

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
MESTRADO EM INFORMÁTICA**

VICTORIO ALBANI DE CARVALHO

**GERÊNCIA DE CONHECIMENTO E DECISÃO EM GRUPO:
UM ESTUDO DE CASO NA GERÊNCIA DE PROJETOS**

**VITÓRIA
2006**

VICTORIO ALBANI DE CARVALHO

**GERÊNCIA DE CONHECIMENTO E DECISÃO EM GRUPO:
UM ESTUDO DE CASO NA GERÊNCIA DE PROJETOS**

Dissertação apresentada ao Mestrado em
Informática da Universidade Federal do Espírito
Santo, como requisito parcial para obtenção do
Grau de Mestre em Informática.

Orientador: Prof. Dr. Ricardo de Almeida Falbo.

VITÓRIA
2006

VICTORIO ALBANI DE CARVALHO

**GERÊNCIA DE CONHECIMENTO E DECISÃO EM GRUPO:
UM ESTUDO DE CASO NA GERÊNCIA DE PROJETOS**

COMISSÃO EXAMINADORA

Prof. Ricardo de Almeida Falbo, D. Sc.
Orientador

Prof. Davidson Cury, D.Sc.
Universidade Federal do Espírito Santo - UFES

Profa. Renata Mendes de Araujo, D. Sc.
Universidade Federal do Estado do Rio de Janeiro - UNIRIO

Vitória, 27 de novembro de 2006.

AGRADECIMENTOS

Aos meus pais e minha irmã, minhas fontes inesgotáveis de carinho e de apoio incondicional;

A todos os amigos e familiares pelo apoio e por compreenderem minha ausência;

Ao meu orientador Ricardo Falbo, por ter sido, além de orientador e professor, um grande amigo e incentivador durante esses anos de trabalho;

Aos meus amigos do projeto ODE, em especial Julio, Bruno Carreira, Bruno Segrini, Lucas e Aline pela colaboração, apoio e companheirismo;

A todos os professores da UFES, pelos ensinamentos;

Aos funcionários da UFES por toda a ajuda;

Ao Flávio Amaral, ao Carlos Mattos e ao Leonardo Bueno, gerentes respectivamente da Unisys, da Xerox e da Infobase, pela flexibilização dos horários de trabalho que me permitiu concluir o mestrado.

RESUMO

Tendo em vista a complexidade das atividades de gerência e a quantidade de variáveis envolvidas nessas atividades, é essencial que o gerente de projetos conte com algum tipo de apoio automatizado para realizá-las. Durante a realização de um projeto de software, muito conhecimento é produzido e utilizado. Visando à reutilização desse conhecimento em projetos futuros, é fundamental que sejam providos meios de se reter e armazenar o conhecimento gerado, de forma a minimizar o esforço para obtê-lo no futuro. Neste contexto, a gerência de conhecimento pode ser usada para capturar o conhecimento e a experiência gerada e acumulada durante o processo de software e promover o surgimento de novo conhecimento. A experiência constitui um fator de fundamental importância para que as atividades de gerência sejam realizadas com sucesso. Assim, os benefícios alcançados pela troca de idéias durante a realização dessas atividades são evidentes.

Durante este trabalho, visando ao apoio de gerência de conhecimento à gerência de projetos de software no ambiente de desenvolvimento de software ODE, foram desenvolvidas e integradas a ODE uma infra-estrutura para caracterização de itens de software e busca de itens similares e uma infra-estrutura de apoio à decisão em grupo.

Para avaliar o potencial dessas infra-estruturas, foram conduzidas especializações das mesmas, respectivamente, para caracterização de projetos e para a elaboração cooperativa de planos de riscos.

Palavras-chave: Gerência de Projetos, Gerência de Conhecimento, Ambientes de Desenvolvimento de Software, Raciocínio Baseado em Casos, Decisão em Grupo.

ABSTRACT

Due to the complexity and the high number of variables involved in the management activities, it is essential to the project manager to have some kind of automated support to perform her tasks. During the accomplishment of a software project, a high amount of knowledge is produced and used. Looking for the reuse of that knowledge in future projects, we need to provide means to retain and store the generated knowledge in a way to minimize the effort to obtain it in the future. In this context, knowledge management can be used to capture the knowledge and experience generated and accumulated during the software process and to promote the appearance of new knowledge. Experience constitutes a key factor in order to management activities can be accomplished with success. Thus, the benefits reached by the change of ideas during the accomplishment of those activities are evident.

During this work, in order to support software project management using knowledge management in the software development environment ODE, we have developed and integrated to ODE an infrastructure to support software items characterization and search for similar items and an infrastructure to support group decision.

To evaluate the potential of these infrastructures, we specialized them, respectively, to support project characterization and cooperative elaboration of risk plans.

Key Words: Project Management, Knowledge Management, Software Development Environments, Case Based Reasoning, Group Decision.

SUMÁRIO

Capítulo 1 – Introdução	01
1.1 Motivação.....	02
1.2 Contexto e Objetivos.....	03
1.3 Metodologia.....	05
1.4 Organização do Trabalho	07
 Capítulo 2 – Gerência de Projetos e Ambientes de Desenvolvimento de Software.....	 09
2.1 Gerência de Projetos de Software.....	10
2.1.1 Definição do Processo e do Escopo do Projeto.....	11
2.1.2 Realização de Estimativas.....	12
2.1.3 Elaboração e Acompanhamento de Cronograma.....	17
2.1.4 Gerência de Riscos.....	19
2.1.5 Elaboração do Plano de Projeto.....	20
2.1.6 Acompanhamento e Encerramento do Projeto.....	21
2.2 Gerência de Projetos e Normas e Modelos de Qualidade de Processo.....	22
2.2.1 Gerência de Projetos no CMMI.....	23
2.2.2 Gerência de Projetos no MPS.BR.....	26
2.2.3 Gerência de Projetos na ISO/IEC 12207.....	27
2.2.4 Gerência de Projetos e o PMBOK.....	27
2.3 Automação do Processo de Software.....	30
2.4 Automação da Gerência de Projetos.....	34
2.5 O Ambiente ODE.....	36
2.6 Considerações Finais do Capítulo.....	37
 Capítulo 3 – Gerência de Conhecimento	 39
3.1 Conceitos Relativos à Gerência de Conhecimento.....	40
3.1.1 Dado, Informação e Conhecimento.....	41
3.2 Processo de Gerência de Conhecimento.....	43
3.3 Abordagens e Ferramentas Utilizados para Apoiar a Gerência de Conhecimento.....	45

3.3.1	Raciocínio Baseado em Casos.....	45
3.3.2	Design <i>Rationale</i>	46
3.3.3	Ferramentas de <i>Groupware</i>	48
3.4	Gerência de Conhecimento em Engenharia de Software.....	50
3.5	Gerência de Conhecimento em ODE.....	54
3.6	Considerações Finais do Capítulo.....	57
Capítulo 4 – Infra-estrutura para Caracterização de Itens de Software em ODE.....		60
4.1	A Utilização do Raciocínio Baseado em Casos no Desenvolvimento da Infra- Estrutura de Caracterização de Itens de Software em ODE.....	61
4.2	A Infra-Estrutura Desenvolvida.....	63
4.2.1	Cadastro de Características.....	64
4.2.2	Caracterização e Busca.....	73
4.3	Definição de um Conjunto Inicial de Características Aplicáveis a Projetos em ODE.....	79
4.4	Considerações Finais do Capítulo.....	83
Capítulo 5 – Infra-estrutura de Apoio à Decisão em Grupo em ODE.....		85
5.1	Adaptação da Técnica Delphi para Apoiar a Construção Cooperativa de Artefatos de Software.....	86
5.2	A Infra-estrutura de Apoio à Decisão em Grupo Desenvolvida.....	89
5.2.1	Preparar Reunião.....	90
5.2.2	Controlar Reunião.....	92
5.2.3	Participar da Reunião.....	95
5.3	Apoio à Elaboração Cooperativa de Planos de Risco.....	100
5.4	Captura do Raciocínio.....	109
5.5	Considerações Finais do Capítulo.....	116
Capítulo 6 – Considerações Finais.....		117
6.1	Conclusões.....	117
6.2	Perspectivas Futuras.....	120
Referências Bibliográficas		123

LISTA DE FIGURAS

Figura 2.1 – Áreas de Processos Básicas do Gerenciamento de Projetos do CMMI.....	24
Figura 2.2 – Áreas de Processos Avançadas de Gerenciamento de Projetos do CMMI.....	25
Figura 3.1 – Espiral do Conhecimento (NONAKA et al., 1997).....	42
Figura 3.2 – Sistema de Gerência de Conhecimento de ODE.....	54
Figura 3.3 – Infra –estrutura de Gerência de Conhecimento de ODE.....	56
Figura 4.1 – Diagrama de Pacotes.....	64
Figura 4.2 – Diagrama de Casos de Uso do Cadastro de Características.....	65
Figura 4.3 – Janela Principal do Subsistema Cadastro de Característica.....	66
Figura 4.4 – Diagrama de Classes do subsistema Cadastro de Característica.....	67
Figura 4.5 – Cadastro de Característica de Valor Ordenado.....	68
Figura 4.6 – Cadastro de Característica de Valor Não Ordenado.....	69
Figura 4.7 – Cadastro de Característica de Conhecimento.....	70
Figura 4.8 – Janela de Edição de Tabelas de Similaridades.....	71
Figura 4.9 – Janela de Edição de Peso para Similaridade por Atividade.....	72
Figura 4.10 – Diagrama de Casos de Uso da Caracterização e Busca.....	73
Figura 4.11 – Diagrama de Classes do subsistema Caracterização e Busca.....	74
Figura 4.12 – Caracterização de Projetos em ODE.....	75
Figura 4.13 – Busca por Projetos Similares.....	77
Figura 5.1 – Funcionalidades Principais da Infra-estrutura.....	89
Figura 5.2 – Diagrama de Casos de Uso detalhando o Caso de Uso Preparar Reunião.....	90
Figura 5.3 – Diagrama de Classes Parcial da Infra-estrutura.....	91
Figura 5.4 – Funcionalidades para Controle de Reuniões pelo Moderador.....	92
Figura 5.5 – Ciclo de Vida de uma Reunião de Decisão em Grupo.....	94
Figura 5.6 – Funcionalidades Providas para Permitir a Participação em Reuniões.....	96
Figura 5.7 – Diagrama de Classes Completo da Infra-estrutura.....	99
Figura 5.8 – Criando uma Nova Reunião.....	101
Figura 5.9 – Funcionalidades Oferecidas para o Moderador da Reunião.....	102

Figura 5.10 – Funcionalidades Oferecidas para os Participantes da Reunião.....	103
Figura 5.11 – Janela de Análise de Impacto das Características do Escopo sobre a Proposta.....	104
Figura 5.12 – Casos de Uso Especializados para Apoiar a Elaboração Cooperativa de Planos de Riscos.....	105
Figura 5.13 – Modelo de Classes de Planos de Riscos.....	106
Figura 5.14 – Interface de Geris no Momento de uma Avaliação de um Risco.....	107
Figura 5.15 – Informações sobre o Projeto Relevantes para a Análise de Riscos.....	108
Figura 5.16 – Relatório Sumário sobre as Propostas de Planos de Riscos da Rodada...	109
Figura 5.17 – Novo Item de Conhecimento Acrescentada à Infra-estrutura de Gerência de Conhecimento de ODE.....	113
Figura 5.18 – Layout Básico de uma História de Reunião.....	114
Figura 5.19 – Layout de uma História de Reunião de Elaboração Cooperativa de Planos de Riscos.....	115

LISTA DE TABELAS

Tabela 4.1– Conjunto Inicial de Características Aplicáveis a Projetos em ODE.....	81
---	----

Capítulo 1

Introdução

Atualmente, vemos uma crescente exigência por qualidade nos produtos de software. Além disso, as restrições de custo e tempo às quais os projetos são submetidos estão cada vez mais severas. Nesse contexto, é evidente a importância da gerência de projetos.

Cabe ao gerente de projetos equilibrar adequadamente o custo, o prazo, a qualidade e o escopo do projeto de modo a satisfazer as expectativas e as necessidades das partes envolvidas. Visando a atingir esse objetivo, o gerente deve realizar diversas atividades, entre elas: definir o escopo do projeto, definir um processo para o projeto, estimar e acompanhar tamanho, esforço, tempo e custos, identificar e monitorar riscos, elaborar e acompanhar o cronograma de execução do projeto, alocar recursos etc.

Tendo em vista a complexidade das atividades de gerência e a quantidade de variáveis envolvidas nessas atividades, é essencial que o gerente de projetos conte com algum tipo de apoio automatizado para realizá-las. No âmbito da gerência de projetos de software, uma forma interessante de se prover esse apoio é por meio de ferramentas integradas a Ambientes de Desenvolvimento de Software (ADSs). Além disso, durante a realização de um projeto de software, muito conhecimento é produzido e utilizado. Visando à reutilização desse conhecimento em projetos futuros, é fundamental que sejam providos meios de capturar, armazenar e recuperar o conhecimento gerado, de forma a minimizar o esforço para obtê-lo no futuro. Neste contexto, a gerência de conhecimento pode ser usada para capturar o conhecimento e a experiência gerada e acumulada durante o processo de software e promover o surgimento de novos conhecimentos.

1.1 Motivação

As ferramentas e técnicas utilizadas no desenvolvimento de novas aplicações de software não têm sido suficientes para garantir o sucesso dos projetos. Problemas significativos ainda têm sido relatados. Observa-se, por exemplo, o não cumprimento de orçamentos, a elaboração de estimativas incorretas ou incoerentes e um número praticamente inaceitável de projetos cancelados, estagnados ou que não tenham atendido às expectativas dos clientes (BARCELLOS, 2003).

Para os executivos de negócio, o número de projetos sem sucesso é um dos pontos mais frustrantes relacionados ao desenvolvimento de software, pois resulta em oportunidades perdidas e insatisfação dos clientes. Assim sendo, além de outros fatores de qualidade, é importante que um produto de software seja desenvolvido com os recursos e o cronograma previstos. De fato, o sucesso de um projeto depende muito da habilidade do gerente em estimar seus custos e prazos no início de seu desenvolvimento e controlá-los ao longo do processo de desenvolvimento (CRUZ, 1998a).

A elaboração dessas estimativas não é uma tarefa simples. São muitas as variáveis e os interesses envolvidos. A situação se agrava ainda mais, porque as estimativas, normalmente, têm de ser realizadas na fase inicial do projeto, quando pouco se sabe sobre o mesmo. Neste contexto, a existência de uma base de dados históricos, contendo dados confiáveis sobre projetos anteriores é de fundamental importância. Utilizando-se do conhecimento armazenado nessa base, o gerente de projetos não fica limitado às suas próprias experiências para realizar as estimativas.

Os benefícios alcançáveis através da utilização de dados históricos não se restringem à realização de estimativas. Experiências anteriores podem guiar as decisões dos gerentes em todas as atividades referentes ao planejamento e ao acompanhamento de um projeto. Os engenheiros de software podem identificar e reutilizar artefatos de software de projetos anteriores cujos requisitos sejam similares. Enfim, as experiências obtidas em projetos anteriores podem ser utilizadas para facilitar e agilizar o trabalho em um novo projeto e, até mesmo, evitar a repetição de erros. Assim, é de fundamental importância gerenciar o conhecimento organizacional, visando a viabilizar a reutilização de experiências adquiridas em projetos anteriores.

O processo de software é constituído por um conjunto de atividades essencialmente cooperativas, nas quais diversos profissionais atuam em conjunto com o intuito de desenvolver um produto de software de qualidade, obedecendo a restrições de custos e prazos. Equipes de desenvolvimento de software tendem a ser mais produtivas em ambientes onde há colaboração mútua e troca de idéias entre seus membros.

Em atividades nas quais a experiência é um fator de fundamental importância, como as atividades de gerência, os ganhos obtidos através da troca de idéias são evidentes. Uma prova dos benefícios gerados pela colaboração em atividades de gerência é que uma das boas práticas para a geração de estimativas apontadas na literatura (PRESSMAN, 2002) (JØRGENSEN, 2002) orienta a combinar as estimativas de diferentes profissionais, a fim de gerar uma estimativa mais segura. Nesse contexto, fica evidente a importância de apoiar a realização das atividades de gerência de forma cooperativa.

1.2 Contexto e Objetivos

O objetivo principal deste trabalho é prover apoio de gerência de conhecimento para a gerência de projetos de software no ambiente de desenvolvimento de software ODE (*Ontology-based software Development Environment*) (FALBO et al., 2004a). ODE é um ADS Centrado em Processo, desenvolvido no Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES/UFES), que tem sua fundamentação baseada em ontologias.

Em sua fase atual, ODE possui ferramentas de apoio a várias atividades de gerência de projetos de software, a saber: apoio à definição de processos de software (BERTOLLO et al., 2006), apoio ao acompanhamento de projetos (DAL MORO et al., 2005), apoio à gerência de recursos humanos, apoio à gerência de riscos (FALBO et al., 2004b) e apoio à realização de estimativas (CARVALHO et al., 2006). ODE dispõe, ainda, de uma infra-estrutura de gerência de conhecimento, cujo objetivo é apoiar a administração do conhecimento gerado durante o processo de software (NATALI, 2003).

Apesar de ODE apoiar a realização de diversas atividades, há a necessidade de gerenciar o conhecimento envolvido nos processos decisórios inerentes às atividades, com destaque para as atividades da gerência de projetos. Utilizando uma abordagem de gerência de conhecimento, o

conhecimento embutido nas diversas atividades da gerência de projetos pode ser capturado, armazenado, disseminado e reutilizado, para oferecer maiores subsídios às tomadas de decisão, permitindo que as atividades de gerência de projetos sejam realizadas com maior segurança e eficiência.

Além de exigirem a tomada de diversas decisões durante sua execução, as atividades do processo de software são essencialmente cooperativas. Nesse contexto, percebeu-se a necessidade de prover, em ODE, uma infra-estrutura de apoio à decisão em grupo. Assim, foi desenvolvida neste trabalho, uma infra-estrutura desse tipo, baseada na técnica de Delphi (PFLEEGER, 2001), cujos objetivos são:

- Permitir a construção de artefatos de software, com foco em artefatos da gerência de projetos, de forma cooperativa através de reuniões virtuais;
- Capturar as sugestões de todos os profissionais envolvidos na construção do artefato e suas respectivas justificativas buscando, assim, traçar o raciocínio seguido pela equipe até chegar ao artefato final;
- Armazenar as alternativas consideradas durante a construção do artefato;
- Gerar um relatório contendo o histórico da reunião, com o intuito de explicitar o raciocínio seguido pelos participantes.

Para experimentar a infra-estrutura de apoio à decisão em grupo proposta, como estudo de caso, ela foi instanciada para apoiar a elaboração de planos de risco em grupo. A gerência de riscos foi a atividade escolhida para o estudo de caso devido a alguns fatores: (i) trata-se de uma atividade na qual a experiência é muito importante, o que evidencia as vantagens do uso de uma abordagem cooperativa que estimule a troca de experiências; (ii) é uma atividade durante a qual diversas decisões devem ser tomadas; (iii) ODE já conta com uma ferramenta de apoio à gerência de riscos, que pode ser utilizada pela infra-estrutura de apoio à decisão em grupo para a construção dos artefatos.

Como uma questão fundamental para a gerência de conhecimento é a recuperação de itens de conhecimento relevantes para uma dada situação, percebeu-se que era importante, ainda, prover em ODE uma estrutura flexível e poderosa para caracterizar itens de software e calcular similaridade entre eles. Assim, neste trabalho, foi desenvolvida uma infra-estrutura genérica,

utilizando técnicas de raciocínio baseado em casos (WANGENHEIM et al., 2003), para a caracterização de itens relacionados ao processo de software.

Alguns objetivos específicos nortearam a definição da infra-estrutura de caracterização proposta neste trabalho, a saber:

- Apoiar a caracterização de quaisquer itens relacionados ao processo de software, visando a permitir que cada organização defina os tipos de itens que deseja caracterizar, bem como as características a serem utilizadas para cada tipo de item e os valores que cada uma delas pode assumir;
- Calcular o grau de similaridade entre itens, levando em consideração o peso de cada característica no cálculo da similaridade. Os pesos das características devem ser definidos por cada organização e podem variar de acordo com a atividade do processo de software.

A infra-estrutura de caracterização construída neste trabalho foi utilizada inicialmente para aperfeiçoar a funcionalidade de caracterização de projetos de ODE.

1.3 Metodologia

Este trabalho teve início com uma revisão bibliográfica sobre a área de Gerência de Projetos, na qual foram estudados livros, trabalhos acadêmicos e artigos científicos. Dentre os tópicos discutidos nessa revisão bibliográfica, merecem destaque: conceitos de gestão de projetos, métricas de processo e projeto de software, estimativas de software, análise e gestão de riscos, elaboração de cronogramas, acompanhamento de projetos e apoio automatizado ao planejamento de software. O objetivo desse estudo foi entender os conceitos fundamentais das atividades de gerência de projetos de software, buscando identificar os principais desafios e dificuldades enfrentadas pelos gerentes de projetos de software e analisar os benefícios que poderiam ser obtidos por meio de um apoio automatizado a essas atividades. No que se refere ao apoio automatizado, foram revistos os conceitos referentes a ferramentas CASE e a Ambientes de Desenvolvimento de Software.

Terminada essa etapa, iniciou-se uma segunda fase da revisão bibliográfica, agora, sobre a área de gerência de conhecimento, visando a entender os conceitos envolvidos na área e a analisar como a gerência de projetos poderia se beneficiar de uma abordagem de gerência de conhecimento. Foram investigadas, ainda, algumas abordagens utilizadas para apoiar as atividades envolvidas na gerência de conhecimento, dentre as quais merecem destaque: raciocínio baseado em casos (RBC), *design rationale*, *groupware* e agentes.

Identificada a importância da identificação e utilização de dados históricos de projetos anteriores para as atividades de gerência de projetos e dado o fato de que a recuperação de itens de conhecimento relevantes para uma dada situação é tida como uma questão fundamental para a gerência de conhecimento, decidiu-se trabalhar na construção de uma infra-estrutura flexível para caracterizar itens de software e calcular similaridade entre eles.

Como a idéia de utilizar a solução de um problema passado para resolver um problema atual é o cerne da técnica de Raciocínio Baseado em Casos, foram aprofundados os estudos nesta área, dando especial atenção para as técnicas de caracterização de casos e de cálculo de similaridade entre eles.

Durante a revisão bibliográfica foram estudados alguns trabalhos que buscavam definir as características a serem utilizadas para caracterizar projetos e se percebeu não haver um consenso quanto à definição desse conjunto de características. Além disso, não foram encontrados trabalhos que buscassem definir as características a serem utilizadas para a caracterização dos demais itens de software. Assim, ficou evidente a importância de se construir uma infra-estrutura que permitisse que cada organização definisse quais características seriam utilizadas para caracterizar cada tipo de item de software.

Com o intuito de identificar um conjunto inicial de características a serem utilizadas na caracterização de projetos de ODE, foi feito um levantamento, na literatura, de características utilizadas para caracterizar projetos.

A infra-estrutura de caracterização desenvolvida e sua utilização para apoiar a realização de estimativas foi um dos temas do artigo intitulado “EstimaODE: Apoio a Estimativas de Tamanho e Esforço no Ambiente de Desenvolvimento de Software ODE” (CARVALHO et al., 2006), publicado nos anais do V Simpósio Brasileiro de Qualidade de Software realizado de 29

de maio a 03 de junho de 2006 em Vitória. Esse artigo recebeu o prêmio de melhor trabalho técnico do evento.

Finalizada a construção da infra-estrutura de caracterização, foi iniciada a escrita da dissertação. Em paralelo com a elaboração da dissertação, iniciou-se o desenvolvimento da infra-estrutura de apoio à decisão em grupo.

Durante o processo de definição da infra-estrutura de apoio à decisão em grupo, percebeu-se a necessidade de pesquisar modelos de representação de *Design Rationale*. A infra-estrutura de apoio à decisão em grupo foi construída e instanciada para apoiar a elaboração cooperativa de planos de risco.

Dessa maneira, foi integrado ao ambiente ODE um ferramental que visa a apoiar a gerência de projetos, empregando técnicas de gerência de conhecimento para a captura, armazenamento, recuperação e reutilização de conhecimento.

1.4 Organização do Trabalho

Nesta dissertação há, além deste capítulo, mais cinco capítulos, como segue:

- Capítulo 2 – *Gerência de Projetos e Ambientes de Desenvolvimento de Software*: procura dar uma visão geral dos conceitos relativos à gerência de projetos de software e ambientes de desenvolvimento de software e discutir sobre o apoio automatizado à gerência de projetos. Esse capítulo aborda, ainda, a forma como algumas normas e modelos de qualidade tratam a gerência de projetos e apresenta o ambiente ODE.
- Capítulo 3 – *Gerência de Conhecimento*: discute a importância de se gerenciar o conhecimento e como a gerência de conhecimento pode ser aplicada no apoio à engenharia de software. São apresentadas tecnologias utilizadas para apoiar as atividades envolvidas na gerência de conhecimento, dando destaque às tecnologias utilizadas neste trabalho. O capítulo apresenta, ainda, a infra-estrutura de apoio à gerência de conhecimento de ODE.

- Capítulo 4 – *Infra-Estrutura para Caracterização de Itens de Software em ODE*: apresenta a infra-estrutura para caracterização de itens de software desenvolvida e integrada ao ambiente ODE. É apresentada a fundamentação teórica que embasa a proposta e são listadas as funcionalidades providas pela infra-estrutura. Apresenta-se, ainda, a ferramenta de caracterização de projetos de ODE desenvolvida utilizando a infra-estrutura.
- Capítulo 5 – *Infra-Estrutura de Apoio à Decisão em Grupo em ODE*: apresenta a infra-estrutura de apoio à decisão em grupo desenvolvida e integrada ao ambiente ODE. A fundamentação teórica utilizada como base da proposta é apresentada, são listadas as funcionalidades providas pela infra-estrutura e é apresentada a instanciação da infra-estrutura para apoiar a elaboração cooperativa de planos de risco em ODE.
- Capítulo 6 – *Considerações Finais*: apresenta as conclusões sobre o trabalho desenvolvido, evidenciando suas contribuições e perspectivas de trabalhos futuros.

Capítulo 2

Gerência de Projetos e Ambientes de Desenvolvimento de Software

O gerenciamento de projetos é a aplicação de conhecimento, habilidades, ferramentas e técnicas às atividades do projeto, a fim de atender aos seus requisitos (PMI, 2004).

A gerência de projetos de software é bastante complexa, envolvendo diversas atividades, sub-atividades e um grande número de recursos. Gerenciar esses recursos de forma eficiente torna-se uma tarefa quase impossível de ser realizada se o processo de desenvolvimento de software não estiver pelo menos parcialmente automatizado (VASCONCELOS et al., 1995).

De fato, a Engenharia de Software como um todo envolve tarefas complexas. Ferramentas que reduzem a quantidade de esforço requerido para produzir um artefato ou alcançar algum marco de projeto podem trazer benefícios substanciais. Ferramentas podem prover novos modos de olhar para a informação em engenharia de software – modos que melhoram a forma do engenheiro fazer o trabalho. Isto conduz a decisões melhores e a qualidade de software mais alta, permitindo assegurar que a qualidade seja projetada antes do produto ser construído (PRESSMAN, 2002).

Com o objetivo de aumentar a qualidade e a produtividade em projetos de software, foram criados os Ambientes de Desenvolvimento de Software (ADSs). Esses ambientes automatizam várias tarefas do processo de desenvolvimento de software, integrando coleções de ferramentas e tornam mais fácil seu controle.

Este capítulo procura dar uma visão geral dos conceitos relativos à gerência de projetos de software e ambientes de desenvolvimento de software. A seção 2.1 apresenta as principais atividades e conceitos que permeiam a gerência de projetos. A seção 2.2 aborda a forma como

algumas normas e modelos de qualidade tratam a gerência de projetos. Na seção 2.3 são apresentados os conceitos referentes a Ambientes de Desenvolvimento de Software (ADSs). A seção 2.4 fala sobre o apoio automatizado à gerência de projetos, apresentando algumas ferramentas que se propõem a apoiar a gerência de projetos. A seção 2.5 apresenta o ADS ODE (FALBO et al., 2004a), ambiente no qual este trabalho está inserido. Na seção 2.6 são apresentadas as considerações finais deste capítulo.

2.1 Gerência de Projetos de Software

A necessidade de gerenciamento é uma importante distinção entre o desenvolvimento profissional de software e a programação em nível amador. A gerência de projetos de software é necessária, porque a engenharia de software profissional está sempre sujeita a restrições de orçamento e prazo (SOMMERVILLE, 2003).

Segundo PRESSMAN (2002), a gerência de projetos de software envolve planejamento, acompanhamento e controle de pessoas, processos e eventos que ocorrem à medida que o software evolui de um conceito inicial até uma implementação operacional. O objetivo do planejamento do projeto é fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custos e prazos. Durante o acompanhamento do projeto, o planejamento deve ser detalhado e atualizado regularmente.

Os gerentes de projeto são responsáveis por planejar e programar o desenvolvimento de software. Eles supervisionam o trabalho para assegurar que o projeto seja realizado em conformidade com os padrões requeridos e monitoram o progresso para verificar se o desenvolvimento está dentro do prazo e do orçamento. O bom gerenciamento não pode garantir o sucesso do projeto. Contudo, o mau gerenciamento, geralmente, resulta em seu fracasso (SOMMERVILLE, 2003).

Os gerentes de projetos freqüentemente falam de uma “restrição tripla” - escopo, tempo e custo do projeto - no gerenciamento de necessidades conflitantes do projeto. A qualidade do projeto é afetada pelo balanceamento desses três fatores. Projetos de alta qualidade entregam o produto, serviço ou resultado solicitado dentro do escopo, no prazo e dentro do orçamento. A relação entre esses fatores ocorre de tal forma que, se algum dos três fatores mudar, pelo menos

um outro fator provavelmente será afetado. Os gerentes de projetos também gerenciam projetos em resposta a incertezas. Um risco do projeto é um evento ou condição incerta que, se ocorrer, terá um efeito negativo em pelo menos um dos objetivos do projeto (PMI, 2004).

O desenvolvimento de software não é apenas uma questão de criar ferramentas e linguagens de programação, mas também é um esforço coletivo, complexo e criativo. Como tal, a qualidade de um produto de software depende fortemente das pessoas, da organização e dos procedimentos usados para criá-lo e disponibilizá-lo (FUGGETTA, 2000).

2.1.1 Definição do Processo e do Escopo do Projeto

Acredita-se que a chave para a qualidade do software seja a qualidade do processo de software usado para construí-lo. Assim, muitos dos problemas de desenvolvimento podem ser resolvidos aperfeiçoando o processo. (FUGGETTA, 2000).

FUGGETTA (2000) define um processo de software como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para conceber, desenvolver, disponibilizar e manter um produto de software.

Segundo KRUCHTEN (1999), um processo de software possui quatro papéis, a saber:

- Guiar a ordem de atividades da equipe;
- Especificar quais artefatos devem ser produzidos e quando;
- Dirigir as tarefas de desenvolvedores individuais e da equipe como um todo;
- Oferecer critérios de monitoração e medição dos produtos e atividades do projeto.

Para FUGGETTA (2000), o processo de software deve identificar:

- Atividades que devem ser realizadas para atingir os objetivos do processo.
- Papéis das pessoas no processo.
- Estrutura e natureza dos artefatos que devem ser criados e mantidos.
- Ferramentas a serem utilizadas.

Sem um processo bem definido, a equipe de desenvolvimento trabalha de maneira *ad hoc* e o sucesso depende dos esforços heróicos de indivíduos dedicados. Esta situação não é sustentável. Em contraste, organizações maduras, empregando um processo bem definido, podem desenvolver sistemas complexos de maneira consistente e previsível, independente de quem os produza (KRUCHTEN, 1999). Neste contexto, é clara a necessidade de se definir um processo de software para o projeto e seguir o processo definido durante a execução do projeto.

Um produto de software é útil apenas se ele executa uma função desejada ou provê um serviço necessário. Nesse contexto, antes que um projeto possa ser planejado, os objetivos e o escopo devem ser estabelecidos, soluções alternativas devem ser consideradas e as restrições administrativas e técnicas, identificadas. Sem essas informações, é impossível definir estimativas de custos razoáveis (e precisas), uma divisão realística das tarefas de projeto ou uma programação de projeto administrável que ofereça indícios significativos de progresso (PRESSMAN, 2002).

Assim, a primeira atividade a ser realizada no planejamento de um projeto é a determinação do escopo do software, descrevendo sua função, seu desempenho, suas restrições, suas interfaces e sua confiabilidade.

2.1.2 Realização de Estimativas

Estabelecido o escopo, devem-se estimar os recursos necessários à realização do esforço de desenvolvimento do software. Os gerentes de projeto têm de procurar responder algumas questões: quanto esforço é necessário para completar uma atividade? Quais recursos serão utilizados? Quantos dias são necessários para completar uma atividade? E qual o custo total do projeto? As respostas a essas perguntas são altamente inter-relacionadas. Assim, uma alteração na resposta de uma delas pode afetar as respostas às demais. Por exemplo, uma alteração nos recursos alocados pode alterar o tempo necessário para a execução do projeto e o custo associado ao mesmo. Da mesma forma, uma restrição de custos ou tempo pode direcionar a uma alteração na alocação de recursos.

Segundo PRESSMAN (2002), os recursos utilizados em um projeto de software podem ser divididos em dois grandes grupos, a saber: ferramentas e recursos humanos. Cada recurso é

especificado segundo quatro características: uma descrição, uma declaração de disponibilidade, tempo que o recurso será requerido, ou seja, data de início e fim da sua utilização, e por quanto tempo ele será aplicado.

A estimativa das ferramentas necessárias envolve a definição do hardware e das ferramentas de software requeridas para a realização do projeto, desde seu desenvolvimento até seu funcionamento em produção.

Para estimar o pessoal necessário para o projeto, é necessário saber que tarefas serão realizadas e que habilidades e experiência serão exigidas para que o trabalho seja realizado de maneira eficiente. A designação de pessoas para as tarefas depende do tamanho do projeto e das habilidades e experiência do pessoal. Há uma grande vantagem em atribuir diferentes responsabilidades para diferentes grupos de pessoas, oferecendo ‘verificações e balanços’ que podem identificar, antecipadamente, problemas no processo de desenvolvimento (PFLEEGER, 2001).

Em algumas situações, o número de pessoas envolvidas no projeto de software só pode ser determinado depois da realização de uma estimativa do esforço necessário à realização do projeto.

Usualmente, estimativas de tamanho de software são utilizadas como base para a realização de estimativas de esforço. A estimativa de esforço, por sua vez, é, normalmente, a base para as estimativas de custo e tempo. Segundo PFLEEGER (2001), para a maioria dos projetos, o maior componente do custo é o esforço. O esforço é, ainda, o componente do custo com maior grau de incerteza. O estilo de trabalho, a organização do projeto, a capacidade, o interesse, o treinamento e outras características da equipe e do projeto podem afetar o tempo exigido para completar uma tarefa.

As estimativas de custo e esforço dificilmente virão a ser uma ciência exata. Um demasiado número de variáveis – humanas, técnicas, ambientais e políticas – pode afetar o custo final do software e o esforço aplicado para desenvolvê-lo. Todavia, a estimativa de projetos de software pode ser transformada de algo sobrenatural numa série de passos sistemáticos que forneçam estimativas com risco aceitável. (PRESSMAN, 2002).

Visando a elaborar estratégias para realização de boas estimativas, nas últimas duas décadas, muitos estudos têm sido realizados na área de estimativas de projetos de software. As

abordagens existentes para a realização de estimativas de projeto de software são: modelos paramétricos, analogia de estimativas e julgamento de especialistas.

A abordagem mais sistemática, embora não necessariamente a mais precisa para a estimativa de software, é a utilização de modelos paramétricos. Um modelo paramétrico pode ser construído analisando-se os custos e os atributos de projetos concluídos. Uma fórmula matemática é utilizada para prever os custos baseados em estimativas sobre o tamanho de projeto, o número de programadores e outros fatores relacionados ao processo e ao produto (SOMMERVILLE, 2003).

A utilização dos modelos paramétricos para a realização de estimativas apresenta a vantagem de ser uma abordagem objetiva e passível de repetição. Porém, esses modelos não eliminam a subjetividade existente na realização de estimativas. A estimativa dos fatores envolvidos nas fórmulas matemáticas utilizadas ainda está sujeita a subjetividade. Assim, a realização de boas estimativas utilizando modelos paramétricos é muito condicionada a um bom processo de calibração dos fatores envolvidos em suas fórmulas de cálculo.

Alguns desses modelos, já considerados clássicos como o COCOMO II (BOEHM et al., 2000), a Análise de Pontos de Função (GARMUS e HERRON, 2001) e a Análise de Pontos de Casos de Uso (KARNER, 1993) têm sua aplicabilidade constatada em experiências e práticas de muitas organizações, registradas em artigos e outras publicações.

As estimativas baseadas em analogias são métodos não paramétricos que utilizam dados históricos de outros projetos para realizar as estimativas para o projeto corrente. As analogias são realizadas levando-se em consideração características comuns aos projetos. São frequentemente utilizadas nas estimativas de custos totais do projeto quando existe uma quantidade limitada de informações detalhadas sobre ele, como, por exemplo, nas fases iniciais. Também são utilizadas para apoiar a distribuição do tempo e custos totais do projeto em suas fases, módulos e/ou atividades (PMI, 2004). A utilização dessa abordagem é dependente da existência de uma base de dados históricos de projetos anteriores.

Muitos estudos têm sido desenvolvidos para analisar a utilização de analogias em estimativas, dentre eles (MENDES e COUNSELL, 2000) e (SHEPPERD e SCHOFIELD, 1997), e os resultados têm sido consideravelmente positivos. Esses resultados tornam evidente a

necessidade de armazenar, em uma base de dados históricos, as informações a respeito de projetos já concluídos.

SHEPPERD e SCHOFIELD (1997) realizaram comparações em 299 projetos de 17 organizações através das quais foi possível constatar que os modelos baseados em algoritmos inteligentes e analogias superavam os paramétricos, sendo o COCOMO e a Análise de Pontos de Função os que apresentavam melhores resultados dentre os modelos paramétricos.

Muitos métodos para estimativas de esforço se baseiam no julgamento do especialista. Alguns são técnicas informais, que têm como base a experiência de um gerente com projetos semelhantes. Assim, a precisão da estimativa é baseada na competência, experiência, objetividade e percepção de quem faz a estimativa. Em sua maneira mais simples, tal estimativa é um palpite sobre o esforço necessário para construir um sistema ou seus subsistemas (PFLEEGER, 2001).

Um especialista é capaz de realizar estimativas que representem desvios aceitáveis quando comparadas aos valores realmente praticados, se ele tiver repetido a atividade de estimar muitas vezes e, com isso, tiver alcançado para si um certo grau de maturidade para realizar as estimativas de um projeto (BOEHM, 2000).

JØRGENSEN (2002) faz um apanhado geral de estudos sobre a estimativa de esforço baseada no julgamento de especialistas em gerência de projetos. Nesse trabalho são apresentados os resultados de várias pesquisas na área e um conjunto de técnicas que podem auxiliar a geração de boas estimativas.

Há projetos de software muito complexos, para os quais é muito difícil realizar estimativas pensando no projeto inteiro. Além disso, a elaboração de uma estimativa para o projeto como um todo dificulta o acompanhamento do andamento durante o processo de desenvolvimento do projeto. Independente da abordagem utilizada para a realização das estimativas, uma boa forma de resolver este problema é usar técnicas de decomposição do problema, do processo ou ambas, de forma a tentar facilitar a caracterização do software.

Usando técnicas de decomposição, o gerente de projeto faz uso de informações de métricas para determinar o tamanho ou o esforço para se desenvolver partes do projeto e, assim, obter a estimativa do projeto como um todo.

Considerando as pesquisas e práticas desenvolvidas na área de estimativas de projetos de software, ARMOUR (2002) realiza considerações sobre algumas questões que são consideradas mitos no processo de realização de estimativas em projetos de software. Algumas conclusões apresentadas por ARMOUR (2002) são: (i) é muito difícil produzir estimativas acuradas, pois os dados necessários para a realização destas só estarão disponíveis nas fases conclusivas do projeto; (ii) o tamanho do projeto não é capaz de, sozinho, determinar as estimativas; (iii) a utilização de dados históricos é útil, mas também não é capaz de garantir a acurácia das estimativas; (iv) um coeficiente médio de produtividade não indica a produtividade real de cada indivíduo; (v) linhas de código, pontos de função e outros sistemas de medida podem não refletir o tamanho real do projeto; (vi) aumentar o número de pessoas em um projeto não significa que o mesmo poderá ser concluído em menos tempo.

Há algumas práticas que podem ser empregadas para minimizar esses problemas, como por exemplo: (i) acompanhar o projeto e refazer as estimativas à medida que novas informações são obtidas, (ii) utilizar dados históricos na geração das estimativas, (iii) usar pelo menos dois métodos para realizar as estimativas, (iv) combinar estimativas de diferentes especialistas, (v) ponderar complexidade e riscos, (vi) conhecer bem as características da equipe de desenvolvimento (PRESSMAN, 2002) (PFLEEGER, 2001) (JØRGESEN, 2002).

JØRGESEN (2002) afirma que as estimativas são influenciadas por ruídos situacionais e humanos. Por exemplo, o desejo de conseguir fechar um contrato pode fazer com que o gerente de projeto subestime o custo, gerando, assim, um “ruído situacional”. Há outros ruídos que são gerados por características inerentes ao ser humano. Por exemplo, a necessidade que nós, seres humanos, temos de sermos bem vistos pelas pessoas que consideramos importantes pode levar o gerente de projeto a estimar que o projeto será concluído no tempo que ele sabe que o usuário quer e não no tempo que ele realmente considera necessário. Assim, Jørgesen sugere seis princípios que visam reduzir essas influências situacionais e humanas sobre as estimativas:

1. Solicitar que os gerentes de projeto justifiquem e critiquem suas estimativas;
2. Evitar informações irrelevantes e irreais, pois estudos mostram que as estimativas podem ser muito influenciadas por informações irrelevantes, mesmo quando os gerentes de projeto sabem que elas são irrelevantes;
3. Usar dados documentados de projetos anteriores;

4. Encontrar especialistas que conheçam bem a tarefa a ser estimada (que tenham experiências anteriores em situações similares à atual) e que tenham bons registros de estimativas anteriores;
5. Avaliar a acurácia das estimativas sem pressionar os gerentes de projeto, pois a pressão de estar sendo avaliado pode diminuir a qualidade das estimativas ;
6. Evitar situações em que o processo de estimativas é impactado por outros objetivos que não a acurácia da estimativa. Objetivos como a maximização dos lucros podem reduzir a exatidão das estimativas.

JØRGESEN (2002) afirma que um dos princípios para a geração de boas estimativas é obter e combinar estimativas de diferentes especialistas utilizando diferentes abordagens. Assim, a técnica de Delphi (PFLEEGER, 2001) aparece como uma boa alternativa para a realização de estimativas.

Na técnica de Delphi, pede-se que especialistas façam suas estimativas individuais secretamente, com base em suas experiências e recorrendo a quaisquer métodos que queiram. Em seguida, a estimativa média é calculada e apresentada ao grupo. Cada especialista pode rever sua estimativa, caso deseje. O processo é repetido até que nenhum dos especialistas deseje rever sua estimativa. Alguns usuários da técnica de Delphi discutem a média antes que novas estimativas sejam feitas, enquanto outras vezes nenhuma discussão é permitida. Em uma outra variação dessa técnica, as justificativas de cada especialista para suas estimativas são distribuídas anonimamente a todos os especialistas (PFLEEGER, 2001).

2.1.3 Elaboração e Acompanhamento de Cronograma

Uma vez estimado o projeto, pode-se passar à elaboração do cronograma. Um cronograma é uma linha do tempo que mostra quando as atividades começarão e terminarão e quando os artefatos relacionados estarão prontos. Em um cronograma, geralmente, são apontados marcos do projeto, nos quais deverão ocorrer atividades de avaliação e acompanhamento do projeto. Enquanto uma atividade é realizada durante um certo período de tempo, um marco é um ponto particular no tempo, normalmente coincidindo com a conclusão de alguma atividade (PFLEEGER, 2001).

A elaboração do cronograma do projeto de software visa a distribuir o esforço estimado pela duração planejada do projeto, partilhando esse esforço por tarefas específicas da engenharia de software. O objetivo do gerente de projeto é definir todas as tarefas do projeto, construir uma rede que mostre suas interdependências, identificar as tarefas que são críticas nessa rede e depois acompanhar seu progresso para certificar-se de que os atrasos sejam reconhecidos tão logo aconteçam. Para conseguir isso, o gerente deve dispor de um cronograma que tenha sido definido num grau de resolução que permita monitorar o progresso e controlar o projeto (PRESSMAN, 2002).

É importante notar, contudo, que o cronograma evolui ao longo do tempo. Nos estágios iniciais de planejamento, um cronograma macroscópico é desenvolvido, identificando as principais atividades de engenharia de software e a quais funções do produto elas se aplicam. À medida que o projeto progride, cada entrada do cronograma macroscópico é refinada em um cronograma detalhado. Neste, tarefas específicas, requeridas para realizar uma atividade, são identificadas e escalonadas.

A elaboração de um cronograma requer a execução de um conjunto de sub-atividades, a saber: (i) dividir o projeto em um certo número de tarefas gerenciáveis, identificando as interdependências entre elas, (ii) atribuir a cada tarefa um número de unidades de trabalho e datas de início e término, (iii) definir o resultado de cada tarefa, (iv) identificar os caminhos críticos do projeto e (v) associar a cada tarefa ou grupo de tarefas um marco de referência do projeto.

Os gerentes devem, ainda, alocar recursos para a execução de cada uma das tarefas. Os recursos podem ser recursos de hardware, ferramentas de apoio, orçamentos para viagem, dentre muitos outros. Mas, o principal recurso tende a ser o esforço humano requerido.

Ao alocar pessoas a tarefas, o gerente deve tentar casar as características individuais de cada pessoa às características requeridas para a execução das tarefas. Algumas características que devem ser avaliadas são capacidade de comunicação, habilidades de gerenciamento, experiências com a tecnologia a ser utilizada, treinamento e interesse em realizar o trabalho. Cabe ao gerente de projeto alocar as pessoas a tarefas em que possam utilizar suas melhores características (PFLEEGER, 2001).

Tão importante quanto elaborar um bom cronograma é acompanhar e controlar sua execução. O controle é empregado por um gerente de projetos de software para administrar os

recursos do projeto, enfrentar problemas e dirigir o pessoal do projeto. Se as coisas estiverem indo bem, o controle é leve. Mas, quando ocorrerem problemas, o gerente de projetos deverá exercer o controle para saná-los o mais rapidamente possível. Depois que o problema tiver sido diagnosticado, recursos adicionais podem ser concentrados na área de problema, o pessoal pode ser novamente disposto ou a programação do projeto, redefinida (PRESSMAN, 2002).

Existem diversos métodos que podem ser utilizados para determinar e avaliar cronogramas, dentre eles o PERT (*Program Evaluation and Review Technique*) e o CPM (*Critical Path Method*) (PRESSMAN, 2002).

2.1.4 Gerência de Riscos

Um cronograma elaborado imaginando uma situação ideal, onde tudo dará certo, dificilmente será cumprido. Todo projeto está sujeito a eventualidades que podem afetar o andamento e os custos do projeto. PFLEEGER (2001) afirma que os gerentes devem determinar a possibilidade de ocorrência de eventos indesejáveis durante o desenvolvimento ou a manutenção e fazer planos para evitar esses eventos ou, caso eles sejam inevitáveis, procurar minimizar as consequências negativas. Ou seja, é preciso gerenciar riscos.

A Gerência de Riscos visa a identificar potenciais problemas antes que eles ocorram, de forma que ações possam ser tomadas a fim de reduzir ou eliminar a probabilidade e impacto destes problemas (IEEE Std 1540-2001, 2001).

De modo simplificado, pode-se pensar no risco como uma probabilidade de que alguma circunstância adversa realmente venha a ocorrer. Os riscos podem ameaçar o projeto, o software que está sendo desenvolvido ou a organização. Os riscos podem surgir como decorrência de requisitos mal definidos, de dificuldades em estimar o prazo e os recursos necessários para o desenvolvimento do software, da dependência de habilidades individuais e de mudanças nos requisitos, em razão de modificações nas necessidades do cliente, dentre outros. O gerente de projeto deve prever riscos, compreender o impacto desses riscos no projeto, no produto e nos negócios e tomar providências para evitar esses riscos. Planos de contingência podem ser traçados para que, se os riscos realmente ocorrerem, seja possível uma ação imediata que vise à recuperação (SOMMERVILLE, 2003).

Os riscos sempre envolvem duas características: *incerteza* - o evento que caracteriza um risco pode ou não acontecer, e *perda* - se um risco se tornar realidade, conseqüências indesejáveis ou perdas irão ocorrer. Quando riscos são analisados, é importante quantificar o nível de incerteza e o grau de perdas associadas a cada risco. É importante realçar que as atividades da gerência de riscos são aplicadas iterativamente durante todo o acompanhamento do projeto de software (PRESSMAN, 2002).

O gerenciamento de riscos inclui as seguintes atividades: (i) identificação dos riscos do projeto, (ii) análise dos riscos, atribuindo a cada risco sua probabilidade e impacto, (iii) avaliação de riscos, o que envolve o estabelecimento de prioridades e de um ponto de corte, indicando quais riscos serão gerenciados e quais não serão, (iv) planejamento de ações, que se refere à identificação de ações a serem tomadas para evitar (ações de mitigação) que um risco ocorra ou para definir o que fazer quando um risco se tornar realidade (ações de contingência), (v) elaboração de um plano de riscos e (vi) monitoramento dos riscos.

2.1.5 Elaboração do Plano de Projeto

Ao término do planejamento, um plano de projeto de software deve ser produzido, fornecendo informações básicas sobre o custo e o escalonamento de recursos, que serão usadas ao longo do processo de software. O plano de projeto é um documento que se destina a um público diverso. Ele deve comunicar o escopo e os recursos à gerência, ao pessoal técnico e ao cliente; definir a organização da equipe e oferecer um quadro de alocação dos recursos ao longo do tempo; definir os riscos e sugerir técnicas para evitá-los ou gerenciá-los; definir custos e prazos; oferecer uma abordagem global ao desenvolvimento de software para todas as pessoas associadas ao projeto; e esboçar como a qualidade será buscada (um plano de controle e garantia da qualidade) e como as alterações serão gerenciadas (PRESSMAN, 2002).

Segundo PFLEEGER (2001), um bom plano de projeto inclui os seguintes itens: o escopo do projeto, as estimativas de custos e de esforço, a organização da equipe do projeto, a descrição técnica do sistema proposto, os padrões, os procedimentos, as técnicas e as ferramentas a serem utilizados no projeto, o cronograma do projeto, o plano de garantia da qualidade, o plano de gerência de configuração, o plano de documentação, o plano de gerência de dados, o plano de

gerência de recursos, o plano de testes, o plano de treinamento, o plano de segurança e o plano de gerência de riscos.

2.1.6 Acompanhamento e Encerramento do Projeto

O trabalho do gerente de projetos não se encerra ao finalizar a elaboração do Plano de Projetos. Tão importante quanto elaborar um bom Plano de Projeto é realizar um eficiente acompanhamento do projeto.

O gerente deve ficar atento a possíveis atrasos em relação ao cronograma inicial, a alterações no custo estimado, à ocorrência de riscos identificados e ao surgimento de novos riscos, ao desempenho do pessoal, enfim, ao andamento geral do projeto.

No início do projeto, pouco se sabe sobre o mesmo. Assim, é fundamental que os planos sejam revistos periodicamente, levando-se em conta os novos dados sobre o projeto, visando à identificação e solução de possíveis problemas em tempo hábil.

Durante o desenvolvimento de um projeto de software, muito conhecimento é construído. Infelizmente, grande parte desse conhecimento passa despercebido e nunca é compartilhado entre os membros da organização. Assim, é importante ter uma atividade de encerramento do projeto na qual as lições aprendidas no projeto sejam discutidas. Nesse contexto, a análise *postmortem* aparece como um excelente método para a gerência de conhecimento, permitindo capturar experiências e sugestões de melhoria a partir de projetos finalizados (BIRK et al., 2002).

Quando utilizada apropriadamente, a análise *postmortem* assegura que todos os membros da equipe de desenvolvimento reconheçam e recordem o que foi aprendido durante o projeto. Os indivíduos compartilham suas experiências com toda a equipe e repassam essas experiências às equipes de outros projetos. A análise *postmortem* permite, ainda, a identificação de oportunidades de melhoria e provê um meio para iniciar mudanças sustentadas (BIRK et al., 2002).

Durante a realização de uma análise *postmortem*, membros da equipe de melhoria de processo trabalham como facilitadores junto a membros da equipe de desenvolvimento. Os facilitadores organizam a análise, conduzem as discussões e documentam os resultados.

Segundo BIRK et al. (2002), o processo de análise *postmortem* compreende três fases, a saber:

- *Preparação*: durante essa fase, toda a história do projeto é estudada para melhor entender o que aconteceu, todos documentos disponíveis são revistos e é determinado um objetivo para a análise, como, por exemplo, identificar oportunidades de melhoria ou identificar as maiores realizações do projeto.
- *Coleta de Dados*: nessa fase são reunidas as experiências relevantes do projeto. É importante que sejam capturados tanto aspectos positivos quanto aspectos negativos do projeto.
- *Análise*: Nessa fase os facilitadores conduzem uma seção de *feedback* com os demais participantes da análise tendo como objetivo verificar se foi entendido tudo o que foi dito sobre o projeto e se todos os fatos relevantes foram capturados.

Para finalizar, os facilitadores documentam os resultados da análise *postmortem* gerando um relatório de experiências do projeto.

2.2 Gerência de Projetos e Normas e Modelos de Qualidade de Processo

Ultimamente, muito se tem falado sobre a adoção de modelos e normas de qualidade no processo de software. Entre os vários modelos, podem ser citados o CMMI (*Capability Maturity Model Integration*) (CHRISSIS et al., 2003) e o MPS.BR (SOFTEX, 2006) e, no caso das normas, um exemplo é a ISO/IEC¹ 12207 (ISO 1995) (ISO 2002) (ISO 2004).

O CMMI é uma evolução o CMM (*Capability Maturity Model*) (FIORINI et al., 1998) e tem como objetivo fornecer diretrizes para a definição e melhoria de processos de uma organização. Entre os vários modelos do CMMI pode-se destacar o CMMI-SE/SW (*Capability Maturity Model Integration for Systems Engineering/Software Engineering*), o qual é um dos componentes do conjunto de produtos do CMMI voltado para as áreas de engenharia de sistemas e engenharia de software.

¹ A sigla ISO origina-se de *International Organization for Standardization* e a sigla IEC origina-se de *International Electrotechnical Commission*.

O MPS.BR (Melhoria de Processo do Software Brasileiro) é um modelo brasileiro que está em desenvolvimento desde dezembro de 2003 e tem como objetivo auxiliar micros, pequenas e médias empresas a definirem e a melhorarem seus processos de maneira a se certificarem. É um modelo direcionado para a realidade brasileira. Um fato relevante é que na concepção do MPS.BR levaram-se em consideração as normas ISO/IEC 12207 e ISO/IEC 15504 (ISO, 2003), além do modelo CMMI. Dessa maneira, procura-se garantir que o MPS.BR está em conformidade com padrões internacionais (SOFTEX, 2006).

Tendo em vista a importância da gerência de projetos para a produção de software de qualidade, cada modelo ou norma de qualidade define processos e/ou diretrizes para que as organizações tenham um processo efetivo de gerência de projetos.

2.2.1 Gerência de Projetos no CMMI

As áreas de processos (*process areas* – PAs) de Gerenciamento de Projetos do CMMI cobrem as atividades de gerenciamento de projetos relacionadas ao planejamento, monitoramento e controle do projeto.

Para descrever as interações entre as PAs de Gerenciamento de Projetos do CMMI, é mais conveniente tratá-las em dois grupos de áreas de processos: as PAs básicas de Gerenciamento de Projetos e as PAs avançadas de Gerenciamento de Projetos.

As PAs básicas de Gerenciamento de Projetos tratam das atividades básicas relacionadas ao estabelecimento e manutenção do plano do projeto, estabelecimento e manutenção de compromissos, monitoramento do progresso em relação ao plano, tomada de ações corretivas e gerenciamento de acordos com fornecedores. Fazem parte deste grupo as seguintes PAs: (i) Planejamento do Projeto (*Project Planning* - PP), (ii) Monitoramento e Controle do Projeto (*Project Monitoring and Control* - PMC) e (iii) Gerenciamento de Acordos com Fornecedores (*Supplier Agreement Management* - SAM).

A Figura 2.1 oferece uma visão geral das interações das PAs básicas do Gerenciamento de Projetos e outras categorias de áreas de processos.

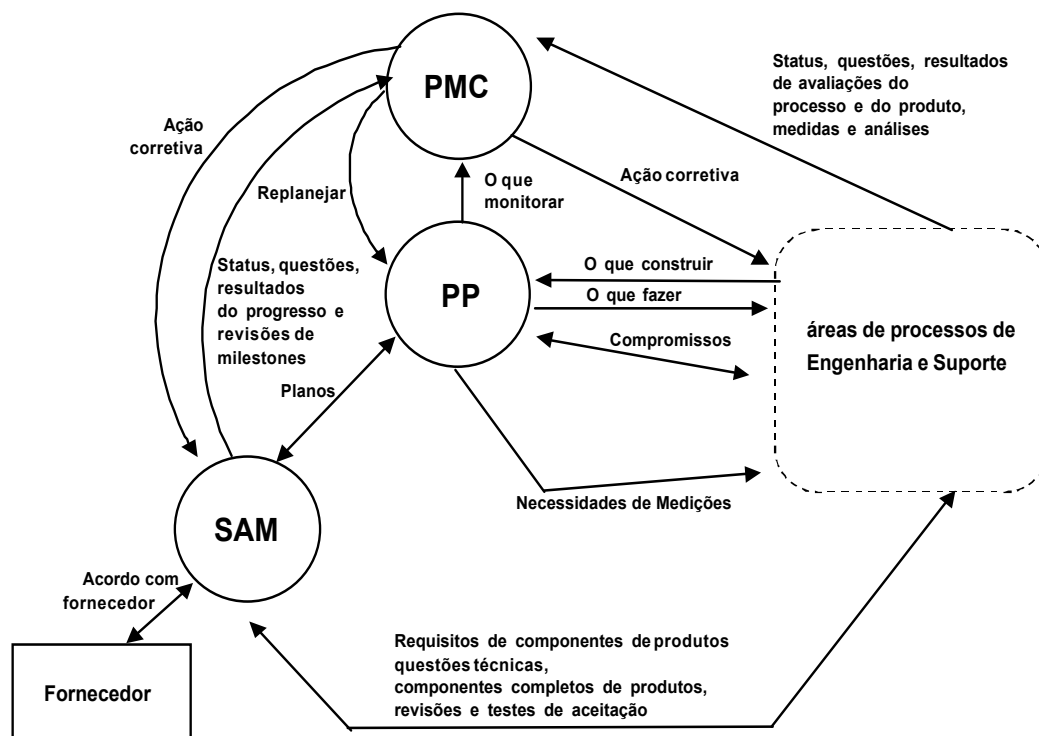


Fig. 2.1 - Áreas de Processos Básicas do Gerenciamento de Projetos do CMMI

As PAs avançadas de Gerenciamento de Projetos tratam de atividades como o estabelecimento de um processo definido que é adaptado a partir do conjunto de processos padrão da organização, a coordenação e colaboração com os envolvidos (*stakeholders*) relevantes, o gerenciamento de riscos, a formação e sustentação de equipes integradas para a condução de projetos e o gerenciamento quantitativo dos processos definidos do projeto. As seguintes PAs compõem este grupo: (i) Gerenciamento Integrado de Projetos (*Integrated Project Management – IPM*), (ii) Gerenciamento de Riscos (*Risk Management - RSKM*), (iii) Integração de Equipes (*Integrated Teaming - IT*), (iv) Gerenciamento Integrado de Fornecedores (*Integrated Supplier Management – ISM*) e (iv) Gerenciamento Quantitativo de Projetos (*Quantitative Project Management - QPM*).

A Figura 2.2 oferece uma visão geral das interações das principais PAs avançadas do Gerenciamento de Projetos e outras categorias de áreas de processos. Cada PA avançada de

Gerenciamento de Projetos é fortemente dependente da capacidade de se planejar, monitorar e controlar o projeto, capacidade essa fornecida pela PAs básicas de Gerenciamento de Projetos.

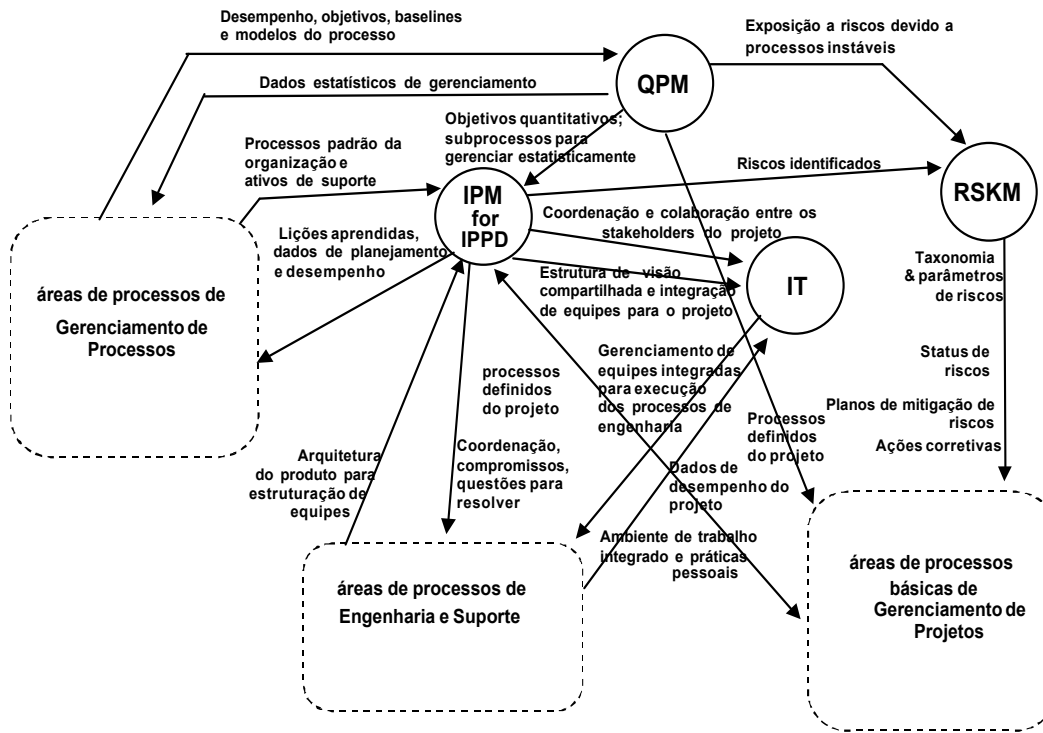


Fig. 2.2 - Áreas de Processos Avançadas de Gerenciamento de Projetos do CMMI

Nos modelos CMMI com representação em estágios, existem cinco níveis de maturidade, definidos pelos números de 1 a 5:

1. Inicial
2. Gerenciado
3. Definido
4. Gerenciado Quantitativamente
5. Otimizado

Todas as três PAs básicas do Gerenciamento de Projetos (Planejamento do Projeto, Monitoramento e Controle do Projeto e Gerenciamento de Acordos com Fornecedores) são

tratadas no nível 2 (Gerenciado). As PAs Gerenciamento Integrado de Projetos e Gerenciamento de Riscos são tratadas no nível 3 (Definido). O Gerenciamento Quantitativo de Projetos é uma PA do nível 4 (Gerenciado Quantitativamente). Por fim, as PAs Integração de Equipes e Gerência Integrada de Fornecedores são também tratadas no nível 3 (Definido). Contudo, vale destacar que elas não são PAs para o corpo de conhecimento do CMMI nas áreas de Engenharia de Sistemas e Engenharia de Software. A Integração de Equipes é uma PA para a área de Desenvolvimento Integrado de Produto e Processo (*Integrated Product and Process Development*), enquanto a PA Gerência Integrada de Fornecedores é uma PA para a área de Fornecimento (*Supplier Sourcing*) (CHRISSIS et al., 2003).

2.2.2 Gerência de Projetos no MPS.BR

O modelo de referência do MPS.BR define sete níveis de maturidade: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado). A escala de maturidade se inicia no nível G e progride até o nível A. Para cada um desses sete níveis de maturidade é atribuído um perfil de processos que indica onde a organização deve colocar o esforço de melhoria. No âmbito da gerência de projetos, merecem destaque quatro desses processos, a saber:

- Processo *Gerência de Projetos* (nível G): visa a identificar, estabelecer, coordenar e monitorar as atividades, tarefas e recursos que um projeto necessita para produzir um produto e/ou serviço, no contexto dos requisitos e restrições do projeto.
- Processo *Adaptação do Processo para Gerência do Projeto* (nível E): tem como propósito estabelecer e gerenciar o projeto e envolver os interessados de acordo com o processo definido e integrado que é adaptado do conjunto de processos-padrão da organização.
- Processo *Gerência de Riscos* (nível C): seu propósito é identificar, gerenciar e reduzir continuamente os riscos em nível organizacional e de projeto.
- Processo *Gerência Quantitativa do Projeto* (nível B): tem como propósito gerenciar quantitativamente o processo definido para o projeto de forma a alcançar

os objetivos para qualidade e para o desempenho do processo estabelecidos para o projeto.

2.2.3 Gerência de Projetos na ISO/IEC 12207

A ISO/IEC 12207, mais especificamente sua emenda 1 (ISO, 2002), define o propósito do Processo de Gerência como sendo organizar, monitorar e controlar a iniciação e a execução de qualquer processo de forma a atingir as suas metas de acordo com as metas de negócio da organização.

Esse processo é composto por seis sub-processos, com destaque para os seguintes: (i) Alinhamento Organizacional, cujo propósito é fazer com que o processo de software da organização seja consistente com as suas metas de negócio, (ii) Gerência da Organização, que visa a estabelecer e executar práticas de gerência de software, durante a execução dos processos, que sejam consistentes com as metas de negócio da organização, (iii) Gerência de Projetos, cujo propósito é identificar, estabelecer, coordenar e monitorar as atividades, tarefas e recursos de que um projeto necessita para produzir um produto e/ou serviço, no contexto dos requisitos e restrições do projeto, e (iv) Gerência de Riscos, que tem como propósito identificar, gerenciar e reduzir continuamente os riscos nos níveis organizacional e de projeto.

2.2.4 Gerência de Projetos e o PMBOK

Pode-se constatar que, devido à importância das atividades de gerência de projetos para que um projeto de software seja desenvolvido com sucesso, essas atividades são abordadas pelos principais modelos e normas de qualidade de processo de software hoje vigentes. De fato, essa importância é tamanha, que foi criado um órgão internacional que visa a promover o profissionalismo e desenvolver o estado da arte na gestão de projetos: o PMI (*Project Management Institute*) (MARTINS, 2005). O PMI é responsável pela elaboração de um corpo de conhecimento sobre gerenciamento de projetos, o PMBOK (*Project Management Body of Knowledge*), que tem como objetivo principal identificar e agrupar os conhecimentos sobre a profissão, além de padronizar o vocabulário da mesma.

O principal objetivo do Guia PMBOK é identificar um subconjunto do conjunto de conhecimentos em gerenciamento de projetos que é amplamente reconhecido como sendo boas práticas. “Identificar” significa fornecer uma visão geral, e não uma descrição completa. “Amplamente reconhecido” significa que o conhecimento e as práticas descritas são aplicáveis à maioria dos projetos, na maior parte do tempo, e que existe um consenso geral em relação ao seu valor e sua utilidade. “Boa prática” significa que existe acordo geral de que a aplicação correta dessas habilidades, ferramentas e técnicas podem aumentar as chances de sucesso em uma ampla série de projetos diferentes. Uma boa prática não significa que o conhecimento descrito deverá ser sempre aplicado uniformemente em todos os projetos. A equipe de gerenciamento de projetos é responsável por determinar o que é adequado para um projeto específico (PMI, 2004).

O PMBOK define quarenta e quatro processos de gerenciamento de projetos. Esses processos são organizados em nove áreas de conhecimento em gerência de projetos, a saber (PMI, 2004):

- Gerenciamento de Integração do Projeto: descreve os processos e as atividades que integram os diversos elementos do gerenciamento de projetos, que são identificados, definidos, combinados, unificados e coordenados dentro dos grupos de processos de gerenciamento de projetos. Essa área consiste nos seguintes processos: Desenvolvimento do termo de abertura do projeto, Desenvolvimento da declaração do escopo preliminar do projeto, Desenvolvimento do plano de gerenciamento do projeto, Orientação e gerenciamento da execução do projeto, Monitoramento e controle do trabalho do projeto, Controle integrado de mudanças e Encerramento do projeto.
- Gerenciamento do Escopo do Projeto: descreve os processos envolvidos na verificação de que o projeto inclui todo o trabalho necessário, e apenas o trabalho necessário, para que seja concluído com sucesso. Essa área engloba os seguintes processos: Planejamento do escopo, Definição do escopo, Criação da Estrutura Analítica do Projeto, Verificação do escopo e Controle do escopo.
- Gerenciamento de tempo do projeto: descreve os processos que visam a garantir o término do projeto no prazo correto. Consiste nos seguintes processos: Definição das atividades, Sequenciamento de atividades, Estimativa de recursos da atividade,

Estimativa de duração da atividade, Desenvolvimento do cronograma e Controle do cronograma.

- Gerenciamento de custos do projeto: descreve os processos envolvidos em planejamento, estimativa, orçamento e controle de custos, de modo que o projeto termine dentro do orçamento aprovado. Os seguintes processos compõem essa área: Estimativa de custos, Elaboração de Orçamento e Controle de custos.
- Gerenciamento da qualidade do projeto: descreve os processos envolvidos na garantia de que o projeto irá satisfazer os objetivos para os quais foi realizado. Consiste nos seguintes processos: Planejamento da qualidade, Realização da garantia da qualidade e Realização do controle da qualidade.
- Gerenciamento de recursos humanos do projeto: descreve os processos que organizam e gerenciam a equipe do projeto. Essa área define os seguintes processos: Planejamento de recursos humanos, Contratação ou mobilização da equipe do projeto, Desenvolvimento da equipe do projeto e Gerenciamento da equipe do projeto.
- Gerenciamento das comunicações do projeto: descreve os processos relativos à geração, coleta, disseminação, armazenamento e destinação final das informações do projeto de forma oportuna e adequada. Os seguintes processos são definidos nessa área: Planejamento das comunicações, Distribuição das informações, Relatório de desempenho e Gerenciamento das partes interessadas.
- Gerenciamento de riscos do projeto: descreve os processos relativos à realização do gerenciamento de riscos em um projeto. Consiste nos seguintes processos: Planejamento do gerenciamento de riscos, Identificação de riscos, Análise qualitativa de riscos, Análise quantitativa de riscos, Planejamento de respostas a riscos e Monitoramento e controle de riscos.
- Gerenciamento de aquisições do projeto: descreve os processos definidos para guiar a compra ou aquisição de produtos, serviços ou resultados, além dos processos de gerenciamento de contratos. Os seguintes processos compõem essa área: Planejamento de compras e aquisições, Planejamento de contratações,

Solicitação de respostas de fornecedores, Seleção de fornecedores, Administração de contrato e Encerramento do contrato.

2.3 Automatização do Processo de Software

Segundo PRESSMAN (2002), um bom ambiente de trabalho para qualquer pessoa, inclusive para um engenheiro de software, deve ter, pelo menos: as ferramentas para execução do trabalho, um ambiente organizado para utilização das ferramentas de forma rápida e eficaz e um profissional que entenda como usar as ferramentas da melhor forma.

Durante muito tempo, os engenheiros de software construíram sistemas e produtos complexos para automatizar o trabalho de outros profissionais, porém, eles próprios não dispunham de um ferramental que apoiasse seus trabalhos. Percebendo a necessidade de automatização de suas tarefas, passaram, então, a dedicar mais tempo na elaboração de ferramentas a serem usadas no desenvolvimento de sistemas. Surgiram, assim, as ferramentas CASE (*Computer Aided Software Engineering*).

Vale ressaltar que as ferramentas CASE são utilizadas para complementar as práticas de Engenharia de Software e não para substituí-las. Antes que as ferramentas possam efetivamente ser usadas, deve ser estabelecido um processo de software, conceitos e métodos de engenharia de software devem ser aprendidos e a qualidade deve ser sempre enfatizada (PRESSMAN, 2002).

As primeiras ferramentas CASE construídas funcionavam isoladamente apoiando fases específicas do desenvolvimento de software, não garantindo a consistência e a qualidade do produto ao longo do processo de desenvolvimento.

Após ter-se verificado que ferramentas isoladas podem oferecer apenas soluções parciais, o que se deseja é utilizar ferramentas de apoio ao longo de todo o processo de desenvolvimento de software (TRAVASSOS, 1994). Neste contexto, surgiram os Ambientes de Desenvolvimento de Software (ADSs), que são sistemas computacionais que provêm suporte para o desenvolvimento e a manutenção de produtos de software, e para o gerenciamento dessas atividades (VILLELA et al., 2002).

Os ADSs podem ser descritos como coleções de ferramentas integradas que facilitam as atividades da engenharia de software, durante todo o ciclo de vida do software ou pelo menos em porções significativas dele. ADSs têm o objetivo de interferir positivamente no tempo de desenvolvimento, no custo e na qualidade de um projeto (HARRISON et al., 2000).

Em comparação com as ferramentas CASE isoladas, os ADSs permitem melhor controle do processo, evitando inconsistências e, conseqüentemente, propiciam ganhos de qualidade e produtividade (FALBO, 2000).

Um ADS deve se preocupar com o apoio às atividades individuais e ao trabalho em grupo, com o gerenciamento de projeto, o aumento da qualidade geral dos produtos e o aumento da produtividade, permitindo ao desenvolvedor acompanhar o projeto e medir a sua evolução (TRAVASSOS, 1994).

A integração de ferramentas em um ADS possibilita a transferência constante de informação de uma ferramenta para outra, o aumento do controle do projeto, a redução do esforço necessário para garantir a qualidade e a consistência das informações, dentre outros benefícios.

Para que o desenvolvimento de software ocorra com qualidade, é necessário que ele seja desenvolvido de acordo com um conjunto de atividades bem definido e ordenado, que juntamente com os recursos e artefatos utilizados e produzidos, constituem um processo de software. Mas processos de software são, muitas vezes, entidades complexas e incluem: atividades, papéis das pessoas no processo, estrutura e natureza dos artefatos e ferramentas utilizadas (FUGGETTA, 2000).

Desta forma, houve necessidade de evoluir ADSs para que pudessem apoiar a definição e execução de processos de software, surgindo, então, os Ambientes de Desenvolvimento de Software Centrados em Processo (ADSCPs).

ADSCPs integram ferramentas para apoiar o desenvolvimento do produto de software e para apoiar a modelagem e a execução do processo de software que desenvolve este produto (HARRISON et al., 2000). Para FUGGETTA (2000), ADSCPs são ambientes que apóiam a criação e a exploração de modelos de processo.

ADSCPs exploram a definição do processo de software, que especifica as atividades, funções e tarefas dos desenvolvedores de software, e definem como controlar ferramentas de desenvolvimento de software (AMBRIOLA et al., 1997).

Um dos desafios do desenvolvimento de software é a correta compreensão daquilo que o sistema necessita realizar. A não compreensão do problema pode levar ao desenvolvimento correto do produto errado, o que pode ser agravado pelo fato da solução estar dispersa entre o conhecimento de vários especialistas. Ou seja, uma das grandes dificuldades no desenvolvimento de software é que, muitas vezes, os desenvolvedores não estão familiarizados com o domínio para o qual o software está sendo desenvolvido (NATALI, 2003).

Nesse contexto, constatou-se que ambientes de desenvolvimento de software poderiam apoiar melhor o desenvolvimento e a manutenção de um produto de software, se fossem capazes de fornecer conhecimento do domínio aos desenvolvedores (VILLELA et al., 2002). Surgiram, assim, os Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSODs).

Esses ambientes propõem um apoio ao entendimento do domínio para desenvolvedores que não têm familiaridade ou experiência em realizar trabalhos no domínio considerado. ADSODs são definidos tendo como base os tradicionais ADSs, mas incorporando um novo fator: o conhecimento de um domínio específico (OLIVEIRA, 1999).

Desenvolvedores de software lidam de forma intensa com diferentes tipos de conhecimento ao longo dos processos de software. O conhecimento do domínio da aplicação é uma parcela do conhecimento necessário. A outra parcela é constituída pelo conhecimento acumulado pela organização e relevante para o contexto específico. Conhecimento sobre diretrizes e melhores práticas organizacionais, técnicas e métodos de desenvolvimento de software, além de experiências anteriores com o uso dessas técnicas e métodos e com o processo de software são exemplos de conhecimento relevante neste contexto. No entanto, identificação, organização, armazenamento e uso de conhecimento não são tarefas triviais (NATALI, 2003).

Para atender a essa necessidade, cada vez mais a gerência de conhecimento tem se tornado presente em ADSs. Um primeiro passo foi utilizar algumas de suas técnicas para gerenciar o conhecimento de domínios de aplicação (ADSOD).

Entretanto, como o foco principal dos ambientes é apoiar o desenvolvimento de software, nada mais natural que gerenciar o conhecimento de seu próprio domínio, o de Engenharia de

Software. Assim, alguns ambientes passaram a incorporar conhecimento a respeito de processos, atividades, recursos, artefatos, métodos, técnicas, paradigmas, tecnologias, entre outros (RUY, 2006).

Segundo RUY (2006), três pesquisas importantes que focam em ADSs com Gerência de Conhecimento em Engenharia de Software são os Ambientes de Desenvolvimento Orientados à Organização (LIMA, 2004), originários da Estação TABA, o Projeto MILOS (HOLZ et al., 2001) e a Gerência de Conhecimento em ODE (NATALI et al., 2003). Esse último será explorado em detalhes no próximo capítulo.

Ambientes de Desenvolvimento de Software Orientados a Organização (ADSOrg) são uma classe de ADSs que apóia a atividade de Engenharia de Software em uma organização, fornecendo conhecimento acumulado pela organização e relevante para esta atividade, ao mesmo tempo em que apóia, a partir dos projetos específicos, o aprendizado organizacional em Engenharia de Software. ADSOrg representam uma evolução dos ADSODs, tendo como objetivo evitar que este conhecimento de software fique disperso ao longo da estrutura organizacional e, conseqüentemente, sujeito a dificuldades de acesso e mesmo a perdas (LIMA et al., 2000).

Para apoiar o gerenciamento do conhecimento em organizações, deve-se considerar alguns requisitos para ADSOrg, dentre eles: (i) ter a representação da infra-estrutura da organização, (ii) reter conhecimento especializado sobre desenvolvimento e manutenção de software, (iii) permitir a utilização deste conhecimento em projetos, (iv) apoiar a atualização constante do conhecimento armazenado no ambiente e (v) facilitar a localização de especialistas da organização que podem ser úteis em um projeto (LIMA et al., 2000).

O Projeto MILOS (MAURER *et al.*, 2002) é uma parceria da Universidade de Calgary, Canadá, com a Universidade de Kaiserslautern, Alemanha, que tem a finalidade de oferecer uma infra-estrutura que integre os conceitos de ADSCP e de Gerência de Conhecimento. O ambiente construído é composto por um Ambiente de Modelagem Estendida de Processos, um Ambiente de Planejamento de Projeto, um Ambiente de Controle de Fluxos de Trabalho (*workflows*) e um Assistente de Informação. Essas ferramentas armazenam e manipulam modelos genéricos de processos, planos de projeto e dados de projetos.

2.4 Automatização da Gerência de Projetos

Existem, atualmente, diversas ferramentas CASE que apóiam, de forma específica, determinadas atividades de gerência de projetos de software e não estão integradas a um ambiente de desenvolvimento de software, o que dificulta o controle do processo e torna a entrada de dados das ferramentas uma tarefa repetitiva e propensa a inconsistências. No entanto, são poucos os relatos de ambientes que integrem ferramentas de apoio às várias atividades de gerência.

Dentre as ferramentas CASE de gerência de riscos, por exemplo, pode ser citada a ferramenta *RiskFree* (KNOB et al., 2006). *RiskFree* foi desenvolvida baseada nas boas práticas do PMBOK e é aderente à área de processo de gerência de riscos do modelo CMMI. Assim, *RiskFree* apóia cada uma das etapas que compõem o processo de gerência de riscos, a saber: planejamento da gerência de riscos, identificação dos riscos, análise dos riscos, planejamento de respostas aos riscos e monitoração e controle dos riscos.

No âmbito de estimativas de tamanho, podem ser citadas, por exemplo, as ferramentas *Function Point WorkBench* (CHARISMATEK SOFTWARE METRICS, 2006) e *FP Recorder* (CHRIS PTY, 2006), que apóiam a análise de pontos de função, permitindo decompor o sistema em módulos (decomposição do produto). Apesar de se limitar a esse tipo de estimativa, essas ferramentas apresentam alguns aspectos bastante interessantes. Um desses aspectos é permitir agrupar itens de estimativa, a partir de estimativas dos subitens, no caso módulos. No caso de *Function Point WorkBench*, também é fornecida ao usuário uma funcionalidade para cadastrar arquivos e itens de dados de cada arquivo, de modo que, durante a estimativa, basta selecionar o arquivo que a ferramenta conta quantos itens de dados estão associados, automatizando as estimativas. *Function Point WorkBench* permite, ainda, exportar estimativas em um formato XML e gerar relatórios. Além disso, ao fazer uma estimativa, permite que essa tarefa seja baseada em modelos já cadastrados (CARVALHO et al., 2006).

Ainda na área de estimativas, a ferramenta *Estimate Easy UC* (DUVESSA SOFTWARE, 2006), apóia a análise de pontos de casos de uso, permitindo que atores e casos de uso sejam importados de modelos XMI, formato XML padrão para intercâmbio de modelos UML e modelos no formato da ferramenta Rational Rose, dentre outros.

É válido notar que há, nas ferramentas CASE de apoio a estimativas acima citadas, uma preocupação em prover funcionalidades que permitam a importação de dados visando a facilitar a entrada de dados nessas ferramentas. No entanto, em ferramentas integradas a Ambientes de Desenvolvimento de Software, essa transferência de dados entre as ferramentas pode ocorrer de forma mais natural e segura.

Assim como acontece com as atividades de estimativas e gerência de riscos, existem diversas ferramentas CASE que apóiam as demais atividades de gerência. Porém, conforme já citado, vários benefícios são obtidos através da integração de ferramentas em um ADS.

Nesse contexto, merece destaque a estação TABA. O Projeto TABA, iniciado em 1990, teve como objetivo inicial construir ambientes de desenvolvimento de software (ADS) centrados em processos e adequados a projetos com diferentes características. Em 1997, iniciou-se uma importante evolução da Estação TABA para torná-la um ambiente capaz de instanciar Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSOD), considerando não apenas as características específicas dos projetos, mas também o domínio da aplicação (OLIVEIRA, 1999). Uma outra evolução dos ambientes TABA definiu e implementou um modelo para construção de ADSs com Gerência de Conhecimento, denominados no contexto da Estação TABA de ADSs Orientados a Organização (ADSOrgs) (VILLELA, 2004).

O conjunto de ferramentas disponíveis na Estação TABA evolui constantemente. BARRETO (2006) relaciona as ferramentas atualmente disponíveis e passíveis de integração aos ambientes TABA. No contexto da gerência de projetos, TABA dispõe de ferramentas para apoiar o planejamento e o controle de tempo e custo em projetos de software, ferramenta para apoiar o planejamento e a monitoração de riscos em projetos de software, ferramentas para apoiar o planejamento, monitoração e avaliação da alocação de profissionais aos projetos de software, ferramenta que informa a situação dos projetos sendo conduzidos pela organização, ferramenta para apoiar a coleta de métricas, ferramenta para apoiar o planejamento do acompanhamento e controle do projeto e ferramenta para apoiar a instanciação de processos de software para projetos específicos.

Conforme citado na seção 2.1, durante a execução de um projeto muitos benefícios podem ser obtidos pela utilização de dados de projetos anteriormente realizados. Porém, para que se

possa realizar o agrupamento de projetos similares é necessário que sejam definidas formas de caracterização de projetos e de cálculo de similaridade entre projetos.

A caracterização de projetos na estação TABA, definida em (BARCELLOS, 2003), leva em consideração critérios fixos definidos com base na literatura. A busca por projetos similares, por sua vez, é realizada através da igualdade de todos os critérios avaliados. Segundo a própria autora, uma técnica de busca por projetos similares deverá ser desenvolvida futuramente. Quanto à caracterização de projetos, apesar do grande trabalho de revisão bibliográfica efetuado para a definição dos critérios a serem utilizados, acreditamos que a definição de uma estrutura de caracterização mais flexível, que permita que as organizações definam os critérios que desejam utilizar poderia trazer ganhos, pois permitiria que cada organização focasse nas características dos projetos mais adequadas à sua realidade.

2.5 O Ambiente ODE

O ADS utilizado como foco de pesquisa neste trabalho é o ambiente ODE (*Ontology-based software Development Environment*) (FALBO et al., 2004a), um ADS centrado em processo, desenvolvido no Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES/UFES), que tem sua fundamentação baseada em ontologias.

A principal característica que distingue ODE de outros ADSs é ser desenvolvido baseado em ontologias. Uma ontologia é a especificação de uma conceituação (GRUBER, 1995), que consiste, basicamente, de conceitos e relações, suas definições, propriedades e restrições, descritos na forma de axiomas. Uma ontologia não é somente uma hierarquia de termos, mas uma teoria completa axiomatizada de um domínio (GUIZZARDI et al., 2001).

Se as ferramentas de um ADS são construídas baseadas em ontologias, a integração dessas ferramentas pode ser amplamente facilitada, pois os conceitos envolvidos estão bem definidos na ontologia. A mesma ontologia pode ser usada para construir diferentes ferramentas que apóiam atividades relacionadas da engenharia de software (FALBO et al., 2002).

ODE é implementado em Java e, em sua fase atual, possui ferramentas de apoio a várias atividades de gerência de projetos de software, a saber: apoio à definição de processos de software (BERTOLLO et al., 2006), acompanhamento de projetos (ControlPro) (DAL MORO et

al., 2005), gerência de recursos humanos (GerênciaRH), gerência de riscos (GeRis) (FALBO et al., 2004a) e realização de estimativas (EstimaODE) (CARVALHO et al., 2006). Vale citar que a ferramenta de apoio a estimativas, EstimaODE, suporta a realização de estimativas de tamanho utilizando as técnicas de análise de pontos de função e análise de pontos de caso de uso. EstimaODE apóia, ainda, a realização de estimativas de esforço com base em dados históricos.

ODE dispõe, ainda, de uma Infra-estrutura de Gerência de Conhecimento (NATALI, 2003). Essa infra-estrutura integra um Sistema de Gerência de Conhecimento a ODE, com o objetivo de suportar a administração do conhecimento gerado durante o desenvolvimento de software.

Apesar de ODE apoiar a realização dessas diversas atividades, há a necessidade de gerenciar o conhecimento envolvido nos processos decisórios inerentes às atividades, com destaque para as atividades da gerência de projetos. Utilizando uma abordagem de gerência de conhecimento, o conhecimento embutido nas diversas atividades da gerência de projetos pode ser capturado, armazenado, disseminado e reutilizado, para oferecer maiores subsídios às tomadas de decisão, permitindo que as atividades de gerência sejam realizadas com maior segurança e eficiência.

2.6 Considerações Finais do Capítulo

O bom gerenciamento de projetos não pode garantir o seu sucesso. Contudo, o mau gerenciamento, geralmente, resulta no fracasso do projeto (SOMMERVILE, 2003).

A gerência de projetos envolve um conjunto de atividades complexas, relacionadas a um grande número de variáveis e muita incerteza. Dentre as atividades de gerência, destacam-se a alocação de recursos, a elaboração de cronogramas, a análise de riscos e a geração de estimativas de tamanho, esforço, custo e tempo. Os resultados de diversos estudos realizados por MIRANDA (2001) e BASILI et al. (2001), dentre outros, demonstram a efetividade da documentação do projeto e da utilização de dados históricos para a produção de melhores estimativas.

Assim, os gerentes de projetos são responsáveis por importantes decisões em um projeto de software, que podem, inclusive, definir o sucesso ou fracasso do mesmo. Neste contexto, torna-se evidente a necessidade de se prover um apoio automatizado aos gerentes de projeto. Em

se tratando de projetos de software, uma boa forma de se prover esse apoio é por meio de Ambientes de Desenvolvimento de Software.

Em processos complexos, como o processo de gerência de projetos, as experiências adquiridas em projetos anteriores são de grande valia para novos projetos. Para viabilizar a reutilização do conhecimento, é preciso retê-lo e armazená-lo de forma a minimizar o esforço para obtê-lo no futuro. Neste contexto, a gerência de conhecimento representa uma forma de assegurar o fluxo, a captura do conhecimento no nível da instituição e promover o surgimento de novos conhecimentos (LIMA et al., 2000).

A gerência de conhecimento é o foco do próximo capítulo.

Capítulo 3

Gerência de Conhecimento

O maior bem de uma organização de software é o seu capital intelectual. O maior problema do capital intelectual é que ele tem pernas e vai para casa todos os dias (RUS et al., 2002). A efetiva implementação de estratégias de aquisição e manutenção de conhecimento é importante para preservar as competências e promover a troca de conhecimento nas organizações (BIRK et al., 2002).

A gerência de conhecimento é uma disciplina emergente que visa a preservar e expandir o capital intelectual das organizações (RUS et al., 2002). De maneira mais detalhada, a gerência de conhecimento tem seu foco voltado para a solução de questões relacionadas a: como as organizações podem tirar maior proveito do conhecimento existente dentro delas, como membros da organização podem distribuir o conhecimento para quem este pode ser útil, como registrar as soluções adotadas para tratar problemas, como reter o conhecimento de seus especialistas mesmo quando esses deixam a organização, além de discutir formas de se gerar novo conhecimento a partir do conhecimento existente dentro da organização ou a partir de fontes externas (DAVENPORT et al., 1998).

Em organizações de desenvolvimento de software, a identificação, a manutenção e a aplicação, de um projeto para outro, de diferentes tipos de conhecimento relacionados ao desenvolvimento de software são importantes para desenvolver software de alta qualidade e aprimorar os processos de software (HOUDEK et al., 1999).

Este capítulo procura dar uma visão geral da área de Gerência de Conhecimento, procurando focar os aspectos mais importantes para este trabalho. A seção 3.1 discute os principais conceitos relativos à gerência de conhecimento, procurando definir o que é gerência de conhecimento, ressaltar a diferença entre dados, informação e conhecimento, e destacando dois

tipos principais de conhecimento: tácito e explícito. Na seção 3.2, o enfoque é direcionado para o processo de gerência de conhecimento, procurando discutir as principais atividades da gerência de conhecimento. Essa seção apresenta, ainda, alguns conceitos relacionados a sistemas de gerência de conhecimento. A seção 3.3 apresenta métodos, técnicas e tecnologias potencialmente úteis para apoiar as atividades envolvidas na gerência de conhecimento. A seção 3.4 aborda como a engenharia de software pode obter benefícios da gerência de conhecimento, apresentando algumas iniciativas de uso de gerência de conhecimento na engenharia de software, em especial, iniciativas de apoio a atividades relacionadas à gerência de projetos e iniciativas visando à captura de conhecimento tácito. Na seção 3.5 é apresentada a infra-estrutura de apoio à gerência de conhecimento de ODE. Finalmente, a seção 3.6 apresenta as considerações finais deste capítulo.

3.1 Conceitos Relativos à Gerência de Conhecimento

Várias definições são encontradas na literatura para o termo gerência de conhecimento. SALLIS et al. (2002) definem gerência de conhecimento como uma forma sistemática de gerenciar o conhecimento individual e organizacional utilizando tecnologia apropriada. Para O'LEARY (2001), a gerência de conhecimento visa a capturar e armazenar os recursos de conhecimento em uma organização, facilitando seu acesso, compartilhamento e reuso, através do uso de tecnologia apropriada. BENJAMINS et al. (1998) ressaltam que gerência de conhecimento não é um produto ou uma solução que possa ser comprada pronta, e sim um processo implementado durante um período de tempo, que envolve tanto relacionamentos humanos como práticas de negócio e tecnologia de informação.

LIMA (2004), sintetizando várias definições encontradas na literatura, afirma que a gerência do conhecimento é a administração, de forma sistemática e ativa, dos recursos de conhecimento de uma organização, utilizando tecnologia apropriada e visando a fornecer benefícios estratégicos à organização, o que envolve: comunicação efetiva e fluxo de conhecimento entre os indivíduos da organização, captura, organização e armazenamento de conhecimento relevante a partir de fontes internas e/ou externas disponíveis para a organização, disponibilização e distribuição do conhecimento capturado de forma adequada às necessidades dos usuários, desenvolvimento de novo conhecimento e atualização de conhecimento defasado.

3.1.1 Dado, Informação e Conhecimento

Para trabalhar com gerência de conhecimento, é essencial entender a diferença entre dado, informação e conhecimento, pois o desempenho organizacional depende, muitas vezes, de saber reconhecer quais deles a organização necessita, quais deles ela já possui e o que se pode fazer ou não com cada um deles (MARKKULA, 1999).

Na literatura há muito debate sobre o que é conhecimento, o que são dados e o que é informação, sendo que é possível perceber que esses são conceitos inter-relacionados, tanto que, muitas vezes, um termo é definido com base em outro, como aponta (STENMARK, 2001). Segundo SPEK et al. (1997), por exemplo, dados são símbolos não interpretados, enquanto informações são dados com significados e conhecimento é a capacidade de atribuir significado.

Para DAVENPORT et al. (1998), dados são constituídos por um conjunto de fatos discretos, informação é uma mensagem significativa para mudar a percepção do receptor e conhecimento são experiência, valores, interpretação e informação contextual.

NONAKA et al. (1997) argumentam que conhecimento e informação são ambos específicos ao contexto e relacionados. Assim, conhecimento e informação são similares em alguns aspectos, mas diferentes em outros. Enquanto informação é mais factual, conhecimento envolve crenças e valores. E, ainda, conhecimento está diretamente relacionado à ação, ou seja, deve ser utilizado para alguma finalidade.

Segundo NONAKA et al. (1997) existem dois tipos de conhecimento: o conhecimento tácito e o conhecimento explícito. O conhecimento tácito representa o conhecimento subjetivo, não documentado e presente apenas nas cabeças das pessoas. Esse tipo de conhecimento depende de experiência pessoal e envolve fatores intangíveis como crenças, perspectivas, intuição e valores. O conhecimento explícito, por sua vez, representa o conhecimento objetivo, documentado e explicitado de alguma forma, de modo que possa ser acessado por várias pessoas. Conhecimento nesse formato pode ser facilmente transmitido e compartilhado na forma de princípios universais, fórmulas científicas, procedimentos codificados, entre outros.

NONAKA et al. (1997) consideram que um trabalho efetivo com o conhecimento somente é possível em um ambiente em que possa ocorrer a contínua conversão entre esses dois

formatos. A conversão de conhecimentos entre os formatos tácito e explícito, como mostra a Figura 3.1, ocorre de quatro modos, a saber:



Figura 3.1 – Espiral do Conhecimento (NONAKA et al., 1997).

- **Socialização:** é o processo de conversão de conhecimento tácito de uma pessoa em conhecimento tácito de outra pessoa. Ocorre através do diálogo, do compartilhamento de experiências, de sessões de *brainstorm* etc;
- **Externalização:** é o processo de conversão de conhecimento tácito em conhecimento explícito, por meio de representação simbólica do conhecimento tácito através de modelos, conceitos, hipóteses etc. Essa conversão pode ocorrer, também, a partir de relatos orais e filmes e de representações simbólicas do conhecimento tácito por meio de planilhas, textos, imagens, figuras etc.
- **Combinação:** é o processo de conversão de conhecimento explícito em explícito por meio do agrupamento e processamento de diferentes conjuntos de

conhecimento explícitos. Pode ocorrer por meio de trocas de documentos, reuniões, conversas etc.

- Internalização: é o processo de incorporação do conhecimento explícito no conhecimento tácito, podendo ocorrer por meio da leitura de documentos, do aprendizado através da prática etc.

3.2 Processo de Gerência de Conhecimento

As organizações aprendem através do aprendizado de seus membros. O aprendizado individual não garante o aprendizado da organização. Porém, sem o aprendizado individual, nenhum aprendizado organizacional pode ocorrer. Grupos de pessoas que precisam trabalhar juntos para solucionar um problema ou executar uma tarefa possuem um mecanismo de aprendizado próprio. O compartilhamento de conhecimento entre diferentes pessoas é um componente crítico do processo de aprendizado (RUS et al., 2002).

O compartilhamento de conhecimento pode ser *ad hoc* ou organizado (sistemático). A gerência de conhecimento tem sido vista por diversos autores (FISCHER et al., 2001; ABECKER et al., 1998; WIIG, 1999; STAAB et al., 2001) como um processo cíclico envolvendo atividades relacionadas entre si.

Apesar da definição das atividades do processo de gerência de conhecimento variarem de autor para autor, de uma forma geral, as propostas de processos de gerência de conhecimento encontradas na literatura apresentam uma atividade de criação e aquisição de conhecimento como atividade inicial do processo, que tem como objetivo capturar o conhecimento e convertê-lo para incorporar as convenções organizacionais. A atividade de criação é seguida por uma atividade relacionada ao armazenamento e à organização do conhecimento, cujo objetivo é organizar, classificar e armazenar o conhecimento em repositórios. A terceira atividade lida com o acesso ao conhecimento. Essa atividade pode acontecer através da distribuição pró-ativa do conhecimento por meio de ferramentas ou através do fornecimento de meios que permitam que os indivíduos acessem o conhecimento quando necessitarem. Alguns autores propõem, ainda, atividades relacionadas ao uso e à preservação do conhecimento, o que envolve a manutenção dos itens de conhecimento armazenados. Dados sobre o uso do conhecimento são fundamentais para a

identificação de novos itens de conhecimento relevantes ou itens obsoletos, sendo a base para a evolução da memória organizacional.

Segundo MARKKULA (1999), os fatores que apóiam o processo de gerência de conhecimento são cultura organizacional, liderança, medição e tecnologia. Para que um projeto de gerência de conhecimento obtenha sucesso, é imprescindível que haja uma cultura organizacional com uma orientação positiva e uma motivação para criar, compartilhar e utilizar conhecimento. No que se refere à liderança, o conhecimento não será bem gerenciado até que um grupo, exercendo uma liderança dentro da organização, tenha responsabilidade sobre o trabalho. Com relação à medição, o valor do conhecimento não pode ser medido diretamente, mas é possível medir seus resultados, como o aumento da lucratividade e a eficiência ou taxa de inovação resultante desses esforços de gerência de conhecimento. Finalmente, o objetivo da tecnologia é capturar o conhecimento existente na cabeça das pessoas ou nos documentos e torná-los disponíveis para toda a organização.

Nesse contexto, ganham importância os sistemas de gerência de conhecimento. Um sistema de gerência de conhecimento é um sistema que interage com outros sistemas da organização para facilitar aspectos do processamento do conhecimento. Sistemas de gerência de conhecimento são as ferramentas para gerenciar o conhecimento, auxiliando as organizações nas soluções de problemas e facilitando a tomada de decisões (SCHREIBER et al., 1999).

Segundo NATALI (2003), um sistema de gerência de conhecimento possui em seu centro uma memória organizacional e, ao redor dessa memória, encontram-se vários serviços para apoiar as atividades do processo de gerência de conhecimento. A memória organizacional é uma representação explícita e persistente do conhecimento e das informações cruciais para uma organização, cuja finalidade é facilitar seu acesso, compartilhamento e reuso pelos diversos membros da organização (HEIJST et al., 1996; DIENG et al., 1999).

Dentro do contexto da engenharia de software, a memória organizacional pode ser representada por um ou diversos repositórios de conhecimento relacionados ao desenvolvimento de software, como, por exemplo, processos, procedimentos, métricas de projetos, lições aprendidas, notícias, competências das pessoas, artefatos de projeto, artefatos para reuso, entre outros (VALENTE, 2003).

3.3 Abordagens e Ferramentas Utilizados para Apoiar a Gerência de Conhecimento

Muitas empresas estão simplesmente renomeando seus produtos e abordagens, por exemplo, de gerência de informação para gerência de conhecimento, de base de dados para base de conhecimento, de *data warehouses* para repositórios de conhecimento. No entanto, soluções de Gerência de Conhecimento não são simplesmente novos nomes. Elas devem incluir características enriquecidas de conhecimento (LIMA et al., 2000).

Dentre as abordagens, ferramentas e técnicas utilizadas para apoiar a gerência de conhecimento estão, ontologias, máquinas de busca, XML, agentes inteligentes, modelos de visualização, bases de dados e *data warehouses*, *groupware*, gerência de documentos, sistemas baseados em conhecimento e sistemas especialistas.

No contexto deste trabalho, merece destaque o uso de raciocínio baseado em casos, *design rationale* e ferramentas de *groupware*.

3.3.1 Raciocínio Baseado em Casos

Gerentes de projetos têm de tomar uma série de decisões durante a realização de projetos, especialmente em sua fase inicial. Tipicamente, tais decisões são guiadas pelas experiências pessoais e intuições dos gerentes. Porém, devido à complexidade do processo de desenvolvimento de software, a intuição não é suficiente e nem todos os gerentes têm vasta experiência (RUS et al., 2002). Por essas razões, a estratégia de gerência de conhecimento pode apoiar a tomada de decisão para projetos futuros baseada em projetos passados. Neste contexto, as abordagens de solução de problemas através de Raciocínio Baseado em Casos aparecem como uma boa opção.

O Raciocínio Baseado em Casos (RBC) é uma abordagem para a solução de problemas e para o aprendizado com base em experiências passadas. Quando um novo problema é encontrado, RBC relembra casos similares e adapta as soluções que funcionaram no passado ao problema corrente. Subjacente a esse enfoque está a suposição de que problemas cujas descrições possuem formas similares apresentam soluções similares. Conseqüentemente, soluções de problemas

prévios similares ao atual são um ponto de partida útil para a solução de um novo problema (WANGENHEIM et al., 2003).

O modelo mais aceito para o processo de RBC é o *Ciclo de RBC* proposto por AAMONDT e PLAZA (AAMODT et al., 1994), que engloba um ciclo de raciocínio contínuo composto por quatro tarefas principais, a saber: recuperar os casos mais similares da base de casos, reutilizar esses casos para resolver o problema, revisar a solução proposta e reter a experiência, representando o caso atual para reutilização futura.

3.3.2 Design Rationale

Segundo BUCKINGHAM (1996), atualmente o processo de desenvolvimento de software é altamente orientado a artefatos, ou seja, está focado em gerar e manter os artefatos durante o ciclo de vida, culminando no sistema final. Apesar desse processo atingir o objetivo final, as razões por trás das decisões que levaram aos artefatos, geralmente, permanecem implícitas e são, conseqüentemente, difíceis de recuperar e reutilizar. Assim, as organizações estão, cada vez mais, reconhecendo a importância de encontrar formas de capturar e reusar o esforço, o conhecimento e a especialidade investidos no desenvolvimento de projetos.

O desafio das pesquisas em *design rationale* é encontrar as representações mais úteis e acessíveis para representar o raciocínio, que minimizem o esforço de codificação do raciocínio e maximizem a utilidade dessa representação (BUCKINGHAM, 1996).

De acordo com LEE (1997), *design rationale* visa a capturar não só as razões existentes por trás de uma decisão, mas também as justificativas para a decisão, as alternativas consideradas e as argumentações que levaram à decisão.

MEDEIROS (2006) apresenta a seguinte lista de requisitos para a representação de *design rationale*:

1. *Representação explícita de decisões.* Uma mesma alternativa de design pode ser considerada, ou não, uma solução para questões de design diferentes. Portanto, as decisões tomadas sobre a aceitação ou não dessa alternativa como uma solução para cada questão de design devem ser registradas separadamente.

2. *Distinção entre argumentos e justificativa final.* Algumas vezes o projetista decide aceitar uma idéia de solução que possui apenas argumentos contrários. Podem existir também idéias que, pelo número de argumentos a favor, parecem representar uma melhor solução de design do que a idéia escolhida pelo projetista para uma dada questão. Isto pode indicar que existe um *rationale* que não está sendo registrado ou que a escolha dessa idéia de design deve ser reconsiderada pelo projetista. Geralmente, o *rationale* que está faltando pode ser registrado como um novo argumento. Porém, em alguns casos não se trata de um novo argumento para a idéia de solução proposta, mas sim de uma justificativa final para a decisão tomada pelo projetista após analisar os argumentos apresentados. Por exemplo, essa justificativa final poderia registrar os motivos pelos quais o projetista decidiu usar uma idéia de solução, apesar de todos os argumentos apresentados para ela serem contrários à sua aceitação.
3. *Integração da argumentação com as descrições dos artefatos gerados.* A complexidade e o volume de informações presentes nas representações de *design rationale* torna difícil para o projetista compreender as diversas idéias de solução e o resultado das decisões tomadas. A relação entre a argumentação registrada e o artefato produzido pode facilitar essa compreensão e ajudar o projetista a decidir se vale a pena reusar um artefato ou não.
4. *Registro de informações sobre as atividades de design.* Registrar apenas a argumentação usada pelos projetistas não é suficiente para compreender o conhecimento usado por eles durante o design. É necessário também ter informações sobre o design em si, como por exemplo, o método de design utilizado, as atividades realizadas e os responsáveis pelas decisões tomadas.
5. *Definição de uma linguagem de representação expressiva.* É necessário descrever *design rationale* em uma linguagem que permita o seu processamento computacional e de forma independente da ferramenta de design utilizada.
6. *Integração do design rationale com o método de design.* Representações genéricas e informais de *design rationale* dificultam o reuso do *rationale* em novos designs. A integração do *rationale* com a semântica formal definida pelos métodos de

design torna o conteúdo registrado mais rico e expressivo, possibilitando a realização de operações computáveis capazes de apoiar novos usos de *design rationale*, como por exemplo, a combinação de *rationales* de artefatos similares.

As atividades da gerência de projetos de software são compostas por diversos processos decisórios. Em muitas dessas atividades, informações que indiquem o raciocínio seguido para produzir um artefato no passado podem ser mais úteis para guiar a produção de um artefato similar em um novo projeto que o próprio artefato em si. Assim, fica evidente o potencial de *Design Rationale* no contexto da gerência de projetos de software apoiada por gerência de conhecimento.

3.3.3 Ferramentas de Groupware

O desenvolvimento de software é uma tarefa essencialmente cooperativa. Gerentes e desenvolvedores agem em conjunto na tentativa de se obter um software de qualidade dentro de prazos e custos definidos.

Neste contexto, as atividades relativas ao desenvolvimento de software podem ser beneficiadas pelos conceitos e resultados obtidos pelas pesquisas que envolvem a área de Trabalho Cooperativo Apoiado por Computador (*Computer-suported Cooperative Work – CSCW*).

CSCW é uma área de estudos ampla, focada no trabalho, ou seja, nas tarefas que as pessoas realizam, nos seus locais de trabalho e nas tecnologias que provêm ou poderiam prover suporte. Assim, CSCW abrange desde análises sociológicas e descrições antropológicas do trabalho até os fundamentos tecnológicos de sistemas (POLTROCK *et al.*, 1994).

As tecnologias geradas pelas pesquisas sobre CSCW são designadas pelo termo *Groupware*. Ferramentas de *groupware* são sistemas baseados em computador que fornecem uma interface para um ambiente compartilhado e permitem a grupos de pessoas colaborarem para realizar atividades ou objetivos comuns (GRUDIN, 1994).

Para CRUZ (1998b), *groupware* engloba diversas tecnologias baseadas no mesmo princípio: pessoas trabalhando juntas para que as atividades sejam realizadas com sucesso em todas as partes do processo, independentemente de quem as desenvolva.

Dentre os requisitos fundamentais da tecnologia de *groupware*, destacam-se (BROOKE, 1993):

- Facilitar a colaboração entre os indivíduos;
- Ser altamente configurável para se adaptar às necessidades dos usuários;
- Permitir acesso aos dados, independentemente da localização dos usuários;
- Permitir recuperação de informações armazenadas na base de dados;
- Gerenciar o acesso aos dados, em especial quando vários usuários tentam modificar os mesmos dados ao mesmo tempo;
- Garantir que as informações usadas no trabalho cooperativo sejam disseminadas entre a equipe.

Dentre os diversos sistemas voltados para apoiar o trabalho cooperativo, podem ser citados: sistemas de comunicação síncrona (*chats*, vídeo-conferência, mensagens instantâneas), sistemas de comunicação assíncrona (correio eletrônico, listas de discussão, fóruns), sistemas de apoio a reuniões, editores cooperativos, sistemas para gerenciamento eletrônico de documentos e sistemas de fluxo de trabalho (*workflows*).

Devido ao caráter cooperativo inerente às ferramentas de *groupware*, elas aparecem como uma ótima opção para apoiar a captura, a disseminação e a socialização do conhecimento, evidenciando, assim, a sinergia que há entre gerência de conhecimento e *groupware*.

Mas, para que seja possível a construção de recursos que apóiem de maneira eficiente o trabalho cooperativo no contexto do processo de software, é necessário compreender as questões que envolvem a participação dos diversos elementos de uma equipe de software. Partindo da visão do processo de desenvolvimento de software como um processo contínuo de entendimento, quatro questões devem ser consideradas para suportar a cooperação neste contexto, a saber: a comunicação entre os participantes envolvidos no processo, a coordenação de suas atividades, o registro do conhecimento comum pela memória de grupo e a percepção do grupo em relação ao contexto de trabalho sendo realizado. Essas questões não podem ser consideradas monoliticamente, uma vez que se encontram intimamente dependentes e inter-relacionadas. (ARAUJO et al., 1997).

3.4 Gerência de Conhecimento em Engenharia de Software

O conhecimento em Engenharia de Software é diversificado e suas proporções são imensas e crescem rapidamente. Organizações têm problemas para identificar, localizar e usar o conhecimento. Um melhor uso desse conhecimento é a motivação básica para a Gerência de Conhecimento em Engenharia de Software (RUS, et al., 2002).

RUS et al. (2002) citam, além dessa motivação básica, outras necessidades que motivam a utilização da gerência de conhecimento em Engenharia de Software: (i) diminuição do tempo e do custo de desenvolvimento e aumento da qualidade; (ii) melhoria no processo de tomada de decisão; (iii) aquisição de conhecimento sobre novas tecnologias; (iv) aquisição de conhecimento sobre novos domínios; (v) compartilhamento do conhecimento sobre políticas, práticas e cultura organizacionais; (vi) conhecimento sobre as competências dos membros da organização; e (vii) compartilhamento do conhecimento adquirido no desenvolvimento de software.

Os benefícios do uso da gerência de conhecimento para a engenharia de software são tão evidentes que podem ser encontradas na literatura abordagens de gerência de conhecimento especialmente projetadas para organizações de software. A mais conhecida dentre essas abordagens é a Fábrica de Experiências (BASILI et al., 1994). A fábrica de experiências reconhece que as organizações precisam aprender através de suas experiências anteriores, de forma a desenvolver produtos com maior rapidez, menor custo e com maior qualidade.

Dentre os vários trabalhos relacionados aos conceitos de Fábrica de Experiências, podem-se citar (NETO et al., 2001) e (VALENTE, 2003). NETO et al. (2001) se basearam nesses conceitos para desenvolver o Sistema de Gerência de Experiências (*Experience Management System* - EMS), um sistema de gerência de conhecimento que visa a apoiar a captura e o reúso de experiências relacionadas ao desenvolvimento de software, desenvolvido para ser utilizado em uma organização multinacional de consultoria em engenharia de software, a Q-Labs.

VALENTE (2003) apresenta uma abordagem baseada em gerência do conhecimento para suportar o armazenamento e compartilhamento de experiências e métricas de projetos que possam apoiar a realização de estimativas. Esse trabalho foi desenvolvido no contexto de uma empresa de desenvolvimento de software, o Centro de Desenvolvimento de Sistemas de Vitória (CDSV) da Xerox. Em trabalhos anteriores com a utilização de gerência de conhecimento nessa

organização, foi desenvolvida uma ferramenta baseada em gerência de conhecimento para apoiar a instanciamento de processos de software para projetos específicos a partir do processo padrão da organização, denominada ProKnowHow (BORGES et al., 2002). Em (VALENTE, 2003) as funcionalidades dessa ferramenta foram estendidas a fim de apoiar também a elaboração de estimativas. A arquitetura dessa extensão é baseada nos conceitos, passos e ciclos de realimentação do Paradigma de Melhoria de Qualidade, proposto pela Fábrica de Experiências (BASILI et al., 1994).

Diversos autores (RUS et al., 2002; SCHNEIDER et al., 2002; REIFER, 2002) ressaltam a importância da gerência de conhecimento no planejamento de projetos. Eles apontam que o principal benefício para o planejamento de projetos obtido pelo uso da gerência de conhecimento é a captura do conhecimento organizacional e de lições aprendidas e a disponibilização desses itens de forma sistemática. Assim, o conhecimento organizacional acumulado pode ser reutilizado, auxiliando na identificação, prevenção e mitigação de riscos, na realização das estimativas e no acompanhamento dos projetos.

Para se obter uma gerência de conhecimento efetiva, deve-se integrar o sistema de gerência de conhecimento ao ambiente de trabalho da organização, permitindo que o conhecimento relevante seja coletado e armazenado à medida que ele é gerado no trabalho (ABECKER et al., 1998), (O'LEARY, 2001). Em organizações de software, esse ambiente de trabalho pode ser um Ambiente de Desenvolvimento de Software (ADS) e, portanto, é muito útil integrar a gerência de conhecimento a ADSs (NATALI et al., 2003).

Neste contexto, merece destaque a Estação TABA (VILLELA, 2004). A Estação TABA, conforme já citado no capítulo anterior, trata-se de um ADS com apoio à gerência de conhecimento. A Estação TABA dispõe de algumas ferramentas que fazem uso da gerência de conhecimento, dentre as quais destacam-se as seguintes que apóiam atividades de gerência de projeto:

- *RiscManager* (FARIAS et al., 2003): Ferramenta que apóia o planejamento e a monitoração de riscos em projetos de software, considerando as características do projeto e a experiência da organização em projetos anteriores, oferecendo aos gerentes de projeto o conhecimento e experiências acumulados por diferentes gerentes em projetos similares ocorridos dentro da própria organização.

- *RHPlan* e *RHManager* (SCHNAIDER, 2003): Ferramentas que apóiam as atividades de planejamento, monitoração e avaliação da alocação de profissionais aos projetos de software, o que inclui a solicitação de contratação e capacitação, o acompanhamento das horas dedicadas a cada atividade, além da atualização, ao final do projeto, das competências por eles possuídas. Essas ferramentas permitem a utilização e a atualização do conhecimento organizacional de competências durante a realização das atividades de planejamento de recursos humanos.
- *TempPlan*, *TempManager*, *CustPlan* e *CustManager* (BARCELLOS, 2003; BARCELLOS *et al.*, 2003): Ferramentas que apóiam o planejamento e o controle de tempo e custo em projetos de software, baseadas na reutilização do conhecimento organizacional e nos modelos paramétricos COCOMO II e Análise de Pontos de Função. Um dos objetivos dessas ferramentas é oferecer aos gerentes de projeto o conhecimento e experiências acumulados por diferentes gerentes em projetos similares ocorridos dentro da própria organização.

Além das ferramentas de apoio às atividades de gerência de projeto, vale citar, ainda na Estação TABA, a ferramenta *Acknowledge* (MONTONI *et al.*, 2004). *Acknowledge* é uma ferramenta que visa a apoiar o processo de captura de conhecimento ao longo dos processos de software, englobando registro, filtragem e empacotamento de conhecimento. *Acknowledge* foi integrada às ferramentas que apóiam o processo de software na Estação TABA, a fim de evitar que os desenvolvedores tenham de sair de suas rotinas normais durante a captura e reuso do conhecimento.

O mais problemático desafio da gerência de conhecimento na Engenharia de Software é que a maior parte do conhecimento nesse domínio é conhecimento tácito e pode permanecer tácito pela ausência de tempo de torná-lo explícito ou mesmo pela dificuldade de explicitá-lo. Uma forma de tentar contornar esse problema pode ser o desenvolvimento de uma cultura de compartilhamento eficiente, assim como apoio tecnológico para a gerência de conhecimento, sem esquecer que o recurso mais importante de uma organização são seus funcionários (NATALI, 2003). Nesse contexto, algumas iniciativas visando a capturar o conhecimento tácito são encontradas na literatura.

OLIVEIRA (2006) propõe uma abordagem para a captura, o armazenamento e a recuperação de informações relacionadas à utilização de sistemas de informação, com o objetivo de contribuir para um melhor entendimento das equipes de manutenção de software em relação aos sistemas da organização. A autora considera que o conhecimento tácito dos usuários é importante para as equipes de manutenção, na medida em que esses usuários podem contribuir com informações de contexto, detalhes de utilização, hábitos e outras informações que não se encontram documentadas e que não são de domínio de toda a organização, devendo, assim, ser capturado e armazenado.

A solução proposta por OLIVEIRA (2006) captura conhecimento tácito a partir de histórias coletivas de uso dos sistemas de informação, contadas por usuários, e gera uma base de conhecimento para a organização, usando a técnica de *Group Storytelling* (PERRET, 2004). *Group Storytelling* é uma técnica aplicada em Gestão de Conhecimento, que utiliza histórias contadas em grupo como forma de comunicação e compartilhamento de conhecimento, na qual o conhecimento gerado é uma combinação do conhecimento tácito dos participantes.

TORRES (2006) propõe um ciclo de captura e disseminação do conhecimento adquirido em projetos de Tecnologia de Informação. O objetivo principal do ciclo proposto é disseminar, de forma ativa, conhecimento que permita aos receptores adaptá-lo para o seu contexto. O autor aplica Histórias de Aprendizagem como abordagem de Gestão de Conhecimento. Uma História de Aprendizagem é uma abordagem voltada para a aprendizagem organizacional, que busca capturar as histórias relacionadas a eventos inovadores que ocorreram na organização, mostrando não apenas o resultado, mas os processos mentais, opiniões e estruturas pessoais que foram partes constituintes do resultado (SENGE et al., 1999).

No contexto do ambiente ODE também foram realizados esforços visando a capturar conhecimento tácito. NATALI (2003), com o intuito de apoiar a captura de conhecimento tácito, definiu, no componente de criação e captura da infra-estrutura de gerência de conhecimento de ODE, mecanismos para a coleta de experiências sob a forma de lições aprendidas.

ARANTES (2004) desenvolveu funcionalidades e uma infra-estrutura de apoio ao trabalho cooperativo em ODE, incluindo ferramentas para troca de mensagens síncronas e assíncronas e funcionalidades para o empacotamento dessas mensagens, visando a capturar, desta

forma, o conhecimento envolvido nas interações entre desenvolvedores durante a realização de atividades do processo de software (FALBO et al., 2004c).

3.5 Gerência de Conhecimento em ODE

O ambiente ODE conta com uma infra-estrutura para gerência de conhecimento proposta originalmente por NATALI (2003) e refinada posteriormente, conforme apresentado em (FALBO et al., 2004c). A abordagem de gerência de conhecimento em ODE posiciona a memória organizacional no centro de sua infra-estrutura, apoiando o compartilhamento e o reúso de conhecimento. Em torno da memória organizacional se localizam cinco tipos de serviços que apóiam as atividades do processo de gerência de conhecimento, como mostra a Figura 3.2.

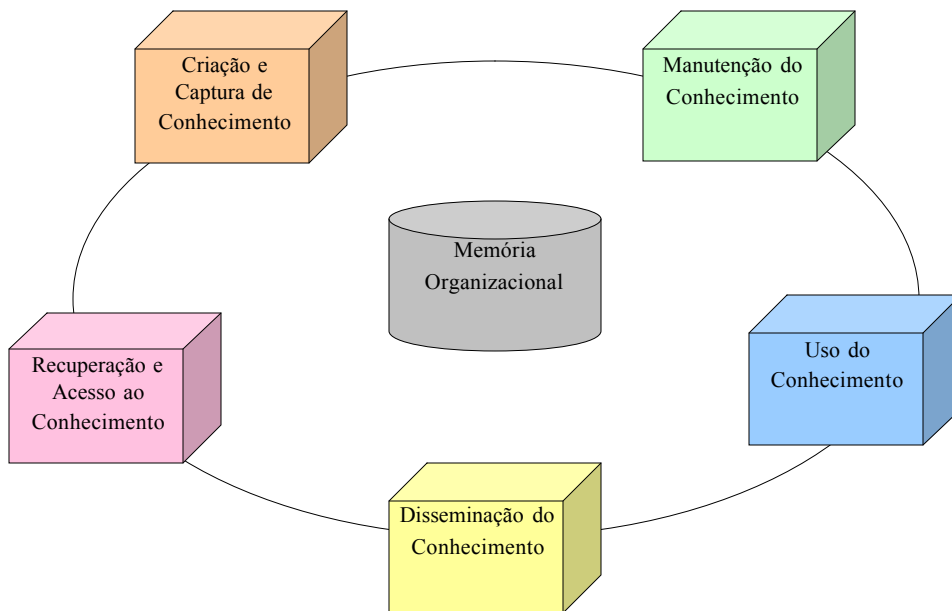


Figura 3.2 - Sistema de Gerência de Conhecimento de ODE.

Foi adotado um processo cíclico, envolvendo as cinco atividades consideradas, a saber:

- *Criação e Captura de Conhecimento*: visa à captura e criação dos itens de conhecimento que são gerados em ODE durante as interações existentes no

contexto do Sistema de Gerência de Conhecimento. Esses itens de conhecimento podem ser de três tipos, a saber: artefatos, lições aprendidas e pacotes de mensagens trocadas por membros da organização. Dessa forma, há facilidades para apoiar a criação e captura de cada um desses tipos de itens de conhecimento.

- *Recuperação e Acesso ao Conhecimento*: apóia as buscas por itens de conhecimento, realizadas por iniciativa dos usuários. Visa a fornecer itens da memória organizacional atendendo às demandas do usuário.
- *Disseminação do Conhecimento*: é um serviço do sistema que visa a identificar as necessidades do usuário e lhe fornecer, de forma pró-ativa, itens de conhecimento relevantes à tarefa que está sendo executada.
- *Uso do Conhecimento*: esse serviço é realizado quando o usuário utiliza um item de conhecimento. Visando a auxiliar a manutenção da memória organizacional, quando o usuário utiliza um item de conhecimento, ele pode fornecer informações acerca da eficácia do item em uma dada situação e até avaliar esse item.
- *Manutenção do Conhecimento*: dá apoio à manutenção dos itens de conhecimento, possibilitando ao gerente de conhecimento atualizar a memória organizacional, bem como efetuar algumas correções nos itens de conhecimento.

Os serviços de gerência de conhecimento de ODE são, ainda, categorizados de duas formas: serviços gerais e serviços específicos. Os serviços gerais são aqueles que estão incorporados ao ambiente. Eles estão disponíveis, da mesma forma, para todas as ferramentas, ou seja, são independentes de características particulares de cada ferramenta. São incluídos nessa categoria os serviços relacionados à captura, recuperação, uso e manutenção do conhecimento. Os serviços específicos são, em geral, relacionados à disseminação do conhecimento de forma pró-ativa. Assim, serviços desse tipo são implementados levando em consideração as particularidades de cada ferramenta e o perfil dos usuários (NATALI et al., 2003).

Cada um dos cinco tipos de serviços definidos em torno da memória organizacional de ODE pode ser especializado de várias maneiras. Uma delas, por exemplo, é a utilização de ferramentas de *groupware*, as quais apóiam a criação e a captura do conhecimento proveniente das interações dos membros de uma organização de software (ARANTES et al., 2004). Outro

exemplo de especialização do serviço de criação e captura do conhecimento é a ferramenta de cadastro de lições aprendidas que apóia o cadastro, a avaliação e a adaptação de lições aprendidas a serem disponibilizadas na memória organizacional de ODE (NATALI, 2003).

A Infra-Estrutura de Gerência de Conhecimento de ODE está em sua terceira revisão passando pelos trabalhos de NATALI (2003), ARANTES et al. (2004) e RUY (2006). O modelo de classes atual da Infra-Estrutura de Conhecimento é apresentado na Figura 3.3.

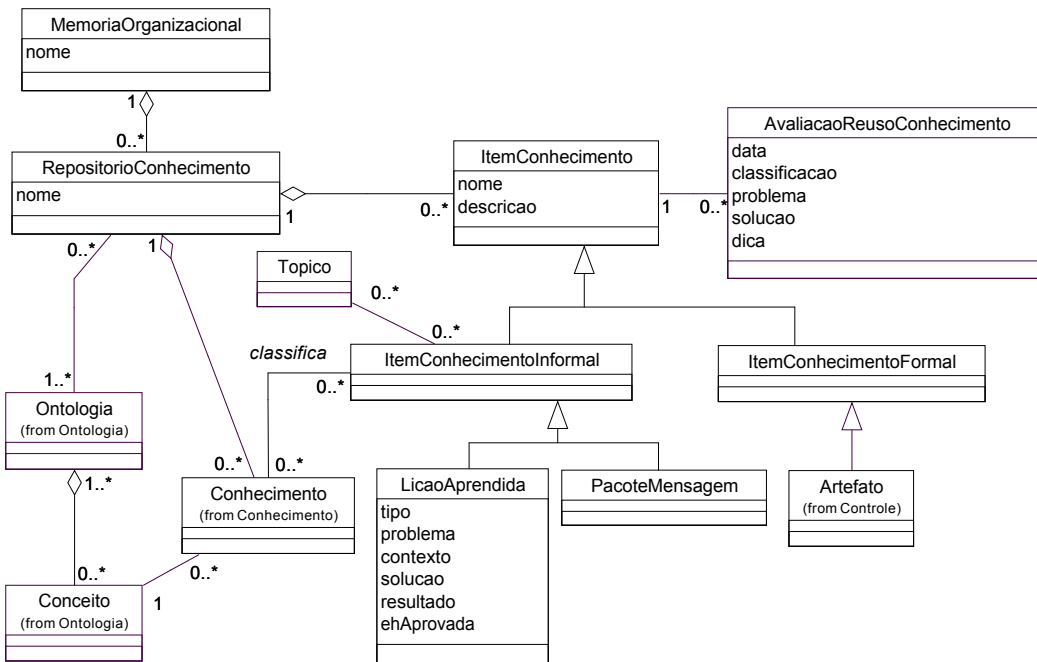


Figura 3.3 – Infra –estrutura de Gerência de Conhecimento de ODE.

Pela Figura 3.3 pode-se notar que a memória organizacional de ODE é composta por repositórios de conhecimento que contêm itens de conhecimento e objetos de conhecimento.

Os objetos de conhecimento são instâncias de conceitos das ontologias que norteiam a definição de todo o ambiente ODE, tratadas como subclasses da classe *Conhecimento* (FALBO et al., 2004a). Os objetos dessas subclasses descrevem conhecimento sobre domínios de aplicação, com destaque para o domínio da Engenharia de Software. Assim, são exemplos de objetos de

conhecimento os modelos de ciclo de vida e os paradigmas de desenvolvimento utilizados pela organização na condução de seus projetos.

Os itens de conhecimento, por sua vez, representam os itens produzidos no contexto dos projetos desenvolvidos e que podem ser reutilizados. Esses itens de conhecimento são categorizados de duas formas: (i) os itens de conhecimento formais, como os artefatos e, (ii) os itens de conhecimento informais, como as lições aprendidas e os pacotes de mensagens.

Com o intuito de prover mecanismos que tornem eficientes as buscas pelos itens de conhecimento, eles são organizados e classificados por meio de instâncias das classes *Topico* e *Conhecimento*.

Visando a auxiliar a manutenção dos repositórios e, conseqüentemente, da memória organizacional, sempre que um item de conhecimento é utilizado, um recurso humano da organização pode avaliá-lo quanto à utilidade para a atividade realizada, fornecendo, assim, parâmetros acerca da sua relevância.

Sobre a estrutura de classes da Figura 3.3 atuam os serviços de Gerência de Conhecimento a fim de apoiar os desenvolvedores na execução de suas tarefas diárias. Tal infra-estrutura foi construída em ODE, pois estudos sobre a Gerência de Conhecimento têm mostrado que se ela for aplicada ao desenvolvimento de software de forma a apoiar suas atividades, muito se tem a ganhar em termos de resolução de problemas (FALBO et al., 2004c).

3.6 Considerações Finais do Capítulo

Conhecimento é um dos recursos mais importantes de uma organização, influenciando a sua competitividade. Por isso, o compartilhamento do conhecimento dentro das organizações tem crescido como um mecanismo estratégico para impulsionar a oferta de serviços ao cliente, diminuir o tempo de desenvolvimento de produtos e compartilhar as melhores práticas (SKYRME, 1998). Neste contexto, a gerência de conhecimento vem ganhando importância nas organizações modernas.

Para DIENG (2000), a gerência de conhecimento visa a atingir os seguintes objetivos: transformar o conhecimento individual em conhecimento coletivo; dar apoio ao aprendizado e à

integração de um novo membro em uma organização; disseminar melhores práticas; melhorar os processos de trabalho corporativos, a qualidade de produtos desenvolvidos e a produtividade; e reduzir tempo de entrega de produtos.

Um dos maiores problemas enfrentados para implementar a gerência de conhecimento em uma organização são o tempo e o esforço necessários antes que se comece a obter retorno do investimento. Para se obter sucesso na gerência de conhecimento, é essencial que haja na organização uma cultura de apoio ao conhecimento, isto é, aquela que valoriza o aprendizado e motiva a criação e o compartilhamento do conhecimento (MARKKULA, 1999).

A gerência de conhecimento pode prover apoio à necessidade das organizações de software de fornecer sistemas com alta qualidade e baixo custo, no menor intervalo de tempo possível (ALTHOFF et al., 1999). Esse apoio se dá através do compartilhamento de conhecimento e de experiências anteriores da organização.

Para que os usuários busquem auxílio na memória organizacional e reutilizem o conhecimento armazenado, é necessário mostrar ao usuário que a memória organizacional possui informações úteis ao seu trabalho e que achá-las não será uma tarefa que consumirá muito tempo. Caso contrário, o usuário relutará em parar a sua atividade principal para buscar auxílio na memória organizacional (NATALI, 2003).

Visando a tornar mais eficiente a busca por itens de conhecimento úteis em um determinado contexto, construiu-se neste trabalho, uma infra-estrutura para caracterização de itens de software que permite a caracterização de itens de software e a recuperação de itens similares. A infra-estrutura construída é o tema do próximo capítulo.

Por fim, devido ao fato de a maior parte do conhecimento em Engenharia de Software ser conhecimento tácito, o maior desafio da gerência de conhecimento nesta área é conseguir explicitar esse conhecimento ou, ao menos, desenvolver um compartilhamento eficiente do mesmo.

Nesse contexto, foi desenvolvida neste trabalho, uma ferramenta para apoiar a decisão em grupo, baseada na técnica de Delphi (PFLEEGER, 2001), que visa a apoiar a construção cooperativa de artefatos de software. Por meio dessa ferramenta, com o intuito de explicitar o conhecimento tácito existente nos processos decisórios, tentou-se capturar o raciocínio seguido

pelos participantes da atividade até chegarem ao artefato final. Para isso foram utilizadas técnicas de *Design Rationale*, conforme discutido no Capítulo 5.

Capítulo 4

Infra-estrutura para Caracterização de Itens de Software em ODE

Em atividades de gerência de projetos de software, as soluções de problemas anteriores podem constituir bons pontos de partida para as soluções de problemas atuais. Exemplo disso é a realização de estimativas baseada em analogias (BARCELLOS, 2003) (PMI, 2004), nas quais dados históricos de projetos anteriores são utilizados para realizar as estimativas para o projeto corrente. Mas, o uso de analogias com projetos anteriores não se restringe às estimativas. Essa técnica pode ser utilizada em diversas atividades de gerência, como análise de riscos e alocação de recursos e, até mesmo, em atividades técnicas como especificação de requisitos, análise e projeto (*design*), dentre outras.

Essa possibilidade de utilização de dados de projetos anteriores em projetos atuais evidencia a aplicabilidade das técnicas de raciocínio baseado em casos no apoio à realização das atividades do processo de software.

O Raciocínio Baseado em Casos (RBC) é uma abordagem para a solução de problemas e para o aprendizado com base em experiências passadas. Quando um novo problema é encontrado, casos similares são recuperados e as soluções que funcionaram no passado são adaptadas ao problema corrente. Subjacente a este enfoque está a suposição de que problemas cuja descrição possui forma similar apresentam soluções similares. Conseqüentemente, soluções de problemas anteriores similares ao atual são um ponto de partida útil para a solução de um novo problema (WANGENHEIM et al., 2003).

Como uma questão fundamental para a gerência de conhecimento é a recuperação de itens de conhecimento relevantes para uma dada situação, percebeu-se que era importante prover em ODE uma estrutura flexível e poderosa para caracterizar itens de software e calculara

similaridade entre eles (CARVALHO et al., 2006). Assim, foi desenvolvida, utilizando técnicas de raciocínio baseado em casos, uma infra-estrutura genérica para caracterização de itens relacionados ao desenvolvimento de software em ODE, apresentada neste capítulo.

A seção 4.1 explica como as técnicas de RBC foram aplicadas na infra-estrutura. Na seção 4.2 são apresentadas as funcionalidades desenvolvidas em cada sub-sistema da infra-estrutura, bem como a estrutura interna desses subsistemas. Essa seção apresenta, ainda, as funcionalidades desenvolvidas para permitir a caracterização de projetos e a recuperação de projetos similares em ODE. A seção 4.3 apresenta o estudo feito para definir um conjunto inicial de características a serem utilizadas na caracterização de projetos em ODE. Finalmente, na seção 4.4 são apresentadas as considerações finais deste capítulo.

4.1 A Utilização de Raciocínio Baseado em Casos no Desenvolvimento da Infra-Estrutura de Caracterização de Itens de Software de ODE

O processo normalmente definido para uma abordagem de Raciocínio Baseado em Casos (RBC) é composto por três atividades principais: (i) representação e armazenamento de casos, (ii) recuperação de casos similares e (iii) adaptação das soluções de casos similares passados para gerar uma solução para o caso atual (WANGENHEIM et al., 2003).

Seguindo esse processo, o primeiro passo é definir a representação de casos. Um caso é composto, tipicamente, de dois componentes principais, a saber: o problema e a solução. Tratando-se do processo de software, podemos considerar a atividade em questão, juntamente com o que se espera produzir nela e o contexto em que a atividade será desenvolvida como sendo o problema e o artefato resultante como a solução. Por exemplo, durante a realização de uma estimativa de tamanho, podemos considerar que essa atividade é o problema e a estimativa a ser gerada a solução para tal problema, sendo as características do projeto e o escopo da estimativa (para o projeto como um todo, para um conjunto de módulos, para uma iteração etc) informações importantes que contextualizam a atividade e, portanto, caracterizam o caso.

Ainda que essa seja a concepção mais clara de um caso em um processo de software, é bastante comum a busca por projetos similares ou módulos e componentes similares e, portanto, para tornar a infra-estrutura mais genérica, considerou-se que quaisquer elementos relacionados

ao processo podem ser tratados como um caso. Assim, esses elementos serão tratados pela designação genérica de *objeto caracterizável*.

As ferramentas de apoio ao planejamento disponíveis em ODE apóiam a criação e o armazenamento das “soluções dos problemas”, ou seja, os artefatos (em nosso exemplo, as estimativas geradas). Para caracterizar nossos “problemas” (o trabalho envolvido nas atividades, bem como o contexto de realização das mesmas), é utilizada a infra-estrutura de caracterização apresentada neste trabalho.

Dentre as formas de representação utilizadas em RBC, optamos por utilizar neste trabalho uma das mais comumente utilizadas: a representação Atributo-Valor. Na representação Atributo-Valor, um caso é representado por um conjunto de pares atributo-valor. Cada par corresponde a uma característica do caso representado. Assim, a estimativa a ser realizada (problema) poderia ser caracterizada por pares como: [grau de instabilidade dos requisitos (atributo), alto (valor)], [nível de experiência da equipe (atributo), médio (valor)], dentre muitos pares atributo-valor possíveis.

Ao definir os atributos (características) que caracterizam um caso, temos de ter em mente que, por meio desses mesmos atributos, teremos de ser capazes de identificar casos similares ao atual. Dentre as medidas de similaridade entre dois casos freqüentemente utilizadas destacam-se a distância geométrica e a distância geométrica ponderada (WANGENHEIM et al., 2003).

A idéia básica da técnica da distância geométrica é que as ocorrências em uma base de casos podem ser vistas como pontos em um espaço multidimensional, onde cada atributo representa uma dimensão. Assim, a similaridade entre dois casos é inversamente proporcional à distância entre os pontos que os representam no espaço. Em suma, a distância geométrica entre dois casos é dada pela soma das diferenças entre os valores de cada um dos atributos dos casos. Ou seja, a distância geométrica (DG) entre dois casos é dada por:

$DG = \sum DC_i$, onde DC_i é a diferença entre os valores do atributo i para os dois casos.

Dessa forma, quanto menor é a distância geométrica (DG) entre dois casos, maior é a similaridade entre os mesmos.

A técnica da distância geométrica ponderada apenas acrescenta à técnica da distância geométrica a possibilidade de considerar a importância de cada atributo no cálculo da

similaridade. Assim, a distância geométrica ponderada (DGP) entre dois casos é dada pela soma dos produtos das diferenças entre os valores de cada um dos atributos dos casos por seus pesos:

$DGP = \sum (DC_i * P_i)$, onde DC_i é a diferença entre os valores do atributo i para os dois casos e P_i é o peso desse atributo no cálculo da distância.

Analogamente à técnica da distância geométrica, uma menor distância geométrica ponderada entre dois casos indica uma maior similaridade entre os mesmos.

Como é evidente o fato de que alguns atributos são mais importantes que outros na determinação de similaridade entre itens de software, neste trabalho utilizamos a técnica da distância geométrica ponderada para calcular o nível de similaridade entre dois casos.

Uma vez identificados na base de casos os problemas similares ao atual, o passo seguinte, segundo a abordagem de RBC, é adaptar as soluções dadas aos problemas similares passados, visando a gerar uma solução para o problema atual. Tendo em vista a complexidade inerente às atividades do processo de software, devido, sobretudo, ao grande número de variáveis envolvidas, acredita-se que a geração de novas soluções com base em soluções anteriores é um trabalho extremamente especializado e muito difícil de ser automatizado. Assim, neste trabalho foram tratadas apenas as etapas de representação e armazenamento de casos e recuperação de casos similares, visando a fornecer aos desenvolvedores dados históricos que os auxiliem durante a realização de atividades do processo de software.

4.2 A Infra-Estrutura Desenvolvida

A infra-estrutura de caracterização desenvolvida neste trabalho tem como requisito básico ser flexível a ponto de permitir a caracterização de diversos itens de software em ODE e o uso dessa informação de caracterização para a recuperação de itens similares. Tomando por base essas premissas, como mostra a Figura 4.1, dois subsistemas foram definidos, o primeiro relacionado ao controle das características a serem usadas na caracterização de itens e o segundo envolvendo as funcionalidades de caracterização e recuperação de itens propriamente ditas. A concepção e as funcionalidades providas por esses subsistemas são abordadas a seguir.



Figura 4.1 - Diagrama de Pacotes.

4.2.1 Cadastro de Características

Em ODE, cabe ao papel de Gerente de Conhecimento a gerência dos itens de conhecimento do ambiente. Dentre as funções atribuídas a esse papel estão a avaliação, adaptação, aprovação e categorização de itens de conhecimento a serem disponibilizados na memória organizacional, bem como a eliminação de itens de conhecimento obsoletos (NATALI, 2003).

Assim, no contexto da infra-estrutura de caracterização, também cabe ao Gerente de Conhecimento a responsabilidade de gerenciar o cadastro das características a serem utilizadas pela organização nas caracterizações de itens de software no ambiente.

O diagrama de casos de uso da Figura 4.2 mostra as funcionalidades oferecidas pela infra-estrutura para que o Gerente de Conhecimento possa cadastrar as características definidas pela organização. Essas funcionalidades são brevemente descritas a seguir.

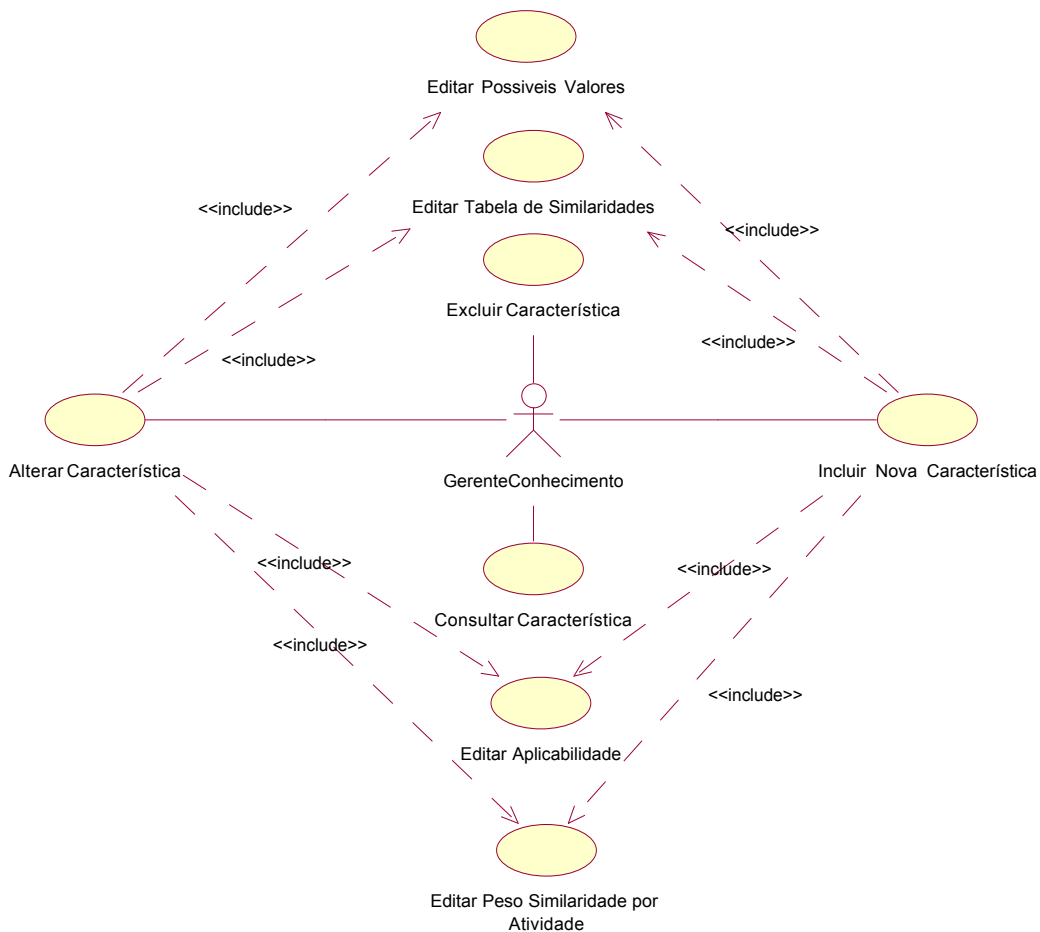


Figura 4.2 – Diagrama de Casos de Uso do Cadastro de Características.

- ***Incluir Nova Característica:*** permite ao gerente de conhecimento adicionar uma nova característica à infra-estrutura de caracterização de ODE. O gerente deve informar nome, descrição e tipo da característica, possíveis valores e a que tipos de itens de software ela se aplica, realizando o caso de uso Editar Aplicabilidade.
- ***Alterar Característica:*** permite que o gerente de conhecimento altere os dados de uma característica previamente cadastrada.
- ***Consultar Característica:*** permite que o gerente de conhecimento consulte os dados de uma característica previamente cadastrada.

- **Excluir Característica:** permite que o gerente de conhecimento exclua uma característica previamente cadastrada.
- **Editar Tabela de Similaridades:** permite ao gerente de conhecimento definir a similaridade entre os possíveis valores de uma característica.
- **Editar Aplicabilidade:** permite que o gerente de conhecimento defina a quais tipos de itens de software a característica se aplica.
- **Editar Pesos para Similaridade por Atividade:** permite que o gerente de conhecimento defina qual será o peso para o cálculo da similaridade entre dois itens sob a ótica de uma determinada atividade.
- **Editar Possíveis Valores:** permite que o gerente de conhecimento defina os possíveis valores que uma característica pode assumir.

A Figura 4.3 mostra a janela principal do Cadastro de Características de ODE.

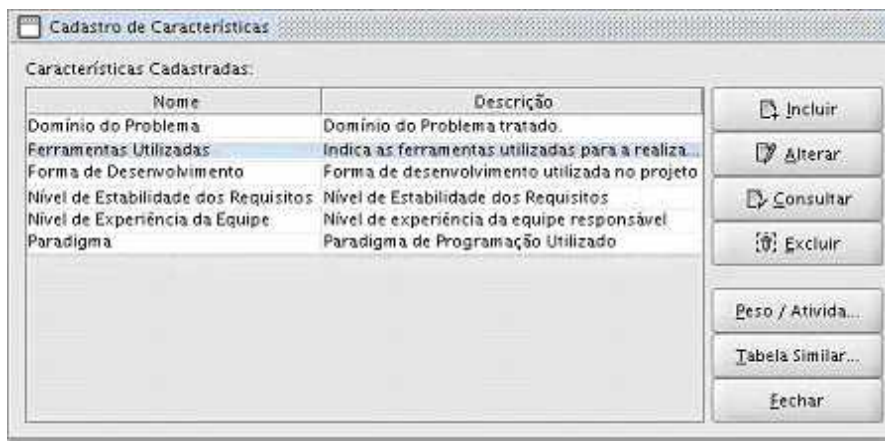


Figura 4.3 – Janela Principal do Subsistema Cadastro de Característica.

Um dos objetivos que nortearam a definição da infra-estrutura de caracterização de ODE foi permitir que a organização defina as características que deseja utilizar para caracterizar cada tipo de item de software. Logo, a primeira atividade a ser desenvolvida pela organização para fazer uso da infra-estrutura de caracterização de ODE será cadastrar as características aplicáveis a cada um dos tipos de itens de software caracterizáveis.

Para tratar os casos de uso descritos anteriormente, foram consideradas no subsistema *Cadastro de Característica* as classes mostradas no diagrama da Figura 4.4.

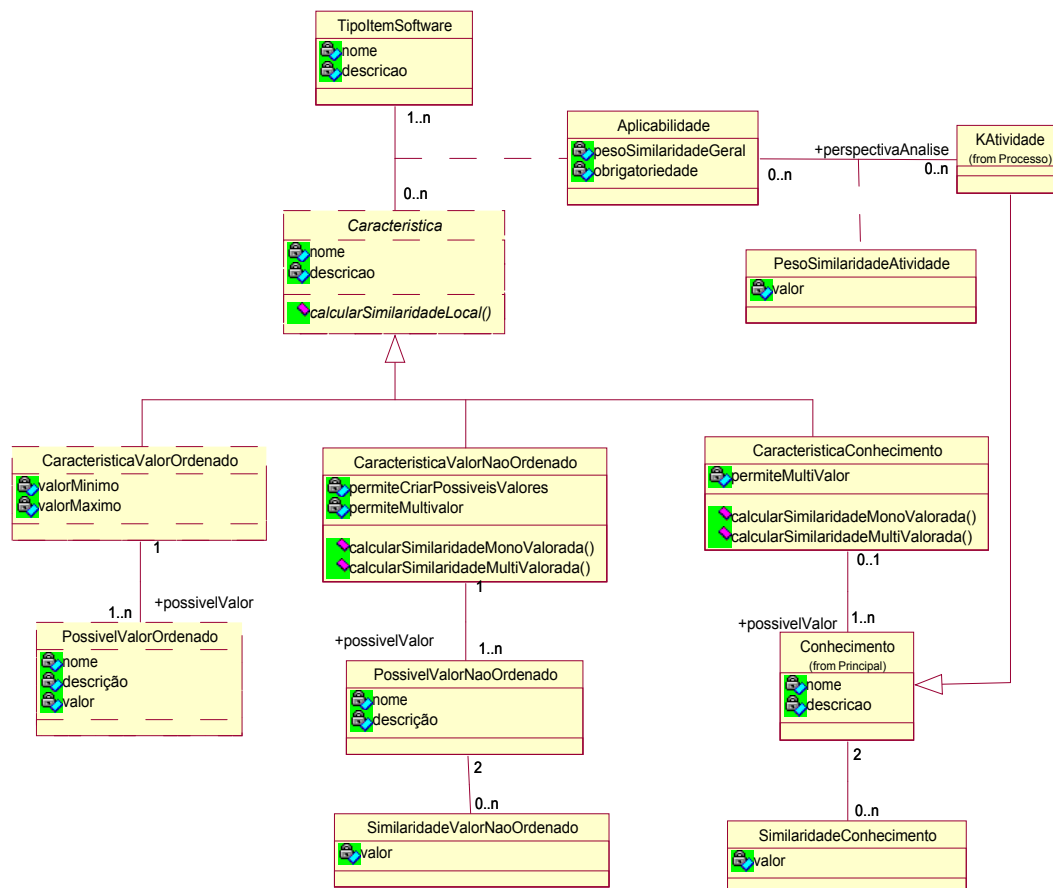


Figura 4.4 – Diagrama de Classes do subsistema Cadastro de Característica.

Como mostra a Figura 4.4, em ODE as características podem ser de três tipos, a saber:

- *Característica de Valor Ordenado*: é aquela cujos possíveis valores têm uma ordem intrínseca estabelecida entre eles, tal como complexidade do projeto (alta, média ou baixa). Para uma característica desse tipo, devem ser informados os possíveis valores que a característica pode assumir (instâncias da classe *PossivelValorOrdenado*). Cada *PossivelValorOrdenado* tem, além de um nome e uma descrição, um valor numérico que é utilizado para definir o grau de similaridade entre dois possíveis valores ordenados. Na *CaracteristicaValorOrdenado* são armazenados, ainda, o maior e o

menor valor numérico dentre os atribuídos a seus possíveis valores. Esses valores são utilizados para o cálculo do grau de similaridade entre dois possíveis valores e, portanto, para características deste tipo não há necessidade de se editarem tabelas de similaridade. A Figura 4.5 mostra a interface de cadastro de uma característica desse tipo.

Dados de Característica

Nome:

Descrição:

Tipo

☒ Valor Ordenado
☐ Valor Não Ordenado
☐ Conhecimento

Comportamento em Caracterizações

☐ Permite atribuição de multi-valor
☐ Permite criação de Possível Valor

Aplicabilidades | **Possíveis Valores**

Valores Cadastrados:

Nome	Valor
Alto	10.0
Médio	6.0
Baixo	4.0
Muito Baixo	1.0

Figura 4.5 –Cadastro de Característica de Valor Ordenado.

- Característica de Valor Não Ordenado: é aquela cujos possíveis valores não têm uma ordem pré-estabelecida entre eles, tal como a forma de desenvolvimento (desenvolvimento interno, desenvolvimento totalmente terceirizado, desenvolvimento com equipe mista). Para características desse tipo, devem ser informados, além dos possíveis valores (instâncias da classe *PossivelValorNaoOrdenado*), dois indicadores, o primeiro apontando se, no contexto de uma caracterização, essa característica pode ter atribuídos a ela mais de um valor (característica multi-valorada) e o segundo apontando se deve ser permitido atribuir a essa característica um valor não cadastrado previamente. Para se definir os graus de similaridade existentes entre os possíveis

valores de uma característica de valor não ordenado, deve-se editar uma tabela de similaridades, realizando o caso de uso *Editar Tabela de Similaridades*. O grau de similaridade entre dois possíveis valores não ordenados é registrado por uma instância da classe *SimilaridadeValorNaoOrdenado*. A Figura 4.6 mostra o cadastro de uma característica desse tipo.

A interface 'Dados de Característica' possui os seguintes campos e elementos:

- Nome:** Campo de texto com o valor 'Natureza do Projeto'.
- Descrição:** Área de texto com o conteúdo 'Indica se o projeto em questão é um projeto de novo desenvolvimento, man'.
- Tipo:** Grupo de botões de opção com as opções:
 - ☐ Valor Ordenado
 - ☒ Valor Não Ordenado
 - ☐ Conhecimento
- Comportamento em Caracterizações:** Grupo de botões de opção com as opções:
 - ☐ Permite atribuição de multi-valor
 - ☒ Permite criação de Possível Valor
- Aplicabilidades / Possíveis Valores:** Aba selecionada, contendo:
 - Tabela de Aplicabilidades:**

Tipo de Objeto	Obrigatória?	Peso Similaridade
Projeto	Sim	5.0
 - Botões de Ação:** Incluir, Alterar, Consultar, Excluir.
- Botões de Finalização:** OK e Fechar.

Figura 4.6 –Cadastro de Característica de Valor Não Ordenado.

- Característica de Conhecimento: é uma característica que tem como possíveis valores objetos de conhecimento, isto é, instâncias de conceitos das ontologias de ODE. Por exemplo, a ontologia de processo de software adotada em ODE (BERTOLLO, 2006) define o conceito Paradigma. Assim, a característica de conhecimento que considera paradigmas de desenvolvimento tem como possíveis valores instâncias da classe de ODE que trata esse conceito (classe *KParadigma*). Para características desse tipo,

devem ser informados a ontologia e o conceito dessa ontologia que será tratado pela característica. Assim, a característica terá, respectivamente, como nome e descrição, o nome e a descrição do conceito informado. Deve ser informado, ainda, no contexto de uma caracterização, se essa característica pode ter atribuídos a ela mais de um valor (característica multivalorada). De maneira análoga às características de valor não ordenado, é necessário editar uma tabela de similaridades entre os possíveis valores da característica, realizando o caso de uso *Editar Tabela de Similaridades*. O grau de similaridade entre duas instâncias de *Conhecimento* é registrado como uma instância da classe *SimilaridadeConhecimento*. A Figura 4.7 mostra o cadastro de uma característica desse tipo.

A janela 'Dados de Característica' possui os seguintes campos e elementos:

- Nome:** Campo de texto com o valor 'Paradigma'.
- Descrição:** Campo de texto com o valor 'Paradigma de Desenvolvimento Utilizado'.
- Tipo:** Grupo de botões de opção com as opções:
 - ☐ Valor Ordenado
 - ☐ Valor Não Ordenado
 - ☒ Conhecimento
- Comportamento em Caracterizações:** Grupo de botões de opção com as opções:
 - ☒ Permite atribuição de multi-valor
 - ☐ Permite criação de Possível Valor
- Aplicabilidades / Possíveis Valores:** Aba selecionada para 'Aplicabilidades'.

Tipo de Objeto	Obrigatória?	Peso Similaridade
Módulo	Sim	5.0
Projeto	Sim	5.0

 Botões de ação: Incluir, Alterar, Consultar, Excluir.
- Botões de Controle:** OK e Fechar.

Figura 4.7 –Cadastro de Característica de Conhecimento.

A Figura 4.8 mostra a interface que permite a edição de tabelas de similaridades entre os possíveis valores de uma *Característica de Valor Não Ordenado* ou de uma *Característica de Conhecimento*. Vale lembrar que não é possível editar tabelas de similaridades entre os possíveis valores de uma *Característica de Valor Ordenado*, pois grau de similaridade local entre dois possíveis valores ordenados é calculado com base nos valores numéricos a eles atribuídos. A forma de determinação dos graus de similaridade é assunto da próxima seção.

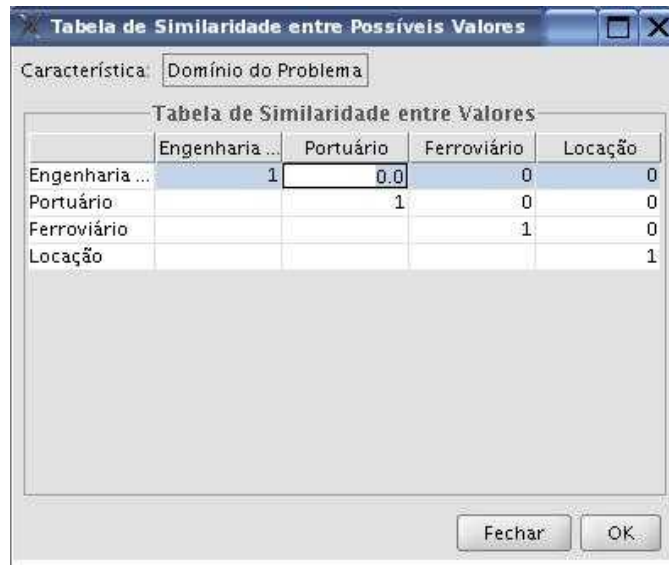


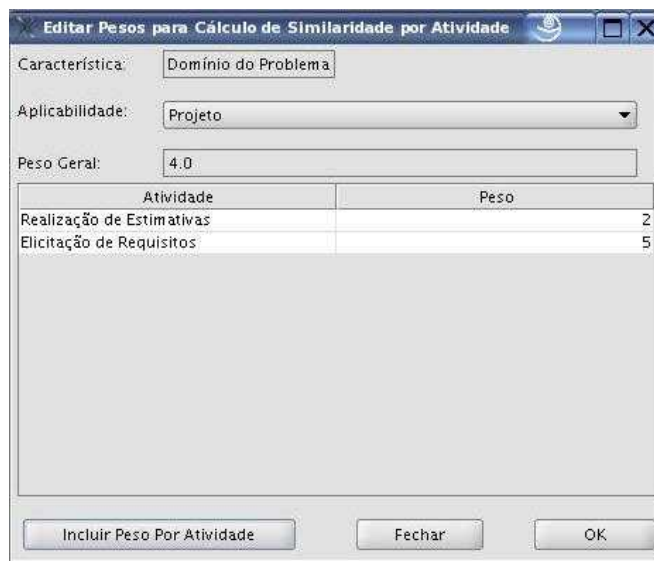
Figura 4.8 – Janela de Edição de Tabelas de Similaridades

Antes da edição da tabela de similaridades entre os possíveis valores de uma característica, por padrão, é atribuído grau de similaridade 0 (zero), entre possíveis valores diferentes e, obviamente, grau de similaridade 1 (um) entre possíveis valores iguais. Ao editar uma tabela de similaridades, devem ser informados valores entre 0 e 1 para os graus de similaridades entre possíveis valores diferentes.

Como diferentes tipos de itens de software são classificados por diferentes características, é importante indicar a aplicabilidade de cada característica a um tipo de item. Além de indicar que uma característica é aplicável a determinado tipo de item, a *Aplicabilidade* define, ainda, se caracterizações de itens de software do tipo em questão deverão, obrigatoriamente, incluir tal característica. Por exemplo, a caracterização de projetos (um tipo de item de software) tem de obrigatoriamente incluir a característica de conhecimento *Paradigma*. Assim, todos os projetos

são obrigados a informar o paradigma adotado. Em *Aplicabilidade* é registrado, ainda, o peso a ser atribuído a essa característica quando do cálculo da similaridade geral entre itens de software de um determinado tipo. Vale ressaltar, ainda, que uma característica deve, obrigatoriamente, ser aplicável a pelo menos um tipo de item de software. Na Figura 4.7 vê-se a aba destinada à edição de *Aplicabilidades* selecionada.

Finalmente, é interessante observar que, dependendo da atividade do desenvolvimento, a similaridade pode ter de ser computada de maneira diferente. Por exemplo, o nível de experiência da equipe com o processo de software adotado (uma característica) é mais importante para se calcular a similaridade entre projetos sob a ótica da atividade de planejamento do que sob a ótica da atividade de análise de requisitos. Na primeira, está-se interessado em recuperar, dentre outros, estimativas e riscos, e o nível de experiência da equipe com o processo de software adotado tem um grande impacto sobre os itens que se deseja recuperar. Já na análise de requisitos, está-se interessado em recuperar modelos de requisitos e, para tal, o nível de experiência da equipe com o processo de software adotado tem menor impacto. Para tratar esse aspecto, é permitido informar o peso de similaridade por atividade. A Figura 4.9 mostra a definição dos pesos para cálculo de similaridade de acordo com a atividade.



A janela de diálogo intitulada "Editar Pesos para Cálculo de Similaridade por Atividade" apresenta os seguintes campos e elementos:

- Característica: Domínio do Problema
- Aplicabilidade: Projeto (menu suspenso)
- Peso Geral: 4.0
- Tabela de pesos por atividade:

Atividade	Peso
Realização de Estimativas	2
Elicitação de Requisitos	5

Na base da janela, há três botões: "Incluir Peso Por Atividade", "Fechar" e "OK".

Figura 4.9 – Janela de Edição de Peso para Similaridade por Atividade

4.2.2 Caracterização e Busca

Em ODE, a função de caracterizar os itens de software é uma função dos desenvolvedores. Em ODE, o ator *Desenvolvedor* representa todos os usuários do sistema: analistas, projetistas, programadores, entre outros.

Visando a facilitar a reutilização de conhecimento, ODE provê aos *Desenvolvedores* uma funcionalidade que viabiliza a busca por itens de software similares.

A Figura 4.10 mostra, por meio de um digrama de casos de uso, as funcionalidades que ODE provê aos *Desenvolvedores* para permitir a caracterização de itens de software e a busca por itens similares. Essas funcionalidades são brevemente descritas a seguir.

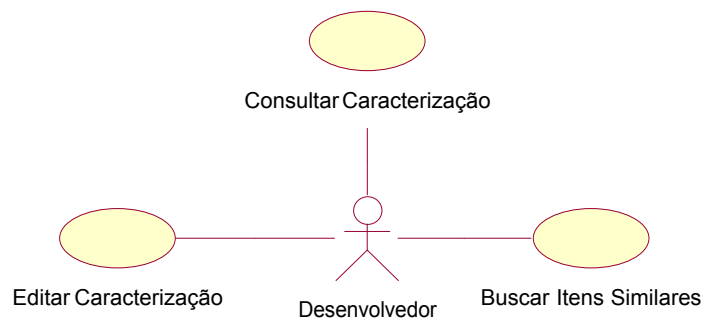


Figura 4.10 – Diagrama de Casos de Uso da Caracterização e Busca.

- **Editar Caracterização:** permite que o desenvolvedor caracterize um determinado item de software, a partir das características definidas para o tipo do item em questão.
- **Consultar Caracterização:** permite que o desenvolvedor consulte uma caracterização previamente efetuada.
- **Buscar Itens de Software Similares:** permite que o desenvolvedor encontre itens de software similares a um determinado item informado.

A Figura 4.11 mostra o diagrama de classes que trata da caracterização e busca de itens similares. Conforme discutido anteriormente, um dos principais requisitos da infra-estrutura de caracterização desenvolvida neste trabalho é que ela seja genérica a ponto de permitir a caracterização de diferentes itens de software em ODE. Assim, quando um item de software precisar ser caracterizado, ele deverá herdar da classe *ObjetoCaracterizavel*. Projetos e módulos,

por exemplo, precisam ser caracterizados e, portanto, têm de ser definidos como subclasses dessa classe e, assim, poderão ter associados a eles uma *Caracterizacao*.

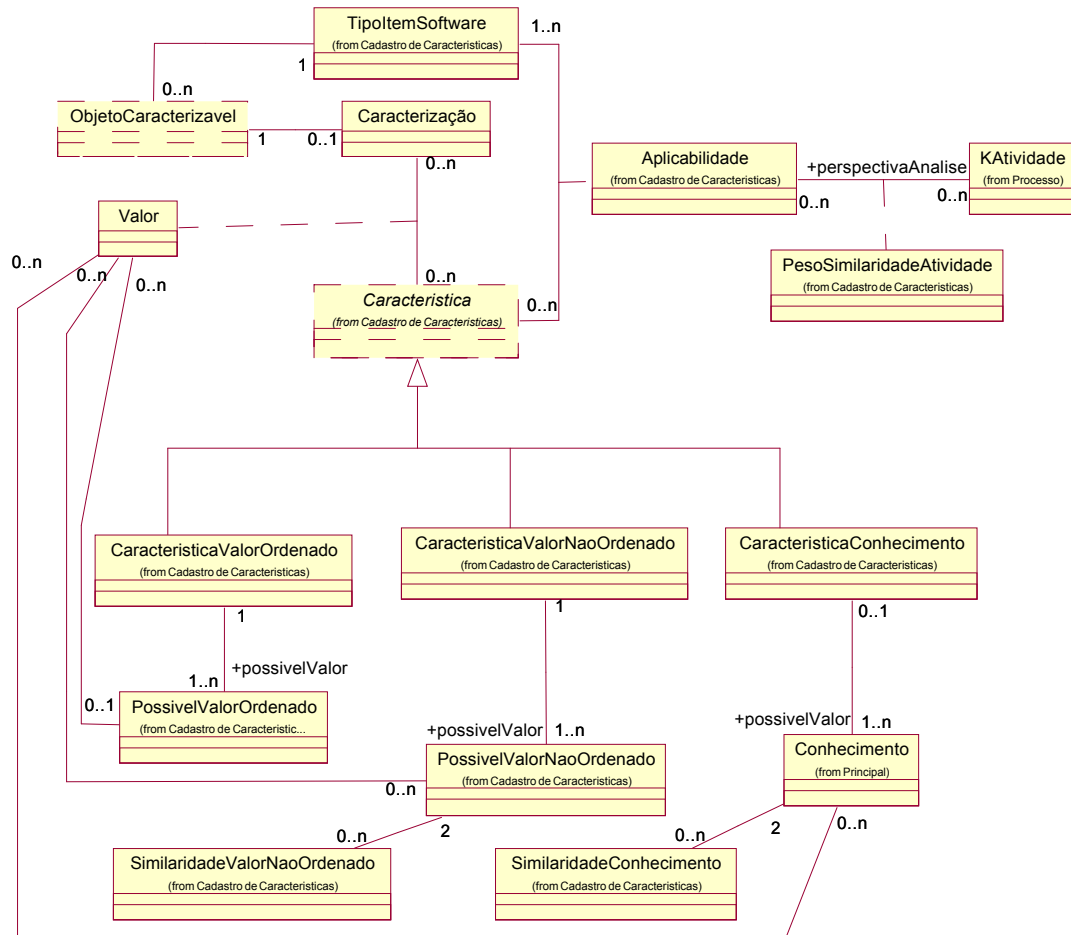


Figura 4.11 – Diagrama de Classes do subsistema Caracterização e Busca.

Uma caracterização de um objeto define os valores deste objeto para as características aplicáveis ao tipo do mesmo *TipItemSoftware*. Assim, caso o objeto seja, por exemplo, um projeto, sua caracterização contemplará apenas características aplicáveis a projetos.

Neste trabalho a infra-estrutura construída foi instanciada para permitir a caracterização de projetos em ODE. A Figura 4.12 mostra a interface de caracterização de projetos em ODE.



Figura 4.12 –Caracterização de Projetos em ODE.

Inicialmente, são apresentadas todas as características cadastradas aplicáveis a projetos, divididas em dois grupos, a saber: obrigatórias e opcionais. Ao selecionar uma característica, são exibidos todos os possíveis valores que essa característica pode assumir, para que o gerente de projeto selecione qual(is) se aplica(m) ao projeto em questão. O gerente de projeto deve caracterizar o projeto segundo todas as características obrigatórias, podendo informar, ainda, as características opcionais que desejar.

Caso o projeto corrente já tenha sido caracterizado, a caracterização do projeto é recuperada e apresentada, mostrando, para cada característica, os valores previamente selecionados para o projeto. Assim é possível, sempre que necessário, que o gerente de projeto altere a caracterização de um projeto.

Após realizada a caracterização de um item de software em ODE, um desenvolvedor pode, a qualquer momento, buscar itens de software similares ao item em questão. Para que dois itens de software possam ser considerados similares, ambos devem ser de um mesmo tipo.

Para determinar o grau de similaridade global (GSG) entre dois itens de software caracterizados (*Item1* e *Item2*) é necessário calcular, primeiro, os graus de similaridades locais

(GSL) de cada uma das características dos itens de software em questão. Esse cálculo varia de acordo com o tipo da característica, a saber:

(i) Para características de valor ordenado, o GSL é dado por:

$$GSL = 1 - |(\text{valorItem1} - \text{valorItem2}) / (\text{valorMaximo} - \text{valorMinimo})|$$

onde *valorItemN* é o valor do *PossivelValorOrdenado* na caracterização do item de software *N* referente à característica de valor ordenado considerada e *valorMaximo* e *valorMinimo* correspondem aos atributos homônimos da mesma característica, ou seja, respectivamente, o maior valor e o menor valor que essa característica pode assumir;

(ii) Para características de valor não ordenado ou características de conhecimento que não aceitem vários valores (conforme indicado pelo atributo *permiteMultiValor*), o GSL dos itens de software em questão é obtido na correspondente tabela de similaridades (respectivamente, os atributos *valor* das classes *SimilaridadeValorNaoOrdenado* e *SimilaridadeConhecimento*). Caso a tabela de similaridades da característica em questão não tenha sido editada, o GSL será considerado 0 caso os valores sejam diferentes, ou 1 caso os valores sejam iguais;

(iii) Para características de valor não ordenado ou características de conhecimento que aceitem vários valores, o grau de similaridade local é dado por:

$$GSL = \text{numInterseção} / \text{numUnião}$$

onde *numInterseção* é o número de elementos da interseção dos conjuntos de valores associados às caracterizações dos itens de software em questão e *numUnião* é o número de elementos da união desses mesmos conjuntos.

Para o cálculo do grau de similaridade global entre dois itens de software caracterizados são consideradas as similaridades locais segundo todas as características que tenham sido utilizadas nas caracterizações dos dois itens.

O grau de similaridade global (GSG) entre dois itens de software caracterizados baseia-se na técnica da distância geométrica ponderada, também conhecida como “vizinho mais próximo ponderado” (WANGENHEIM et al. , 2003), sendo dado por:

$$GSG = (\sum GSL_i * P_i) / \sum P_i$$

onde *GSL_i* é o grau de similaridade local da característica *i* e *P_i* é seu peso no cálculo. O peso de cada característica no cálculo da similaridade global é definido de acordo com a ótica sob a qual a busca está sendo realizada: geral (atributo *pesoSimilaridadeGeral* de *Aplicabilidade*) ou por uma atividade específica (atributo *valor* de *PesoSimilaridadeAtividade*).

A forma de cálculo de similaridade descrita acima é utilizada pela infra-estrutura na busca de itens de software similares. Na instanciação da infra-estrutura para itens do tipo projeto, essa funcionalidade é utilizada para apoiar a busca por projetos similares, como ilustra a Figura 4.13.

Característica	Peso	Imprescindível?
Paradigma	5	☐
Forma de Desenvolvimento	2	☐
Domínio do problema	4	☐
Nível de experiência da equipe	2	☐
Ferramentas Utilizadas	4	☐
Complexidade	4	☐
Nível de Estabilidade dos Requisitos	3	☐

Figura 4.13 –Busca por Projetos Similares

Essa funcionalidade só fica disponível quando há um projeto aberto em ODE. Quando o desenvolvedor solicita a busca por projetos similares, são exibidas as características utilizadas na caracterização do projeto em questão, seguidas pelos seus respectivos pesos definidos para cálculo de similaridade. Características aplicáveis a projetos mas que não foram utilizadas na caracterização do projeto em questão não são consideradas para o cálculo de similaridade entre o projeto em questão e os demais projetos.

O desenvolvedor pode selecionar a perspectiva de análise (atividades) sob a qual a similaridade será calculada. É exibido, em frente a cada característica, seu respectivo peso sob a ótica selecionada. Para aquelas características que não tenham pesos definidos especificamente sob a ótica selecionada (atividade), o peso geral é utilizado.

Assim, os pesos apresentados são os praticados pela organização de acordo com a perspectiva selecionada. No entanto, caso o desenvolvedor deseje fazer uma busca utilizando

pesos diferentes dos praticados pela organização, permite-se que ele informe, para cada característica que desejar, os pesos que gostaria de utilizar.

Além disso, é dado ao desenvolvedor a possibilidade de indicar características que ele considera imprescindíveis na busca por projetos similares. Neste caso, só são considerados projetos que tenham 100% de similaridade local para as características consideradas imprescindíveis. Por exemplo, se o desenvolvedor considerar que o paradigma é uma característica imprescindível e um determinado projeto for 100% similar ao projeto em questão segundo todas as outras características, mas o grau de similaridade local (GSL) segundo a característica paradigma for inferior a 100%, este projeto será ignorado.

Finalmente, o desenvolvedor deve informar o grau de similaridade global (GSG) mínimo que deseja para que projetos sejam considerados similares ao projeto em questão. Assim, é calculada a similaridade entre o projeto em questão e cada um dos outros projetos caracterizados existentes na base de ODE. Para o cálculo do GSG, são consideradas apenas as características utilizadas nas caracterizações dos dois projetos (o projeto em questão e o testado). Seja, por exemplo, a característica “Domínio da Aplicação”, aplicável a projetos, mas não obrigatória. Assim, caso aconteça dessa característica ser utilizada na caracterização de um projeto e não ser utilizada na caracterização de um segundo projeto, a mesma não será considerada para efeito do cálculo do grau de similaridade entre esses dois projetos.

Ao fim da busca é exibida uma lista contendo todos os projetos cujos GSGs sejam maiores ou iguais ao mínimo solicitado. Em NARDI (2006) essa funcionalidade foi estendida, de forma a permitir que, após a recuperação dos projetos similares, se possa selecionar um dos projetos similares retornados e solicitar a exibição de seus requisitos ou modelos de objetos. Dessa forma, NARDI (2006) pretende promover o reúso de requisitos e modelos de objetos em ODE.

4.3 Definição de um Conjunto Inicial de Características Aplicáveis a Projetos em ODE

A infra-estrutura para caracterização proposta neste trabalho permite que cada organização defina quais características utilizará para caracterizar projetos. Porém, essa flexibilidade não elimina a necessidade de se definir um conjunto inicial de características a serem fornecidas para apoiar a caracterização de projetos em ODE.

Assim, visando a facilitar o trabalho que as organizações terão para definir as características a serem utilizadas nas caracterizações de seus projetos, foi feito neste trabalho um levantamento de características aplicáveis a projetos de software, gerando um conjunto inicial de características a serem distribuídas com ODE, de forma a servir como um ponto de partida para que cada organização defina as características que deseja utilizar.

Vale ressaltar que não foram despendidos esforços no sentido de determinar os pesos das características para efeito do cálculo de similaridade. Também não foi objeto de pesquisa a aplicabilidade dessas características de acordo com a atividade desenvolvida. O objetivo desse estudo foi apenas definir um conjunto inicial de características, de forma que cada organização o ajuste à sua realidade.

Deve-se ressaltar que o quesito similaridade entre projetos de *software* não tem sido assunto de estudos profundos, mesmo sendo freqüentemente utilizado em estimativas de esforço baseadas em analogias (IDRI et al., 2001). Assim, o conjunto inicial de características proposto foi baseado em alguns trabalhos da literatura que discutem o assunto, dentre eles: (IDRI et al., 2001), (MENZIES et al., 2000), (JONES, 2000), (KURTZ, 2001) e (BARCELLOS, 2003).

Segundo IDRI et al. (2001), a complexidade dos módulos e a experiência dos programadores são citadas como fatores capazes de descrever projetos de software.

MENZIES et al. (2000) destacam a necessidade de considerar a coesão da equipe de desenvolvimento, flexibilidade de desenvolvimento, análise arquitetural, resolução de riscos, grau de maturidade do processo, atributos do produto (tais como confiabilidade requerida, complexidade, nível de reúso e requisitos de documentação), atributos da plataforma (tais como restrição de tempo de execução e volatilidade da plataforma), atributos do projeto (tais como distribuição do desenvolvimento, uso de ferramentas de software, experiência com a linguagem,

ferramentas utilizadas e pressões de cronograma) e atributos da equipe (tais como experiência e capacidade dos analistas e dos programadores e experiência com a plataforma).

JONES (2000), por sua vez, considera haver sete tipos básicos de informação para caracterizar projetos, a saber: o país, a cidade para o qual o projeto se destina, o domínio da aplicação (siderurgia, extração de óleo, comunicação etc), a natureza do projeto (novo desenvolvimento, adição de funcionalidades, reparo de funcionalidades, melhoria de desempenho, migração de plataforma etc), o escopo do projeto (módulo, protótipo, subrotina etc), a classe do projeto (aplicação para uso corporativo desenvolvida com recursos internos, aplicação para uso corporativo desenvolvida com recursos externos, aplicação *shareware* ou *freeware*, aplicação para uso externo via Internet etc) e o tipo do projeto (aplicação interativa, aplicação *batch*, não procedural etc).

KURTZ (2001) acrescenta, ainda, a importância de se considerar características de investimentos e da organização que atua no desenvolvimento do projeto.

Os modelos paramétricos para estimativas de tamanho de software também são apontados como uma boa fonte de inspiração para a definição das características a serem utilizadas para caracterizar projetos de software. O método da Análise de Pontos de Função, por exemplo, define catorze características gerais que devem ser utilizadas para determinar o fator de ajuste da contagem. IDRI et al. (2001) destacam, ainda, os dezessete atributos utilizados pelos métodos COCOMO e COCOMO II para descrever um projeto.

BARCELLOS (2003) definiu critérios de caracterização de projetos para a Estação TABA (LIMA, 2004). Esses critérios foram escolhidos com base na literatura, principalmente na proposta de JONES (2000), e levando-se em consideração os critérios que já estavam presentes na Estação TABA quando o trabalho foi realizado.

A Tabela 4.1 apresenta as características selecionadas para compor o conjunto inicial de características aplicáveis à caracterização de projetos de software em ODE. A definição desse conjunto foi feita com base em três tipos principais de fontes:

- Trabalhos Científicos: (JONES, 2000), (BARCELLOS, 2003), (IDRI et al., 2001) e (MENZIES et al., 2000);
- Livros de Engenharia de Software: (PRESSMAN, 2005), (PFLEEGER, 2001) e (SOMMERVILLE, 2003);

- Modelos Paramétricos para Estimativas: Análise por Pontos de Função (GARMUS et al., 2001) e COCOMOII (BOEHM et al., 2000).

Visando a uma melhor apresentação, as características foram organizadas em cinco grupos, a saber:

- Dados do Projeto: reúne características inerentes ao projeto, como a natureza e o escopo;
- Dados da Equipe: reúne as características referentes à equipe responsável pelo desenvolvimento do projeto;
- Ferramental: apresenta as características relacionadas a ferramentas, processos, paradigmas e métodos e técnicas utilizados;
- Cuidados Extras: agrupa as características que visam a identificar restrições e necessidades especiais impostas ao projeto;
- Fatores Organizacionais: reúne características que visam a retratar a organização na qual o projeto foi desenvolvido e suas restrições.

Tabela 4.1 – Conjunto Inicial de Características Aplicáveis a Projetos em ODE.

Dados do Projeto
Tipo de software (Sistema de Informação, Sistema para <i>Web</i> etc)
Natureza do Projeto (Novo desenvolvimento, manutenção corretiva, evolução etc)
Finalidade (Uso interno, Desenvolvimento sob encomenda, software de prateleira)
Escopo do Projeto (sistema, módulo, protótipo, apenas algumas fases do processo)
Domínio da aplicação (Siderurgia, Ferrovia, Governamental, Portuário etc)
Tarefas (Faturamento, Contabilidade, Venda, Produção etc)
Análise Arquitetural (Sistema Batch, Cliente-Servidor, Cliente-Servidor em 3 camadas etc)
Complexidade do projeto
Tamanho do projeto
Grau de estabilidade dos requisitos

Tabela 4.1 – Conjunto Inicial de Características Aplicáveis a Projetos em ODE (Continuação).

Dados da Equipe
Coesão da equipe de desenvolvimento
Forma de desenvolvimento (equipe da própria empresa, equipe terceirizada, equipe parcialmente terceirizada)
Distribuição geográfica da equipe
Nível de experiência dos gerentes do projeto
Nível de experiência da equipe
Nível de experiência da equipe no domínio da aplicação
Nível de experiência da equipe nas tecnologias utilizadas
Nível de experiência da equipe em aplicações de mesmo tamanho/complexidade
Nível de experiência da equipe com o processo
Ferramental
Processo utilizado
Maturidade do processo
Paradigma utilizado (Paradigma Estruturado, Paradigma OO etc)
Procedimentos (Métodos, técnicas, roteiros para elaboração de documentos etc)
Tecnologias utilizadas: Tipo de Banco de Dados (Ex.: Relacional), Linguagem de Programação (Ex.: Java) etc.
Ferramentas utilizadas: Ferramentas CASE (Ex.: Rose), Banco de Dados (Ex.: Oracle), Ambiente de Programação (Ex.: NetBeans)
Uso de tecnologia inovadora
Cuidados extras
Restrições de usabilidade (facilidade de operação e instalação)
Restrição de desempenho (performance)
Restrições computacionais (ex. tem que rodar em uma máquina específica)
Restrição de segurança
Preocupação com manutenibilidade
Preocupação com reusabilidade
Necessidade de portabilidade

Tabela 4.1 – Conjunto Inicial de Características Aplicáveis a Projetos em ODE (Continuação).

Fatores Organizacionais
Experiência do cliente em apoiar o desenvolvimento de software e no domínio da aplicação a ser desenvolvida
Participação do usuário na especificação dos requisitos
Restrição de cronograma
Restrição de custos

4.4 Considerações Finais do Capítulo

Neste capítulo foi apresentada a infra-estrutura para caracterização de itens relacionados ao desenvolvimento de software de ODE. Essa infra-estrutura permite que sejam definidos quais tipos de itens de software serão caracterizados e, para cada um deles, quais características serão utilizadas na caracterização. A infra-estrutura conta, ainda, com uma máquina de busca por objetos similares configurável que permite, dentre outras coisas, definir o peso que cada característica terá no cálculo da similaridade.

Segundo JABLONSKI et al. (2001), a informação deve ser organizada de maneira que possa ser encontrada quando necessário, o que significa que existe a necessidade de uma estrutura para classificar as informações. NATALI (2003) ressalta a importância de se mostrar ao usuário que a memória organizacional possui informações úteis ao seu trabalho e que achá-las não será uma tarefa que consumirá muito tempo, caso contrário o usuário relutará em parar a sua atividade principal para buscar auxílio na memória organizacional.

Ao desenvolver a infra-estrutura apresentada, a intenção era prover em ODE a estrutura necessária para classificar os itens relativos ao desenvolvimento de software, fornecendo, ainda, uma maneira eficiente de se buscar itens úteis quando necessário. Mas, para que esses objetivos sejam alcançados, é necessário, ainda, realizar um estudo, para cada tipo de item a ser caracterizado em ODE, visando a definir quais são as características mais representativas de objetos de tal tipo. Neste trabalho, foram desenvolvidas, apenas, as funcionalidades de caracterização de projetos e de busca por projetos similares em ODE, instanciando a infra-estrutura proposta.

RAMESH (2002) observa a necessidade de associação dos fragmentos de conhecimento relacionados para facilitar a transferência e a reutilização do conhecimento. No contexto de ODE, acredita-se que as funcionalidades de caracterização de projetos e de busca por projetos similares facilitarão a reutilização de conhecimento produzido em projetos anteriores (como artefatos por exemplo). Nesse contexto, vale ressaltar que, em (NARDI, 2006) são apresentadas iniciativas visando a promover o reúso de requisitos e modelos de objeto tendo como base a recuperação de projetos similares desenvolvida neste trabalho.

Prover formas de recuperação de itens de software similares aparece como uma boa estratégia para a promoção da reutilização de conhecimento, porém não é suficiente. Há situações em que é mais importante conhecer as justificativas e razões existentes por trás de uma decisão tomada em um projeto anterior do que conhecer o produto gerado por tal decisão. Assim, é evidente a importância de se registrar não apenas os artefatos gerados em cada atividade do processo de software, mas também as razões por trás de cada decisão tomada, incluindo justificativas, alternativas consideradas e os argumentos que conduziram à decisão.

Assim, foi desenvolvida neste trabalho uma infra-estrutura que visa a apoiar a construção cooperativa de artefatos de software em ODE e a capturar o raciocínio seguido durante a construção dos artefatos. Essa infra-estrutura é apresentada no próximo capítulo.

Capítulo 5

Infra-estrutura de Apoio à Decisão em Grupo em ODE

As atividades do processo de software são reconhecidamente cooperativas. Durante um projeto de software, diversos profissionais trabalham em conjunto com o objetivo comum de desenvolver um produto de software de qualidade, que atenda às necessidades do cliente, obedecendo a restrições de custo e prazo. Equipes de desenvolvimento de software tendem a ser mais produtivas em ambientes onde há colaboração mútua e troca de idéias entre seus membros.

Em especial, os gerentes de projeto são responsáveis pela tomada de decisões que podem levar o projeto ao sucesso ou ao fracasso. Na literatura, a experiência é unanimemente apontada como uma das características desejáveis em um bom gerente de projetos de software. Porém, a Engenharia de Software é uma disciplina extremamente complexa, sujeita a constantes mudanças de tecnologia. Cada projeto de software é desenvolvido em um contexto específico, obedecendo a restrições específicas e utilizando as mais variadas tecnologias. Assim, por mais experiente que seja o gerente do projeto, muitos ganhos podem ser obtidos através da troca de experiências durante a realização das atividades de gerência.

A utilização de uma base de dados históricos, contendo dados confiáveis sobre projetos anteriores, aparece como outra boa alternativa para que o gerente de projetos não fique limitado às suas próprias experiências.

Para tomar uma decisão em um projeto, muitas vezes, é mais importante conhecer as justificativas e razões existentes por trás de uma decisão tomada em um projeto anterior do que conhecer a decisão em si e os produtos dela decorrentes. Assim, é evidente a importância de se registrar não apenas os artefatos gerados em cada atividade do processo de software, mas também as razões por trás de cada decisão tomada, incluindo justificativas, outras alternativas levadas em consideração e os argumentos que conduziram à decisão.

Com os objetivos principais de apoiar a construção cooperativa de artefatos de software e de capturar o raciocínio seguido durante a construção dos mesmos, foi desenvolvida neste trabalho uma infra-estrutura de apoio à decisão em grupo integrada ao ambiente ODE, apresentada neste capítulo.

A seção 5.1 apresenta a abordagem para apoiar a construção cooperativa de artefatos de software, proposta tomando por base a técnica Delphi. Na seção 5.2 é apresentada a infra-estrutura de apoio à decisão em grupo desenvolvida neste trabalho. Na seção 5.3 é apresentada uma instanciação dessa infra-estrutura para apoiar a elaboração cooperativa de planos de risco, construída como estudo de caso. A seção 5.4 trata da captura do raciocínio seguido pelos participantes da reunião durante a construção cooperativa dos artefatos. Na seção 5.5 são apresentadas as considerações finais deste capítulo.

5.1 Adaptação da Técnica Delphi para Apoiar a Construção Cooperativa de Artefatos de Software

Diversos autores (PRESSMAN, 2002) (PFLEEGER, 2001) (JØRGENSEN, 2002) apontam a combinação de estimativas de diferentes especialistas utilizando diferentes abordagens como uma boa prática para a geração de estimativas confiáveis. Nesse contexto, a técnica Delphi (PFLEEGER, 2001) aparece como uma boa alternativa para a realização de estimativas, pois utilizando-se essa técnica, as estimativas de diversos especialistas são combinadas visando à geração de uma estimativa consensual, sendo que cada participante é livre para utilizar a abordagem que lhe for mais conveniente para a elaboração de sua proposta. Além disso, a cada rodada da reunião, os especialistas têm a oportunidade de rever suas estimativas com base na troca de idéias promovida pelo moderador.

Mas o uso da técnica Delphi não se restringe às estimativas. Essa técnica pode ser utilizada em qualquer atividade que tenha como objetivo a construção cooperativa de um artefato. Dentre as atividades do processo de software que podem ser beneficiadas com a utilização da técnica Delphi, podem ser citadas, além das estimativas, a análise de riscos, a alocação de recursos, análise e projeto (*design*), dentre outras. Porém, vale ressaltar que, como o objetivo de uma reunião de Delphi é chegar a um consenso sobre um artefato a ser gerado, a condução da reunião ao consenso pode ser mais difícil em algumas atividades devido à grande quantidade de

alternativas existentes para o artefato a ser gerado e à complexidade envolvida para realizar a combinação das alternativas apresentadas para gerar um artefato consensual.

Como dito anteriormente, a técnica Delphi foi originalmente concebida para apoiar a realização de estimativas. Essa técnica consiste na realização de uma reunião com vários participantes, na qual um deles exerce o papel de moderador e tem a responsabilidade de conduzir a reunião. A reunião é composta por várias rodadas, sendo que, a cada rodada, todos os participantes realizam suas estimativas e as entregam ao moderador. O moderador tem a função de compilar as estimativas dos participantes, apresentando o resultado dessa compilação e conduzindo o processo de decisão entre finalizar a reunião elegendo uma estimativa consensual ou realizar uma nova rodada de estimativas para que os participantes revejam suas opiniões.

Antes da realização de uma reunião, há uma etapa de preparação que é de fundamental importância para que bons resultados sejam alcançados. Durante a preparação, são reunidas informações que devem ser levadas em consideração durante a realização das estimativas para que as mesmas sejam apresentadas aos participantes na abertura da reunião. A intenção é contextualizar os participantes e fazer eventuais recomendações, a fim de que todos conheçam os principais dados envolvidos na atividade em questão.

Tendo em vista a ampla aplicabilidade e o caráter cooperativo da técnica Delphi, neste trabalho é proposta uma abordagem que generaliza essa técnica de forma a apoiar a realização cooperativa de qualquer atividade do processo de software. A abordagem proposta define a realização de uma reunião para construção cooperativa de artefatos de software através de um fluxo de trabalho iterativo composto por sete etapas, a saber:

1. Identificação da Necessidade de Realizar uma Atividade de Forma Cooperativa: o gerente de projeto identifica uma atividade que pode ser realizada de forma cooperativa e propõe uma reunião de decisão em grupo para a realização da mesma.
2. Planejamento Inicial da Reunião: o gerente de projeto define o moderador, os participantes, o artefato a ser discutido e o tipo da reunião, e prepara materiais / recomendações que julga importante estarem disponíveis para os participantes.
3. Planejamento de Rodada: o moderador estabelece as datas de início e término de uma rodada e notifica os participantes da mesma.

4. Elaboração de Propostas: os participantes elaboram suas propostas para o artefato, explicando os motivos que os levaram a tais propostas, e as submetem para apreciação do moderador.
5. Elaboração de Proposta de Consenso: o moderador, de posse das propostas dos participantes, procura elaborar uma proposta consensual que contemple as principais considerações de cada participante. Essa proposta é submetida para apreciação dos participantes. Caso considere as propostas dos participantes divergentes a ponto de não permitirem a elaboração de uma proposta de consenso, o moderador pode apresentar novas informações e recomendações aos participantes e iniciar uma nova rodada de discussões, retomando o processo a partir do passo 3
6. Discussão sobre a Proposta de Consenso: os participantes emitem opiniões sobre a proposta de consenso, justificando seus pontos de vista.
7. Avaliação da Proposta de Consenso: tomando por base as opiniões dos participantes, o moderador avalia se a proposta de consenso pode ser aprovada, se precisa de algumas alterações e pode ser aprovada, ou se uma nova rodada de discussões é necessária, retomando o processo a partir do passo 3. Caso acredite ser inviável chegar a um consenso entre os participantes, o moderador pode, ainda, finalizar a reunião sem que uma proposta de consenso seja aprovada.

Segundo a abordagem proposta, uma reunião pode ser de dois tipos, a saber: (i) aberta, na qual, a cada rodada, os participantes têm acesso, de forma anônima, às propostas apresentadas por todos os outros participantes na rodada anterior e a um relatório sumário dessas propostas, ou (ii) fechada, na qual, cada participante só tem acesso à sua própria proposta da rodada anterior e ao relatório sumário das propostas apresentadas na rodada anterior.

Tomando por base a abordagem para apoiar a construção cooperativa de artefatos descrita, foi implementada e integrada ao ambiente uma infra-estrutura de apoio à decisão em grupo. Essa infra-estrutura é apresentada na próxima seção.

5.2 A Infra-estrutura de Apoio à Decisão em Grupo Desenvolvida

Conforme previamente citado, a infra-estrutura desenvolvida procura apoiar a abordagem definida tomando por base a técnica Delphi apresentada na seção anterior. Dessa forma, como mostra a Figura 5.1, a infra-estrutura é composta por três casos de uso principais, a saber: (i) Preparar Reunião, que permite a preparação e criação de uma nova reunião, bem como o cancelamento da mesma, (ii) Controlar Reunião, que provê ao moderador as ferramentas necessárias à condução da reunião, e (iii) Participar da Reunião, que visa a fornecer ao moderador e aos participantes todas as informações e funcionalidades necessárias para participarem da reunião e elaborarem suas propostas.

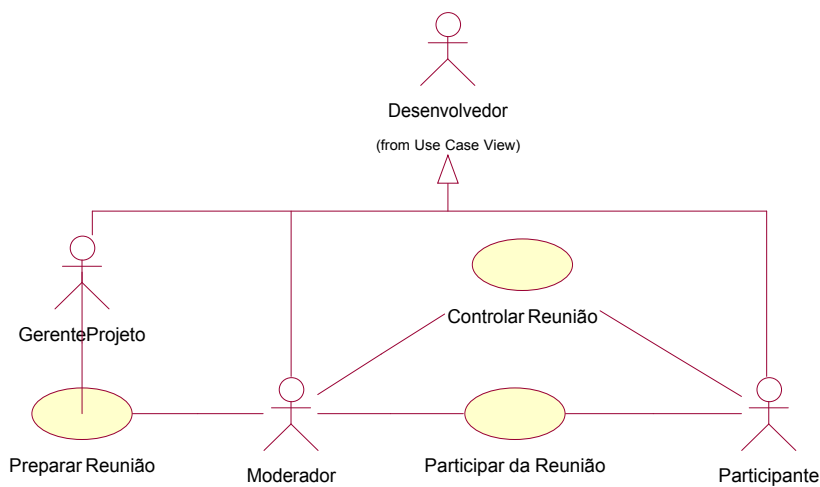


Figura 5.1 – Funcionalidades Principais da Infra-estrutura.

As funcionalidades providas por cada um desses casos de uso bem como os respectivos modelos de classes envolvidos em cada um deles são apresentadas a seguir.

5.2.1 Preparar Reunião

Cabe ao gerente do projeto decidir pela realização de uma reunião de decisão em grupo para elaborar um determinado artefato do projeto. Assim, o gerente do projeto pode agendar uma reunião dessa natureza, definindo, dentre outras coisas, quem será o moderador da reunião (que pode, inclusive, ser ele próprio). A partir de então, cabe ao moderador conduzir a reunião, sendo que o moderador pode, ainda, alterar dados da reunião ou mesmo cancelá-la, como mostra a Figura 5.2.

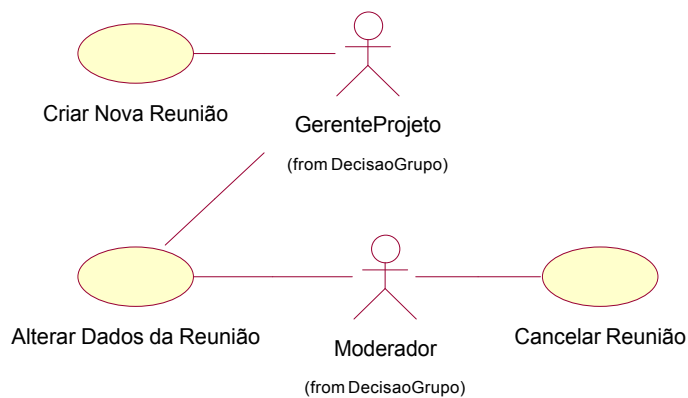


Figura 5.2 – Diagrama de Casos de Uso detalhando o Caso de Uso Preparar Reunião

- **Criar Nova Reunião:** permite ao gerente do projeto agendar uma nova reunião de decisão em grupo. Para tal, o gerente do projeto deve informar a atividade a ser discutida na reunião, o objeto de análise da mesma (artefato), o seu escopo de abrangência (projeto como um todo, um módulo, uma iteração etc) e o tipo de reunião. Além disso, ele deve selecionar, dentre os recursos humanos alocados ao projeto, ao menos dois participantes para a reunião e eleger o moderador. Opcionalmente pode registrar informações (recomendações) que considere que os participantes devam ter conhecimento para realizar a atividade. Um aviso é enviado para o moderador, informando sobre o agendamento da reunião e um fórum para discussões é criado.

- **Alterar Dados da Reunião:** permite que o moderador e o gerente do projeto alterem a lista de participantes da reunião, o moderador, as recomendações e o tipo da reunião.
- **Cancelar Reunião:** Enquanto a primeira rodada não for iniciada, o moderador pode cancelar a realização da reunião.

O diagrama de classes mostrado na Figura 5.3 apresenta as classes consideradas para tratar os casos de uso anteriormente descritos. Analisando esse diagrama, é possível notar que uma reunião de decisão em grupo acontece no contexto da realização de uma atividade (*Atividade*) do processo de software, tendo como objetivo a produção de um artefato fruto dessa atividade e considerando um determinado escopo.

Em ODE, toda Atividade de um projeto está associada a uma instância de KAtividade que indica o tipo da atividade (análise de riscos, estimativas etc). Durante a realização de uma atividade, diversos tipos de artefatos (KArtefato) podem ser gerados. Assim, há, no relacionamento entre reunião, atividade e artefato, uma restrição de integridade de que somente um tipo de artefato produzido na atividade em questão poderia estar sendo produzido na reunião.

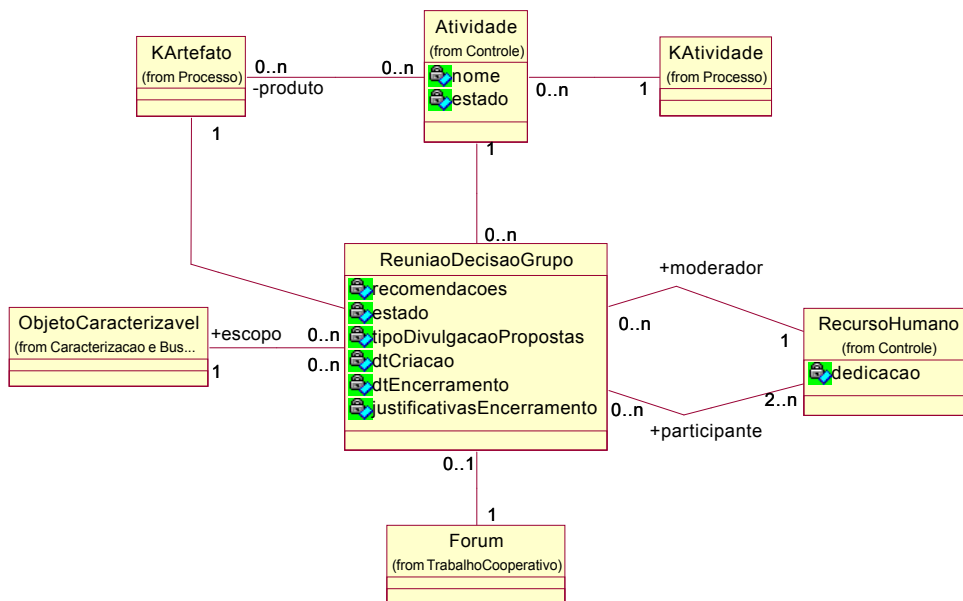


Figura 5.3 – Diagrama de Classes Parcial da Infra-estrutura.

Para a realização de uma reunião devem ser selecionados, dentre os *Recursos Humanos* alocados ao projeto, ao menos dois participantes. Além disso, um *Recurso Humano* deve ser selecionado para exercer o papel de moderador.

O escopo da reunião deve, por sua vez, ser um *ObjetoCaracterizavel*, de forma que os participantes possam refletir sobre as características do escopo que influenciam a atividade em questão.

Finalmente, a fim de estimular a cooperação, toda reunião tem, associada a si, um fórum (*Forum*) através do qual os participantes poderão trocar idéias durante a realização das atividades.

5.2.2 Controlar Reunião

Ao nomear um membro da equipe para ser o moderador de uma reunião de decisão em grupo, o gerente de projetos está confiando a esse indivíduo a responsabilidade de controlar a reunião. Diversas funcionalidades são providas ao moderador para a realização dessa tarefa, como mostra a Figura 5.4.

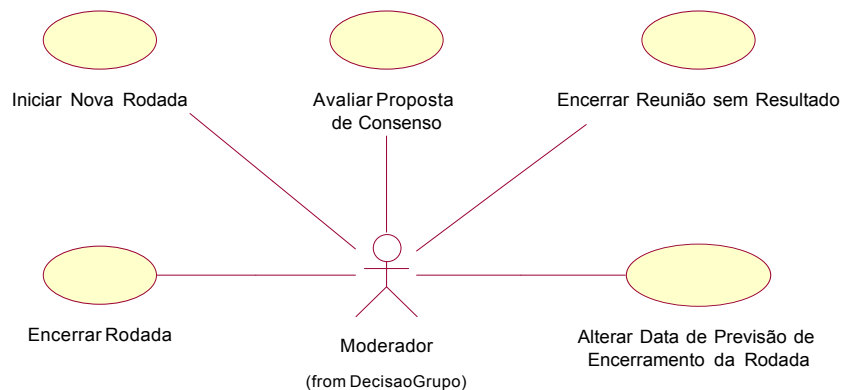


Figura 5.4 – Funcionalidades para Controle de Reuniões pelo Moderador.

- ***Iniciar Nova Rodada:*** permite que o moderador inicie uma nova rodada de discussão. O moderador deve informar a data prevista para o encerramento da rodada, sendo que a data atual é registrada como data de início da rodada. Um aviso é enviado para todos os participantes, notificando sobre a nova rodada e a data prevista para seu encerramento.
- ***Encerrar Rodada:*** permite que o moderador encerre a rodada em andamento. O sistema deve alertar o moderador caso a data prevista para encerramento ainda não tenha sido atingida ou algum participante ainda não tenha entregado sua proposta referente à rodada em questão. A data atual é registrada como data de encerramento da rodada e, a partir de então, não são mais aceitas submissões de propostas. É enviado um aviso a todos os participantes, notificando sobre o encerramento da rodada e um aviso é enviado ao moderador listando os participantes que não submeteram propostas.
- ***Alterar Data de Previsão de Encerramento da Rodada:*** permite que o moderador altere a data prevista para encerramento da rodada em andamento, sendo um aviso enviado para todos os participantes informando sobre a nova data definida.
- ***Encerrar Reunião sem Resultado:*** permite que o moderador encerre a reunião sem atingir efetivamente seu objetivo, ou seja, gerar um artefato a partir das idéias dos participantes. O moderador deve tecer considerações sobre sua decisão de encerrar a reunião sem resultado e a data de finalização da reunião é registrada.
- ***Avaliar Proposta de Consenso:*** Após propor um consenso e avaliar as opiniões postadas pelos participantes, o moderador tem três opções, a saber: (i) Aceitar a proposta de consenso sem alterações, elegendo-a como proposta consensual da reunião. Nesse caso, a reunião é encerrada; (ii) Com base nas opiniões dos participantes, o moderador pode optar por efetuar pequenas alterações na proposta de consenso e definir a proposta alterada como sendo o resultado final da reunião, encerrando-a; (iii) Devido às opiniões e argumentos dos participantes contrários à proposta, o moderador pode rejeitar a proposta de consenso, restando, então, duas opções: encerrar a reunião sem um resultado ou iniciar uma nova rodada a fim de chegar a um artefato consensual.

Visando a prover um melhor entendimento sobre as etapas de uma reunião de decisão em grupo em ODE, a Figura 5.5 mostra um diagrama de estados descrevendo o ciclo de vida de uma reunião de decisão em grupo.

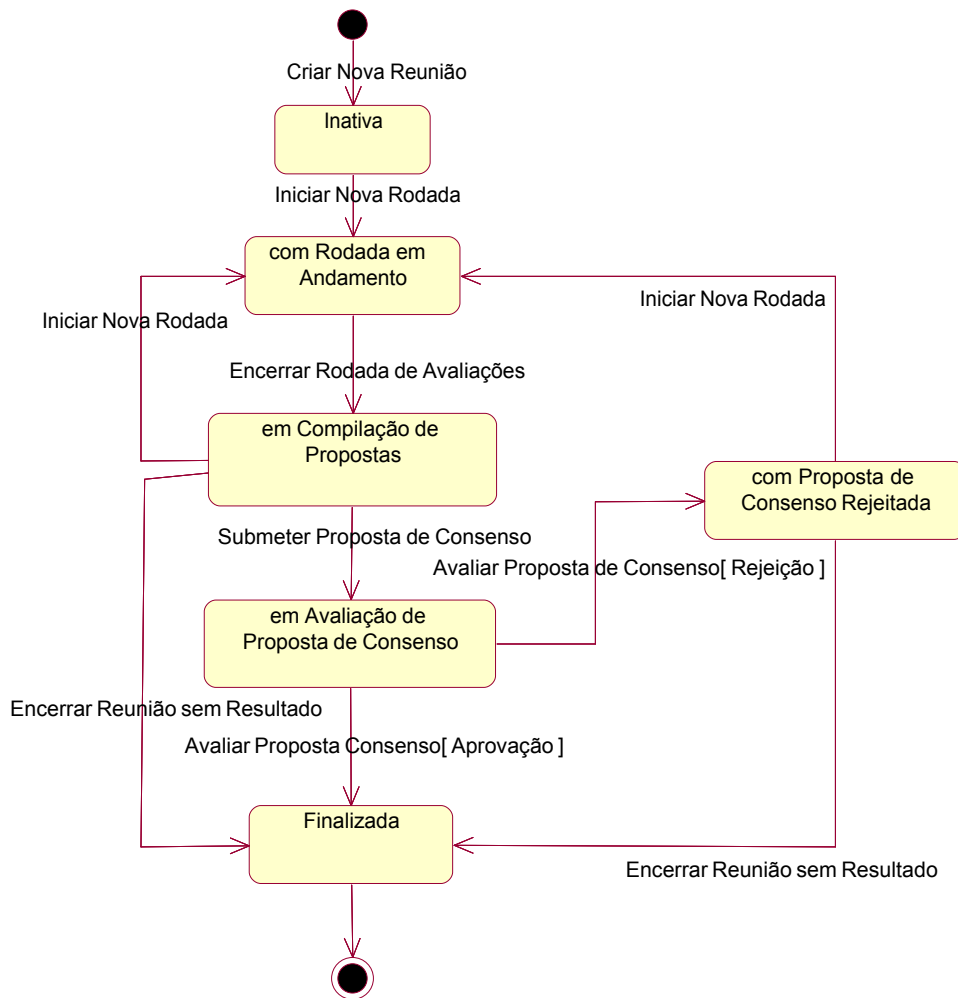


Figura 5.5 – Ciclo de Vida de uma Reunião de Decisão em Grupo

Conforme pode ser observado, quando uma reunião é criada, ela fica *inativa* até que o moderador inicie a primeira rodada de discussão da reunião, quando a reunião passa a estar *com rodada em andamento*. Com o encerramento da rodada, a reunião passa para um estágio de *compilação de propostas*, durante o qual o moderador vai analisar todas as propostas

apresentadas pelos participantes e optar por uma de três possibilidades, a saber: (i) iniciar uma nova rodada para que os participantes revejam suas propostas com base nas propostas dos demais participantes e/ou em novos dados apresentados, (ii) mediante a divergência de opiniões, encerrar a reunião sem resultado ou (iii) elaborar uma proposta de consenso a ser avaliada pelos participantes, o que levaria a reunião a ficar *em avaliação de proposta de consenso*.

Após avaliar as opiniões postadas pelos participantes sobre a proposta de consenso, o moderador deverá decidir entre aceitar a proposta sem alterações, aceitar a proposta efetuando alterações na mesma ou rejeitar a proposta. Caso o moderador opte por aceitar a proposta, efetuando ou não alterações sobre a mesma, a proposta aceita será considerada o artefato resultante da reunião e a reunião será *finalizada*. Em caso de rejeição da proposta de consenso, o moderador poderá encerrar a reunião sem ter um artefato consensual gerado, ou seja, a reunião será *finalizada*, ou iniciar uma nova rodada de discussão a fim de chegar a um artefato consensual, o que levará a reunião a ficar novamente *com rodada em andamento*.

5.2.3 Participar da Reunião

Em uma reunião de decisão em grupo, a cada rodada de discussão, cada participante deve gerar sua proposta de artefato e, ao final da rodada, o moderador pode elaborar uma proposta de consenso. Para que os participantes e o moderador possam elaborar propostas bem fundamentadas, é necessário que tenham acesso ao maior número possível de informações sobre a atividade que está sendo desenvolvida, o escopo de análise etc. Enfim, é necessário prover o maior número de informações relevantes, de forma que o esforço necessário para que os participantes as utilizem seja o menor possível. A Figura 5.6 mostra as funcionalidades providas para fornecer essas informações e permitir a elaboração das propostas. A seguir, essas funcionalidades são brevemente descritas:

- **Consultar Recomendações:** permite que os participantes tenham acesso às recomendações feitas pelo gerente do projeto e pelo moderador da reunião de modo a apoiá-los na elaboração de suas propostas.

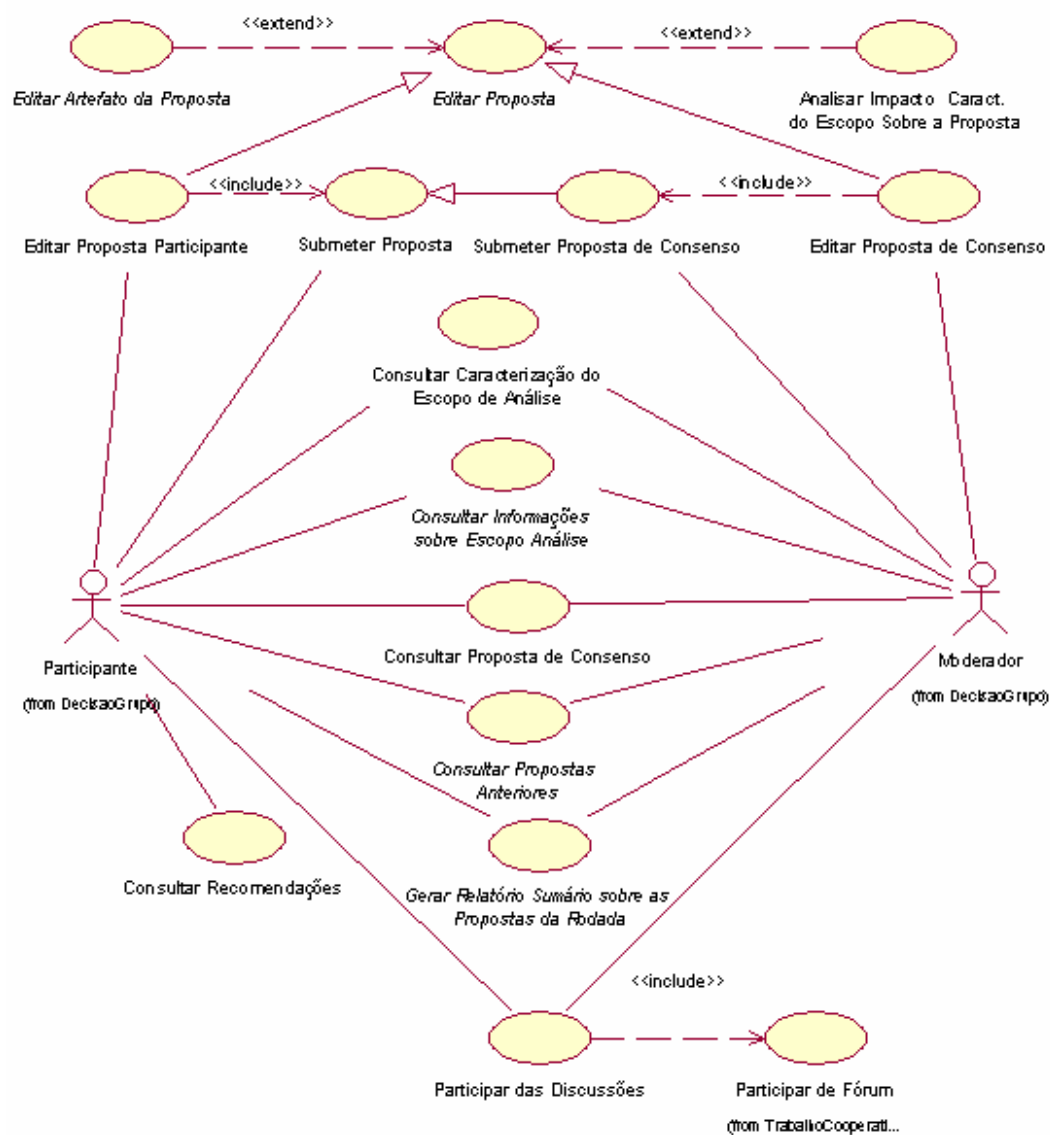


Figura 5.6 – Funcionalidades Providas para Permitir a Participação em Reuniões.

- **Consultar Caracterização do Escopo de Análise:** permite que o moderador e os demais participantes da reunião visualizem a caracterização do escopo envolvido na elaboração das propostas (p.ex. , projeto, módulo, iteração).

- ***Consultar Informações sobre o Escopo de Análise:*** trata-se de um caso de uso abstrato que deve ser implementado em cada especialização da infra-estrutura, de acordo com o escopo de análise e com a atividade a ser apoiada. O objetivo é que sejam exibidas informações sobre o escopo de análise que sejam úteis na elaboração da proposta.
- ***Editar Proposta:*** trata-se de um caso de uso abstrato que permite que propostas sejam elaboradas do zero ou utilizando uma proposta anteriormente elaborada como ponto de partida. A proposta se divide em duas partes, a saber: análise de impacto das características do escopo sobre a proposta (efetuada por meio da realização do cenário “*Analisar Impacto das Características do Escopo sobre a Proposta*”) e geração do artefato da proposta (efetuada pela realização do caso de uso “*Editar Artefato da Proposta*”). Conforme pode ser visto no diagrama da Figura 5.8, os casos de uso “*Editar Proposta Participante*” e “*Editar Proposta Consenso*” especializam este caso de uso.
- ***Analisar Impacto das Características do Escopo sobre a Proposta:*** permite que sejam indicados os níveis de impacto que cada característica do objeto de análise exerceram sobre a proposta elaborada.
- ***Editar Artefato da Proposta:*** visa a permitir a edição do artefato da proposta. Como a geração do artefato vai variar de acordo com a atividade da reunião, este caso de uso é abstrato. Tipicamente, ele será responsável por iniciar a ferramenta de ODE que apóia a execução da atividade em questão.
- ***Editar Proposta Participante:*** permite que os participantes elaborem suas propostas. Este caso de uso especializa o caso de uso abstrato “*Editar Proposta*”, adicionando as seguintes particularidades: (i) como ponto de partida para a elaboração de sua proposta, o participante só pode usar sua própria proposta apresentada na rodada anterior da reunião; (ii) o participante pode, opcionalmente, submeter sua proposta através da realização do caso de uso “*Submeter Proposta*”.
- ***Editar Proposta de Consenso:*** permite que o moderador elabore uma proposta de consenso. Este caso de uso especializa o caso de uso abstrato “*Editar Proposta*”, adicionando as seguintes particularidades: (i) como ponto de partida para a

elaboração de sua proposta, o moderador pode usar a proposta apresentada por qualquer dos participantes na rodada corrente ou alguma proposta de consenso já apresentada anteriormente; (ii) o moderador pode, opcionalmente, submeter sua proposta de consenso realizando o caso de uso “*Submeter Proposta de Consenso*”.

- ***Submeter Proposta:*** permite que os participantes submetam suas propostas ao moderador.
- ***Submeter Proposta de Consenso:*** permite que o moderador submeta uma proposta de consenso para avaliação dos participantes, informando, ainda, uma data prevista para a decisão sobre o aceite ou rejeição da mesma. É enviado um aviso aos participantes da reunião, notificando sobre a proposta de consenso postada.
- ***Consultar Proposta de Consenso:*** permite que o moderador e os participantes consultem a proposta de consenso corrente.
- ***Consultar Propostas Anteriores:*** permite que o moderador e os participantes consultem propostas submetidas em rodadas anteriores.
- ***Gerar Relatório Sumário sobre as Propostas da Rodada:*** permite que o moderador e os participantes tenham acesso a um relatório sumário das propostas entregues. Assim como as propostas, o relatório também se divide em duas partes, a saber: análise de impacto das características do escopo sobre a proposta e dados dos artefatos propostos na rodada. Sobre a análise de impacto das características, são apresentadas estatísticas, tais como o percentual de propostas nas quais cada característica foi lembrada e o peso médio dado a cada característica. Os dados a serem exibidos sobre os artefatos gerados variam de acordo com o tipo de artefato em discussão. Assim, trata-se de um caso de uso abstrato que deverá ser especializado em cada especialização da infra-estrutura.
- ***Participar das Discussões:*** permite que o moderador e os participantes leiam as mensagens já postadas no fórum da reunião, postem novas questões nesse fórum e respondam a questões postadas no mesmo. Neste caso de uso são utilizadas funcionalidades do caso de uso Participar de Fórum proposto em (ARANTES, 2004).

A Figura 5.7 apresenta o diagrama de classes completo da infra-estrutura, incluindo as classes de apoio à elaboração de propostas.

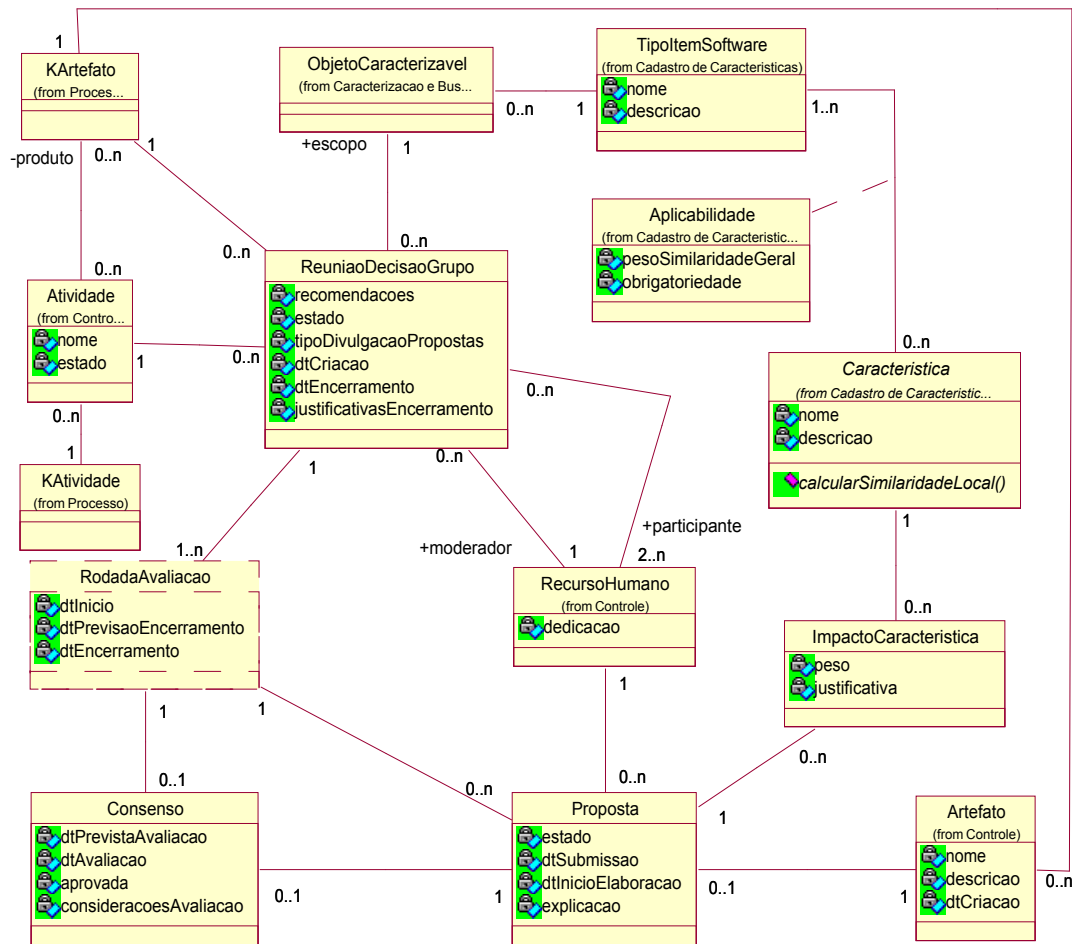


Figura 5.7 – Diagrama de Classes Completo da Infra-estrutura.

Durante uma reunião de decisão em grupo, podem ser realizadas várias rodadas de avaliação. Em cada rodada, cada um dos participantes deve elaborar uma proposta de artefato. Sobre cada proposta são registradas a data de início da sua elaboração, a data de submissão e uma explicação textual do autor da proposta sobre a mesma.

Conforme citado anteriormente, a proposta é composta de duas partes: uma análise de impacto das características do escopo sobre a proposta e o artefato da proposta.

Em ODE, todo *Artefato* está associado a uma instância de *KArtefato* que indica o tipo do artefato (plano de riscos, modelo de classes etc). Uma reunião de decisão em grupo tem como objetivo a produção de um artefato de determinado tipo. Assim, o artefato de uma proposta de uma reunião tem de ser do tipo do artefato (*KArtefato*) a ser construído durante essa reunião.

A análise de impacto associa, a cada característica aplicável ao escopo de análise, o impacto que essa característica exerceu sobre a proposta. As características aplicáveis ao escopo de análise são definidas por instâncias da classe *Aplicabilidade*, de acordo com o tipo do objeto de análise, conforme definido pela infra-estrutura de caracterização apresentada no Capítulo 4 deste trabalho. O impacto de cada característica sobre a proposta é representado por uma instância da classe *ImpactoCaracteristica*, que registra, além do peso do impacto (que deve ser um valor entre zero e um), a justificativa do autor da proposta para a atribuição de tal valor.

Ao fim de cada rodada, o moderador pode optar por propor um consenso. A classe *Consenso*, além de conter uma proposta, registra, ainda, a data prevista para avaliação de seu aceite, a data quando efetivamente foi avaliada, o resultado de sua avaliação (se aprovada ou rejeitada) e as considerações do moderador sobre os motivos que o levaram a aprovar ou rejeitar tal proposta de consenso.

5.3 Apoio à Elaboração Cooperativa de Planos de Risco

Uma vez desenvolvida a infra-estrutura, foi realizada neste trabalho uma especialização da mesma para apoiar a elaboração cooperativa de planos de risco.

Para realizar uma reunião visando à construção cooperativa de um Plano de Risco, o Gerente do Projeto deve utilizar a funcionalidade de Criar Nova Reunião, informando que a atividade a ser discutida será uma atividade de Análise de Riscos. De fato, a Análise de Riscos aparece como única opção de atividade para criação de reunião, pois se trata da única especialização da infra-estrutura desenvolvida até então. À medida que novas especializações forem sendo construídas, novas opções de atividades serão adicionadas. A Figura 5.8 apresenta a interface que permite a criação de novas reuniões de decisão em grupo em ODE.

Dados da Reunião

Atividade a ser Discutida: **Análise de Riscos**

Artefato a ser Produzido: **Plano de Riscos**

Escopo

Tipo Objeto: **Projeto**

Objeto: **Projeto ODE**

Tipo Reunião

☒ Aberta

☐ Fechada

Participantes **Moderador** **Recomendações**

Verificar os dados sobre o escopo;
Verificar a caracterização do escopo;
Analisar dados de projetos similares;
Trocar idéias com outros membros da Equipe.

OK Fechar

Figura 5.8 –Criando uma Nova Reunião

Dentre os participantes de uma reunião de decisão em grupo, um é eleito moderador da reunião. Cabe ao moderador da reunião a responsabilidade de conduzi-la. A Figura 5.9 mostra a interface da infra-estrutura de apoio à decisão em grupo que visa a prover ao moderador as funcionalidades necessárias à condução da reunião, dentre elas: iniciar nova rodada, encerrar rodada, editar proposta de consenso e avaliar proposta de consenso.

Conduzir Reuniao

Dados da Reuniao

Atividade a ser Discutida: Escopo de Análise:

Artefato a ser Produzido: Data Criação:

Estado:

Dados da Última Rodada

Rodada: Início: Encerramento Previsto: Encerramento Efetivo:

Consultas

Proposta Consenso

Figura 5.9 – Funcionalidades Oferecidas para o Moderador da Reunião

A Figura 5.10, por sua vez, mostra a interface da infra-estrutura de apoio à decisão em grupo que fornece aos participantes da reunião acesso às funcionalidades de apoio à elaboração das propostas, dentre elas: elaborar proposta, consultar informações sobre o escopo de análise, consultar propostas anteriores e consultar recomendações.

Participar Reuniao

Dados da Reunião

Atividade Discutida:

Artefato a ser Produzido:

Escopo de Análise:

Estado: Data Criação:

Consultas

Rodada

Rodada: Início: Encerramento Previsto: Encerramento Efetivo:

Proposta

Data de Início: Data de Submissão:

Figura 5.10 – Funcionalidades Oferecidas para os Participantes da Reunião

Conforme anteriormente citado, uma proposta é dividida em duas partes a saber: análise de impacto das características do escopo sobre a proposta e geração do artefato da proposta, que no caso dessa especialização para apoiar a elaboração cooperativa de planos de risco, trata-se da geração de planos de riscos. A Figura 5.11 mostra a interface que permite a análise de impacto das características do escopo sobre a proposta.

Características

Considerada?	Característica
☐	Paradigma
☒	Estabilidade dos Requisitos
☐	Tipo de Desenvolvimento
☐	Experiência da Equipe

Impacto

Peso: (Slider from 0 to 10)

Justificativa:

Apresenta alto risco de alteração de requisitos

Salvar

Fechar

Figura 5.11 – Janela de Análise de Impacto das Características do Escopo sobre a Proposta

A especialização da infra-estrutura de apoio à decisão em grupo para apoiar a elaboração cooperativa de planos de riscos consiste, basicamente, da definição e tratamento dos casos de uso abstratos da infra-estrutura, apontados na seção anterior. O diagrama de casos de uso da Figura 5.12 mostra os casos de uso abstratos que foram especializados para apoiar a elaboração cooperativa de planos de risco. A seguir, esses casos de uso são brevemente descritos.

- **Editar Plano de Riscos:** permite que os participantes e o moderador elaborem os planos de riscos a serem submetidos como artefatos de suas propostas. Este caso de uso especializa o caso de uso abstrato “*Editar Artefato da Proposta*”. Para a elaboração dos Planos de Riscos, foi utilizada a ferramenta de apoio à gerência de riscos de ODE (GeRis) (FALBO et al., 2004a). Assim, esse caso de uso é, de fato, uma chamada à funcionalidade de GeRis que cria o plano de riscos e o associa criado à proposta em questão, como mostra o diagrama de classes da Figura 5.13.

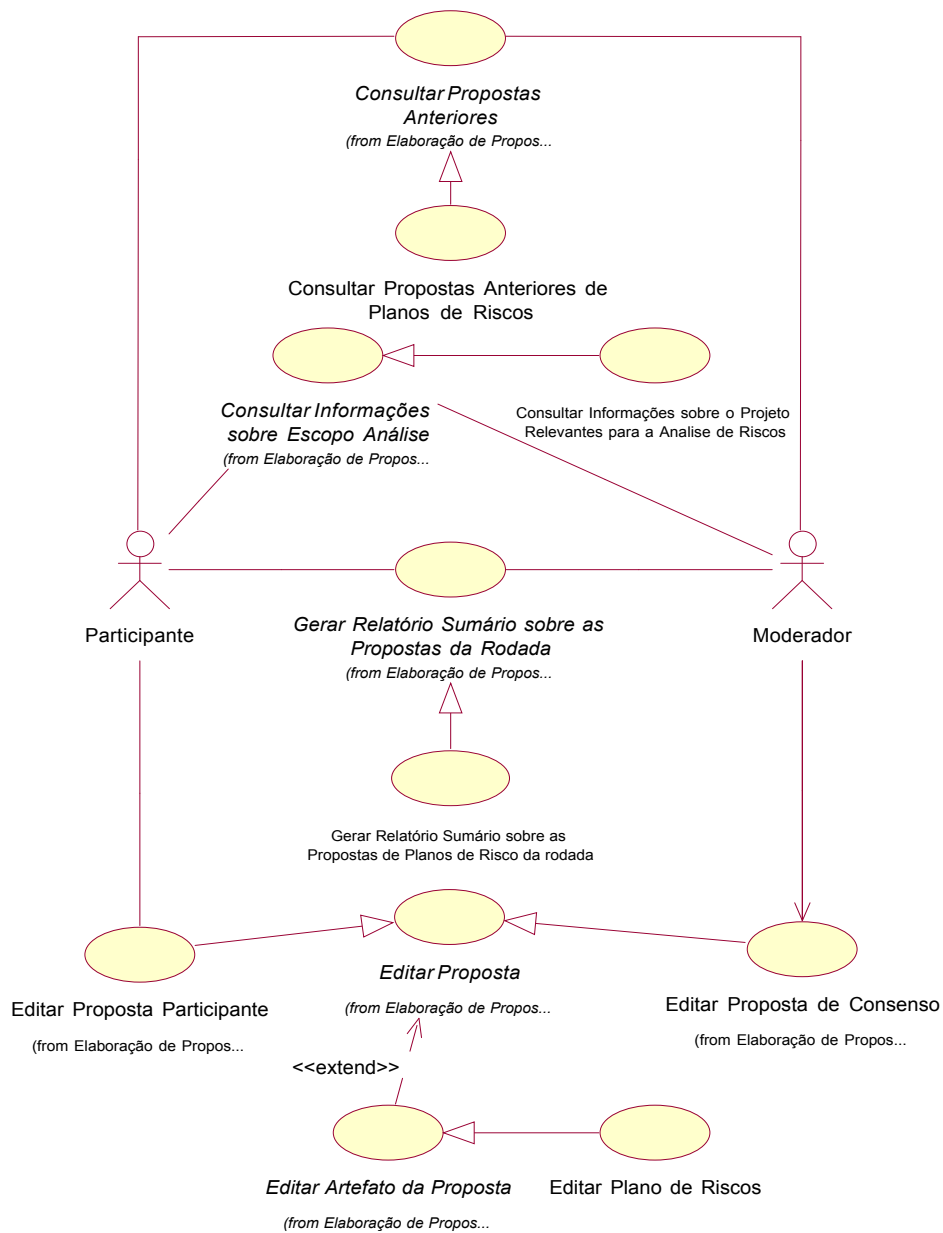


Figura 5.12 – Casos de Uso Especializados para Apoiar a Elaboração Cooperativa de Planos de Riscos.

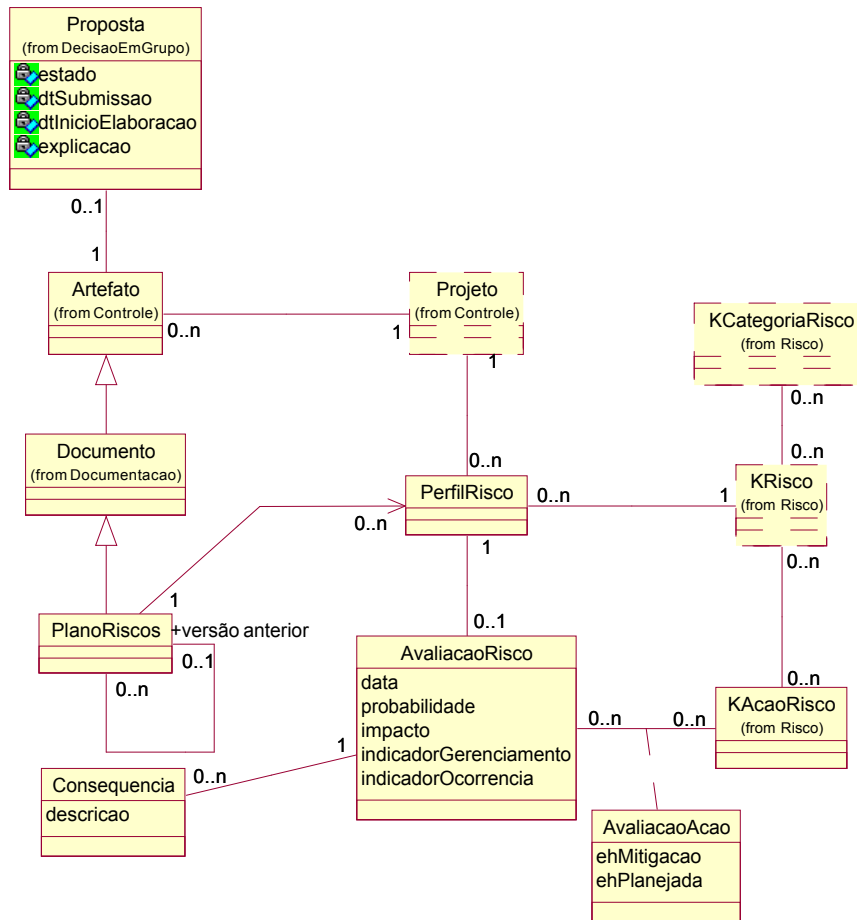


Figura 5.13 – Modelo de Classes de Planos de Riscos

Analisando o diagrama exibido na figura 5.13, é possível notar que um plano de riscos define quais são os riscos considerados em um projeto (*PerfilRisco*). Para cada risco considerado em um projeto, há uma avaliação (*AvaliacaoRisco*), indicando sua probabilidade, impacto e se será gerenciado ou não. Caso se tenha definido gerenciar o risco, ações de contingência e/ou mitigação devem ser planejadas. Por fim, caso um risco venha a ocorrer, suas consequências são informadas.

Nessa especialização da infra-estrutura de apoio ao trabalho em grupo para apoiar a elaboração cooperativa de planos de risco, a edição do plano de riscos é feita a

partir das funcionalidades normalmente fornecidas em GeRis, como mostra a Figura 5.14.

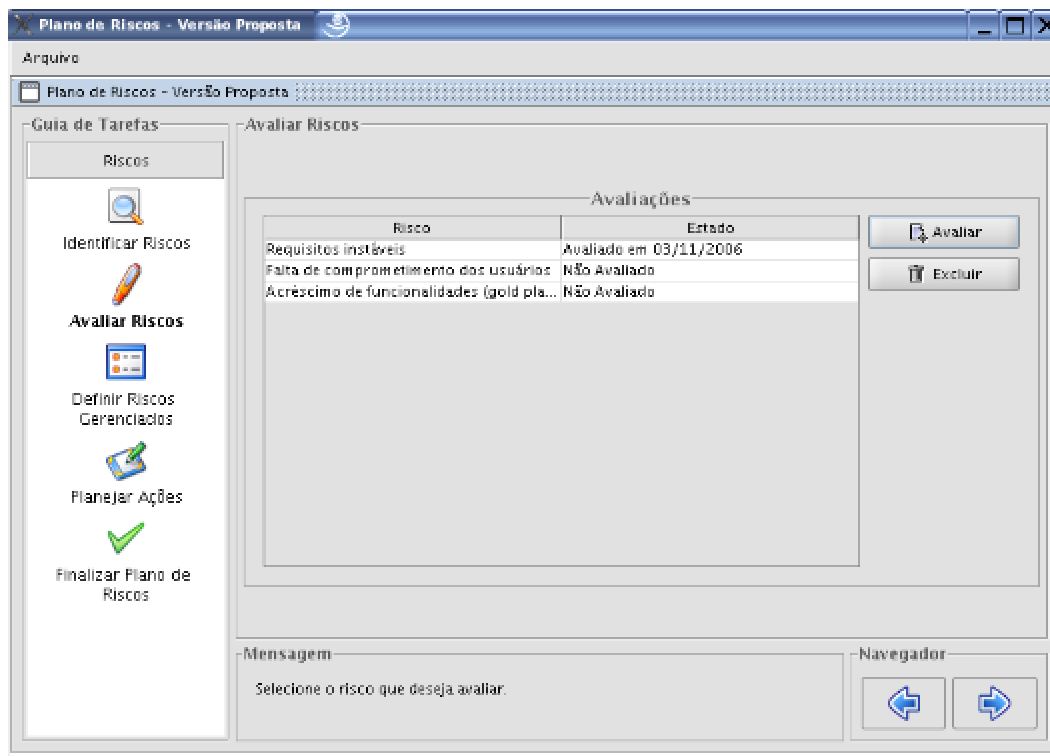
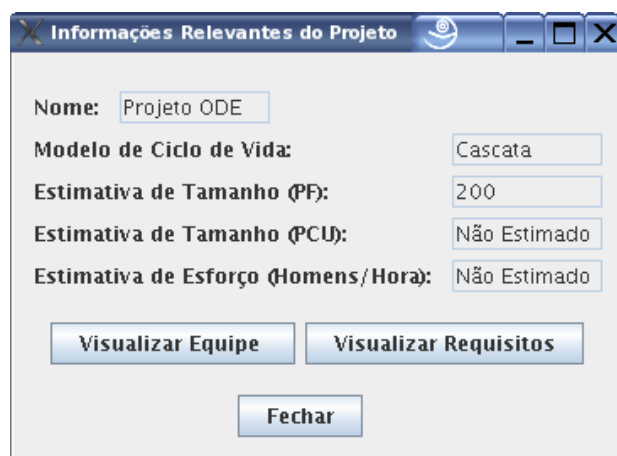


Figura 5.14 – Interface de Geris no Momento de uma Avaliação de um Risco

- **Consultar Propostas Anteriores de Planos de Riscos:** permite que o moderador e os participantes consultem propostas de Planos de Riscos submetidas em rodadas anteriores. Este caso de uso especializa o caso de uso abstrato “Consultar Propostas Anteriores”. Por meio deste caso de uso é possível consultar tanto a análise de impacto das características do escopo sobre a proposta quanto o artefato da proposta, no caso, o Plano de Riscos. A exibição dos planos de riscos constituintes das propostas é feita usando diretamente GeRis.
- **Consultar Informações sobre o Projeto Relevantes para a Análise de Riscos:** este caso de uso especializa o caso de uso abstrato “Consultar Informações sobre Escopo de Análise”, de modo a exibir informações sobre o escopo de análise que

sejam úteis na elaboração das propostas de Planos de Riscos. Tendo em vista que GeRis apóia a gerência de riscos de projetos, o escopo de análise da reunião é um projeto e todas as informações referentes ao planejamento e aos requisitos do mesmo são potencialmente úteis para a análise de riscos. Dentre tais informações merecem destaque o modelo de ciclo de vida adotado, o processo de desenvolvimento utilizado, as estimativas de tamanho e esforço, o cronograma do projeto, a alocação de recursos e os requisitos funcionais e não funcionais do projeto. Como mostra a Figura 5.15, na atual versão, podem ser consultadas as seguintes informações do projeto: o modelo de ciclo de vida adotado, as estimativas de tamanho e esforço realizadas, a equipe alocada e os requisitos funcionais e não funcionais. Em versões futuras, as demais informações referentes ao planejamento do projeto devem ser adicionadas a essa funcionalidade.



Informações Relevantes do Projeto	
Nome:	Projeto ODE
Modelo de Ciclo de Vida:	Cascata
Estimativa de Tamanho (PF):	200
Estimativa de Tamanho (PCU):	Não Estimado
Estimativa de Esforço (Homens/Hora):	Não Estimado
Visualizar Equipe Visualizar Requisitos	
Fechar	

Figura 5.15 –Informações sobre o Projeto Relevantes para a Análise de Riscos

- **Gerar Relatório Sumário sobre as Propostas de Planos de Riscos da Rodada:** permite que o moderador e os participantes tenham acesso a um relatório sumário das propostas entregues. Este caso de uso especializa o caso de uso “Gerar Relatório Sumário sobre as Propostas da Rodada”, de modo que, na seção do relatório que traz o sumário dos dados dos artefatos, seja apresentado um resumo dos planos de riscos propostos. Assim, como mostra a Figura 5.16, são apresentados dados estatísticos como: a quantidade de propostas em que cada risco

foi citado, a probabilidade média e o impacto médio definidos para cada risco, bem como o desvio padrão da probabilidade e impacto apresentados para esses riscos.

Risco	Quantidade de Citações	Probabilidade Média	Desvio Padrão Probabilidade	Impacto Médio	Desvio Padrão Impacto
Risco 1	N	P%	D%	I	D
Risco 2	M	X%	Y%	Z	W

Figura 5.16 – Relatório Sumário sobre as Propostas de Planos de Riscos da Rodada

5.4 Captura do Raciocínio

Segundo VALENTE et al. (2002), a memória organizacional deve conter vários tipos de itens de conhecimento, que podem ser relevantes para os usuários. Esse conhecimento compreende tanto o conhecimento estruturado quanto o conhecimento informal. O primeiro é o conhecimento mais fácil de ser compartilhado e compreende processos, procedimentos, métricas de projetos e os artefatos desenvolvidos e reutilizados durante o projeto. Já o conhecimento informal não é tão fácil de ser compartilhado, pois, na maioria das vezes, compreende a experiência que a equipe ou a pessoa teve no decorrer do projeto, como, por exemplo, na realização da análise de riscos desse projeto. No entanto, essa experiência é muito importante, pois complementa o conhecimento formal e aponta observações que devem ser consideradas, servindo como base para melhorar o processo de desenvolvimento de software como um todo.

Conforme citado anteriormente, em seu estágio atual, ODE conta com diversas ferramentas que apóiam a geração e armazenamento dos artefatos desenvolvidos durante as diversas atividades do processo de software. Porém, não há em ODE, mecanismos que apóiem a captura do raciocínio seguido durante a construção desses artefatos. Dessa forma, o reúso do conhecimento é dificultado, pois os usuários têm acesso aos artefatos gerados em projetos anteriores, porém desconhecem o contexto em que os mesmos foram produzidos, as razões que

levaram ao conteúdo de tal artefato, as características do projeto que foram consideradas pelos desenvolvedores do artefato, as alternativas propostas etc.

Nesse contexto, a infra-estrutura desenvolvida visa a, além de apoiar a elaboração cooperativa dos artefatos, capturar a visão dos desenvolvedores envolvidos na produção do artefato sobre as características do projeto que influenciaram o artefato gerado, as alternativas propostas pelos desenvolvedores e as justificativas apresentadas por cada um deles para fundamentar seu ponto de vista. Enfim, o objetivo é tentar traçar o raciocínio seguido pelo conjunto de desenvolvedores desde o início da atividade até a geração do artefato final.

A fim de alcançar tais objetivos, foram empregadas técnicas de *Design Rationale* na construção da infra-estrutura. Como o objetivo ao utilizar *Design Rationale* era expressar as razões para as diversas decisões tomadas pelos desenvolvedores durante a elaboração de um artefato, foi utilizada uma abordagem baseada em argumentação.

O modelo de representação desenvolvido neste trabalho tenta capturar a visão dos desenvolvedores envolvidos na produção do artefato de três formas, a saber:

- Registro das argumentações apresentadas para as propostas: Ao propor uma alternativa de artefato, um desenvolvedor deve apresentar justificativas que fundamentem sua proposta. Além disso, ele deve informar, ainda, quais características do projeto influenciaram sua proposta e qual o nível de impacto de cada uma delas sobre a proposta. As justificativas e a análise de impacto das características sobre a proposta também são apresentadas pelo moderador quando é feita uma proposta de consenso. Dessa forma, consegue-se representar as alternativas de artefatos consideradas e a argumentação utilizada para justificar cada uma dessas alternativas;
- Registro das argumentações apresentadas nas avaliações das propostas de consenso: Após apresentar uma proposta de consenso e analisar as opiniões dos participantes sobre a mesma, o moderador deve avaliar a proposta de consenso e decidir se a aceita ou a rejeita. Quando o moderador apresenta sua decisão sobre o aceite ou rejeição de uma proposta de consenso, ele deve apresentar justificativas para a decisão tomada. Assim, são capturadas as decisões sobre o aceite ou a

rejeição das propostas de consenso e as argumentações apresentadas para justificar essas decisões;

- Registro da cooperação entre os participantes: A fim de estimular a cooperação entre os participantes, a infra-estrutura de apoio à decisão em grupo provê um fórum de discussão. Assim, o fórum deve ser utilizado para o compartilhamento de idéias e a realização de discussões. Dessa forma, a troca de mensagens no fórum é capturada como uma maneira de se representar a cooperação existente entre os participantes da atividade e de se capturar a visão de cada um sobre os assuntos discutidos.

Em suma, a representação implementada procura capturar o raciocínio seguido durante a elaboração cooperativa de um artefato por meio do registro das argumentações dos participantes para justificar suas propostas e decisões e por meio da captura das interações desses participantes.

A representação implementada satisfaz a quatro dos seis requisitos para a representação de *design rationale* apresentados por MEDEIROS (2006) e citados no Capítulo 3 deste trabalho, a saber:

1. *Representação explícita de decisões*: Esse requisito é satisfeito através do registro das justificativas apresentadas pelo moderador para aceitar ou rejeitar uma proposta de consenso;
2. *Distinção entre argumentos e justificativa final*: na infra-estrutura proposta, cabe ao moderador da reunião a decisão final sobre o aceite de uma proposta de consenso, bem como as justificativas para a decisão tomada. Essa decisão independe das propostas e argumentos apresentados pelos participantes. Assim, o moderador pode se decidir por aceitar uma proposta, apesar de todos os argumentos apresentados para ela serem contrários à sua aceitação, e registrar em uma justificativa final os motivos que o levaram a essa decisão;
3. *Integração da argumentação com as descrições dos artefatos gerados*: todas as propostas de artefatos geradas durante a reunião e suas respectivas justificativas são armazenadas com o intuito de traçar todo o raciocínio seguido até a geração do

artefato final. Com isso visa-se a facilitar a compreensão sobre o artefato produzido e, conseqüentemente, a decisão sobre reusá-lo ou não

4. *Registro de informações sobre as atividades de design*: diversas informações sobre a atividade de geração do artefato são capturadas, como por exemplo, quais foram os participantes da atividade e o responsável pelas decisões, as recomendações dadas pelo gerente de projeto para a realização da atividade e as datas em que as propostas foram elaboradas e as decisões tomadas.
5. *Definição de uma linguagem de representação expressiva*: a representação do *design* em uma linguagem que permita seu processamento computacional é uma tarefa bastante complexa que foge aos objetivos deste trabalho. A intenção desse trabalho é registrar o raciocínio seguido durante a elaboração do artefato de forma que ele possa ser recuperado e utilizado para repetir uma decisão que tenha se mostrado eficiente ou evitar a repetição de um erro.
6. *Integração do design rationale com o método de design*: Conforme citado no item anterior, não é objetivo deste trabalho possibilitar a realização de operações computáveis.

Apenas capturar o raciocínio seguido pelos participantes da reunião não é suficiente. Para que esse conhecimento capturado seja utilizado, é muito importante que ele seja apresentado de forma clara. Dessa forma, foi adicionado à Gerência de Conhecimento ODE mais um tipo de item de conhecimento: a História da Reunião. O novo modelo de classes da infra-estrutura de Gerência de Conhecimento de ODE é mostrado na Figura 5.17, destacando esse novo tipo de item de conhecimento.

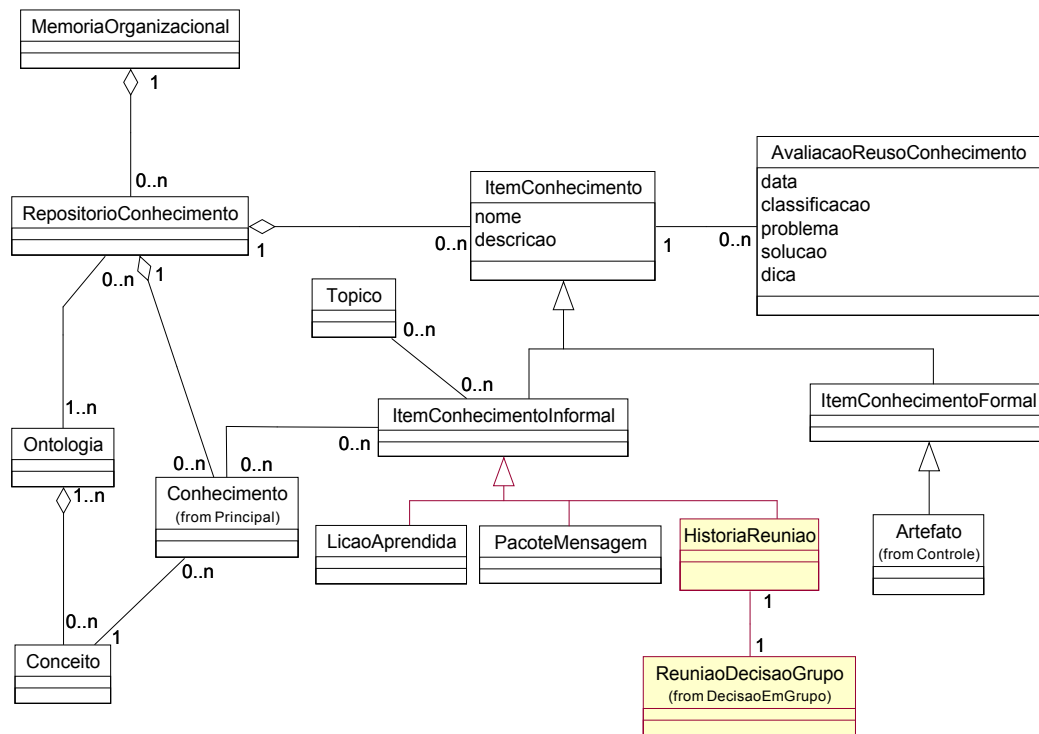


Figura 5.17 – Novo Item de Conhecimento Acrescentado à Infra-estrutura de Gerência de Conhecimento de ODE

Esse item de conhecimento corresponde a um relatório contendo toda a história da reunião com base na atuação de cada um dos participantes em cada uma das rodadas, seguindo o *layout* mostrado na Figura 5.18. Os trechos apresentados em preto no relatório da História da Reunião independem da atividade realizada na mesma. Em azul, são apresentados os *hyperlinks* para itens importantes da reunião. Os trechos em vermelho, por sua vez, variam de acordo com a atividade realizada na reunião. O *layout* apresentado na Figura 5.19 refere-se ao relatório desenvolvido na especialização para apoiar a Gerência de Riscos e, portanto, mostra os trechos em vermelho já adaptados para essa atividade.

História da Reunião

O objetivo desta reunião foi realizar a atividade *[atividade da reunião]* gerando o artefato *[artefato da reunião]* no contexto do *[tipo do Objeto]* *[Nome do Objeto]*.

A reunião contou com *[número de participantes]* participantes. Os participantes exerciam os seguintes papéis no projeto: *[papel participante 1, papel participante 2, ..., papel participante n]*.

O início da reunião aconteceu no dia *[data de início da reunião]* e o encerramento em *[data de encerramento da reunião]*. Durante esse período foram realizadas *[número de rodadas de da reunião]* rodadas de discussão, sendo geradas *[número de propostas de participantes]* propostas pelos participantes e *[número de propostas de consenso]* propostas de consenso.

Quanto ao [resultado da reunião](#) [<LINK PARA RESULTADO DA REUNIÃO>](#), *[foi obtido através de uma proposta de consenso aprovada sem alterações ou foi obtido através de uma proposta de consenso aprovada com alterações ou não se chegou a um consenso ficando a reunião sem um resultado]*.

A seguir é apresentada a história da reunião, passando por cada uma de suas rodadas.

RODADA *[número da rodada]*

Durante a rodada *[número da rodada]* foram submetidas *[n]* [mensagens](#) [<LINK PARA MSGS DA RODADA>](#) ao fórum de discussão.

Foram submetidas *[n]* propostas pelos participantes e as características mais freqüentemente apontadas nas análises de impacto foram: *[carac1, carac2, carac3, carac4]*. As características apontadas com maiores médias de impacto foram: *[carac1, carac2, carac3, carac4]*.

[Texto referenciando a atividade da reunião e apresentando um resumo do resultado final, com um link para o sumário da rodada. <LINK PARA SUMARIO RODADA>]

PARTICIPANTE *[n]* : As características que tiveram mais impacto sobre a [proposta](#) [<LINK PARA PROPOSTA>](#) apresentada foram: *[carac1, carac2, carac3, carac4]*.

[Texto referenciando a atividade da reunião e apresentando um resumo da proposta do participante].
A explicação para a proposta apresentada foi a seguinte: *[Explicação textual da Proposta]*.

MODERADOR: Nesta rodada o moderador *[apresentou ou não apresentou]* uma proposta de consenso.

As características que tiveram mais impacto sobre a [proposta de consenso](#) [<LINK PARA PROPOSTA DE CONSENSO DA RODADA>](#) apresentada foram: *[carac1, carac2, carac3, carac4]*.

[Texto referenciando a atividade da reunião e apresentando um resumo da proposta de consenso da rodada].

A explicação apresentada para a proposta de consenso foi a seguinte: *[Explicação textual da Proposta]*.

Após avaliação da proposta de consenso, esta foi *[aprovada sem alterações, consenso aprovada com alterações ou rejeitada]*.

As considerações apresentadas sobre a avaliação da proposta foram das seguintes: *[Considerações da Avaliação do Consenso]*.

Assim, após esta rodada, *[foi iniciada uma nova rodada, a reunião foi encerrada com resultado ou a reunião foi encerrada sem resultado]*.

Figura 5.18 – Layout Básico de uma História de Reunião.

História da Reunião

O objetivo desta reunião foi realizar a atividade *Análise de Riscos* gerando o artefato *Plano de Riscos* no contexto do Projeto [Nome do Projeto].

A reunião contou com [número de participantes] participantes. Os participantes exerciam os seguintes papéis no projeto: [papel participante 1, papel participante 2, ..., papel participante n].

O início da reunião aconteceu no dia [data de início da reunião] e o encerramento em [data de encerramento da reunião]. Durante esse período foram realizadas [número de rodadas de da reunião] rodadas de discussão, sendo geradas [número de propostas de participantes] propostas pelos participantes e [número de propostas de consenso] propostas de consenso.

Quanto ao [resultado da reunião <LINK PARA RESULTADO DA REUNIÃO>](#), [foi obtido através de uma proposta de consenso aprovada sem alterações ou foi obtido através de uma proposta de consenso aprovada com alterações ou não se chegou a um consenso ficando a reunião sem um resultado].

A seguir é apresentada a história da reunião, passando por cada uma de suas rodadas.

RODADA [número da rodada]

PARTICIPANTE [n] : As características que tiveram mais impacto sobre a proposta apresentada foram: [carac1, carac2, carac3, carac4].

Os riscos apontados nesta proposta [<LINK PARA PROPOSTA COMPLETA>](#) foram os seguintes: [risco1, risco2].

A explicação para a proposta apresentada foi a seguinte: [Explicação textual da Proposta].

MODERADOR: Nesta rodada o moderador [apresentou ou não apresentou] uma proposta de Consenso.

As características que tiveram mais impacto sobre a [proposta de consenso <LINK PARA PROPOSTA DE CONSENSO DA RODADA>](#) apresentada foram: [carac1, carac2, carac3, carac4].

Os riscos apontados foram os seguintes: [risco1, risco2].

A explicação apresentada para a proposta de consenso foi a seguinte: [Explicação textual da Proposta].

Após avaliação da proposta de consenso, esta foi [aprovada sem alterações, consenso aprovada com alterações ou rejeitada].

As considerações apresentadas sobre a avaliação da proposta foram das seguintes: [Considerações da Avaliação do Consenso].

Assim, após esta rodada, [foi iniciada uma nova rodada, a reunião foi encerrada com resultado ou a reunião foi encerrada sem resultado].

Durante a rodada [número da rodada] foram submetidas [n] mensagens [<LINK PARA MSGS DA RODADA>](#) ao fórum de discussão. Foram submetidas [n] propostas pelos participantes e as características mais freqüentemente apontadas nas análises de impacto foram: [carac1, carac2, carac3, carac4]. As características apontadas com maiores médias de impacto foram: [carac1, carac2, carac3, carac4]. Os riscos mais apontados pelos participantes foram: [risco1, apontado n vezes com média de probabilidade y e de impacto x; risco2, apontado m vezes com média de probabilidade y e de impacto x:]. Veja o [sumário da rodada. <LINK PARA SUMARIO RODADA>](#)

Figura 5.19 – Layout de uma História de Reunião de Elaboração Cooperativa de Planos de Riscos.

5.5 Considerações Finais do Capítulo

Neste capítulo foi apresentada a infra-estrutura de apoio à decisão em grupo de ODE. Essa infra-estrutura tem por objetivos principais apoiar a construção cooperativa de artefatos de software e capturar o raciocínio seguido durante a construção dos mesmos.

Visando a atingir o objetivo de apoiar a construção cooperativa de artefatos de software, a infra-estrutura foi concebida baseada em uma abordagem que generaliza a técnica Delphi, de forma a apoiar a realização cooperativa de quaisquer atividades do processo de software. Essa abordagem define a realização de uma reunião para construção cooperativa de artefatos de software por meio de um fluxo de trabalho iterativo composto por sete etapas. A proposta dessa abordagem também foi objeto deste capítulo.

A fim de alcançar o objetivo de capturar o raciocínio seguido durante a construção dos artefatos, foram empregadas técnicas de *Design Rationale* na construção da infra-estrutura. Assim, foi desenvolvido um modelo de representação utilizando uma abordagem baseada em argumentação.

Como resultado dos esforços para capturar o raciocínio seguido durante a construção dos artefatos, foi acrescentado à infra-estrutura de gerência de conhecimento de ODE um novo item de conhecimento: a história de reunião.

Outro requisito que norteou a concepção dessa infra-estrutura foi o de construir uma infra-estrutura flexível, facilmente especializável, de forma a minimizar o esforço necessário para prover, em ODE, apoio à construção cooperativa de outros tipos de artefatos.

Visando a experimentar a infra-estrutura, foi realizada uma especialização da mesma para apoiar a elaboração cooperativa de planos de riscos. Durante a construção dessa especialização, pôde-se constatar que a infra-estrutura atende ao requisito de flexibilidade, abrindo caminho para que o apoio à construção cooperativa de outros tipos de artefatos seja disponibilizado em ODE.

No próximo capítulo são apresentadas as conclusões finais desta dissertação além de perspectivas para trabalhos futuros.

Capítulo 6

Considerações Finais

O sucesso de um projeto de software depende muito da habilidade do gerente em estimar seus custos e prazos no início de seu desenvolvimento e controlá-los ao longo do processo de desenvolvimento. No entanto, devido à grande quantidade de variáveis e interesses envolvidos, gerenciar um projeto de software não é uma tarefa simples.

Dentre as características desejáveis em um bom gerente de projetos de software, merece destaque a experiência. Porém, para se adquirir experiência em gerência de projetos não basta participar de treinamentos e estudos sobre a área. A maior gama de experiências em gerência é acumulada através do contato direto com as atividades de gerência. Assim, é importante fornecer meios para que o gerente de projetos não fique limitado às suas próprias experiências.

Neste trabalho, procurou-se utilizar a gerência de conhecimento para apoiar a gerência de projetos em ODE, com o intuito de viabilizar a captura, o armazenamento e o reuso de experiências anteriores, assim como estimular a troca de experiências e a colaboração entre gerentes de projetos.

Neste capítulo, são apresentadas as considerações finais a respeito do trabalho realizado. A seção 6.1 apresenta as principais conclusões, destacando as contribuições do trabalho, enquanto a seção 6.2 enfoca as perspectivas futuras para continuidade deste trabalho.

6.1 Conclusões

Para a realização deste trabalho, foram estudadas as seguintes áreas: Gerência de Projetos de Software, Gerência de Conhecimento, Raciocínio Baseado em Casos, *Design Rationale* e Ambientes de Desenvolvimento de Software. A partir desses estudos, buscou-se propor formas

de apoio de gerência de conhecimento para a gerência de projetos de software no ambiente de desenvolvimento de software ODE.

Neste contexto, foram identificadas algumas oportunidades de melhoria e de integração de novos recursos a ODE, dentre as quais merecem destaque:

1. Aperfeiçoar o mecanismo de caracterização de projetos em ODE, estendendo esse mecanismo a outros itens de software;
2. Prover meios que permitissem a recuperação de itens de conhecimento relevantes para uma dada situação;
3. Apoiar a construção cooperativa de artefatos, incentivando a socialização do conhecimento através do compartilhamento de idéias;
4. Apoiar a captura do conhecimento tácito envolvido na elaboração dos artefatos de software;
5. Apresentar, de forma explícita, o raciocínio seguido pelos desenvolvedores durante a elaboração de um artefato.

Visando a atender aos requisitos indicados nos itens 1 e 2, foi desenvolvida a infra-estrutura de caracterização apresentada no Capítulo 4. Essa infra-estrutura acrescenta a ODE uma forma flexível de se caracterizar itens relacionados ao desenvolvimento de software, bem como provê um mecanismo que permite a recuperação de itens de software similares.

A idéia de se construir uma infra-estrutura que permitisse a recuperação de itens de software relevantes em um determinado contexto surgiu da percepção da importância do uso de dados históricos para apoiar a realização das atividades da gerência de projetos, em especial para a realização de estimativas.

Porém, nota-se que o uso de dados históricos e artefatos gerados em projetos anteriores similares extrapola a gerência de projetos, sendo útil para diversas atividades do processo de software. Assim, buscou-se generalizar a solução desenvolvida, de modo que a infra-estrutura pudesse ser utilizada para trazer benefícios a muitas outras atividades do processo de software apoiadas por ODE, não ficando restrita às atividades de gerência. Prova disso é a iniciativa de NARDI (2006) que, na ferramenta de apoio à Gerência de Requisitos, faz uso da infra-estrutura de caracterização para recuperar e exibir os requisitos e os modelos de objetos de projetos similares ao projeto em questão. A utilização por BERTOLLO (2006) da infra-estrutura de

caracterização desenvolvida nesse trabalho para caracterizar processos especializados de software é outro exemplo de aplicação dessa infra-estrutura.

Já para atender aos requisitos indicados nos itens 3, 4 e 5, foi construída e integrada a ODE uma infra-estrutura de apoio à decisão em grupo. Para a construção dessa infra-estrutura foi proposta uma abordagem para a realização de reuniões baseada na técnica Delphi. Para tentar traçar o raciocínio seguido pelos desenvolvedores durante a realização da atividade, foi utilizada uma abordagem baseada em argumentação.

Devido ao fato da experiência ser unanimemente apontada na literatura como uma das características desejáveis em um bom gerente de projetos de software, um dos objetivos que pautaram a criação da infra-estrutura de apoio à decisão em grupo foi apoiar a troca de experiências entre os participantes envolvidos na realização de uma determinada atividade.

Mas, a exemplo do que acontece com a infra-estrutura de caracterização desenvolvida neste trabalho, a aplicabilidade da infra-estrutura de apoio à decisão em grupo não fica restrita às atividades de gerência. Essa infra-estrutura pode ser utilizada para apoiar qualquer atividade que tenha como objetivo a construção cooperativa de um artefato.

Dessa maneira, durante este trabalho, foi integrado ao ambiente ODE um ferramental que emprega técnicas de gerência de conhecimento para apoiar a captura, recuperação, reutilização e compartilhamento de conhecimento.

Em suma, são contribuições deste trabalho:

- A construção, em ODE, de uma infra-estrutura genérica para caracterizar itens de software e calcular similaridade entre eles;
- A especialização da infra-estrutura de caracterização construída para aperfeiçoar as funcionalidades de caracterização de projetos e busca de projetos similares em ODE;
- A proposta de uma abordagem para a construção cooperativa de artefatos baseada na técnica Delphi;
- A construção de uma infra-estrutura de apoio à decisão em grupo em ODE;
- A especialização da infra-estrutura de apoio à decisão em grupo para apoiar a elaboração cooperativa de planos de riscos;
- O acréscimo à Gerência de Conhecimento de ODE de mais um tipo de item de conhecimento: a História de Reunião.

A utilização da Gerência de Conhecimento como forma de apoio à Gerência de Projetos é um tema muito amplo e impossível de ser esgotado em um único trabalho. Além disso, conforme anteriormente apontado, apesar deste trabalho ter sido conduzido visando a apoiar as atividades de gerência, o ferramental desenvolvido pode ser aplicado a diversas outras atividades do processo de software. Dessa forma, este trabalho abre espaço para uma série de novos trabalhos em ODE, sejam eles objetivando o apoio a atividades de gerência de projeto, sejam visando a utilizar as infra-estruturas aqui propostas para apoiar atividades de outros processos. Na próxima seção são apresentadas algumas dessas perspectivas de trabalhos futuros.

6.2 Perspectivas Futuras

Conforme citado anteriormente, a infra-estrutura de caracterização desenvolvida neste trabalho foi concebida de forma a apoiar a caracterização de diversos tipos de itens de software e especializada para permitir a caracterização de projetos. Dessa forma, trabalhos futuros devem especializar essa infra-estrutura visando a permitir a caracterização e a determinação do grau similaridade de outros itens de software, como por exemplo, módulos e atividades.

Como na infra-estrutura de caracterização desenvolvida foram tratadas apenas as etapas de representação e armazenamento de casos e recuperação de casos similares, esforços futuros podem ser despendidos a fim de tratar a adaptação de soluções dadas a problemas similares em projetos passados, com o objetivo de gerar uma solução inicial para um problema atual. Acredita-se não ser possível automatizar essa atividade de forma genérica, pois para adaptar soluções passadas a contextos atuais é necessário que sejam levadas em consideração características inerentes a uma atividade e ao escopo em questão, dentre outros fatores. Assim, vários trabalhos podem ser desenvolvidos nessa linha.

Outro trabalho interessante seria acrescentar à funcionalidade de busca a projetos similares a possibilidade de, dado um projeto similar, recuperar determinados artefatos do mesmo, como por exemplo estimativas, planos de riscos etc. O mesmo poderia ser feito para outros tipos de itens de software que venham a ser caracterizados. Uma primeira iniciativa nesse sentido foi realizada por NARDI (2006) que na ferramenta de apoio à gerência de requisitos provê ao desenvolvedor a possibilidade de visualizar os requisitos e os modelos de objetos dos projetos similares ao projeto em questão.

Ainda no contexto da caracterização de itens de software, pode ser interessante fazer uma pesquisa junto a profissionais com o objetivo de identificar as características a serem utilizadas para caracterizar cada um dos tipos de itens relativos ao desenvolvimento de software que terão a caracterização apoiada em ODE. Em tais trabalhos, deve-se buscar definir, também, o peso, de acordo com a atividade considerada, que cada uma dessas características terá sobre o cálculo do grau de similaridade entre dois itens.

No que tange à infra-estrutura de apoio à decisão em grupo, trabalhos futuros podem utilizá-la para acrescentar a ODE apoio à construção cooperativa de outros artefatos, como por exemplo, estimativas de esforço, de tamanho e de tempo.

Ainda no contexto da infra-estrutura de apoio à decisão em grupo, podem ser desenvolvidos trabalhos visando a fornecer um maior apoio ao moderador da reunião, principalmente no que se refere à elaboração de uma proposta de consenso. Estudos podem ser feitos no sentido de apoiar a geração das propostas de consenso com base nas propostas dos participantes. Tais estudos devem ser feitos levando em consideração as características inerentes ao artefato a ser gerado. Dessa forma, não se crê no desenvolvimento de uma funcionalidade genérica para automatização da geração de propostas de consenso que funcione para todas as especializações da infra-estrutura. O caminho é a realização de trabalhos que visem à automatização dessa tarefa para especializações da infra-estrutura que apoiem a construção cooperativa de artefatos específicos.

Neste trabalho foram despendidos esforços com o intuito de capturar o raciocínio seguido pelos participantes durante a construção cooperativa de artefatos, porém nada foi feito no que tange à captura do raciocínio durante a utilização convencional das ferramentas de ODE. Assim, novos estudos podem ser conduzidos a fim de planejar formas de capturar o raciocínio seguido pelo gerente de projeto durante a utilização das ferramentas de planejamento existentes em ODE.

Neste contexto acredita-se que a tecnologia de agentes pode ser bastante útil. Nesse sentido, um bom trabalho poderia ser a construção de agentes que monitorassem a utilização das ferramentas de apoio à gerência de projetos em ODE e sempre que julgassem pertinente, solicitassem ao gerente explicações sobre o raciocínio seguido para chegar a determinadas conclusões ou, ainda, sugerisse ao gerente que cadastrasse novos itens de conhecimento sobre tal contexto de trabalho.

Mas apenas capturar o conhecimento envolvido nessas tarefas não é suficiente. Dessa forma, a utilização de recursos pró-ativos para a disseminação de conhecimento aparece como uma boa opção. Em iniciativas anteriores foram implementados em ODE agentes que, com base nas ações dos usuários, agiam de forma pró-ativa, buscando e oferecendo itens de conhecimento potencialmente relevantes no contexto em questão. Para definir a relevância de um item de conhecimento, esses agentes utilizam um cálculo de similaridade bastante limitado. Assim, tais agentes podem, agora, ser aperfeiçoados pelo uso do cálculo do grau de similaridade entre itens de software provido pela infra-estrutura de caracterização desenvolvida neste trabalho.

Uma característica que chama a atenção quando são estudadas as atividades envolvidas na gerência de projetos é o alto grau de relacionamento e dependência que há entre essas atividades. Por exemplo, a identificação de novos riscos em um projeto pode afetar as estimativas de esforço, trazendo reflexos diretos na elaboração do cronograma e na alocação de recursos. Assim, é desejável que as ferramentas que apoiem tais tarefas estejam integradas de forma a facilitar o trabalho do gerente de projetos no trato de todas essas variáveis. Para alcançar tal objetivo, pode ser feito um estudo sobre a correlação entre essas variáveis a fim de identificar e propor possibilidades de melhoria na integração entre as ferramentas de planejamento de ODE. Mais uma vez, a tecnologia de agentes pode ser uma boa opção. Agentes poderiam monitorar a utilização das ferramentas de apoio à gerência de projetos e dar orientações, tal como, quando fosse efetuada alguma alteração no plano de riscos ou na alocação de recursos, recomendar que as estimativas fossem reavaliadas tendo em vista essas alterações.

Enfim, apesar de muito ter sido estudado e feito neste trabalho, muitos outros trabalhos podem surgir das idéias aqui discutidas.

Referências Bibliográficas

- AAMODT, A., PLAZA, E., “*Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*”, AICOM, Março de 1994.
- ABECKER, A., BERNARDI, A., HINKELMAN, K., “Toward a Technology for Organizational Memories”, *IEEE Intelligent Systems*, v. 13, n. 3 (Maio/Junho), pp. 40-48, 1998.
- ALTHOFF, K., BIRK, A., HARTKOPF, S., MULLER, W., NICK, M., SURMANN, D., TAUTZ, C., “Managing Software Engineering Experience for Comprehensive Reuse”, In: *Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering*, Kaiserslautern, Germany, Junho 1999.
- AMBRIOLA, V., CONRADI, R., FUGGETTA, A., “Assessing Process-Centered Software Engineering Environments”, *ACM Transactions on Software Engineering and Methodology*, v. 6, n. 3 (Jul), pp. 283-328, 1997.
- ARANTES, D. O., FALBO, R. A., “Gerenciando o Conhecimento Proveniente de Interações entre Membros de Organizações de Software”, II Workshop de Tecnologia de Informação e Gerência de Conhecimento, III Simpósio Brasileiro de Qualidade de Software - SBQS'2004, Brasília, Brasil, Junho 2004.
- ARAÚJO, R. M., DIAS, M. S., BORGES, M. R. S. “Suporte por Computador ao Desenvolvimento Cooperativo de Software: Classificação e Propostas”, XI Simpósio Brasileiro de Engenharia de Software - SBES'1997, Fortaleza, Brasil, Outubro 1997.
- ARMOUR, P., 2002, “*Then Unmyths of Project Estimation*”, *Communications of the ACM*, vol 45, nº 11, Nov, pp.15 – 18.
- BARCELLOS, M. P., “Planejamento de Custos em Ambientes de Desenvolvimento de *Software* Orientados à Organização”. Tese de D.Sc., Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2003.

- BARCELLOS, M.P., FIGUEIREDO, S.M., ROCHA, A.R.C., TRAVASSOS, G.H., 2003, “Utilização de Métodos Paramétricos, Analogias, Julgamento de Especialistas e Conhecimento Organizacional no Planejamento de Tempo e Custos de Projetos de Software”, In: *Anais do II Simpósio Brasileiro de Qualidade de Software*, pp. 17-31, Fortaleza, Brasil.
- BARRETO, A. O. S., “Apoio à Verificação de Software em Ambientes de Desenvolvimento de *Software* Orientados à Organização”. Tese de D.Sc., Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2006.
- BASILI, V., CALDIERA, G., ROMBACH, H., “The Experience Factory”, *Volume 1 of Encyclopedia of Software Engineering*, Chapter X, John Wiley & Sons. 1994.
- BENJAMINS, V. R., FENSEL, D., PÉREZ, A. G., “Knowledge Management through Ontologies”, Proc. of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM98), Switzerland, 1998
- BERTOLLO, G., FALBO R. A., “*Definição de Processos em um Ambiente de Desenvolvimento de Software Baseado em Ontologias*”. Anais do V Simpósio Brasileiro de Qualidade de Software, p. 72-86, Vila Velha, Brasil, Maio 2006.
- BIRK, A., DINGSOYR, T., STÁLHANE, T., 2002, “*Postmortem: Never Leave a Project without It*”, IEEE Software, May/Jun, pp. 43-45.
- BOEHM, B. W., ABTS, C., CHULANI, S., CLARK, B.K., HOROWITZ, E., MADACHY, R., REIFER, D., STEECE, B., 2000, “Software Cost Estimation with COCOMO II”, Prentice Hall.
- BOEHM, B.W., 2000, “Safe and Simple Software Cost Analysis”, IEEE Software, Sep/Oct, pp. 14-17.
- BORGES, L. M. S., FALBO, R. A., “*Managing Software Process Knowledge*”, In: *Proceedings of the CSITeA'2002*, Junho 2002.

- BROOKE, J., “*User Interfaces for CSCW Systems*”, In: *CSCW in practice: an Introduction and Case Studies*, Dan Dapier e Colston Sanger (eds.), Springer-Verlag. 1993.
- BUCKINGHAM, S. S., “*Design Argumentation as Design Rationale*”, The Encyclopedia of Computer Science and Technology (Marcel Dekker Inc: NY), Vol. 35 Supp. 20, 95-128.
- CARVALHO, V. A., FALBO, R. O., OLIVEIRA, L. O., “EstimaODE: Apoio a Estimativas de Tamanho e Esforço no Ambiente de Desenvolvimento ODE”, Anais do V Simpósio Brasileiro de Qualidade de Software, Vitória, Brasil, pp. 12-26, 2006.
- CHARISMATEK SOFTWARE METRICS, 2006, Function Point WORKBENCH
http://www.charismatek.com.au/_public1/index.htm
- CHRIS PTY LTD, 2006, FP Recorder, <http://www.fpreorder.com>
- CHRISSIS, M. B., KONTAD M., SHRUM S., “CMMI: Guidelines for Process Integration and Product Improvement”, Addison Wesley, 2003.
- CRUZ, C. D., “*Em Direção a um Modelo de Custos de Desenvolvimento de Software Orientado a Objetos*”, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, 1998a.
- CRUZ, T., “Workflow: A Tecnologia que Vai Revolucionar Processos”, São Paulo : Atlas, 1998b.
- DAL MORO, R., NARDI, J.C., FALBO, R.A., “*ControlPro: Uma Ferramenta de Acompanhamento de Projetos Integrada a um Ambiente de Desenvolvimento de Software*”. XII Sessão de Ferramentas do Simpósio Brasileiro de Engenharia de Software, SBES'2005, Uberlândia, Brasil, Outubro 2005.
- DAVENPORT, T. H., PRUSAK, L., “*Working Knowledge: How Organizations Manage What They Know*”. Harvard Business School Press, Boston, MA, 1998.
- DIENG, R., CORBY, O., GIBOIN, A., RIBIÈRE, M., “*Methods and Tools for Corporate Knowledge Management*”, In: Proceedings of the 11th Knowledge Acquisition, Modeling and Management Workshop, KAW'98, Banff, Canada, April 1999.

DIENG, R., “*Knowledge Management and the Internet*”, *IEEE Intelligent Systems*, vol. 15, n. 3 (May/June), pp.-14-17, 2000.

DUVESSA SOFTWARE, 2006, Estimate Easy UC, <http://www.duessa.com>

FALBO, R. A., “*Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*”, Rio de Janeiro, RJ, Tese de Doutorado, COPPE/UFRJ, 1998.

FALBO, R. A., “A Experiência na Definição de um Processo Padrão Baseado no Processo Unificado,” *Anais do II Simpósio Internacional de Melhoria de Processo de Software, SIMPROS’2000*. São Paulo – SP: Setembro 2000.

FALBO, R.A, GUIZZARDI, G., DUARTE, K.C., “An Ontological Approach to Domain Engineering”, In: *Proc. of the 14th Int. Conference on Software Engineering and Knowledge Engineering, SEKE’02*, Ischia, Itália, 2002.

FALBO, R.A., RUY, F.B., PEZZIN, J., DAL MORO, R., “Ontologias e Ambientes de Desenvolvimento de Software Semânticos”. 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering, JISIC’2004, Vol. I, 277-292, Madrid, Spain, November 2004a.

FALBO, R.A., RUY, F.B., BERTOLLO, G., TOGNERI, D.F., “*Learning How to Manage Risks Using Organizational Knowledge*”. *Proceedings of the 6th International Workshop on Advances in Learning Software Organizations, LSO’2004*, pp. 7-18, Banff, Canada, June 2004b.

FALBO, R.A., ARANTES, D.O., NATALI, A.C.C., “*Integrating Knowledge Management and Groupware in a Software Development Environment*”, *Proceedings of the 5th International Conference on Practical Aspects of Knowledge Management - PAKM’2004*, Karagiannis, D., Reimer, U. (Eds.): LNAI 3336, pp. 94-105, Springer-Verlag Berlin Heidelberg, Vienna, Austria, December 2004c.

FALBO, R. A., BERTOLLO, G., “Establishing a Common Vocabulary for Software Organizations Understand Software Processes”, *International Workshop on Vocabularies*,

Ontologies and Rules for the Enterprise, VORTE'2005, Enschede, The Netherlands, September 2005.

FARIAS, L. et al., 2003, "RiscManager: uma Ferramenta par Planejamento de Riscos com apoio de Gerência do Conhecimento", I Workshop Tecnologias da Informação e Gerência do Conhecimento, Fortaleza, Brasil.

FIORINI, S. T., STAA, A. V., BAPTISTA, R. M., "Engenharia de Software com CMM", Brasport, Rio de Janeiro, 1998.

FISCHER, G., OSTWALD, J., "Knowledge Management: Problems, Promises, and Challenges". *IEEE Intelligent Systems*, v. 16, n. 1 (Janeiro/Fevereiro), pp. 60-72 2001.

FUGGETTA, A., "Software Process: A Roadmap", In: *Future of Software Engineering*, A Finkelstein (ed). 2000.

GARMUS, D., HERRON, D., 2001, "*Function Point Analysis: Measurement Practices for Successful Software Projects*", Addison Wesley.

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

GRUDIN, J., "Computer-supported cooperative work: history and focus", *IEEE Computer*, v. 27, n.5, p. 19-26, mai. 1994.

GUIZZARDI, G., FALBO, R.A., PEREIRA FILHO, J.G., "Using Objects and Patterns to Implement Domain Ontologies", *Anais do XV Simpósio Brasileiro de Engenharia de Software*, Outubro de 2001.

HARRISON, W., OSSHER, H., TARR, P., "*Software Engineering Tools and Environments: A Roadmap*", Proc. of The Future of Software Engineering, ICSE'2000, Irlanda, 2000.

HEIJST, G., SPEK, V., KUIZINGA, E., "*Organizing Corporate Memory*", In: Proc. 10th Banff Workshop on Knowledge Acquisition for Knowledge-Based Systems (KAW 96), Canada, 1996.

- HOLZ, H., KÖNNECKER, A., MAURER, F., “*Task-Specific Knowledge Management in a Process-Centred SEE*”. In: *Advances in Learning Software Organizations*, v. 2176, Lecture Notes in Computer Science, Springer, pp. 163-177, 2001.
- HOUDEK, F., BUNSE, C., “Transferring Experience: A Practical Approach and its Application on Software Inspections”, In: *Proc. of SEKE Workshop on Learning Software Organizations*, Kaiserslautern, Alemanha, 1999.
- IDRI, A., ABRAN, A., 2001, “*A Fuzzy Logic Based Set of Measures for Software Projects Similarity: Validation and Possible Improvements*”, *Software Metrics Symposium – METRICS*, pp. 85-96.
- IEEE Std 1540-2001, “*IEEE Standard for Software Life Cycle Processes – Risk Management*”, 2001.
- ISO/IEC 12207 Information Technology – Software life cycle processes, 1995.
- ISO/IEC 12207 Information Technology – Software life cycle processes, Amendment 1, 2002.
- ISO/IEC 12207 Information Technology – Software life cycle processes, Amendment 2, 2004.
- ISO/IEC 15504 Information Technology – Process Assessment – Part 1: Concepts and Vocabulary, 2003.
- JABLONSKI, S., HORN, S., SCHLUNDT, M., “Process Oriented Knowledge Management”. In: *Proceedings of the 11th International Workshop on Research Issues in Data Engineering*, pp. 77-84, Heidelberg, Germany, Apr. 2001.
- JAUFMANN, O., FREIMUT, B., RUS, I., “Reusing knowledge on software quality for developing measurement programs” In: *Proc. of the 16th Int. Conference on Software Engineering and Knowledge Engineering*, SEKE’04, Banff, Canadá, 2004.
- JONES, C., 2000, “*Software Assessments, Benchmarks and Best Practices*”, Addison-Wesley Information Technology Series.

- JØRGENSEN, M., 2004, “A Review of Studies on Expert Estimation of Software Development Effort”, *The Journal of Systems and Software*, ed. 70, pp. 37-60.
- KARNER, G., “Metrics for Objectory”, Diploma Thesis, University of Linköping, Sweden, December 1993.
- KNOB, F., ORTH, A. I., PRIKLADNICKI, R., SILVEIRA F., “RiskFree – Uma Ferramenta de Gerenciamento de Riscos Baseada no PMBOK e Aderente ao CMMI”, *Anais do V Simpósio Brasileiro de Qualidade de Software*, Vitória, Brasil, pp. 203-217, 2006.
- KRUCHTEN, P., “The Rational Unified Process – An Introduction”, Addison-Wesley, 3rd Edition, 2003.
- KURTZ, T., 2001, “*Ask Pete, Software Planning and Estimation Through Project Characterization*”, *Requirements Engineering, Fifteenth IEEE International Symposium*, pp. 286-287.
- LEE, J. , “*Design Rationale Systems: Understanding the Issues*”, *IEEE Expert*, Vol. 12, No. 3, pp. 78-85, 1997.
- LIMA, K.V.C., 2004, “Definição e Construção de Ambientes de Desenvolvimento de Software Orientados à Organização”, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.
- MARKKULA, M., “Knowledge Management in Software Engineering Projects”, In: *Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering*. Kaiserslautern, Germany, Junho 1999.
- MARTINS, J.C.C., “Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML”, 2ª edição. Rio de Janeiro: Brasport, 2005.
- MAURER, F., HOLZ, H., “*Integrating Process Support and Knowledge Management for Virtual Software Development Teams*”, *Annals of Software Engineering*, v. 14, n. 1-4, pp. 145-168, 2002.

- MEDEIROS, A. P., 2006, “Kuaba: Uma Abordagem para Representação e Reuso de Design Rationale em designs Baseados em Modelo”, Tese de D. Sc., PUC - RJ, Rio de Janeiro, RJ, Brasil.
- MENDES, E., COUNSELL, S., 2000, “Web Development Effort Estimation Using Analogy”, Software Engineering Conference, pp. 203-212.
- MENZIES, T., SINSEL, E., 2000, “*Practical Large Scale what-if Queries: Case Studies with Software Risk Assessment*”, Automated Software Engineering, Fifteenth IEEE International Conference, pp. 165-173.
- MONTONI, M., MIRANDA, R., ROCHA, A.R., TRAVASSOS, G.H., 2004, “Knowledge Acquisition and Communities of Practice: an Approach to Convert Individual Knowledge into Multi-organizational Knowledge”, In: *Proceedings of the LSO 2004*, pp. 110-121, Banff, Canadá.
- NARDI, J. C., 2006, “Apoio de Gerência de Conhecimento à Engenharia de Requisitos em um Ambiente de Desenvolvimento de Software”, Tese de M. Sc., UFES, Vitória, ES, Brasil.
- NATALI, A. C. C., 2003, “Uma Infra-Estrutura para Gerência de Conhecimento em um Ambiente de Desenvolvimento de Software”, Tese de M. Sc., UFES, Vitória, ES, Brasil.
- NATALI, A. C. C., FALBO, R. A., “Gerência de Conhecimento em ODE”, Anais do XVII Simpósio Brasileiro de Engenharia de Software, Manaus, Outubro de 2003.
- NETO, M. G. M., SEAMAN, C. B., BASILI, V., KIM, Y., “A Prototype Experience Management System for a Software Consulting Organization.” In: *Proc. of the 13th Int. Conference on Software Engineering and Knowledge Engineering*, SEKE’01, Buenos Aires, Argentina, 2001.
- NONAKA, I., TAKEUCHI, H., “Criação de Conhecimento na Empresa –Como as Empresas Japonesas Geram a Dinâmica da Inovação”, Campus, 1997.
- O’LEARY, D., “*How Knowledge Reuse Informs Effective Systems Design and Implementation*”, *IEEE Intelligent Systems*, v. 16, n. 1 (Janeiro/Fevereiro), pp. 44-49. 2001.

- OLIVEIRA A. C., 2006, “Histórias Coletivas de Uso de Sistemas: Capturando Conhecimento Tácito para Manutenção”, Tese de M. Sc., UFRJ/IM/NCE , Rio de Janeiro, RJ, Brasil.
- OLIVEIRA, K., “Modelo para Construção de Ambientes de Desenvolvimento Orientados a Domínio”, Tese de DSc. da COPPE/UFRJ, Rio de Janeiro, Brasil, Out 1999.
- PERRET, R., 2004, “A técnica de group storytelling aplicada à Gestão do Conhecimento.”, Tese de M. Sc., UFRJ/IM/NCE, Rio de Janeiro, RJ, Brasil.
- PFFLEGER, S.L., *Software Engineering: Theory and Practice*, 2nd Edition, New Jersey: Prentice Hall, 2001.
- PMI – *Project Management Institute*, “Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos – Guia PMBOK – *Project Management Body of Knowledge*”, 3^a ed., 2004.
- POLTROCK, S., GRUDIN, J., “*Computer-Supported Cooperative Work and Groupware*”, In: *Proc. of the Conference Companion, CHI’94*, Boston, EUA, 1994.
- PRESSMAN, R. S., 2002, “*Engenharia de Software*”, McGraw-Hill, 5^a Ed..
- RAMESH, B., “Process Knowledge Management with Traceability”, *IEEE Software*, v. 19, n. 3, pp. 50-52, May/Jun. 2002.
- REIFER, J. D., “A Little Bit of Knowledge is a Dangerous Thing”, *IEEE Software*, v. 19, n. 3, pp. 14-15, May/Jun. 2002.
- ROCHA, A. R. C., AGUIAR, T. C., SOUZA, J. M., 1990, “*TABA: A Heristic Workstation for Software Development*”, In: *Proceedings of COMPEURO’90*, Tel Aviv, Israel.
- RUS, I., LINDVALL, M., and SINHA, S., “Knowledge Management in Software Engineering”, *DACS State-of-the-Art-Report*, 2001.
- RUS, L., LINDVALL, M., “*Knowledge Management in Software Engineering*”, *IEEE Software*, May/June 2002, pp. 26 – 38.

- RUY, F. B., 2006, “Semântica em um Ambiente de Desenvolvimento de Software”, Tese de M. Sc., UFES, Vitória, ES, Brasil.
- SCHNAIDER, L., 2003, “Planejamento da Alocação de Recursos Humanos em Ambientes de Desenvolvimento, de Software Orientados à Organização”, Dissertação de M. Sc., COPPE/UFRJ, Rio de Janeiro, Brasil.
- SCHNEIDER, K., HUNNIUS, J., BASILI, V. R., “Experience in Implementing a Learning Software Organization”, *IEEE Software*, May/June 2002, pp. 46 – 49.
- SCHREIBER, G., AKKERMANS, H., ANJEWIERDEN, A., de HOOG, R., SHADBOLT, N., de VELDE, W.V., WIELINGA, B., *Knowledge Engineering and Management: The CommonKADS Methodology*, MIT Press, Massachusetts, 1999.
- SENGE, P., KLEINER, A., ROBERTS, C., ROSS, R., ROTH, G., SMITH, B., “A Dança das Mudanças”, 7ª Ed., Rio de Janeiro: Campus, 1999.
- SHEPPERD, M., SCHOFIELD C., 1997, “*Estimating Software Project Effort Using Analogies*”, *Software Engineering*, *IEEE Transactions on*, V. 23, Issue 11, Nov 1997, pp. 736-743.
- SKYRME, D., “Knowledge Management Solutions – The IT Contribution. *SIGGROUP Bulletin*”, v. 19. n. 1 (Abril). pp. 34-39. 1998.
- SOFTEX, “MPS.BR – Melhoria de Processo do Software Brasileiro – Guia Geral”, v. 1.1, Maio de 2006.
- SOMMERVILE, I., *Engenharia de Software*, São Paulo: Addison-Wesley, 6ª edição, 2003.
- SPEK, R. v.d. and SPIJKERVET, A., *Knowledge Management: Dealing Intelligently with Knowledge*, CIBIT, Utrecht, 1997.
- STAAB, S., STUDER, R., SCHNURR, H., SURE, Y., “Knowledge Process and Ontologies”. *IEEE Intelligent Systems*. v. 16. n. 1 (Janeiro/Fevereiro). pp. 26-34. 2001.
- STENMARK, D., “Leveraging Tacit Organisational Knowledge”, *Journal of Management Information Systems*, Vol. 17, No. 3, Winter 2000-2001, pp. 9-24, 2001.

TOGNERI, D. F., 2002, “*Apoio Automatizado à Engenharia de Requisitos Cooperativa*”. Tese de M.Sc., Programa de Mestrado em Informática, UFES, Vitória, ES, Brasil.

TORRES, A. H. S., 2006, “Captura e Disseminação do Conhecimento em Projetos de Software”, Tese de M. Sc., Universidade Católica de Brasília, Brasília, DF, Brasil.

TRAVASSOS, G. H., “*O Modelo de Integração de Ferramentas da Estação TABA*”. Tese de D.Sc., Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1994.

VALENTE, F. F. R., 2003, “Uso de Gerência de Conhecimento para Apoiar a Realização de Estimativas”, Tese de M. Sc., UFES, Vitória, ES, Brasil.

VASCONCELOS JR., Francisco Moacyr, BRAGA, Henry Rangel, TRAVASSOS, Guilherme Horta, WERNER, Cláudia Maria Lima. *Representação e Controle do Processo em Ambientes de Desenvolvimento de Software*. 1995.

VILLELA, K., SANTOS, G., TRAVASSOS, G., Rocha, A.R., “Melhoria de Processos de Software e Evolução de Ambientes de Desenvolvimento de Software com base no Conhecimento do Domínio e na Cultura Organizacional”, *Simpósio Brasileiro de Qualidade de Software*, Gramado, Brasil, 2002.

VILLELA, K., “Definição e Construção de Ambientes de Desenvolvimento de *Software* Orientados a Organização”. Tese de D.Sc., Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2004.

WANGENHEIM, C. G., WANGENHEIM A., 2003, “*Raciocínio Baseado em Casos*”, Manole, 1ª Ed..

WIIG, K., “Comprehensive Knowledge Management”,
http://www.knowledgeresearch.com/downloads/compreh_km.pdf, Knowledge Research Institute, Inc., 1999.

Formatado: Cor da fonte:
Branco