fase Processo*
Não contém caso de uso	*Instanciar Atividade*
Não contém caso de uso	*Instanciar Seqüência Atividade Processo*
Não contém caso de uso	*Instanciar Artefato*
Não contém caso de uso	*Instanciar Usuário Atividade*
Não contém caso de uso	*Instanciar Cargo Usuário*
Não contém caso de uso	*Instanciar Perfil Atividade*
Não contém caso de uso	*Instanciar Recurso Material Atividade*

Não contém caso de uso	*Instanciar Recurso Material Interno Atividade*
Não contém caso de uso	*Instanciar Recurso Material Externo Atividade*
Não contém caso de uso	*Instanciar Métricas*
Não contém caso de uso	*Definir Cargo Usuário*
Não contém caso de uso	*Definir Recurso Material Atividade*
Não contém caso de uso	*Definir Recurso Material Interno Atividade*
Não contém caso de uso	*Definir Recurso Material Externo Atividade*
Não contém caso de uso	*Definir Contratação de Serviços*
Não contém caso de uso	*Cadastrar Estimativas Projeto*
Não contém caso de uso	*Permitir Mudanças Dinâmicas Processo*
Não contém caso de uso	*Atualizar Processo Projeto*

Os Engenheiros de *Software*, em nível operacional, são responsáveis pela execução do processo, ou seja, realização das atividades do processo. Dentre os engenheiros de *software*, incluem os analistas de sistemas, os programadores, os técnicos, entre outros.

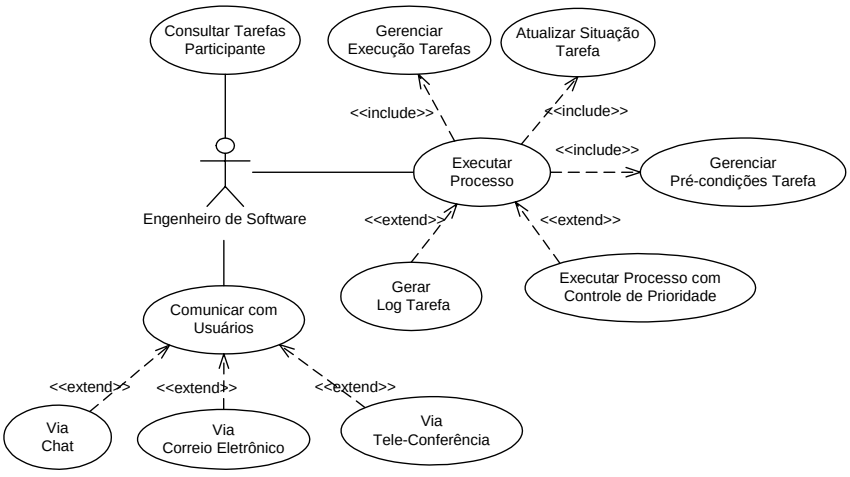

Figura 3.4: Diagrama de Casos de Uso para o ator Engenheiro de *Software*.

O diagrama de casos de uso para o ator Engenheiro de *Software* é composto pelos casos de uso descritos na Tabela 3.10.

Tabela 3.10: Descrição dos casos de uso para o ator Engenheiro de *Software*.

Caso de Uso	Descrição
Executar Processo	Permite iniciar a execução do processo de um determinado projeto.
Gerenciar Pré-Condições Tarefa	Permite gerenciar as pré-condições necessárias para que uma tarefa seja realizada.
Executar Processo com Controle de Prioridade	Permite executar as tarefas com ordem de prioridade.
Gerenciar Execução Tarefas	Permite gerenciar a execução das tarefas de um determinado projeto.
Atualizar Situação Tarefa	Permite gerenciar a situação (estado) das tarefas sendo realizadas, tais como, executar, interromper, reiniciar, cancelar e finalizar.
Gerar Log Tarefa	Permite descrever os problemas ocorridos durante o desenvolvimento de uma determinada atividade
Consultar Tarefas Participantes	Permite consultar a agenda de tarefas dos participantes envolvidos em um determinado projeto.
Comunicar com Usuários	Permite a comunicação dos participantes por meio de via tele-conferência, via correio eletrônico e via *chat*.

A elaboração do diagrama de caso de uso para o Engenheiro de *Software* propõe algumas sugestões de inclusão de novos nomes dos casos de uso, conforme mostra a Tabela 3.11.

Tabela 3.11: Sugestão de Inclusão de Casos de Uso para o Engenheiro de *Software*.

Caso de Uso (ENAMI, 2006)	**Caso de Uso Incluso**
Não contém caso de uso	*Executar Processo*
Não contém caso de uso	*Gerenciar Pré-Condições Atividade*
Não contém caso de uso	*Executar Processo com Controle de Prioridade*
Não contém caso de uso	*Gerenciar Execução Tarefas*
Não contém caso de uso	*Comunicar com Usuários*

Convém lembrar que a estrutura organizacional para os projetos é flexível. Em cada projeto pode haver uma troca de papéis. Um gerente local pode ser um gerente de projeto e,

um gerente local pode ser o gerente geral. De um modo geral, o gerente geral é um dos gerentes locais. Só pode haver um gerente geral que possui privilégios de consultar todas as informações sobre todos os projetos da organização. O gerente local é o responsável por cada local ou unidade administrativa geograficamente dispersa. O gerente de projetos é o responsável por um projeto específico e o engenheiro de *software* é aquele que irá desenvolver as atividades do projeto (ENAMI, 2006).

3.2 ESPECIFICAÇÃO DO SISTEMA

Neste estágio é tratada a modelagem da solução de *software* identificada nos modelos de domínio obtidos na fase de especificação de requisitos. A análise das ações do sistema representadas nos diagramas de caso de uso torna possível a identificação dos tipos, permitindo a associação das referidas ações aos respectivos tipos relacionados. Um tipo em Catalysis não é o mesmo que uma classe. Uma classe descreve a implementação de um objeto, enquanto um tipo descreve as características comuns a um conjunto de objetos. Dessa maneira, os tipos são especificações de comportamento dos objetos que documentam e mostram somente a visão externa de um determinado objeto.

O artefato mais importante produzido neste estágio é o Modelo Estático de Tipos, conforme mostrada na Figura 3.5. O modelo estático de tipos foi elaborado com base no padrão *Process Manager* (WEISS, 1998) que contém um diagrama de classes para o gerenciador de processos do ambiente ExPSEE, e no modelo de gerenciamento de projeto para o ambiente DiSEN proposto por ENAMI (2006).

Como pode ser observado na Figura 3.5, as partes central e inferior do diagrama estático de tipos foram definidas com base no conceito de reutilização de arquiteturas de processo. A parte central envolve a definição de arquiteturas de processo e os tipos de objetos relacionados a esta arquitetura, enquanto a parte inferior envolve a instanciação de

arquiteturas de processo já definidas, bem como os seus tipos de objetos. A parte superior representa os tipos referentes aos módulos de gerenciamento.

De acordo com o diagrama de tipos, uma arquitetura de processo pode possuir uma ou mais fases de processo, uma fase pode possuir um ou mais tipos de atividades, artefatos, recursos ou cargos. A arquitetura tem um relacionamento direto com os tipos de fases do processo e indireto, por meio dos tipos de fases, com os demais tipos do diagrama, que são recursos e restrições para a sua realização.

Na parte central do diagrama, existe uma relação entre os tipos envolvidos na arquitetura. Um tipo de fase pode ser precedido e/ou composto por zero ou mais tipos de fases. Uma vez definido as fases da arquitetura de processo é necessário definir as suas seqüência. O tipo de fase contém uma ou mais atividades. Uma vez definido as atividades das fases, também é necessário definir as suas seqüência. Este tipo de atividade pode utilizar um ou mais tipos de artefatos na produção de um ou mais tipos de artefatos. Esta ação de transformação pode requerer o emprego de um ou mais tipos de cargos. Além disso, esta ação deverá ser realizada por um ou mais tipos de cargos. Um tipo de cargo define os direitos e deveres de um cargo, que será ocupado por um ou mais atores na parte inferior do diagrama. Um tipo de cargo pode ser liderado por outro tipo de cargo. Como parte de seus direitos, os tipos de cargos podem fazer uso de um ou mais tipos de recursos, e/ou manipular um ou mais tipos de artefatos, segundo uma lista de direitos (ex. leitura, escrita e/ou execução). Tipos de recursos podem operar sobre tipos de artefatos segundo uma lista de tipos de ações válidas sobre estes tipos de artefatos.

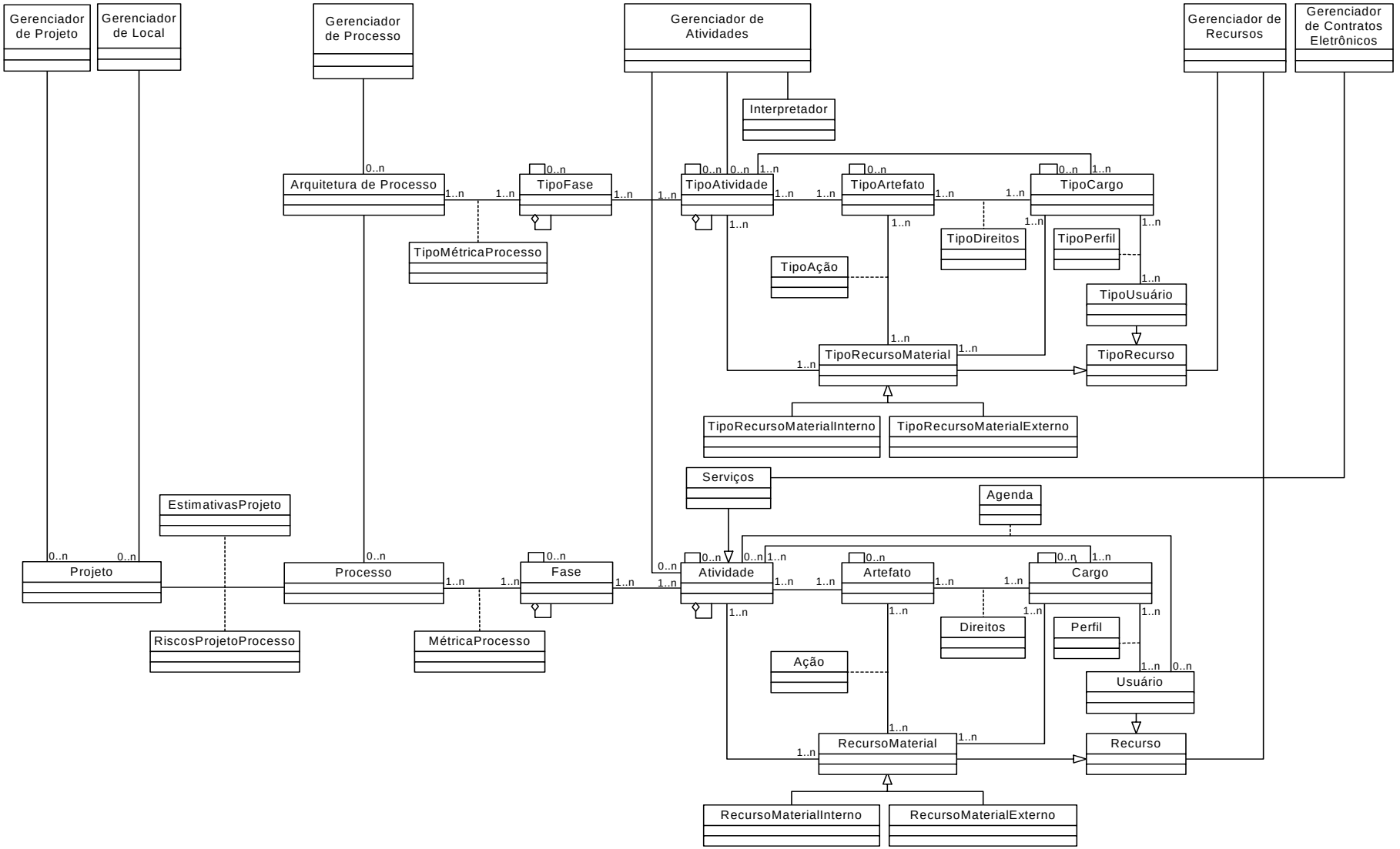

Figura 3.5: Modelo Estático de Tipos.

Os objetos presentes na parte inferior do diagrama são instâncias dos tipos definidos na arquitetura de processo na parte central do diagrama. Dessa forma, os relacionamentos entre esses objetos comportam-se da mesma maneira descrita para a arquitetura. Assim, nesta etapa tem-se um processo definido, com suas fases, atividades ou serviços que podem ser realizados por outras organizações, artefatos, recursos, ações e cargos. Além disso, tem-se também os atores (Usuário) responsáveis pelo desenvolvimento do processo. Cada ator tem associado a si uma agenda onde estão relacionadas as atividades destinadas a ele e os dados mais importantes relacionados a estas atividades. Outro ponto diferencial entre esta etapa e a etapa de definição da arquitetura de processo é o fato de que a lista de tipos e ações que um tipo de recurso pode executar sobre um tipo deverá ser refinada para uma lista de ações contendo os parâmetros necessários para a realização de ações efetivas sobre os artefatos.

3.3 PROJETO DA ARQUITETURA

O projeto da arquitetura de um sistema de *software* visa definir suas estruturas gerais, descrevendo os elementos que compõem os sistemas e as interações entre estes apoiando também questões importantes de projeto, como, por exemplo, a organização do sistema como composição de componentes.

Neste estágio é realizado um refinamento desde os artefatos de mais alto nível até à arquitetura do sistema. O Catalysis considera os pacotes como unidade de decomposição de mais alto nível, sendo que eles podem ser considerados partes do sistema e serem tratados como unidades independentes e candidatas a se tornarem componentes no estágio seguinte.

Para a elaboração dos artefatos gerados neste estágio tomou-se como base alguns elementos da arquitetura proposta por Zhao, Liu & Yang, tais como o componente Gerenciador de Domínio de *Workflow*, responsável pelo gerenciamento dos processos de negócios, incluindo sua definição e execução, e pela coordenação dos outros componentes; o componente Serviço de Gerenciamento de Contratos Eletrônicos, responsável pelas atividades

de comunicação interorganizacional; e o componente Serviços Relacionados ao Sistema, que oferecem as principais funções de gerenciamento de processo para o Gerenciador de Domínio de *Workflow*.

O primeiro passo para realizar o particionamento em pacotes consiste em analisar as dependências existentes entre os tipos representados na Figura 3.5. Uma visão do modelo de negócios é representada pelo Diagrama de Colaboração de Alto Nível apresentado na Figura 3.6.

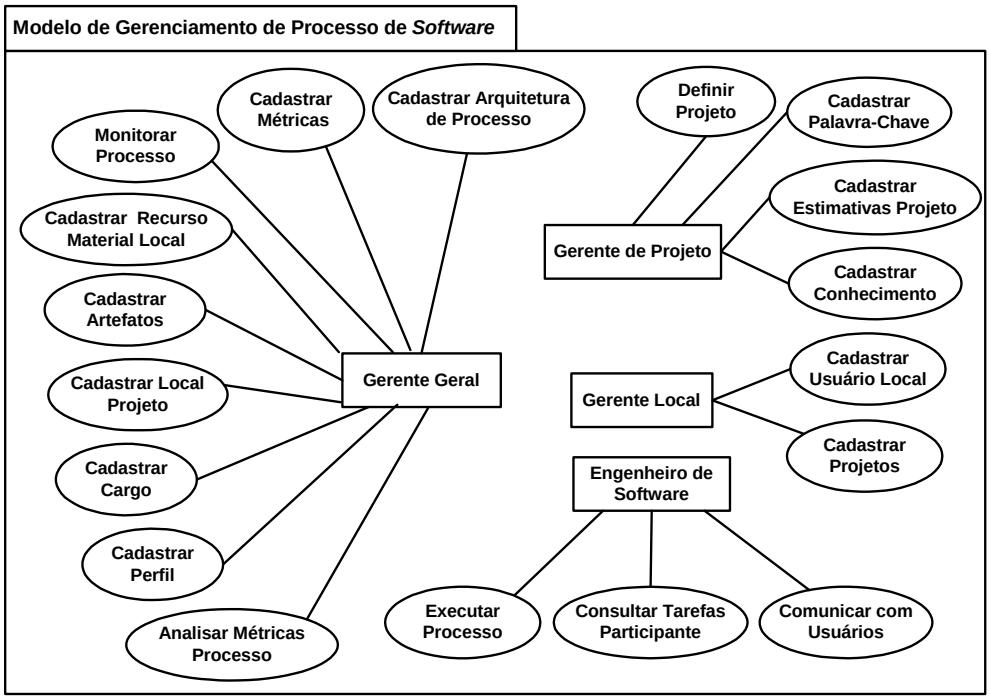

Figura 3.6: Diagrama de Colaboração de Alto Nível.

O particionamento dos pacotes segue a abordagem de camadas verticais e horizontais proposta por Catalysis, conforme diagrama mostrado na Figura 3.7 . As camadas verticais representam o particionamento em nível de negócio, de acordo com as ações compreendidas pelos principais atores que interagem com o modelo. Esses atores são o gerente geral, que define as arquiteturas reutilizáveis de processo; o gerente local que define quais projetos devem ser realizados em determinados locais; o gerente de projeto que controla a

instanciação, alocação de recursos e agendamento das atividades para o processo; e o engenheiro de *software*, que executa as atividades do processo.

As camadas horizontais representam o particionamento da arquitetura, separando os pacotes desde os de mais alto nível até os pacotes de infra-estrutura. Este particionamento identifica pacotes de serviço que são compartilhados com os pacotes de mais alto nível. Esta divisão deve buscar a otimização da importação de pacotes.

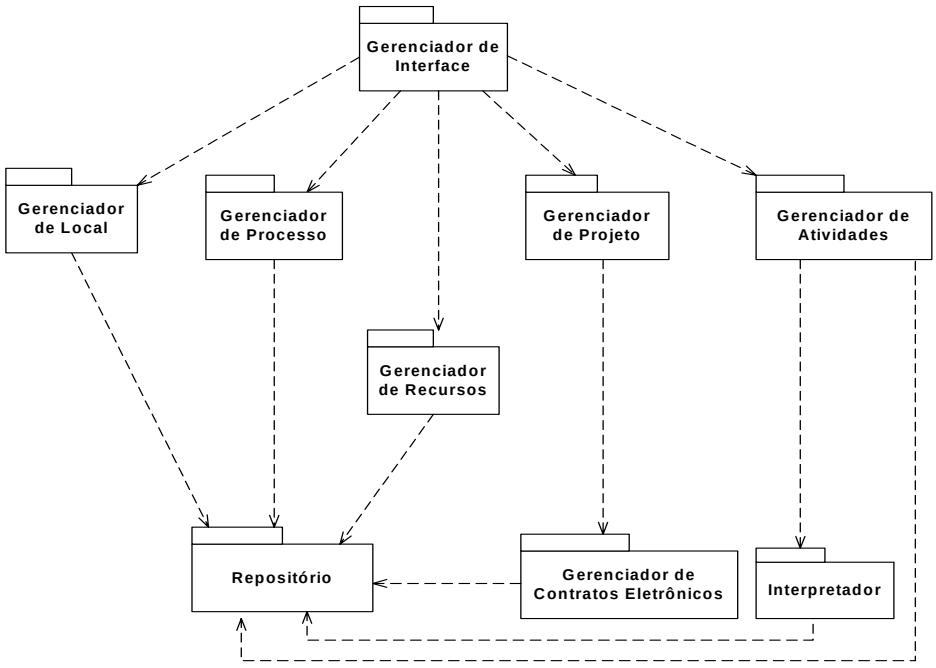

Figura 3.7:Diagrama de Camadas Verticais de Alto Nível.

A partir do diagrama de camadas verticais de alto nível foi possível projetar o Diagrama de Camadas Verticais. A construção deste modelo tomou como base os pacotes definidos anteriormente pelo Diagrama de Camadas Verticais de Alto Nível juntamente com a especificação dos tipos de cada pacote e o relacionamento entre eles. Na Figura 3.8 é apresentado o Diagrama de Camadas Verticais, juntamente com a especificação de tipos de cada pacote.

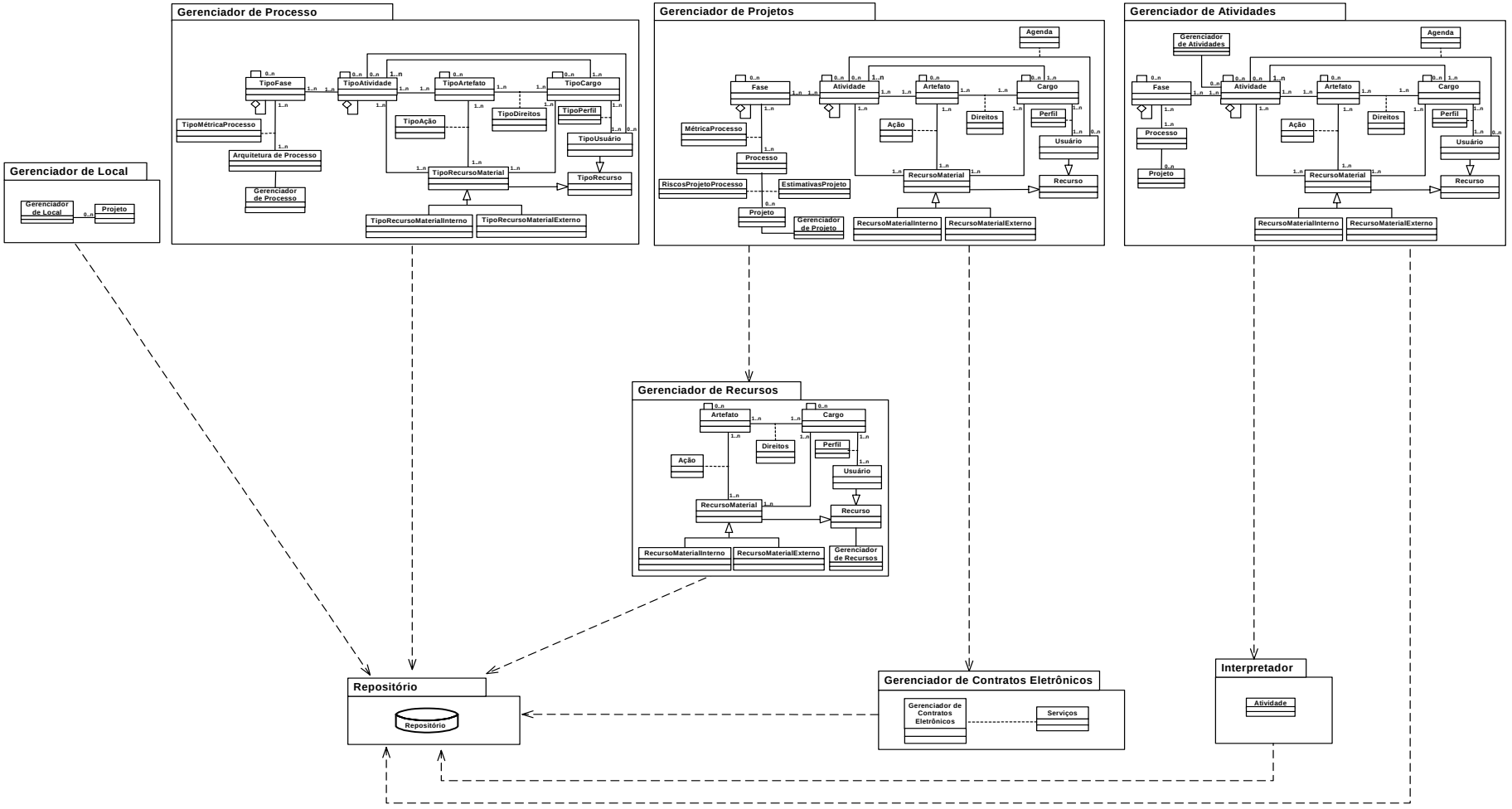

Figura 3.8: Diagrama de Camadas Verticais.

3.4 PROJETO INTERNO DOS COMPONENTES

Neste estágio os componentes individuais da aplicação são refinados até o nível de interfaces, classes ou componentes pré-existentes, satisfazendo os requisitos funcionais e não funcionais definidos na especificação do sistema, seguindo a arquitetura definida. Na Figura 3.9 é apresentada a arquitetura proposta para o ambiente DiSEN. A arquitetura representa o modelo de gerenciamento de processo de *software* proposto, neste trabalho, para o ambiente DiSEN.

A construção do Diagrama de Camadas Verticais apresentado na Figura 3.8 finaliza o processo de particionamento em componentes, apresentando os pacotes juntamente com a especificação dos tipos de cada pacote e o relacionamento entre eles. Os componentes podem então ser vistos como pacotes genéricos, constituídos por tipos e relacionamentos. A partir deste diagrama foi possível elaborar o modelo de gerenciamento de processo de *software* proposto para o ambiente DiSEN.

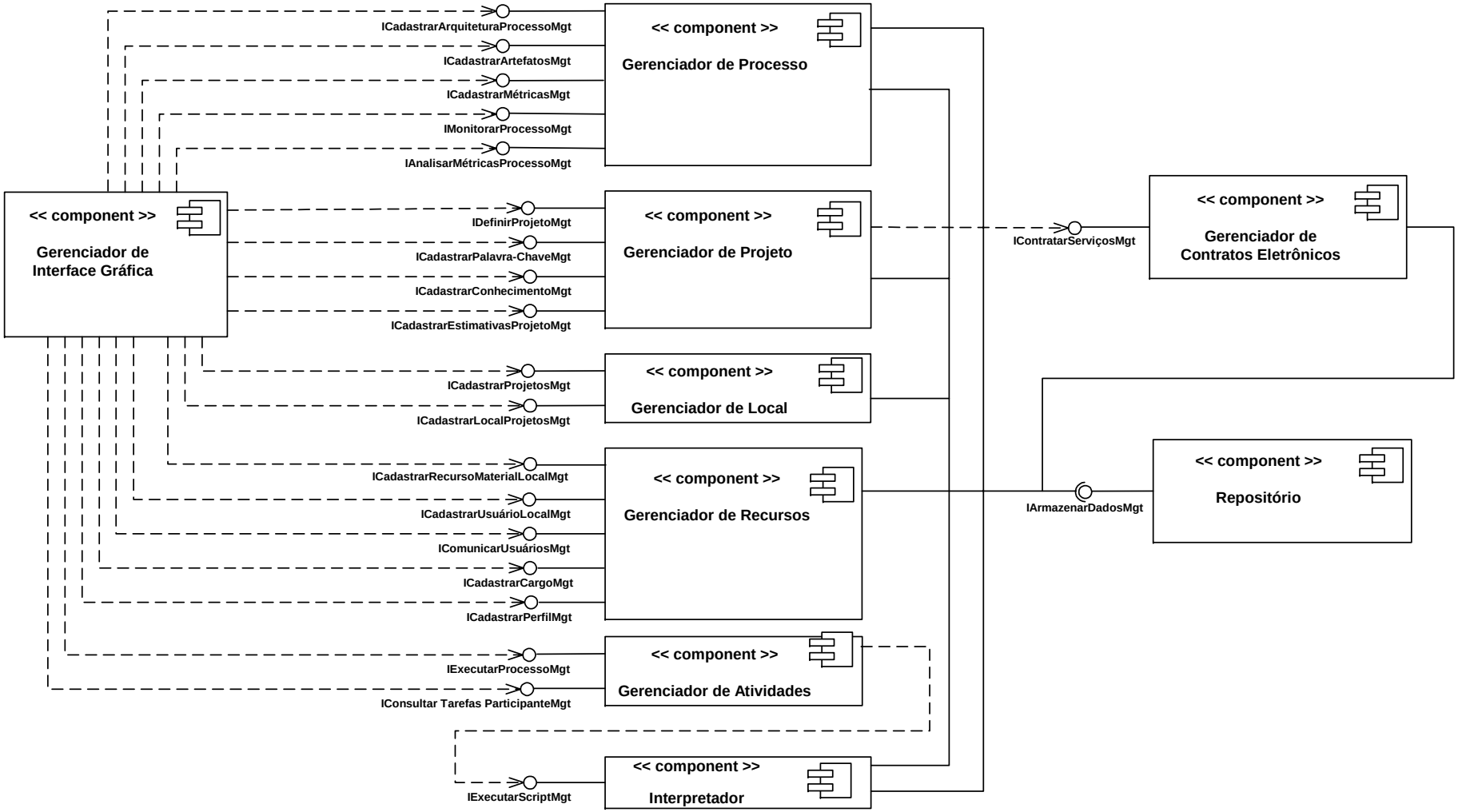

Figura 3.9: Arquitetura de Componentes proposta para o ambiente DiSEN.

O modelo de gerenciamento de processo de *software* proposto é composto pelos seguintes componentes:

- **Gerenciador de Processo**: responsável pelo gerenciamento dos processos de *software*, incluindo sua definição e execução, e pela coordenação dos outros componentes. Além da modelagem e execução dos processos de *software*, o componente gerenciador de processo é responsável pelo gerenciamento dos artefatos gerados pelos projetos durante a execução do processo, pelas métricas associadas ao processo, e pelo monitoramento do processo analisando possíveis problemas que venham a ocorrer durante a execução do processo. O Gerenciador de Processo oferece as seguintes interfaces:

ICadastrarArquiteturaProcessoMgt	Permite inserir/atualizar/excluir as arquiteturas de processo criadas.
ICadastrarArtefatosMgt	Permite inserir/atualizar/excluir os artefatos gerados pelos projetos.
ICadastrarMétricasMgt	Permite inserir/atualizar/excluir as métricas associadas a um processo no ambiente
IMonitorarProcessoMgt	Permite monitorar todo o processo analisando possíveis problemas que venham a ocorrer durante a sua execução.
IAnalisarMétricasProcessoMgt	Permite analisar as métricas dos processos executados no ambiente.

- **Gerenciador de Projeto**: responsável pelo gerenciamento de projetos. Este componente permite a instanciação de uma arquitetura de processo que implica em definir qual dentre as arquiteturas de processos da organização melhor se adequa ao desenvolvimento de um determinado projeto. Além disso, este componente define juntamente com o gerente geral quais métricas, além das que já existem, para o processo devem ser realizadas; permite mudanças dinâmicas no processo realizadas em tempo de execução; e, permite programação das atividades do processo instanciado, associando os recursos a serem utilizados no desenvolvimento do projeto,

definindo os participantes do projeto que devem desenvolver cada atividade do projeto, definindo os recursos necessários para desenvolver cada projeto, agendando as atividades na agenda de cada participante do projeto, definindo os conhecimentos adquiridos com o desenvolvimento de projetos, associados a palavras-chave, e contratando serviços realizados por outras organizações. O Gerenciador de Projeto oferece as seguintes interfaces:

IDefinirProjetoMgt	Permite determinar um projeto a ser desenvolvido em uma determinada unidade (local).
ICadastrarPalavra-ChaveMgt	Permite inserir/atualizar/excluir as palavras-chaves a serem associadas aos conhecimentos.
ICadastrarConhecimentoMgt	Permite inserir/atualizar/excluir os conhecimentos adquiridos com o desenvolvimento de projetos, cadastrando os conhecimentos.
ICadastrarEstimativasProjetoMgt	Permite inserir/atualizar/excluir as estimativas do projeto.

- **Gerenciador de Local**: responsável pelo gerenciamento das unidades de trabalho (local). Este componente permite que projetos sejam gerenciados em determinados locais, e que locais, onde projetos são desenvolvidos, sejam gerenciados. O Gerenciador de Local oferece as seguintes interfaces:

ICadastrarProjetosMgt	Permite inserir/atualizar/excluir os projetos a serem desenvolvidos no local.
ICadastrarLocalProjetosMgt	Permite inserir/atualizar/excluir os projetos destinados a um determinado local.

- **Gerenciador de Recursos**: responsável pelo gerenciamento de recursos, tanto recursos humanos, quanto recursos materiais internos ao ambiente necessários para a execução das atividades dos projetos. Além disso, este componente permite o gerenciamento de perfil, e a comunicação entre usuários do ambiente. O Gerenciador de Recursos oferece as seguintes interfaces:

ICadastrarRecursoMaterialLocalMgt	Permite inserir/atualizar/excluir os recursos materiais e uma determinada unidade (local).

ICadastrarUsuárioLocalMgt	Permite inserir/atualizar/excluir cada usuário de uma determinada unidade (local).
ICadastrarCargoMgt	Permite inserir/atualizar/excluir os cargos existentes no ambiente a serem ocupados pelos usuários, tais como analistas, engenheiros.
IComunicarUsuáriosMgt	Permite a comunicação dos participantes por meio de via tele-conferência, via correio eletrônico e via *chat*.
ICadastrarPerfilMgt	Permite gerenciar o perfil de cada usuário do ambiente, que engloba a habilidade, o conhecimento e o treinamento que cada um possui.

- **Gerenciador de Contratos Eletrônicos**: responsável pelo gerenciamento dos serviços *Web* contratados de outras organizações envolvidas em atividades de comunicação interorganizacionais. A comunicação é realizada com outros serviços de gerenciamento de contratos eletrônicos de outras organizações. O Gerenciador de Contratos Eletrônicos oferece a seguinte interface:

IContratarServiçosMgt	Permite o gerenciamento dos serviços *Web* contratados de outras organizações.

- **Gerenciador de Atividades**: responsável pelo controle e gerenciamento das atividades e ações a serem realizadas no processo de *software*, além de apoiar a interação com os usuários do ambiente (agenda), auxiliando-os na identificação de suas atividades no processo. O Gerenciador de Atividades oferece a seguinte interface:

IExecutarProcessoMgt	Permite a execução das atividades específicas para cada processo definido.
IConsultarTarefasParticipanteMgt	Permite consultar a agenda de atividades dos participantes do projeto.

- **Repositório**: responsável pelo relacionamento com os mecanismos de armazenamento de dados manipulados pelo ambiente. Suas funções trabalham com objetos, que podem ser considerados desde dados de controle do processo, dados relevantes do processo ou até mesmo instâncias de processo de *software* e atividades. Todos os componentes pertencentes ao modelo proposto utilizam seus serviços. Sua presença torna o restante

dos componentes independentes de uma implementação particular de um sistema gerenciador de objetos, garantindo flexibilidade e portabilidade para toda a coleção de componentes existentes. O Repositório oferece a seguinte interface:

IArmazenarDadosMgt	Permite o gerenciamento de dados manipulados pelo ambiente.

- **Interpretador**: Uma linguagem para modelagem de processos é necessária para que o processo possa ser programado e executado pelo gerenciador de processo. As atividades que compõem o processo são então programadas utilizando uma linguagem de modelagem de processos e recursos que permitem a execução cooperativa dessas atividades. Para a execução dessas atividades, o gerenciador de atividades realiza chamadas ao interpretador da linguagem para que este possa executá-las.

3.4.1 DIAGRAMAS DE SEQÜÊNCIA

Com base no modelo de gerenciamento de processo de *software* proposto, foi possível elaborar diagramas de seqüência que representam a comunicação entre os componentes do modelo, conforme apresentados nas Figura 3.10, Figura 3.11, Figura 3.12 e Figura 3.13. Um diagrama de seqüência mostra a colaboração dinâmica entre os vários objetos de um sistema. O mais importante aspecto deste modelo é que a partir dele pode-se perceber a seqüência de mensagens enviadas entre os objetos. Neste caso, a mesma idéia é aplicada aos componentes.

Na Figura 3.10 é apresentado o diagrama de seqüência do ator Gerente Geral. De acordo com o diagrama, o Gerente Geral tem permissão para cadastrar as arquiteturas de processo do ambiente (*CadastrarArquiteturaProcesso*); cadastrar os tipos e a quantidade de recursos necessários para cada unidade (local) (*CadastrarRecursoMaterialLocal*); cadastrar os cargos que os usuários podem obter ou exercer dentro do ambiente (*CadastrarCargo*); cadastrar o perfil de cada usuário do ambiente, que engloba a habilidade, o conhecimento e o

treinamento que cada um possui (*CadastrarPerfil*); cadastrar os artefatos gerados pelas atividades do processo (*CadastrarArtefatos*); cadastrar as métricas do processo (*CadastrarMétricas*); monitorar os processos (*MonitorarProcesso*); analisar as métricas de um determinado processo (*AnalisarMétricasProcesso*); e, cadastrar os projetos a serem realizados em determinada unidade (local) (*CadastrarLocalProjeto*). Estas operações são realizadas com o apoio do Gerenciador de Processo, Gerenciador de Local, Gerenciador de Recursos e o Repositório.

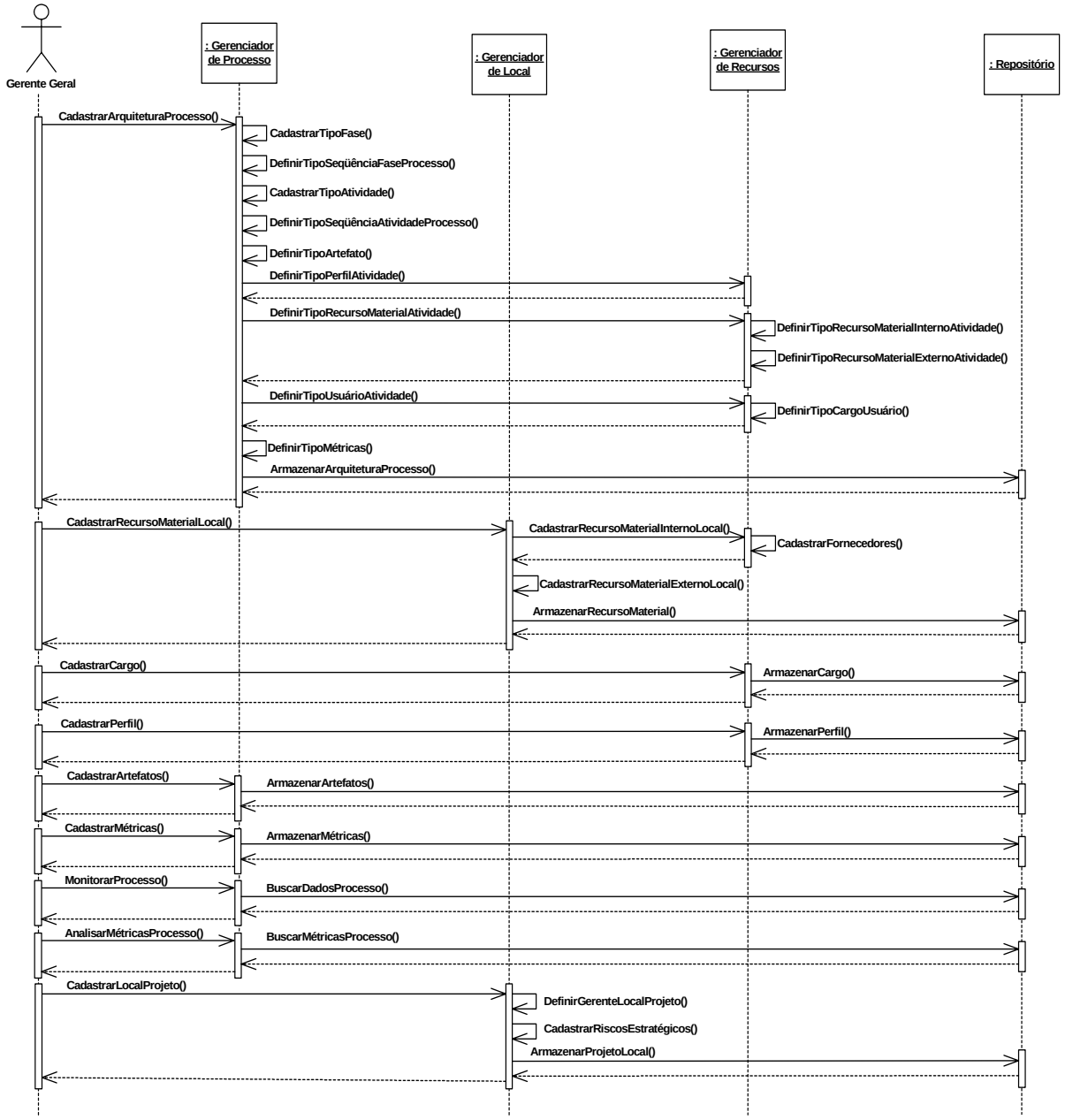

Figura 3.10: Diagrama de Seqüência – Gerente Geral.

Na Figura 3.11 é apresentado o diagrama de seqüência do ator Gerente Local. De acordo com o diagrama, o Gerente Local tem permissão para cadastrar os projetos que devem ser realizados nos locais determinados (*CadastrarProjetos*) e cadastrar usuários de determinados locais (*CadastrarUsuárioLocal*), de acordo com o perfil e o seu nível de afinidade com outros usuários que fazem parte do mesmo local. O Gerenciador de Projeto com o apoio do Gerenciador de Recursos e o Repositório, permite que o Gerente Local defina o Gerente de Projeto responsável pelo gerenciamento do projeto e os recursos necessários para a realização das atividades do projeto.

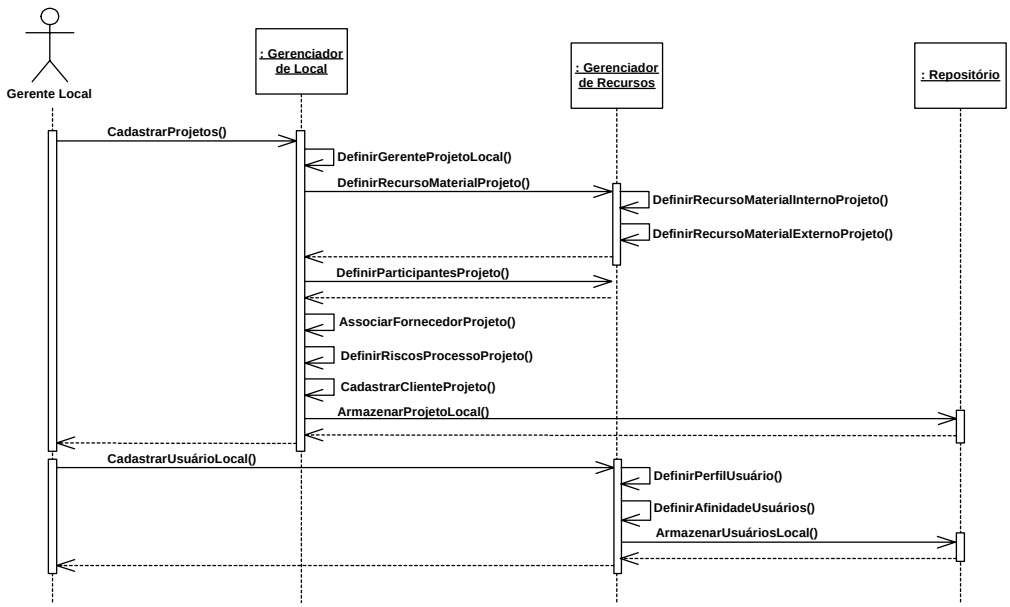

Figura 3.11: Diagrama de Seqüência – Gerente Local.

Na Figura 3.12 é apresentado o diagrama de seqüência do ator Gerente de Projeto. De acordo com o diagrama, o Gerente de Projeto tem permissão para definir o projeto a ser gerenciado por ele (*DefinirProjeto*); instanciar uma arquitetura de processo (*DefinirArquiteturaProcesso*); programar as atividades das fases do processo (*CadastrarFaseProjeto*); possibilitando que atividades sejam contratadas por meio do Gerenciador de Contratos Eletrônicos (*DefinirContrataçãoServiços*); permitir que mudanças dinâmicas sejam realizadas no processo definido (*PermitirMudançasDinâmicasProcesso*);

definir os valores das métricas do processo definido (*DefinirValorMétrica*); cadastrar as estimativas do projeto (*CadastrarEstimativasProjeto*); cadastrar os conhecimentos adquiridos com o desenvolvimento de projetos (*CadastrarConhecimento*) e palavras-chave (*CadastrarPalavra-Chave*). Estas operações são realizadas com o apoio do Gerenciador de Projeto, Gerenciador de Recursos, Gerenciador de Atividades, Gerenciador de Contratos Eletrônicos e o Repositório.

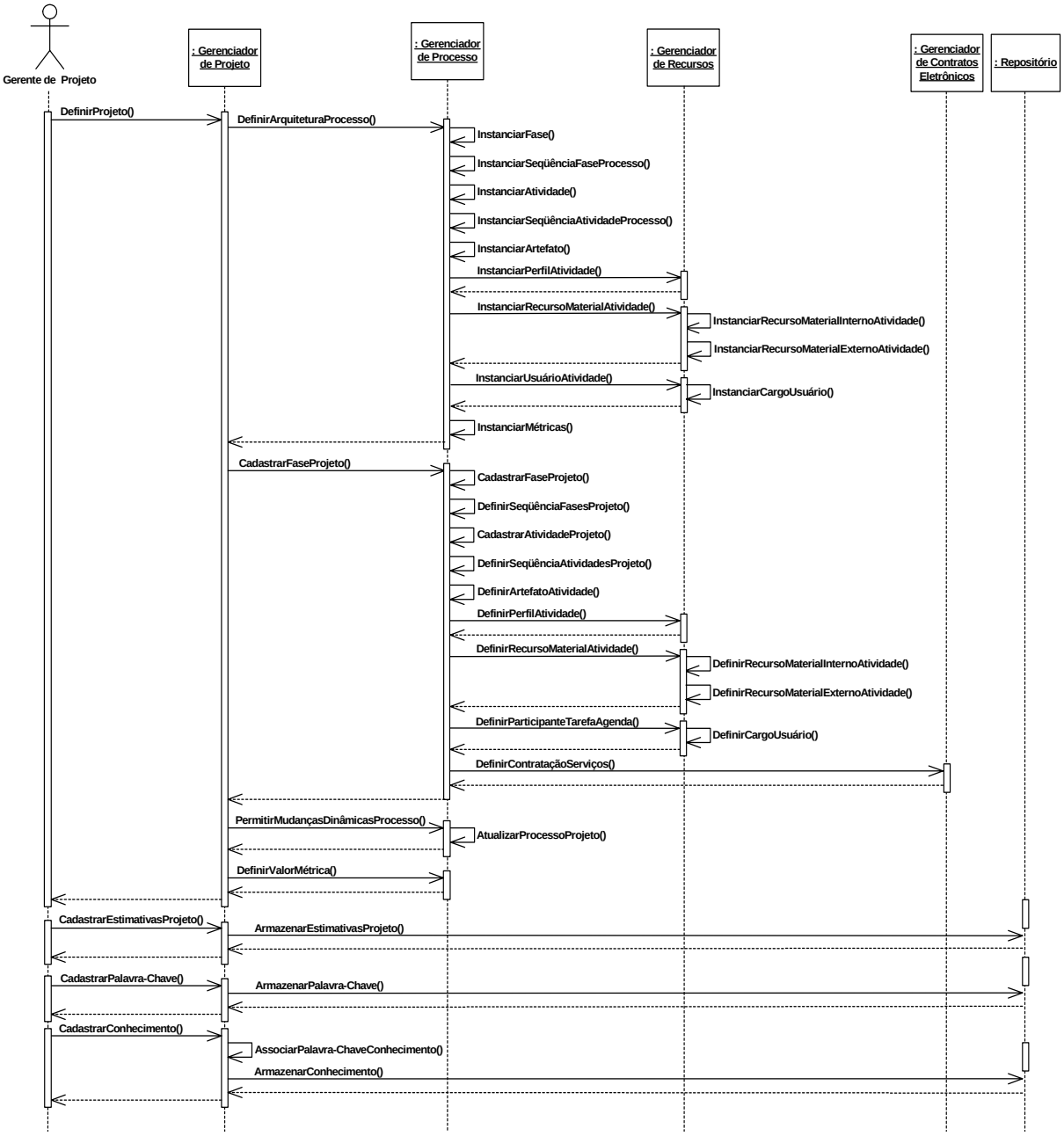

Figura 3.12: Diagrama de Seqüência – Gerente de Projeto.

Na Figura 3.13 é apresentado o diagrama de seqüência do ator Engenheiro de *Software*. De acordo com o diagrama, o Engenheiro de *Software* tem permissão para executar as atividades do processo (*ExecutarProcesso*). O Gerenciador de Atividades com o apoio do Interpretador e do Repositório, permite que o Engenheiro de *Software* tenha acesso a sua agenda de atividades (*ConsultarTarefasParticipante*), e execute cada atividade de acordo com as pré-condições estabelecidas, com o controle de prioridade, e a atualização da situação de cada tarefa (executar, interromper, reiniciar, finalizar e cancelar), além de poder comunicar-se com outros usuários do ambiente (*ComunicarUsuários*), via *chat*, tele-conferência e correio eletrônico.

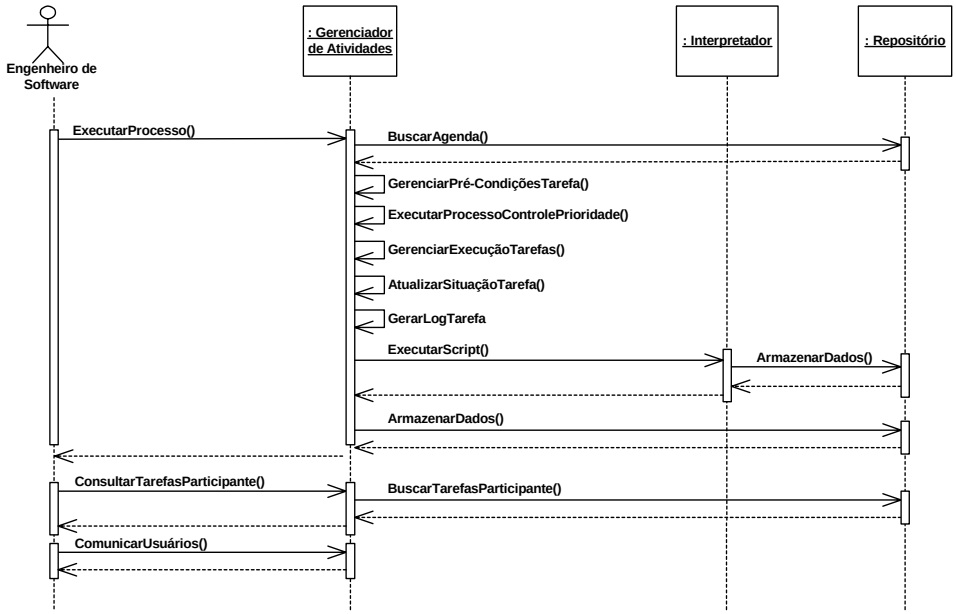

Figura 3.13: Diagrama de Seqüência – Engenheiro de *Software*.

3.5 CONSIDERAÇÕES FINAIS

Este Capítulo apresentou o modelo de gerenciamento de processo de *software* proposto para o ambiente DiSEN. O referido modelo foi desenvolvido com base na *(i)* arquitetura do ambiente DiSEN proposta por Pascutti; *(ii)* na especificação do ambiente DiSEN; *(iii)* na arquitetura genérica do Sistema de *Workflow* proposto pela WfMC (*Workflow*

Management Coaliton); e, *(iv)* no modelo de Sistema de Gerenciamento de Processos de Negócios baseada em Serviços *Web*, proposto por Zhao, Liu & Yang.

O modelo de gerenciamento de processo de *software* proposto permite:

- a definição de arquiteturas de processos de *software (Cadastrar Arquitetura de Processo)*;
- a instanciação das arquiteturas de processos de *software (Instanciar Arquitetura de Processo)*;
- a execução do processo de *software* instanciado e programado *(Executar Processo)*, possibilitando a execução de atividades do processo como serviços *Web*, com o apoio do Componente Gerenciador de Contratos Eletrônicos e da classe Serviços;
- o estabelecimento de contratos eletrônicos para permitir que as organizações participantes de um processo cheguem a acordos sobre os detalhes do processo de *software* a serem realizados de forma cooperativa *(Definir Contratação Serviços)*;
- o gerenciamento de recursos para alocar tanto recursos humanos quanto recursos materiais para a execução do processo *(Definir Recurso Material Atividade e Definir Participante Tarefa Agenda)*;
- o monitoramento e auditoria do processo de *software* em execução *(Monitorar Processo)*;
- a verificação e otimização de desempenho do processo de *software (Analisar Métricas do Processo)*.

Além disso, o modelo proposto possui como características dinâmicas a flexibilidade e a adaptabilidade. O modelo permite que sejam criadas e desfeitas cooperações entre organizações para a execução de um determinado processo de *software*, em tempo de

execução *(Definir Contratação Serviços)*; além de permitir que modelos de processos de *software* sejam modificados para se adaptar a mudanças no ambiente *(Permitir Mudanças Dinâmica Processo).*

De acordo com a especificação do ambiente realizada neste trabalho, chegou-se a conclusão de que o gerenciador de objetos da camada de aplicação pudesse ser alterado para gerenciador de processo, conforme mostra a Figura 3.14. Desse modo, o gerenciador de processo seria composto pelo Gerenciador de Local, Gerenciador de Projetos, Gerenciador de Recursos, Gerenciador de Atividades e Gerenciador de Contratos Eletrônicos, conforme mostra a Figura 3.15.

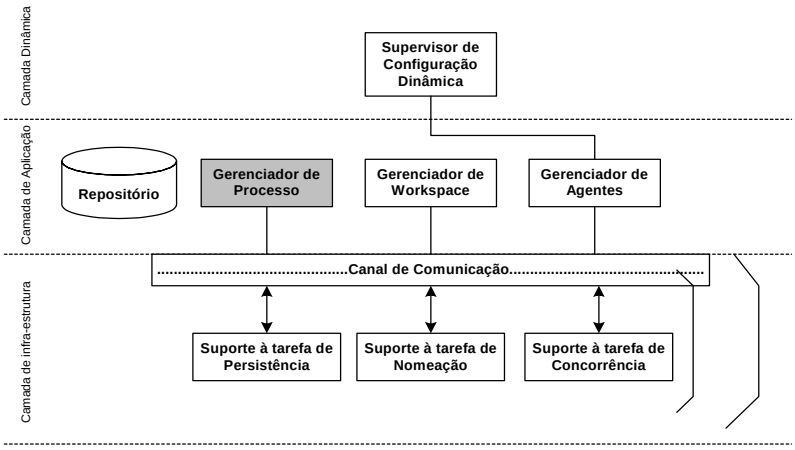

Figura 3.14: Nova arquitetura proposta para o ambiente DiSEN.

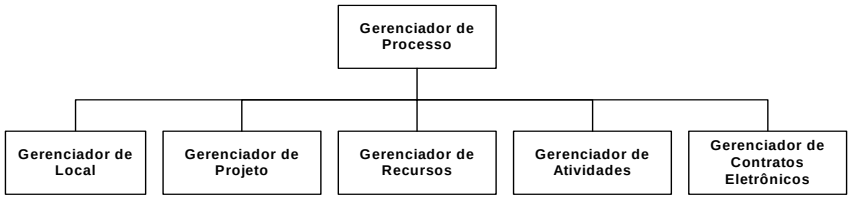

Figura 3.15: Entidades agregadas ao Gerenciador de Processo.

Outra sugestão de alteração da especificação do ambiente DiSEN, diz respeito a definição de um processo de *software* no ambiente. Ao invés de considerar fases do processo e suas atividades, como por exemplo, Fase *Análise* composta pela atividade *Elaborar Modelo*

de Caso de Uso de Domínio, seria interessante considerarmos apenas as atividades do processo, como por exemplo, *Elaborar Modelo de Caso de Uso de Domínio*, *Elaborar Diagrama de Seqüência*. As fases existem somente como separados de atividades. Desse modo, ass atividades do processo, por sua vez, são definidas como serviços *Web*, e documentadas em arquivos WSDL.

Além, de um modelo para gerenciamento de processo, é necessário uma linguagem de especificação de processo, o próximo capitulo ilustra como BPEL poderia ser utilizada para tal.

CAPÍTULO 4

ESPECIFICAÇÃO DE PROCESSO DE *SOFTWARE* NO AMBIENTE DISEN UTILIZANDO BPEL

Como visto na Seção 2.2.1, os modelos de processo de *software* podem ser representados por meio de técnicas como diagramas SADT, diagramas de módulos, diagramas de transição de estados, diagramas de fluxo de dados e regras de produção. Porém, a principal forma de representar processos de *software* é por meio de uma linguagem de programação de processo, ou linguagem de modelagem de processo (OSTERWEIL, 1987).

Neste sentido, este Capítulo tem como objetivo apresentar uma especificação de um processo de *software* no ambiente DiSEN utilizando BPEL.

4.1 PROCESSO DE *SOFTWARE* NO AMBIENTE DISEN

Conforme o Modelo Estático de Tipos apresentado na Figura 3.5, as classes localizadas na parte central do diagrama envolvem a definição de arquiteturas de processo e os tipos de objetos relacionados a esta arquitetura, enquanto as classes localizadas na parte inferior do diagrama envolvem a instanciação de arquiteturas de processo já definidas, bem como os seus tipos de objetos. As classes localizadas na parte superior do diagrama representam os tipos referentes aos módulos de gerenciamento.

Com base neste modelo, um processo de *software* no ambiente DiSEN é composto por uma ou mais fases de processo de *software*, uma fase pode possuir um ou mais tipos de atividades, artefatos e recursos. As classes presentes na parte inferior do diagrama representam instâncias dos tipos definidos na arquitetura de processo. Dessa forma, os

relacionamentos entre essas classes comportam-se da mesma maneira descrita para a arquitetura. Assim, tem-se um processo de *software* definido, com suas fases, atividades ou serviços, artefatos, recursos, ações e cargos.

Um processo de *software* no ambiente DiSEN pode ser definido, como mostrado na Figura 4.1, composto por:

- Fase: define as etapas do processo de *software*, como por exemplo, Requisitos, Análise, Implementação, Teste;
- Atividade: cada fase é composta por um número de atividades, como por exemplo, Requisitos é composto pela atividade “Elaborar o Modelo de Domínio”;
- Artefato: produto criado, modificado ou utilizado durante um processo, como por exemplo, a atividade “Elaborar o Modelo de Domínio”, produz o artefato Modelo de Domínio;
- Recurso: cada atividade desenvolvida utiliza recursos para produzir o artefato. Estes recursos podem ser recursos humanos, tais como engenheiros, analistas, e recursos materiais internos ou externos ao ambiente, tais como ferramentas, papéis.

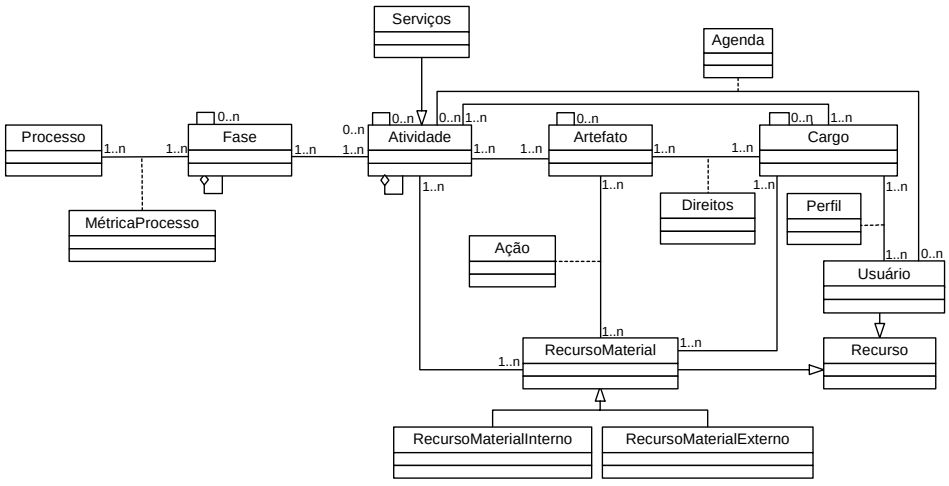

Figura 4.1: Processo de *Software* no Ambiente DiSEN.

O desafio, esse momento, é especificar um processo de *software* definido para o ambiente DiSEN utilizando a linguagem BPEL.

4.2 ESPECIFICAÇÃO DE UM PROCESSO DE SOFTWARE NO AMBIENTE DISEN UTILIZANDO BPEL

A definição de um processo utilizando BPEL, conforme mostra a Figura 4.2, consiste de dois tipos de arquivos (MALDANER e PASQUAL, 2006):

- arquivos WSDL, que especificam os tipos de ligações entre as interfaces, propriedades, tipo de portas e operações, mensagens e parte de interesse do processo, incluindo serviços implementados e invocados pelo processo; e
- um arquivo BPEL, que especifica as definições de processo, incluindo todas as atividades e principais variáveis envolvidas.

Um arquivo WSDL é um documento XML que obedece às regras de uma especificação para descrever serviços *Web*. WSDL possibilita uma separação da descrição das funcionalidades abstratas oferecidas por um serviço, dos detalhes concretos da descrição de um serviço, como "onde" e "como" as funcionalidades são oferecidas (WSDL, 2006). Na Figura 4.3 são apresentados os principais elementos de um documento WSDL. Um exemplo de documento WSDL pode ser visto no Anexo B.

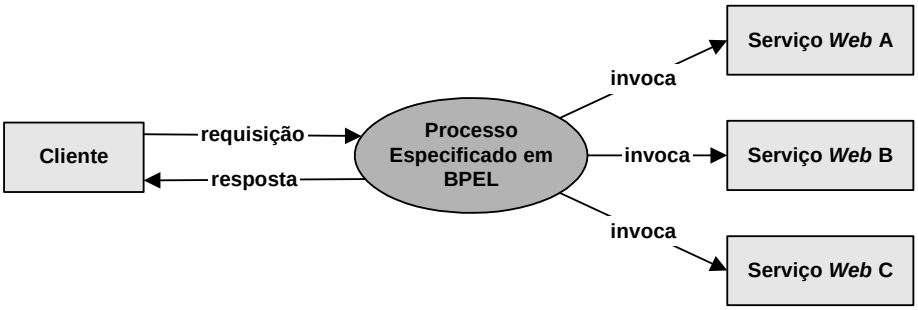

Figura 4.2: Exemplo de um Processo Especificado em BPEL. Fonte: Maldaner e Pasqual (2006).

A descrição abstrata de um serviço *Web* é composta de elementos que descrevem as suas capacidades, conforme apresentado a seguir.

<*wsdl: types*> O elemento <*types*> contém os tipos de dados que estão presentes na mensagem. O elemento *import* funciona como separação entre as várias partes de uma definição WSDL. Ele possui dois atributos, os quais definem a localização do documento importado e o seu *namespace*. Os *namespaces* são espaços para nomes, definidos no interior dos documentos XML. Um documento WSDL é um XML, que utiliza os *namespaces* para maximizar a taxa de reutilização dos componentes de um documento WSDL, utilizando-se de atributos para fazer referência a outros elementos, seja dentro ou fora do documento.

<*wsdl: message*> O elemento <*message*> define os dados a serem transmitidos. Cada elemento *message* recebe um ou mais elementos <*part*>, que formam as partes reais da mensagem. O elemento <*part*> define o conteúdo da mensagem representando os parâmetros que são passados e a resposta que o serviço retorna.

<*wsdl: portType*>
<*wsdl: operation*> Os elementos <*portType*> possuem um conjunto de operações (<*operation*>), as quais possuem um atributo *name* e um atributo opcional para especificar a ordem dos parâmetros usados nas operações.Existem quatro diferentes tipos de mensagens de operações:

- Operação unidirecional: Define uma mensagem enviada de um cliente para um serviço, sem resposta.
- Solicitação-Resposta: O mais utilizado. O cliente envia uma solicitação a um serviço, e recebe como resultado uma mensagem de resposta com o resultado dessa solicitação. Pode ser vista como um RPC (*Remote Procedure Call*).
- Pedido-Resposta: Não utilizada freqüentemente. Define uma mensagem enviada do serviço para o cliente, resultando em uma mensagem enviada do cliente de volta para o serviço.
- Notificação: É quando o serviço envia uma mensagem para o cliente.

A descrição concreta de um serviço *Web é* composta dos elementos que vinculam fisicamente o cliente ao serviço *Web*, conforme descrito a seguir.

<*wsdl: binding***>**	O elemento <*binding*> mapeia os elementos <*operation*> em um elemento <*portType*>, para um protocolo específico. Ele associa o elemento <*portType*> ao protocolo SOAP, utilizando-se de um elemento de extensão SOAP, chamado **<***wsdlsoap : binding***>**, por meio de dois parâmetros: protocolo de transporte e o estilo da requisição (*rpc* ou *document*).
<*wsdl: service***>**	Os elementos <*service*> e <*port*> definem a localização real do serviço, tendo em vista
<*wsdl: port***>**	que o mesmo pode conter várias portas e cada uma delas é específica para um tipo de ligação, descrita no elemento <*binding*>.

O arquivo BPEL também é um documento XML, porém obedece às regras de uma especificação para descrever processos. Conforme mostrado na Seção 2.4.

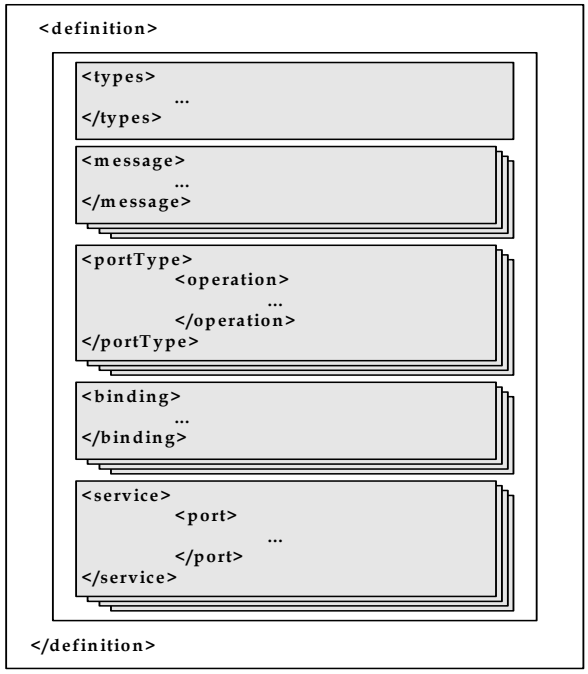

Figura 4.3: Principais elementos de um arquivo WSDL. Fonte: RECKZIEGEL (2006).

Desse modo, a especificação de serviços *Web* e a especificação de um processo em BPEL permite realizar um mapeamento de processo de *software* no ambiente DiSEN utilizando BPEL, conforme mostra a Tabela 4.1. Assim, pode-se definir que uma Fase de um processo de *software* no ambiente DiSEN corresponde ao elemento *Service* de um documento

WSDL, a Atividade de uma Fase corresponde ao elemento *Operation*, um Artefato e um Recurso correspondem ao elemento *Message*.

Tabela 4.1: Processo de *Software* utilizando BPEL

Processo de *Software* no Ambiente DiSEN	**BPEL**
Fase	Serviço *Web* - Service
Atividade	Serviço *Web* - Operation
Artefato	Serviço *Web* - message
Recurso	Serviço *Web* - message

O fluxo das fases de um processo de *software* no ambiente DiSEN e a execução de suas atividades é especificada de acordo com os recursos que a linguagem BPEL oferece para descrever processos. As atividades podem ser invocadas e executadas em fluxo seqüencial ou paralelo, permitindo o tratamento de exceções e transações, dentre outros recursos, conforme mostrado na Figura 4.4.

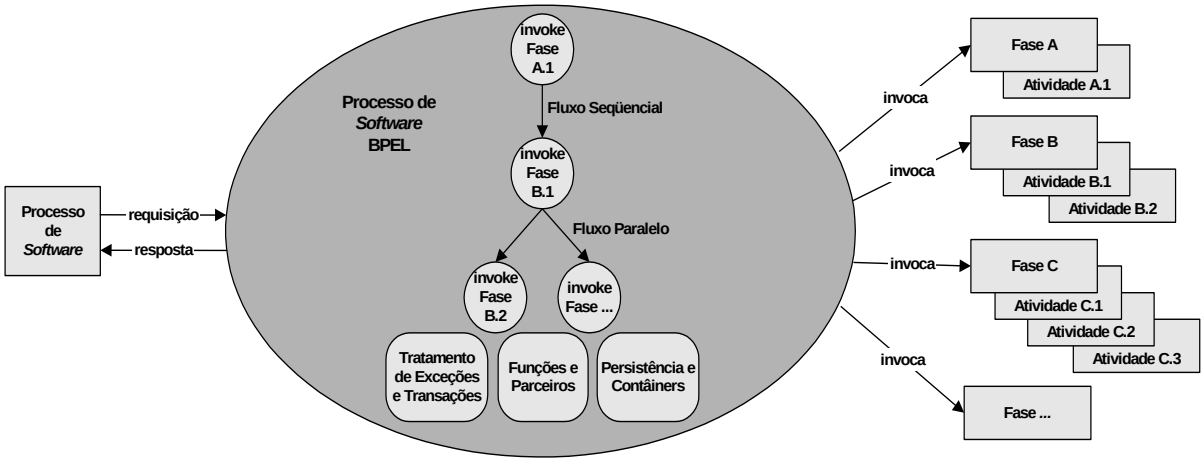

Figura 4.4: Processo de *Software* no ambiente DiSEN utilizando BPEL.

4.2.1 ESTUDO DE CASO

O processo de *software* utilizado como exemplo é apresentado como um diagrama de atividades UML, conforme mostrado na Figura 4.5. O estudo de caso realizado utilizou o mapeamento apresentado na Tabela 4.1.

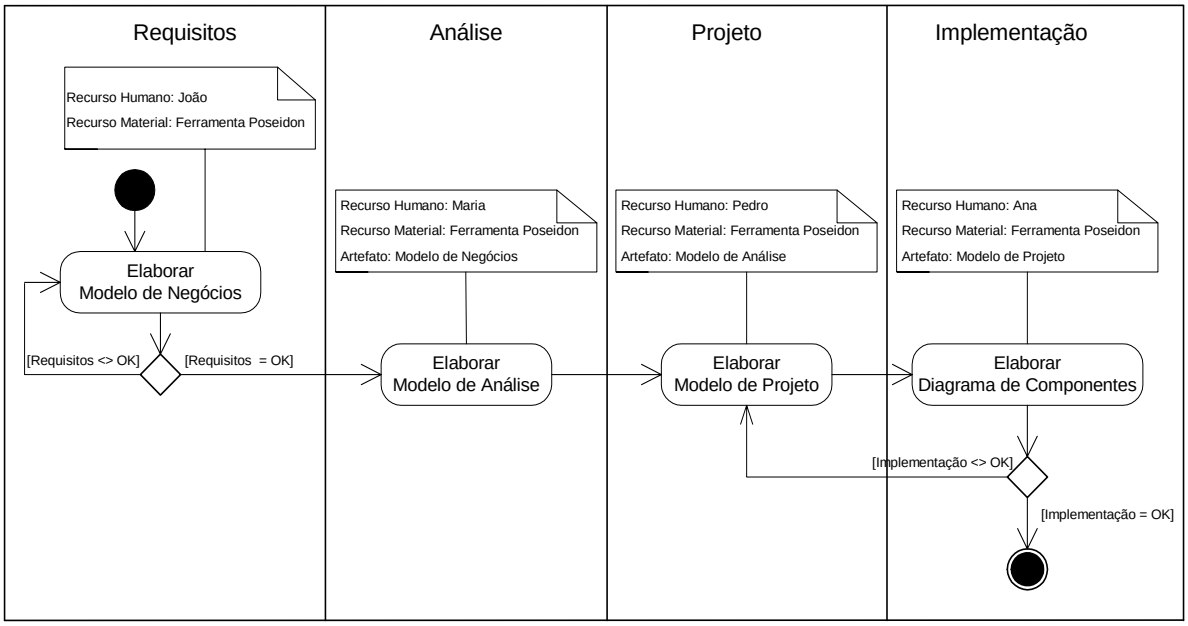

Figura 4.5: Processo de *Software* utilizado no Estudo de Caso.

O diagrama de atividades, como visto na Figura 4.5, pode ser usado de acordo em muitas metodologias e para muitos propósitos, desde modelagem descritiva de alto nível até modelagem detalhada dirigida à execução de processo. Quando um dos propósitos é definir a execução de processo, então o mesmo deve ser desenvolvido com o apoio de uma ferramenta de modelagem projetada para este propósito. O diagrama de atividades representa a estrutura básica e o fluxo de atividades de um processo de *software*.

O processo é composto por quatro fases: *Requisitos*, *Análise*, *Projeto* e *Implementação*. Cada fase possui atividades, como *Elaborar Modelo de Negócios*, *Elaborar Modelo de Análise*, *Elaborar Modelo de Projeto* e *Elaborar Diagrama de Componentes*, respectivamente. Cada atividade utiliza-se de recursos humanos e materiais para que possam

ser realizados, e produz como resultado, artefatos. As atividades *Elaborar Modelo de Negócios* e *Elaborar Diagrama de Componentes* são repetidas até que os resultados desejados sejam produzidos. O processo tem início com a fase *Requisitos* e a atividade *Elaborar Modelo de Negócios*, e é finalizado com a fase *Implementação* e a atividade *Elaborar Diagrama de Componentes*.

Para a realização deste estudo de caso, a ferramenta NetBeans 5.5 (NETBEANS, 2006) foi utilizada para apoiar a especificação de um processo de *software* utilizando BPEL, conforme mostra a Figura 4.6. O lado esquerdo da figura apresenta o processo de *software* exemplo, apresentado na Figura 4.5, modelado com a ferramenta NetBeans. O lado direito acima apresenta parte do código do processo em BPEL referente a fase *Requisitos*, e o lado direito abaixo apresenta o código da fase *Requisitos* como serviço *Web* (arquivo WSDL).

O processo de *software*, especificado em BPEL, tem início com o recebimento de uma mensagem (*IniciarProcesso*) solicitando a inicialização do processo. Logo após receber a mensagem, o processo associa os dados contidos na mensagem recebida com as variáveis utilizadas no processo (*CopiarProcessoSoftware*).

Neste momento, o processo está pronto para iniciar a execução das fases e atividades definidas para o processo. De acordo com o processo definido, a primeira fase a ser desenvolvida é a fase de *Requisitos*. O serviço *Requisitos (<service name="RequisitosSoapService">)* é então invocado por meio de uma mensagem *(invoke name="ElaborarModeloNegocios")*, de modo que, a atividade *Elaborar Modelo de Negócios (operation="elaborarModeloNegocios")* possa ser realizada. A mensagem *(<input message="tns:ElaborarModeloNegocios"/>)* enviada ao serviço *Requisitos* é composta pelos recursos necessários para a execução da atividade, conforme mostra a Figura 4.7. A atividade *Elaborar Modelo de Negócios* é executada e, retorna o resultado *(<input message =*

"tns:ElaborarModeloNegocios"/>) ao processo. Após executar a atividade, os dados são atualizados dentro do processo (*CopiarRequisitosOK*).

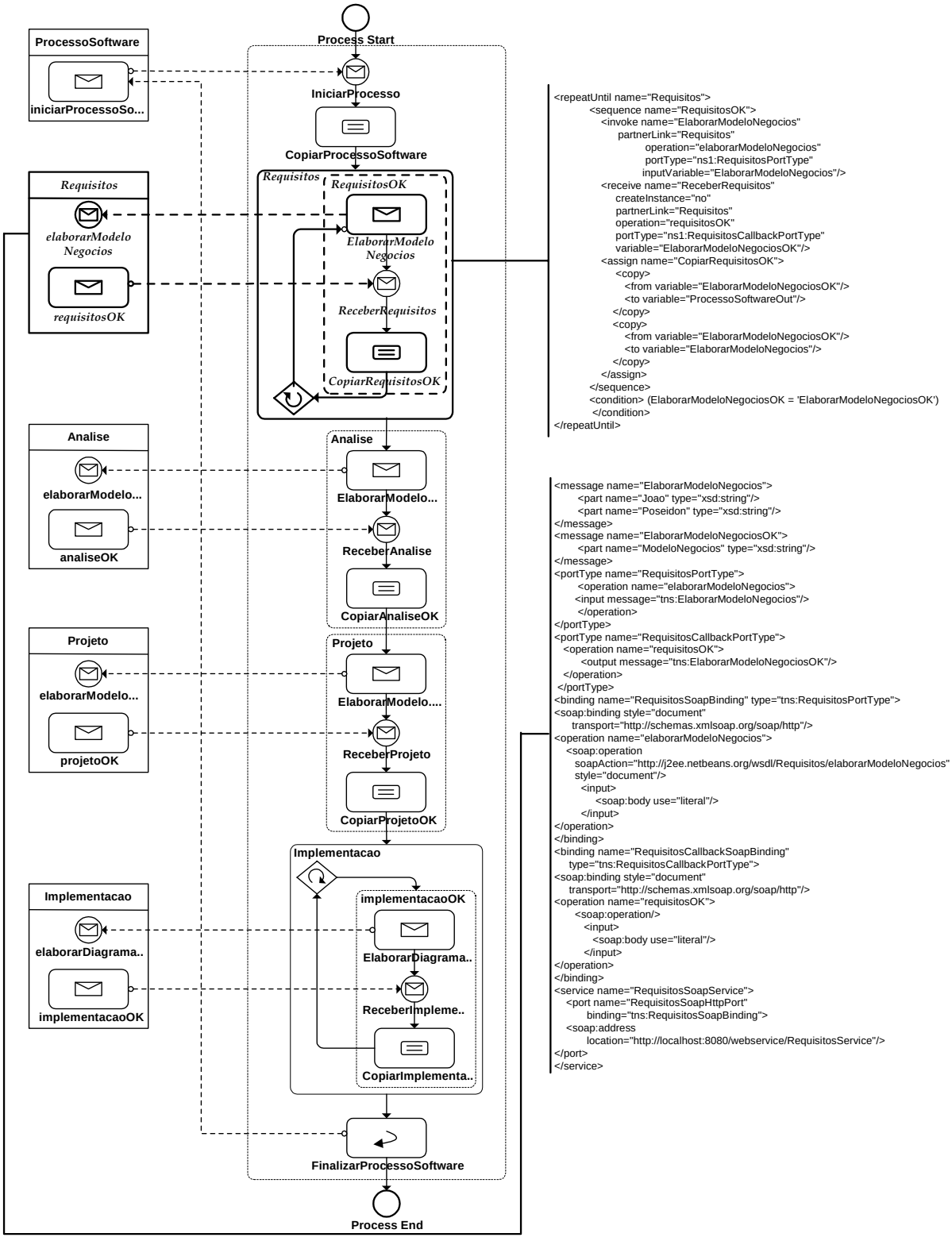

Figura 4.6: Processo de *Software* utilizando BPEL.

```
<message name="ElaborarModeloNegocios">
        <part name="Joao" type="xsd:string"/>
        <part name="Poseidon" type="xsd:string"/>
</message>
```

Figura 4.7: Mensagem (*<input message="tns:ElaborarModeloNegocios"/>*)

Após a fase de *Requisitos* ser concluída com sucesso, a fase *Análise (<service name=" AnaliseSoapService">)* é então invocada também por meio de uma mensagem *(invoke name="ElaborarModeloAnalise ")*, de modo que, a atividade *Elaborar Modelo de Análise (operation="elaborarModeloAnalise")* possa ser realizada. A mensagem *(<input message="tns:ElaborarModeloAnalise"/>)* enviada ao serviço *Análise* é composta pelos recursos necessários para a execução da atividade, além do artefato produzido na fase de *Requisitos*. A atividade *Elaborar Modelo de Análise* é executada e, retorna o resultado *(<input message="tns:ElaborarModeloAnalise"/>)* ao processo. Após executar a atividade, os dados são atualizados dentro do processo (*Copiar Analise OK*).

Após a fase de *Análise* ser concluída, inicia-se a fase de *Projeto*, e por fim, a fase de *Implementação*. O processo é finalizado quando todas as fases e suas atividades forem realizadas com sucesso.

4.3 CONSIDERAÇÕES FINAIS

Este Capítulo apresentou um estudo de caso ilustrando como um processo de *software* pode ser especificado utilizando BPEL.

De acordo, com os recursos oferecidos por BPEL, foi possível realizar um mapeamento dos elementos de processos de *software* para os elementos BPEL, possibilitando que atividades dos processos sejam executadas como serviços *Web*.

No entanto, a definição de processos de *software* incorpora tipicamente pessoas como um tipo adicional de participante, de modo que eles participam e influenciam na execução dos

processos, conforme mostrada na Figura 4.7. Logo, os processos vão além da orquestração das atividades expostas como serviços *Web*. Desse modo, foi identificada a dificuldade em realizar o mapeamento de processos de *software* em BPEL envolvendo a interação humana nas atividades dos processos, uma vez que BPEL que é uma linguagem apropriada para a especificação de atividades automatizadas que não necessitam de interação com pessoas. Para apoiar cenários em que pessoas são envolvidas, uma extensão de BPEL é requerida.

CAPÍTULO 5

CONCLUSÃO

Atualmente, é evidente a necessidade existente nas organizações de aumentarem sua produtividade. A engenharia de *software* vem buscando desenvolver métodos, técnicas, ferramentas e notações que sirvam de suporte a este aumento de produtividade, o que tem permitido que grupos de diferentes localidades e culturas, com diferentes expectativas possam formar uma equipe para trabalhar em projetos distribuídos.

O desenvolvimento distribuído de *software* é uma realidade. Muitas empresas estão distribuindo seu processo de desenvolvimento de *software* ao redor do mundo com o objetivo de reduzir custos e melhorar a qualidade no processo de desenvolvimento de *software*. Dada a relevância cada vez maior da internet na vida cotidiana e a tendência de uma sociedade cada vez mais voltada para o mundo digital, as empresas estão se reestruturando de maneira a adequar-se a esta nova realidade.

Porém, essas mudanças estão causando um grande impacto não apenas no mercado de *software* propriamente dito, mas na maneira como os produtos de *software* estão sendo criados, modelados, construídos, testados e entregues aos seus clientes. Os engenheiros de *software* têm reconhecido a grande vantagem desta nova forma de trabalho no seu dia-a-dia e estão em busca de modelos que facilitem o desenvolvimento de *software* com equipes geograficamente dispersas.

Este trabalho propõe um modelo de gerenciamento de processo de *software* para o ambiente DiSEN. O modelo proposto é composto por componentes que interagem para permitir a definição e execução de um processo de *software*. O modelo foi desenvolvido e

especificado com base na abordagem de desenvolvimento baseado em componentes e tem como notação a linguagem UML (JACOBSON e BOOCH e RUMBAUGH, 1999). O desenvolvimento do modelo seguiu o método Catalysis (D'SOUZA, 1999). A especificação do modelo de gerenciamento de processo de *software* para o ambiente DiSEN foi necessária para que os componentes do modelo fossem definidos e descritos.

De acordo com o modelo de gerenciamento de processo de *software* proposto neste trabalho, sugere-se que a arquitetura proposta por Pascutti (2002), apresentada na Figura 2.1, seja revisada para contemplar os componentes do modelo proposto para o gerenciamento de processo em ambiente de desenvolvimento de *software*.

Além disso, um estudo de caso foi realizado ilustrando como um processo de *software* no ambiente DiSEN pode ser especificado utilizando BPEL. Com a especificação BPEL foi possível realizar um mapeamento dos elementos de um processo de *software* para os elementos BPEL. Desse modo, os recursos que BPEL oferece para descrever processos de negócios podem ser utilizados para especificar processos de *software*, possibilitando a execução de atividades do processo como serviços *Web*.

BPEL é uma linguagem apropriada para a especificação de atividades automatizadas que não necessitam de interação com pessoas. Entretanto, como os processos de *software* são atividades que inerentemente necessitam de interação humana, foi possível analisar a dificuldade encontrada no mapeamento de processos de *software* em BPEL. Um dos impedimentos encontrados para a adoção de BPEL no DiSEN é o fato de atualmente o interpretador de processos do ambiente DiSEN trabalhar com um formato proprietário de especificação de processos. A utilização de BPEL exigiria uma mudança em toda a estrutura utilizada pela execução dos processos. Por outro lado, o fato do ambiente utilizar BPEL também traria o benefício de que várias funcionalidades que hoje devem ser implementadas no ambiente já estarem disponibilizadas em máquinas BPEL disponíveis no mercado. Uma

sugestão para trabalhos futuros é a possibilidade da utilização de uma especificação recente denominada de BPEL4People, ou WS-BPEL *Extension for People*. Esta especificação possui terminologias e instrumentos adequados para cenários onde a interação de pessoas nos processos de negócios sejam fundamental.

Como contribuições deste trabalho, têm-se a proposta do modelo de gerenciamento de processo de *software* para o ambiente DiSEN, que tem como objetivo oferecer apoio à especificação, execução e gerenciamento de processos de *software*; a colaboração interorganizacional entre organizações para a realização das atividades de um processo por meio do componente Gerenciador de Contratos Eletrônicos; e a especificação de um processo de *software* no ambiente DiSEN utilizando uma linguagem de modelagem de processos de negócios.

Como trabalhos futuros, pode-se citar a validação do modelo de gerenciamento de processo de *software* dentro do ambiente DiSEN, utilizando técnicas de validação apropriadas; e o detalhamento do componente Gerenciador de Contratos Eletrônicos, de forma que o mesmo possa ser integrado ao ambiente DiSEN.

REFERÊNCIAS

ARMENISE, P. *Software Processes Representation Languages: Survey and Assessment*. In: *International conference on Software Engineering and Knowledge Engineering*, 4., 1992, Capri, Italy. Proceedings... [S.l.]: IEEE Press, June 1992.

BURDETT, D., CHEN, Q., HSU, M. *Conductor: An Enabler for Web Services-based Business Collaboration*. IEEE Data(base) Engineering Bulletin Volume 25, Number 4, December 2002: 22-26. 2002.

CURTIS, B. *Process Modelling*. Communications of the ACM, New York, v. 35, n. 9, Sept. 1992.

D'SOUZA, D.; WILLS, A. *Objects, Components and Frameworks with UML – The Catalysis Approach*. [S.l.]:Addison-Wesley, 1999. 816 p.

DAYAL, U., Hsu, M. Ladin, R. *Business Processo Coordination: State pf tje Art, Trends, and Opens Issues*, VLDB 2001, Proceedings of 27th International Conference n Very Large Data Bases, Roma, Italy. Morgan Kaufmann 2001, ISBN 1-55860-804-4. Pages: 3-13.

DOWSON, M.; NEJMEH, B.; RIDDLE, W. *Fundamental software Process Concepts*. In: EUROPEAN WORKSHOP ON *SOFTWARE* PROCESS MODELLING, 1., 1991, Milan, Italy. Proceedings... [S.l.]: AICA Press, May 1991.

ENAMI, Lúcia Norie Matsueda. *Um modelo de gerenciamento de projetos de software para ambiente distribuído de software*. Dissertação (Mestrado em Ciência da Computação). Universidade Estadual de Maringá, Maringá , 2006.

Extensible Markup Language (XML). http://www.w3.org/TR/REC-xml

FANTINATO, M., TOLEDO, M. B. F., GIMENES, I. M. S.; *Linhas de Pesquisas em Workflows Interorganizacionais*. Relatório Técnico. Setembro 2004.

FANTINATO, M., TOLEDO, M. B. F., GIMENES, I. M. S.; *Arquiteturas de Sistemas de Gerenciamento de Processos de Negócio Baseados em Serviços*. Relatório Técnico. Abril 2005.

FEILER, P. H.; HUMPHREY, W.S. Software Process Development and Enactment: Concepts and Definitions. In: INTERNATIONAL CONFERENCE ON THE *SOFTWARE* PROCESS, 2., 1993, Berlin. Proceedings... Berlin: IEEE Computer Society Press, Mar. 1993.

GANESARAJAH, D., LUPU, E. *Workflow-Based Composition of Web-Services: A Business Model or a Programming Paradigm*? 6th International Enterprise Distributed Object Computing Conference (EDOC 2002), 17-20 September 2002, Lausanne, Switzerland, Proceedings. IEEE Computer Society 2002, ISBN 0-7695-1742-0: 273-284.

GIMENES, I.M.S. *Uma introdução ao Processo de Engenharia de Software: Ambientes e Formalismos.* Caxambu-MG: SBC, 1994. 42f. Trabalho apresentado na Jornada de Atualização em Informática, 13., 1994, Caxambu.

GIMENES, I. *O Processo de Engenharia de software: Ambientes e Formalismos*, XIII Jornada de Atualização em Informática, XIV Congresso da Sociedade Brasileira de Computação, Caxambu-MG, 1994.

GIMENES, I. M. S. et. al. *Um Padrão para Definição de um Gerenciador de Processos de Software*. In: WORKSHOP IBERO AMERICANO DE ENGENHARIA DE REQUISITOS E AMBIENTES DE *SOFTWARE*, 2., 1999. Memorias... Cartago: Cit, 1999.

GIMENES, I. M. S., HUZITA, E.H.M., CARNIELLO, A., FANTINATO, M. "ExPSEE – *An Experimental Process Centred Software Engineering Environment*", Relatório final, Março, 1999.

GRAVENA, J. P., *Aspectos Importantes de uma Metodologia para Desenvolvimento de Software com Objetos Distribuídos*. Trabalho de Graduação. Departamento de Informática – Universidade Estadual de Maringá, 2000.

HERBSLEB, J.D., Grinter, R. "*Splitting the organization and integrating the code: Conway's Law revisited*", In: ICSE 1999, Carolina do Norte, EUA, 1999.

HERBSLEB, J. D., Moitra, D. "*Global software Developmen*t", IEEE *Software*, March/April, EUA, 2001, p. 16-20.

HUZITA, E.H.M. MOOPP - *Uma Metodologia para Auxiliar o Desenvolvimento de Aplicações para Processamento Paralelo*. Tese (Doutorado) - Escola Politécnica. São Paulo: Universidade de São Paulo, 1995.

HUZITA, E. H. M. *Uma Metodologia para o Desenvolvimento Baseado em Objetos Distribuídos Inteligentes*. Projeto de pesquisa em andamento, Universidade Estadual de Maringá. Departamento de Informática, 1999.

JACOBSON, I; BOOCH, G.; RUMBAUGH, J. *The Unified software Development Process*, Addison Wesley, 1999.

KAROLAK, D. W. "*Global Software Development – Managing Virtual Teams and Environments*". Los Alamitos, IEEE Computer Society, EUA, 1998, 159p.

KIEL, L. "*Experiences in Distributed Development: A Case Study*", In: Workshop on Global *Software* Development at ICSE, Oregon, EUA, 2003, 4p.

Language for *Web* Services (BPEL4WS). http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

LONCHAMP, J. *A Structured Conceptual and Terminological Framework for the Software Process Engineering.* In: INTERNATIONAL CONFERENCE ON THE *SOFTWARE* PROCESS, 2., 1993, Berlin. Proceedings... Berlin: IEEE Computer Society Press, Mar. 1993.

MALDANER, L. A., PASQUAL, E. S. *Uma Análise de Linguagens de Composição de Serviços: A Utilização de BPEL e YAWL*. II Congresso Sul Catarinense de Computação, 2006.

McGREGOR, C., KUMARAN, S. *Business Process Monitoring Using Web Services in B2B e-Commerce*. 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), 15-19 April 2002, Fort Lauderdale, FL, USA, CDROM/ Abstracts Proceedings. IEEE Computer Society 2002, ISBN 0-7695-1573-8.

NETBEANS Home Page – http://netbeans.org - 2006.

OCAMPO, C.; BOTELLA, P. *Some Reflections on applying Workflow Technology to software Process*. Internal Report LSI-98-5-R, Department de Lenguatges i Sistemes Informàtics, UPC, Barcelona, Feb 1998.

OMG - *Object Management Group* – Agent Platform Special Interest Group. Agent Technology – Green Paper. Versão 1.0, Setembro 2000.

OSTERWEIL, Leon. *Software Processes are Software Too*. INTERNATIONAL CONFERENCE ON *SOFTWARE* ENGINEERING,9., 1987, Monterey, California. Proceedings... Monterey: IEEE Computer Society Press, 1987.

PAPAZOGLOU, M., GEORGAKOPOULOS, D. *Service-oriented computing*, Communications of the ACM: Service-Oriented Computing 46 (10) (2003) 25-28.

PASCUTTI, M. C. D. *Uma Proposta de Arquitetura de um Ambiente de Desenvolvimento de software Distribuído Baseado em Agentes*. Porto Alegre, 2002. Dissertação (Mestrado em Ciência da Computação), Universidade Federal do Rio Grande do Sul.

PIRES, P. F., BENEVIDES, M. R. F., MATTOSO, M. *Building Reliable Web Services Compositions. Web, Web-Services, and Database Systems*, NODe 2002 *Web* and Database-Related Workshops, Erfurt, Germany, October 7-10, 2002, Revised Papers. Lecture Notes in Computer Science 2593 Springer 2003, ISBN 3-540-00745-8: 59-72.

POSEIDON Home Page - http://www.gentleware.com - 2006.

RECKZIEGEL,M. Home Page - http://www.imasters.com.br/artigo/4422/*Web*services/descrevendo_um_*Web*_service_-_wsdl/ - 2006.

REIS, C. A. L. *Introdução à Modelagem de Processos de software*. Belém, 2004.

Simple Object Access Protocol (SOAP). http://www.w3.org/TR/SOAP.

ROSA, A. C. A. et al. *Requisitos para uma Linguagem de Programação de Processo de software*. In: ENCONTRO INTERUNIVERSITÁRIO DE INFORMÁTICA DO PARANÁ, 2., 1994, Maringá, Paraná. Anais... Maringá: [S.n.], 1994.

ROSS, D.; SCHOMAN, K. *Structured Analysis for Requirements Definition*. IEEE Transactions on *Software* Engineering, New York, v. 3, n.1, Jan.1977. p. 6-15

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

SUTTON, S.; OSTERWEIL, L. *An Analysis of Process Languages*. Disponível na Internet por www em http://laser.cs.mass.edu/abstracts/process.html

TANAKA, S. A. *Um Framework para Desenvolvimento de Gerenciadores de Workflow*. Porto Alegre: CPGCC-UFRGS, 1999, 64p. (TI-857).

TOYOTA, C. M., ROSA, A. C. A. *Projeto de uma Linguagem Multiparadigma para Programação do Processo de Software*. In: ENCONTRO INTERUNIVERSITÁRIO DE INFORMÁTICA DO PARANÁ, 3., 1995, Curitiba, Paraná. Anais... Curitiba: [S.n.], 1995.

Universal Description, Discovery, and Integration (UDDI). http://www.udddi. org.

Web Services Description Language (WSDL). http://www.w3.org/TR/2002/WD-wsdl12-20020709/#intro

Web Services Flow Language (WSFL). http://www-4.ibm.com/*software*/solutions/*Web*services/pdf/WSFL.pdf.

Web Services for Business Process Design (XLANG). http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

WEISS, G. M., *Um Padrão para Definição de um Gerenciador de Processos de Software*, Trabalho de Graduação, Departamento de Informática, Universidade Estadual de Maringá, 1998.

WORKFLOW MANAGEMENT COALITION. *Workflow* Reference Model. Document number TC00-1003, January, 1995.

ZHAO, X., LIU, C., YANG, Y. *Web Service Based Architecture for Workflow Management Systems. Database and Expert Systems Applications*, 15th International Conference, DEXA 2004 Zaragoza, Spain, August 30-September 3, 2004, Proceedings. Lecture Notes in Computer Science 3180 Springer 2004, ISBN 3-540-22936-1: 34-43.

APÊNDICE A

Atividades Primitivas do BPEL4WS:

```
<invoke partnerLink = "ncname" portType="qname" operation="ncname"
                inputVariable="ncname"? outputVariable="ncname"?
                standard-attributes>
        standard-elements
        <correlations>?
                <correlation set = "ncname" initiate = "yes|no"?
                        pattern="in|out|out-in"/>+
        </correlations>
        <catch faultName= "qname" faultVariable="ncname"?>*
                activity
        </catch>
        <catchAll>?
                activity
        </catchAll>
        <compensationHandler>?
                activity
        </compensationHandler>
</invoke>
```

Figura A.1: Atividade <invoke>

```
<receive partnerLink = "ncname" portType="qname" operation="ncname"
                variable="ncname"? createInstance="yes|no"?
                standard-attributes>
        standard-elements
        <correlations>?
        <correlation set = "ncname" initiate = "yes|no"?>+
        </correlations>
</receive>
```

Figura A.2: Atividade <receive>

```
 <reply partnerLink = "ncname" portType="qname" operation="ncname"
                variable="ncname"? faultName="qname"?
                standard-attributes>
        standard-elements
        <correlations>?
                <correlation set = "ncname" initiate = "yes|no"?>+
        </correlations>
</reply>
```

Figura A.3: Atividade <reply>

```
<wait (for="duration-expr" | until= "deadline-expr") standard-attributes>
        standard-elements
</wait>
```

Figura A.4: Atividade <wait>

```
<assign standard-attributes>
        standard-elements
        <copy>+
                from-spec
                to-spec
        </copy>
</assign>
```

Figura A.5: Atividade <assign>

```
<throw faultName="qname"faultVariable="ncname"? standard-attributes>
        standard-elements >
</throw>
```

Figura A.6: Atividade < throw >

```
<empty standard-attributes>
        standard-elements
</empty>
```

Figura A.7: Atividade <empty>

Atividades Estruturais do BPEL4WS:

```
<sequence standard-attributes>
        standard-elements
        activity+
</sequence>
```

Figura A.8: Atividade <sequence>

```
<switch standard-attributes>
        standard-elements
        <case condition="bool-expr">+
                activity
        </case>
        <otherwise>?
                activity
        </otherwise>
</switch>
```

Figura A.9: Atividade <switch>

```
<pick createInstance="yes|no"? standard-attributes>
        standard-elements
        <onMessage partnerLink="ncname" portType="qname"
                operation="ncname"variable="ncname"?>+
                <correlations>?
                        <correlation set="ncname" initiate="yes|no"?>+
                </correlations>
                activity
        </onMessage>
        <onAlarm (for="duration-expr" | until= "deadline-expr") >*
                activity
        </onAlarm>
</pick>
```

Figura A.10: Atividade <pick>

```
<while condition="bool=expr" standard-attributes>
        standard-elements
        activity
</while>
```

Figura A.11: Atividade <while>

```
<flow standard-attributes>
        standard-elements
        <links>?
                <link name="ncname">+
        </links>
        activity+
</flow>
```

Figura A.12: Atividade <flow>

```
<scope variableAcsessSerializable="yes|no"? standard-attributes>
        standard-elements
        <variables>?
                ... see above under <process> for syntax...
        </variables>
        <correlationSet>?
                ... see above under <process> for syntax...
        </correlationSet>
        <faultHandler>?
                ... see above under <process> for syntax...
        </ faultHandler >
        <compensationHandler>?
                ... see above under <process> for syntax...
        </ compensationHandler >
        <eventHandler>?
                ...
        </eventHandler >
        activity
</scope>
```

Figura A.13: Atividade <scope>

APÊNDICE B

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"    xmlns:xsi=http://www.w3.org/2001/XMLSchema-
        instance
        xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:tns="urn:server.hello"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:server.hello">
    <types>
        <xsd:schema targetNamespace="urn:server.hello">
            <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
            <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
        </xsd:schema>
    </types>
    <message name="helloRequest">
        <part name="name" type="xsd:string" />
    </message>
    <message name="helloResponse">
        <part name="return" type="xsd:string" />
    </message>
    <portType name="server.helloPortType">
        <operation name="hello">
            <documentation>Retorna o nome</documentation>
            <input message="tns:helloRequest" />
            <output message="tns:helloResponse" />
        </operation>
    </portType>
    <binding name="server.helloBinding" type="tns:server.helloPortType">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
        <operation name="hello">
            <soap:operation soapAction="urn:server.hello#hello" style="rpc" />
            <input>
                <soap:body use="encoded" namespace="urn:server.hello"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:server.hello"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
    </binding>
    <service name="server.hello">
        <port name="server.helloPort" binding="tns:server.helloBinding">
            <soap:address location="http://localhost/imasters2/nuSOAP/server2.php" />
        </port>
    </service>
</definitions>
```

APÊNDICE C

Processo de *Software* utilizando BPEL:

```
<?xml version="1.0" encoding="UTF-8"?>
<process
   name="ProcessoSoftwareService"
   targetNamespace="http://enterprise.netbeans.org/bpel/ProcessoSoftwareService"
   xmlns="http://schemas.xmlsoap.org/ws/2004/03/business-process/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:bpws="http://schemas.xmlsoap.org/ws/2004/03/business-process/"
   xmlns:wsdlNS="http://enterprise.netbeans.org/bpel/ProcessoSoftware"
   xmlns:ns1="http://j2ee.netbeans.org/wsdl/RequisitosService"
   xmlns:ns2="http://j2ee.netbeans.org/wsdl/ProcessoSoftwareService"
   xmlns:ns3="http://j2ee.netbeans.org/wsdl/AnaliseService"
   xmlns:ns4="http://j2ee.netbeans.org/wsdl/ProjetoService"
   xmlns:ns5="http://j2ee.netbeans.org/wsdl/ImplementacaoService">
    <import namespace="http://j2ee.netbeans.org/wsdl/RequisitosService"     location="RequisitosService.wsdl"
                importType="http://schemas.xmlsoap.org/wsdl/"/>
   <import namespace="http://j2ee.netbeans.org/wsdl/AnaliseService" location="AnaliseService.wsdl"
                importType="http://schemas.xmlsoap.org/wsdl/"/>
   <import namespace="http://j2ee.netbeans.org/wsdl/ProjetoService" location="ProjetoService.wsdl"
                importType="http://schemas.xmlsoap.org/wsdl/"/>
   <import namespace="http://j2ee.netbeans.org/wsdl/ImplementacaoService"
                location="ImplementacaoService.wsdl"
                importType="http://schemas.xmlsoap.org/wsdl/"/>
   <import namespace="http://j2ee.netbeans.org/wsdl/ProcessoSoftwareService" location="ProcessoSoftwareService.wsdl"
                importType="http://schemas.xmlsoap.org/wsdl/"/>

   <partnerLinks>
         <partnerLink name="Requisitos" partnerLinkType="ns1:RequisitosPartnerLinkType"
                              myRole="RequisitosCallbackServiceRole"
                              partnerRole="RequisitosServiceRole"/>
         <partnerLink name="ProcessoSoftware" partnerLinkType="ns2:ProcessoSoftwarePartnerLinkType"
                              myRole="ProcessoSoftwareServiceRole"/>
         <partnerLink name="Analise" partnerLinkType="ns3:AnalisePartnerLinkType" myRole="AnaliseCallbackServiceRole"
                              partnerRole="AnaliseServiceRole"/>
         <partnerLink name="Projeto" partnerLinkType="ns4:ProjetoPartnerLinkType" myRole="ProjetoCallbackServiceRole"
                              partnerRole="ProjetoServiceRole"/>
         <partnerLink name="Implementacao" partnerLinkType="ns5:ImplementacaoPartnerLinkType"
                              myRole="ImplementacaoCallbackServiceRole" partnerRole="ImplementacaoServiceRole"/>
   </partnerLinks>

   <variables>
    <variable name="ElaborarDiagramaComponentesOK" messageType="ns4:ElaborarDiagramaComponentesOK"/>
    <variable name="ElaborarModeloProjetoOK" messageType="ns3:ElaborarModeloProjetoOK"/>
    <variable name="ElaborarModeloAnaliseOK" messageType="ns1:ElaborarModeloAnaliseOK"/>
    <variable name="ElaborarModeloNegociosOK" messageType="ns1:ElaborarModeloNegociosOK"/>
    <variable name="ProcessoSoftwareIn" messageType="ns2:ProcessoSoftwareIn"/>
    <variable name="ProcessoSoftwareOut" messageType="ns2:ProcessoSoftwareOut"/>
    <variable name="ElaborarModeloAnalise" messageType="ns1:ElaborarModeloAnalise"/>
    <variable name="ElaborarModeloNegocios" messageType="ns1:ElaborarModeloNegocios"/>
    <variable name="ElaborarModeloProjeto" messageType="ns3:ElaborarModeloProjeto"/>
    <variable name="ElaborarDiagramaComponentes" messageType="ns4:ElaborarDiagramaComponentes"/>
   </variables>

   <sequence>
            <receive name="IniciarProcesso" createInstance="yes"
                        partnerLink="ProcessoSoftware"
                        operation="iniciarProcessoSoftware"
                        portType="ns2:ProcessoSoftwarePortType"
                        variable="ProcessoSoftwareIn"/>
```

```
    <assign name="CopiarProcessoSoftware">
            <copy>
                <from variable="ProcessoSoftwareIn"/>
                <to variable="ProcessoSoftwareOut"/>
            </copy>
            <copy>
            <from variable="ProcessoSoftwareIn"/>
                <to variable="ElaborarModeloNegocio"/>
            </copy>
            <copy>
            <from variable="ProcessoSoftwareIn"/>
            <to variable="ElaborarModeloAnalise"/>
            </copy>
            <copy>
                <from variable="ProcessoSoftwareIn"/>
                <to variable="ElaborarModeloProjeto"/>
            </copy>
            <copy>
                <from variable="ProcessoSoftwareIn"/>
                <to variable="ElaborarDiagramaComponentes"/>
            </copy>
  </assign>
  <repeatUntil name="Requisitos">
            <sequence name="RequisitosOK">
                <invoke name="ElaborarModeloNegocios"
                    partnerLink="Requisitos"
                    operation="elaborarModeloNegocios"
                    portType="ns1:RequisitosPortType"
                    inputVariable="ElaborarModeloNegocios"/>
                    <receive name="ReceberAnalise" createInstance="no"
                        partnerLink="Requisitos"
                        operation="analiseOK"
                        portType="ns1: RequisitosCallbackPortType"
                        variable="ElaborarModeloNegociosOK"/>
                        <assign name="CopiarRequisitosOK">
                            <copy>
                                <from variable="ElaborarModeloNegociosOK"/>
                                <to variable="ProcessoSoftwareOut"/>
                            </copy>
                            <copy>
                                <from variable="ElaborarModeloNegociosOK"/>
                                <to variable="ElaborarModeloAnalise"/>
                            </copy>
                        </assign>
            </sequence>
            <condition>   (ElaborarModeloNegociosOK = 'ElaborarModeloNegociosOK' )   </condition>
  </repeatUntil>
  <sequence name="Analise">
      <invoke name="ElaborarModeloAnalise "
            partnerLink=" Analise "
            operation=" elaborarModeloAnalise "
            portType="ns2 AnalisePortType"
            inputVariable=" ElaborarModeloAnalise "/>
    <receive name="ReceberAnalise "  createInstance="no"
            partnerLink=" Analise "
            operation=" AnaliseOK"
            portType="ns2: AnaliseCallbackPortType"
            variable=" ElaborarModeloAnaliseOK"/>
    <assign name="CopiarAnaliseOK">
            <copy>
                <from variable=" ElaborarModeloAnaliseOK"/>
                <to variable="ProcessoSoftwareOut"/>
            </copy>
            <copy>
                <from variable=" ElaborarModeloAnaliseOK"/>
                <to variable="ElaborarProjeto"/>
            </copy>
    </assign>
</sequence>
```

```
        <sequence name="Projeto">
                <invoke name="ElaborarModeloProjeto"
                        partnerLink="Projeto"
                        operation="elaborarModeloProjeto"
                        portType="ns3:ProjetoPortType"
                        inputVariable="ElaborarModeloProjeto"/>
                <receive name="ReceberProjeto"  createInstance="no"
                        partnerLink="Projeto"
                        operation="projetoOK"
                        portType="ns3:ProjetoCallbackPortType"
                        variable="ElaborarModeloProjetoOK"/>
                <assign name="CopiarProjetoOK">
                        <copy>
                                <from variable="ElaborarModeloProjetoOK"/>
                                <to variable="ProcessoSoftwareOut"/>
                        </copy>
                        <copy>
                                <from variable="ElaborarModeloProjetoOK"/>
                                <to variable="ElaborarDiagramaComponentes"/>
                        </copy>
        </assign>
       </sequence>
       <while name="Implementacao">
                <condition>  (ElaborarDiagramaComponentesOk != 'ElaborarDiagramaComponentesOK' )  </condition>
                <sequence name="ImplementacaoOK">
                <invoke name="ElaborarDiagramaComponentes"
                                partnerLink="Implementacao"
                                operation="elaborarDiagramaComponentes"
                                portType="ns4: ImplementacaoPortType"
                                inputVariable="ElaborarDiagramaComponentes"/>
                <receive name="ReceberImplementacao"  createInstance="no"
                                partnerLink="Implementacao"
                                operation="implementacaoOK"
                                portType="ns4: ImplementacaoCallbackPortType"
                                variable="ElaborarDiagramaComponentesOK"/>
                <assign name="CopiarImplementacaoOK">
                                <copy>
                                        <from variable="ElaborarDiagramaComponentesOK"/>
                                        <to variable="ProcessoSoftwareOut"/>
                                </copy>
                </assign>
                </sequence>
    </while>
     <reply name="FinalizarProcessoSoftware" partnerLink="ProcessoSoftware" operation="iniciarProcessoSoftware"
     portType="ns2:ProcessoSoftwarePortType" variable="ProcessoSoftwareOut"/>
  </sequence>
</process>
```

Fase de Requisitos em WSDL:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Requisitos" targetNamespace="http://j2ee.netbeans.org/wsdl/AnaliseService"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd=http://www.w3.org/2001/XMLSchema
    xmlns:tns="http://j2ee.netbeans.org/wsdl/RequisitosService"
    xmlns:plink="http://schemas.xmlsoap.org/ws/2004/03/partner-link/">
   <types/>
   <message name="ElaborarModeloNegocios">
           <part name="Joao" type="xsd:string"/>
           <part name="Poseidon" type="xsd:string"/>
   </message>
   <message name="ElaborarModeloNegociosOK">
           <part name="ModeloNegocios" type="xsd:string"/>
   </message>
   <portType name="RequisitosPortType">
     <operation name="elaborarModeloNegocios">
       <input message="tns: ElaborarModeloNegocios"/>
```

```
    </operation>
  </portType>
  <portType name="RequisitosCallbackPortType">
        <operation name="requisitosOK">
            <input message="tns: ElaborarModeloNegociosOK"/>
        </operation>
  </portType>
  <binding name="RequisitosSoapBinding" type="tns: RequisitosPortType">
                <soap:binding style="document"    transport="http://schemas.xmlsoap.org/soap/http"/>
                    <operation name="elaborarModeloNegocios">
                     <soap:operation soapAction="http://j2ee.netbeans.org/wsdl/Analise/elaborarModeloNegocios"
            style="document"/>
                                <input>
                                        <soap:body use="literal"/>
                                </input>
                    </operation>
  </binding>
  <binding name="RequisitosCallbackSoapBinding" type="tns: RequisitosCallbackPortType">
            <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
                    <operation name="requisitosOK">
                                <soap:operation/>
                                <input>
                                        <soap:body use="literal"/>
                                </input>
                    </operation>
 </binding>
 <service name="RequisitosSoapService">
            <port name="RequisitosSoapHttpPort" binding="tns: RequisitosSoapBinding">
                    <soap:address location="http://localhost:8080/webservice/RequisitosService"/>
            </port>
 </service>
 <plink:partnerLinkType name="RequisitosPartnerLinkType">
       <plink:role name="RequisitosServiceRole"
              portType="tns: RequisitosPortType"/>
      <plink:role name="RequisitosCallbackServiceRole"
             portType="tns: RequisitosCallbackPortType"/>
 </plink:partnerLinkType>
</definitions>
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo