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

Uso de Sistema de Gerência de *Workflow* para Apoiar o Desenvolvimento de Software Baseado no Processo Unificado da Rational Estendido para Alcançar Níveis 2 e 3 do Modelo de Maturidade

por
LISANDRA VIELMO MANZONI

Trabalho de Conclusão de Mestrado submetido à avaliação,
como requisito parcial, para a obtenção do grau de
Mestre em Informática

Prof. Dr. Roberto Tom Price
Orientador

Porto Alegre, dezembro de 2001.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Manzoni, Lisandra Vielmo

Uso de Sistema de Gerência de *Workflow* para Apoiar o Desenvolvimento de Software Baseado no Processo Unificado Rational Estendido para Alcançar Níveis 2 e 3 do Modelo de Maturidade / por Lisandra Vielmo Manzoni. – Porto Alegre: PPGC da UFRGS, 2001.

202 f.:il.

Trabalho de Conclusão (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2001. Orientador: Price, Roberto Tom.

1.Rational Unified Process. 2.Capability Maturity Model. 3.Processos de desenvolvimento de *software*. 4.Avaliação de Processos de Software. 5.Ambiente de Engenharia de Software. 6.Sistemas de Gerência de *Workflow*. 7.Exchange 2000 Server. I Price, Roberto Tom. II. Título.

Agradecimentos

Agradeço a Deus, por sempre estar ao meio lado e me dar vontade de seguir em frente, mesmo nos momentos mais difícies.

Agradeço ao meu orientador, Prof. Dr. Roberto Tom Price, por sua dedicação, paciência, e confiança no trabalho que estávamos realizando.

Agradeço aos colegas do Projeto Amadeus, pela troca de idéias e pelo compartilhamento do conhecimento.

Agradeço ao Paiva, gerente do SERPRO; a Angélica, líder de nossa equipe e demais colegas de trabalho, pelo apoio e compreensão recebidos.

Agradeço ao Adriano, por seu amor, paciência e apoio. Sua visão simples e prática da vida foi muito importante para vencer mais esta etapa.

Agradeço aos meus pais, minhas irmãs e amigos, que souberam entender os motivos pelos quais muitas vezes tive que estar ausente.

E , por fim, agradeço a todos que me ajudaram a tornar este trabalho uma realidade.

Sumário

Lista de Abreviaturas

ADO	*ActiveX Data Objects*
ADSI	*Active Directory Directory Service Interfaces*
AE	Atividades Executadas (característica-comum)
AGP	Ambiente de Gerenciamento de Projetos
APES	Autoridade de Processo de Engenharia de Software
ASP	*Active Server Pages*
CCA	Conselho de Controle de Alteração
CDO	*Collaboration Data Objects*
CE	Compromisso de Executar (característica-comum)
CMM	*Capability Maturity Model.*
CMMI	*Capability Maturity Model Integration*
COCOMO	*Constructive Cost Model*
CSCW	*Computer Supported Cooperative Work*
DoD	*Department of Defense*
EPOS	*Expert System for Program and System Development.*
GAC	Gerenciamento de Alteração e Configuração
GC	Gerenciamento de Configuração
GCS	Gerenciamento de Configuração de Software
GQS	Garantia de Qualidade de Software
GSA	Gerenciamento de solicitação de alteração
HE	Habilidade de Executar (característica-comum)
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IIS	Internet Information Service
ISO	*International Standards Organization*.
KPA	*Key Process Area*
ME	Medição e Análise (característica-comum)
MTS	Microsoft Transaction Server
OMT	*Object Modeling Technique.*
OOSE	*Object-Oriented Software Engineering.*
OWA	*Outlook Web Access*
PDS	Plano de Desenvolvimento de Software
PGQ	Plano de Garantia de Qualidade
PGS	Plano de Gerenciamento de Subcontratos
PRA	*Project Review Authority*
PSEE	*Process-Centered Software Engineering Environment.*
RUP	*Rational Unified Process* (Processo Unificado Rational)
SEE	*Software Engineering Environment*
SEI	*Software Engineering Institute.*
SGW	Sistema de Gerenciamento de *Workflow*
SPICE	*Software Process Improvement and Capability Determination.*
SQA	*Software Quality Assurance*

SQL	*Structured Query Language* – Linguagem de Consulta Estruturada
SQUID	*Software Quality In the Development*
SW-CMM	*Software Capability Maturity Model*
UML	*Unified Modeling Language.*
URL	*Universe Resource Locator*
VI	Verificação de Implementação (característica-comum)
WfMC	*Workflow Management Coalition*
WMS	*Workflow Management System*
WSS	*Web Storage System*
XML	*eXtensible Markup Language*

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho descreve a avaliação do Processo Unificado Rational (RUP) realizada com base no Modelo de Maturidade da Capacitação (CMM ou SW-CMM), e a utilização de um sistema de gerência de *workflow* comercial, Exchange 2000 Server, na implementação de um protótipo de um ambiente de apoio a este processo, chamado de Ambiente de Gerenciamento de Projetos (AGP).

O Processo Unificado Rational (RUP) foi avaliado com relação às práticas-chave descritas pelo Modelo de Maturidade da Capacitação (CMM) do *Software Engineering Institute* (SEI), da *Carnegie Mellon University*. A avaliação identificou o suporte fornecido por este modelo de processo às organizações que desejam alcançar níveis 2 e 3 do CMM. A avaliação resultou na elaboração de propostas para complementar as macro-atividades (*Core Workflows*) do RUP, visando satisfazer as práticas-chave do CMM.

O CMM apresenta um modelo de avaliação de processo que busca atingir a maturidade dos processos da organização, é específico para o desenvolvimento de software, os aspectos de melhoria contínua são fortemente evidenciados e várias organizações já estão utilizando-o com sucesso. O RUP surgiu como uma proposta de unificar as melhores práticas de desenvolvimento de software.

Foi experimentada a utilização de um sistema de gerência de *workflow*, de fato um servidor de colaboração, para apoiar o processo de desenvolvimento de software. A ferramenta desenvolvida foi avaliada com base em requisitos considerados, por alguns autores da área, desejáveis em um ambiente de apoio ao processo de desenvolvimento.

O protótipo do ambiente de gerenciamento de projetos é uma ferramenta de suporte baseada na *Web*, que visa auxiliar os gerentes de projeto de software nas atividades de gerenciamento e controle, e ajudar na interação e troca de informações entre os membros da equipe de desenvolvimento.

O Processo Unificado apresenta uma abordagem bem-definida dos processos de engenharia de software e de gerenciamento de projetos de software, mas não se concentra em atividades de gerenciamento de sistemas. Ele apresenta lacunas em atividades envolvendo gerenciamento de recursos humanos, gerenciamento de custos e gerenciamento de aquisição.

AGP é uma ferramenta flexível que pode ser acessada pela Internet, suporta a colaboração entre os membros de uma equipe, e oferece os benefícios da Web, como navegação intuitiva através de links e páginas. Esta ferramenta ajuda no suporte ao gerenciamento, fornecendo opções para planejar e monitorar o projeto, e suporta eventos, como mudança de estados, e comunicação aos usuários de suas novas tarefas.

Palavras-chaves: *Capability Maturity Model*, *Rational Unified Process*, Modelo de Processo de Software, Avaliação de Processos de Software, Ambientes de Apoio a Processos de Software, Sistema de Gerência de *Workflow*, Exchange 2000 Server.

TITLE: "USING A WORKFLOW MANAGEMENT SYSTEM TO SUPPORT SOFTWARE DEVELOPMENT BASED ON EXTENDED RATIONAL UNIFIED PROCESS TO REACH MATURITY MODEL LEVELS 2 AND 3"

Abstract

This master dissertation describes the assessment of the Rational Unified Process (RUP) based on the Capability Maturity Model for Software (SW-CMM or CMM), and the implementation of a prototype tool to support this process based on of-the-shelf Workflow Management System, Exchange 2000 Server. The prototype developed is called Project Management Environment (PME).

Rational Unified Process (RUP) was assessed based on the key practices described for the Capability Maturity Model (CMM) at the Carnegie Mellon Software Engineering Institute. The assessment identified the facilities that RUP offers to support an organization aiming at CMM levels 2 and 3. The assessment resulted in the elaboration of propositions to complement the Rational Unified Process in order to satisfy the key process areas of CMM.

CMM shows a process model that is far fetched to reach the process maturity of an organization, is specific for the software development, and strongly emphasizes the aspects of continuous improvement and several organizations already used it with success. RUP describes how to apply best practices of software engineering.

It was experimented the use of a Workflow Management System, in fact a collaboration server, to support the software development process. The experimental environment was assessed considering the requirements identified by various researchers for an environment to effectively support a software development process.

The prototype software development environment is a web-based process support system, which provides means to assist the management of software development projects and help the interaction and exchange of information between disperse members of a development.

The Rational Unified Process presents a well defined approach on software project management and software engineering processes, but it is not an approach centered on systems management concerns. Therefore it lacks activities involving issues as cost management, human resource management, communications management, and procurement management.

PME is a flexible tool that can be accessed through the Internet, supporting the collaboration between team members, and offering the benefits of the Web, with intuitive navigation through of links and pages. It helps to support management control, providing options to plan and monitor the project, and supports events of the process, as changing states, and communicates users of their attributed tasks.

Keywords: Capability Maturity Model, Rational Unified Process, Software Process Model, Software Process Assessment, Software Engineering Software, Workflow Management System, and Exchange 2000 Server.

1 Introdução

Durante as últimas décadas, a necessidade de produzir produtos de software de alta qualidade tem aumentado a complexidade e dificultado o gerenciamento dos processos de desenvolvimento de software. Uma das razões é a rápida evolução das tecnologias e dos métodos para desenvolver software, junto com um aumento da complexidade das aplicações desenvolvidas [FIT 99].

Outro aspecto, que dificulta o gerenciamento do processo de desenvolvimento, é a ausência, em muitas organizações, de uma descrição formal do processo de software. A descrição formal de um processo permite que o mesmo seja avaliado, compreendido e automatizado (executado) [GIM 94].

A avaliação dos processos de software é uma prática já executada por várias empresas e cada vez mais necessária. A organização pode utilizar um modelo de avaliação para classificar o processo de desenvolvimento de acordo com níveis de maturidade ou capacitação, e então estabelecer prioridades de melhoria.

Sendo o desenvolvimento de software um trabalho cooperativo entre pessoas, que implica em alta integração de ferramentas e alto nível de descrição das tarefas que compõem as atividades do processo [OCA 98], identifica-se a necessidade de prover um ambiente de desenvolvimento que integre ferramentas, pessoas e o processo.

Com esse objetivo, surgiram os ambientes de Engenharia de Software Centrados no Processo de Desenvolvimento (PSEE), que se caracterizam por suportar a descrição e a execução de processos, de modo a auxiliar e controlar todas as atividades envolvidas na produção e na manutenção de um software [BAN 96, FIN 94, GIM 94].

Mas, verifica-se que poucos desses ambientes se tornaram produtos de mercado, e muitos não suportam integração com ferramentas de terceiros, e muitas vezes usam metodologias de desenvolvimento específicas [OCA 98, CHA 97a].

Existem vários estudos que analisam a aplicação de sistema de gerenciamento de *workflow* (SGW) no suporte a processos de software [CHA 97, CHA 97a, OCA 98, ARA 99, MAN 96]. Um dos fatores que tem levado a utilização deste tipo de sistema na área de desenvolvimento de software, é a flexibilidade oferecida por estes sistemas e a possibilidade de integrar ferramentas de diferentes fabricantes.

Um SGW pode ser definido como “um sistema que define, cria e executa *workflows* por meio da utilização de um software”, e *workflow* é “a automação de um processo de negócio, no todo ou em parte” [WFM 99].

No caso de desenvolvimento de software, o processo de negócio é o processo de desenvolvimento de software, e o fluxo de trabalho é definido pelo modelo de processo de software utilizado.

Então, faz-se necessário definir o modelo de processo que será modelado no SGW. Este modelo descreve a seqüência das atividades, o papel associado a atividade (descrito no RUP por meio de *workers*) e a ferramenta que deve ser invocada na execução da atividade.

O Processo Unificado surgiu como uma tentativa de unificar as melhores práticas de desenvolvimento de software. RUP propõe um processo iterativo e incremental para controlar o processo de desenvolvimento, e descreve como utilizar a UML de forma efetiva e assim usufruir toda a sua potencialidade [JAC 98, KRU 2000, RUP 2001].

Após a análise de vários modelos de avaliação, verificou-se que o CMM (e seu sucessor SW-CMM) [SEI 2000] apresenta um modelo completo e compreensivo em busca da maturidade dos processos organizacionais. É um modelo bastante utilizado e já consagrado em várias organizações de software [FIT 99].

Este trabalho analisa o Processo Unificado Rational descrevendo onde ele oferece suporte as práticas descritas pelo CMM, e onde a organização deve desenvolver esforços para complementar o RUP com relação às práticas do CMM não satisfeitas.

Em vários pontos na literatura referente ao RUP [RAT 98, KRU 2000, JAC 98, RAT 2001] está descrito que RUP é "um *framework* de processo", em outros que RUP é "um processo", e o mais confuso ainda, que RUP é "um processo e um *framework* de processo", confundindo o leitor com relação ao sentido dado no RUP a palavra "*framework*".

Após o estudo do RUP, concluiu-se que o *framework* referido pelos autores de RUP, é a totalidade dos processos descritos em RUP, o qual pode ser configurado para o processo padrão da organização ou para processos de projetos específicos (subconjunto adaptado do RUP). Essa conclusão tem como base, a afirmação "em muitos casos, o Processo Unificado Rational é o processo padrão da organização", descrita em "*Concepts: Process Configuration*" [RAT 2001]. Neste caso, a organização usa o RUP completo. A avaliação descrita neste trabalho considerou o *framework* do processo, isto é, o RUP completo.

O resultado da avaliação pode ajudar as organizações que desejam alcançar níveis 2 e 3 de CMM e estejam utilizando, ou venham a utilizar o RUP, por identificar as práticas-chave do CMM suportadas pelo RUP, e as práticas-chave que a organização deve definir, porque estão fora do escopo do RUP.

A identificação dos elementos do RUP que estão suportando as práticas do CMM, possibilita também, que as organizações que tenham processos definidos, não baseados no RUP, e que pretendam alcançar níveis de CMM mais altos, adotem subconjuntos do RUP para melhorar ou implementar áreas-chave deficientes da organização.

A avaliação pode auxiliar na customização do RUP para projetos específicos, porque na customização é possível eliminar atividades do *framework* de processo e não é desejável que os elementos que estão apoiando práticas do CMM sejam eliminados, se a organização tem intenção de atingir níveis mais altos de CMM.

Após a verificação das práticas-chave do CMM que não são suportadas pelo RUP, elaborou-se propostas de extensão ao RUP, para que este suporte as práticas-chave dos níveis 2 e 3 do CMM. As propostas são descritas usando diagramas de atividades, seguindo o modelo de apresentação do RUP. Não é objetivo do trabalho descrever guias para elaboração das atividades propostas e nem modelos (*templates*) para os artefatos propostos.

Após a avaliação do RUP realizou-se um experimento, onde o RUP estendido foi modelado por meio de um SGW comercial, no caso o Microsoft Exchange 2000 Server,

visando identificar a aplicabilidade de SGW comerciais na implementação de ferramentas para apoio a processo de software.

O protótipo desenvolvido é baseado na *Web*, podendo ser acessado por meio de uma URL específica. O protótipo possibilita o compartilhamento das informações do projeto entre os membros da equipe, o planejamento e o acompanhamento dos projetos de software, a distribuição de atividades entre as pessoas da equipe, e a atualização dos dados sobre as atividades, etc.

Sistemas de gerência de *workflow* têm seu foco em processos com finalidades mais gerais, provendo flexibilidade para modelar processos de negócio e permitindo alterações dinâmicas nesses.

O capítulo 2 descreve a situação atual em termos de processos de software, modelos de avaliação de processos, ambientes de apoio ao processo de software. Na seção 2.1 são descritos alguns trabalhos relacionados, visando descrever o contexto onde esse trabalho está inserido.

No capítulo 3, é descrito o Processo Unificado Rational, em termos de suas macro-atividades, atividades, trabalhadores e artefatos.

No capítulo 4 são descritos alguns modelos de avaliação de processos. É realizada uma breve explanação do CMM em termos de sua origem, características, estrutura e principais componentes.

No capítulo 5 é descrita a avaliação do RUP com base nas práticas-chave do CMM níveis 2 e 3, e a análise dos resultados obtidos na avaliação.

O capítulo 6 mostra as propostas de extensão ao RUP, elaboradas com base nos resultados obtidos na avaliação.

O capítulo 7 mostra conceitos relacionados a sistemas de gerência de *workflow*, a arquitetura genérica proposta pela WfMC, alguns sistemas de gerência de *workflow* comerciais e as principais características do Microsoft Exchange Server.

O capítulo 8 descreve o protótipo do ambiente de apoio ao processo implementado, e a avaliação da ferramenta considerando os requisitos desejáveis a um ambiente de apoio ao processo de software, e o capítulo 9 conclui este trabalho e apresenta os trabalho futuros.

2 Estado Atual

A finalidade de engenharia de software é desenvolver produtos de alta qualidade, em tempos e custos razoáveis. Mas, desde a chamada "Crise de Software", o que se vê nas organizações são cronogramas longos e instáveis, sempre em atraso e em modificação, estimativas erradas, baixa produtividade das pessoas e baixa qualidade dos produtos [ELL 96]. O aumento da efetividade dos processos de software é um passo longo e óbvio para enfocar a crise de software [FIT 99].

O processo de desenvolvimento de software, ou processo de software [GIM 94], corresponde ao conjunto de atividades que são desempenhadas pelos desenvolvedores desde a concepção, até a liberação do produto. Uma forma de se analisar e amadurecer tal processo é por meio de sua descrição, a qual consiste de um modelo de processo de software.

Muitos modelos de processo de software existem, como o Modelo Espiral [PRE95, HUM 90], Cascata [PRE 95], Prototipação [PRE 95], Processo Unificado Rational [RAT 99, RAT 2001] e o *Open Process Framework* (OPEN) [HEN 2000, HEN 99]. Os dois últimos são mais específicos para processos orientados a objetos [HEN 99].

Cada modelo de processo foca em diferentes funcionalidades e pode ser melhor adaptado para certos tipos de projetos de software.

O Processo Unificado Rational [JAC 98] é um processo de engenharia de software que provê uma abordagem disciplinada para assinalar tarefas e responsabilidades dentro de uma organização de desenvolvimento [KRU 2000]. Seu objetivo é assegurar a produção de software de alta qualidade, que encontra as necessidades de seus usuários finais, dentro de cronograma e orçamento previsíveis [JAC 98, KRU 2000]. Este modelo de processo foi escolhido para ser utilizado neste trabalho, porque representa um esforço de unificação das melhores práticas em desenvolvimento de software existentes, e é um guia de como efetivamente utilizar a Linguagem de Modelagem Unificada (UML) [JAC 98, RAT 2001].

Nos últimos anos, o interesse das organizações que desenvolvem software não tem sido somente no uso de modelos de processo de desenvolvimento de software, cada vez mais as organizações estão interessadas em avaliar e melhorar seus próprios processos. A prova deste interesse pode ser verificada pela quantidade de modelos de avaliação de processos encontrada na literatura [FIT 99, KRI 99], ilustrada pelo CMM [PAU 93, PAU 93a, KRI 99], Bootstrap [HAA 94], Trillium [BEL 94] e SQUID [BOE 99] e os esforços de padronização como as normas ISO9000-3 [ISO 91], ISO/IEC12207 [ISO 95] e ISO/IEC15504 [ISO 95a].

O *Capability Maturity Model* (CMM), desenvolvido pela *Carnegie Mellon University Software Engineering Institute* (CMU/SEI), provê um guia para selecionar estratégias de melhoria de processo por determinar as capacidades do processo e identificar as questões mais críticas para melhoria de processo e qualidade de software.

A primeira versão do CMM foi desenvolvida em 1991, enquanto a última versão (SW-CMM versão 1,1), disponível no *site* da SEI, foi publicada em 1993 [PAU 93]. A nova versão do CMM, SW-CMM versão 2.0, está sendo desenvolvida, mas ainda encontra-

se em uma versão *draft*. A versão utilizada na avaliação descrita neste trabalho foi a versão 1.1.

A escolha do CMM é porque ele é um dos mais bem conhecidos modelos de avaliação da maturidade de processos de software [FIT 99], apresenta uma estrutura detalhada de processo de avaliação de software e sua usabilidade prática tem sido demonstrada por vários casos de avaliação [HER 94]. CMM apresenta um modelo de processo que busca alcançar a maturidade dos processos de uma organização, é específico para desenvolvimento de software, e enfatiza fortemente os aspectos de melhoria contínua [FIT 99].

Na área de engenharia de software, há um interesse em compreender e desenvolver ambientes de engenharia de software centrados em processos (PSEE), que tornem possível especificar explicitamente o processo de desenvolvimento de software. O suporte a cooperação é crucial para esses ambientes, pois o desenvolvimento de software é um esforço de equipes.

Um ambiente de desenvolvimento de software oferece vantagens como:

- Melhoria da comunicação entre as pessoas da equipe de desenvolvimento;
- Representação do processo de desenvolvimento;
- Acompanhamento das atividades realizadas;
- Coleta de informações para avaliação do processo de software;
- Integração de documentos.

Um PSEE está centralizado na descrição explícita de um processo de software, chamada de modelo de processo. A execução de um modelo de processo dentro de um PSEE provê suporte as pessoas envolvidas nas atividades de desenvolvimento de software (agentes do processo), por automatizar algumas etapas do processo.

Atualmente, já existe um certo número de modelos e ambientes experimentais que suportam a abordagem orientada a processos. Dentre esses projetos estão os projetos ARCADIA [UCI 95], SPADE [BAN 96], E3 [FIN 94], e EPOS [FIN 94, NGU 97].

Mas, a utilização desses ambientes em projetos reais de desenvolvimento não tem sido expressiva. Uma justificativa para este fato é que alguns desses ambientes se restringem a apoiar processos baseados em uma metodologia, e em alguns casos, com ferramentas de desenvolvimento específicas [ARA 99], conferindo-lhes baixa flexibilidade para adaptação do processo a mudanças. Esses ambientes apresentam linguagem de modelagem de processos um pouco complexas, dificultando a definição dos processos de software [OCA 98].

Um sistema de gerência de *workflow* (SGW) provê automação de um processo de negócio pelo gerenciamento da seqüência de atividades e a invocação apropriada de recursos de tecnologia de informação e/ou humano, associados com os vários passos da atividade [WFM 99].

Nas últimas duas décadas, sistemas de gerenciamento de *workflow* têm demonstrado serem úteis na automação de processos de negócios. O processo de software pode ser visto

como um processo de negócio, composto de vários sub-processos, com diferentes granularidades de detalhes [OCA 98].

Em sistemas de *workflow*, o elemento básico é o processo, sua composição e as pessoas que o executam [ARA 99]. Sistemas de *workflow* são maleáveis, possibilitam a evolução dos processos de desenvolvimento, permitem a integração de ferramentas de diferentes fabricantes e muitos desses sistemas apresentam interface gráfica para a definição dos processos, facilitando a sua utilização.

Existem vários sistemas de gerência de *workflow* disponíveis no mercado hoje, focando em diferentes funcionalidades e integração de dados, tais como: Oracle Workflow [ORA 2001], Genexus [ART 2000], Ultimus [ULT 98, ULT 98a], Lotus Notes [AMA 97] e Exchange 2000 Server [MAR 2000, RIZ 2000]. Esses sistemas de *workflow* serão descritos brevemente no capítulo 7.

A utilização de sistemas de *workflow* comerciais, como o Exchange 2000 Server, pode ser uma boa alternativa já que a construção de ferramentas com características como: gerenciamento de processo, integração com ferramentas, atribuição de atividades e monitoramento de projetos; pode ser complexa e cara.

Utilizando *workflow* comerciais, é possível modelar o processo utilizando as facilidades providas pelo WMS, sem se preocupar como estas estão implementadas, concentrando-se somente no processo a ser definido.

O Exchange 2000 Server permite a construção de aplicações que modelam e automatizam processos de negócio [MAR 2000].

O Exchange 2000 Server foi o servidor de colaboração escolhido para este experimento porque algumas organizações já o utilizam como servidor de e-mail, e poderiam passar a utilizá-lo também como servidor de *workflow,* não necessitando adquirir um novo produto.

Exchange pode ser integrado com as ferramentas do Microsoft Office, utiliza a tecnologia *Web Storage System* para gerenciar dados. O *Web Storage System* possibilita acessar dados por meio de uma URL, apresenta várias tecnologias para acesso a dados (ADO, CDO e XML), descreve um modelo de eventos e possibilita o uso de lógica de *workflow* para criar aplicações de colaboração.

2.1 Trabalhos Relacionados

Em [RAT 99a], é descrita uma avaliação do Processo Unificado Rational com relação à norma ISO/IEC15504 - Parte 5. A Parte 5 detalha os processos obrigatórios e define as práticas base, produtos de trabalho, e práticas de gerenciamento associadas, as quais são usadas para determinar a capacidade para cada processo avaliado.

RUP foi também avaliado com relação ao suporte fornecido à implantação da norma ISO 9000-3. Nesta avaliação são descritas as conformidades do Processo Unificado Rational 5.0, com a Norma e as áreas nas quais o processo não atende aos requisitos de qualidade especificados pela ISO 9000-3 [ALC 99].

Considerando que várias empresas estão adotando o Processo Unificado Rational, e muitas delas como a Uniway [UNI 2000] e o SERPRO, objetivam alcançar níveis mais

altos de CMM, este trabalho propõe identificar os elementos descritos no Processo Unificado que suportam as práticas-chave descritas pelo CMM e onde o RUP apresenta lacunas com relação ao modelo CMM níveis 2 e 3.

Após a avaliação, serão elaboradas propostas visando estender o Processo Unificado para atender todas as práticas-chave descritas no CMM para os níveis 2 e 3. Os trabalhos referenciados acima não apresentam essa preocupação.

A utilização de sistemas de *workflow* para apoiar processos de software tem sido alvo de estudo de alguns grupos [OCA 98, CHA 97, CHA 97a]. Alguns desses trabalhos estão descritos abaixo.

Chan e Leung [CHA 97, CHA 97a] propõem o uso de sistemas de *workflow* para representar processos de desenvolvimento de software. Segundo esses autores, por ser de uso mais geral e ser mais flexível, o paradigma de *workflow* suporta características ausentes em outras abordagens. Chan e Leung identificam alguns requisitos para o paradigma de *workflow* que são aplicáveis em muitas outras aplicações, e propõem um modelo de *workflow* que pode encontrar esses requisitos.

Veraart e Wright em [VER 97], descrevem o Ambiente de Gerenciamento de Tarefas do Processo de Software (Software Process Task Management Environment – SPTME), usando princípios de PSEEs e de sistemas de *workflow*.

Gimenes et al. [GIM 2000] têm proposto um *framework* de agenda de tarefas para gerenciadores de processo. O *framework* proposto pode ser utilizado em PSEEs, em sistemas gerenciadores de *workflow* e em domínios correlatos.

Araujo e Borges [ARA 99] discutem a aplicabilidade de sistemas de *workflow* no suporte a processos de software, comparando as funcionalidades desses sistemas com as possibilidades de suporte a esse tipo de processo.

Maurer et. al. [MAU 99] apresentam o ambiente MILOS, formado por uma arquitetura baseada em componentes que suporta processos de desenvolvimento de software na Internet. Este ambiente pode ser conectado com o MS-Project para suporte ao planejamento, e usa uma ferramenta de métricas para medir diferentes dados durante a execução do processo.

Barnes e Gray em [BAR 2000] propõem o tratamento de gerenciamento de processos de software como uma especialização de um tipo mais geral de gerenciamento de processos, conhecido como *workflow*. O protótipo da ferramenta inicial usa um sistema de gerenciamento de base de dados (SGBD) relacional como um repositório CASE. Os modelos de processo são implementados como códigos SQL (procedimentos) armazenados no SGBD.

Callahan e Ramakrishnam desenvolveram um ambiente chamado WISE (Web Integrated Software Environment) [CAL 96]. Este ambiente permite o gerenciamento e a medição de projetos de desenvolvimento de software baseado na análise dinâmica das alterações das atividades no *workflow*.

Nos trabalhos citados acima não é comum o uso de sistemas de *workflow* comerciais para a implementação dos ambientes propostos. Visto a complexidade, o custo e o tempo necessário para se desenvolver sistemas de *workflow*, este trabalho propõe a utilização de

um sistema de *workflow* comercial para a elaboração de um protótipo de ambiente de apoio ao processo.

3 Processo Unificado Rational

O Processo Unificado Rational (RUP) é um processo de negócio genérico para engenharia de software orientado a objetos. Ele provê um enfoque disciplinado para distribuição de tarefas e responsabilidades dentro de uma organização de desenvolvimento. Seu objetivo é assegurar a produção de produtos de alta qualidade que satisfaçam as necessidades de seus usuários finais, dentro de cronogramas e orçamentos previsíveis. O Processo Unificado Rational captura as melhores práticas em desenvolvimento de software moderno, num formato que pode ser adaptado para uma grande variedade de projetos e organizações [KRU 2000]. As melhores práticas do RUP serão descritas na seção 3.1.

O Processo Unificado descreve um conjunto de atividades necessárias para transformar os requisitos do usuário em um sistema de software. O RUP utiliza a Linguagem de Modelagem Unificada (UML) para descrever os modelos e diagramas do sistema.

O Processo Unificado pode ser caracterizado através de três palavras-chaves: dirigido a casos de uso, centrado na arquitetura, e iterativo e incremental [JAC 98].

Dirigido a Casos de Uso

O Processo Unificado utiliza casos de usos para descrever os requisitos do usuário. Todos os casos de uso formam o modelo de casos de uso, o qual descreve todas as funcionalidades do sistema, em síntese, descreve o que o sistema faz. Casos de uso vão além da especificação dos requisitos de um sistema, dirigindo seu projeto, implementação, e teste [JAC 98]. Baseado no modelo de casos de uso, os desenvolvedores criam uma série de modelos de projeto e implementação. Os testadores usam os casos de uso para verificar se a implementação está conforme o modelo de casos de uso.

Dirigido a casos de uso, significa que o processo de desenvolvimento segue um fluxo, especificado por uma série de macro-atividades que derivam dos casos de uso. Casos de uso são especificados, casos de uso são projetados, e no final, casos de uso são usados pelos testadores para a construção dos casos de teste [JAC 98].

Casos de uso são desenvolvidos junto com a arquitetura do sistema, e um influencia o outro, e ambos amadurecem enquanto o ciclo de vida continua [JAC 98].

Centrado na Arquitetura

A arquitetura em um sistema de software é descrita como várias visões do sistema que está sendo construído. O conceito de arquitetura de software incorpora os aspectos estáticos e dinâmicos mais importantes na construção do sistema. A arquitetura origina-se das necessidades da organização, como percebido pelo usuário, e dos casos de uso. Porém, ela é também influenciada por muitos outros fatores, como a plataforma na qual o software está executando (arquitetura do computador, sistema operacional, sistema gerenciador de banco de dados, protocolos para comunicação de redes), os componentes reusáveis disponíveis (*frameworks* para interfaces gráficas de usuários), considerações de desenvolvimento, sistemas legados, e requisitos não-funcionais (desempenho, segurança) [JAC 98].

As funções do sistema correspondem aos casos de uso e a forma é expressa pela arquitetura. É necessário haver influência mútua entre casos de uso e arquitetura, e estes devem evoluir em paralelo [KRU 2000, JAC 98].

A arquitetura deve ser projetada de forma a permitir que o sistema evolua, não somente pensando em seu desenvolvimento inicial, mas pensando em gerações futuras. Para encontrar uma forma, o arquiteto deve trabalhar em cima de um entendimento geral das funções chaves, isto é, os casos de uso principais do sistema. Estes casos de uso principais, talvez uma quantia de 5% a 10% de todos os casos de uso do sistema, constituem o núcleo das funções do sistema [JAC 98].

Iterativo e Incremental

Desenvolver um produto de software comercial é uma atividade que pode durar vários meses, talvez um ano ou mais. É prático dividir o trabalho em partes ou mini-projetos. Cada mini-projeto é uma iteração que resulta em um incremento. Iteração se refere aos passos em uma macro-atividade, e incrementos ao crescimento do produto [JAC 98].

Em cada iteração, os desenvolvedores identificam e especificam os casos de uso relevantes, criam um projeto usando a arquitetura escolhida como um guia, implementam o projeto em componentes, e testam este mini-projeto.

Há muitos benefícios para um processo iterativo e controlado [JAC 98]:

- Iterações controladas reduzem o custo de riscos para as despesas de um único incremento. Se os desenvolvedores precisam repetir a iteração, a organização perde somente os esforços mal-direcionados de uma iteração, não do produto inteiro.
- Iterações controladas reduzem o risco de não se desenvolver o produto no cronograma planejado. No desenvolvimento tradicional, onde problemas complexos são somente desvendados pelo teste de sistema, o tempo requerido para resolvê-los, geralmente, excede o tempo restante no cronograma e quase sempre força um atraso na entrega.
- Iterações controladas aceleram o ritmo dos esforços de desenvolvimento, porque os desenvolvedores trabalham voltados a resultados claros, com foco limitado e não com um cronograma longo em constante alteração.
- Iterações controladas confirmam uma realidade freqüentemente ignorada - que as necessidades dos usuários e os requisitos correspondentes não podem ser completamente definidos no início. Eles são tipicamente refinados em iterações sucessivas. Este modo de operação torna fácil adaptar os requisitos as mudanças.

Estes conceitos - dirigido a casos de uso, centrado na arquitetura, e desenvolvimento iterativo e incremental - são igualmente importantes . Arquitetura provê a estrutura para guiar o trabalho em iterações, enquanto que casos de uso definem os objetivos e dirigem o trabalho de cada iteração. Remover uma das três idéias chaves, poderia severamente reduzir o valor do Processo Unificado [JAC 98, KRU 2000].

O Processo Unificado Rational captura as melhores práticas de desenvolvimento de software moderno de uma forma que é adequada para uma grande variedade de projetos e

organizações [RAT 99, KRU2000]. As seis melhores práticas de desenvolvimento de software serão descritas brevemente.

3.1 Melhores Práticas

Processo Unificado Rational descreve como efetivamente implantar comercialmente abordagens comprovadas para desenvolvimento de software para equipes de desenvolvimento de software [RAT 98, RAT 2001]. Estas são chamadas "melhores práticas", porque elas são comumente usadas por organizações bem-sucedidas. Estas melhores práticas são as seguintes [RAT 98, RAT 2001, KRU 2000]:

1. Desenvolvimento iterativo
2. Gerência de requisitos
3. Uso de arquitetura baseada em componentes
4. Modelagem visual
5. Verificação contínua da qualidade de software
6. Gerência de alteração

Desenvolvimento Iterativo

O Processo Unificado Rational suporta uma abordagem iterativa para desenvolvimento, que focaliza nos itens de altos riscos em cada estágio do ciclo de vida. Da perspectiva de desenvolvimento, o ciclo de vida de software é uma sucessão de iterações, através das quais o software é desenvolvido incrementalmente. Cada iteração é concluída com o *release* de um produto executável [RAT 98, RAT 99, RAT 2001].

Gerência de Requisitos

O Processo Unificado Rational descreve uma abordagem sistemática para elicitar, organizar, comunicar e gerenciar as alterações nos requisitos de um software ou aplicação [RAT 99].

Uso de arquitetura baseada em componentes

O foco principal das primeiras iterações do processo é produzir e validar uma arquitetura de software que, no ciclo de desenvolvimento inicial, tem a forma de um protótipo arquitetural executável que gradualmente evolui para tornar-se o sistema final nas últimas iterações. O Processo Unificado Rational suporta desenvolvimento de software baseado em componentes [RAT 98, RAT 99, RAT 2001].

Modelagem Visual

O processo mostra como modelar o software visualmente para capturar a estrutura e o comportamento da arquitetura e de componentes. A Linguagem de Modelagem Unificada (UML), é a base para a modelagem visual [RAT 98].

Verificação contínua da qualidade

Qualidade deve ser revisada com respeito a seus requisitos com base na segurança, funcionalidade, e desempenho. O Processo Unificado Rational assiste no planejamento,

projeto, implementação, execução e avaliação destes tipos de testes. Qualidade é a responsabilidade de todos os membros da organização de desenvolvimento [RAT 98].

Gerência de Alteração

O Processo Unificado descreve uma abordagem sistemática para gerenciar alterações nos requisitos, projeto e implementação. O RUP descreve como automatizar integração e gerenciamento de versões [RAT 98, RAT 2001].

3.2 Ciclo de Vida do Processo Unificado

O Processo Unificado pode ser visto de duas perspectivas diferentes, porém integradas, a perspectiva gerencial aborda os aspectos financeiros, estratégicos, comerciais e humanos e a perspectiva técnica aborda aspectos de qualidade, engenharia e projeto [KRU 2000].

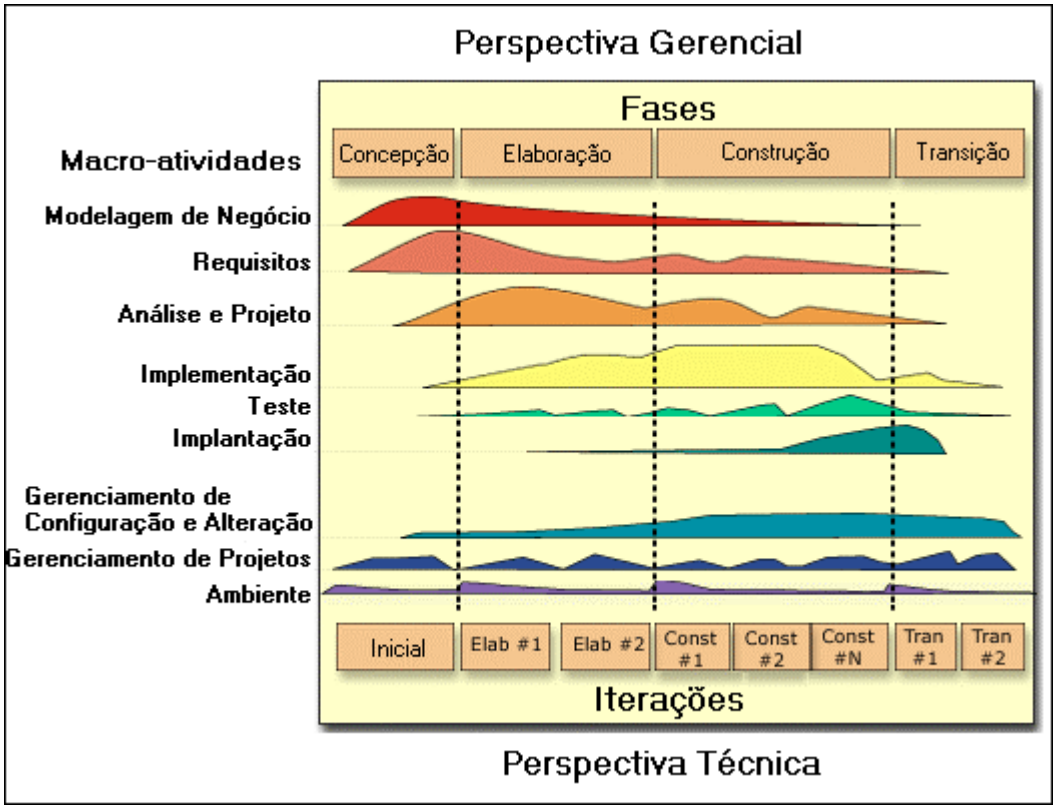


FIGURA 3.1 - Estrutura do Processo – Duas Dimensões [Extraído de RAT 2001]

Da perspectiva gerencial, o ciclo de vida do software é decomposto em quatro fases (concepção, elaboração, construção e transição), concluídas por um marco [RAT 98].

Da perspectiva técnica, o ciclo de vida do software consiste de várias macro-atividades de desenvolvimento, e cada passo, através dessa seqüência de macro-atividades, é chamado de iteração. Então, o ciclo de vida do software é uma sucessão de iterações, por meio das quais o software é desenvolvido incrementalmente [RAT 99, RAT 2001].

Cada iteração finaliza com uma versão (*release*) de um produto executável. Cada versão é acompanhada por artefatos de suporte: descrição da versão, documentação do usuário, planos, e modelos atualizados do sistema. Uma versão pode ser interna ou externa. Uma versão interna é usada somente pela organização de desenvolvimento, como parte de um marco (*milestone*). Uma versão externa é entregue aos usuários finais [RAT 99].

O produto final precisa incluir vários artefatos de projeto, não apenas os componentes executáveis. Para realizar o novo ciclo eficientemente, os desenvolvedores precisam de toda a representação do produto de software, que inclui [JAC 98]:

- Um modelo de casos de uso com todos os casos de uso e suas relações com os usuários;
- Um modelo de análise, o qual tem duas finalidades: refinar os casos de uso em mais detalhes e fazer uma alocação inicial de comportamento do sistema para um conjunto de objetos que provêem seu comportamento.
- Um modelo de projeto que define a estrutura estática do sistema como subsistemas, classes, e interfaces e os casos de uso realizados como colaboração entre os subsistemas, classes e interfaces.
- Um modelo de implementação, que inclui componentes (representando código fonte) e o mapeamento das classes para componentes.
- Um modelo de implantação, o qual define os nodos físicos de computadores e o mapeamento dos componentes para aqueles nodos.
- Um modelo de teste, o qual descreve os casos de teste que verificam os casos de uso.
- E, uma representação da arquitetura.

Todos estes modelos são relacionados, e juntos representam o sistema como um todo.

3.2.1 Iterações e Fases em um Ciclo de Vida

Cada fase no Processo Unificado Rational pode ser dividida em iterações. Uma iteração é um ciclo de desenvolvimento completo, resultando em uma versão, um subconjunto do produto final, que cresce incrementalmente de iteração a iteração para se transformar no produto final [JAC 98, RAT 98].

Através de uma seqüência de modelos, é possível visualizar o que cada uma das fases faz. Dentro de cada fase, gerentes ou desenvolvedores podem dividir seu trabalho ainda mais - em iterações e incrementos subseqüentes. Cada fase termina com um marco, que pressupõe a disponibilidade de um conjunto de artefatos; isto é, certos modelos ou documentos devem ter sido desenvolvidos na fase [JAC 98, RAT 99].

O marco serve para vários propósitos. O mais crítico é que gerente tem que tomar certas decisões cruciais antes do trabalho poder prosseguir para a próxima fase. Marcos permitem o gerenciamento e o monitoramento do progresso do trabalho. Finalmente, por guardar o tempo e esforços gastos em cada fase, pode-se usar estes dados em estimativas de tempo para outros projetos. Marcos são pontos de controle [JAC 98].

Ao final de cada fase é necessária uma avaliação quanto as seguintes questões [RAT 99, RAT 2001]:

- Os objetivos definidos foram satisfeitos;
- Os artefatos definidos estão completos;
- Os vários modelos estão atualizados.

Um passo através das quatro fases é um ciclo de desenvolvimento. Cada passo, através das quatro fases, produz a geração de um produto, que evolui para a próxima geração por repetir a mesma seqüência, mas com objetivos diferente. Estes ciclos subseqüentes são chamados ciclos evolutivos [RAT 99, RAT 2001].

Fase de Concepção (*Inception Phase*)

Esta fase identifica uma visão inicial do produto final, transformando-o em um projeto. Seu propósito é definir uma visão do produto e os casos de negócio associados, definindo o escopo do projeto [RAT 98].

Os objetivos primários dessa fase incluem [RAT 99, RAT 2001]:

- Estabelecer o escopo do projeto e as condições limites, incluindo conceitos operacionais, critérios de aceitação, etc;
- Discriminar os casos de uso críticos do sistema, os cenários primários de operação que dirigirão os maiores projetos;
- Exibir, e talvez demonstrar, ao menos uma arquitetura candidata em frente a alguns dos cenários primários;
- Estimar o custo total e o cronograma para o projeto todo;
- Estimar os riscos potenciais;
- Preparar o ambiente de suporte para o projeto.

Marco

O final da fase de concepção é o primeiro dos marcos maiores do projeto, e é chamado de Marco dos Objetivos do Ciclo de vida. Neste ponto é necessário avaliar os seguintes critérios [RAT 99, RAT 2001]:

- Participação dos interessados (*stakeholders*) na definição do escopo e nas estimativas de custo/cronograma;
- Acordo de que os requisitos foram capturados estão corretos e que há um entendimento comum destes requisitos;
- Acordo entre as partes que as estimativas de custo/cronograma, prioridades, riscos, e processo de desenvolvimento são apropriadas;
- Identificação dos riscos do projeto, associados às estratégias de suavização.

O projeto pode ser encerrado ou revisto se não alcançou algum desses critérios.

Fase de Elaboração (*Elaboration Phase*)

A especificação do produto é refinada, definindo uma arquitetura padrão, e desenvolvendo um plano mais preciso para seu desenvolvimento e implantação [RAT 98].

Durante a fase de elaboração, muitos dos casos de uso do produto são especificados em detalhes e a arquitetura do sistema é projetada. A arquitetura é expressa como visões de todos os modelos do sistema, os quais juntos representam o sistema completo. Durante essa fase de desenvolvimento, os casos de uso mais críticos identificados durante a fase de elaboração são realizados. O resultado desta fase é uma arquitetura básica e uma avaliação de riscos [RAT 98, RAT 2001]. A estabilidade da arquitetura é avaliada por meio de um ou mais protótipos da arquitetura [RAT 2001].

No final da fase de elaboração, o gerente de projeto pode planejar as atividades e estimar os recursos requeridos para completar o projeto.

Os objetivos primários dessa fase são [RAT 2001]:

- Assegurar que a arquitetura, requisitos e planos estão suficientemente estáveis, e os riscos mitigados para determinar cursos e planejamento do desenvolvimento completo;
- Enfocar todos os riscos significativos associados à arquitetura para o projeto;
- Estabelecer uma arquitetura básica, derivada dos cenários (descrição de instâncias de casos de uso) significantes arquiteturalmente, os quais expõem os riscos técnicos principais do projeto;
- Produzir um protótipo evolutivo de componentes de qualidade, e se possível um ou mais protótipos exploratórios para mitigar riscos específicos;
- Estabelecer um ambiente de suporte;
- Demonstração de que a arquitetura pode suportar os requisitos do sistema em custos e tempos razoáveis.

Marco

Este é o marco da Arquitetura do Ciclo de Vida. Os critérios principais de avaliação para a fase de elaboração envolvem respostas para as seguintes questões [RAT 99, RAT 2001]:

- A visão do produto, os requisitos e a arquitetura estão estáveis;
- Os protótipos executáveis podem demonstrar que os maiores elementos de risco têm sido enfocados e podem ser resolvidos;
- O plano de iteração para a fase de construção está suficientemente detalhado e fiel para permitir o prosseguimento do trabalho;
- O plano de iteração para a construção está suportado por estimativas com credibilidade;

- Todos os interessados acreditam que a visão do sistema pode ser encontrada se os planos forem executados no contexto da arquitetura atual;
- Os recursos atuais gastos são aceitáveis *versus* os recursos planejados;

O projeto pode ser encerrado ou revisto se não alcançou algum desses critérios.

Fase de Construção (*Construction Phase*)

O objetivo da fase de construção é clarificar os requisitos restantes e completar o desenvolvimento do sistema baseado na arquitetura definida [RAT 2001].

A fase de construção é semelhante a um processo de manufatura, onde a ênfase é gerenciar recursos, controlar operações para otimizar custos, cronogramas e qualidade [RAT 2001].

Os objetivos primários da fase de construção são [RAT 2001]:

- Minimizar custos de desenvolvimento por otimizar recursos e evitar fragmentos e re-trabalho desnecessários;
- Alcançar qualidade adequada rapidamente;
- Alcançar versões utilizáveis (*alfa*, *beta* e outros testes *releases*) rapidamente;
- Complementar a análise, projeto, implementação e teste para as funcionalidades requeridas;
- Desenvolver iterativamente e incrementalmente um produto completo que está pronto para implantar na comunidade de usuários;
- Decidir se o software, os *sites* e os usuários estão todos prontos para a aplicação que será implantada;
- Possibilitar algum nível de paralelismo no trabalho das equipes de desenvolvimento. Este paralelismo pode acelerar as atividades de desenvolvimento significativamente; mas também aumenta a complexidade de gerenciamento de recursos e sincronização das macro-atividades. Uma arquitetura robusta é essencial para que qualquer paralelismo significante seja alcançado.

Marco

Este é o marco da Capacidade Operacional Inicial. Os critérios principais de avaliação para a fase de construção envolvem respostas para as seguintes questões [RAT 98]:

- A versão do produto está estável e madura o suficiente para ser implantada na comunidade de usuários;
- Todos os interessados estão prontos para a transição;
- Os gastos atuais *versus* os gastos planejados são ainda aceitáveis.

Fase de Transição (*Transition Phase*)

O foco da fase de transição é assegurar que o software está disponível para seus usuários finais. A fase de transição pode passar por várias iterações, cada iteração inclui testar o produto na preparação de um release, e fazer ajustes menores baseados no feedback do usuário [RAT 2001].

Os objetivos primários da fase de transição incluem [RAT 99, RAT 2001]:

- Teste beta para validar o novo sistema frente às expectativas dos usuários;
- Conversão de bases de dados operacionais;
- Treinamento de usuários e mantenedores;
- Atividades relacionadas a marketing, distribuição e vendas;
- Direcionar para as atividades que objetivam determinar problemas, aumentar a performance e usabilidade;
- Avaliação das configurações-base de desenvolvimento frente a visão completa e os critérios de aceitação para o produto;
- Alcançar portabilidade para os usuários;
- Obter um acordo com os interessados que as configurações-base estão completas e consistentes com os critérios de avaliação da visão.

Ao final da fase de transição tem-se como saída o produto que será entregue ao usuário, os documentos atualizados, e um relatório com as vantagens geradas como resultado neste ciclo.

Marco

Este é o marco de Versões do Produto. Os critérios principais de avaliação para a fase de transição envolvem respostas para as seguintes questões [RAT 99, RAT 2001]:

- O usuário está satisfeito ?
- Os gastos atuais *versus* os gastos planejados são ainda aceitáveis ?

3.3 Um Processo Integrado

O Processo Unificado é baseado em componentes, e utiliza a Linguagem de Modelagem Unificada (UML). O Processo Unificado integra ciclos, fases, macro-atividade, suavização de riscos, controle de qualidade, gerenciamento de projetos e controle de configuração. Este *framework* também trabalha como um guarda-chuva sob o qual vendedores de ferramentas e desenvolvedores podem construir ferramentas para suportar a automação de processos, suportar macro-atividades individuais, construir modelos diferentes e integrar o trabalho através do ciclo de vida e dos demais modelos. A figura 3.2 mostra a relação entre os elementos do RUP [JAC 98, RAT 2001].

Trabalhador (*Worker)*

O trabalhador define o comportamento e as responsabilidades de um indivíduo, ou um conjunto de indivíduos trabalhando em uma equipe, dentro do contexto de uma organização de software. O trabalhador representa um cargo executado por indivíduos em um projeto e define como eles devem realizar o trabalho [RAT 2001, RAT 99].

Atividade (*Activity*)

Uma atividade é uma unidade de trabalho que um indivíduo que representa um trabalhador é encarregado de executá-la. Uma atividade tem uma finalidade clara, usualmente expressa em termos de criação ou atualização de algum artefato, como um modelo, uma classe, um plano [RAT 2001, RAT 99].

As atividades são associadas a [RAT 2001, RAT 99]:

Guias de trabalho (*work guidelines)* - representam técnicas e conselhos práticos que são úteis para o trabalhador executar a atividade.

Conceitos (*Concepts*) - Alguns dos conceitos-chave de processo, tais como: interação, fase, risco, performance de teste, são introduzidos em seções separadas do processo, e são ligadas às macro-atividades adequadas.

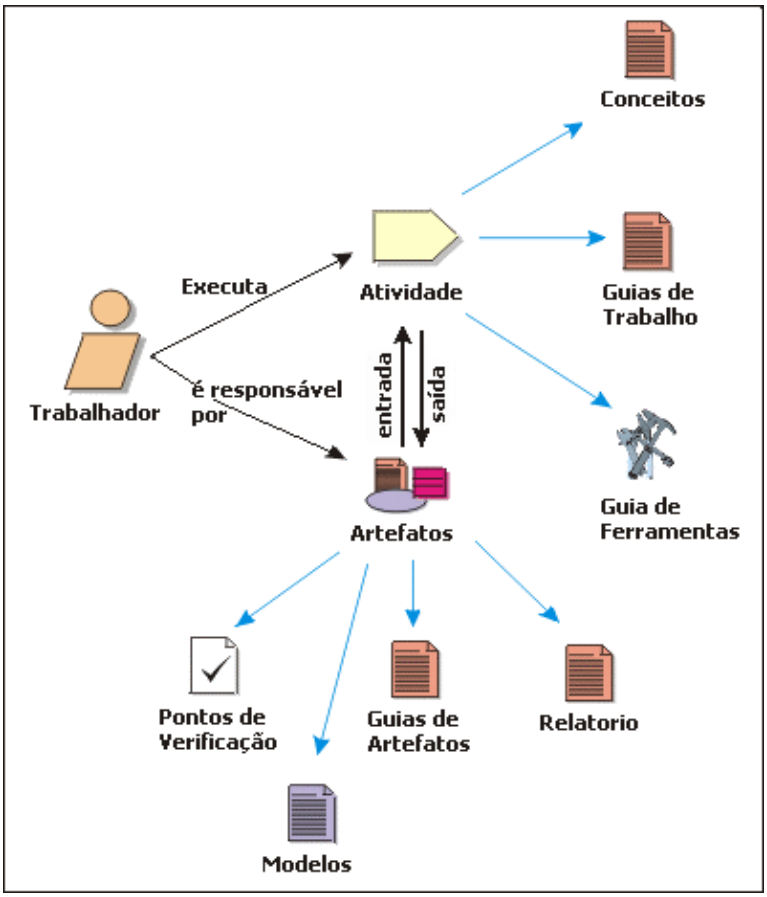


FIGURA 3.2 – Conceitos Básicos do Processo Unificado [Extraído de RAT 2001]

Guia de Ferramentas (*Tool Mentor)* - são meios adicionais de prover guias mostrando como executar atividades, usando ferramentas específicas.

Artefatos (*Artifact*)

Atividades têm artefatos como entrada e saída. Um artefato é um produto de trabalho do processo, trabalhadores usam artefatos para executar suas atividades, e produzem artefatos durante a execução de suas atividades. Artefatos são de responsabilidade de um único trabalhador, em um processo, cada porção de informação é de responsabilidade de uma pessoa específica [RAT 98, RAT 99].

Artefatos podem ser: modelos, como modelo de casos de uso, modelo de projeto, modelo de elementos e diagrama de classes; ou documentos, como casos de negócio e documento da arquitetura do software; ou códigos-fonte ou executável.

Os artefatos estão associados a [RAT 98, RAT 99]:

Modelos (*Templates*) - são modelos ou protótipos de artefatos. Associados aos artefatos estão um ou mais modelos que podem ser usados para criar os artefatos correspondentes. Modelos estão relacionados às ferramentas que os utilizam. Exemplos de modelos: modelo do Microsoft Project é usado para elaborar o plano do projeto, e modelos do Word são usados para elaborar documentos, alguns relatórios, etc [RAT 98, RAT 99].

Relatórios (*Reports*) - modelos ou elementos do modelo podem ter relatórios associados a eles. Um relatório extrai informações sobre o modelo ou elementos do modelo de uma ferramenta. Por exemplo, um relatório apresenta elementos para uma revisão [RAT 98, RAT 99].

Guia de Artefatos e Pontos de Verificação (*Artifact Guidelines* e *Checkpoints*) - estes elementos apresentam informação de como desenvolver, avaliar e usar os artefatos. Os guias de artefatos descrevem como fazer os artefatos, já os pontos de verificação provêem um guia de referência rápida para ajudar a avaliar a qualidade do artefato desenvolvido [RAT 98, RAT 99].

3.4 Macro-atividades

Uma mera enumeração dos trabalhadores, atividades e artefatos não constituem um processo. É necessário descrever a seqüência das atividades que produz algum resultado e mostrar a interação entre os trabalhadores. Uma macro-atividade é uma seqüência de atividades que produz um resultado de valor considerável [RAT 98, RAT 99].

No Processo Unificado Rational existem nove núcleos de macro-atividades (*core workflows*), que são divididos em seis macro-atividades de processo e três de suporte [KRU 2000].

As macro-atividades de processo são: Modelagem de Negócio, Requisitos, Análise e Projeto (desenho), Implementação, Teste e Implantação.

As macro-atividades de suporte são: Gerenciamento de Configuração e Alteração, Gerenciamento de Projetos e Ambiente.

As macro-atividades são expressas em termos de diagramas de atividades.

Cada passo pela macro-atividade é descrito em com mais detalhes, nas unidades da macro-atividade (*workflow detail*). Estas unidades são grupos de atividades que são executadas juntas, e estão relacionadas a artefatos de entrada e de saída [RAT 2001].

Em UML, um *workflow* pode ser expresso por meio de um diagrama de seqüência, um diagrama de colaboração ou um diagrama de atividades [RAT 2001]. No Processo Unificado Rational optou-se por descrever as macro-atividades por meio de diagramas de atividades [RAT 2001].

Diagramas de atividades são usados para modelar os aspectos dinâmicos de sistemas, isto envolve, na maioria das vezes, a modelagem de etapas seqüenciais (e possivelmente concorrente) em um processo computacional [BOO 98]. Um diagrama de atividades é essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra [BOO 98].

A seguir, serão descritas brevemente, as macro-atividades do Processo Unificado. Para cada macro-atividade são descritos a finalidade, os trabalhadores e artefatos e as atividades.

3.4.1 Macro-atividade de Modelagem de Negócio

As finalidades da modelagem de negócio são [KRU 2000, RAT 2001]:

- Entender a estrutura e a dinâmica da organização na qual o sistema será implantado (organização alvo);
- Entender os problemas atuais da organização alvo e identificar potenciais de melhoria;
- Assegurar que clientes, usuários finais e desenvolvedores têm um entendimento comum da organização alvo;
- Derivar os requisitos do sistema para apoiar a organização.

Para alcançar esses objetivos, a macro-atividade de modelagem de negócio descreve como desenvolver uma visão da organização, e baseado nesta visão, definir o processo, cargos e responsabilidades desta organização em um modelo de negócio. Este modelo compreende um modelo de casos de uso de negócio e um modelo de objetos de negócio [KRU 2000].

Em adição a esses modelos são desenvolvidos os artefatos: especificação de negócio complementar e o glossário [RAT 2001].

Trabalhadores e Artefatos

Os principais trabalhadores envolvidos na modelagem de negócio são os seguintes [RAT 2001, KRU 2000]:

- **Analista de processo de negócio:** o analista de processo de negócio conduz e coordena a modelagem de casos de uso, por esboçar e delimitar a organização

sendo modelada. Por exemplo, estabelece quais atores e casos de uso de negócio existem e como eles interagem.

- **Projetista de negócio:** o projetista de negócio detalha a especificação de uma parte da organização por descrever um ou mais casos de uso de negócio. Ele determina as entidades e os trabalhadores de negócio necessários para realizar um caso de uso de negócio, e o relacionamento destes para a realização do caso de uso. O projetista de negócio define as responsabilidades, operações, atributos e a relação entre um ou mais trabalhadores de negócio e entidades de negócio.

São também envolvidos nesta macro-atividade os seguintes trabalhadores [RAT 2001]:

- **Interessados (*stakeholders*):** representam várias partes da organização. Provêem dados de entrada para o sistema e participam de revisões.
- **Revisor de negócio:** participa de revisões formais dos artefatos desenvolvidos (modelo de casos de uso de negócio e modelo de objetos de negócio).

Os artefatos principais da macro-atividade de modelagem de negócio são os seguintes [RAT 2001]:

- **Visão do negócio:** define os objetivos e metas dos esforços de modelagem de negócio.
- **Modelo de casos de uso de negócio:** modelo das funções pretendidas de negócio.
- **Modelo de objetos de negócio:** modelo de objetos que descreve a realização de casos de uso de negócio.

Outros artefatos incluídos são os seguintes [RAT 2001]:

- **Avaliação da organização:** descreve o status atual da organização, na qual o sistema será implantado.
- **Regras de negócio:** declarações de políticas ou condições que devem ser satisfeitas.
- **Especificação de negócio complementar:** um documento que apresenta definições de negócio não incluídas no modelo de casos de uso de negócio ou no modelo de objetos de negócio.
- **Glossário de negócio:** define termos importantes usados na modelagem de negócio.
- **Documento de arquitetura de negócio:** provê uma visão compreensiva de negócio, usando diferentes visões arquiteturais para descrever diferentes aspectos de negócio.
- **Caso de uso de negócio:** define um conjunto de instâncias de casos de uso, onde cada instância é uma seqüência de ações executadas por um ator de negócio para obter um resultado.

- **Realização de caso de uso de negócio:** descreve como um caso de uso de negócio particular é realizado dentro de um modelo de objetos de negócio, em termos de colaboração entre os objetos (instâncias de trabalhadores de negócio e objetos de negócio).

Atividades da Macro-atividade de Modelagem de Negócio

1. Avaliar status de negócio

A finalidade desta unidade é [RAT 2001]:

- Avaliar o status da organização na qual o sistema será implantado (organização alvo);
- Entender como categorizar o projeto e quais os cenários de modelagem de negócio serão melhor combinados;
- Decidir como continuar o trabalho na iteração corrente e esboçar como trabalhar nas iterações subseqüentes com os artefatos de modelagem de negócio;
- Desenvolver um primeiro entendimento das metas e objetivos, uma visão de negócio, da organização alvo que pode ser acordada entre os interessados e a equipe de desenvolvimento de negócio.

É composta das seguintes atividades: capturar um vocabulário de negócio comum, manter regras de negócio, avaliar a organização alvo e fixar e ajustar metas, executadas pelo analista de processo de negócio [RAT 2001].

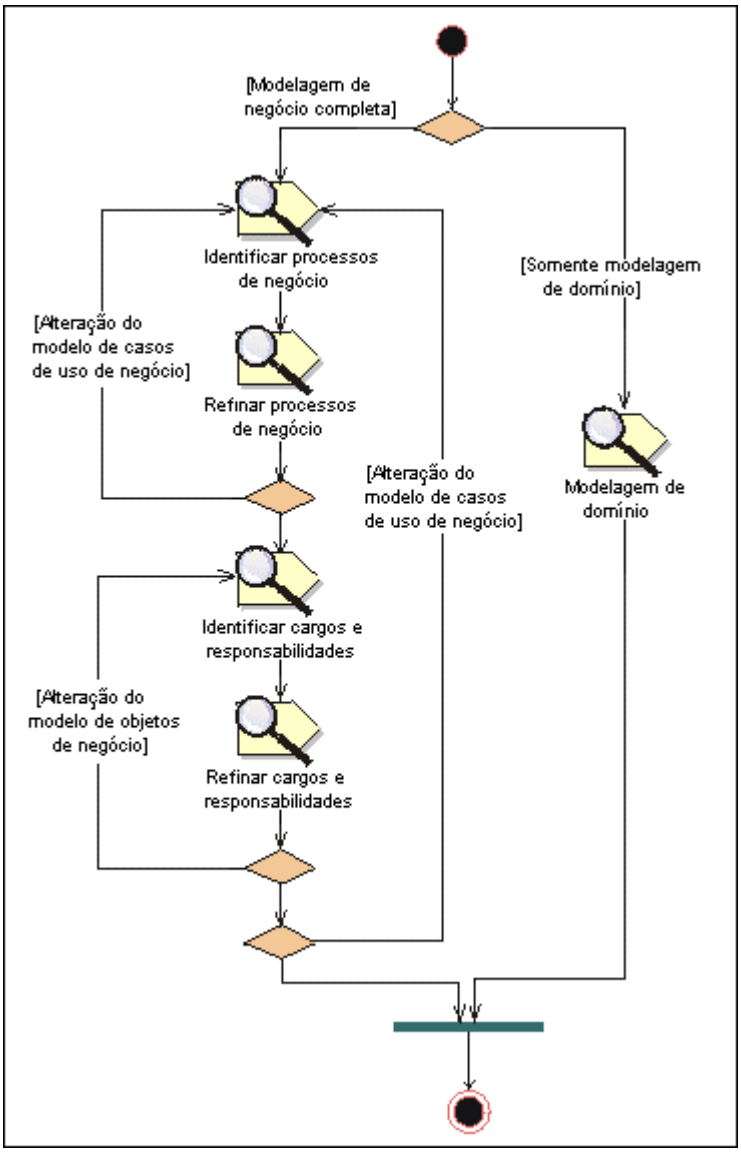


FIGURA 3.3 – Macro-atividade de Modelagem de Negócio [Extraído de RAT 2001]

2. Descrever negócio atual

A finalidade desta unidade é [RAT 2001]:

- Entender os processos e estrutura da organização alvo;
- Baseado neste entendimento, refinar os objetivos dos esforços de modelagem de negócio.

É composta das seguintes atividades: avaliar a organização alvo, fixar e ajustar metas e encontrar atores e casos de uso de negócio, executadas pelo analista de processo de negócio; e encontrar trabalhadores e unidade de negócio, executada pelo projetista de negócio [RAT 2001].

3. Identificar processos de negócio

Dentro de uma equipe é preciso chegar a um entendimento comum de quais limites da organização estão sendo descritos, e decidir quais os processos devem ser descritos em detalhes [RAT 2001].

A finalidade desta unidade é [RAT 2001]:

- Decidir uma terminologia;
- Esboçar o modelo de casos de uso de negócio;
- Priorizar os casos de uso que devem ser descritos em maiores detalhes.

É composta das seguintes atividades: manter regras de negócio, fixar e ajustar metas, definir a arquitetura de negócio, capturar um vocabulário de negócio comum e encontrar atores e casos de uso de negócio, executadas pelo analista de processo de negócio [RAT 2001].

4. Refinar processos de negócio

Cada caso de uso de negócio deve ser designado a um membro da equipe, que é responsável por descrevê-lo em detalhes. Agindo como um projetista de negócio, esta pessoa completará a definição do caso de uso de negócio, e conduzirá uma sessão de revisão para o caso de uso de negócio. Outros membros da equipe de modelagem de negócio são convidados para essa sessão de revisão para agir como revisores de modelo de negócios. O projetista de negócio pode também convidar representantes dos interessados no projeto, como os usuários finais [RAT 2001].

A finalidade desta unidade é [RAT 2001]:

- Detalhar a definição de um caso de uso de negócio;
- Verificar se o caso de uso de negócio reflete corretamente como o negócio é realizado.

É composta das seguintes atividades: estruturar o modelo de casos de uso de negócio, executada pelo analista de processo de negócio; detalhar um caso de uso de negócio, executada pelo projetista de negócio; e revisar o modelo de casos de uso de negócio, executada pelo revisor do modelo de negócio [RAT 2001].

5. Projetar realização de processos de negócio

A finalidade desta unidade é [RAT 2001]:

- Identificar todos os cargos, produtos, artefatos entregáveis, e eventos do negócio;
- Descrever como as realizações dos casos de uso de negócio são executadas pelos trabalhadores de negócio e entidades de negócio.

É composta das seguintes atividades: capturar um vocabulário comum, manter regras de negócio, definir arquitetura de negócio, executadas pelo analista de processo de

negócio; e encontrar trabalhadores e entidades de negócio, executadas pelo projetista de negócio [RAT 2001].

6. Refinar cargos e responsabilidades

A finalidade desta unidade é [RAT 2001]:

- Detalhar a definição de uma entidade de negócio;
- Detalhar as responsabilidades de um trabalhador de negócio;
- Formalmente verificar se o resultado da modelagem de negócio está conforme a visão de negócio dos interessados.

É composta das seguintes atividades: detalhar um trabalhador de negócio e detalhar uma entidade de negócio, executadas pelo projetista de negócio; e revisar o modelo de objetos de negócio, executada pelo revisor de modelo de negócio [RAT 2001].

7. Explorar automação de processos

A finalidade desta unidade é [RAT 2001]:

- Explorar quais as partes dos processos de negócio podem e devem ser automatizadas;
- Entender como os sistemas existentes, referem-se em relação a sistemas legados, instalados na organização;
- Derivar requisitos de sistema.

É composta das seguintes atividades: fixar e ajustar metas, executada pelo analista de processo de negócio; e definir requisitos de automação, executadas pelo projetista de negócio [RAT 2001].

8. Modelagem de domínio

A finalidade desta unidade é [RAT 2001]:

- Identificar os produtos, os bens entregáveis e os eventos importantes para o domínio de negócio;
- Detalhar a definição de uma entidade de negócio;
- Formalmente verificar se o resultado da modelagem de negócio obedece à visão de negócio dos interessados.

É composta das seguintes atividades: capturar um vocabulário comum e manter regras de negócio, executadas pelo analista de processo de negócio; encontrar trabalhadores e entidades de negócio, detalhar uma entidade de negócio, executadas pelo projetista de negócio; e revisar o modelo de objetos de negócio, executada pelo revisor de modelo de negócio [RAT 2001].

3.4.2 Macro-atividade de Requisitos

A macro-atividade de requisitos descreve como capturar e gerenciar os requisitos do sistema efetivamente e projetar uma interface de usuário, focando nas necessidades e objetivos dos usuários e de outros interessados importantes [RAT 99, RAT 2001].

As finalidades da macro-atividade de requisitos são [KRU 2000]:

- Estabelecer e manter um acordo com o cliente e outros interessados sobre o que o sistema deve fazer;
- Prover aos desenvolvedores do sistema um melhor entendimento dos requisitos do sistema;
- Definir os limites do sistema (delimitar o sistema);
- Prover a base para o planejamento do conteúdo técnico das iterações;
- Prover a base para as estimativas de custo e tempo necessários para se desenvolver o sistema;
- Definir uma interface de usuário para o sistema, focando nas necessidades e objetivos dos usuários.

Para alcançar esses objetivos, em primeiro lugar deve-se entender a definição e o escopo do problema que será resolvido com este sistema. As regras de negócio, modelo de casos de uso de negócio e modelo de objetos de negócio desenvolvidos durante a Modelagem de Negócio servem como entrada para este esforço. Os interessados são identificados e as solicitações dos interessados são elicitadas, agrupadas e analisadas [RAT 2001].

A macro-atividade de requisitos descreve como definir uma visão do sistema e traduzir a visão para o modelo de casos de uso que, com a especificação complementar, define os requisitos de software detalhados do sistema. Além disso, a macro-atividade de requisitos ajuda a gerenciar o escopo e a alteração dos requisitos do sistema [KRU 2000].

Trabalhadores e Artefatos

Os principais trabalhadores envolvidos nos requisitos são os seguintes [RAT 99, KRU 2000]:

- **Analista de sistemas:** conduz a elicitação de requisitos e modelagem de casos de uso por esboçar as funcionalidades e delimitar o sistema. Estabelece quais autores e casos de uso existem e como eles interagem.
- **Especificador de requisitos:** detalha toda ou parte da funcionalidade do sistema por descrever aspectos dos requisitos de um ou vários casos de uso e outros requisitos de suporte ao software.
- **Projetista de interface do usuário:** conduz e coordena a prototipação e projeto da interface do usuário, por:
  - Capturar os requisitos da interface do usuário, incluindo os requisitos de usabilidade;

- Construir protótipos de interfaces do usuário;
- Envolver outros interessados na interface do usuário, tais como usuários finais, em revisões de usabilidade e sessões de exame de uso;
- Revisar e fornecer retorno apropriado da implementação final da interface do usuário.

São também envolvidos nesta macro-atividade os seguintes trabalhadores [KRU 2000, RAT 2001]:

- **Arquiteto de software:** conduz e coordena atividades técnicas e os artefatos por todo o projeto. O arquiteto de software estabelece a estrutura completa para cada visão arquitetural: a decomposição da visão, o agrupamento de elementos e as interfaces entre os maiores grupos. Em contraste com a visão de outros trabalhadores, a visão do arquiteto é mais ampla do que profunda.
- **Revisor de requisitos:** planeja e conduz revisões formais do modelo de casos de uso.

Os artefatos principais da macro-atividade de modelagem de negócio são os seguintes [KRU 2000, RAT 99, RAT 2001]:

- **Plano de gerenciamento de sistemas:** descreve a documentação dos requisitos, tipos dos requisitos e seus atributos de requisitos respectivamente, especificando as informações e os mecanismos de controle que serão coletados e usados para medir, relatar, e controlar alterações nos requisitos do produto.
- **Documento visão:** define a visão dos interessados do produto que será desenvolvido, especificando em termos das necessidades e características dos interessados. Contendo um esboço geral do núcleo de requisitos do projeto, e provê uma base em alto-nível - as vezes contratual - para detalhar melhor os requisitos técnicos.
- **Modelo de casos de uso:** é um modelo das funções pretendidas para o sistema e seu ambiente. Serve como um contrato entre o cliente e os desenvolvedores. O modelo de casos de uso é usado como uma entrada essencial para as atividades de análise, projeto e teste.
- **Caso de uso:** define um conjunto de instâncias de casos de uso, onde cada instância é uma seqüência de ações que o sistema executa e que produz um resultado de valor considerável para um ator em particular.
- **Especificação complementar:** captura os requisitos do sistema que não são capturados no modelo de casos de uso (requisitos não-funcionais), tais como: requisitos regulatórios ou legais, e padrões da aplicação; atributos de qualidade para o sistema que está sendo construído, incluindo requisitos de usabilidade, segurança, desempenho e suportabilidade; e outros requisitos como sistema operacional, ambiente, requisitos de compatibilidade e considerações de projeto.

- **Especificação dos requisitos de software:** captura os requisitos de software completos para o sistema. Este artefato consiste de um pacote contendo os casos de uso do modelo de casos de uso e a especificação complementar.

Outros artefatos incluídos são os seguintes:

- **Glossário:** define termos importantes usados no projeto.
- **Atributos dos requisitos:** é um repositório de requisitos do projeto, atributos e dependências, são usados no acompanhamento da perspectiva de gerenciamento de requisitos.
- **Protótipo de interface do usuário:** protótipo em papel, imagens ou iterativo.
- **Cenário (*storyboard*) de um caso de uso:** descrição conceitual e lógica de como um caso de uso é executado pela interface do usuário, incluindo a interação entre o ator e o sistema.
- **Boundary Class:** modela a interação entre um ou mais atores e o sistema.
- **Ator:** define um conjunto coerente de papéis que usuários do sistema podem assumir quando interagem com o sistema.
- **Pacotes de casos de uso:** é uma coleção de casos de uso, atores, relacionamentos, diagramas e outros pacotes; é usado para estruturar o modelo de casos de uso por dividi-lo em pequenas partes.

Atividades da Macro-atividade de Requisitos

1. Analisar o problema

A finalidade desta unidade é chegar a um acordo do problema que será resolvido, identificar os interessados, definir os limites do sistema e identificar restrições impostas para o sistema [RAT 2001].

É composta das seguintes atividades: capturar um vocabulário comum, gerenciar dependências, desenvolver o plano de gerenciamento de requisitos, encontrar atores e casos de uso e desenvolver a visão, executadas pelo analista do sistema [RAT 2001].

2. Entender as necessidades dos interessados

A finalidade desta unidade é coletar e elicitar informações dos interessados para o projeto [RAT 2001].

É composta das seguintes atividades: elicitar requisitos dos interessados, capturar um vocabulário comum, gerenciar dependências, encontrar atores e casos de uso e desenvolver a visão, executadas pelo analista do sistema [RAT 2001].

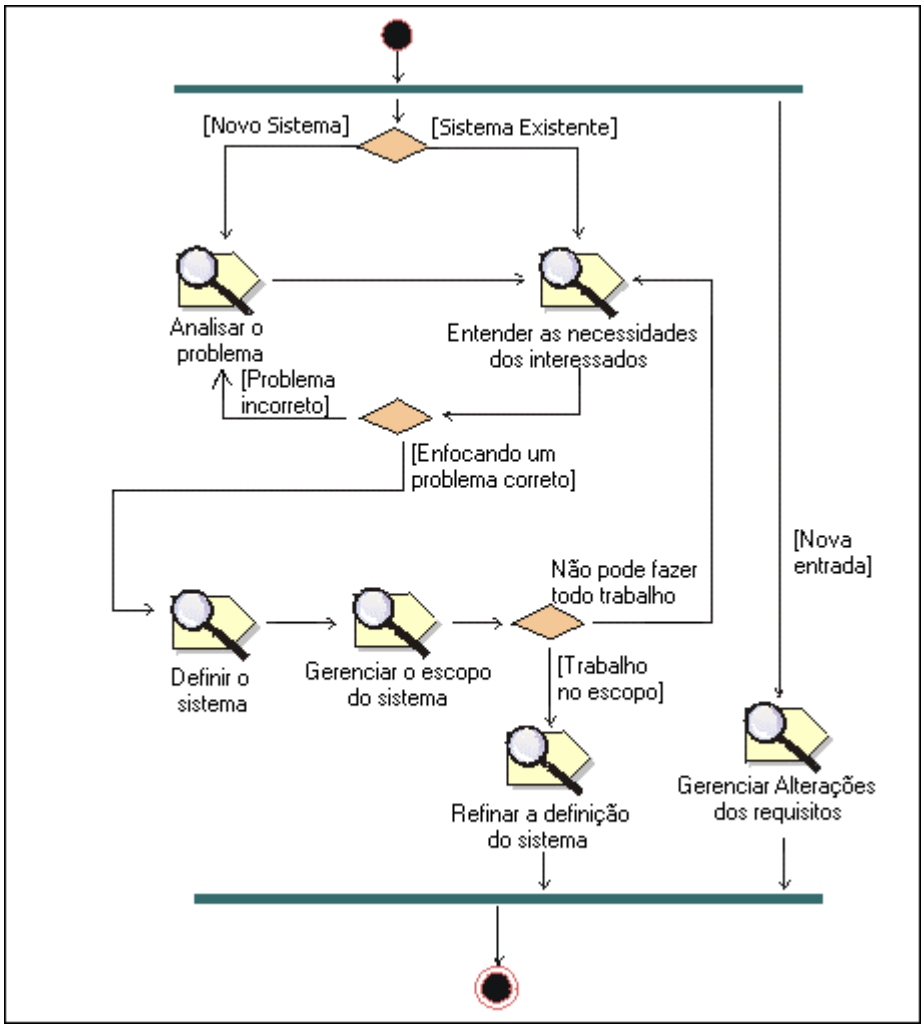


FIGURA 3.4 - Macro-atividade de Requisitos [Extraído de RAT 2001]

3. Definir o sistema

A finalidade desta unidade é nivelar a equipe do projeto no seu entendimento do sistema, executar uma análise de alto-nível das solicitações dos interessados coletadas, refinar a visão para incluir características do sistema, refinar o modelo de casos de uso, para incluir casos de uso esboçados e documentar formalmente os resultados em modelos e documentos [RAT 2001].

É composta das seguintes atividades: capturar um vocabulário comum, desenvolver a visão, gerenciar dependências e encontrar atores e casos de uso, executadas pelo analista do sistema [RAT 2001].

4. Gerenciar o escopo do sistema

A finalidade desta unidade é priorizar e refinar entradas para a seleção de características e requisitos que serão incluídos na iteração atual, definir o conjunto de casos

de uso que representam alguma funcionalidade central ou significante e definir quais atributos de requisitos e rastreabilidade serão mantidos [RAT 2001].

É composta das seguintes atividades: desenvolver a visão e gerenciar dependências, executadas pelo analista do sistema; e priorizar casos de uso, executada pelo arquiteto [RAT 2001].

5. Refinar a definição do sistema

A finalidade desta unidade é descrever o fluxo de eventos do caso de uso em detalhes, modelar e prototipar a interface do usuário, detalhar a especificação complementar e desenvolver especificação de requisitos de software (se mais detalhes é necessário) [RAT 2001].

É composta das seguintes atividades: detalhar um caso de uso e detalhar os requisitos do software, executadas pelo especificador de casos de uso; modelar a interface do usuário e prototipar a interface do usuário, executadas pelo projetista de interface do usuário [RAT 2001].

6. Gerenciar alteração nos requisitos

A finalidade desta unidade é avaliar formalmente solicitações de alteração submetidas e determinar seu impacto no conjunto de requisitos existentes, estruturar o modelo de casos de uso, levantar atributos dos requisitos e rastreabilidade apropriados e formalmente verificar se os resultados da macro-atividade de requisitos obedece a visão do cliente do sistema [RAT 2001].

É composta das seguintes atividades: gerenciar dependências, estruturar o modelo de casos de uso, executadas pelo analista do sistema; e revisar requisitos, executada pelo revisor de requisitos [RAT 2001].

3.4.3 Macro-atividade de Análise e Projeto

A macro-atividade de análise e projeto traduz os requisitos para especificações que descrevem como implementar o sistema. Para fazer esta tradução, é necessário entender os requisitos e transformá-los em um projeto (desenho) do sistema por selecionar a melhor estratégia de implementação. No início do projeto, é necessário estabelecer uma arquitetura robusta, de forma que seja possível projetar um sistema que é fácil de entender, construir e evoluir [KRU 2000].

As finalidades da macro-atividade de análise e projeto são [RAT 2001]:

- Transformar os requisitos em um projeto do que será sistema;
- Desenvolver uma arquitetura robusta para o sistema;
- Adaptar o projeto para adequar-se ao ambiente de implementação, projetando seu desempenho, robustez, escalabilidade e testabilidade, entre outras qualidades.

Trabalhadores e Artefatos

Os principais trabalhadores envolvidos na análise e projeto são os seguintes [RAT 2001, KRU 2000]:

- **Arquiteto de software:** conduz e coordena atividades técnicas e os artefatos por todo o projeto. Ele estabelece a estrutura completa para cada visão arquitetural: a decomposição da visão, o agrupamento de elementos e as interfaces entre os maiores grupos. Em contraste com a visão de outros trabalhadores, a visão do arquiteto é mais ampla do que profunda.
- **Projetista:** define responsabilidades, operações, atributos e relações de uma ou várias classes e determina como elas devem ser ajustadas para o ambiente de implementação. Além disso, o projetista deve ter responsabilidade por um ou mais pacotes ou subsistemas de projeto, incluindo qualquer classe pertencente ao pacote ou subsistema.

São também envolvidos nesta macro-atividade os seguintes trabalhadores [KRU 2000, RAT 2001]:

- **Projetista de base de dados:** é responsável por definir as estruturas e elementos específicos da base de dados do sistema, que são necessários para armazenar, recuperar e excluir objetos persistentes.
- **Projetista de cápsulas (somente em sistemas de tempo real):** focaliza em assegurar que o sistema é capaz de responder a eventos na hora certa, através do uso apropriado de técnicas de concorrência.
- **Revisor da arquitetura:** planeja e conduz revisões formais da arquitetura em geral.
- **Revisor de projeto:** planeja e conduz revisões formais do modelo de projeto.

Os artefatos principais da macro-atividade de análise e projeto são os seguintes [RAT 2001]:

- **Modelo de projeto:** é um modelo de objetos descrevendo a realização dos casos de uso e serve como uma abstração do modelo de implementação e seu código fonte. O modelo de projeto é usado como uma entrada essencial para as atividades de implementação e teste.
- **Documento de arquitetura do software:** fornece uma visão geral da arquitetura do sistema, usando algumas visões arquiteturais diferentes para descrever aspectos diferentes do sistema.

Outros artefatos incluídos na macro-atividade de análise e projeto são os seguintes [KRU 2000, RAT 2001]:

- **Modelo de implantação:** mostra a configuração de nodos de processamento em tempo de execução, os *links* de comunicação entre eles, e as instâncias de componentes e objetos que residem neles;

- **Modelo de análise:** é um modelo de objetos que descreve a realização de casos de uso, o qual serve como uma abstração do modelo de projeto.
- **Prova de conceito arquitetural:** é a solução – que pode ser simplesmente conceitual, para os requisitos significantes arquiteturalmente que são identificados na iteração inicial.
- **Arquitetura de referência:** é um padrão arquitetural pré-definido, ou conjunto de padrões, projetados e comprovados para uso em um contexto técnico e de negócio particular, junto com artefatos de suporte para permitir seu uso.
- **Interface:** que define um conjunto de comportamentos oferecido por um elemento do modelo, tais como: uma classe, um subsistema ou um componente.
- **Sinal:** é uma entidade de comunicação assíncrona que pode causar uma transição de estado em uma máquina de estados de um objeto que recebe este sinal.
- **Evento:** a especificação de uma ocorrência em espaço e tempo, uma ocorrência de algo para qual o sistema deve responder.
- **Protocolo:** uma especificação comum para um conjunto de portas do artefato cápsula.
- **Cápsula:** é um padrão de projeto específico o qual representa uma *thread* encapsulada de controle do sistema.
- **Realização de caso de uso:** descreve como um caso de uso particular é realizado dentro de um modelo de projeto, em termos de colaboração entre os objetos.
- **Classe de projeto:** é a descrição de um conjunto de objetos que compartilham as mesmas responsabilidades, relações, operações, atributos e semântica.
- **Classe de análise:** representa um modelo conceitual inicial para coisas no sistema as quais têm responsabilidades e comportamento.
- **Projeto de subsistema:** um elemento do modelo, o qual tem a semântica de um pacote (ele pode conter outros elementos do modelo) e uma classe (tem comportamento).
- **Projeto de pacote:** é uma coleção de classes, relacionamentos, realizações de casos de uso, diagramas e outros pacotes. É usado para estruturar o modelo de projeto por dividi-lo em pequenas partes.
- **Modelo de dados:** é um subconjunto do modelo de implementação, o qual descreve a representação física e lógica de dados persistentes no sistema.

Atividades da Macro-atividade de Análise e Projeto

1. Definir uma arquitetura candidata

A finalidade desta unidade é [RAT 2001]:

- Criar um desenho inicial da arquitetura do sistema, para isso é necessário definir um conjunto inicial de elementos significantes arquiteturalmente para ser usado como base nesta análise, definir um conjunto inicial de mecanismos de análise, definir a organização e camadas iniciais do sistema e definir as realizações de casos de uso que serão enfocadas pela iteração corrente;
- Identificar classes de análise dos casos de uso significantes arquiteturalmente;
- Atualizar a realização dos casos de uso com interação das classes de análise.

É composta das seguintes atividades: análise arquitetural, executada pelo arquiteto; e análise de casos de uso, executada pelo projetista [RAT 2001].

2. Refinar a arquitetura

A finalidade desta unidade é [RAT 2001]:

- Prover uma transição natural das atividades de análise para as atividades de projeto, identificando os elementos de projeto apropriados de elementos de análise e mecanismos de projeto apropriados de mecanismos de análise;
- Manter a consistência e integridade da arquitetura assegurando que novos elementos de projeto identificados para a iteração atual são integrados com os elementos de projeto pré-existentes e o reuso máximo de componentes disponíveis e elementos de projeto é alcançado o mais cedo possível nos esforços de projeto;
- Descrever a organização da arquitetura de implantação e execução do sistema;
- Organizar o modelo de implementação para fazer a transição entre o projeto e implantação.

É composta das seguintes atividades: identificar mecanismos de projeto, identificar elementos de projeto, incorporar elementos de projeto existentes, descrever a arquitetura de execução e descrever a distribuição, executadas pelo arquiteto; e revisar a arquitetura, executada pelo revisor de arquitetura [RAT 2001].

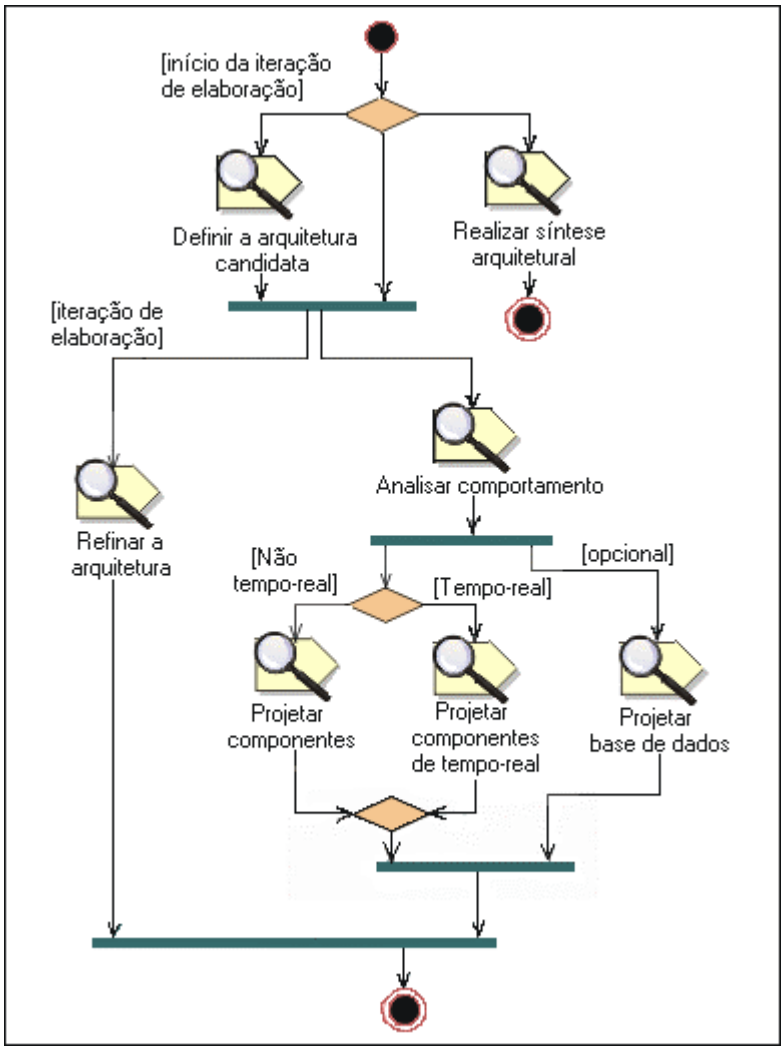


FIGURA 3.5 - Macro-atividade de Análise e Projeto [Extraído de RAT 2001]

3. Analisar comportamento

A finalidade desta unidade é transformar as descrições de comportamento fornecidas pelos casos de uso em um conjunto de elementos, sob os quais o projeto pode ser baseado [RAT 2001].

É composta das seguintes atividades: analisar casos de uso, executada pelo projetista; identificar elementos do projeto, executada pelo arquiteto de software; e revisar o projeto, executada pelo revisor do projeto [RAT 2001].

4. Projetar componentes

A finalidade desta unidade é [RAT 2001]:

- Refinar a definição de elementos do projeto por trabalhar os detalhes de como os elementos do projeto implementam o comportamento requerido deles;

- Refinar e atualizar as realizações de casos de uso, baseada nos novos elementos de projeto introduzidos;
- Revisar o projeto desenvolvido;
- Implementar os elementos de projeto como componentes;
- Testar os componentes implementados para verificar funcionalidades e satisfação dos requisitos ao nível de componentes/unidades.

É composta das seguintes atividades: projetar caso de uso, projetar subsistema e projetar classe, executadas pelo projetista; e revisar o projeto, executada pelo revisor do projeto [RAT 2001].

5. Projetar componentes de tempo-real

Esta macro-atividade é aplicada para projetos que usam o artefato cápsula como um elemento de projeto primário, dentro do contexto de um sistema de tempo real ou reativo [RAT 2001].

A finalidade desta unidade é [RAT 2001]:

- Refinar a definição dos elementos do projeto por trabalhar os detalhes de como os elementos do projeto implementam o comportamento requerido deles;
- Refinar e atualizar as realizações de casos de uso, baseada nos novos elementos de projeto identificados;
- Revisar o projeto desenvolvido;
- Implementar os elementos de projeto como componentes;
- Testar os componentes implementados para verificar funcionalidades e satisfação dos requisitos ao nível de componentes/unidades.

É composta das seguintes atividades: projetar cápsula, executada pelo projetista de cápsula; projetar caso de uso, projetar subsistema e projetar classe, executadas pelo projetista; e revisar o projeto, executada pelo revisor do projeto [RAT 2001].

6. Projetar a base de dados

A finalidade desta unidade é [RAT 2001]:

- Identificar as classes persistentes do projeto;
- Projetar as estruturas de base de dados apropriadas para armazenar as classes persistentes;
- Definir mecanismos e estratégias para armazenar e recuperar dados persistentes de modo que os critérios de desempenho para o sistema sejam encontrados.

É composta das seguintes atividades: projetar classes, executada pelo projetista; projetar base de dados, executada pelo projetista de base de dados; e revisar o projeto, executada pelo revisor do projeto [RAT 2001].

7. Projetar a base de dados

A finalidade desta unidade é construir e avaliar a prova de conceito arquitetural, com o objetivo de mostrar o que existe, ou é provável que exista, uma solução na qual satisfará os requisitos arquiteturalmente significantes, mostrando que o sistema, como projetado, é viável [KRU 2000, RAT 2001].

É composta das seguintes atividades: análise arquitetural, construir prova de conceito arquitetural, avaliar a viabilidade da prova de conceito arquitetural, executada pelo arquiteto de software [RAT 2001].

3.4.4 Macro-atividade de Implementação

A macro-atividade de implementação introduz os conceitos de protótipos e integração incremental [RAT 2001].

As finalidades da macro-atividade de implementação são as seguintes [RAT 98, RAT 2001]:

- Definir a organização do código em termos de implementação de subsistemas organizados em camadas;
- Implementar classes e objetos em termos de componentes (arquivos fontes, binários, executáveis e outros);
- Testar os componentes desenvolvidos como unidades;
- Integrar em um sistema executável os resultados produzidos pelos implementadores individuais ou equipes.

O escopo do teste, dentro da macro-atividade de implementação, é limitado ao teste de unidade de componentes individuais.

Trabalhadores e Artefatos

Os principais trabalhadores envolvidos na implementação são os seguintes [RAT 2001, KRU 2000]:

- **Implementador:** é responsável por desenvolver e testar os componentes de acordo com os padrões adotados pelo projeto para que estes possam ser integrados em subsistemas maiores.
- **Integrador de sistema:** é responsável por planejar a integração do sistema, e por integrar os componentes testados pelos implementadores para produzir um *build* (versão operacional do sistema).

São também envolvidos nesta macro-atividade os seguintes trabalhadores [RAT 2001]:

- **Arquiteto de software:** conduz e coordena atividades técnicas e os artefatos por todo o projeto. Ele estabelece a estrutura completa para cada visão arquitetural:

a decomposição da visão, o agrupamento de elementos e as interfaces entre os maiores grupos. Em contraste com a visão de outros trabalhadores, a visão do arquiteto é mais ampla do que profunda.

- **Revisor de código:** assegura a qualidade do código-fonte, e planeja e conduz revisões do código-fonte. O revisor de código é também responsável por qualquer retorno de re-trabalho que resulta das atividades de revisão.

Os artefatos principais da macro-atividade de implementação são os seguintes:

- **Subsistema de implementação:** é uma coleção de componentes e outros subsistemas de implementação que os contém. Componentes incluem tanto os componentes entregáveis, tais como executáveis, e componentes a partir dos quais os componentes entregáveis são produzidos, tais como arquivos código-fonte.
- **Componente:** uma parte de código de software (fonte, binário, ou executável) ou um arquivo contendo informação (por exemplo, um arquivo de inicialização ou um arquivo leiame). Um componente pode também ser um agregado de outros componentes.
- **Plano de integração:** fornece um plano detalhado para integração dentro de uma iteração.
- **Modelo de implementação:** é uma coleção de componentes e os subsistemas de implementação que contém esses componentes.
- **Build:** compreende um ou mais componentes, construídos a partir de outros componentes, geralmente por um processo de compilação ou ligação de código-fonte.

Atividades da Macro-atividade de Implementação

1. Estruturar o modelo de implementação

A finalidade desta unidade é assegurar que o modelo de implementação é organizado de tal maneira que torne o desenvolvimento de componentes e o processo de construção livres de conflitos. Um modelo bem organizado previne problemas de gerenciamento de configuração e permite que o produto seja construído sucessivamente em versões de integração. Essa atividade é realizada no início da fase de elaboração [RAT 2001].

Consiste da atividade estruturar o modelo de implementação, executada pelo arquiteto [RAT 2001].

2. Planejar a integração

A finalidade desta unidade é planejar quais os subsistemas que devem ser implementados, e a ordem na qual os subsistemas devem ser integrados na iteração corrente [RAT 2001].

Consiste da atividade planejar a integração do sistema, executada pelo integrador de sistema [RAT 2001].

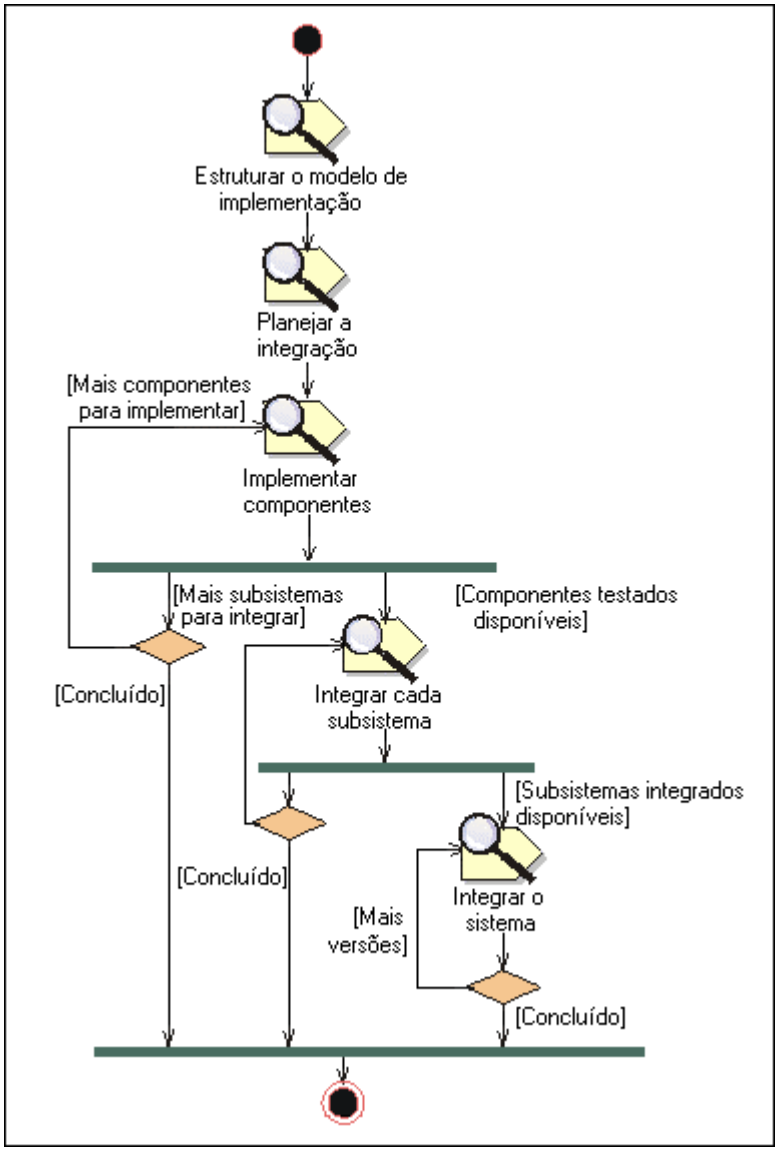


FIGURA 3.6 - Macro-atividade de Implementação [Extraído de RAT 2001]

3. Implementar os componentes

Esta macro-atividade tem as seguintes finalidades [RAT 2001]:

- O implementador implementa as classes e objetos do modelo de projeto, escreve o código fonte, adapta componentes existentes, compila, liga (*link*) e executa. Se

defeitos no projeto são descobertos, o implementador submete retorno de re-trabalho no projeto.

- O implementador também determina defeitos no código e executa testes de unidade para verificar as alterações. Então, o código é revisado para assegurar qualidade e as guias de programação são seguidas.

É composta das seguintes atividades: planejar integração do subsistema, executada pelo integrador; implementar componentes, determinar um defeito e executar teste de unidade, executadas pelo implementador; e revisar o código, executada pelo revisor de código [RAT 2001].

4. Integrar cada subsistema

Se vários implementadores trabalham em um mesmo subsistema de implementação, um dos implementadores é responsável por integrar os componentes novos e alterados em uma nova versão do subsistema de implementação [RAT 2001].

Consiste da atividade integrar subsistema, executada pelo implementador [RAT 2001].

5. Integrar o sistema

O integrador integra o sistema, de acordo com o plano de integração, acrescentando os subsistemas liberados em uma área de integração, e criando a versão (*build*). A versão passa por um teste de integração, e após o último incremento pode passar por um teste de sistema [RAT 2001].

Consiste da atividade integrar o sistema, executada pelo integrador de sistema [RAT 2001].

3.4.5 Macro-atividade de Teste

As finalidades da macro-atividade de teste são as seguintes [RAT 2001]:

- Verificar a interação de componentes;
- Verificar a integração apropriada de todos os componentes;
- Verificar se todos os requisitos têm sido implementados corretamente;
- Identificar e assegurar que os defeitos são corrigidos antes da implantação do sistema.

Trabalhadores e Artefatos

Os principais trabalhadores envolvidos no teste são os seguintes [RAT 99, KRU 2000]:

- **Projetista de teste:** é responsável pelo planejamento, projeto, implementação e avaliação do teste. Isto envolve geração de planos de teste e modelos de teste; implementação de procedimentos de teste; avaliação do teste de cobertura, dos resultados do teste e efetividade do teste; e a geração de um sumário de avaliação do teste.
- **Testador:** é responsável por organizar e executar o teste de sistema. Isto inclui esforços de montagem e execução de testes, avaliação da execução do teste, recuperação de erros, avaliação dos resultados do teste e documentação de defeitos identificados.

São também envolvidos nesta macro-atividade os seguintes trabalhadores:

- **Projetista:** é responsável por projetar teste de pacotes e classes.
- **Implementador:** executa teste de unidade nos componentes que desenvolve e implementa componentes e subsistemas de teste (utilizados nos testes).

Os artefatos principais da macro-atividade de testes são os seguintes [RAT 2001]:

- **Plano de teste:** contém informações sobre a finalidade e objetivos de testes no projeto. Este plano especifica as estratégias que serão usadas e os recursos necessários para implementar e executar o teste.
- **Modelo de teste:** é a representação do que será testado e como será testado. O modelo de teste inclui a coleção dos casos de teste, procedimentos de teste, scripts de teste, e os resultados esperados dos testes com a descrição de seus relacionamentos.
- **Resultados de teste:** contém um repositório de dados capturados durante a execução de testes e é usado para calcular diferentes medidas principais de teste.
- **Documento de Análise *Workload*:** identifica as variáveis e define seus valores usados em diferentes testes de desempenho para simular e emular as características de atores, funções de negócio do usuário final, carga e volume.
- **Procedimentos de teste:** é um conjunto de instruções detalhadas para organização, execução e avaliação de resultados para um dado caso de teste.
- **Caso de teste:** é um conjunto de entradas de teste, condições de execução, resultados esperados desenvolvidos para um objetivo particular, tais como exercitar um caminho de programa particular, ou verificar conformidade com um requisito específico.
- **Scripts de teste:** instruções legíveis ao computador que automatizam a execução de um procedimento de teste. São bastante utilizados em ferramentas de teste.
- **Sumário de avaliação do teste:** organiza e apresenta os resultados do teste e medições principais de teste para revisão e avaliação.

Outros artefatos incluídos são os seguintes [RAT 2001]:

- **Classes e pacotes de teste:** identificar a funcionalidade específica do teste para facilitar ou automatizar os testes.
- **Componentes e subsistemas de teste:** identificar a funcionalidade específica do teste para facilitar ou automatizar os testes.

Atividades da Macro-atividade de Teste

1. Planejar o teste

A finalidade desta unidade é identificar e descrever o teste que será implementado e executado. Esta atividade é acompanhada por gerar um plano de teste que contém os requisitos para o teste e estratégias de teste. Um plano de teste único pode ser desenvolvido, descrevendo todos os tipos de teste que serão implementados e executados, ou um plano de teste por tipo de teste pode ser desenvolvido. Quando o plano de teste é concluído, os esforços de teste podem ser mensurados e gerenciados [RAT 2001].

Consiste da atividade planejar o teste, executada pelo projetista de teste [RAT 2001].

2. Projetar o teste

A finalidade desta unidade é identificar, descrever e gerar o modelo de teste e os artefatos relacionados. O projeto de teste é feito para assegurar que os esforços de implementação e execução do teste são eficientes e efetivos [RAT 2001].

Consiste da atividade projetar o teste, executada pelo projetista de teste [RAT 2001].

3. Implementar o teste

A finalidade desta unidade é implementar os procedimentos de teste que foram definidos no projeto de teste. A programação e a gravação de procedimentos de teste podem ser feitas dentro do contexto de uma ferramenta de automação de teste ou ambiente de programação [RAT 2001]. O artefato de saída é uma versão executável pelo computador de um procedimento de teste, referenciado como um script de teste.

É composta das seguintes atividades: implementar o teste, executada pelo implementador de teste; projetar classes e pacotes de teste, executada pelo projetista; e implementar componentes e subsistemas de teste, executada pelo implementador [RAT 2001].

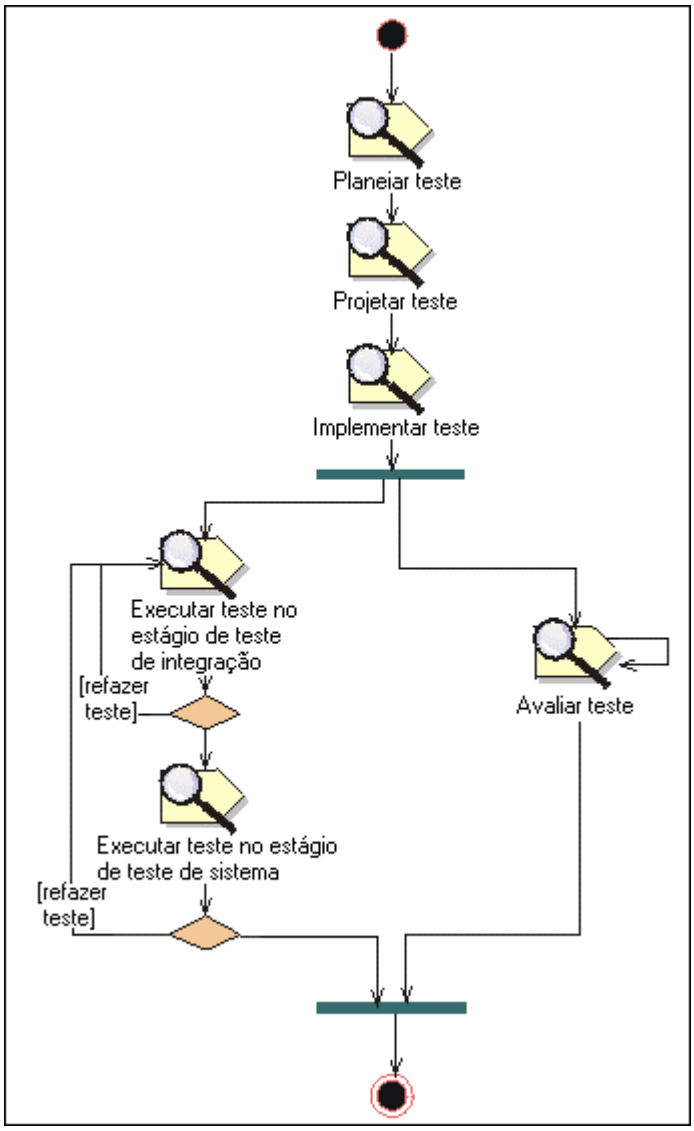


FIGURA 3.7 - Macro-atividade de Teste [Extraído de RAT 2001]

4. Executar teste no estágio de teste de integração

A finalidade desta unidade é assegurar que, durante a montagem, os componentes do sistema colaboram como pretendido, e também os incrementos têm o comportamento certo [RAT 2001, KRU 2000]. O integrador do sistema compila e liga o sistema em incrementos [RAT 2001]. Para cada incremento, a funcionalidade que foi adicionada é testada, e testes de regressão são executados, produzindo os resultados de teste [RAT 2001].

Dentro de uma iteração, testes de integração são executados várias vezes, até que todo sistema tenha sido integrado com sucesso [RAT 2001].

Consiste da atividade executar teste, executada pelo testador [RAT 2001].

5. Executar teste no estágio de teste de sistema

A finalidade desta unidade é assegurar que o sistema completo funciona como pretendido. O integrador do sistema compila e liga o sistema em incrementos. Para cada incremento, a funcionalidade que foi adicionada é testada, e testes de regressão são executados, produzindo os resultados de teste [RAT 2001].

Dentro de uma iteração, testes de sistema são executados várias vezes, até que todo sistema completo funcione como pretendido e encontre os critérios de conclusão ou sucesso do teste [RAT 2001].

Consiste da atividade executar teste, executada pelo testador [RAT 2001].

6. Avaliar o teste

A finalidade desta unidade é entregar e um resumo de avaliação do teste. Essa atividade é acompanhada por revisar e acompanhar os resultados do teste identificando e registrando solicitações de alteração, e calculando medidas principais de teste em um formato organizado e é usado para avaliar a qualidade do objetivo de teste e a qualidade do processo de teste [RAT 2001].

Consiste da atividade avaliar o teste, executada pelo projetista de teste [RAT 2001].

3.4.6 Macro-atividade de Implantação

A macro-atividade de implantação descreve as atividades associadas a assegurar que o produto de software está disponível para seus usuários finais. A macro-atividade de implantação descreve três modos de implantação do produto [RAT 2001]:

- Instalação customizada;
- Oferecer produto *"shrink wrap"*
- Acesso ao software pela internet.

Trabalhadores e Artefatos

Os principais trabalhadores envolvidos na implantação são os seguintes [KRU 2000]:

- **Gerente de implantação:** planeja a transição do produto para a comunidade de usuários finais e documenta o sistema em vários documentos associados.
- **Escritor técnico:** planeja e produz o material de suporte ao usuário final.
- **Desenvolvedor de curso:** planeja e produz o material de treinamento
- **Artista gráfico:** é responsável por todo o trabalho de arte relacionado ao produto.
- **Implementador:** produz scripts de instalação e artefatos relacionados, que ajudarão o usuário final a instalar o produto.

Os artefatos principais da macro-atividade de implantação são os seguintes [RAT 2001]:

- **Plano de implantação:** descreve o conjunto de tarefas necessárias para instalar e testar o produto desenvolvido e como ele pode ser efetivamente repassado para a comunidade de usuários.
- **Produto:** o sistema executável.
- **Artefatos de instalação:** referem-se as instruções documentadas e ao software requerido para instalar o produto.
- **Notas de versões:** identifica as alterações e erros conhecidos, em uma versão de um *build*, ou unidade de implantação, que é disponibilizada para uso.
- **Notas de materiais:** lista as partes constituintes de uma determinada versão de um produto, e onde as partes físicas podem ser encontradas. Descreve as alterações feitas na versão e refere-se a como o produto pode ser instalado.
- **Material de suporte ao usuário final:** material que apóia o usuário final no aprendizado, uso, operação e manutenção do produto.
- **Material de treinamento:** refere-se ao material que é usado nos programas de treinamento ou cursos para apoiar o usuário final no uso, operação e/ou manutenção do produto.

Outros artefatos incluídos são os seguintes [RAT 2001]:

- **Unidade de implantação:** consiste de um *build* (uma coleção executável de componentes), documentos (material de suporte ao usuário final e notas de versões) e artefatos de instalação.
- **Arte do produto:** inclui o texto e a arte final que será usada para "marcar" o produto.

Atividades da Macro-atividade de Implantação

1. Planejar a implantação

A finalidade desta unidade é planejar a implantação do produto. Para planejar a implantação é necessário decidir como e onde o produto será disponibilizado para o usuário final [RAT 2001].

Consiste da atividade desenvolver plano de implantação e definir notas de materiais, executadas pelo gerente de implantação [RAT 2001].

2. Desenvolver material de suporte

A finalidade desta unidade é produzir o material de suporte requerido pelo usuário final para instalar, operar, usar e manter o sistema entregue. Esta atividade também inclui material de treinamento para todos os que necessitam utilizar o novo sistema [RAT 2001].

Consiste das atividades desenvolver material de treinamento, executada pelo desenvolvedor de curso; e desenvolver material de suporte, executada pelo escritor técnico [RAT 2001].

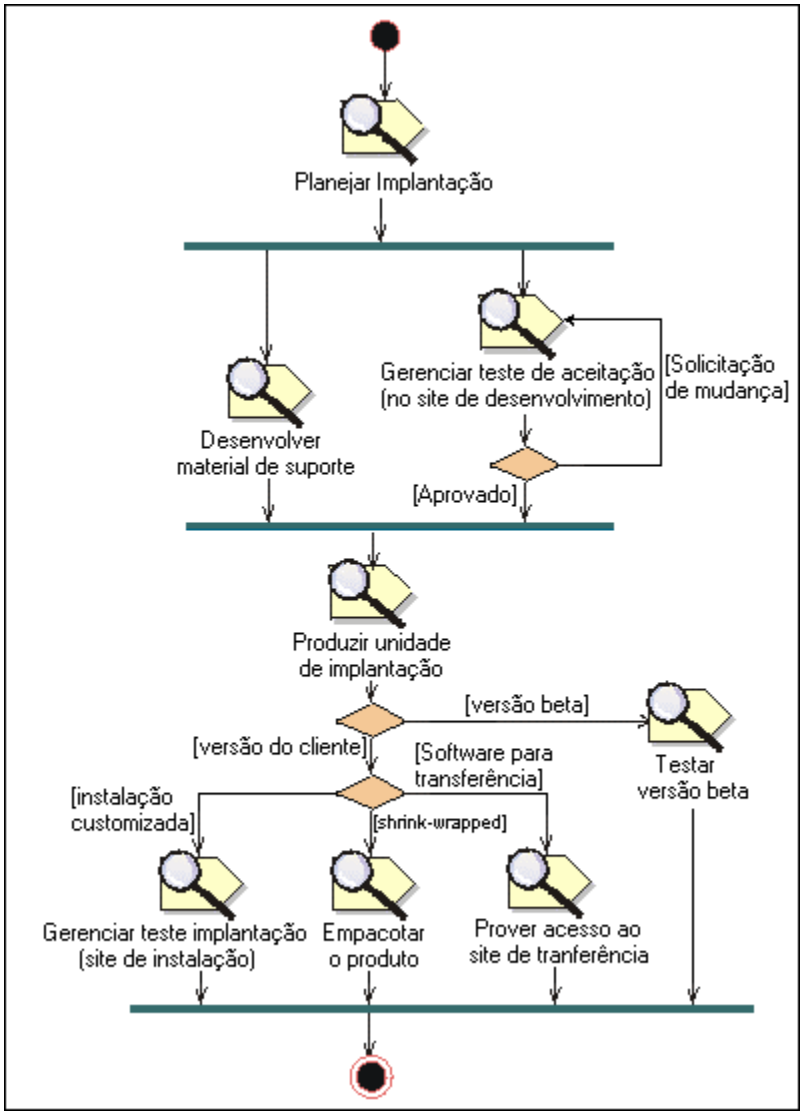


FIGURA 3.8 - Macro-atividade de Implantação [Extraído de RAT 2001]

3. Gerenciar teste de aceitação

A finalidade desta unidade é garantir que o produto é adequadamente testado antes de suas versões. O teste de aceitação pode ser no *site* de desenvolvimento e/ou no *site* de instalação [RAT 2001].

Consiste da atividade gerenciar testes de aceitação, executada pelo gerente de implantação [RAT 2001].

4. Produzir unidade de implantação

A finalidade desta unidade é criar uma unidade de implantação. Uma unidade de implantação consiste do software e dos artefatos requeridos para efetivamente instalar e usar o produto [RAT 2001].

Consiste das atividades escrever notas de versões, executada pelo gerente de implantação; e desenvolver artefatos de instalação, executada pelo implementador [RAT 2001].

5. Testar versão *beta*

Testes *beta* requerem a entrega do software para ser instalado pelos usuários finais [RAT 99]. Os usuários finais devem fornecer retorno sobre o desempenho e usabilidade do software [KRU 2000].

Consiste da atividade gerenciar testes *beta*, executada pelo gerente de implantação [RAT 2001].

6. Testar produto (no *site* de instalação)

A finalidade desta unidade é instalar o produto no *site* e este ser testado pelo cliente. Baseado nos testes de iteração precedentes e compromisso do cliente, o teste final no *site*, deve ser apenas a formalização de aceitação do sistema pelo cliente [RAT 2001].

Consiste das atividades escrever notas de versões, executada pelo gerente de implantação; e desenvolver artefatos de instalação, executada pelo implementador [RAT 2001].

7. Empacotar produto

A finalidade desta unidade é empacotar o produto, nos casos em que é necessário [RAT 2001].

Consiste das atividades liberar para fabricação e verificar o produto fabricado, executadas pelo gerente de implantação; e criar arte final do produto, executada pelo artista gráfico [RAT 2001].

8. Prover acesso ao *site* de transferência

Garantir que o produto está inteiramente acessível, através dos navegadores e *sites* [RAT 2001].

Consiste da atividade prover acesso ao *site* de transferência, executada pelo gerente de implantação [RAT 2001].

3.4.7 Macro-atividade de Gerenciamento de Alteração e Configuração

A finalidade da macro-atividade de Gerenciamento de Alteração e Configuração é acompanhar e manter a integridade dos bens envolvidos no projeto. Durante o ciclo de vida de desenvolvimento, muitos artefatos valiosos são criados. O desenvolvimento destes artefatos é um trabalho intensivo, e eles representam um investimento significante. Estes bens devem ser guardados e estar disponíveis para o reuso. Estes artefatos evoluem e, especialmente no desenvolvimento iterativo, são atualizados repetidamente. Os membros do projeto devem ser capazes de identificar e localizar os artefatos, selecionar a versão apropriada de um artefato, olhar sua história para entender seu estado corrente e a razão de suas alterações, e verificar quem é atualmente o responsável por ele [KRU 2000, RAT 2001].

Ao mesmo tempo, a equipe de projeto deve acompanhar a evolução do produto, capturar e gerenciar solicitações de alteração e então implementar essas modificações de um modo consistente, através do conjunto de artefatos.

Finalmente, para suportar A macro-atividade de gerenciamento de projetos, é necessário fornecer informações sobre o status dos artefatos principais do programa e reunir métricas relacionadas a suas alterações [KRU 2000].

O Cubo GAC

Gerenciamento de Alteração e Configuração (GAC) cobre três funções interdependentes. Estes aspectos principais podem ser ilustrados usando um cubo. O cubo tem três fases e cada fase examina um aspecto do problema [RAT 99, KRU 2000]:

- **Gerenciamento de configuração (GC):** descreve a estrutura do produto e seus itens de configuração constituintes, que são tratados como entidades únicas versionáveis. GC trata com a definição de configuração, construção, identificação e coleção de artefatos versionados em conjuntos constituintes e manutenção da rastreabilidade entre estas versões [RAT 99, KRU 2000].
- **Gerenciamento de solicitação de alteração (GSA):** enfoca a infra-estrutura organizacional requerida para avaliar o custo, cronograma, e impactos de uma solicitação de alteração de um produto existente. Gerenciamento de solicitação de alteração enfoca o trabalho da Equipe de Revisão de Alterações ou Conselho de Controle de Alterações [RAT 99, KRU 2000].
- **Medição:** trata com a extração de informações para gerenciamento do projeto, as ferramentas que suportam o gerenciamento de configuração e as funções de gerenciamento de solicitação de alteração. As medições são usadas para determinar o estado do produto, baseado no tipo, número, taxa e severidade de defeitos, encontrados e fixados, durante o desenvolvimento do produto [RAT 99, KRU 2000].

Trabalhadores e Artefatos

Os principais trabalhadores envolvidos no gerenciamento de alteração e configuração são os seguintes [KRU 2000]:

- **Gerente de configuração:** é responsável por instalar a estrutura do produto no sistema de GC, por definir e alocar área de trabalho (*workspaces*) para desenvolvedores e para integração. O gerente de configuração extrai o relatório de métricas e status para o gerente do projeto [RAT 99, KRU 2000].
- **Gerente de controle de alteração:** administra o processo de controle de alteração. Este cargo é normalmente representado por um **conselho de controle de Alteração**, o qual deve consistir de representantes de todas partes interessadas, incluindo cliente, desenvolvedores e usuários. O gerente de controle de alteração é também responsável por definir o processo de gerenciamento de solicitação de alteração, o qual deve ser documentado no plano de gerenciamento de controle.

Os seguintes trabalhadores também estão envolvidos nesta macro-atividade [RAT 99, KRU 2000]:

- **Integrador:** acessa alterações na área de trabalho de integração e constrói o produto.
- **Qualquer trabalhador:** pode, de acordo com seus privilégios de acesso, acessar os artefatos que necessitam para implementar as alterações, pelas as quais são responsáveis.

Os artefatos principais da macro-atividade de gerenciamento de alteração e configuração são os seguintes [RAT 99, KRU 2000]:

- **Plano de gerenciamento de configuração:** descreve a política e as práticas que serão usadas no projeto para gerenciamento de configuração: versões, variáveis, áreas de trabalho e procedimentos para gerenciamento de alteração, *builders* e *releases*. O plano define as regras e as responsabilidades para o conselho de controle de alterações. Este plano faz parte do plano de desenvolvimento de software. O plano de gerenciamento de configuração detalha o cronograma das atividades, as responsabilidades assinaladas, e os recursos requeridos.
- **Solicitação de alteração:** são usadas para documentar defeitos, alteração nos requisitos, ou novas funcionalidades para o sistema. Cada solicitação de alteração é associada com o originador e a causa origem. Por último, a análise de impacto reúne o impacto da alteração em termos de artefatos afetados, custo e cronograma.
- **Configuration Audit Findings:** identifica uma configuração-base, identifica qualquer artefato requerido ausente, e identifica requisitos falhos ou testados não-testados completamente.
- **Repositório do projeto:** armazena todas as versões de arquivos e diretórios do projeto. Ele armazena também todos os dados e metadados associados com os arquivos e diretórios.

- **Área de trabalho:** permite acessar os artefatos e recursos requeridos para desenvolver e montar o produto que será entregue. Há dois tipos de área de trabalho, que são de desenvolvimento e de integração.

Atividades da Macro-atividade de Gerenciamento de Alteração e Configuração

Existem duas macro-atividades inter-relacionadas no gerenciamento e configuração de alterações; um da perspectiva de gerenciamento de configuração da estrutura de produto, e um segundo de uma perspectiva de ciclo de vida de uma solicitação de alteração [KRU 2000].

1. Planejar a configuração do projeto e controle de alteração

O plano de gerenciamento de configuração descreve todas as atividades relacionadas ao gerenciamento de configuração, que devem ser executadas ao longo do ciclo de vida do projeto. A finalidade desta unidade é [RAT 2001, KRU 2000]:

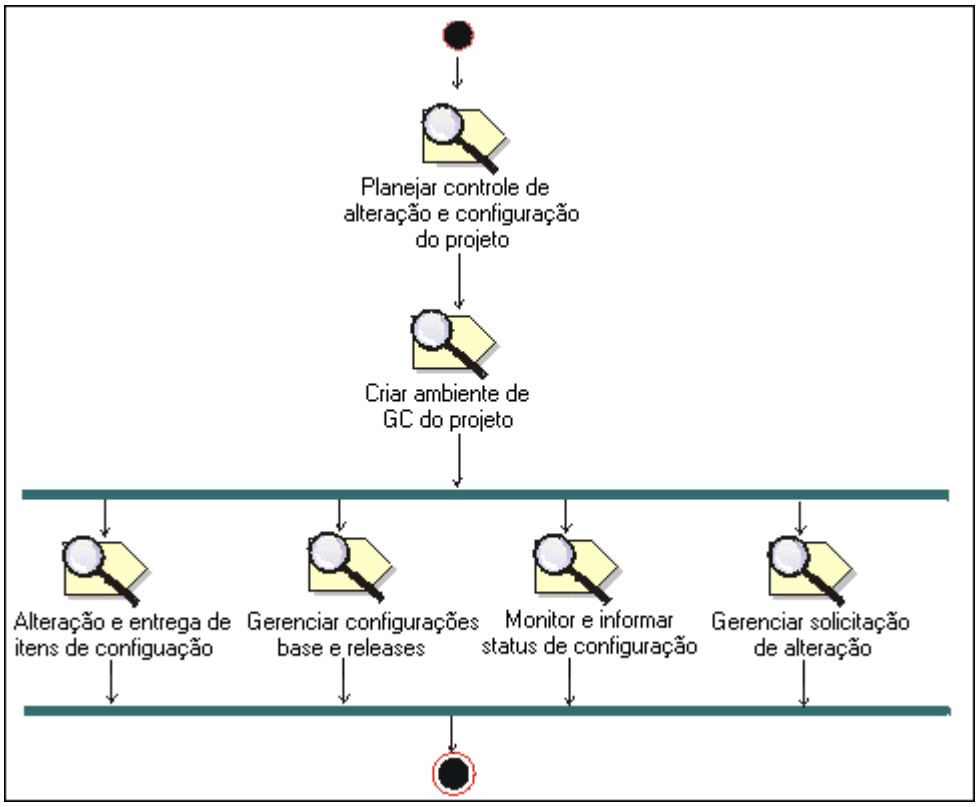


FIGURA 3.9 - Macro-atividade de Gerenciamento de Configuração e Alteração [Extraído de RAT 2001]

- Estabelecer políticas de gerenciamento de configuração do projeto;
- Estabelecer políticas e processos para controlar as alterações nos produtos;

- Documentar essas informações no plano de gerenciamento de configuração.

A finalidade de se ter padrões e processos de controle de alteração documentados, é assegurar que alterações realizadas no projeto são feitas de uma maneira consistente, e os interessados são informados do estado do produtos, alterações e impactos no cronograma e nos custos dessas alterações [RAT 2001, KRU 2000].

Esta unidade é composta pelas atividades: estabelecer políticas de GC para o projeto, escrever o plano de GC, executadas pelo gerente de configuração; e estabelecer um processo de controle de alteração do projeto, executada pelo gerente de controle de alteração [RAT 2001, KRU 2000].

2. Criar um ambiente de gerenciamento de configuração do projeto

A finalidade desta unidade é estabelecer um ambiente, criar e manter repositórios de dados, onde todos os produtos podem ser desenvolvidos, construídos e estar disponíveis para manutenção e reuso. O ambiente de GC provê aos desenvolvedores e integradores áreas de trabalho públicas e privadas, onde eles constroem e integram o software [RAT 2001, KRU 2000].

Esta unidade é composta pelas atividades: configurar o ambiente GC, executada pelo gerente de configuração; e criar uma área de trabalho para integração, executada pelo integrador de sistema [RAT 2001, KRU 2000].

3. Alterar e entregar itens de configuração

Esta unidade descreve como qualquer trabalhador pode criar uma área de trabalho. Dentro de uma área de trabalho, um trabalhador pode acessar artefatos do projeto, fazer alterações nestes artefatos e entregar essas alterações para inclusão no produto completo. A entrega das alterações é feita em uma área de trabalho de integração que pode incluir submissões de múltiplos trabalhadores. A idéia é entregar contribuições individuais e tornar estas visíveis para os desenvolvedores no próximo ciclo de desenvolvimento [RAT 2001].

O integrador de sistemas, a partir da área de trabalho de integração, constrói o sistema, cria configurações-base, e então torna essas configurações-base disponíveis para o resto da equipe de desenvolvimento [RAT 2001].

Esta unidade é composta pelas atividades: criar áreas de trabalho de desenvolvimento, fazer alterações para os itens de configuração, entregar alterações, e atualizar áreas de trabalho, executadas por qualquer trabalhador; criar configurações-base e promover configurações-base, executadas pelo integrador do sistema [RAT 2001].

4. Gerenciar configurações-base e *releases*

Uma configuração-base é uma descrição de todas as versões de artefatos que constituem o produto em qualquer tempo [RAT 2001].

A finalidade desta unidade é assegurar que subsistemas, quando alcançam um determinado nível de maturidade, são criadas configurações-base, e disponibilizadas para *release* ou reuso em iterações subseqüentes do projeto e/ou outros projetos [RAT 2001].

É composta pelas atividades: criar unidade de implantação, executada pelo gerente de configuração; criar configurações-base e promover configurações-base, executadas pelo integrador do sistema [RAT 2001].

5. Monitorar e informar status de configuração

A finalidade desta unidade é [RAT 2001]:

- Determinar que produtos encontram tanto as funcionalidades físicas e funcionais;
- Determinar que artefatos estão em uma biblioteca controlada;
- Assegurar que artefatos e configurações-base estão disponíveis;
- Suportar atividades de apresentação do status de configuração, que são baseadas em registros formalizados, relatórios do status das alterações propostas e o status de implementação das alterações propostas;
- Facilitar a revisão dos produtos, através do rastreamento de defeitos e relatório das atividades.

É composta pelas atividades: relatar o status de configuração do projeto, e executar auditorias de configuração, executadas pelo gerente de configuração [RAT 2001].

6. Gerenciar solicitações de alteração

A finalidade de um processo de controle de alteração documentado e padronizado é assegurar que as alterações em um projeto são feitas de uma maneira consistente e os interessados são informados do estado do produto, alterações neste, e os impactos no custo e cronograma destas alterações [RAT 2001].

Esta unidade é composta pelas atividades: submeter solicitação de alteração, atualizar solicitação de alteração, executadas por qualquer trabalhador; revisar a solicitação de alteração, e confirmar solicitação de alteração rejeitada ou duplicada, executada pelo gerente de controle de alteração; verificar alterações nas versões, executada pelo integrador do sistema [RAT 2001].

3.4.8 Macro-atividade de Gerenciamento de Projetos

Gerenciamento de Projetos de Software é a arte de balancear objetivos, gerenciamento de riscos, e superação de obstáculos para entregar um produto que encontre as necessidades do cliente e dos usuários. O fato de que poucos projetos são 100% bem-sucedidos é um indicador da dificuldade desta tarefa. O objetivo da macro-atividade de gerenciamento de projetos de software do Processo Unificado Rational é tornar essa tarefa o mais fácil possível, por prover guias nesta área [RAT 2001].

A macro-atividade de gerenciamento de projetos tem as seguintes finalidades [RAT 2001]:

- Prover um *framework* para gerenciamento de projetos de software;
- Prover guias práticas para planejamento, recrutamento, execução e monitoramento de projetos;
- Prover um *framework* para gerenciamento de riscos.

Esta macro-atividade foca nos aspectos específicos de um processo de desenvolvimento iterativo e incremental [RAT 2001]:

- Planejamento de um projeto iterativo, através do ciclo de vida e planejamento de uma iteração em particular;
- Gerenciamento de riscos;
- Monitoramento do progresso de um projeto iterativo e de métricas.

Planejando um Projeto Iterativo

Entre os objetivos do planejamento de projetos estão os seguintes [KRU 2000]:

- Alocar tarefas e responsabilidades para uma equipe de pessoas ao longo do tempo;
- Monitorar o progresso relativo para o plano e detectar potenciais problemas.

Em um processo iterativo, é recomendável que o desenvolvimento seja baseado em dois tipos de planos [KRU 2000]:

Um plano de granularidade grossa, o plano de fase, e um plano de granularidade fina, o plano de iterações.

Plano de Fases

Plano de fases é um plano de granularidade fina, e há somente um por projeto de desenvolvimento. Ele captura o "envelope" completo do projeto por um ciclo (e talvez os ciclos seguintes, se apropriado), e pode ser resumido como segue [KRU 2000]:

- Datas dos marcos maiores:
  - Objetivos do ciclo de vida (final da concepção)
  - Arquitetura do ciclo de vida (final da elaboração)
  - Capacidade Operacional Inicial (final da construção)
  - Versões (*releases*) do Produto (final da transição e do ciclo)
- Perfil equipe: quais recursos são requeridos durante determinado tempo
- Datas dos marcos menores: final de cada iteração e seus objetivos primários, se eles são conhecidos.

O plano da fase é produzido muito cedo na fase de concepção e é atualizado com freqüência, conforme necessário. Ele não requer mais do que uma ou duas páginas. Refere-se ao documento visão para definir o escopo e compreensão do sistema [KRU 2000].

Plano de Iteração

Um plano de iteração é um plano de granularidade fina, e existe um por iteração. Um projeto normalmente tem dois planos de iteração "ativos" em um determinado tempo [KRU 2000]:

- Plano de iteração corrente, que é usado para acompanhar o progresso.
- Plano da próxima iteração, o qual é construído durante a segunda metade da iteração corrente e está pronto no final da iteração corrente.

O plano de iteração é construído usando técnicas e ferramentas de planejamento tradicionais (gráficos de Gantt) para definir as tarefas e suas alocações para indivíduos e equipes.

Conceito de Risco

O processo de desenvolvimento de software se preocupa, principalmente, com os aspectos conhecidos de desenvolvimento de software. Gerenciamento de riscos se preocupa com os aspectos não-conhecidos. Muitas organizações trabalham na forma de negação do risco, estimativas e planejamento procedem como se todas as variáveis fossem conhecidas, e todo o trabalho fosse mecânico e as pessoas fossem substituíveis [KRU 2000, RAT 2001].

Muitas decisões em um ciclo de vida iterativo são dirigidas por riscos. Para tomar decisões efetivas, é necessária uma boa captação dos riscos do projeto e estratégias claras para suavizá-los ou tratar com eles [RAT 2001].

No desenvolvimento de software, risco é uma variável que dentro de sua distribuição normal, pode tomar um valor que ponha em perigo ou elimine o sucesso para o projeto [RAT 2001].

A idéia em gerenciamento de riscos é que não se espere passivamente até que o risco torne-se um problema (ou acabe com o projeto), antes é preciso decidir o que fazer com o ele. Quando um risco é identificado, três rotinas principais são possíveis [RAT 2001]:

- Evitar riscos: reorganizar o projeto de forma a não ser afetado pelo risco.
- Transferir riscos: reorganizar o projeto de forma que alguém ou alguma outra coisa sustente o risco (o cliente, vendedor, banco ou outro elemento).
- Aceitar o risco: decidir sobreviver com o risco como uma contingência. Monitorar o sintoma risco e determinar o que fazer se o risco se materializar.

Se a opção for aceitar o risco, as seguintes ações podem ser tomadas [RAT 2001]:

- Suavizar o risco: executar passos pró-ativos para reduzir a possibilidade ou o impacto do risco.
- Definir um plano de contingência: determinar quais as ações que serão tomadas se o risco se tornar um problema atual.

Conceito de Métricas

Métricas são utilizadas para conquistar o controle do projeto e então gerenciá-lo [RAT 2001]. As medições servem para avaliar o quão longe ou perto o projeto está com relação aos objetivos do plano, em termos de conclusão, qualidade e concordância com os requisitos [RAT 2001]. Medições também são usadas para melhorar estimativas de esforço, custo e qualidade de novos projetos baseados em experiência. Finalmente, as medições servem para avaliar a melhoria em aspectos principais de desempenho do processo durante um tempo, para ver quais são os efeitos das mudanças [RAT 2001].

Para medir é necessário selecionar objetivos precisos para um esforço de medição e coletar somente métricas que permitam satisfazer estes objetivos.

Há dois tipos de objetivos [KRU 2000, RAT 2001]:

- Objetivos de Conhecimento: estes objetivos são expressos pelo uso de verbos, tais como: avaliar, predizer, monitorar. Eles expressam um desejo de entender melhor seu processo de desenvolvimento.
- Objetivos de Alteração ou Realização: estes objetivos são expressos pelo uso de verbos, tais como: aumentar, reduzir, melhorar ou alcançar. Eles expressam um interesse em ver como as coisas mudam ou melhoram durante o tempo, de uma iteração para outra, e de um projeto para outro.

Os objetivos devem ser traduzidos em sub-objetivos (ou objetivos de ação), os quais identificam as ações que os membros do projeto devem tomar para alcançar os objetivos.

Trabalhadores e Artefatos

Os principais trabalhadores da macro-atividade de gerenciamento de projetos são [RAT 2001]:

- **Gerente de projeto:** é responsável por alocar recursos, definir prioridades, coordenar iterações com clientes e usuários, e manter a equipe de projeto focada nos objetivos. O gerente de projeto também estabelece um conjunto de práticas que asseguram a integridade e qualidade dos artefatos do projeto.
- **Revisor de projeto:** é responsável por avaliar os artefatos de planejamento de projetos e artefatos de avaliação de projeto, nos pontos de revisão principal do ciclo de vida do projeto.

Os artefatos principais da macro-atividade de gerenciamento de projetos são os seguintes [RAT 2001]:

- **Plano de desenvolvimento de software (PDS):** este plano é um compreensivo artefato composto, que reúne todas as informações requeridas para gerenciar o projeto. Ele reúne alguns artefatos desenvolvidos durante a fase de concepção e é mantido durante todo o projeto. Artefatos englobados pelo PDS:
  - **Plano de aceitação do produto:** este plano descreve como o cliente avaliará os artefatos entregues do projeto, para determinar se eles encontram um conjunto pré-definido de critérios de aceitação.
  - **Plano de gerenciamento de riscos:** este plano especifica como gerenciar os riscos associados ao projeto.
  - **Lista de riscos:** cita os riscos conhecidos e descobertos para o projeto, ordenados em ordem decrescente de importância, associados com ações de suavização e contingência.
  - **Plano de resolução de problemas:** este plano descreve o processo usado para registrar, analisar e resolver problemas que ocorrem durante o projeto. Em projetos mais complexos, este plano deve criado separado do PDS.
  - **Plano de medição:** define os objetivos das medidas, as métricas associadas e as métricas primitivas, que serão coletadas durante o projeto para monitorar seu progresso.
- **Avaliação da iteração:** captura o resultado de uma iteração, o grau para o qual os critérios de avaliação foram encontrados, lições aprendidas e alterações feitas.
- **Plano de iteração:** um conjunto de atividades e tarefas seqüenciadas, com recursos assinalados, contendo dependência entre tarefas para cada iteração.
- **Caso de negócio:** fornece as informações necessárias do ponto de vista de negócio, para determinar se o projeto merece ou não investimento.
- **Ordem de trabalho:** esta ordem é um meio do gerente de projeto comunicar o que será feito e quando para a equipe responsável. Ela torna-se um contrato interno entre o gerente de projeto e as equipes.
- **Avaliação do status:** um dos objetivos do processo é assegurar que as expectativas de todas as partes são sincronizadas e consistentes. Avaliações periódicas do status fornece um mecanismo para gerenciar expectativas de todos, durante o ciclo de vida.
- **Medidas do projeto:** o artefato de medidas do projeto é um repositório de métricas de dados do projeto. Ele contém as medições que são feitas e que se tornam disponíveis. Ele também armazena as métricas derivadas que são calculadas a partir dos dados primitivos e a forma de calcular (procedimentos e algoritmos) essas métricas derivadas. Os dados coletados durante os projetos são usados na elaboração de relatórios e acompanhamento do projeto.
- **Plano de garantia de qualidade:** provê uma visão clara de como a qualidade do processo, do produto e do artefato será assegurada. Contém o plano de auditoria e revisão e referencia vários outros artefatos desenvolvidos durante a fase de concepção.

- **Registro de revisão:** é criado para registrar o resultado da revisão de um artefato.

Atividades da Macro-atividade de Gerenciamento de Projetos

1. Conceber o novo projeto

A finalidade desta unidade é trazer o projeto de uma idéia inicial para um ponto na qual pode-se fazer uma decisão argumentada para continuar ou abandonar o projeto [RAT 2001].

É composta das seguintes atividades: identificar e avaliar riscos, desenvolver um caso de negócio, iniciar o projeto, executadas pelo gerente de projeto; e revisão da aprovação do projeto, executada pelo revisor de projeto [RAT 2001].

2. Avaliar o escopo e riscos do projeto

A finalidade desta unidade é reavaliar as capacidades e características pretendidas pelo projeto, e os riscos associados ao alcance destas. Esta avaliação é feita somente uma vez depois que o projeto é iniciado, para dar uma base sólida para planejamento detalhado, e então no final de cada iteração, quanto mais é aprendido e riscos são retirados [RAT 2001].

É composta das seguintes atividades: identificar e avaliar riscos e desenvolver caso de negócio, executadas pelo gerente de projeto [RAT 2001].

3. Desenvolver o plano de desenvolvimento do software

A finalidade desta unidade é desenvolver os componentes e inseri-los no PDS e então tê-los formalmente revisado, para viabilidade e aceitabilidade dos interessados e serve como base para um plano de granularidade fina para a próxima iteração (o plano de iteração) [RAT 2001].

É composto pelas seguintes atividades: desenvolver um plano de medição, desenvolver um plano de gerenciamento de riscos, desenvolver um plano de aceitação do produto, desenvolver um plano de resolução de problemas, desenvolver um plano de garantia de qualidade, definir organização e equipe do projeto, definir processos de monitoramento e controle, planejar fases e iterações e juntar o plano de desenvolvimento de software, executadas pelo gerente de projeto; e revisar o planejamento do projeto, executada pelo revisor do planejamento do projeto [RAT 2001].

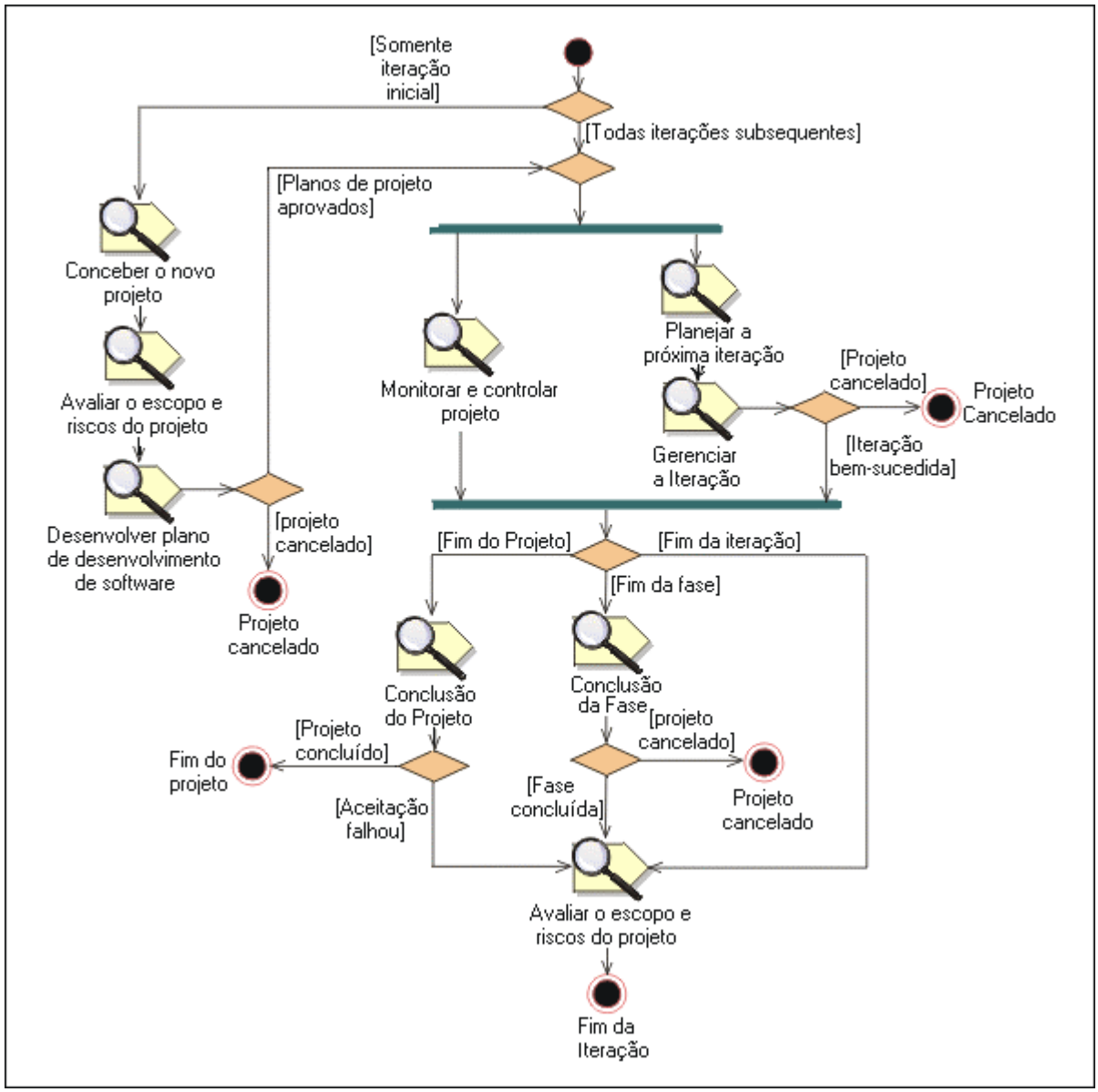


FIGURA 3.10 - Macro-atividade de Gerenciamento de Projetos[Extraído de RAT 2001]

4. Planejar a próxima iteração

A finalidade desta unidade é criar um plano de iteração para guiar a próxima iteração. Depois de criar o plano, ajustes podem ser necessários para o plano de desenvolvimento de software e o caso de negócio [RAT 2001].

As atividades desta unidade são: desenvolver um plano de iteração, desenvolver o caso de negócio, e desenvolver o plano de desenvolvimento de software, executadas pelo gerente de projeto; e revisar o plano de iteração, executada pelo revisor do projeto [RAT 2001].

5. Monitorar e controlar o projeto

Esta unidade captura o dia-a-dia do trabalho do gerente de projeto, cobrindo [RAT 2001]:

- Tratar com alterações nos requisitos que foram sancionadas pelo gerente de controle de mudanças, e cronogramar estas para a iteração corrente ou futura;
- Continuamente monitorar o projeto em termos de riscos ativos e medidas objetivas de progresso e qualidade;
- Regularmente informar o status do projeto, na Avaliação de Status, para a autoridade revisora do projeto, que é uma entidade organizacional para a qual o gerente de projeto deve dar conta;
- Tratar com questões e problemas quando eles forem descobertos e conduzi-los para o fechamento de acordo com o plano de resolução de problemas.

É composta pelas atividades: determinar cronograma e trabalho, monitorar o status do projeto, manipular exceções e problemas, e informar status, executadas pelo gerente de projeto; e revisão do projeto pela autoridade de revisão do projeto (PRA), executada pelo revisor do projeto [RAT 2001].

6. Gerenciar a iteração

Esta macro-atividade contém as atividades que iniciam, terminam e revisam uma iteração.

A finalidade desta unidade é adquirir os recursos necessários para executar a iteração, alocar o trabalho que será executado, e finalmente avaliar os resultados da iteração. Uma iteração conclui com a revisão de aceitação da iteração, que determina se os objetivos da iteração foram encontrados [RAT 2001].

É composta das seguintes atividades: adquirir recursos, iniciar a iteração, e avaliar a iteração, executadas pelo gerente de projeto; revisão dos critérios de avaliação da iteração, e revisão de aceitação da iteração, executadas pelo revisor do projeto [RAT 2001].

7. Conclusão da fase

Nesta unidade o gerente conduz a fase ao fechamento, por assegurar o seguinte [RAT 2001]:

- Todas as questões principais da iteração anterior estão resolvidas;
- O estado de todos artefatos é conhecido;
- Os artefatos requeridos têm sido distribuídos aos interessados;
- Qualquer problema de implantação (instalação, transição, treinamento) tem sido enfocado;
- As finanças do projeto estão resolvidas, se o contrato atual está concluindo.

É composta pelas atividades: preparar para o fechamento da fase, executada pelo gerente de projeto; e revisão do marco do Ciclo de Vida, executada pelo revisor do projeto [RAT 2001].

8. Conclusão do projeto

Nesta unidade, o gerente de projeto prepara o projeto para a conclusão. A avaliação final do status é preparada pela revisão de aceitação do projeto, se bem-sucedida marca o ponto no qual o cliente formalmente aceita a posse do produto de software. O gerente de projeto então completa a conclusão do projeto por dispor dos bens restantes e redirecionar a equipe restante [RAT 2001].

É composta pelas atividades: preparar para a conclusão do projeto, executada pelo gerente de projeto; e revisar a aceitação do projeto, executada pelo revisor do projeto [RAT 2001].

3.4.9 Macro-atividade de Ambiente

A macro-atividade de ambiente foca nas atividades necessárias para configurar o processo para o projeto. Ele descreve as atividades requeridas para desenvolver guias para apoio a um projeto. A finalidade das atividades de ambiente é prover a organização de desenvolvimento de software com o ambiente de desenvolvimento de software - tanto processos e ferramentas - que serve para suportar a equipe de desenvolvimento [RAT 2001].

O suporte inclui [KRU 2000]:

- Seleção e aquisição de ferramentas;
- Instalação e configuração de ferramentas, adaptadas a organização;
- Configuração do processo;
- Melhoria do processo;
- Serviços técnicos para suportar o processo: infra-estrutura de tecnologia de informação, administração de contas, cópias de segurança, etc.

Trabalhadores e Artefatos

O principal trabalhador envolvido na macro-atividade de ambiente é: [KRU 2000, RAT 2001]:

- **Engenheiro de processo:** é responsável pelo processo de desenvolvimento de software. Isto inclui, configurar o processo antes do projeto iniciar e continuamente melhorar o processo, durante os esforços de desenvolvimento.

Em certos aspectos do processo, para estabelecer guias, o engenheiro de processo precisa da competência de outros trabalhadores [KRU 2000]:

- **Analista de processo de negócio:** desenvolve guias para modelagem de negócios.
- **Analista de sistemas:** desenvolve guias para modelagem de casos de uso e um plano para documentação dos requisitos, seus atributos e guias para rastreabilidade.
- **Projetista de interface do usuário:** desenvolve guias para interface do usuário.
- **Arquiteto de software:** desenvolve guias para projetos e programação.
- **Escritor Técnico:** produz o material de suporte ao usuário final, tais como: guia do usuário, textos de ajuda, notas de versões, etc.
- **Projetista de teste:** desenvolve guias para testes.

Os seguintes trabalhadores estão envolvidos com o ambiente de ferramentas [RAT 2001]:

- **Especialista em ferramentas:** selecionar e adquirir ferramentas para suporte ao desenvolvimento. O especialista em ferramentas instala ferramentas e configura as ferramentas para adaptar as necessidades do projeto. Há geralmente vários profissionais agindo como especialistas em ferramentas, cada um responsável por uma ferramentas ou um grupo de ferramentas relacionadas.
- **Administrador do sistema:** mantém o ambiente de desenvolvimento, tanto hardware como software, e executa tarefas de administração do sistema, tais como: administração de contas, cópias de segurança, etc.

O artefato principal da macro-atividade de ambiente é o seguinte [KRU 2000]:

- **Caso de desenvolvimento:** especifica o processo adaptado para um projeto específico. O caso de desenvolvimento descreve, para cada macro-atividade de processo, como o projeto aplicará o processo. Para cada macro-atividade de processo, são decididos os artefatos que serão usados e como serão usados. Um caso de desenvolvimento deve ser breve e referir-se ao processo para detalhes.

Outros artefatos incluídos são os seguintes [RAT 2001, KRU 2000]:

- **Avaliação da organização de desenvolvimento:** descreve o status atual da organização de desenvolvimento em termos de processos, ferramentas, competência de pessoas, atitudes de pessoas, clientes, concorrentes, tendências técnicas, problemas e áreas para melhoria.
- **Modelos específicos do projeto:** modelos (*templates*) para artefatos e relatórios usados no projeto. Pode também ter modelos para diagramas e elementos de diagramas usados no projeto.
- **Manual de guia de estilo:** descreve como o manual de suporte ao usuário final deve ser descrito.

- **Guia de modelagem de casos de uso:** descreve os guias de modelagem dos casos de uso.
- **Guia de modelagem de negócio:** descreve os guias de modelagem de negócio.
- **Guia de interface do usuário:** descreve os guias de como criar interfaces do usuário.
- **Guia de testes:** descreve os guias de teste.
- **Guia de projeto:** descreve os guias de projeto e implementação.
- **Guia de programação:** descreve as convenções que serão usadas quando se trabalhar com uma determinada linguagem de programação.
- **Ferramentas:** recursos de hardware e software que serão usadas para apoiar o desenvolvimento.
- **Guia de ferramentas:** descreve os guias de ferramentas.
- **Infra-estrutura de desenvolvimento:** inclui o hardware e o software, tais como: computadores e sistema operacional, nos quais as ferramentas executam. A infra-estrutura de desenvolvimento também inclui hardware e software necessários para interconectar computadores e usuários.

Atividades da Macro-atividade de Ambiente

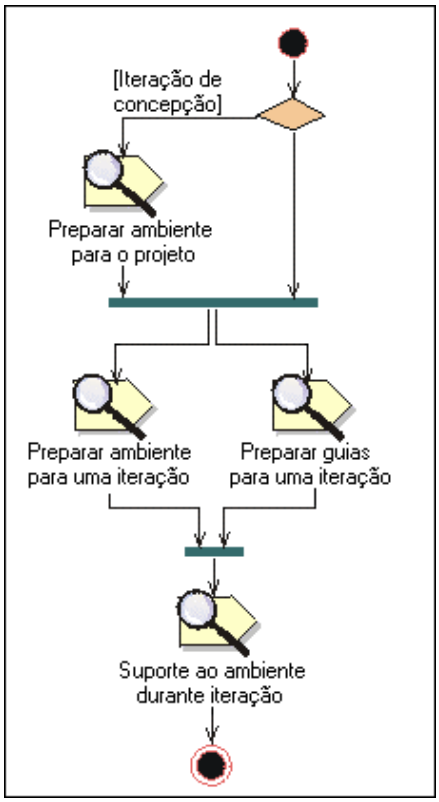


FIGURA 3.11 - Macro-atividade de Ambiente [Extraído de RAT 2001]

1. Preparar ambiente para o projeto

A finalidade de preparar o ambiente para o projeto é [RAT 2001]:

- Avaliar a organização de desenvolvimento atual;
- Avaliar o suporte a ferramenta atual;
- Desenvolver o primeiro esboço do caso de desenvolvimento;
- Produzir uma lista de ferramentas candidatas para uso do desenvolvimento;
- Produzir uma lista de modelos específicos do projeto candidatos para os artefatos principais, de acordo com o caso de desenvolvimento.

É composta das seguintes atividades: avaliar a organização atual, desenvolver caso de desenvolvimento e desenvolver modelos específicos de projeto, executadas pelo engenheiro de processo; e selecionar e adquirir ferramentas, executada pelo especialista em ferramentas [RAT 2001].

2. Preparar o ambiente para uma iteração

A finalidade de preparar o ambiente para uma iteração inclui [RAT 2001]:

- Completar o caso de desenvolvimento para a iteração;
- Preparar, e se necessário, customizar ferramentas para uso dentro de uma iteração;
- Verificar se as ferramentas têm sido configuradas e instaladas corretamente;
- Produzir um conjunto de modelos específicos para o projeto, que serão usados dentro da iteração.

É composta das seguintes atividades: desenvolver caso de desenvolvimento, desenvolver modelos específicos de projeto e divulgar o caso de desenvolvimento, executadas pelo engenheiro de processo; instalar ferramentas e verificar instalação e configuração das ferramentas, executadas pelo especialista em ferramentas [RAT 2001].

3. Preparar guias para uma iteração

A finalidade desta unidade é preparar o ambiente para uma iteração é preparar ou atualizar guias para a iteração [RAT 2001].

É composta das seguintes atividades: desenvolver guias de modelagem de negócio, executada pelo analista de processo de negócio; desenvolver guias de modelagem de caso de uso, executada pelo analista de sistema; desenvolver guias de interface do usuário, executada pelo projetista de interface do usuário; desenvolver guias de ferramentas, executada especialista em ferramentas; desenvolver guias de projeto e desenvolver guias de programação, executadas pelo arquiteto de software; desenvolver manual de guia de estilo, executada pelo escritor técnico; e desenvolver guias de testes, executada pelo projetista de teste [RAT 2001].

4. Suporte ao ambiente durante a iteração

A finalidade desta unidade é suportar os desenvolvedores no uso das ferramentas e do processo durante a iteração [RAT 2001].

É composto da seguinte atividade: suporte ao desenvolvimento, executada pelo administrador de sistemas [RAT 2001].

4 Avaliação de Processos de Software

A qualidade de software é largamente determinada pela qualidade dos processos utilizados para seu desenvolvimento. Desde modo, a melhoria da qualidade de software é obtida pela melhoria da qualidade dos processos. Essa visão orientou a elaboração de modelos de definição, avaliação e melhoria de processos de software, tais como:

Bootstrap: esquema derivado do *Capability Maturity Model* (CMM), para avaliação de organizações, desenvolvido pela Comunidade Européia [HAA 94].

Trillium: esquema derivado do CMM e outros modelos, para avaliação de organizações, desenvolvido no Canadá [BEL 94].

***Software Quality In the Development* (SQUID):** é um consórcio de empresas e institutos da Europa. Está baseado nos conceitos e definições das normas ISO 9001, ISO/IEC 9126 e ISO/IEC 14598-3.

ISO/IEC 12207 - Processos de Ciclo de Vida de Software: norma de definição dos processos do ciclo de vida do software [TSU 96].

A seguir serão descritas as normas ISO 9000-3 e ISO/IEC 15504 e o *Capability Maturity Model*, que também propõem a melhoria dos processos de desenvolvimento.

4.1 ISO 9000-3

A ISO 9000-3 é um guia para a aplicação da ISO 9001 para o desenvolvimento, fornecimento e manutenção de software [ISO 91]. Para cada item da ISO 9001 existe um correspondente na ISO9000-3 que o detalha e o especifica ao software.

As diretrizes propostas na ISO 9000-3 cobrem questões como o entendimento comum entre as partes (contratante e contratado) de requisitos funcionais e uso de metodologias consistentes para o desenvolvimento de software e gerenciamento de projeto como um todo, da concepção até a manutenção [ISO 91].

Essa norma é dividida em três partes principais, que são: estrutura, descreve aspectos organizacionais, relacionados ao sistema de qualidade; atividades do ciclo de vida que descrevem as atividades de desenvolvimento de software; e atividades de suporte que descrevem as atividades que apóiam as atividades do ciclo de vida de desenvolvimento [ISO 91].

4.2 ISO/IEC 15504 - *Software Process Improvement and Capability Determination* (SPICE)

Em junho de 1991, o comitê de engenharia de software da ISO, aprovou a realização de um período de estudos para analisar as necessidades e os requisitos de um padrão para avaliação do processo de software. Este estudo apontou um consenso internacional sobre a necessidade de um padrão para avaliação do processo de software e a importância de se implementar melhorias no processo de desenvolvimento.

Em 1993, foi criado o projeto SPICE (*Software Process Improvement and Capability Determination*), com o objetivo de gerar normas que orientem a avaliação do processo de software, visando a melhoria contínua do processo e a determinação de sua capacitação [ISO 95a].

O SPICE estabelece um modelo de referência, que serve de base para o processo de avaliação. Este modelo é um conjunto universal de processos que são fundamentais para engenharia de software, e um roteiro racional para a melhoria de cada processo (capacitação de processos).

Este modelo é dividido em cinco grandes categorias de processo: Cliente-Fornecedor, Engenharia, Projeto, Suporte e Organização.

Na categoria Cliente-Fornecedor são descritos os processos que impactam diretamente o cliente, suporte e a transição do software para o cliente e provêem a operação e uso correto do software/sistema [ISO 95a]. Em Engenharia, são descritos os processos que especificam, implementam ou mantém um sistema ou produto de software e sua documentação [ISO 95a]. A categoria Projeto consiste dos processos que estabelecem o projeto, coordenam e gerenciam seus recursos para produzir um produto ou prover serviços que satisfaçam o cliente [ISO 95a]. Em Suporte, são definidos os processos que podem ser empregados por qualquer outro processo [ISO 95a]. Em Organização, são descritos os processos que estabelecem os objetivos de negócio da organização [ISO 95a].

O SPICE define seis níveis de capacitação, que são:

Processo Incompleto (nível 0): o processo não está implementado e falha na tentativa de atingir seus objetivos.

Processo Executado (nível 1): o processo implementado atinge seu objetivo definido.

Processo Gerenciado (nível 2): o processo executado entrega produtos de trabalho de qualidade definida, obedecendo a cronogramas e recursos definidos.

Processo Estabelecido (nível 3): o processo gerenciado é executado usando um processo definido baseado em bons princípios de engenharia de software.

Processo Previsível (nível 4): o processo estabelecido é executado consistentemente dentro de limites definidos de controle para atingir seus objetivos.

Processo Otimizado (nível 5): o processo previsível otimiza o seu desempenho para atender as necessidades de negócio atuais e futuras e atinge repetibilidade em atender seus objetivos definidos de negócios.

4.3 Capability Maturity Model

O modelo *Capability Maturity Model for Software* (CMM ou SW-CMM) foi desenvolvido em 1991, com o objetivo de avaliar a capacidade e a maturidade de uma organização. O CMM foi desenvolvido pelo *Software Engineering Institute* (SEI), da *Carnegie Mellon University*, e foi apoiado pelo *Department of Defense* (DoD) dos EUA, que é um grande consumidor de software e precisava de um modelo formal que permitisse selecionar seus fornecedores de software de maneira adequada.

O CMM é um *framework* que se caracteriza por uma melhoria de processo, voltado a organizações mais maduras. Uma organização pode usar o CMM para determinar sua classificação de maturidade do processo de software e então estabelecer prioridades para melhoria [PAU 93].

No CMM, os processos de software são organizados em cinco níveis de maturidade. Essa estrutura em níveis do CMM está baseada nos princípios de qualidade propostos por Walter Shewart, W. Edwards Deming, Joseph Juran e Philip Crosby [PAU 93a].

Com exceção do Nível 1, cada nível de maturidade é composto de várias áreas-chave de processo, que priorizam ações de melhoria para aumentar a maturidade do processo de software. A Figura 4.1 mostra os cinco níveis de maturidade do CMM e suas áreas-chave de processo, o rótulo da seta indica o tipo de capacidade de processo que está sendo institucionalizada, em cada passo, no *framework* de maturidade.

O CMM foi o modelo de avaliação escolhido porque é um dos mais bem conhecidos modelos de avaliação da maturidade de processos de software [FIT 99], apresenta uma estrutura detalhada de avaliação de software, enfatiza fortemente os aspectos de melhoria contínua [FIT 99] e sua usabilidade prática tem sido demonstrada por vários casos de avaliação [HER 94].

Outro motivo que influenciou na escolha do CMM, foi o fato do Serviço Federal de Processamento de Dados (SERPRO) ter adotado o CMM como modelo de avaliação para melhorar seus processos de software. Esta iniciativa faz parte de um projeto corporativo, chamado de Projeto de Modernização do Desenvolvimento (PMoD), que tem como meta atingir o nível 2 de CMM até dezembro de 2002. Para atingir essa meta o SERPRO está investindo no Processo Unificado Rational e nas ferramentas que apóiam este processo.

Os cinco níveis de maturidade podem ser caracterizados quanto às mudanças nos processos primários principais, realizadas em cada nível [PAU 93]:

1) Inicial	O processo de software é caracterizado como *ad hoc* e ocasionalmente até caótico. Poucos processos são definidos e o sucesso depende de esforços individuais e heróicos.
2) Repetível	Os processos básicos de gerenciamento de projeto estão estabelecidos e permitem acompanhar custo, cronograma e funcionalidade. É possível repetir o sucesso de um processo utilizado anteriormente, em outros projetos similares.
3) Definido	As atividades de gerenciamento e de engenharia do processo de software estão documentadas, padronizadas, e integradas em um processo de software padronizado da organização. Todos os projetos seguem este processo, para desenvolver e manter o software.
4) Gerenciado	Medidas detalhadas da qualidade dos produtos e processos de software são coletadas. Tanto o processo de software, quanto o produto, são entendidos e controlados quantitativamente.

5) Otimizado — A melhoria contínua do processo é possibilitada pela realimentação quantitativa do processo, e conduzida a partir de idéias e tecnologias inovadoras.

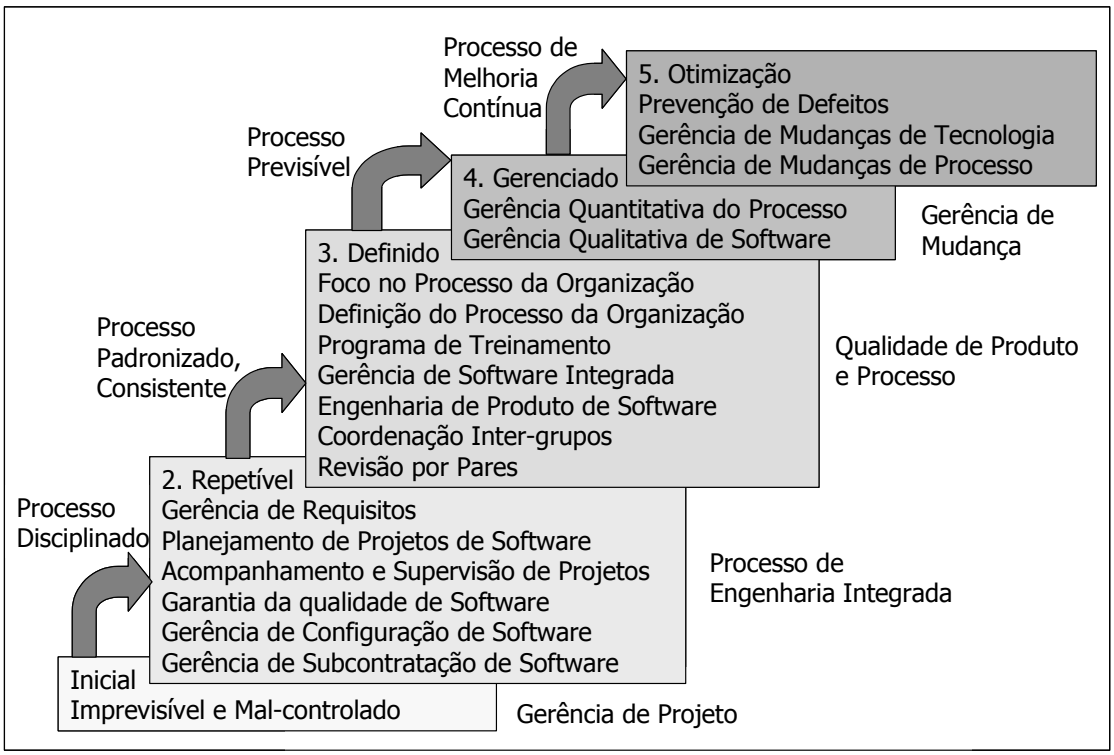


FIGURA 4.1 - Níveis de Maturidade do CMM [Extraído de PAU 93]

4.3.1 Estrutura Interna dos Níveis de Maturidade

O CMM é composto de cinco níveis de maturidade. Com exceção do Nível 1, cada nível de maturidade é composto de várias áreas-chave de processo. Cada área-chave de processo é organizada em cinco seções, chamadas de características-comum. As características-comum especificam as práticas-chave que, quando coletivamente enfocadas, acompanham as metas da área-chave de processo [PAU 93]. Esta estrutura está ilustrada na Figura 4.2. Os componentes do CMM incluem:

Níveis de Maturidade: é um platô evolucionário bem definido para que se possa alcançar um processo maduro [PAU 93]. Os níveis são uma forma de priorizar as ações de melhoria de tal forma que se aumente a maturidade do processo de software. Por exemplo, no nível 2 são tratados aspectos gerenciais dos projetos, e no nível 3 são tratados aspectos técnicos de desenvolvimento.

Capacidade de Processo: descreve o conjunto de resultados esperados, e que podem ser alcançados, por seguir o processo de software. A capacidade de processo de

software de uma organização, provê os meios de predizer os resultados esperados para o próximo projeto da organização [PAU 93].

Áreas-chave de Processo: cada área-chave identifica um conjunto de atividades relacionadas que, quando executadas coletivamente, alcançam as metas consideradas importantes para estabelecer a maturidade de processo de determinado nível de maturidade. Cada nível de maturidade define seu conjunto de áreas-chave de processo.

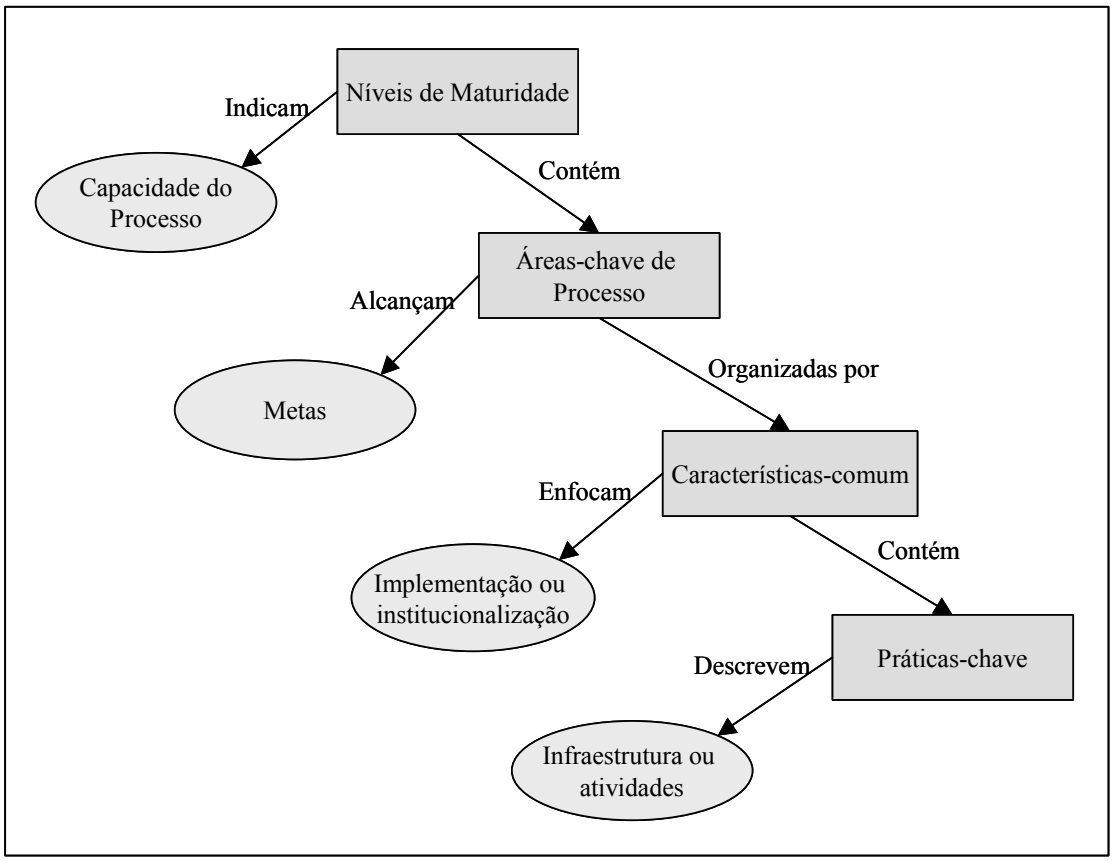


FIGURA 4.2 - Estrutura do CMM [Extraído de PAU 93]

Metas: resumem as práticas-chave de uma área-chave de processo, e podem ser usadas para determinar se uma organização ou projeto implementou efetivamente uma área-chave de processo.

Práticas-chave: cada área-chave de processo é descrita em termos de práticas-chave que, quando implementadas, ajudam a satisfazer as metas de cada área-chave de processo. As práticas-chave descrevem a infra-estrutura e atividades que contribuem para a efetiva implementação e institucionalização da área-chave de processo. As práticas-chave são descritas independentes de qualquer implementação. No CMM completo são descritas 316 práticas-chave.

Características-comum: são atributos que indicam se a implementação ou institucionalização de uma área-chave de processo é efetiva, repetível e duradoura. Representam uma forma de organizar as práticas-chave.

As cinco características-comum são descritas brevemente a seguir:

1) Compromisso de Executar: descreve as ações que a organização deve adotar para garantir que o processo está estabelecido e vai perdurar. Normalmente envolve políticas organizacionais e envolvimento direto da gerência sênior.

2) Capacidade para Executar: descreve as pré-condições que devem existir no projeto ou na organização para implementar o processo de software de forma competente. Normalmente envolve recursos, estruturas organizacionais, treinamento e delegação de responsabilidades.

3) Atividades Executadas: descreve as atividades, papéis e procedimentos necessários para implementar a área-chave de processo. Normalmente envolve estabelecer planos e procedimentos, executar o trabalho, acompanhá-lo, e tomar ações corretivas quando necessário.

4) Medição e análise: descreve as práticas básicas de medição que são necessárias para determinar o status relativo ao processo. Essas medições são usadas para controlar e melhorar o processo. Normalmente, inclui exemplos de medidas que podem ser obtidas.

5) Verificação da implementação: descreve as etapas para assegurar que as atividades são executadas de acordo com o processo estabelecido. Normalmente abrange revisões e auditorias pela gerência e garantia da qualidade de software.

As áreas-chave de processo dos níveis 2 e 3 serão exploradas em maiores detalhes no próximo capítulo. O próximo capítulo descreve como foi realizada a avaliação do Processo Unificado Rational com relação às práticas-chave do *Capability Maturity Model* e a análise dos resultados obtidos.

5 Avaliando RUP Baseado no CMM

As organizações que desenvolvem software estão buscando definir seus modelos de processo, avaliá-los e melhorá-los. Esta realidade pode ser verificada pela quantidade de modelos de avaliação disponíveis [FIT 99, KRI 99], ilustrada pelos modelos CMM [PAU 93, PAU 93a, KRI 99], Bootstrap [HAA 94], Trillium [BEL 94] e SQUID [BOE 99]; e pelos esforços de padronização tais como as normas ISO 9000-3 [ISO 91], ISO 12207[ISO 95] e ISO 15504 (ou SPICE) [ISO 95a].

Por outro lado, não é muito comum que proponentes de processos de software sejam avaliados usando modelos de avaliação de processos estabelecidos. A existência de um processo ou *framework* de processo avaliado, usando um modelo ou norma, pode ajudar as organizações na seleção de processos de software de acordo com critérios de avaliação estabelecidos, prevenindo frustrações futuras quando da avaliação de seus processos.

As práticas-chave do CMM são expressas em termos de práticas esperadas em organizações que trabalham com grandes contratos governamentais. Em qualquer contexto no qual o CMM é aplicado, uma interpretação de como as práticas devem ser aplicadas deve ser usada. Guias para interpretação do CMM são disponibilizadas pelo SEI [PAU 93a, PAU 99].

O CMM-SW especifica o "que deve" ser feito no processo de software, mas não especifica "como". O Processo Unificado Rational descreve um modelo de processo de desenvolvimento de software, especificando como desenvolver software, isto é, as atividades que devem ser seguidas, os artefatos que devem ser gerados, os trabalhadores responsáveis e o relacionamento entre as atividades.

O objetivo deste trabalho é avaliar o RUP, um modelo de processo existente, em relação a um modelo de avaliação já consagrado, o CMM, determinando o suporte fornecido por este processo a organizações que objetivam alcançar níveis 2 e 3 do CMM, e as lacunas apresentadas pelo RUP com relação ao CMM, para que as organizações desenvolvam práticas próprias para enfocar estas lacunas, ou utilizem as sugestões propostas neste trabalho.

O resultado desta avaliação pode ajudar as organizações que desejam alcançar níveis 2 e 3 de CMM e estejam utilizando, ou venham a utilizar o RUP, pois a avaliação identifica as práticas suportadas pelo RUP, e as práticas que a organização deve definir, porque estão fora do escopo do RUP.

A avaliação também pode auxiliar organizações que estejam utilizando outros processos de software, mas que desejam alcançar níveis 2 e 3 de CMM, e para atingir este objetivo precisam complementar seus processos. Essas organizações podem verificar nesta dissertação, o subconjunto do RUP que está apoiando os processos a serem complementados, e implementar apenas este subconjunto do RUP.

A avaliação pode auxiliar as organizações no momento de adaptar o processo unificado para o processo padrão da organização ou para projetos específicos, por identificar os elementos do RUP que estão apoiando as práticas-chave do CMM, e que seria aconselhável serem mantidos no processo customizado, não eliminando-os na adaptação.

A avaliação foi baseada nas práticas-chave descritas pelo Relatório Técnico CMU/SEI-93-TR-025, *Key Practices of the Capability Maturity Model, Version 1.1* [PAU 93], e no Processo Unificado Rational, Versão 2001.03.00, distribuído pelo Rational Corporation [RAT 2001].

A avaliação descrita neste trabalho foi baseada nos guias de como interpretar o CMM, identificando no RUP, os elementos descritos pelo CMM, independente dos nomes utilizados, desde que o RUP cumpra os objetivos propostos pelo CMM. Por exemplo, a característica-comum do CMM *Verificação de Implementação,* contém práticas que relatam a revisão de atividades pelo gerenciamento sênior, no RUP esta atividade é executada pelo *Revisor do Projeto*. O objetivo é cumprido, independente de serem utilizados nomes e papéis diferentes para as atividades.

Nesta avaliação, todas as práticas-chave dos níveis 2 e 3, são consideradas aplicáveis. Para algumas organizações ou projetos, algumas práticas podem ser consideradas não aplicáveis, sendo desconsideradas na avaliação. Por exemplo, todas as práticas-chave descritas em *Gerenciamento de Subcontratação* podem ser ignoradas se a organização não pratica subcontratação.O processo de avaliação é brevemente descrito na próxima seção.

5.1 Processo de Avaliação

A avaliação do processo seguiu uma rotina precisa. Para cada prática-chave identificada no Relatório do CMM, o Processo Unificado Rational foi avaliado para determinar se ele satisfaz ou não a prática.

A prática-chave é considerada satisfeita se, existe no RUP, alguma atividade, artefato, trabalhador ou macro-atividade, ou um conjunto desses elementos, que implementam a prática.

Por exemplo, o CMM descreve a seguinte prática-chave: “O grupo de engenharia de software[1] usa os requisitos alocados[2] como base para os planos de software, produtos de trabalho e atividades”.

A justificativa para esta prática-chave é a seguinte:

“Na macro-atividade de Requisitos do RUP é descrita uma abordagem sistemática para encontrar, documentar, organizar e acompanhar alterações dos requisitos do sistema. Os requisitos do sistema são documentados por meio de Casos de Uso, e os Casos de Uso são usados como base para o planejamento iterativo e controlado”.

Nesta avaliação, uma prática é considerada satisfeita quando uma parte substancial (> 75%) das sub-práticas associadas a práticas está estabelecida pelo RUP. Sub-práticas, também conhecidas como práticas-chave subordinadas, são listadas abaixo das práticas-

[1] O grupo de engenharia de software é composto pelo pessoal técnico de desenvolvimento de software, tais como: programadores, analistas, arquitetos, etc.

[2] Requisitos alocados são um subconjunto dos requisitos do sistema que serão implementados nos componentes de software do sistema.

chave e descrevem o que é esperado encontrar implementado para a prática-chave [PAU 93].

5.2 Áreas-Chave do CMM - Nível 2

5.2.1 Gerenciamento de Requisitos

A finalidade do Gerenciamento de Requisitos é estabelecer um acordo formal entre o cliente e a equipe de projeto dos requisitos do cliente que devem ser enfocados pelo projeto de software [PAU 93]. Gerenciamento de requisitos envolve estabelecer e manter um acordo com o cliente dos requisitos do projeto de software. Este acordo cobre tanto os requisitos técnicos, quanto os não-técnicos (prazos de entrega,...). Este acordo forma as bases para as atividades de estimativa, planejamento, execução e acompanhamento do projeto de software, durante todo o ciclo de vida. Sempre que os requisitos alocados forem alterados, os planos de software, os produtos de trabalho e as atividades afetadas devem ser ajustados para permanecer consistentes com os requisitos alterados.

O CMM é voltado, primariamente, para o desenvolvimento de grandes sistemas, onde o software é apenas uma parte do sistema e os requisitos provenientes dos clientes e usuários finais podem ser alocados a projetos que envolvam tanto a produção de hardware quanto de software. Cada requisito do sistema deve ser alocado aos componentes do sistema (hardware, software ou ambos). Esta tarefa é chamada de alocação de requisitos. O CMM chama os requisitos de sistema alocados para o software de requisitos alocados.

Requisitos de sistema: Definem o comportamento do sistema do ponto de vista do negócio que apóiam, do usuário ou do cliente. Geralmente descrevem o ambiente e a operação do sistema a ser construído.

Requisitos de sistema alocados para o software (Requisitos alocados): são um subconjunto dos requisitos do sistema e serão implementados nos componentes de software do sistema. Nas etapas iniciais do processo de desenvolvimento, os requisitos alocados são elaborados e refinados, resultando no documento Especificação de Requisitos de Software.

Meta (1/2)

Requisitos alocados para o software são controlados para estabelecer uma configuração-base[3] para uso gerencial e da engenharia de software [PAU 93].

Meta (2/2)

Planos de software, atividades e produtos são mantidos consistentes com os requisitos de sistema alocados para o software [PAU 93].

Características-comum

Compromisso de Executar

---

[3] Configuração-base (*Baseline*): conjunto de produtos de trabalho aceitos e controlados e que serão utilizados em atividades posteriores à sua aceitação.

(1/1) O projeto segue uma política organizacional[4] para gerenciar os requisitos de sistema alocados para o software.

O *Processo Unificado Rational* descreve na macro-atividade de *Requisitos*, as atividades que devem ser executadas para gerenciar os requisitos [RAT 98]. A política específica para cada projeto deve ser descrita no *Caso de Desenvolvimento* [RAT 2001].

Habilidade de Executar

(1/4) Para cada projeto, responsabilidades são estabelecidas para analisar os requisitos do sistema e alocá-los ao hardware, software e outros componentes do sistema.

Cada atividade, descrita na macro-atividade de *Requisitos*, está associada ao trabalhador responsável por executa-la, guias de auxílio e os artefatos que devem ser gerados [RAT 2001].

(2/4) Os requisitos do sistema estão documentados.

Os requisitos do sistema são documentados no artefato *Especificação dos Requisitos do Software*, que é composto pelos *Casos de Uso* do *Modelo de Casos de Uso* e pela *Especificação Complementar* [RAT 2001].

Os requisitos não-técnicos são documentados nos artefatos: *Solicitação dos Interessados* e *Visão*.

Os critérios de aceitação do produto são descritos no *Plano de Aceitação do Produto*.

(3/4) Estão disponíveis os recursos e fundos necessários para gerenciar os requisitos alocados.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP para satisfazer essa prática-chave.

(4/4) Membros do grupo de engenharia de software e outros grupos relacionados ao software são treinados para executar suas atividades de gerenciamento de requisitos.

No Processo Unificado Rational é descrito que atividades relacionadas a gerenciamento de pessoas não são cobertos pelo processo.

No capítulo 6, é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/3) O grupo de engenharia de software revisa os requisitos alocados antes destes serem incorporados ao projeto de software.

RUP descreve a atividade *Revisar Requisitos*, realizada pelo *Revisor de Requisitos*, com a participação do *Especificador de Requisitos* e do *Analista de Sistemas*. Essa atividade tem como objetivo verificar, formalmente, se os requisitos estão conforme a visão do cliente do sistema. RUP descreve pontos de verificação associados a esta atividade.

---

[4] Políticas organizacionais se referem a políticas documentadas, geralmente por escrito, que devem ser seguidas pelo projeto ou pela organização para a execução das práticas de uma determinada área-chave.

(2/3) O grupo de engenharia de software usa os requisitos alocados como base para o desenvolvimento do software.

RUP descreve na macro-atividade de *Requisitos* uma abordagem sistemática para encontrar, documentar, organizar e acompanhar alterações dos requisitos do sistema. Os requisitos do sistema são documentados por meio de *Casos de Uso*, e os *Casos de Uso* são usados como base para o planejamento iterativo e controlado.

(3/3) Mudanças nos requisitos alocados são revisadas e incorporadas ao projeto de software.

O *Processo Unificado* propõe a criação de um *Conselho de Controle de Mudanças* para identificar o escopo e os impactos (orçamentários, técnicos e cronograma) de uma mudança proposta ou de defeitos descobertos durante o desenvolvimento. As alterações são revisadas por este conselho, todos os interessados são comunicados e os planos são alterados para refletir essas modificações.

Medição e Análise

(1/1) Medições são feitas e usadas para determinar o status das atividades de gerenciamento de requisitos.

O RUP propõe a elaboração de um *Plano de Medição*. Neste plano, devem ser definidos os objetivos da medida, as métricas associadas e as métricas primitivas, que serão coletadas no projeto para monitorar o seu progresso. O RUP propõe várias métricas associadas aos requisitos como:

Estabilidade: taxa de alteração nos requisitos ou implementação, tamanho ou complexidade.

Completeza: uma medida de extensão para a qual um artefato encontra todos os requisitos (declarados ou implícitos). O gerente deve tentar fazer um uma medida o mais explícita possível, para limitar os riscos de expectativas não realizadas.

Rastreabilidade: um indicador de que os requisitos em um nível estão sendo satisfeitos por artefatos em mais baixo nível, e olhando de outra maneira, indica que um artefato em qualquer nível tem uma razão de existir.

Volatilidade: o nível de mudança ou alteração em um requisito por causa de defeitos ou mudança nos requisitos.

O artefato *Atributo dos Requisitos* é um repositório dos requisitos do projeto, atributos e dependências para acompanhar os requisitos da perspectiva de gerenciamento.

Verificação da Implementação

(1/3) As atividades para gerenciar os requisitos alocados são revisadas periodicamente junto à gerência sênior.

As atividades para gerenciamento dos requisitos são revisadas pelo *Revisor de Projeto*, em pontos de revisão durante o ciclo de vida do projeto.

O *Revisor de Projeto* revisa o *Plano de Iterações* e o *Plano de Desenvolvimento de Software*, onde são descritas as atividades de Gerenciamento de Requisitos.

(2/3) As atividades para gerenciamento dos requisitos alocados são revisadas periodicamente e em determinados eventos com o gerente de projeto.

O *Gerente do Projeto* faz avaliações periódicas do status do projeto e estas são documentadas na *Avaliação de Status*.

(3/3) O grupo de qualidade de software revisa e/ou audita as atividades e os produtos de trabalho para gerenciar os requisitos alocados e informa os resultados.

RUP descreve a elaboração de um *Plano de Garantia de Qualidade*, pelo *Gerente do Projeto*. Neste plano devem ser descritas todas as informações necessárias para realizar as atividades de garantia de qualidade para o projeto. O plano é usado para planejar as revisões e as auditorias que verificarão se processo definido para o projeto está sendo seguido corretamente.

O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* (APES) deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas no *Plano de Garantia de Qualidade*.

Resumo da Avaliação

A macro-atividade de *Requisitos* do RUP, descreve a seqüência de atividades que devem ser executadas para gerenciar os requisitos do sistema, e como devem ser realizadas alterações nestes requisitos.

Os requisitos do sistema são documentados na *Especificação dos Requisitos do Software*, e os requisitos não-técnicos são documentados na *Solicitação dos Interessados* e na *Visão*. RUP descreve no *Plano de Medição*, as medidas relacionadas a requisitos.

O *Conselho de Controle de Alteração* é responsável por aprovar e revisar as alterações propostas nos requisitos. *Revisor de Requisitos* revisa os requisitos do software, e as verificações de implementação são realizadas pelo *Revisor de Projeto*, *Gerente de Projeto* e APES.

5.2.2 Planejamento de Projetos de Software

A finalidade do Planejamento de Projetos de Software é estabelecer planos razoáveis para a execução das tarefas de engenharia de software e para gerenciar o projeto de software.

O Planejamento do Projeto de Software envolve definir o plano de realização do trabalho e realizar estimativas de software, estabelecendo os compromissos com as partes envolvidas.

Este plano provê as bases para executar e gerenciar as atividades do projeto de software e enfocar os compromissos com o cliente do projeto de software, conforme recursos, limitações e capacidades do projeto de software [PAU 93].

Meta 1/3

As estimativas do software são documentadas para uso no planejamento e acompanhamento do projeto de software.

Meta 2/3

As atividades e compromissos do projeto de software são planejados e documentados.

Meta 3/3

Grupos e pessoas envolvidas concordam com seus compromissos relacionados ao projeto de software.

Características-comum

Compromisso de Executar

(1/2) Um gerente de projeto de software é responsável por negociar compromissos e por desenvolver o plano de desenvolvimento do projeto de software.

RUP descreve o papel *Gerente de Projeto*, o que é responsável por negociar compromissos e desenvolver o *Plano de Desenvolvimento de Software* (PDS), dentre outras atividades.

(2/2) O projeto segue uma política organizacional para planejar um projeto de software.

RUP descreve na macro-atividade de *Gerenciamento de Projetos*, as atividades que devem ser executadas para planejar o projeto, os artefatos que devem ser gerados e os cargos associados a cada atividade.

O *Caso de Desenvolvimento*, elaborado pelo *Engenheiro de Processo*, no início do projeto, adapta as atividades descritas pelo RUP para um projeto específico.

Habilidade para Executar

(1/4) Uma declaração aprovada e documentada de trabalho existe para o projeto de software.

O artefato *Visão*, descreve a visão geral dos requisitos do projeto, e provê a base contratual para os requisitos técnicos, e o *Caso de Negócio* descreve uma análise econômica do projeto. Estes artefatos são revisados pelo *Revisor do Projeto*, e são gerenciados e controlados.

(2/4) Responsabilidades para desenvolver o plano de desenvolvimento de software estão determinadas.

O RUP descreve os cargos associados a cada atividade e o *Plano de Iteração* assinala a responsabilidade a pessoa que ocupa o cargo. A elaboração do *Plano de Desenvolvimento de Software*, segundo o RUP, é de responsabilidade do *Gerente de Projeto*.

(3/4) Estão disponíveis os recursos e fundos necessários para planejar o projeto de software.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(4/4) Os gerentes de software, engenheiros de software, e outros indivíduos envolvidos no planejamento são treinados em procedimentos de planejamento e estimativas aplicáveis em suas áreas de responsabilidades.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/15) O grupo de engenharia de software participa da equipe que propõe o projeto.

O grupo de engenharia de software é composto por trabalhadores com diferentes perfis que participam desde a proposta do projeto até a implantação .

(2/15) O planejamento do projeto de software é iniciado nos estágios iniciais, e em paralelo com o planejamento completo do projeto.

Não se aplica, o escopo do *Processo Unificado Rational* limita-se a gerenciamento de projetos de software.

(3/15) O grupo de engenharia de software participa, com outros grupos afetados, no planejamento global do projeto ao longo do ciclo de vida do projeto.

Não se aplica, o escopo do *Processo Unificado Rational* limita-se a gerenciamento de projetos de software.

(4/15) Os compromissos do projeto de software, feitos a indivíduos e grupos externos à organização, são revisados com o gerenciamento sênior de acordo com o procedimento documentado.

Os compromissos do projeto, tais como: orçamento, cronograma e recursos, são documentados no *Plano de Desenvolvimento de Software*, que é revisado pelo *Revisor do Projeto*, conforme procedimento descrito na atividade *Revisar Planejamento do Projeto*.

(5/15) O ciclo de vida do software com estágios pré-definidos e de tamanho gerenciável é identificado e definido.

O RUP propõe um ciclo de vida iterativo e incremental. O projeto é dividido em iterações sucessivas para facilitar seu gerenciamento, diminuir os riscos e acelerar seu desenvolvimento. O *Caso de Desenvolvimento* descreve o ciclo de vida específico para o projeto, considerando o número de iterações do projeto e a duração de cada iteração e fase.

(6/15) O plano de desenvolvimento de software do projeto é desenvolvido conforme um procedimento documentado.

O *Plano de Desenvolvimento de Software* é desenvolvido de acordo com os procedimentos descritos na atividade Desenvolver Plano de Desenvolvimento de Software, da macro-atividade de Gerenciamento de Projetos.

(7/15) O plano para o projeto de software é documentado.

O plano de projeto de software é documentado no artefato *Plano de Desenvolvimento de Software*. Os procedimentos, métodos e padrões de desenvolvimento são documentados e mantidos no *Caso de Desenvolvimento*.

(8/15) Produtos de trabalho de software que são necessários para estabelecer e manter o controle do projeto de software são identificados.

O RUP identifica artefatos que serão desenvolvidos para controlar o projeto no *Caso de Desenvolvimento*.

(9/15) Estimativas de tamanho dos produtos de trabalho de software (ou mudanças no tamanho dos produtos de software) são derivados conforme um procedimento documentado.

Segundo o RUP, o tamanho dos produtos e alterações pode ser estimado por analogia ou análise. Na estimativa de tamanho por analogia, o novo produto que será desenvolvido é comparado com produtos desenvolvidos anteriormente. Várias características do produto podem ser comparadas como: número de casos de uso, número de atores, tamanho/complexidade da base de dados, número de programas *online* e *batch* [RAT 2001].

Quando se têm mais informações sobre o produto, é possível utilizar técnicas analíticas para estimar o tamanho do produto. Estas técnicas se baseiam na descrição funcional do produto de software e na aplicação de regras de contagem padrão, para determinar medidas de tamanho para estas descrições. Como exemplo, pode-se citar pontos por função, pontos por casos de uso, etc [RAT 2001].

Os procedimentos para estimar tamanho são descritos na atividade *Planejar Fases e Iterações*, da macro-atividade de *Gerenciamento de Projetos*. As estimativas devem ser documentadas no *Plano de Desenvolvimento de Software*.

(10/15) Estimativas de esforço e custos do projeto são derivadas conforme um procedimento documentado.

O RUP sugere o uso de modelos científicos já estabelecidos para estimar esforço e custos de um projeto, tais como: o *Constructive Cost Model* (COCOMO), desenvolvido por Barry Boehm, e da *Putnam Methodology* de Larry Putnam [RAT 2001].

Os dados históricos dos projetos são armazenados no artefato *Medidas dos Projetos*.

(11/15) Estimativas de recursos computacionais críticos para o projeto são derivados conforme um procedimento documentado.

RUP não descreve procedimentos para estimar recursos computacionais críticos. A proposta 2, descrita no capítulo 6 deste trabalho, propõe como essa atividade pode ser realizada.

(12/15) O cronograma de software do projeto é derivado de acordo com um procedimento documentado.

O cronograma de software é um dos elementos constituintes do *Plano de Desenvolvimento de Software*, e sua elaboração é descrita na atividade *Planejar Fases e Iterações,* da macro-atividade de *Gerenciamento de Projetos*.

O cronograma é revisado na atividade *Revisar Planejamento do Projeto*, da macro-atividade de *Gerenciamento de Projeto*, e acompanhado nos marcos do projeto.

(13/15) Os riscos de software associados com os custos, recursos, cronograma e aspectos técnicos do projeto são identificados, estimados e documentados.

O *Plano de Gerenciamento de Riscos*, englobado pelo PDS, especifica como gerenciar riscos associados ao projeto e inclui uma lista de riscos conhecidos, associados com ações de suavização e contingência.

(14/15) Planos para infra-estrutura de engenharia de software do projeto e ferramentas de suporte são preparados.

Na macro-atividade de *Ambiente* é desenvolvido o *Caso de Desenvolvimento*. Este artefato descreve, entre outros, as ferramentas de apoio às atividades, guias de como usar as ferramentas, e modelos específicos por ferramentas.

RUP falha na revisão dos planos por todos os grupos interessados.

(15/15) Dados do planejamento do software são registrados.

Os dados do planejamento são registrados no *Plano de Desenvolvimento de Software*, e o acompanhamento é registrado na *Avaliação do Status* e no *Registro de Revisão*. Os dados do planejamento são registrados e controlados por meio dos procedimentos descritos na macro-atividade de *Gerenciamento de Configuração e Alteração*. Os dados históricos são armazenados no artefato *Medidas dos Projetos*.

Medição e Análise

(1/1) Medições são feitas e usadas para determinar o status das atividades de planejamento de software.

Na atividade *Definir Processos de Controle e Monitoramento*, são descritos indicadores e processos usados para monitorar e controlar o progresso, qualidade e riscos do projeto.

Verificação da Implementação

(1/3) As atividades para planejamento do projeto de software são revisadas periodicamente junto à gerência sênior.

As atividades de planejamento são revisadas de acordo com o PDS, em pontos pré-definidos, que são os marcos de conclusão de iterações (marcos menores) e de fases (marcos maiores), pelo *Revisor do Projeto*.

(2/3) As atividades para planejamento do projeto são revisadas periodicamente e em determinados eventos com o gerente de projeto.

O *Gerente de Projeto* faz avaliações periódicas do status do projeto e estas são documentadas na *Avaliação de Status*.

(3/3) O grupo de garantia da qualidade de software revisa e/ou audita as atividades e produtos de trabalho de software para o planejamento do projeto de software e informa os resultados.

RUP descreve a elaboração de um *Plano de Garantia de Qualidade*, pelo *Gerente do Projeto*. Neste plano devem ser descritas todas as informações necessárias para realizar as atividades de garantia de qualidade para o projeto. O plano é usado para planejar as revisões e auditorias que verificarão se processo definido para o projeto está sendo seguido corretamente.

O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* (APES) deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas no *Plano de Garantia de Qualidade*.

Resumo da Avaliação

RUP descreve na macro-atividade de *Gerenciamento de Projetos*, as atividades que devem ser executadas para planejar o projeto, dentre elas cita-se: *Desenvolver Plano de Desenvolvimento de Software*, *Planejar Fases e Iterações*, *Revisar Planejamento do Projeto* e *Definir Processos de Controle e Monitoramento*. O RUP propõe um ciclo de vida iterativo e incremental.

O artefato *Plano de Desenvolvimento de Software* (PDS) descreve o planejamento do projeto, a *Visão* descreve a visão geral dos requisitos do projeto, o *Caso de Negócio* descreve uma análise econômica do projeto, o *Plano de Gerenciamento de Riscos* descreve os riscos associados ao projeto, o *Caso de Desenvolvimento* descreve os planos de infra-estrutura e as ferramentas, a *Avaliação de Status* descreve os dados de acompanhamento e o *Registro de Revisão* descreve o resultado da revisão.

O *Gerente de Projeto*, é responsável por negociar compromissos e desenvolver o PDS. Os dados históricos dos projetos são armazenados no artefato *Medidas dos Projetos.*

As verificações de implementação são realizadas pelo *Revisor de Projeto*, *Gerente de Projeto* e APES.

5.2.3 Acompanhamento e Supervisão de Projetos de Software

A finalidade do Acompanhamento e Supervisão de projetos é fornecer uma visão realista do efetivo progresso do projeto para que a gerência possa tomar ações eficazes quando o desempenho do projeto de software desvia significativamente dos planos de software.

Acompanhamento e Supervisão de Projetos envolve acompanhar e revisar o progresso (execução) e os resultados contra as estimativas documentadas, os compromissos e os planos, bem como ajustar esses planos baseados no progresso atual e resultados reais [PAU 93].

Meta 1/3

Os resultados atuais e o desempenho são acompanhados contra os planos de software.

Meta 2/3

Ações corretivas são tomadas e gerenciadas quando os resultados ou desempenho reais desviam significativamente dos planos de software.

Meta 3/3

Mudanças nos compromissos de software são acordados entre os grupos afetados e indivíduos.

Características-comum

Compromisso de Executar

(1/2) Um Gerente de Projeto de software é designado responsável pelas atividades e resultados do projeto de software.

RUP descreve o papel *Gerente de Projeto*, o que é responsável por negociar compromissos, desenvolver o *Plano de Desenvolvimento de Software* (PDS) e acompanhar o progresso do projeto, dentre outras atividades.

(2/2) O projeto segue uma política organizacional para gerenciamento do projeto de software.

O RUP descreve as atividades de acompanhamento de projetos na macro-atividade de *Gerenciamento de Projetos*. Para cada atividade, são descritos os artefatos que devem ser gerados, o trabalhador responsável pela execução e guias. Depende da organização seguir essas atividades.

O *Caso de Desenvolvimento*, elaborado pelo *Engenheiro de Processo*, no início do projeto, adapta as atividades descritas pelo RUP para um projeto específico.

Habilidade para Executar

(1/5) Um plano de desenvolvimento de software para o projeto de software é documentado e aprovado.

O *Plano de Desenvolvimento de Software* (PDS) é usado para gerenciar o projeto de software. PDS é um artefato composto por vários planos, como o *Plano de Iteração*, *Plano de Gerenciamento de Riscos*, *Plano de Medição*, etc. O PDS é desenvolvido durante a atividade *Desenvolver o Plano de Desenvolvimento de Software* e é revisado pelo *Revisor do Projeto*, na atividade *Revisar Planejamento de Projetos*. Essas atividades são descritas na macro-atividade de *Gerenciamento de Projetos*.

(2/5) O Gerente do Projeto de software designa explicitamente responsabilidades para a realização das atividades e elaboração dos produtos de trabalho.

O *Plano de Iteração* descreve os responsáveis pelos produtos de trabalho, atividades e cronograma para o projeto. Na atividade *Definir Equipe e Estrutura Organizacional do Projeto* são descritos procedimentos para definir a estrutura organizacional e a equipe necessária, baseada nas estimativas de esforço e custo.

(3/5) Estão disponíveis os recursos e fundos necessários para acompanhar o projeto de software.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(4/5) Os gerentes de software são treinados em gerenciar aspectos técnicos e pessoais do projeto de software.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(5/5) Gerentes de software de primeira-linha recebem orientação em aspectos técnicos do projeto de desenvolvimento de software.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/13) Um plano de desenvolvimento de software documentado é usado para acompanhar as atividades de software e comunicar o progresso.

O *Plano de Desenvolvimento de Software* é usado para gerenciar o projeto e documentar seu progresso, já as atividades são acompanhadas por meio do *Plano de Iteração*, que faz parte do *Plano de Desenvolvimento de Software*.

(2/13) O plano de desenvolvimento de software é revisado de acordo com um procedimento documentado.

O *Plano de Desenvolvimento de Software* é revisado pelo *Revisor do Projeto*, na atividade *Revisar Planejamento do Projeto*, descrita na macro-atividade de *Gerenciamento de Projetos*.

(3/13) Os compromissos do projeto de software, ou as alterações de compromissos do projeto, firmados com indivíduos e grupos externos a organização são revisados junto à gerência sênior, de acordo com um procedimento documentado.

Os compromissos do projeto, tais como: orçamento, cronograma e recursos, e alterações nesses compromissos, são documentados no *Plano de Desenvolvimento de Software*, que é revisado pelo *Revisor do Projeto*, conforme procedimento descrito na atividade *Revisar Planejamento do Projeto*.

(4/13) Alterações aprovadas nos compromissos que afetam o projeto de software são comunicadas aos membros do grupo de engenharia de software e outros grupos de software relacionados.

As alterações são aprovadas pelo *Conselho de Controle de Alteração*, que é formado por representantes de todos os grupos de software envolvidos no projeto.

(5/13) O tamanho dos produtos de trabalho de software (ou tamanho das alterações nos produtos de trabalho de software) é acompanhado, e ações corretivas são tomadas quando necessário.

As alterações são aprovadas e acompanhadas pelo *Conselho de Controle de Alteração*. Ações corretivas são descritas no *Plano de Gerenciamento de Riscos*.

(6/13) Os custos e os esforços do projeto de software são acompanhados, e ações corretivas são tomadas quando necessário.

Os marcos maiores têm como objetivo acompanhar custos e esforços, para indicar a viabilidade do projeto. O *Plano de Gerenciamento de Riscos* identifica como gerenciar riscos e associa os riscos com ações de suavização e contingência

(7/13) O uso de recursos computacionais críticos do projeto são acompanhados, e ações corretivas são tomadas quando necessário.

Essa prática-chave não é englobada pelo RUP. A proposta 2, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(8/13) O cronograma do projeto é acompanhado, e ações corretivas são tomadas se necessário.

O cronograma do projeto é documentado no *Plano de Iteração* e seu progresso é acompanhado contra este plano. Os marcos são pontos de revisão formal que definem se o projeto deve continuar ou não.

Não deixa claro como comunica os grupos envolvidos no projeto sobre as alterações no cronograma.

(9/13) Atividades técnicas de engenharia de software são acompanhadas, e ações corretivas são tomadas se necessário.

As atividades técnicas são acompanhadas conforme descrito na atividade *Monitorar e Controlar o Projeto*, da macro-atividade de *Gerenciamento de Projetos*. O *Plano de Gerenciamento de Riscos* identifica as ações de contingência e suavização associadas a cada risco. Os marcos do projeto são também pontos de acompanhamento das atividades e do progresso.

(10/13) Os riscos de software associados com custos, recursos, cronograma e aspectos técnicos do projeto são acompanhados.

Na atividade *Desenvolver Plano de Desenvolvimento de Software*, da macro-atividade de *Gerenciamento de Projetos* são desenvolvidos um *Plano de Gerenciamento de Riscos* e uma *Lista de Riscos*, com ações de suavização e contingência para o projeto, que são acompanhados nos marcos maiores e menores do projeto.

(11/13) Dados de medições e de replanejamento para o projeto de software são registrados.

Os dados atuais e os re-planejados estão disponíveis nos *Planos de Desenvolvimento de Software* e *de Iteração* e no artefato de *Medidas de Projetos*.

(12/13) O grupo de engenharia de software conduz revisões internas periódicas para acompanhar o progresso técnico, planos, desempenho e outros itens, contra o plano de desenvolvimento de software.

O RUP propõe a existência de revisões ao final do desenvolvimento dos artefatos principais, para verificar se os objetivos foram satisfeitos. Ao final das iterações e fases também são propostas revisões para acompanhar o progresso do projeto.

(13/13) Revisões formais para verificar o progresso e resultados do projeto são conduzidas nos marcos do projeto selecionados de acordo com procedimentos documentados.

O RUP propõe que ao final de cada fase seja realizada uma revisão formal do projeto. Estas revisões são chamadas de *Marco dos Objetivos do Ciclo de Vida*, *Marco da Arquitetura do Ciclo de Vida*, *Marco da Capacidade Operacional Inicial* e *Marco de Versões do Produto*. Para cada marco são propostos alguns critérios de conclusão que devem ser verificados.

Medição e Análise

(1/1) Medições são feitas e usadas para determinar o status das atividades de acompanhamento e supervisão de projetos.

A atividade *Monitorar e Controlar o Projeto* serve para capturar o status atual do projeto (progresso, esforço, recursos, cronograma, riscos, atividades) e avaliar este status

contra o plano. Os procedimentos utilizados para coletar as métricas estão descritos no *Plano de Medição*.

Verificação da Implementação

(1/3) As atividades de acompanhamento e supervisão de projetos são revisadas periodicamente junto à gerência sênior.

As atividades de acompanhamento e supervisão de projetos e os artefatos de controle desenvolvidos são revisados pelo *Revisor de Projeto*, em pontos de revisão durante o ciclo de vida do projeto.

(2/3) As atividades para acompanhamento e supervisão de projetos são revisadas periodicamente e em determinados eventos com o gerente de projeto.

O *Gerente do Projeto* faz avaliações periódicas do status do projeto e estas são documentadas na *Avaliação de Status*.

(3/3) O grupo de qualidade de software revisa e/ou audita as atividades e produtos de trabalho de software para acompanhamento e supervisão de projetos e relata os resultados.

O *Plano de Garantia de Qualidade*, elaborado pelo *Gerente do Projeto* descreve as atividades que devem ser realizadas para garantir a qualidade do projeto. O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas neste plano.

Resumo da Avaliação

RUP descreve na macro-atividade de *Gerenciamento de Projetos*, as atividades que devem ser executadas para acompanhar e supervisionar o projeto, dentre elas cita-se: *Monitorar e Controlar o Projeto*, *Desenvolver o Plano de Desenvolvimento de Software, Revisar o Planejamento do Projeto* e a realização de revisões formais ao final de cada fase.

O *Gerente de Projeto*, é responsável por negociar compromissos, desenvolver o *Plano de Desenvolvimento de Software* e acompanhar o projeto. O *Conselho de Controle e Alteração* é o responsável por aprovar mudanças.

O *Plano de Iteração* descreve os responsáveis pelos artefatos, atividades e cronograma do projeto. O *Plano de Gerenciamento de Riscos* descreve os principais riscos do projeto e ações corretivas. Os dados históricos dos projetos são armazenados no artefato *Medidas dos Projetos.*

As verificações de implementação são realizadas pelo *Revisor de Projeto*, *Gerente de Projeto* e APES.

5.2.4 Gerenciamento de Subcontratação de Software

A finalidade do Gerenciamento de Subcontratação de Software é selecionar subcontratadas de software qualificadas e gerenciá-las efetivamente.

Gerenciamento de subcontratação de software envolve selecionar uma subcontratada, estabelecendo compromissos com a subcontratada, acompanhar e revisar o

desempenho e os resultados da subcontratada. Estas práticas cobrem o gerenciamento de subcontratação de software (somente), assim como o gerenciamento dos componentes de software da subcontratada, que inclui software, hardware, e possivelmente outros componentes de sistema [PAU 93].

Esta área-chave não é englobada pelo *Processo Unificado Rational*. Este trabalho apresenta no capítulo 6, a proposta de uma macro-atividade de gerenciamento de subcontratação (Proposta 4), que visa satisfazer essa área-chave do CMM.

Meta 1/4

A contratante seleciona subcontratadas de software qualificadas.

Meta 2/4

A contratante e a subcontratada de software concordam com a reciprocidade de seus compromissos.

Meta 3/4

A contratante e a subcontratada de software mantém comunicação ao longo do projeto.

Meta 4/4

A contratante acompanha o desempenho e os resultados reais da subcontratada contra seus compromissos.

Características-comum

Compromisso de Executar

(1/2) O projeto segue uma política organizacional para gerenciar subcontratação de software.

(2/2) Um gerente de subcontratação deve ser designado responsável por estabelecer e gerenciar o subcontrato de software.

Habilidade para Executar

(1/3) Estão disponíveis os recursos e fundos necessários para selecionar subcontratados de software e para gerenciar o contrato.

(2/3) Gerentes de software e outros indivíduos, que estão envolvidos em estabelecer e gerenciar a subcontratação de software, são treinados para executar estas atividades.

(3/3) Gerentes de software e outros indivíduos, que estão envolvidos em gerenciar o contrato de software, recebem orientação nos aspectos técnicos do contrato.

Atividades Executadas

(1/13) O trabalho a ser contratado é definido e planejado de acordo com um procedimento documentado.

(2/13) A subcontratada de software é selecionada, com base em uma avaliação da habilidade dos proponentes em executar o trabalho, de acordo com um procedimento documentado.

(3/13) O acordo contratual entre a contratante e a subcontratada de software é usado como base para gerenciar a subcontratação.

(4/13) O plano de desenvolvimento de software da subcontratada é revisado e aprovado pela contratante.

(5/13) O plano de desenvolvimento de software documentado e aprovado da subcontratada é usado para acompanhar as atividades e informar o progresso.

(6/13) Alterações na declaração de trabalho, termos do subcontrato e condições, e outros compromissos da subcontratada de software são resolvidas de acordo com um procedimento documentado.

(7/13) A contratante conduz revisões periódicas de coordenação e de avaliação do progresso, junto à gerência da subcontratada de software.

(8/13) Revisões técnicas periódicas e intercâmbio são mantidos com a subcontratada de software.

(9/13) Revisões formais, para abordar os resultados e compromissos de engenharia de software, são conduzidas nos marcos de acompanhamento, de acordo com um procedimento documentado.

(10/13) O grupo de garantia de qualidade de software da contratante monitora as atividades de garantia de software da subcontratada, de acordo com um procedimento documentado.

(11/13) O grupo de gerenciamento de configuração da contratante monitora as atividades para o gerenciamento de configuração de software da subcontratada, de acordo com um procedimento documentado.

(12/13) A contratante conduz teste de aceite, como parte da entrega dos produtos de software da subcontratada, de acordo com um procedimento documentado.

(13/13) O desempenho da subcontratada de software é avaliado periodicamente e a avaliação é revista com a subcontratada.

Medição e Análise

(1/1) Medições são feitas e usadas para determinar o status das atividades de gerência de subcontratação de software.

Verificação da Implementação

(1/3) As atividades de gerência de subcontratação de software são revisadas periodicamente junto à gerência sênior.

(2/3) As atividades de gerência de subcontratação de software são revisadas periodicamente e em determinados eventos com o gerente de projeto.

(3/3) O grupo de garantia da qualidade de software revisa e/ou audita as atividades e produtos de trabalho para gerência de subcontratação e informa os resultados.

5.2.5 Garantia da Qualidade de Software

A finalidade da Garantia de Qualidade de Software é fornecer à gerência a visibilidade apropriada dos processos usados pelo projeto de software e dos produtos sendo construídos.

Garantia da qualidade de software envolve revisar e auditar os produtos de software e atividades, para verificar se eles cumprem os procedimentos e padrões aplicáveis, bem como prover os resultados destas revisões e auditorias ao projeto de software e aos gerentes envolvidos [PAU 93].

Meta 1/4

Atividades de garantia de qualidade de software são planejadas.

Meta 2/4

A conformidade dos produtos de software e atividades aos padrões, procedimentos e requisitos aplicáveis é verificada objetivamente.

Meta 3/4

Grupos e indivíduos afetados são informados sobre as atividades e resultados de garantia de qualidade de software.

Meta 4/4

Questões que não podem ser resolvidas internamente no projeto de software são encaminhadas à gerência sênior.

Características-comum

Compromisso de Executar

(1/1) O projeto segue uma política organizacional para implementar garantia da qualidade de software (GQS).

RUP propõe a elaboração de um *Plano de Garantia de Qualidade* (PGQ). Este plano deve descrever como a qualidade do processo, dos artefatos e do produto será assegurada. A atividade *Desenvolver Plano de Garantia de Qualidade*, da macro-atividade de *Gerenciamento de Projetos,* descrever os procedimentos para desenvolver o PGQ. Este trabalho propõe uma alteração nesta atividade, descrita no capítulo 6, visando compatibilizar com o CMM.

Habilidade para Executar

(1/4) Um grupo que é responsável pela coordenação e implementação de SQA para o projeto (grupo de SQA) existe.

O Processo Unificado Rational recomenda que a *Autoridade de Processo de Engenharia de Software* (APES) deva ter a responsabilidade pelos aspectos de qualidade, executar revisões e auditorias, bem como assegurar o planejamento apropriado e a condução de revisões descritas na seção *Auditoria e Revisão*, do *Plano de Garantia de Qualidade*. Este trabalho propõe que a APES seja encarregada de desenvolver o *Plano de Garantia de Qualidade* para o projeto.

(2/4) Estão disponíveis os recursos e fundos necessários para executar as atividades de SQA.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(3/4) Membros do grupo de SQA são treinados para executar suas atividades de SQA.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(4/4) Os membros do projeto de software recebem orientação sobre o papel, as responsabilidades, e o valor do grupo de SQA.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/8) Um plano de SQA é preparado para o projeto de software de acordo com um procedimento documentado.

Para cada projeto é criado um *Plano de Garantia da Qualidad*e. Este plano serve para planejar todas as atividades relacionadas à garantia da qualidade, como as auditorias e as revisões.

(2/8) As atividades do grupo de SQA são executadas de acordo com o plano SQA.

O *Plano de Garantia de Qualidade* descreve as atividades, depende da organização seguir o plano.

(3/8) O grupo de SQA participa na preparação e revisão do plano de desenvolvimento de software do projeto, dos padrões e dos procedimentos.

Considerando que segundo no RUP, as atividades de Garantia de Qualidade são realizadas pela Autoridade de Engenharia de Processo, a proposta 3 descreve como satisfazer essa prática-chave.

(4/8) O grupo de SQA revisa as atividades de engenharia de software para verificar o seu cumprimento.

RUP descreve revisões de atividades de engenharia de software pela APES, mas não deixa claro como os desvios são identificados, documentados e acompanhados até a conclusão, nem como as ações corretivas são verificadas.

(5/8) O grupo de SQA audita produtos de trabalho de software, para verificar a sua conformidade com os padrões, os procedimentos e os requisitos estabelecidos.

O *Plano de Garantia de Qualidade* descreve as revisões e auditorias nos artefatos, mas não deixa claro como os desvios são identificados, documentados e acompanhados até a conclusão, nem como as ações corretivas são verificadas.

(6/8) O grupo de SQA periodicamente informa o resultado de suas atividades para o grupo de engenharia de software.

RUP recomenda que o resultado das atividades de garantia de qualidade deva ser submetido diretamente a Autoridade de Revisão do Projeto. Não deixa claro como o resultado é informado ao grupo de engenharia de software.

(7/8) Desvios identificados nas atividades de software e produtos de trabalho são documentados e tratados de acordo com um procedimento documentado.

Durante as revisões executadas pelo *Revisor do Projeto*, o resultado é documentado no *Registro de Revisão*.

(8/8) O grupo de SQA conduz revisões periódicas de suas atividades de SQA e dos seus relatórios com o grupo de SQA do cliente, quando apropriado.

Essa prática-chave não é englobada pelo RUP. A proposta 3, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

Medição e Análise

(1/1) Medições são feitas e usadas para determinar o custo e o status no cronograma das atividades de SQA.

Essa prática-chave não é englobada pelo RUP. A proposta 3, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

Verificação da Implementação

(1/3) As atividades de SQA são revisadas periodicamente com gerenciamento sênior.

As atividades de Garantia de Qualidade de Software descritas pelo RUP são realizadas ao final das iterações e das fases pelo *Revisor do Projeto*.

(2/3) As atividades de SQA são revisadas periodicamente e em determinados eventos com o gerente de projeto.

O *Gerente de Projeto* revisa as atividades de SQA ao final das fases e iterações e prepara o material para as revisões pelo *Revisor do Projeto*.

(3/3) Especialistas independentes do grupo de SQA periodicamente revisam as atividades e produtos de trabalho de software do grupo de SQA do projeto.

Essa prática-chave não é englobada pelo RUP. A proposta 3, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

Resumo da Avaliação

RUP propõe a elaboração de um *Plano de Garantia de Qualidade* para cada projeto. Este plano deve descrever as atividades que devem ser realizadas para garantir a qualidade do processo, dos artefatos e do produto.

A atividade *Desenvolver Plano de Garantia de Qualidade*, da macro-atividade de *Gerenciamento de Projetos*, descreve como desenvolver este plano. Este trabalho propõe que essa atividade seja realizada pela *Autoridade de Processo de Engenharia de Software*, e não pelo *Gerente de Processo*, como determina o RUP.

O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* (APES) deva ter a responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas na seção *Auditoria e Revisão*, do *Plano de Garantia de Qualidade*.

RUP não deixa claro como são tratados os problemas identificados nas revisões e auditorias, e nem como os resultados dessas revisões são repassados ao *Grupo de Engenharia de Software*.

5.2.6 Gerenciamento de Configuração de Software

A finalidade do Gerenciamento de Configuração de Software é estabelecer e manter a integridade dos produtos do projeto de software, ao longo do ciclo de vida do software.

O Gerenciamento de Configuração de Software envolve identificar a configuração de software (isto é, os produtos de trabalho de software selecionados e suas descrições) em um dado ponto no tempo, controlar sistematicamente as mudanças na configuração, e manter a integridade e a rastreabilidade da configuração ao longo do ciclo de vida do software [PAU 93].

Meta 1/4

As atividades de gerenciamento de configuração de software são planejadas.

Meta 2/4

Produtos de trabalho de software são identificados, controlados e estão disponíveis.

Meta 3/4

Alterações nos produtos de trabalho de software identificados são controlados.

Meta 4/4

Grupos ou indivíduos afetados são informados do status ou conteúdo das configurações-base[5] de software.

Características-comum

Compromisso de Executar

(1/1) O projeto segue uma política organizacional para implementar gerenciamento de configuração de software (GCS).

A atividade *Planejar a Configuração do Projeto e Controle de Alterações,* da macro-atividade de *Gerenciamento de Configuração e Alteração* tem a finalidade de estabelecer políticas de gerenciamento de configuração do projeto e processos para controlar as alterações nos produtos e documentar essas informações no *Plano de Gerenciamento de Configuração*.

Habilidade para Executar

---

[5] Configurações-base (*Baselines*): conjunto de produtos de trabalho aceitos e controlados e que serão utilizados em atividades posteriores à sua aceitação.

(1/5) Uma comissão tendo a autoridade para gerenciar as configurações-base do projeto de software (isto é, conselho de controle de configuração de software) existe ou está estabelecida.

O RUP propõe a criação de um *Conselho de Controle de Alteração* (CCA) que tem a finalidade de definir o processo de solicitação de alteração e administrar as solicitações de alteração. O CCA é formado por representantes de todos os grupos interessados no projeto, incluindo clientes, desenvolvedores e usuários.

(2/5) Um grupo que é responsável pela coordenação e implementação de GCS para o projeto de software (isto é, grupo de GCS) existe.

RUP descreve o papel *Gerente de Configuração,* que é responsável por instalar a estrutura do produto no sistema de gerência de configuração, definir e alocar áreas de trabalho (*workspaces*) para desenvolvedores e para integração. O *Gerente de Configuração* também é o responsável por elaborar o *Plano de Gerenciamento de Configuração*.

(3/5) Estão disponíveis os recursos e fundos necessários para executar as atividades de GCS.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(4/5) Membros do grupo de GCS são treinados nos objetivos, procedimentos, e métodos para executar suas atividades de GCS.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(5/5) Membros do grupo de engenharia de software e outros grupos relacionados são treinados para executar suas atividades GCS.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/10) Um plano de GCS é preparado para cada projeto de software de acordo com um procedimento documentado.

Para cada projeto é criado um *Plano de Gerenciamento de Configuração,* que descreve a política e as práticas que serão usadas no projeto para gerenciamento de configuração: versões, variáveis, áreas de trabalho e procedimentos para gerenciamento de alteração, *builders* e *releases*. Este plano é realizado de acordo com o procedimento descrito na atividade *Planejar Configuração do Projeto e Controle de Alteração*, da macro-atividade de *Gerenciamento de Configuração e Alteração*.

(2/10) Um plano de GCS aprovado e documentado é usado como base para executar as atividades de GCS.

O *Plano de Gerenciamento de Configuração* detalha o cronograma das atividades, as responsabilidades assinaladas e os recursos requeridos.

(3/10) Um sistema de biblioteca de gerência de configuração está estabelecida como um repositório para as configurações-base de software.

As *configurações-base* do software são criadas e gerenciadas pelo *Integrador do Sistema*, a partir da área de trabalho de integração e são disponibilizadas para as pessoas da equipe. RUP descreve como realizar essa atividade usando a ferramenta ClearCase, da Rational, mas o processo é genérico, possibilitando a organização escolher a ferramenta que deseja utilizar.

(4/10) Os produtos de trabalho de software colocados sob gerência de configuração são identificados.

Na atividade *Estabelecer Políticas de Gerenciamento de Configuração* são definidos os artefatos que serão colocados sob Gerenciamento de Configuração.

(5/10) Mudanças nos requisitos e relatório de problemas para todos os itens/unidades são identificados, registrados, revisados, aprovados, e acompanhados de acordo com um procedimento documentado.

As alterações nos requisitos podem ser identificadas por qualquer membro da equipe, que encaminha esta alteração por meio do artefato *Solicitação de Alteração*. Esta solicitação, será julgada e revisada pelo *Conselho de Controle de Alteração*, de acordo o processo de controle de alteração estabelecido, e caso seja aprovada é encaminhada para alteração ao *Gerente de Projeto*.

(6/10) Alterações nas configurações-base são controladas de acordo com um procedimento documentado.

Na atividade *Estabelecer Processo de Controle de Alteração* são definidos os procedimentos para alteração de artefatos. O *Plano de Gerenciamento de Configuração* documenta os procedimentos de controle de alterações.

(7/10) Controlar a liberação dos produtos criados a partir da biblioteca (repositório) de configurações-base do software, de acordo com um procedimento documentado.

A atividade *Gerenciar Configurações-base e Releases* tem a finalidade de controlar as versões do sistema e os produtos que compõem cada versão. Os procedimentos para executar essa atividade são descritos no *Plano de Gerenciamento de Configuração*.

(8/10) O status dos itens/unidades de configuração é registrado de acordo com um procedimento documentado.

A atividade *Monitorar e Informar Status*, da macro-atividade *Gerenciamento de Configuração e Alteração*, tem a finalidade de controlar e registrar o status dos itens de configuração.

(9/10) Relatórios padrão documentando as atividades de GCS e o conteúdo das configurações-base do software são desenvolvidos e disponibilizados aos grupos e indivíduos afetados.

A atividade *Monitorar e Relatar Status de Configuração*, da macro-atividade de *Gerenciamento de Configuração e Alteração*, descreve procedimentos para monitor e relatar o status das atividades de GCS, e a atividade *Gerenciar Configurações-base e Releases,* descreve como o conteúdo das configurações-base é disponibilizado aos grupos interessados.

(10/10) Auditorias nas configurações-base de software são conduzidas de acordo com um procedimento documentado.

A atividade *Executar Auditorias de Configuração*, da macro-atividade de *Gerenciamento de Configuração e Alteração* descreve auditorias nas configurações-base para determinar se esta contém todos os artefatos requeridos e encontra seus requisitos. Os resultados da auditoria são relatados no artefato *Resultado da Auditoria de Configuração*.

Medição e Análise

(1/1) Medidas são feitas e usadas para determinar o status das atividades de GCS.

A atividade *Monitorar e Informar Status* descreve os procedimentos para coletar e registrar medidas dos projetos relacionadas a GCS, como exemplo de medições cita-se: número de alterações propostas e status de implementação dessas alterações.

Verificação da Implementação

(1/4) As atividades de GCS são revisadas periodicamente junto à gerência sênior.

As atividades de gerenciamento de configuração são revisadas pelo *Conselho de Controle de Alteração*.

(2/4) As atividades de GCS são revisadas com o gerente de projeto periodicamente e em determinados eventos

O *Gerente de Configuração* apresenta para o *Gerente do Projeto* o relatório de status periodicamente.

(3/4) O grupo de GCS periodicamente audita configurações-base de software para verificar que elas estão conforme a documentação que as define.

O *Gerente de Configuração* audita as configurações-base para determinar se elas encontram seus requisitos, e contêm todos os artefatos requeridos, esse procedimento está descrito na atividade *Executar Auditorias de Configuração*.

(4/4) O grupo de garantia de qualidade de software revisa e/ou audita as atividades e os produtos de trabalho para GCS e relata os resultados.

O *Plano de Garantia de Qualidade*, elaborado pelo *Gerente do Projeto* descreve as atividades que devem ser realizadas para garantir a qualidade do projeto. O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas neste plano.

Resumo da Avaliação

A macro-atividade *Gerenciamento de Configuração e Alteração*, descreve as atividades para gerenciamento de configuração e controle de alteração, dentre estas cita-se: *Planejar a Configuração do Projeto e o Controle de Alterações, Estabelecer Políticas de Gerenciamento de Configuração, Estabelecer Processo de Controle de Alteração, Gerenciar Configurações-base, Executar Auditorias nas Configurações-base e Monitorar e Informar Status*

O *Plano de Gerenciamento de Configuração* descreve as atividades relacionadas ao gerenciamento de configuração e ao controle de mudanças.

O *Conselho de Controle de Alteração* (CCA) avalia as solicitações de mudanças, aprovando ou reprovando-as, supervisiona o processo de solicitação de alteração. O *Gerente de Configuração* é responsável pela infra-estrutura de gerenciamento de configuração, por elaborar o plano e relatar o progresso baseado nas solicitações de mudança. O I*ntegrador* cria e gerencia as *configurações-base*.

5.3 Áreas-Chave do CMM - Nível 3

5.3.1 Foco nos Processos da Organização

A finalidade do Foco nos Processos da Organização é estabelecer a responsabilidade organizacional para as atividades do processo de software que melhoram a capacidade geral do processo de software da organização.

Foco nos Processos da Organização envolve desenvolver e manter um entendimento dos processos de software dos projetos e da organização, e coordenar as atividades para avaliar, desenvolver, manter e melhorar esses processos [PAU 93].

Meta 1/3

O desenvolvimento dos processos de software e as atividades de melhoria são coordenados através da organização.

Meta 2/3

Os pontos fortes e as oportunidade de melhoria dos processos de software usados na organização são identificados comparando com o processo-padrão.

Meta 3/3

As atividades de melhoria e desenvolvimento do processo no nível da organização são planejadas.

Características-comum

Compromisso de Executar

(1/3) A organização segue uma política organizacional para coordenar as atividades de desenvolvimento e melhoria dos processos de software através da organização.

O RUP descreve a atividade *Avaliar a Organização Atual*, na macro-atividade de *Ambiente*. A proposta 6, descrita no capítulo 6 deste trabalho visa inserir características de melhoria de processo na atividade *Avaliar Organização Atual*.

(2/3) A gerência sênior apóia as atividades de desenvolvimento e melhoria do processo de software da organização.

O RUP sugere que as atividades relacionadas à avaliação dos processos de desenvolvimento e melhoria devam ser coordenadas pela *Autoridade de Processo de Engenharia de Software* (APES), que é representada por um gerente geral da organização.

(3/3) A gerência sênior fiscaliza as atividades de desenvolvimento e melhoria do processo de software da organização.

A proposta 6, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

Habilidade de Executar

(1/4) Existe um grupo que é responsável pelas atividades de processo de software da organização.

O *Engenheiro de Processo* é responsável pelo processo de desenvolvimento de software, e a Autoridade de Engenharia de Processo participa das avaliações de processo promova a troca de informações entre os projetos.

(2/4) Estão disponíveis os recursos e fundos necessários para as atividades de processo de software da organização.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(3/4) Membros do grupo responsável pelas atividades de processo de software da organização recebem treinamento para executar essas atividades.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(4/4) Membros do grupo de engenharia de software e outros grupos relacionados ao software recebem orientação nas atividades de processo de software da organização e em seus papéis nessas atividades.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/7) O processo de software é avaliado periodicamente, e planos de ações são desenvolvidos para enfocar o resultado das avaliações.

O processo de software é avaliado no início de cada projeto e revisto no final de cada iteração, na atividade *Avaliar Organização Atual*, e os resultados são documentados na *Avaliação da Organização*. A proposta 6, descrita no capítulo 6 deste trabalho propõe a elaboração de um *Plano de Melhoria de Processo* que deve ser implementado pela *Autoridade de Engenharia de Processo da Organização*.

(2/7) A organização desenvolve e mantém um plano para as atividades de desenvolvimento e melhoria de seus processos de software.

A proposta 6, descrita no capítulo 6, propõe a inserção de uma atividade para desenvolver um *Plano para Melhoria do Processo da Organização*, baseada na avaliação da organização.

(3/7) As atividades para desenvolvimento e melhoria dos processos de software dos projetos e da organização são coordenadas no nível da organização.

RUP propõe a criação de uma entidade no nível da organização chamada *Autoridade de Processo de Engenharia de Software* (APES). Esta entidade visa manter avaliações sobre os processos da organização.

(4/7) O uso de base de dados do processo de software da organização é coordenado no nível da organização.

RUP propõe o artefato *Medidas dos Projetos*, com a finalidade de armazenar dados do processo e dos projetos da organização. Cada *Gerente de Projeto* é responsável pelos dados de seus projetos.

(5/7) Novos processos, métodos e ferramentas em uso limitado na organização são monitorados, avaliados, e quando apropriado, transferidos a outras partes da organização.

A *Autoridade de Processo de Engenharia de Software* tem como responsabilidade promover a troca de informação entre os projetos e disseminar melhores práticas.

(6/7) Treinamento nos processos de software da organização e nos projetos é coordenado através da organização.

O *Plano de Treinamento*, descrito na macro-atividade de *Treinamento*, extensão do RUP proposta por este trabalho, deve ser coordenado no nível da organização.

(7/7) Os grupos envolvidos na implementação dos processos de software são informados das atividades para desenvolvimento e melhoria dos processos de software dos projetos e da organização.

A *Autoridade de Processo de Engenharia de Software* (APES) é responsável por facilitar a troca de informações e guias de processos entre todos os grupos, catalisar a captura e disseminação das melhores práticas de software, quando entender que as melhorias são desejadas no contexto do projeto.

Medição e Análise

(1/1) Medições são feitas e usadas para determinar o status das atividades de desenvolvimento e melhoria dos processos da organização.

O *Plano de Melhoria do Processo,* proposto no capítulo 6, deve descrever as métricas utilizadas para acompanhar as atividades descritas e possibilitar o acompanhamento do status dessas atividades.

Verificação da Implementação

(1/1) As atividades para desenvolvimento e melhoria dos processos de software da organização são revisadas periodicamente junto à gerência sênior.

As atividades de desenvolvimento e melhoria devem ser revisadas pela *Autoridade de Revisão do Processo*, conforme descrito na proposta 6, apresentada neste trabalho.

Resumo da Avaliação

O RUP descreve a atividade *Avaliar a Organização Atual*, na macro-atividade de *Ambiente*, que tem como objetivo avaliar o processo de desenvolvimento e documentar na Avaliação da Organização.

A *Autoridade de Processo de Engenharia de Software* (APES) mantém avaliações sobre os processos da organização, promove a troca de informação entre os projetos e dissemina as melhores práticas. O *Engenheiro de Processo* é responsável por definir o processo de desenvolvimento de software para cada projeto.

RUP não descreve como as melhorias decorrentes da avaliação do processo são implementadas, incluindo elaboração de plano e realização de revisões, como preconiza o CMM.

5.3.2 Definição dos Processos da Organização

A finalidade da Definição dos Processos da Organização é desenvolver e manter um conjunto utilizável de bens do processo de software[6], que melhore o desempenho do processo, ao longo dos projetos e forneça um histórico que traga benefícios a organização.

Definição dos Processos da Organização envolve desenvolver e manter o processo de software padrão da organização, assim como os bens relacionados ao processo, tais como: descrição do ciclo de vida do software, guias e critérios de adaptação para o processo, base de dados de processos de software da organização e uma biblioteca de documentação relacionada ao processo de software [PAU 93].

Meta 1/2

Um processo de software padrão para a organização é desenvolvido e mantido.

Meta 2/2

Informações relacionadas ao uso do processo de software padrão da organização pelos projetos de software são coletadas, revisadas, e estão disponíveis.

Características-comum

Compromisso de Executar

(1/1) A organização segue uma política para desenvolver e manter um processo de software padrão e os bens relacionados ao processo.

O Processo Unificado propõe guias de configuração do RUP para uma organização. Estes guias levam em consideração o domínio das aplicações, práticas de reuso, e as tecnologias usadas pela organização. A organização pode ter mais de um processo, cada processo adaptado para um tipo de desenvolvimento. Em muitos casos o Processo Unificado serve como processo padrão da organização. Na macro-atividade de *Ambiente* são descritos esses procedimentos de configuração e demais atividades para manter o processo de software da organização.

Habilidade de Executar

(1/2) Estão disponíveis os recursos e fundos necessários para desenvolver e manter o processo de software padrão da organização e bens relacionados ao processo.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

---

[6] Bens do Processo de Software é um conjunto de artefatos mantidos pela organização e utilizados pelos projetos para o desenvolvimento, adaptação, manutenção e implementação de seus processos de software.

(2/2) Os indivíduos que desenvolvem e mantém o processo de software padrão da organização e os bens relacionados ao processo recebem treinamento para executar estas atividades.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/6) O processo de software padrão da organização é desenvolvido e mantido de acordo com um procedimento documentado.

O Processo Unificado apresenta procedimentos para configurar o processo as mais diversas necessidades e projetos. RUP também sugere que exista uma *Autoridade de Processo de Engenharia de Software*, com a função de desenvolver e manter o processo padrão da organização.

(2/6) O processo de software padrão da organização é documentado de acordo com padrões estabelecidos pela organização.

O processo de software padrão da organização deve ser documentado em um *site*, possibilitando o acesso a toda organização, de acordo com padrões estabelecidos pelo RUP e descritos no processo.

(3/6) Descrições do ciclo de vida da organização que estão aprovadas para uso pelos projetos são documentadas e mantidas.

O *Processo Unificado Rational* propõe um ciclo de vida iterativo e incremental, e possibilita configurá-lo para os mais diversos tipos de sistema. O processo customizado para um projeto específico deve ser descrito no *Caso de Desenvolvimento*.

(4/6) Guias e critérios para adaptação do processo de software padrão da organização para projetos específicos são desenvolvidos e mantidos.

A atividade *Desenvolver um Caso de Desenvolvimento,* da macro-atividade *Ambiente,* descreve um procedimento documentado de como configurar o processo de software padrão da organização para projetos específicos. Como resultado desta atividade tem-se o *Caso de Desenvolvimento* específico para o projeto, descrevendo o processo de desenvolvimento configurado de acordo com as necessidades do projeto.

(5/6) A base de dados do processo de software da organização está estabelecida e é mantida.

O RUP propõe o artefato *Medidas dos Projetos,* para armazenar os dados dos projetos e dos processos. O controle e manutenção deste artefato são de responsabilidade do *Gerente de Projeto*.

(6/6) A biblioteca de documentação relacionada ao processo de software está estabelecida e é mantida.

A macro-atividade de *Gerenciamento de Alteração e Configuração* descreve como gerenciar os bens do processo de desenvolvimento.

Medição e Análise

(1/1) Medidas são feitas e usadas para determinar o status das atividades de definição do processo da organização.

Medidas para acompanhamento das atividades de definição de processos da organização são armazenadas no artefato de *Medidas dos Projetos*. O monitoramento e controle do progresso das atividades são realizados na atividade *Monitorar Status do Projeto*, da macro-atividade de *Gerenciamento de Projetos.*

Verificação da Implementação

(1/1) O grupo de garantia da qualidade de software revisa e/ou audita as atividades da organização e produtos de trabalho para desenvolver e manter o processo de software padrão da organização e bens relacionados ao processo e informa os resultados.

O *Plano de Garantia de Qualidade*, elaborado pelo *Gerente do Projeto* descreve as atividades que devem ser realizadas para garantir a qualidade do projeto. O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas neste plano.

Resumo da Avaliação

RUP propõe na macro-atividade de *Ambiente* as atividades e os guias para configurar o *Processo Unificado Rational* para um processo padrão da organização, e para configurar o processo padrão da organização para um projeto específico. Em muitos casos, o processo padrão da organização é o Processo Unificado.

O *Caso de Desenvolvimento* descreve o processo de desenvolvimento configurado de acordo com as necessidades do projeto. O processo de software da organização deve ser documentado em um *site*, possibilitando o acesso a toda organização.

A *Autoridade de Processo de Engenharia de Software*, tem a função de desenvolver e manter o processo padrão da organização. O *Engenheiro de Processo* é o responsável por elaborar o *Caso de Desenvolvimento.*

As atividades de definição do processo são descritas no *Plano de Desenvolvimento de Software* e acompanhadas na atividade *Monitorar Status do Projeto*, da macro-atividade de *Gerenciamento de Projetos.* A verificação de implementação é realizada pela *Autoridade de Processo de Engenharia de Software*.

5.3.3 Programa de Treinamento

A finalidade da área-chave de processo Programa de Treinamento é desenvolver as habilidades e conhecimento dos indivíduos de modo que eles possam desempenhar efetiva e eficientemente seus papéis.

Programa de Treinamento envolve, primeiro identificar os treinamentos necessários para a organização, projetos e indivíduos, e então desenvolver ou procurar treinamento que enfoquem estas necessidades [PAU 93].

Esta área-chave não é englobada pelo *Processo Unificado Rational*. Este trabalho apresenta no capítulo 6 uma proposta de macro-atividade, que tem o objetivo de estender o RUP para incorporar as práticas-chave descritas no Programa de Treinamento.

Meta 1/3

Atividades de treinamento são planejadas.

Meta 2/3

São fornecidos treinamentos para desenvolver as habilidades e os conhecimentos necessários para desempenhar papéis técnicos e gerenciais para o desenvolvimento de software.

Meta 3/3

Indivíduos do grupo de engenharia de software e grupos relacionados ao software recebem treinamento necessário para desempenhar seus papéis.

Características-comum

Compromisso de Executar

(1/1) A organização segue uma política para atingir suas necessidades de treinamento.

Habilidade de Executar

(1/4) Um grupo responsável por identificar e satisfazer as necessidades de treinamento da organização existe.

(2/4) Estão disponíveis os recursos e fundos necessários para implementar o programa de treinamento.

(3/4) Membros do grupo de treinamento têm as habilidades e os conhecimentos para executar as atividades de treinamento.

(4/4) Gerentes de software recebem orientação sobre o programa de treinamento.

Atividades Executadas

(1/6) Cada projeto de software desenvolve e mantém um plano de treinamento que especifica as necessidades de treinamento do projeto.

(2/6) O plano de treinamento da organização é desenvolvido e revisado de acordo com um procedimento documentado.

(3/6) Os treinamentos são executados de acordo com o plano de treinamento da organização.

(4/6) Cursos preparados no nível da organização são desenvolvidos e mantidos de acordo com padrões da organização.

(5/6) Um procedimento de dispensa de treinamento[7] capaz de determinar se os indivíduos já possuem os conhecimentos ou as habilidades necessários ao pleno desempenho de seus papéis, está estabelecido e é usado.

---

[7] Procedimento de Dispensa de Treinamento é uma aprovação documentada dispensando uma pessoa de um treinamento que deveria fazer para poder desempenhar um papel específico. Esta dispensa é concedida somente se for comprovado que a pessoa já possui os conhecimentos e habilidades necessárias para desempenhar eficazmente seu papel.

(6/6) Registros de treinamento são mantidos.

Medição e Análise

(1/2) Medições são feitas e usadas para determinar o status das atividades do programa de treinamento.

(2/2) Medições são feitas e usadas para determinar a qualidade do programa de treinamento.

Verificação da Implementação

(1/3) As atividades do programa de treinamento são revisadas periodicamente junto à gerência sênior.

(2/3) O programa de treinamento é avaliado independentemente, e periodicamente para verificar a consistência e a relevância do programa de treinamento em relação às necessidades da organização.

(3/3) As atividades e produtos de trabalho do programa de treinamento são revisados e/ou auditados e os resultados são informados.

5.3.4 Gerenciamento Integrado de Software

A finalidade do Gerenciamento Integrado de Software é integrar as atividades de engenharia e gerência software em um processo de software coerente e definido, que é adaptado para o processo de software padrão da organização e bens de processo relacionados, os quais são descritos na Definição do Processo da Organização.

Gerenciamento Integrado de Software envolve desenvolver o processo de software definido do projeto e gerenciar o projeto de software, usando este processo de software definido. O processo de software definido do projeto é adaptado do processo de software padrão da organização para enfocar as características específicas do projeto [PAU 93].

Meta 1/2

O processo de software definido do projeto é uma versão adaptada do processo de software padrão da organização.

Meta 2/2

O projeto é planejado e gerenciado de acordo com o processo de software definido do projeto.

Características-comum

Compromisso de Executar

(1/1) O projeto segue uma política organizacional que exige que o projeto de software seja planejado e gerenciado usando o processo de software padrão da organização e bens de processo relacionados.

A macro-atividade de *Ambiente* especifica como configurar o processo de software da organização, ou o *Processo Unificado Rational* se a organização não tem um processo próprio, para um projeto específico. Este processo adaptado é descrito no *Caso de*

Desenvolvimento, que descreve as atividades para planejar e gerenciar todo o projeto. Depende da organização seguir esse processo.

Habilidade de Executar

(1/3) Estão disponíveis os recursos e fundos necessários para gerenciar o projeto de software usando o processo de software definido para o projeto.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(2/3) Os indivíduos responsáveis por desenvolver o processo de software definido recebem treinamento em como adaptar o processo de software padrão e como usar os bens de processo relacionados.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(3/3) Os gerentes de software recebem treinamentos para gerenciar os aspectos técnicos, administrativos e de pessoal do projeto de software, baseado no processo de software definido do projeto.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/11) O processo de software definido do projeto é desenvolvido adaptando-se o processo de software padrão da organização de acordo com um procedimento documentado.

A macro-atividade de *Ambiente* descreve como configurar o processo de software da organização para um projeto específico. Na atividade desenvolver *Caso de Desenvolvimento,* o processo é adaptado e descrito no artefato *Caso de Desenvolvimento*. RUP fornece guias para auxiliar na configuração de cada macro-atividade.

(2/11) Cada processo de software definido do projeto é revisado de acordo com um procedimento documentado.

Uma das atividades da macro-atividade de *Gerenciamento de Projetos* é *Revisar o Planejamento do Projeto*. Nesta atividade, o *Revisor do Projeto* revisa o *Plano de Desenvolvimento de Software*, no qual está inserido o *Caso de Desenvolvimento*. A *Autoridade de Processo de Engenharia de Software* é responsável por manter o processo de software da organização, e conforme proposto neste trabalho (proposta 6), deve revisar as melhorias propostas para o processo de software padrão da organização.

(3/11) O plano de desenvolvimento de software do projeto, que descreve o uso do processo de software definido do projeto, é desenvolvido e revisado de acordo com um procedimento documentado.

Na atividade *Revisar Planejamento do Projeto*, da macro-atividade de *Gerenciamento de Projetos*, o *Revisor do Projeto* revisa o P*lano de Desenvolvimento do Software*.

(4/11) O projeto de software é gerenciado de acordo com o processo de software definido do projeto.

Depende da organização utilizar os procedimentos definidos no *Caso de Desenvolvimento* para gerenciar o projeto.

(5/11) A base de dados do processo de software da organização é usada para planejamento e estimativa de software.

Os dados do repositório, *Medidas dos Projetos,* são usados nas atividades: *Monitorar Status do Projeto* e *Informar Status.* Na atividade *Planejar Fases e Iterações,* RUP propõe que os dados coletados sejam usados em estimativas futuras.

(6/11) O tamanho dos produtos de trabalho de software (ou tamanho das alterações para os produtos de software) é gerenciado de acordo com um procedimento documentado.

O RUP propõe que para estimar o tamanho do software, seja considerado o tamanho dos produtos de trabalho e das alterações.

Na Fase de *Concepção do Sistema*, a atividade *Planejar Fases e Iterações* (macro-atividade de *Gerenciamento de Projetos*) são identificadas duas formas de estimar o tamanho dos produtos, que são: por analogia ou análise.

Na estimativa de tamanho por analogia, o novo produto que será desenvolvido é comparado com produtos desenvolvidos anteriormente. Várias características do produto podem ser comparadas como: número de casos de uso, número de atores, tamanho/complexidade da base de dados, número de programas *online* e *batch*.

Após a fase de *concepção* se têm mais informações sobre o produto, sendo possível utilizar técnicas analíticas para estimar o tamanho do produto. Estas técnicas se baseiam na descrição funcional do produto de software e aplicação de regras de contagem padrão para determinar medidas de tamanho para estas descrições. Como exemplo, pode-se citar *Pontos por Função*, *Pontos por Casos de Uso*, etc.

(7/11) Os esforços e custos do projeto de software são gerenciados de acordo com um procedimento documentado.

Nos marcos de acompanhamento do projeto, os custos e esforços são acompanhados para determinar a viabilidade de se continuar o projeto ou não.

(8/11) Os recursos computacionais críticos do projeto são gerenciados de acordo com um procedimento documentado.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(9/11) As dependências críticas e caminhos críticos do cronograma de software do projeto são gerenciados de acordo com um procedimento documentado.

O projeto é gerenciado baseado no *Plano de Iteração*. Na atividade *Desenvolver Plano de Iteração* são descritas considerações quanto à elaboração desse plano, incluindo marcos de acompanhamento e cronograma. O RUP orienta, por exemplo, que os *Casos de Uso* mais críticos e complexos devam ser desenvolvidos nas iterações iniciais.

(10/11) Os riscos de software do projeto são identificados, avaliados, documentados, e gerenciados de acordo com um procedimento documentado.

O *Plano de Gerenciamento de Riscos* detalha como gerenciar os riscos associados ao projeto, detalhando as atividades de gerenciamento de riscos que devem ser executadas, responsabilidades assinaladas e quaisquer recursos adicionais requeridos para a atividade de gerenciamento de riscos. A *Lista de Riscos* identifica os riscos para o projeto, associando-os com as ações de contingência e suavização.

(11/11) Revisões no projeto de software são executadas periodicamente para determinar ações necessárias para ajustar o desempenho e os resultados do projeto de software às necessidades atuais e projetadas do negócio, cliente e usuários finais, quando apropriado.

Revisões de projeto são executadas ao final de cada *Iteração* (marcos menores) e ao final de cada *Fase* (marcos maiores).

Medição e Análise

(1/1) Medições são feitas e usadas para determinar a efetividade das atividades de gerenciamento integrado de software.

Na atividade *Definir Processos de Controle e Monitoramento*, são descritos indicadores e processos usados para monitorar e controlar o progresso, qualidade e riscos do projeto.

Verificação da Implementação

(1/3) As atividades para gerenciamento do projeto de software são revisadas periodicamente junto à gerência sênior.

As atividades para gerenciamento do projeto são revisadas pelo *Revisor de Projeto*, em marcos de revisão durante o ciclo de vida do projeto.

(2/3) As atividades para gerenciamento do projeto de software são revisadas com o *Gerente do Projeto* periodicamente e em determinados eventos.

O *Gerente do Projeto* faz avaliações periódicas do status do projeto e estas são documentadas na *Avaliação de Status*.

(3/3) O grupo de garantia de qualidade de software revisa e/ou audita as atividades e produtos de trabalho de gerenciamento do projeto de software e informa os resultados.

O *Plano de Garantia de Qualidade*, elaborado pelo *Gerente do Projeto* descreve as atividades que devem ser realizadas para garantir a qualidade do projeto. O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas neste plano.

Resumo da Avaliação

A macro-atividade de *Ambiente* especifica como configurar o processo de software da organização. Este processo é descrito no *Caso de Desenvolvimento,* e deve ser usado para planejar e gerenciar todo o projeto.

A atividade *Revisar o Planejamento do Projeto*, tem como objetivo revisar o *Plano de Desenvolvimento de Software* e o *Caso de Desenvolvimento*.

Os dados do repositório, *Medidas dos Projetos,* são usados nas atividades: *Monitorar Status do Projeto* e *Informar Status.* Na atividade *Planejar Fases e Iterações,* RUP propõe que os dados coletados sejam usados em estimativas futuras.

Os marcos servem para acompanhar o planejamento, custos e esforços para determinar a viabilidade do projeto.

O *Plano de Gerenciamento de Riscos* detalha atividades associadas ao gerenciamento de riscos, e a *Lista de Riscos* identifica os riscos para o projeto, associando-os com as ações de contingência e suavização

A *Autoridade de Processo de Engenharia de Software* é responsável por manter o processo de software da organização, e as verificações de implementação são realizadas pelo *Revisor de Projeto*, *Gerente de Projeto* e APES.

5.3.5 Engenharia de Produto de Software

A finalidade da Engenharia de Produto de Software é executar consistentemente um processo de engenharia bem-definido, que integra todas as atividades de engenharia de software para produzir produtos de software corretos e consistentes, efetiva e eficientemente.

Engenharia de Produtos de Software envolve executar as tarefas de engenharia para construir e manter o software, usando o processo de software definido para o projeto (o qual é descrito na área-chave de Gerenciamento Integrado de Software), métodos e ferramentas apropriadas [PAU 93].

Meta 1/2

As tarefas de engenharia de software são definidas, integradas e executadas consistentemente para produzir o software.

Meta 2/2

Produtos de trabalho de software são mantidos consistentes uns com os outros.

Características-comum

Compromisso de Executar

(1/1) O projeto segue uma política organizacional para executar as atividades de engenharia de software.

As atividades de engenharia de software são descritas no *Processo Unificado Rational*. Quando este processo é adaptado para um projeto específico, as atividades devem ser descritas no *Caso de Desenvolvimento* para o projeto.

Habilidade de Executar

(1/4) Estão disponíveis os recursos e fundos necessários para executar as tarefas de engenharia de software.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(2/4) Membros da equipe técnica de engenharia de software recebem treinamento para executar suas tarefas técnicas.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(3/4) Membros da equipe técnica de engenharia de software recebem orientação em disciplinas relacionadas à engenharia de software.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(4/4) O Gerente de Projeto e todos os gerentes de software recebem orientação nos aspectos técnicos do projeto de software.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/10) Métodos adequados e ferramentas de engenharia de software são integrados no processo de software definido do projeto.

Essa atividade faz parte da macro-atividade de *Ambiente*. O *Especialista em Ferramentas* é responsável por selecionar e adquirir ferramentas para suporte ao desenvolvimento, e por instalar e configurar as ferramentas para adaptá-las as necessidades do projeto. O *Engenheiro de Processo* é responsável por definir *Modelos* e *Guias* específicos para o projeto.

(2/10) Os requisitos de software são desenvolvidos, mantidos, documentados e verificados, analisando sistematicamente os requisitos alocados, de acordo com o processo de software definido do projeto.

Casos de Uso representam uma sistemática para obter, organizar e comunicar requisitos do usuário. Os *Casos de Uso* fornecem uma maneira de documentar os requisitos funcionais do sistema, que servem como base para o desenvolvimento do projeto, teste e planejamento de iterações. No Processo Unificado, os *casos de uso* são mantidos em um *Modelo de Casos de Uso* e devem permanecer consistentes por todo o ciclo de vida do projeto. As atividades relacionadas a requisitos são descritas na macro-atividade *Requisitos.*

As necessidades dos interessados (requisitos do sistema) são reunidas e analisadas, um documento com a *Visão do Sistema* (documento com uma visão geral dos requisitos centrais do projeto), um *Modelo de Casos de Uso* (casos de uso e descrição) e uma *Especificação Complementar* (captura requisitos não-técnicos) são desenvolvidos para descrever o que o sistema faz.

(3/10) O projeto (*design*) de software é desenvolvido, mantido, documentado e verificado, de acordo com o processo de software definido do projeto, para acomodar os requisitos de software e formar um marco de referência para a codificação.

A macro-atividade de *Análise e Projeto* descreve os procedimentos que devem ser executados para transformar os requisitos do usuário em um projeto de sistema,

desenvolver uma arquitetura robusta para o sistema e adaptar o projeto para compatibilizar ao ambiente de implementação.

(4/10) O código do software é desenvolvido, mantido, documentado e verificado, de acordo com o processo de software definido do projeto, de modo a implementar os requisitos de software e o projeto de software.

A macro-atividade de *Implementação* descreve as atividades que devem ser implementadas para a codificação do software. As atividades de implementação são baseadas nos artefatos desenvolvidos na macro-atividade de *Análise e Projeto*.

(5/10) O teste de software é executado de acordo com o processo de software definido do projeto.

O teste é executado de acordo com as atividades e procedimentos definidos na macro-atividade de *Teste*. Inicialmente é realizado um planejamento de teste, que identifica e descreve os testes que serão implementados e executados. Esta atividade é concluída quando é gerado o *Plano de Teste* que contém todos os requisitos e as estratégias para o teste.

(6/10) O teste de integração do software é planejado e executado de acordo com o processo de software definido do projeto.

Uma das atividades da macro-atividade de *Teste* é *Executar Teste de Integração*, que é planejado na atividade *Planejar Teste* e documentado no *Plano de Teste*.

(7/10) O teste de sistema e aceitação do software é planejado e executado para demonstrar que o software satisfaz seus requisitos.

O teste de sistema e aceitação é executado conforme definido na macro-atividade de *Teste* e no *Plano de Teste*.

(8/10) A documentação, que será usada para operar e manter o software, é desenvolvida e mantida de acordo com o processo de software definido do projeto.

O *Escritor Técnico* é o responsável pela documentação para o usuário final. O *Manual de Guia de Estilo* descreve como o material de suporte ao usuário final deve ser descrito e a atividade *Desenvolver Material de Suporte* descreve o procedimento a ser seguido. A documentação é mantida de acordo com as atividades da macro-atividade de *Gerenciamento de Configuração e Alteração*.

(9/10) Dados dos defeitos identificados em revisões conjuntas e testes são coletados e analisados de acordo com o processo de software definido do projeto.

Os defeitos devem ser documentados na *Solicitação de Alteração*. As *Solicitações de Alteração* são analisadas pelo *Conselho de Controle de Alteração*, que julga e encaminha a solicitação para correção.

(10/10) A consistência é mantida através dos produtos de trabalho do software, incluindo os planos de software, descrições de processo, requisitos alocados, requisitos de software, projetos de software, código, planos de teste e procedimentos de teste.

Os *artefatos* são elaborados conforme procedimentos documentados e são relacionados entre si, isto é, um complementa o outro, e um serve de entrada para a produção do outro.

Alterações são definidas pelo *Conselho de Controle de Alteração* e devem ser realizadas de forma consistente.

Medição e Análise

(1/2) Medições são feitas e usadas para determinar a funcionalidade e a qualidade dos produtos de software.

O *Plano de Garantia de Qualidade* fornece uma visibilidade de como a qualidade do processo, do produto e dos artefatos está sendo assegurada.

(2/2) Medições são feitas e usadas para determinar o status das atividades de engenharia de produto de software.

Na atividade *Definir Processos de Controle e Monitoramento*, são descritos indicadores e processos usados para monitorar e controlar o progresso, qualidade e riscos do projeto.

Verificação da Implementação

(1/3) As atividades de engenharia de produto de software são revisadas periodicamente junto à gerência sênior.

As atividades de engenharia do produto são revisadas pelo *Revisor do Projeto* periodicamente.

(2/3) As atividades de engenharia de produto de software são revisadas periodicamente e em determinados eventos com o gerente de projeto.

O *Gerente do Projeto* faz avaliações periódicas do status do projeto e estas são documentadas na *Avaliação de Status*.

(3/3) O grupo de garantia da qualidade de software revisa e/ou audita as atividades e produtos de trabalho para engenharia de produto de software e informa os resultados.

O *Plano de Garantia de Qualidade*, elaborado pelo *Gerente do Projeto* descreve as atividades que devem ser realizadas para garantir a qualidade do projeto. O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas neste plano.

Resumo da Avaliação

As atividades de engenharia de software são descritas no *Processo Unificado Rational*. Quando este processo é adaptado para um projeto específico, as atividades "customizadas" devem ser descritas no *Caso de Desenvolvimento* para o projeto.

RUP descreve seis macro-atividades de processo, que são: *Modelagem de Negócio*, *Requisitos*, *Análise e Projeto*, *Implementação*, *Teste* e *Implantação*, e três macro-atividades de suporte, que são: *Gerenciamento de Configuração e Alteração*, *Gerenciamento de Projetos* e *Ambiente*. O RUP descreve o relacionamento entre as macro-atividades, entre as atividades de cada macro-atividade, os artefatos relacionados às atividades e os trabalhadores responsáveis pelas atividades, mantendo assim, a consistência do processo.

As verificações de implementação são realizadas pelo *Revisor de Projeto*, *Gerente de Projeto* e APES.

5.3.6 Coordenação Intergrupos

A finalidade da Coordenação Intergrupos é estabelecer uma maneira do grupo de engenharia de software participar ativamente com outros grupos de engenharia, para que o projeto esteja mais capacitado a satisfazer as necessidades de seus clientes efetiva e eficientemente.

Coordenação Intergrupos envolve a participação do grupo de engenharia de software com outros grupos de engenharia do projeto para enfocar requisitos, objetivos e questões no nível de sistema. Representantes dos grupos de engenharia de projeto participam no estabelecimento dos requisitos, objetivos e planos no nível do sistema, por trabalhar com o cliente e o usuário, quando apropriado. Estes requisitos, objetivos e planos tornam-se a base para todas as atividades de engenharia [PAU 93].

Meta 1/3

Os requisitos do cliente são acordados por todos grupos afetados

Meta 2/3

Os compromissos entre os grupos de engenharia são acordados por todos os grupos afetados.

Meta 3/3

Os grupos de engenharia identificam, acompanham, e resolvem questões intergrupos.

Características-comum

Compromisso de Executar

(1/1) O projeto segue uma política organizacional para estabelecer equipes de engenharia interdisciplinares.

O RUP propõe equipes de desenvolvimento com base nos papéis descritos pelo processo e requeridos para executar as atividades propostas para o sistema, por exemplo, o CMM descreve o Grupo de Gerenciamento de Configuração, no RUP as atividades do Grupo de Gerenciamento de Configuração são executadas pelo *Gerente de Configuração*, que é um dos trabalhadores descritos pelo processo.

Habilidade de Executar

(1/5) Estão disponíveis os recursos e fundos necessários para coordenar as atividades de engenharia de software com outros grupos de engenharia.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(2/5) Ferramentas de suporte usadas por grupos de engenharia diferentes são compatíveis para possibilitar a comunicação e coordenação efetivas.

A macro-atividade de *Ambiente* se preocupa em definir ferramentas de suporte ao processo. A *Autoridade de Processo de Engenharia de Software* (APES) tem a finalidade de facilitar a troca de informação entre os participantes do processo, bem como disseminar as melhores práticas.

(3/5) Todos os gerentes na organização recebem treinamento em trabalho em equipe.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(4/5) Todos líderes de tarefas, em cada grupo de engenharia, recebem orientação nos processos, métodos e padrões usados pelos outros grupos de engenharia.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(5/5) Os membros do grupo de engenharia recebem orientação sobre como trabalhar em equipe.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/7) O grupo de engenharia de software e os outros grupos de engenharia participam com o cliente e usuários finais, conforme apropriado, para estabelecer os requisitos do sistema.

Nas reuniões de definição e revisão dos requisitos devem participar o cliente, usuário final, *Analista de Sistema*, *Especificador de Casos de Uso*, *Arquiteto de Software* e quaisquer interessados.

(2/7) Representantes do grupo de engenharia de software do projeto trabalham com representantes de outros grupos de engenharia para monitorar e coordenar atividades técnicas e resolver questões técnicas.

O *Gerente de Projeto* é o responsável por gerenciar e coordenar as atividades do projeto.

(3/7) Um plano documentado é usado para comunicar compromissos intergrupos, coordenar e acompanhar o trabalho executado.

O *Plano de Desenvolvimento de Software* documenta as atividades relacionadas ao projeto. Não está claro como são coordenadas as atividades no nível do sistema.

(4/7) Dependências críticas entre grupos de engenharia são identificadas, negociadas, e acompanhadas de acordo com um procedimento documentado.

O *Gerente de Projeto* é o responsável pelo planejamento do projeto de software, e tem a responsabilidade de gerenciar dependências críticas entre as atividades.

(5/7) Produtos de trabalho produzidos como entrada para outros grupos de engenharia são revisados por representantes do grupo receptor para assegurar que os produtos de trabalho satisfazem suas necessidades.

RUP descreve várias atividades de revisão ao longo do ciclo de vida do projeto para garantir que os artefatos satisfazem seus requisitos.

(6/7) Problemas intergrupos não resolvidos pelos representantes individuais dos grupos de engenharia de projeto são manipuladas de acordo com um procedimento documentado.

O Gerente de Projeto tem a responsabilidade de gerenciar problemas referentes a equipe de desenvolvimento, tais como: cronograma, riscos técnicos, etc. Problemas de relacionamento inter-pessoal está fora do escopo do RUP.

(7/7) Representantes do grupo de engenharia de projeto conduzem intercâmbio e revisões técnicas periódicas.

RUP descreve atividades de revisão e auditoria de artefatos durante o ciclo de vida do projeto. A *Autoridade de Processo de Engenharia de Software* tem a finalidade de facilitar a troca de informação entre os participantes do processo, bem como disseminar as melhores práticas.

Medição e Análise

(1/1) Medições são feitas e usadas para determinar o status das atividades de coordenação intergrupos.

Na atividade *Definir Processos de Controle e Monitoramento*, são descritos indicadores e processos usados para monitorar e controlar o progresso, qualidade e riscos do projeto.

Verificação da Implementação

(1/3) As atividades de coordenação intergrupos são revisadas periodicamente junto à gerência sênior.

As atividades do projeto são revisadas pelo *Revisor do Projeto* periodicamente.

(2/3) As atividades para coordenação intergrupos são revisadas periodicamente e em determinados eventos com o gerente de projeto.

O *Gerente do Projeto* faz avaliações periódicas do status do projeto e estas são documentadas na *Avaliação de Status*.

(3/3) O grupo de garantia da qualidade revisa e/ou audita as atividades e os produtos de trabalho para coordenação intergrupos e informa os resultados.

O *Plano de Garantia de Qualidade*, elaborado pelo *Gerente do Projeto* descreve as atividades que devem ser realizadas para garantir a qualidade do projeto. O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas neste plano.

Resumo da Avaliação

RUP não apresenta a separação entre o grupo de engenharia de software e outros grupos de engenharia, descreve esses grupos como trabalhadores do processo que se relacionam por meio dos artefatos que produzem e das atividades que executam.

O *Plano de Desenvolvimento de Software* descreve todas as atividades relacionadas ao projeto de software. O *Gerente de Projeto* tem a responsabilidade de gerenciar

dependências críticas entre as atividades.São realizadas atividades de revisão e auditoria ao longo do processo de desenvolvimento.

A *Autoridade de Processo de Engenharia de Software* (APES) tem a finalidade de facilitar a troca de informação entre os participantes do processo.

Não fica claro no RUP como são coordenados as atividades e compromissos no nível de sistema.

5.3.7 Revisão Conjunta

A finalidade da Revisão Conjunta é remover eficientemente os defeitos dos produtos de software o mais cedo possível e, como conseqüência, identificar melhorias para estes artefatos. Um resultado importante é desenvolver nos grupos envolvidos com o software, um melhor entendimento dos produtos de trabalho e dos defeitos que podem ser prevenidos.

Meta 1/2

Atividades de revisão conjunta são planejadas.

Meta 2/2

Defeitos nos produtos de trabalho do software são identificados e removidos.

Características-comum

Compromisso de Executar

(1/1) O projeto segue uma política organizacional para executar revisão conjunta.

O *Plano de Revisão e Auditoria,* contido no *Plano de Garantia de Qualidade,* especifica quais os artefatos que devem ser revisados, e nas macro-atividades são descritas as atividades de revisão. Por exemplo, *Revisar Modelo de Casos de Uso de Negócio,* descrita na macro-atividade de Modelagem de Negócio, *Revisar Design* na macro-atividade de Análise e Projeto, etc.

RUP apresenta um guia de como as revisões devem ser conduzidas.

Habilidade de Executar

(1/3) Estão disponíveis os recursos e fundos necessários para executar revisão conjunta em cada produto de trabalho de software que será revisado.

Essa prática-chave não é englobada pelo RUP. A proposta 1, descrita no capítulo 6, propõe uma alteração no RUP buscando satisfazer essa prática-chave.

(2/3) Líderes de revisão conjunta recebem treinamento em como liderar uma revisão conjunta.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

(3/3) Revisores, que participam de revisões conjuntas, recebem treinamento nos objetivos, princípios e métodos de revisões conjuntas.

O RUP não cobre essa prática-chave, no capítulo 6 é descrita a macro-atividade de Treinamento, que propõe uma alteração no RUP para satisfazer essa prática-chave.

Atividades Executadas

(1/3) Revisões conjuntas são planejadas e os planos são documentados.

As revisões conjuntas são planejadas e documentadas no *Plano de Garantia de Qualidade*. RUP propõe que as reuniões de revisão sejam marcadas com antecedência, para que as pessoas que participarão possam revisão o material da revisão.

(2/3) Revisões conjuntas são executadas de acordo com um procedimento documentado.

O RUP descreve o procedimento para cada atividade de revisão e guias de como as revisões devem ser realizadas.

(3/3) Dados sobre a condução e os resultados da revisão conjunta são registrados.

O resultado de cada revisão é documentado no *Registro de Revisão*.

Medição e Análise

(1/1) Medições são feitas e usadas para determinar o status das atividades de revisão conjunta.

O *Registro de Revisão*, elaborado ao final da revisão, documenta os dados sobre cada revisão, incluindo o resultado da revisão, defeitos ou problemas encontrados, tempo da revisão (em horas), participantes e pendências.

Verificação da Implementação

(1/1) O grupo de garantia da qualidade revisa e/ou audita as atividades e os produtos de trabalho das revisões conjuntas e informa os resultados.

O *Plano de Garantia de Qualidade*, elaborado pelo *Gerente do Projeto* descreve as atividades que devem ser realizadas para garantir a qualidade do projeto. O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* deva ter responsabilidade pelos aspectos de qualidade e conduza as revisões e auditorias, descritas neste plano.

Resumo da Avaliação

O *Plano de Revisão e Auditoria,* contido no *Plano de Garantia de Qualidade,* especifica o cronograma, recursos, métodos e procedimentos usados na condução de revisões e auditorias. Associados ao plano estão guias de como executar revisões.

O *Registro de Revisão*, elaborado ao final da revisão, documenta os dados sobre cada revisão, incluindo o resultado da revisão, defeitos ou problemas encontrados, tempo da revisão (em horas), participantes , etc.

O RUP recomenda que a *Autoridade de Processo de Engenharia de Software* deva ter responsabilidade pelos aspectos de qualidade do processo e conduza as revisões e auditorias no processo.

5.3.8 Mapeamento entre termos equivalentes do CMM com o RUP

A tabela abaixo associa os termos descritos pelo CMM com os termos descritos pelo RUP, visando facilitar o entendimento da avaliação, e posterior análise desses modelos.

TABELA 5.1 – Mapeamento entre os termos do CMM e os termos do RUP[8]

Termos CMM	Termos RUP
Gerente Sênior	*Revisor do Projeto*
Gerente de Projeto	*Gerente de Projeto*
Gerente de Projeto de Software	*Gerente de Projeto*
Produtos de Trabalho de Software	*Artefatos*
Plano de Desenvolvimento de Software	*Plano de Desenvolvimento de Software*
Plano de Garantia de Qualidade de Software	*Plano de Garantia de Qualidade*
Plano de Gerenciamento de Configuração de Software	*Plano de Gerenciamento de Configuração*
Documentos de Requisitos de Software	*Especificação dos Requisitos do Software*
Declaração de Trabalho	*Visão*
Processo de Software Padrão da Organização	*Processo Unificado Rational customizado para a organização*
Base de dados de processo de software da organização	*Medidas de Projeto*
Estágios	*Fases*
Grupo de Engenharia de Software	*Trabalhadores que executam atividades de engenharia de software, como: analista de sistemas, arquiteto de software, implementador, etc.*
Grupo de Processo de Engenharia de Software	*Engenheiro de Processo* *APES*
Grupo de Teste	*Projetista de Teste* *Testador*
Grupo de Garantia de Qualidade	*APES*
Grupo de Gerenciamento de Configuração de Software	*Gerente de Configuração* *Gerente de Controle de Alteração*

[8] No texto convencionou-se utilizar sublinhado para os termos do CMM e itálico para os termos do RUP.

5.4 Análise dos Resultados da Avaliação

Os resultados obtidos na avaliação estão descritos nas Tabelas 5.2 a 5.5. A Tabela 5.2 mostra a cobertura de práticas-chave do CMM pelo RUP, expressa em percentuais de práticas-chave, agrupadas por áreas-chave de processo. A Tabela 5.3 lista as práticas-chave ausentes ou incompletas. A Tabela 5.4 mostra o percentual de práticas-chave satisfeitas por característica-comum, em cada área-chave de processo. Tabela 5.5 mostra o percentual de práticas-chave suportado pelo CMM, ignorando-se a característica-comum Habilidade de Executar.

Esta característica-comum foi eliminada porque ela descreve as pré-condições que devem existir no projeto ou na organização, tais como recursos, estruturas organizacionais e treinamento, questões consideradas fora do escopo do RUP.

Por exemplo, a área-chave Gerenciamento de Requisitos recomenda doze práticas-chave que são divididas em cinco características-comum. A Tabela 5.2 mostra que dez das doze práticas-chave são suportadas. A Tabela 5.3 mostra explicitamente as práticas-chave não suportadas. A Tabela 5.4 mostra como as práticas-chave suportadas estão distribuídas nas cinco características-comum (Compromisso de Executar, Habilidade de Executar, Atividades Executadas, Medição e Análise e Verificação da Implementação). A Tabela 5.5 mostra que, eliminando as práticas-chave da característica-comum Habilidade de Executar; todas as oito práticas-chave são suportadas para esta área-chave de processo.

5.4.1 Práticas-chave Cobertas pelo RUP

A Tabela 5.2 mostra que o RUP estabelece procedimentos que satisfazem quase todas as práticas-chave descritas pelo CMM para as áreas-chave de processo Gerenciamento de Requisitos, Planejamento de Projetos de Software, Acompanhamento e Supervisão de Projetos, Gerenciamento de Configuração de Software, Definição dos Processos da Organização e Engenharia de Produto de Software.

Em geral, as práticas-chave ausentes decorrem do fato do RUP não descrever como recursos humanos adequados e fundos são disponibilizados para executar as atividades do processo, e nem assegurar que os trabalhadores são treinados para executar suas atividades.

Na macro-atividade de *Requisitos*, é descrita a seqüência de atividades que deve ser executada para gerenciar os requisitos de software e prover mudanças nesses requisitos.

As atividades relacionadas com Planejamento de Projetos de Software e Acompanhamento e Supervisão de Projetos são descritas na macro-atividade de *Gerenciamento de Projetos*.

RUP está limitado a planejamento de projetos de software, não descrevendo as relações do projeto de software com o projeto completo. Planejamento completo do projeto envolve planejar as atividades executadas por outros grupos, como infra-estrutura de rede, hardware, marketing, etc. RUP não descreve como os recursos computacionais críticos, tais como: capacidade de memória, uso do processador, capacidade do canal de comunicação, etc; são identificados, estimados e acompanhados.

Na macro-atividade de *Gerenciamento de Configuração e Alteração* são descritas as atividades relacionadas à área-chave de Gerenciamento de Configuração de Software.

TABELA 5.2 – Práticas-chave do CMM Suportadas pelo RUP por KPA

Áreas-chave de Processo	Práticas-chave		
	Total	Suportadas	%
Gerenciamento de Requisitos	12	10	83,3%
Planejamento de Projetos	25	20	80,0%
Acompanhamento e Supervisão de Projetos	24	20	83,3%
Gerenciamento de Subcontratação	22	0	0,0%
Garantia de Qualidade	17	10	58,8%
Gerenciamento de Configuração	21	18	85,7%
Foco nos Processos da Organização	16	7	43,7%
Definição dos Processos da Organização	11	9	81,8%
Programa de Treinamento	16	0	0,0%
Gerenciamento de Software Integrado	19	15	78,9%
Engenharia de Produto de Software	20	16	80,0%
Coordenação Intergrupos	17	13	76,4%
Revisão por Pares	9	6	66,6%

Definição dos Processos da Organização envolve desenvolver e manter o processo de software padrão da organização, assim como os bens relacionados ao processo, tais como: guias e critérios de adaptação do processo, base de dados e ciclo de vida [PAU 93]. RUP descreve guias para customização do Processo Unificado para o processo padrão da organização, considerando o domínio das aplicações da organização, práticas de reuso, e tecnologias utilizadas. Em muitos casos, o Processo Unificado é o próprio processo da organização. Na macro-atividade de *Ambiente* são descritos os procedimentos para customização e manutenção do processo customizado.

Engenharia de Produto descreve as tarefas de engenharia para construir e manter o software, no RUP estas atividades são descritas, ao longo das macro-atividades de processo, que são: *Modelagem de Negócio, Requisitos, Análise e Projeto, Implementação, Teste* e *Implantação.*

RUP oferece um bom suporte para as áreas-chave de Gerenciamento Integrado de Software e Coordenação Intergrupos. Nestes casos, quase 70% das práticas-chave recomendadas pelo CMM são satisfeitas.

Gerenciamento Integrado de Software descreve como usar o processo de software da organização, desenvolvido na área-chave de Definição do Processo da Organização, para criar um processo específico para um projeto. RUP descreve na macro-atividade de *Ambiente*, procedimentos para customização do processo da organização (que pode ser o próprio Processo Unificado), para um projeto específico, e como este processo específico do projeto deve ser documentado no *Caso de Desenvolvimento*.

Coordenação Intergrupos envolve estabelecer meios para os trabalhadores de um processo participarem ativamente no desenvolvimento de um sistema.

RUP oferece pouco suporte para as práticas-chave de Garantia de Qualidade de Software, Foco nos Processos da Organização e Revisão por Pares.

No RUP, versão 2001.00.03, foram incluídas atividades para garantir a qualidade de software do projeto, incluindo a elaboração de um *Plano de Garantia de Qualidade*. Mas, ainda identificam-se lacunas no processo de tratamento de problemas identificados nas revisões e auditorias e no repasse dos resultados das revisões a equipe do projeto. RUP não descreve como são realizadas medições para determinar custos e status do cronograma das atividades de QGS.

Na atividade *Avaliar a Organização*, o status atual da organização deve ser descrito, principalmente com relação ao processo atual da organização e as áreas para melhoria. Mas, no RUP não está descrito como as melhorias identificadas na avaliação do processo da organização são implementadas, incluindo a elaboração de planos de melhorias e a realização de revisões desses planos, como preconiza o CMM, por isso o baixo percentual alcançado na área-chave de processo Foco nos Processos da Organização.

O RUP suporta todas as práticas-chave descritas na característica-comum Atividades Executadas da área-chave de Revisão por Pares, mas oferece pouco suporte a TOTALIDADE das práticas desta área-chave, porque não suporta bem a característica-comum Habilidade de Executar.

As áreas-chave de processo Gerenciamento de Subcontratação e Programa de Treinamento não são suportadas pelo Processo Unificado Rational. RUP considera estas áreas-chave de processo uma responsabilidade organizacional que está acima do escopo do processo de software [RAT 2001], necessitando ser suportadas de outras maneiras na organização.

Nas análises seguintes (Tabelas 5.4 e 5.5) serão desconsideradas as áreas-chave *Gerenciamento de Subcontratação* e *Treinamento*, já que estão fora do escopo do RUP.

TABELA 5.3– Práticas-chave do CMM Ausentes ou Incompletas

Áreas-chave de Processo	Práticas-chave Ausentes ou Incompletas
Gerenciamento de Requisitos	- Recursos adequados e fundos são providos para gerenciamento dos requisitos alocados. - Membros do grupo de engenharia de software e outros grupos relacionados ao software são treinados para executar suas atividades de gerenciamento de requisitos.
Planejamento de Projetos	- Recursos adequados e fundos são providos para planejar o projeto de software. - Os gerentes de software, engenheiros de software, e outros indivíduos envolvidos no planejamento são treinados em procedimentos de planejamento e estimativas aplicáveis em suas áreas de responsabilidades. - O planejamento do projeto de software é iniciado nos estágios iniciais, e em paralelo com o planejamento completo do projeto. - O grupo de engenharia de software participa com outros grupos afetados no planejamento completo do projeto por todo ciclo de vida do projeto. - Estimativas de recursos computacionais críticos para o projeto são derivados conforme um procedimento documentado.

Acompanhamento e Supervisão de Projetos	- Estão disponíveis os recursos e fundos necessários para acompanhar o projeto de software. - Os gerentes de software são treinados em gerenciar aspectos técnicos e pessoais do projeto de software. - Gerentes de software de primeira-linha recebem orientação em aspectos técnicos do projeto de desenvolvimento de software. - O uso de recursos computacionais críticos do projeto são acompanhados, e ações corretivas são tomadas quando necessário.
Garantia de Qualidade	- Estão disponíveis os recursos e fundos necessários para executar as atividades de SQA. - Membros do grupo de SQA são treinados para executar suas atividades de SQA. - Os membros do projeto de software recebem orientação sobre o papel, as responsabilidades, e o valor do grupo de SQA. - O grupo de SQA participa na preparação e revisão do plano de desenvolvimento de software do projeto, dos padrões e dos procedimentos. - O grupo de SQA conduz revisões periódicas de suas atividades de SQA e dos seus relatórios com o grupo de SQA do cliente, quando apropriado. - Medições são feitas e usadas para determinar o custo e o status no cronograma das atividades de SQA. - Especialistas independentes do grupo de SQA periodicamente revisam as atividades e produtos de trabalho de software do grupo de SQA do projeto.
Gerenciamento de Configuração	- Estão disponíveis os recursos e fundos necessários para executar as atividades de GCS. - Membros do grupo de GCS são treinados nos objetivos, procedimentos, e métodos para executar suas atividades de GCS. - Membros do grupo de engenharia de software e outros grupos relacionados são treinados para executar suas atividades GCS.
Foco nos Processos da Organização	- A organização segue uma política organizacional para coordenar as atividades de desenvolvimento e melhoria dos processos de software através da organização. - A gerência sênior fiscaliza as atividades de desenvolvimento e melhoria do processo de software da organização. - Estão disponíveis os recursos e fundos necessários para as atividades de processo de software da organização. - Membros do grupo responsável pelas atividades de processo de software da organização recebem treinamento para executar essas atividades. - Membros do grupo de engenharia de software e outros grupos relacionados ao software recebem orientação nas atividades de processo de software da organização e em seus papéis nessas atividades. - A organização desenvolve e mantém um plano para as atividades de desenvolvimento e melhoria de seus processos de software. - Treinamento nos processos de software da organização e nos projetos é coordenado através da organização. - Medições são feitas e usadas para determinar o status das atividades de desenvolvimento e melhoria dos processos da organização. - As atividades para desenvolvimento e melhoria dos processos de software da organização são revisadas periodicamente junto à gerência sênior.
Definição dos Processos da Organização	- Estão disponíveis os recursos e fundos necessários para desenvolver e manter o processo de software padrão da organização e bens relacionados ao processo. - Os indivíduos que desenvolvem e mantém o processo de software padrão da organização e os bens relacionados ao processo recebem treinamento para executar estas atividades.

Gerenciamento de Software Integrado	- Estão disponíveis os recursos e fundos necessários para gerenciar o projeto de software usando o processo de software definido para o projeto. - Os indivíduos responsáveis por desenvolver o processo de software definido recebem treinamento em como adaptar o processo de software padrão e como usar os bens de processo relacionados. - Os gerentes de software recebem treinamentos para gerenciar os aspectos técnicos, administrativos e de pessoal do projeto de software, baseado no processo de software definido do projeto. - Os recursos computacionais críticos do projeto são gerenciados de acordo com um procedimento documentado.
Engenharia de Produto de Software	- Estão disponíveis os recursos e fundos necessários para executar as tarefas de engenharia de software. - Membros da equipe técnica de engenharia de software recebem treinamento para executar suas tarefas técnicas. - Membros da equipe técnica de engenharia de software recebem orientação em disciplinas relacionadas à engenharia de software. - O Gerente de Projeto e todos os gerentes de software recebem orientação nos aspectos técnicos do projeto de software.
Coordenação Intergrupos	- Estão disponíveis os recursos e fundos necessários para coordenar as atividades de engenharia de software com outros grupos de engenharia. - Todos os gerentes na organização recebem treinamento em trabalho em equipe. - Todos líderes de tarefas, em cada grupo de engenharia, recebem orientação nos processos, métodos e padrões usados pelos outros grupos de engenharia. - Os membros do grupo de engenharia recebem orientação sobre como trabalhar em equipe.
Revisão por Pares	- Estão disponíveis os recursos e fundos necessários para executar revisão conjunta em cada produto de trabalho de software que será revisado. - Líderes de revisão conjunta recebem treinamento em como liderar uma revisão conjunta. - Revisores, que participam de revisões conjuntas, recebem treinamento nos objetivos, princípios e métodos de revisões conjuntas.

5.4.2 Práticas-chave Organizadas por Características-Comum

Na Tabela 5.4, as práticas-chave são mostradas organizadas por características-comum em cada área-chave de processo. O objetivo é identificar o percentual de práticas-chave suportado por característica-comum em cada área-chave de processo.

As características-comum são Compromisso de Executar (CE), Habilidade para Executar (HE), Atividades Executadas (AE), Medição e Análise (MA) e Verificação da Implementação (VI).

Compromisso de Executar (CE) envolve a definição de políticas organizacionais [PAU 93]. Habilidade para Executar (HE) envolve definir recursos e fundos, estruturas organizacionais e treinamento. Atividades Executadas (AE) envolve estabelecer planos e procedimentos, executar o trabalho, acompanhá-lo, e tomar ações corretivas quando necessário. Medição e Análise (MA) envolve definir práticas básicas de medição. *Verificação da implementação* (VI) abrange revisões e auditorias para garantir que o processo está sendo executado corretamente. As características-comuns estão descritas em maiores detalhes no capítulo 4.

A Tabela 5.4 mostra o baixo percentual de suporte alcançado pelo RUP na característica-comum Habilidade para Executar. Esta característica-comum envolve recursos, estrutura organizacional e treinamento.

A Tabela 5.4 destaca que o RUP foca nos processos de construção de software e gerenciamento de projetos de software, sem se preocupar com aspectos relacionados a gerenciamento de sistemas, como gerenciamento de custo, treinamento, gerenciamento de recursos humanos, gerenciamento de comunicação e gerenciamento de aquisição, isto é, as pré-condições descritas na característica-comum Habilidade para Executar. Isto é muito natural e perfeitamente compreensível, porque RUP evoluiu da unificação de metodologias para desenvolvimento de software [BOO 94, JAC 92, RUM 91], e não de processos de gerenciamento de projetos. Mas, apesar disto, os índices mostram a necessidade de examinar cuidadosamente as práticas ausentes.

Na macro-atividade de *Gerenciamento de Projetos*, está descrito que a finalidade do RUP é fornecer um *framework* para gerenciamento intensivo de projetos de software, prover guias para planejamento, recrutamento, e monitoramento de projetos, e fornecer um *framework* para gerenciamento de riscos [RAT 2001]. Mas, RUP não pretende cobrir todos os aspectos de gerenciamento de projetos, e não cobre questões como gerenciamento de pessoas, cronogramas e contratos [RAT 2001].

Procedimentos para satisfazer estas questões estão fora do escopo do RUP e devem ser definidas pela organização [RAT 2001]. A organização deve usar uma abordagem específica para gerenciamento de projetos para complementar o RUP; como o *Project Management Body of Knowledge* (PMBOK) [PMI 2000], proposto pelo Project Management Institute [PMI 2000]. O PMBOK identifica áreas de conhecimento específicas para tratar gerência de recursos humanos, gerência de aquisição, gerência de integração, e outras questões de gerência.

TABELA 5.4 – Percentual de Práticas-chave Suportadas por Características-comuns

Key Process Areas	**Características-comum**				
	CE	**HE**	**AE**	**MA**	**VI**
Gerenciamento de Requisitos	100,0%	50,0%	100,0%	100,0%	100,0%
Planejamento de Projetos	100,0%	50,0%	80,0%	100,0%	100,0%
Acompanhamento e Supervisão de Projetos	100,0%	40,0%	92,3%	100,0%	100,0%
Garantia de Qualidade	100,0%	25,0%	75,0%	0,0%	66,6%
Gerenciamento de Configuração	100,0%	40,0%	100,0%	100,0%	100,0%
Foco nos Processos da Organização	33,3%	25,0%	71,4%	0,0%	0,0%
Definição dos Processos da Organização	100,0%	0,0%	100,0%	100,0%	100,0%
Gerenciamento de Software Integrado	100,0%	0,0%	90,9%	100,0%	100,0%
Engenharia de Produto de Software	100,0%	0,0%	100,0%	100,0%	100,0%
Coordenação Intergrupos	100,0%	20,0%	100,0%	100,0%	100,0%
Revisão por Pares	100,0%	0,0%	100,0%	100,0%	100,0%

5.4.3 Práticas-chave Cobertas pelo RUP Eliminando as Práticas-chave da Característica-comum Habilidade para Executar

A Tabela 5.5 mostra o percentual de práticas-chave do CMM atendido pelo RUP em cada área-chave de processo, eliminando-se as práticas descritas na característica-comum Habilidade de Executar. Essa característica-comum foi eliminada porque as pré-condições organizacionais estão fora do escopo do RUP.

Analisando esta tabela, verifica-se que excluindo as práticas-chave do CMM descritas na característica-comum Habilidade de Executar, os percentuais suportados pelo RUP em cada área-chave de processo aumentam substancialmente, atingindo 100% em várias áreas-chave de processo.

A área-chave Revisão por Pares, que obteve um percentual de suporte de 66,6% na avaliação da Tabela 5.2, teve seu percentual elevado para 100%. Isto decorre do fato de serem descritas apenas 9 práticas-chave nessa área-chave de processo, e três se referirem a pré-condições organizacionais, que não são consideradas no RUP.

TABELA 5.5 – Percentuais Suportados Eliminando Questões não Tratadas pelo RUP.

Áreas-chave de Processo	**Práticas-chave**		
	Total	**Suportadas**	**%**
Gerenciamento de Requisitos	8	8	100,0%
Planejamento de Projetos	21	18	85,7%
Acompanhamento e Supervisão de Projetos	19	18	94,7%
Garantia de Qualidade	13	9	69,2%
Gerenciamento de Configuração	16	16	100,0%
Foco nos Processos da Organização	12	6	50,0%
Definição dos Processos da Organização	9	9	100,0%
Gerenciamento de Software Integrado	16	15	93,7%
Engenharia de Produto de Software	16	16	100,0%
Coordenação Intergrupos	12	12	100,0%
Revisão por Pares	6	6	100,0%

5.4.4 Melhorias da Avaliação do RUP

Este estudo começou no início de 2000, e foi então primeiramente conduzido na versão disponível do RUP (Versão 5.5), publicada em [RAT 99]. A avaliação mostrou alguns resultados pobres, como exemplificado na Tabela 5.6. Os projetistas do RUP fizeram um excelente trabalho como pode ser visto comparando a Tabela 5.6 com a Tabela 5.2. Em todas as áreas-chave de processo os percentuais de práticas-chave suportadas foram aumentados, e principalmente na área chave de processo Garantia de Qualidade de Software.

As melhorias decorrem, principalmente, da definição de procedimentos para garantir a qualidade de software, incluindo a elaboração do Plano de Garantia de Qualidade.

Com a inclusão das atividades de garantia de qualidade, RUP passa a satisfazer as práticas-chave da característica-comum Verificação da Implementação, que preconizam a realização de revisão e/ou auditoria para garantir a qualidade do software desenvolvido, incluída em quase todas as áreas-chave de processo.

O Processo Unificado recomenda que a Autoridade de Engenharia de Software (APES) deve ter responsabilidade pelos aspectos de qualidade do processo, e execute as revisões e auditorias [RAT 2001]. APES cumpre o papel do Grupo de Garantia de Qualidade descrito no CMM.

TABELA 5.6 - CMM Key Practices Coverage for RUP 5.5 by Key Process Area

Key Process Areas	Áreas-chave de Processo		
	Total	Suportadas	%
Gerenciamento de Requisitos	12	9	75,0%
Planejamento de Projetos	25	19	76,0%
Acompanhamento e Supervisão de Projetos	24	19	79,2%
Gerenciamento de Subcontratação	22	0	0,0%
Garantia de Qualidade	17	6	35,3%
Gerenciamento de Configuração	21	15	71,4%
Foco nos Processos da Organização	16	7	43,8%
Definição dos Processos da Organização	11	8	72,7%
Treinamento	16	0	0,0%
Gerenciamento de Software Integrado	19	14	73,7%
Engenharia de Produto de Software	20	15	75,0%
Coordenação Intergrupos	17	12	70,6%
Revisão por Pares	9	5	55,6%

No próximo capítulo estão descritas as propostas de extensão do Processo Unificado Rational para satisfazer as áreas-chave do CMM.

6 Extensão do Processo Unificado Rational

Um dos objetivos deste trabalho é propor alterações no *Processo Unificado Rational* para satisfazer as práticas-chave do modelo de avaliação de processo *Capability Maturity Model*. As propostas descritas por este trabalho levam em consideração as características do *Processo Unificado Rational*, e se dividem basicamente em quatro tipos de propostas, que são caracterizadas por inclusões e/ou alterações de atividades, inclusões e/ou alterações de artefatos, inclusões de trabalhadores e inclusões de macro-atividades.

A inclusão de atividades tem o objetivo de incluir mais alguma atividade nas macro-atividades do Processo Unificado, obedecendo as demais características da macro-atividade onde a atividade está sendo inserida, como os planos desenvolvidos pela macro-atividade, os trabalhadores que executam as atividades e a seqüência das atividades. A alteração se refere à inclusão de uma sub-atividade, ou novo passo, em uma atividade já existente. Como no caso de inclusão, também respeita as características da atividade.

A inclusão de artefatos tem o objetivo de desenvolver um artefato não englobado pelo RUP, mas que é necessário para satisfazer alguma prática-chave do CMM. Alterações nos artefatos significa modificar um artefato descrito no RUP para englobar alguma característica descrita no CMM. Quando as propostas são formuladas procura-se identificar o artefato onde a informação pode ser documentada, devido as suas características, como: escopo, informações que estão documentadas, trabalhador que desenvolve, etc. Caso este artefato seja localizado no modelo, então é proposta uma alteração no artefato, senão é proposta uma inclusão de um novo artefato.

Propostas do tipo inclusão de trabalhadores, têm o objetivo de definir um novo papel que aparece no CMM e que o Processo Unificado não apresenta.

Inclusões de macro-atividades se referem a áreas-chave inteiras do CMM não satisfeitas pelo RUP, como é o caso de Gerência de Contratos e Treinamento. Nestas macro-atividades é proposto um fluxo de atividades e a descrição suscinta de cada atividade, juntamente com os trabalhadores responsáveis pela atividade e os artefatos que devem ser gerados.

As propostas descritas neste trabalho têm como objetivo esboçar como a prática-chave pode ser alcançada, mas não é o objetivo apresentar templates e guias, como o Processo Unificado apresenta.

A seção 6.1 descreve as propostas que visam atingir o nível 2 do CMM, e as propostas descritas na seção 6.2 visam atingir o nível 3 do CMM.

6.1 Propostas para Alcançar o Nível 2 do CMM

As propostas descritas nesta seção são necessárias a uma organização que utilize o Processo Unificado e deseje atingir o nível 2 do CMM.

Em algumas propostas são descritas referências bibliográficas relacionadas a área-chave que a proposta se destina a estender, visando facilitar a implantação da proposta e inclusive fornecer subsídios para a melhoria da proposta elaborada.

6.1.1 Proposta 1: Estimar Recursos e Fundos

Em [AUS 96, BAS 94, CAR 92, GRA 92] podem ser encontradas mais informações relacionadas a medições e estimativas.

Finalidade:

Incluir nos planos desenvolvidos durante a execução do processo de desenvolvimento, informações quanto aos recursos adequados e fundos necessários para a execução das atividades descritas pelo plano, como mostra a Figura 6.1.

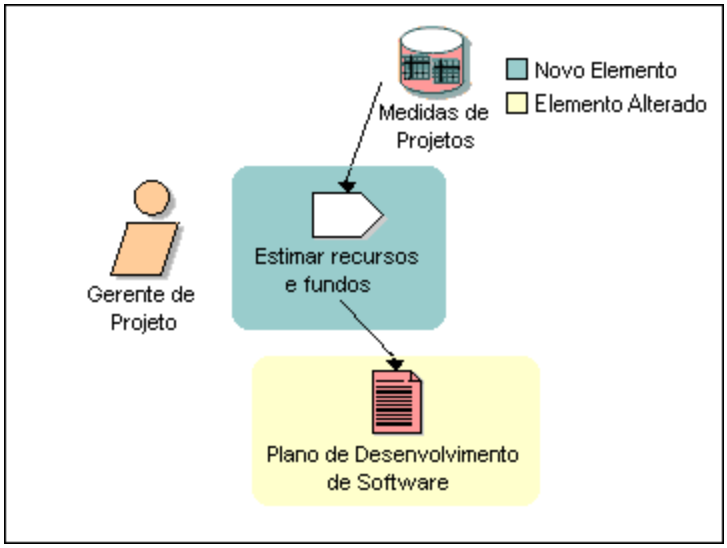


FIGURA 6.1 – Estimar Recursos e Fundos

Como exemplo de planos que devem sofrer essa alteração pode-se citar o Plano de Desenvolvimento de Software e seus sub-planos tais como: o Plano de Gerenciamento de Requisitos, o Plano de Iteração, o Plano de Gerenciamento de Configuração, o Plano de Garantia de Qualidade, o Plano de Melhoria de Processo, etc.

Passos:

- No item recursos adequados é necessário descrever qual a experiência e especialidade esperada dos trabalhadores que desenvolverão as atividades descritas no plano.
- No item fundos necessários devem ser descritos o gasto em ferramentas para a apoiar cada atividade do plano, como: planilhas eletrônicas, no Plano de Desenvolvimento de Software, ferramentas para gerenciamento de configuração no Plano de Gerenciamento de Configuração, ferramentas para gerenciamento de testes no Plano de Testes, ferramentas de auditoria no Plano de Garantia de Qualidade, etc.

Artefatos de Entrada: Dados históricos de projetos similares e o Plano que será estimado

Artefatos de Saída: Plano de Desenvolvimento de Software

Freqüência: A cada alteração de atividades no plano de desenvolvimento de software ou seus sub-planos

Trabalhador: Gerente de Projeto

6.1.2 Proposta 2 – Estimar Recursos Computacionais Críticos

A proposta 2 tem o objetivo de definir procedimentos para estimar e monitorar recursos computacionais críticos, como mostra a Figura 6.2. A seguir serão descritos esses procedimentos.

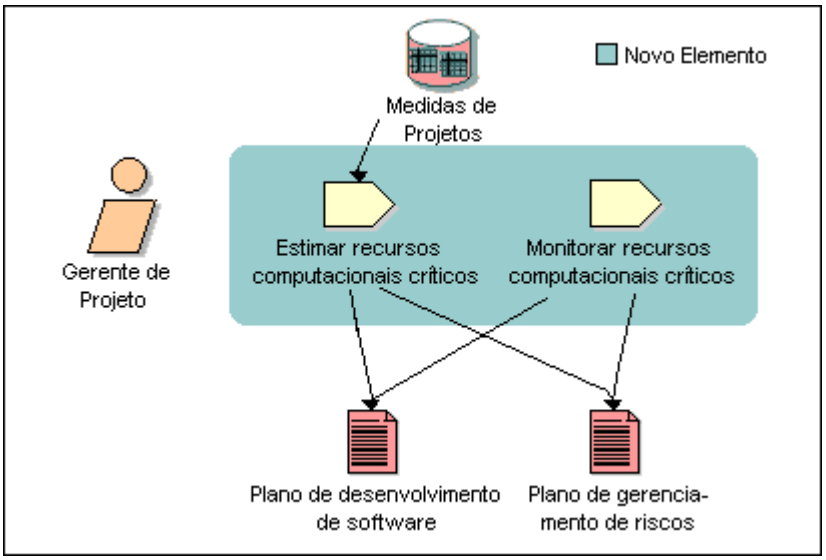


FIGURA 6.2 – Estimar Recursos Computacionais Críticos

Atividade: Estimar Recursos Computacionais Críticos

Finalidade:

Prover um procedimento para estimar recursos computacionais críticos (capacidade de memória, processador do computador, capacidade do canal de comunicação, etc.) necessários ao projeto.

A identificação e estimativa de recursos computacionais críticos devem ser realizadas com base em dados históricos de projetos similares.

As informações referentes aos recursos de computador necessários devem ser documentadas no plano de desenvolvimento de software.

Riscos associados com recursos computacionais críticos devem ser documentados no plano de gerenciamento de riscos, para assegurar que eles serão monitorados e controlados. É importante identificar na lista de riscos, ações que podem ser tomadas no caso de problemas com os recursos de computador (plano de contingência).

Passos:

- Estimar recursos computacionais críticos com base em dados históricos.

- Documentar os recursos computacionais críticos no Plano de Desenvolvimento de Software.
- Identificar os riscos associados com os recursos computacionais críticos e associar ações de contingência aos riscos identificados.

Artefatos de Entrada: Dados históricos de projetos similares

Artefatos de Saída: Plano de desenvolvimento de software

Freqüência: início do projeto e a revisto a cada iteração

Trabalhador: Gerente de Projeto

Atividade: Monitorar Recursos Computacionais Críticos

Finalidade:

Acompanhar os recursos computacionais críticos contra o planejado e tomar ações corretivas caso seja necessário.

Passos:

- Os recursos computacionais críticos devem ser acompanhados junto com as demais atividades do projeto, nos marcos menores e marcos maiores, e caso riscos sejam identificados tomar ações descritas no plano de gerenciamento de riscos.

Artefatos de Entrada: Plano de Desenvolvimento de Software, Plano de Gerenciamento de Riscos.

Artefatos de Saída: Plano de Desenvolvimento de Software, Plano de Gerenciamento de Riscos.

Freqüência: nos marcos maiores e nos marcos menores

Trabalhador: Gerente de Projeto

6.1.3 Proposta 3: Garantia de Qualidade de Software

As atividades descritas nessa proposta consideram que as atividades de garantia de qualidade são de responsabilidade da Autoridade de Processo de Engenharia de software, como propõe o RUP. A Figura 6.3 mostra as atividades propostas para Garantia de Qualidade de Software.

Em [BUS 84, BUC 87] podem ser encontradas mais informações relacionadas a garantia de qualidade de software.

Atividade: Informar Resultado das Auditorias

Finalidade:

Autoridade de Processo de Engenharia de software deve dar visibilidade ao Grupo de Engenharia de Software dos resultados das auditorias.

Passos:

- Informar ao Grupo de Engenharia de Software os resultados das auditorias através de um Relatório de Auditoria.

Artefatos de Entrada: Registro de Revisão

Artefatos de Saída: Relatório de Auditoria

Freqüência: periodicamente (após cada auditoria)

Trabalhador: Autoridade de Processo de Engenharia de software

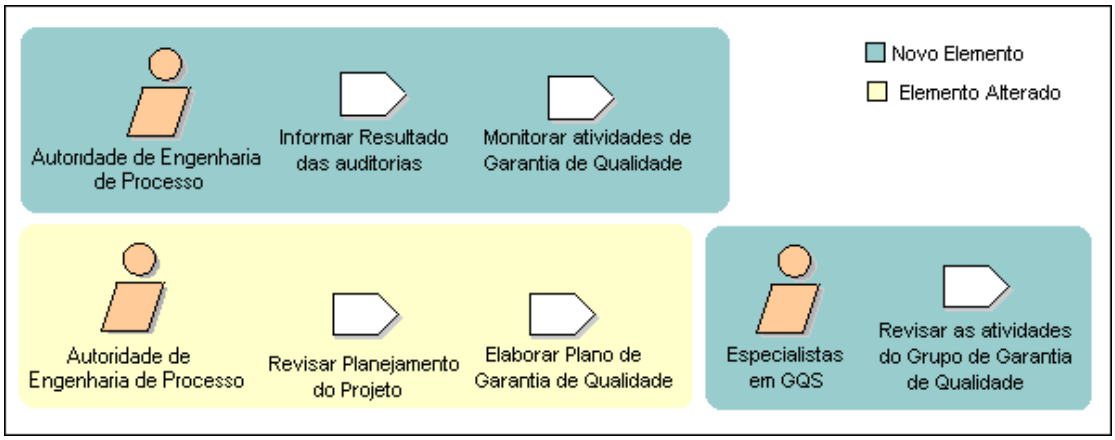


FIGURA 6.3 – Atividades de Garantia de Qualidade

Atividade: Monitorar atividades de Garantia de Qualidade de Software

Finalidade:

Acompanhar quantitativamente o progresso e o custo das atividades de garantia de qualidade com o planejado e documentado no Plano de Garantia de Qualidade. Manter o histórico dessas medições.

Passos:

- O status das atividades de GQS e o custo dessas atividades devem ser acompanhados nos marcos maiores e nos marcos menores do projeto.
- As medições devem ser armazenadas na base de dados históricos.

Artefatos de Entrada: Plano de Garantia de Qualidade

Artefatos de Saída: Plano de Garantia de Qualidade

Freqüência: marcos maiores e menores do projeto

Trabalhador: Autoridade de Processo de Engenharia de software

Atividade: Revisar as atividades do Grupo de GQS

Finalidade:

Especialistas em GQS devem revisar as atividades e os artefatos gerados pelo Grupo de GQS do projeto. Estes especialistas podem ser membros da Autoridade de Processo de Engenharia de software desde que não estejam vinculados as atividades de garantia de qualidade do projeto.

Passos:

- Revisar as atividades executadas e os artefatos gerados pela APES do projeto.
- Documentar o resultado da revisão no Registro de Revisão.
- Informar o resultado da revisão aos membros da APES do projeto e ao Gerente de Projeto.

Artefatos de Entrada: Plano de Garantia de Qualidade

Artefatos de Saída: Registro de revisão

Freqüência: periodicamente

Trabalhador: APES

Atividade: Revisar o Planejamento do Projeto

Finalidade:

A atividade Revisar o Planejamento do Projeto existente na macro-atividade de Gerenciamento de Projetos deve ser alterada, para que APES participe junto com o Revisor do Projeto desta atividade.

Passos:

- Planejar a reunião de revisão do planejamento de projeto.
- Distribuir material da reunião.
- Conduzir a reunião de revisão do planejamento do projeto.
- Registrar a decisão da reunião.

Artefatos de Entrada: Visão, Lista de Riscos, Caso de Negócio, Plano de Desenvolvimento de Software

Artefatos de Saída: Registro de revisão

Freqüência: no início do projeto e sempre que o Plano de Desenvolvimento de Software for alterado

Trabalhador: Revisor do Projeto e APES

Atividade: Elaborar Plano de Garantia de Qualidade de Software

O RUP descreve a atividade Desenvolver Plano de Garantia de Qualidade de Software como uma atribuição do Gerente do Projeto. É necessário alterar essa atividade, para que a APES gere esse Plano de Garantia de Qualidade, já que a garantia de qualidade de software é de sua responsabilidade.

Finalidade:

Passos:

- Assegurar que os objetivos de qualidade estão definidos para o projeto.
- Definir cargos e responsabilidade para garantia de qualidade.
- Sincronizar o cronograma com os desenvolvedores dos planos referenciados.
- Definir tarefas e cronograma para garantia de qualidade.

Artefatos de Entrada: Visão, Lista de Riscos, Caso de Negócio, Plano de Medições, Plano de Desenvolvimento de Software, Plano de Resolução de Problemas, Plano de Teste.

Artefatos de Saída: Plano de Garantia de Qualidade

Freqüência: no início do projeto

Trabalhador: Autoridade de Processo de Engenharia de software

6.1.4 Proposta 4 - Macro-atividade de Gerenciamento de Subcontratação de Software

O Processo Unificado não apresenta uma definição quanto ao gerenciamento de subcontratação, apenas cita no plano de desenvolvimento a existência de um plano de gerenciamento de subcontratação, como sendo um plano de processo de apoio.

Para compatibilizar o RUP com o CMM, é necessário formalizar o processo de gerenciamento de subcontratação.

Este trabalho propõe a inclusão de uma nova macro-atividade de apoio ao Processo Unificado, chamada de Gerenciamento de Subcontratação. Esta macro-atividade é descrita por meio de um diagrama de atividades, como mostra a Figura 6.4.

Em [FER 96, NIE 96] podem ser encontradas mais informações relacionadas a subcontratação.

A seguir serão descritas as atividades propostas na macro-atividade de gerenciamento de subcontratação:

1. Desenvolver um plano de gerenciamento de subcontratação (PGS)

Finalidade:

- Descrever todas as atividades relacionadas ao gerenciamento de subcontratação.
- Estabelecer procedimentos formais de comunicação e documentar no PGS.
- Definir a política utilizada para gerenciamento de subcontratação, identificando como as atividades devem ser planejadas, implementadas, controladas e organizadas.

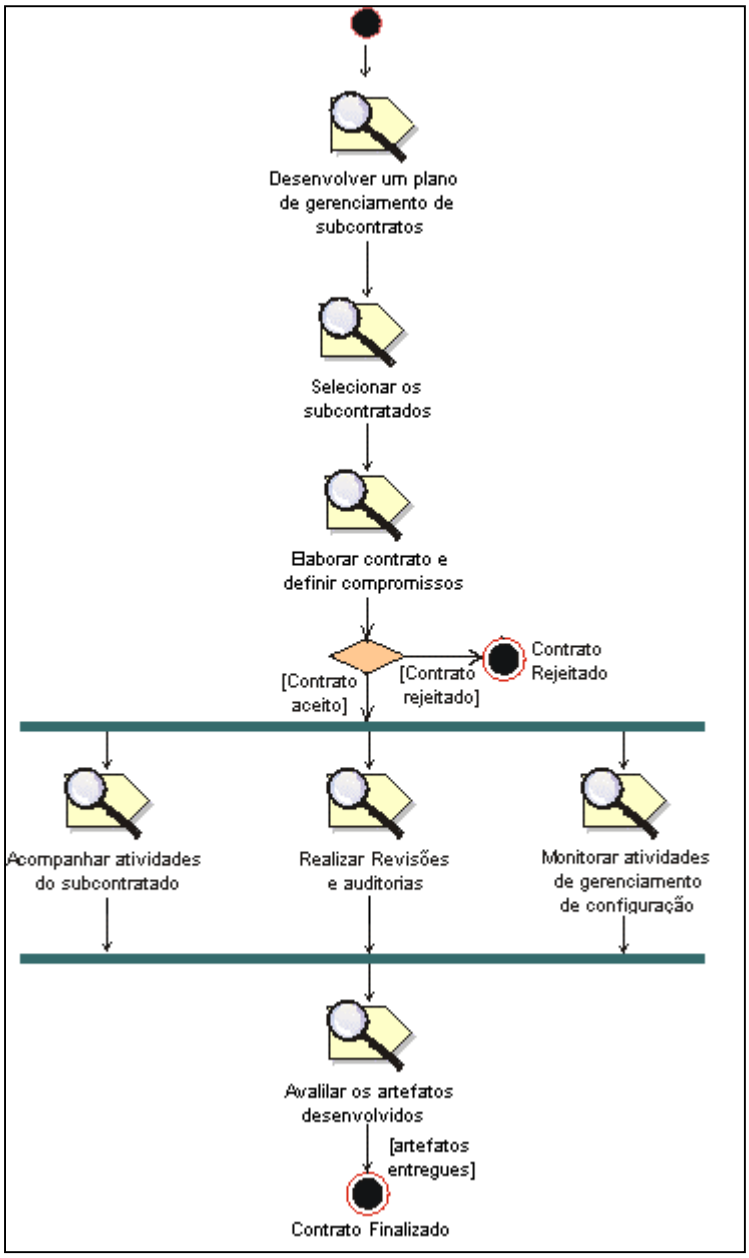


FIGURA 6.4 – Macro-atividade de Gerenciamento de Subcontratação

Passos:

- Escrever o plano de gerenciamento de subcontratação.
- Revisar e aprovar o plano.
- Manter o plano durante o ciclo de vida do projeto.

Artefatos de Entrada: plano de desenvolvimento de software, caso de negócio e caso de desenvolvimento.

Artefatos Resultantes: plano de gerenciamento de subcontratação

Freqüência: escrito no início da fase de elaboração e mantido durante todo o projeto.

Trabalhador: gerente de subcontratação

2. Selecionar Subcontratadas

Finalidade:

- ✓ Definir a qualificação necessária à subcontratada, baseando-se no plano de gerenciamento de subcontratação do projeto.
- ✓ Recrutar subcontratadas.

Passos:

- Definir habilidades (requisitos) necessárias às subcontratadas.
- Revisar os requisitos com o gerente de projeto.
- Recrutar as subcontratadas.

Artefatos de Entrada: plano de gerenciamento de subcontratação

Artefatos Resultantes: plano de gerenciamento de subcontratação (atualizado)

Freqüência: no início do projeto e conforme necessidade

Trabalhador: gerente de subcontratação e gerente de projeto.

3. Elaborar contratos definindo compromissos

Finalidade:

Definir explicitamente os compromissos das subcontratadas e do contratante.

Passos:

- Definir os compromissos de cada uma das partes, contratante e subcontratada.
- Revisar contrato.
- Assinar contrato.

Artefatos de Entrada: plano de gerenciamento de subcontratação

Artefatos Resultantes: contrato

Freqüência: a cada nova contratação

Trabalhador: gerente de subcontratação

O acompanhamento das atividades da subcontratada, realização de revisões e auditorias, monitoramento das atividades de gerenciamento de configuração ocorrem como definido no Processo Unificado Rational, não importando se o trabalho é realizado por empresar subcontratadas ou pela organização. As atividades a serem realizadas, progresso, datas de entrega são documentadas no plano de desenvolvimento de software.

4. Acompanhar atividades da subcontratada

O acompanhamento das atividades da subcontratada é realizado conforme definido no macro-atividade de gerenciamento de projetos.

5. Realizar revisões e auditorias

As auditorias, marcos de revisão, acompanhamento de riscos são realizadas conforme procedimentos padronizados para o projeto e definidos no plano de desenvolvimento de software.

6. Monitorar as atividades de gerenciamento de configuração

O gerenciamento de configuração para as subcontratadas se dá conforme definição do macro-atividade de gerenciamento de configuração.

7. Avaliar os artefatos desenvolvidos

Finalidade:

Verificar se os artefatos desenvolvidos pelas subcontratadas estão de acordo com o Plano de Desenvolvimento de Software.

Passos:

- Analisar os artefatos desenvolvidos de acordo com os planos.
- Realizar testes de aceitação e registrar na avaliação do status.
- Realizar revisão final dos artefatos pela Autoridade de Revisão do Projeto e o grupo de garantia de qualidade.

Artefatos de Entrada: plano de desenvolvimento do software

Artefatos Resultantes: avaliação do status

Freqüência: na entrega dos artefatos

Trabalhador: gerente de projeto

6.2 Propostas para Alcançar o Nível 3 do CMM

As propostas descritas nesta seção são necessárias a uma organização que utilize o Processo Unificado, esteja no nível 2 do CMM e deseje atingir o nível 3 do CMM.

6.2.1 Proposta 5 - Macro-atividade de Treinamentos

O processo Unificado Rational não cobre essa área-chave de processo do CMM. Em muitos de seus planos, como: plano de garantia de qualidade, plano de gerenciamento de

configuração e plano de desenvolvimento de software, é citado os treinamentos necessários a equipe de desenvolvimento, mas sem descrever procedimentos e gerenciamento dos treinamentos.

Em [ARG 91, GAR 93, WIG 90] podem ser encontradas mais informações relacionadas a treinamento.

A seguir será descrita uma macro-atividade que visa suprir incompatibilidade do RUP com o CMM.

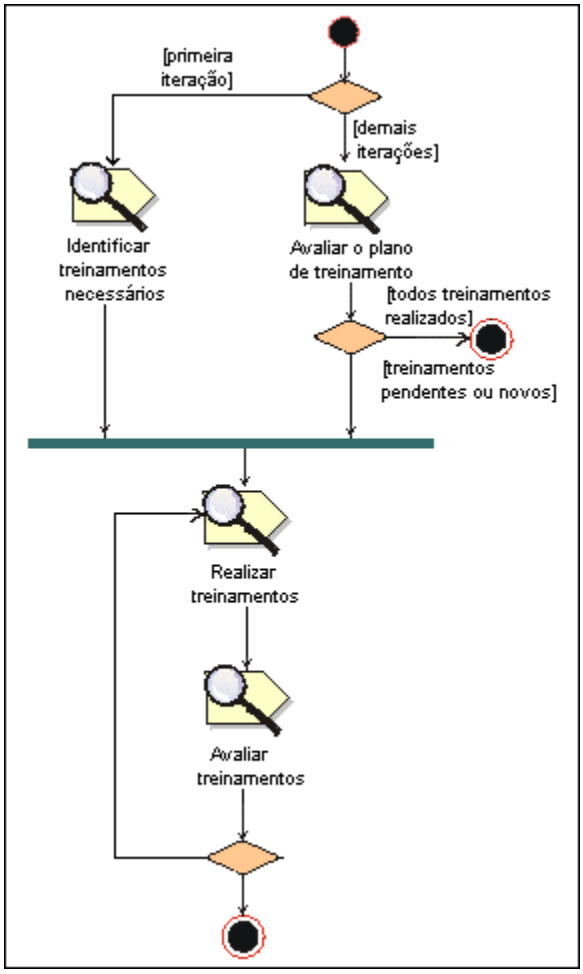


FIGURA 6.5 – Macro-atividade de Treinamentos

1. Identificar treinamentos necessários

Finalidade:

Identificar os treinamentos necessários à organização e a um projeto em específico. Considerar os seguintes aspectos: tipo de treinamento (técnico ou gerencial), pessoas a serem treinadas (grupo de engenharia de software, outros grupos relacionados).

Passos:

- Definir treinamentos necessários, através de um procedimento padronizado.

- Revisar os treinamentos necessários com o revisor do projeto
- Desenvolver o plano de treinamentos para o projeto, documentando os treinamentos necessários, as pessoas que serão treinadas e as prioridades.
- Revisar o plano com o revisor do projeto.

Artefatos de Entrada: plano de desenvolvimento de software

Artefatos Resultantes: plano de treinamento

Freqüência: início do projeto e intervalos de tempo definidos (semestral, trimestral, anual,...).

Trabalhador: gerente de projeto.

Observação: na definição dos treinamentos a serem realizados é necessário considerar as atividades técnicas e demais atividades envolvidas no processo de desenvolvimento, como: planejamento de projetos, atividades de prevenção de defeitos, gerenciamento da qualidade de software, gerenciamento quantitativo de processo, etc.

2. Avaliar o plano de treinamento

Finalidade:

Revisar o plano de treinamento. Identificar se todos os treinamentos presentes no plano de treinamento foram realizados e se a equipe identifica necessidade de um novo treinamento.

Passos:

- Revisar o plano de treinamento com revisor do projeto.
- Identificar necessidades da equipe.
- Atualizar o plano de treinamentos.
- Revisar o plano com o revisor do projeto.

Artefatos de Entrada: plano de treinamento

Artefatos Resultantes: plano de treinamento (atualizado)

Freqüência: início das iterações, exceto a primeira.

Trabalhador: gerente de projeto.

3. Realizar treinamento

Finalidade:

Realizar treinamento identificado e priorizado pelo plano de treinamento.

Passos:

- Identificar a possibilidade de realização de treinamento interno, isto é, verificar se na organização existem pessoas capacitadas para realizar o treinamento

requerido. Caso não seja possível realizar treinamento interno, identificar as opções de mercado.

- Verificar recursos financeiros necessários e disponíveis.
- Planejar as atividades do treinamento.
- Realizar treinamento.
- Registrar dados do treinamento.

Artefatos de Entrada: plano de treinamento

Artefatos Resultantes: plano de treinamento (atualizado)

Freqüência: conforme necessidades de treinamento

Trabalhador: gerente de projeto.

4. Avaliar treinamentos realizados

Finalidade:

Avaliar os treinamentos realizados, para que se tenham informações sofre a eficiência e a qualidade do treinamento.

Passos:

- Avaliar os treinandos.
- Avaliar o instrutor.
- Acompanhar as pessoas treinadas para avaliar se o treinamento cumpriu sua finalidade.

Artefatos de Entrada: plano de treinamento

Artefatos Resultantes: plano de treinamento (atualizado)

Freqüência: início do projeto

Trabalhador: gerente de projeto

6.2.2 Proposta 6: Melhorar o Processo de Software da Organização

Em [GRA 92, GRA 97] podem ser encontradas mais informações relacionadas a melhoria de processo de software da organização.

Na macro-atividade de ambiente é realizada a atividade avaliar a organização atual. Esta atividade tem como objetivo avaliar o status atual da organização em termos de seus processos, ferramentas, competências e atitudes de pessoas, clientes, tendências técnicas, problemas e áreas de melhoria.

Para cumprir as práticas descritas pelo CMM quanto a foco nos processos da organização, é necessário inserir mais algumas atividades, conforme Figura 6.6.

Foram incluídas as atividades revisar a avaliação do processo, elaborar plano de melhoria do processo de desenvolvimento e revisar plano de melhorias do Processo, e o artefato Plano de Melhorias de Processo.

Atividade: Revisar Avaliação do Processo

Finalidade:

A atividade revisar a avaliação do processo tem como objetivo revisar a Avaliação da Organização de Desenvolvimento realizada pelo engenheiro de processo.

Passos:

- Planejar a reunião de revisão da Avaliação do Processo.
- Distribuir material da reunião.
- Conduzir a reunião de revisão da Avaliação do Processo.
- Registrar a decisão da reunião.

Artefatos de Entrada: Avaliação da organização de desenvolvimento

Artefatos de Saída: Registro de Revisão

Freqüência: no final do projeto

Trabalhador: Autoridade de Engenharia de Software

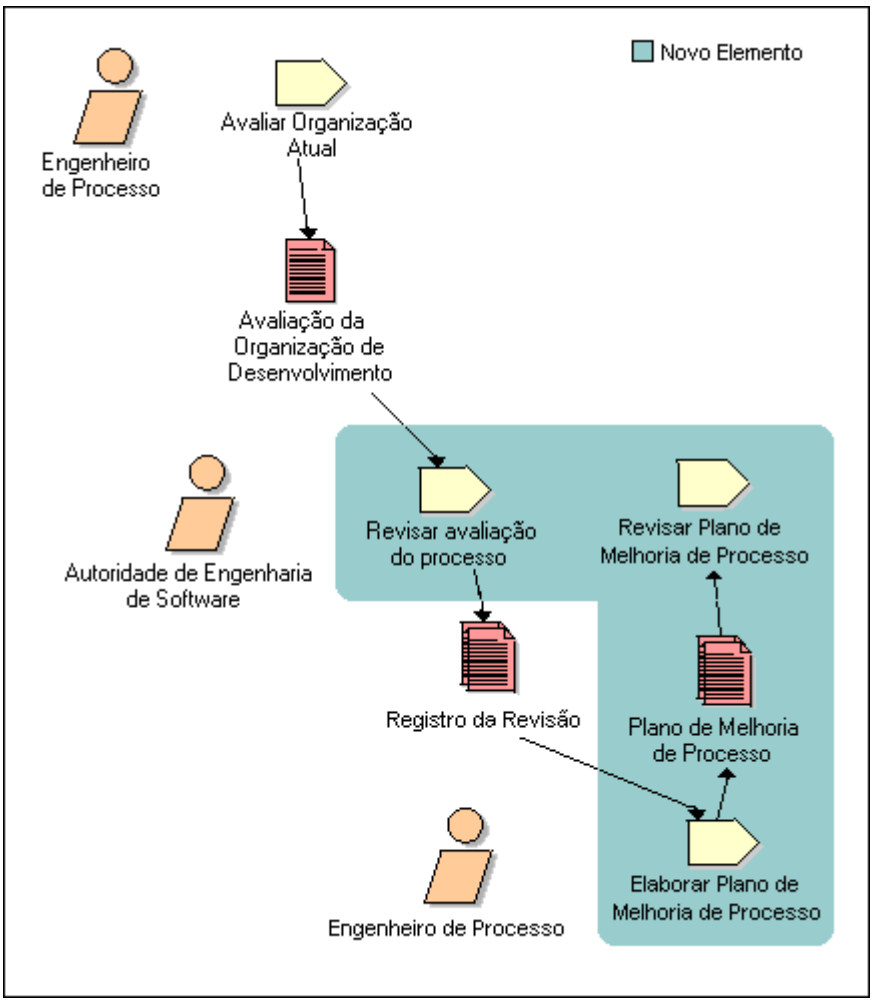


FIGURA 6.6 – Melhorar Processo de Software da Organização

Atividade: Elaborar Plano de Melhoria de Processo

Finalidade:

Na atividade Avaliar a Organização atual são identificadas as melhorias necessárias ao processo, e a atividade elaborar um plano de melhoria, se propõe a desenvolver um plano para implementar as melhorias identificadas e revisadas no processo de desenvolvimento da organização.

Passos:

- Identificar as melhorias que precisam ser implementadas com base na Avaliação da Organização e no Registro de Revisão.
- Analisar como essas melhorias podem ser inseridas no processo de desenvolvimento da organização.
- Atualizar o processo de desenvolvimento para contemplar essas melhorias.
- Informar as partes interessadas das alterações no processo.

Artefatos de Entrada: Avaliação da organização de desenvolvimento e Registro de Revisão

Artefatos de Saída: Plano de Melhorias de Processo

Freqüência: no final do projeto

Trabalhador: Engenheiro de Processo

Atividade: Revisar Plano de Melhoria de Processo

Finalidade:

A Autoridade de Processo de Engenharia de Software deve revisar o Plano de Melhoria de Processos elaborado pelo Engenheiro de Processo.

Passos:

- Planejar a reunião de revisão do Plano de Melhoria de Processo.
- Distribuir material da reunião.
- Conduzir a reunião de revisão.
- Registrar a decisão da reunião.

Artefatos de Entrada: Plano de Melhoria de Processo

Artefatos de Saída: Plano de Melhoria de Processo

Freqüência: no final do projeto

Trabalhador: Autoridade de Engenharia de Software

Artefato: Plano de Melhoria de Processo

Finalidade:

O plano de melhoria do processo deve descrever as atividades que serão executadas para desenvolver e melhorar o processo de desenvolvimento da organização.

Trabalhador: Engenheiro de Processo

Entrada para as Atividades:

- ✓ Revisar Plano de Melhoria de Processo
- ✓ Desenvolver Caso de Desenvolvimento
- ✓ Auditar atividades e artefatos para desenvolver e manter o processo de software padrão da organização

Saída para as Atividades:

- ✓ Elaborar Plano de Melhoria de Processo

7 Sistemas de Gerenciamento de *Workflow*

O desenvolvimento de aplicações de software não-triviais, raramente é executado por uma única pessoa, normalmente, demanda a participação de equipes de pessoas. A interação e a troca de informações entre os membros da equipe é um dos fatores que pode determinar o sucesso ou o fracasso de qualquer iniciativa de desenvolvimento [BAN 96].

Process-Centered Software Engineering Environments (PSEE) têm sido desenvolvido para suportar a definição e execução das várias fases do processo de software [FIN 94, BAN 96]. Vários projetos têm sido criados com este objetivo, como por exemplo: EPOS [FIN 94], SPADE [BAN 96], ARCADIA [UCI 95]. Porém, estes ambientes, geralmente, usam uma metodologia específica, que impõe restrições na maneira como o software deve ser desenvolvido, e se limitam a apoiar suas ferramentas próprias [CHA 97a, OCA 98].

Nas últimas duas décadas, sistemas de gerenciamento de *workflow* (SGW) têm sido utilizados com sucesso na modelagem de processos de negócio [OCA 98, WFM 99, GEO 95]. O processo de desenvolvimento de software pode ser visto como um processo complexo, composto de várias fases para a criação e evolução de sistemas de software.

Sistemas de gerenciamento de *workflow* suportam fluxos de controle complexos, ricas estruturas de dados e integração de ferramentas [CHA 97a]. Um SGW provê a automação de procedimentos de processos de negócio por gerenciar a seqüência das atividades e invocar recursos humanos e de tecnologia de informação associados aos vários passos das atividades [WFM 99].

Um processo de software, geralmente, apresenta-se na forma de uma metodologia, a qual identifica as fases importantes e suas relações, provendo guias para assegurar a qualidade de algumas fases, e formando a base para ferramentas de suporte automatizadas [CHA 97, OCA 98].

SGWs têm sido usados com algum sucesso para estruturar e otimizar processos de negócio e para suportar a implementação prática de reengenharia de processos de negócio, porém estes sistemas têm chamado atenção de outros domínios de aplicação como aplicações científicas e de engenharia [CHA 97a]. Como em muitos desses domínios de aplicação, o processo de software é também um processo complexo, onde ferramentas automatizadas para seu gerenciamento e execução são altamente desejadas.

Alguns autores propõem o uso do paradigma de *Workflow* para suportar a execução e o gerenciamento dos processos de software [CHA 97, OCA 98, CHA 97a, BAR 2000].

Este trabalho propõe experimentar o uso de um SGW comercial, Exchange 2000 Server, para implementar um protótipo de um ambiente de apoio a um processo de software baseado no Processo Unificado Rational estendido para suportar a implantação dos níveis 2 e 3 do CMM.

Este capítulo está assim estruturado: na seção 7.1 são caracterizados os sistemas de gerenciamento de *workflow* e a arquitetura genérica de um SGW proposta pela *Workflow Management Coalition* (WfMC). Na seção 7.2 são descritos brevemente alguns SGW

comerciais, e na seção 7.3 são descritas as características principais do Exchange 2000 Server. O ambiente desenvolvido será descrito no capítulo 8.

7.1 Sistemas de Gerenciamento de *Workflow*

A *Workflow Management Coalition* (WfMC), define *workflow* como "a automação de um processo de negócio, no todo ou em parte, durante o qual documentos, informações ou tarefas são passadas de um participante para o outro de acordo com um conjunto de regras estabelecidas" [WFM 99].

Já um sistema de gerência de *workflow* (SGW), é definido como: "um software que permite a definição, criação e gerência de *workflows,* sendo capaz de interpretar a definição do processo, de interagir com os participantes do *workflow* e, quando necessário, de invocar ferramentas e aplicativos de sistemas de informação [WFM 99]".

A *Workflow Management Coalition* (WfMC) é uma organização sem fins lucrativos que tem como objetivo estabelecer uma terminologia comum e padrões para a tecnologia de *workflow*, permitindo a interoperabilidade entre os diversos sistemas de *workflow.*

Apesar da grande variedade de produtos de *workflow* encontrados no mercado, é viável conceber um modelo genérico de implementação de sistemas de *workflow,* que abranja a grande maioria destes produtos. Assim, torna-se mais fácil compreender o funcionamento de um sistema de *workflow* [WFM 99].

A WfMC propôs a arquitetura descrita na Figura 7.1. Nesta arquitetura, são identificados os principais componentes de um sistema de *workflow,* juntamente com as respectivas interfaces, formando um modelo abstrato. Este modelo abstrato pode possuir diversas implementações, diferindo em aspectos como plataformas, protocolos de rede e tecnologias de distribuição. Conseqüentemente, as interfaces especificadas precisam suportar essa diversidade de ambientes operacionais.

Na arquitetura do SGW, conforme Figura 7.1 são identificados alguns componentes funcionais principais, que são [WFM 99]:

Ferramenta de definição de processo

A ferramenta de definição de processo é utilizada para transformar a descrição do processo em uma forma processável pelo computador. Esta ferramenta pode ser baseada em uma linguagem formal de definição de processos, em um modelo de relacionamento de objetos ou, em sistemas mais simples, um *script* ou um conjunto de comandos de roteamento para transferir informações entre os participantes [WFM 99, AMA 97].

Definição do processo

A definição do processo contém todas as informações necessárias sobre o processo, para permitir que ele seja controlado pelo *workflow*. Isto inclui informações sobre as condições de início e de término, sobre as atividades que o compõem e as regras para a transição entre elas, sobre os aplicativos que devem ser invocados e sobre os dados relevantes ao *workflow* que podem ser referenciados [WFM 99, AMA 97].

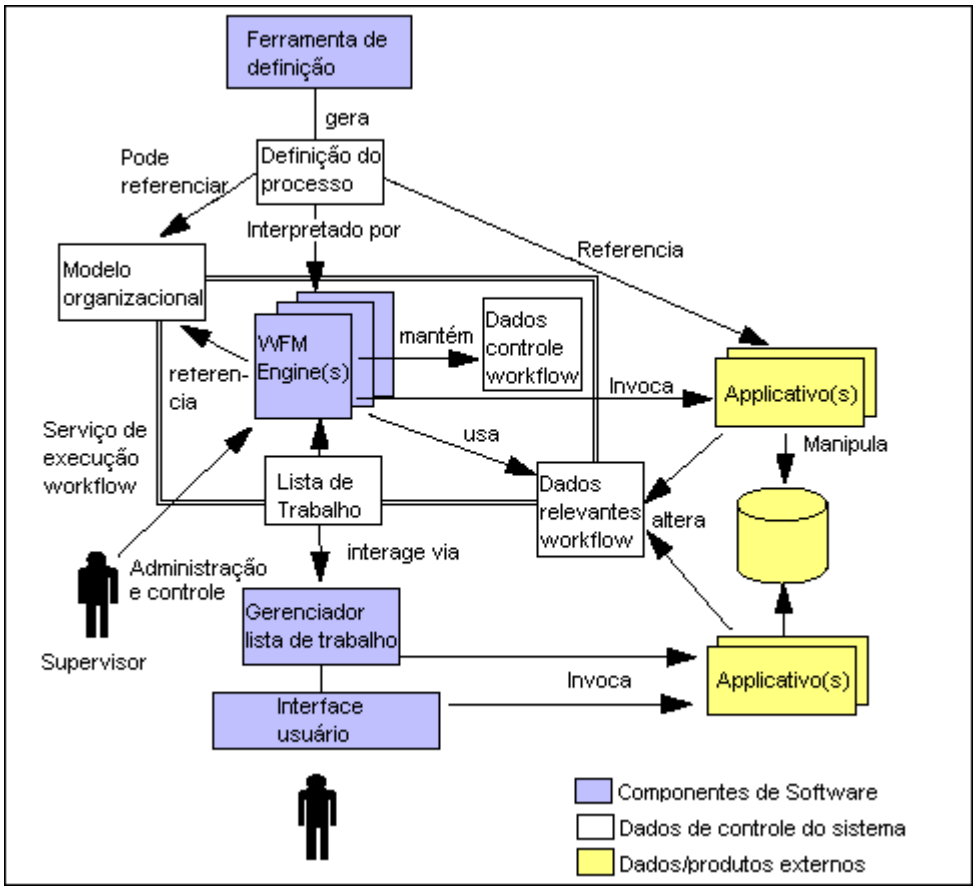


FIGURA 7.1 - Arquitetura genérica de um sistema de *workflow*

Serviço de execução do *workflow*

O serviço de execução de *workflow* constitui-se no núcleo de um sistema de *workflow*. Este serviço interpreta a definição do processo e controla a instanciação dos processos e a seqüência de atividades, adicionando itens de trabalho às listas dos usuários e invocando aplicativos, quando necessário. Dentro desse serviço, encontra-se o *workflow engine*, os dados relevantes ao *workflow* e os dados de controle do *workflow*.

O *workflow engine* é o módulo de software que efetivamente controla os processos, gerenciando suas respectivas instâncias. Os dados relevantes ao *workflow* são os dados gerados ou atualizados pelos aplicativos, mas que são acessíveis à máquina de *workflow*, e são usados em controles do *workflow*. Os dados de aplicativos são manipulados pelos aplicativos de *workflow* [WFM 99, AMA 97].

Listas de Trabalho

Sempre que um participante for necessário para a execução de uma tarefa, o *workflow engine* colocará itens na lista de trabalho do participante escolhido. A alocação de itens de trabalho a um participante pode ser feita estaticamente (na definição do processo) ou dinamicamente [WFM 99, AMA 97].

Gerenciador da lista de trabalho e interface com o usuário

O participante interage com sua lista de trabalho por meio de um gerenciador de lista de trabalho. Este componente representa a interface do usuário com o serviço de execução de *workflow*. Por meio dessa interface os itens de trabalho são apresentados ao usuário, e também o participante pode informar o término da atividade atribuída [WFM 99, AMA 97].

Aplicativos

Os aplicativos correspondem às ferramentas externas ao sistema de *workflow* que são necessárias a execução das atividades. Os aplicativos podem ser editores de texto, planilhas eletrônicas, formulários eletrônicos, etc [WFM 99, AMA 97].

Supervisão

Em um sistema de *workflow* existem algumas funções de supervisão, como alterar regras de alocação de tarefas. Para executar essas funções é necessário privilégio de supervisão, que normalmente é fornecido para uma estação de trabalho ou para um usuário particular [WFM 99, AMA 97].

7.2 Sistemas de Gerenciamento de Workflow Comerciais

Existe hoje, disponível no mercado, uma variedade de ferramentas de *workflow* com diferentes finalidades. Qualquer sistema de informação que se baseia na ordenação e controle de tarefas especializadas pode ser automatizado usando técnicas de *workflow* [AMA 97].

Em linhas gerais, sistemas de *workflow* podem ser classificados de acordo com o tipo de processos de trabalho que gerenciam [ARA 99], e podem ser agrupados em *workflows ad hoc* e de produção.

Sistemas de *workflow ad hoc* são caracterizados por suportarem processos onde as tarefas e suas regras de execução são desconhecidas. Nestes sistemas, não existe uma estrutura pré-definida para o processo, ou esta estrutura pode ser modificada em tempo de execução. A lógica de execução é definida pelas ações executadas pelo usuário em tempo de execução do *workflow* e a sincronização das atividades é realizada pela troca de mensagens entre os participantes do *workflow* [GEO 95, AMA 97]. Aplicações típicas de *workflows ad hoc* são: criação de documentos técnicos, desenvolvimento de software, requisição de viagens, campanha de marketing, etc.

Sistema de produção caracterizam-se pela habilidade de gerenciar processos complexos, que possuem uma estrutura fixa e um conjunto de regras de roteamento entre as atividades. Este tipo de *workflow* é usado quando se necessita monitorar o status e a localização dos processos e documentos manipulados pelo sistema em cada instante de tempo [GEO 95, AMA 97]. Aplicações típicas de *workflows* de produção são: procedimentos de empréstimo, reclamatória de seguros, aprovação de ordens de compra, solicitação de orçamentos, etc.

A seguir serão descritos brevemente alguns sistemas de gerência de *workflow* comerciais.

7.2.1 Oracle Workflow

Oracle Workflow é uma ferramenta desenvolvida pela Oracle Corporation [ORA 2001].

Oracle Workflow gerencia os processos de negócios de acordo com as regras definidas pelo projetista do modelo. O Oracle Workflow é formado pelos seguintes componentes: *Oracle Workflow Builder, Workflow Engine, Workflow Definitions Loader,* e *Workflow Monitor* [ORA 2001].

As principais características do Oracle Workflow incluem: capacidade de extensão por meio de procedures PL/SQL, notificação eletrônica, integração com e-mail, workflow baseado na Internet, sistema de eventos de negócio, monitoramento dos *workflow*, e interoperabilidade [ORA 2001].

7.2.2 Ultimus

Ultimus Workflow Suite é uma aplicação cliente-servidor baseada na Web, desenvolvida pela Ultimus. Ultimus Workflow Suite versão 4.2 é baseada no Microsoft Transaction Server (MTS) e no Microsoft Internet Information Server (IIS) [ULT 98, ULT 98a].

Algumas das principais características de Ultimus incluem [ULT 98]: projeto de *workflow* colaborativo, documentação de *workflow* automática, suporte a várias plataformas, interfaces abertas, e habilidade de criar clientes customizados usando COM e ou DCOM.

Os principais componentes dessa ferramenta são [ULT 98, ULT 98a]: Ultimus Designer, Ultimus Administrator, Ultimus Org Chart, Ultimus Workflow Server, Ultimus Client e Ultimus FloStation.

7.2.3 Lotus Notes

A ferramenta da Lotus provê a infra-estrutura necessária à gerência de *workflow*, por meio do uso combinado de formulários eletrônicos, gerenciamento de documentos, sistema de mensagens e replicação de dados. O Lotus Notes utiliza uma linguagem proprietária chamada de "LotusScript", baseada em Visual Basic Script, para descrição das atividades e do processo. O roteamento das tarefas é controlado pelos formulários via macros ou scripts e a troca de mensagens e notificações de usuários são totalmente executadas por meio de um sistema proprietário de gerenciamento de correio eletrônico [AMA 97].

Na seção 7.3, será descrito o Exchange 2000 Server, sistema de *workflow* escolhido para a realização do experimento descrito no capítulo 8.

A utilização de sistemas de *workflow* comerciais para construção de ambientes de apoio ao processo de software pode ser uma forma de facilitar a implementação desses ambientes. A implementação de um ambiente de apoio ao processo partindo-se do zero, pode ser uma tarefa complexa, cara e demorada, visto que estes ambientes possuem características como: gerenciamento de processo, integração com ferramentas, atribuição de

atividades e monitoramento de projetos. Utilizando-se um sistema de *workflow* comercial, várias dessas características já estão incorporadas no sistema, e podem ser utilizadas de forma transparente.

O Exchange 2000 Server foi o servidor de colaboração escolhido para este experimento, porque algumas organizações já o utilizam como servidor de e-mail, e poderiam passar a utilizá-lo também como servidor de *workflow,* não necessitando adquirir um novo produto. Exchange é integrado com as ferramentas do Microsoft Office, e essas ferramentas são bastante utilizadas no processo definido para o experimento.

7.3 Exchange 2000 Server

Exchange 2000 Server permite construir aplicações que modelam e automatizam processos de negócio [MAR 2000]. Os benefícios oferecidos pelo Exchange são [MAR 2000, RIZ 2000]:

- ✓ Tecnologia Web Storage System, permite o armazenamento de qualquer tipo de dado;
- ✓ Endereçamento por URL para acessar dados, possibilita o armazenamento distribuído de informações;
- ✓ Várias tecnologias de acesso a dados, tais como: ActiveX Data Objects (ADO), Collaboration Data Objects (CDO), Hypertext Transfer Protocol (HTTP), e eXtensible Markup Language (XML);
- ✓ Suporte a eventos (eventos síncronos, assíncronos e de sistema);
- ✓ Lógica de *workflow*, possibilitando o controle sobre o comportamento de um item em um folder.

A seguir serão descritas brevemente as características do Exchange consideradas mais importantes.

7.3.1 Active Directory

No Windows 2000, o Active Directory gerencia todas as informações dos usuários e dos grupos em um domínio Windows. Exchange 2000 entrega a responsabilidade de gerenciamento de usuários e caixas de correio ao Windows 2000 e ao Active Directory. Como o Active Directory já gerencia as contas dos usuários do domínio, faz sentido que ele também gerencie as caixas de correio desses mesmos usuários [MAR 2000].

7.3.2 Web Storage System

No núcleo do paradigma de desenvolvimento para Exchange está o Web Storage System (WSS) [MAR 2000]. O Web Storage System possibilita interagir e gerenciar dados. Cada WSS é organizado como um sistema de arquivos tradicional, isto é, como uma hierarquia de pastas, onde cada pasta pode conter itens. Itens podem ser arquivos padrão do Exchange, como: contatos, mensagens, compromissos ou arquivos mais complexos, como:

Active Server Pages, documentos Word e Planilhas Excel, e também outras pastas [MAR 2000].

O Web Storage System Schema é usado para definir todos os recursos que serão armazenados, tais como: pastas, itens e arquivos *Web*. Este esquema é uma compilação de propriedades pré-definidas que definem os atributos, tais como o nome de um recurso que deve ser exibido. Essas propriedades são manipuladas como colunas em uma tabela de uma base de dados relacional, e podem ser usadas para eficientemente organizar, indexar, classificar, visualizar e consultar os dados, usando a Linguagem de Consulta Estruturada (SQL) [MAR 2000].

O Web Storage System implementa um modelo de eventos para reagir às ações dos usuários [RIZ 2000]. Os eventos síncronos e assíncronos reagem à solicitação de dados. Uma solicitação de dados pode ser a solicitação de um novo recurso, a alteração de um recurso já existente, a exclusão de um recurso, ou operações como mover ou copiar um recurso [MAR 2000].

Eventos síncronos disparam antes da solicitação de dados estar gravada no Web Storage System, enquanto que os eventos assíncronos ocorrem depois da solicitação estar gravada. Eventos de sistema não reagem a solicitações de dados, e disparam quando a base de dados inicia ou pára em um tempo particular [MAR 2000].

7.3.3 Acesso a Dados

Exchange usa o Web Storage System para armazenar recursos como itens e arquivos, e usa o Active Directory para armazenar e gerenciar os dados sobre as caixas de correio do Exchange. Para acessar os dados no Web Storage System podem ser usados as seguintes tecnologias: ActiveX Data Objects (ADO), Collaboration Data Objects (CDO), Hypertext Transfer Protocol (HTTP), e eXtensible Markup Language (XML). Para acessar dados do Active Directory, podem ser usado tanto o Active Directory Directory Service Interfaces (ADSI) como o CDO [RIZ 2000, MAR 2000].

7.3.4 Exibir Dados

Os dados do Exchange podem ser exibidos usando o Outlook 2000, o Microsoft Outlook Web Access e o Web Storage Systems Forms [MAR 2000].

Outlook 2000 pode ser utilizado como interface cliente do Exchange 2000.

Outlook Web Access (OWA) é um conjunto de componentes *Web* instalado junto com o Exchange 2000 Server. OWA é acessado por meio de um *web browser*, utilizando como endereço a localização da caixa de correio.

Web Storage System Forms oferecem funcionalidades de aplicações cliente customizadas, possibilitando a reutilização de componentes do OWA e a criação de páginas HTML ou ASP, para mostrar as informações armazenadas no Web Storage System [MAR 2000, RIZ 2000].

7.3.5 Workflow Designer

Workflow Designer é uma ferramenta gráfica que pode ser usada para construir processos de *workflow* [MAR 2000].

Um processo de *workflow* automatiza o controle de itens em uma pasta por movê-los através de uma série de estados, dependendo das ações e condições especificadas. Aplicações de *workflow* criadas no Exchange podem rastrear qualquer tipo de item disponível no Microsoft Outlook, tais como: mensagens, formulários ou tarefas, e também qualquer tipo de arquivo, como documentos (.DOC), planilhas eletrônicas (.XLS), ou arquivos texto (.TXT) [MAM 2000].

A Figura 7.2 mostra a interface da ferramenta Workflow Designer.

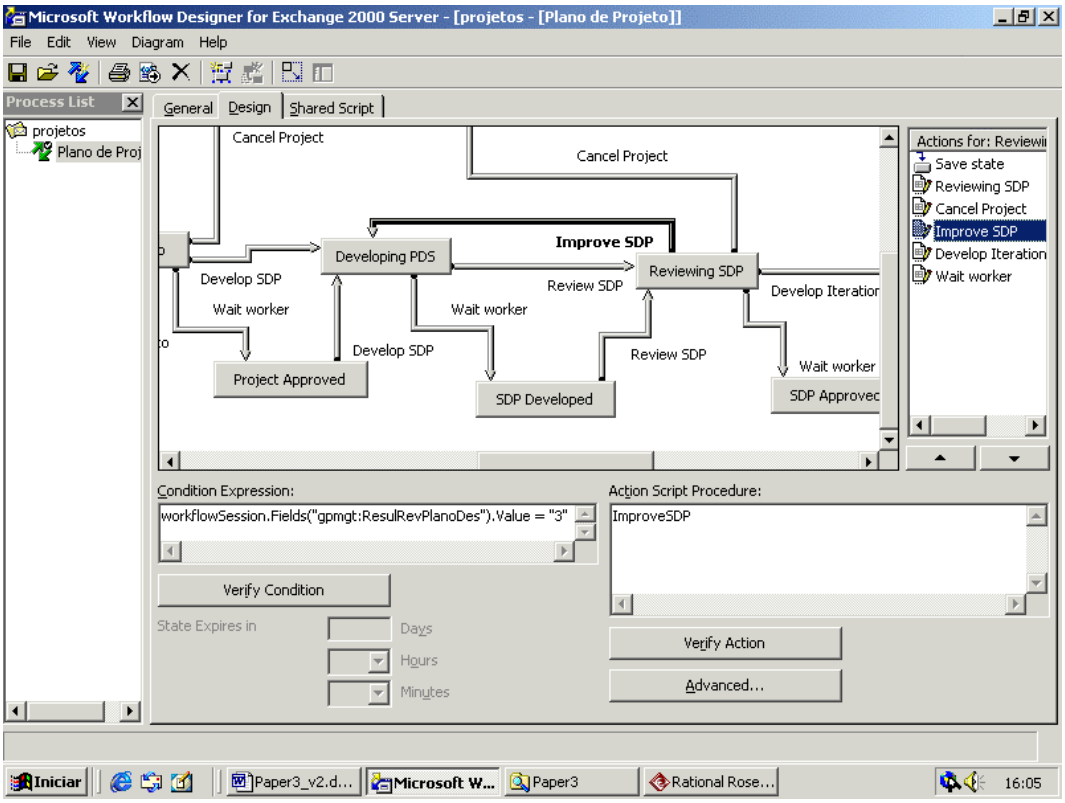


FIGURA 7.2 – Interface da Ferramenta Workflow Designer

7.3.6 Compatibilidade entre o Exchange Server e a Arquitetura Proposta pela WfMC

Com o objetivo de verificar a compatibilidade entre o Exchange 2000 Server e a arquitetura proposta pela *Workflow Management Coalition* (WfMC), o Exchange foi analisado em relação os principais componentes da arquitetura genérica (Fig. 7.1). A seguir será descrito se o Exchange apresenta o componente descrito na arquitetura ou não apresenta.

Ferramenta de definição de processo

A ferramenta gráfica Workflow Designer for Exchange pode ser utilizada para construir processos de *workflow* em um *folder* de maneira rápida e fácil, quando se utiliza o Exchange Server.

Workflow Designer for Exchange é disponibilizada como parte do Office 2000 Developer Versão 1.5 e também no pacote do Exchange Server.

Essa ferramenta pode ser instalada em qualquer máquina cliente que esteja sendo usada pelo autor do *workflow*.

Definição do processo

Utilizando a ferramenta Workflow Designer é possível definir o processo de negócio. O suporte fornecido por esta ferramenta a definição dos processos é limitado, e muitas funcionalidades, tais como: envio de mensagens, recuperar, alteração e inserção de dados, precisam ser programadas em scripts utilizando a linguagem VBScript.

As mudanças de estado são feitas automaticamente pela ferramenta, apenas é necessário modelar os estados e indicar as transições de estado, juntamente com as condições para que a mudança ocorra.

Serviço de execução do *workflow*

O *workflow engine* controla as mudanças de estados dos itens armazenados em um *folder*, reagindo aos eventos e disparando as ações definidas no processo. Os dados relevantes ao *workflow* e os dados acessados pelos aplicativos são armazenados no *Web Storage System*.

Listas de Trabalho

A alocação de itens de trabalho a um participante tem que ser feita por meio de programação de ações em VBScripts.

Gerenciador da lista de trabalho e interface com o usuário

Os dados do Exchange podem ser exibidos usando o Outlook 2000, Microsoft Outlook Web Access e Web Storage Systems Forms [MAR 2000]. É necessário criar as interfaces e manter as listas de trabalho por meio de programação.

Aplicativos

Workflow Designer não oferece suporte a ativação de ferramentas. No experimento descrito no capítulo 8, a ativação de ferramentas é realizada por meio de *links* a documentos em páginas HTML. O Outlook 2000 é totalmente integrado com as ferramentas do Office, possibilitando ao usuário utilizar os documentos do Word ou planilhas do Excel como base para as aplicações de colaboração.

Supervisão

Para criar e registrar aplicações de *workflow*, usando o Exchange, é necessário que o usuário esteja registrado no servidor.

O Exchange 2000 Server facilita a definição dos processos de negócio a serem modelados por meio do uso de uma ferramenta gráfica. Nessa ferramenta é possível

especificar os estados do *workflow*, as transições de estados e as ações associadas a cada transição de estado.

Por outro lado, várias tarefas devem ser programadas, como a inserção e recuperação de dados do Web Storage System, envio de mensagens, ativação de ferramentas, etc.

8 Protótipo de um Ambiente de Gerenciamento de Projetos Baseado no Exchange Server

Este capítulo descreve o protótipo de um ambiente de apoio a um processo experimental criado usando Exchange Server, como servidor de colaboração, e o Processo Unificado estendido para suportar níveis 2 e 3 de CMM, como modelo de processo.

Este Ambiente de Gerenciamento de Projetos (AGP) é uma ferramenta de suporte ao processo baseada na Web, que melhora a coordenação, acompanhamento e troca de informação entre os membros de uma equipe de desenvolvimento. Um usuário usando um *web browser* pode acessar o AGP por meio de uma URL (*Universe Resource Locator*) específica.

Na seção 8.1 é descrito o processo de software modelado no Exchange. Na seção 8.2 é descrita a arquitetura do ambiente. Na seção 8.3 são descritos os casos de uso, na seção 8.4 os diagramas de classe e na seção 8.5 os diagramas de estado. Na seção 8.6 é descrito o protótipo implementado baseado no modelo de processo descrito na seção 8.1. Na seção 8.7 é descrita a avaliação do protótipo com base em alguns requisitos considerados desejáveis em ambientes de apoio a processos de software.

8.1 Modelagem de Processos de Software

RUP descreve ferramentas para facilitar a integração de artefatos produzidos por equipes distribuídas baseado no gerenciamento de configuração [RAT 2001] (Rational Corporation oferece ferramentas, como: ClearCase), mas não menciona ferramentas para suportar o controle e o monitoramento de projetos.

Com o objetivo de facilitar o gerenciamento de projetos, esta dissertação propõe a experimentação do uso de um Sistema de Gerenciamento de *Workflow* comercial para desenvolver uma ferramenta *Web* que suporte processos de desenvolvimento de software e ajude as organizações que desejam alcançar níveis 2 e 3 do CMM.

Processos de software em nível 2 do CMM têm habilidade de estabelecer políticas para gerenciar um projeto de software e procedimentos para implementar estas políticas.

Considerando, que áreas-chave do CMM nível 2, têm como objetivo principal estabelecer processos de gerenciamento de projetos, a ferramenta proposta se concentra nas atividades que visam o gerenciamento de projetos. O modelo de processo proposto é baseado na macro-atividade de Gerenciamento de Projetos, descrito pelo RUP, e estendido por este trabalho. Porém esse modelo de processo foi simplificado objetivando facilitar a implementação.

A Figura 8.1 mostra o modelo de processo de gerenciamento de projetos simplificado, usando a notação de diagrama de atividades da UML para descrever como o processo flui por meio dos diferentes trabalhadores. O uso de diagramas de atividades decorre do fato do Processo Unificado ter optado por descrever suas macro-atividades usando este tipo de diagrama [RAT 99, RAT 2001].

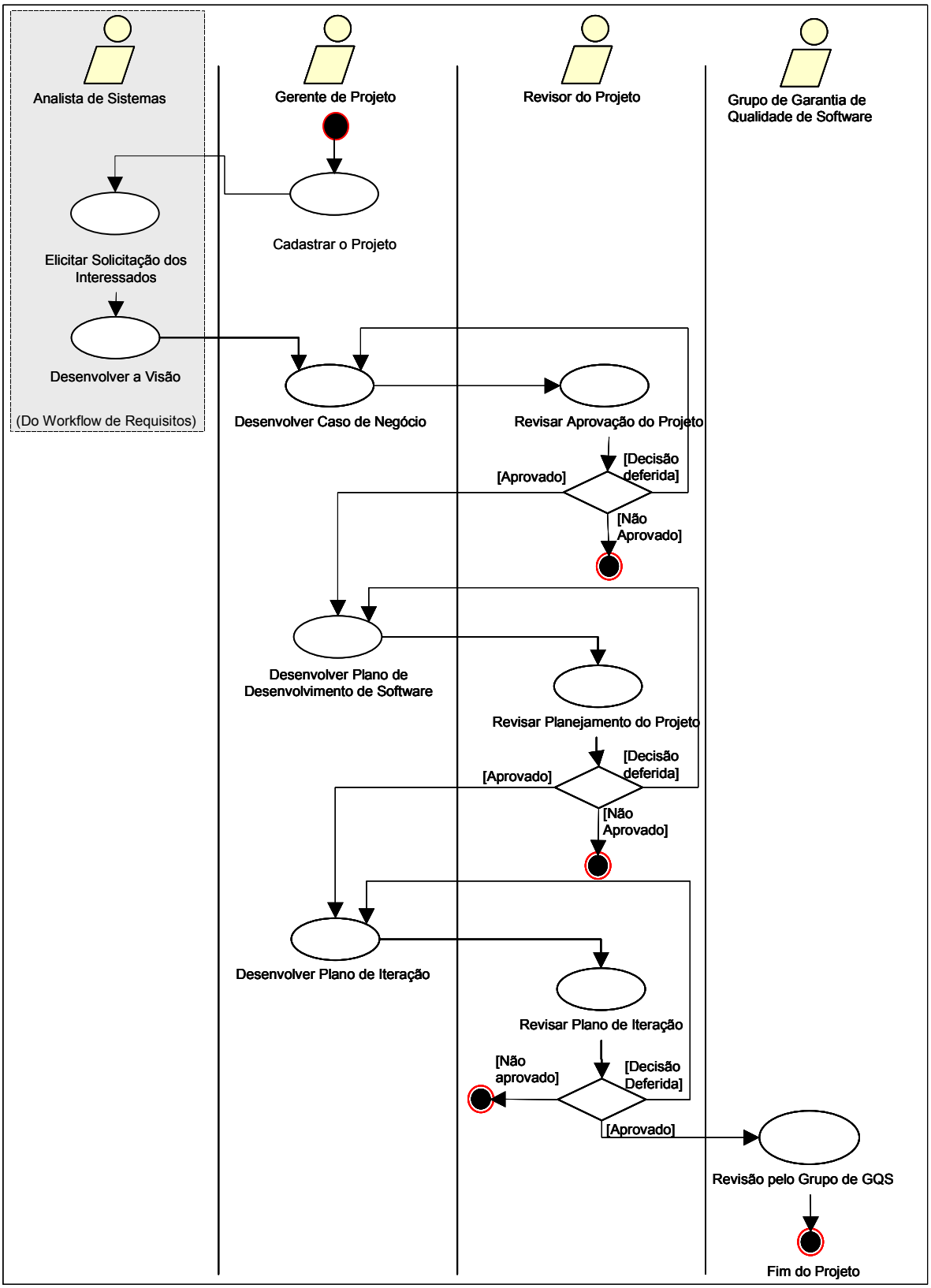


FIGURA 8.1 – Macro-atividade de Gerenciamento de Projetos Simplificada

O processo inicia quando o gerente de projeto assinala as responsabilidades pelas tarefas descritas pelo processo, e explicitamente determina que o processo deve iniciar

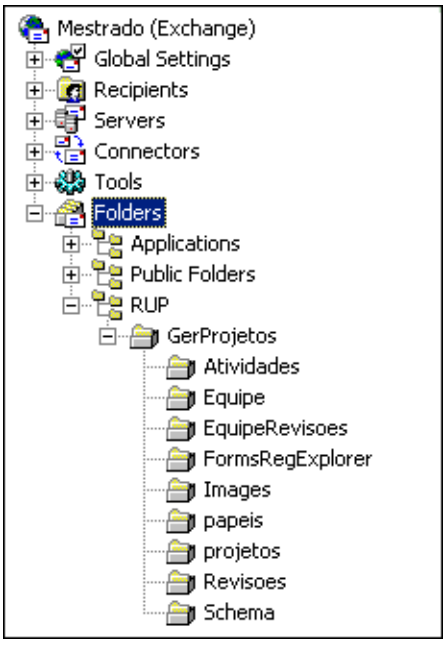
Cada atividade descrita pelo processo é associada com um trabalhador, a descrição da atividade (RUP) e um modelo.

Um trabalhador representa um cargo ocupado por indivíduos em um projeto, e define como eles devem executar seu trabalho [JAC 98]. A descrição da atividade é um *link* para o procedimento que deve ser realizado para executar a atividade, descrito pelo Processo Unificado Rational. A descrição da atividade está associada a guias de trabalho, que apresentam técnicas e conselhos práticos que são úteis para os trabalhadores executarem suas atividades [RAT 2001]. *Templates* são modelos, ou protótipos, de artefatos, que devem ser usados para criar os artefatos correspondentes [RAT 2001].

8.2 Arquitetura da Ferramenta de Gerenciamento de Projetos

Esta ferramenta foi construída usando o Microsoft Exchange 2000 Server como servidor de colaboração. O armazenamento dos dados ficou sob responsabilidade do Web Storage System. Na camada de apresentação foi utilizado o Web Storage System Forms, e na camada de negócios foram utilizados: ActiveX Data Objects (ADO), Collaboration Data Objects (CDO), eventos e lógica de *workflow*.

Inicialmente, foram criados no System Manager do Exchange uma pasta pública com nome de RUP, e um novo local de armazenamento nomeado de RUP *Store*. Mais tarde foram criadas novas pastas, como mostra a Figura 8.2.



Web Storage System Forms foram implementados usando Active Server Pages (ASP), para codificação do código que executa no servidor da aplicação e para retornar os dados mais atuais toda vez que as páginas *Web* são acessadas.

Foi incorporada lógica de *workflow* no folder projetos, usando a ferramenta Workflow Designer.

Um processo de *workflow* é definido por utilizar dois tipos de componentes – estados e ações – os quais automatizam e forçam a ordem nas quais as tarefas devem ser executadas, do início ao fim do processo [MAR 2000]. Estados no Workflow Designer não são muito interessantes porque eles não executam tarefas; eles somente servem como destino para um item do *workflow*.

FIGURA 8.2 – Hierarquia de Pastas

Ações são um dos mais importantes aspectos do Workflow Designer, desde que elas são associadas com a transição entre estados. Estas transições definem como o trabalho no *workflow* é executado [RIZ 2000].

8.3 Casos de Uso

Com base nas atividades descritas na macro-atividade de Gerenciamento de Projetos Simplificada (Fig. 8.1), os casos de uso identificados são mostrados na Fig. 8.3.

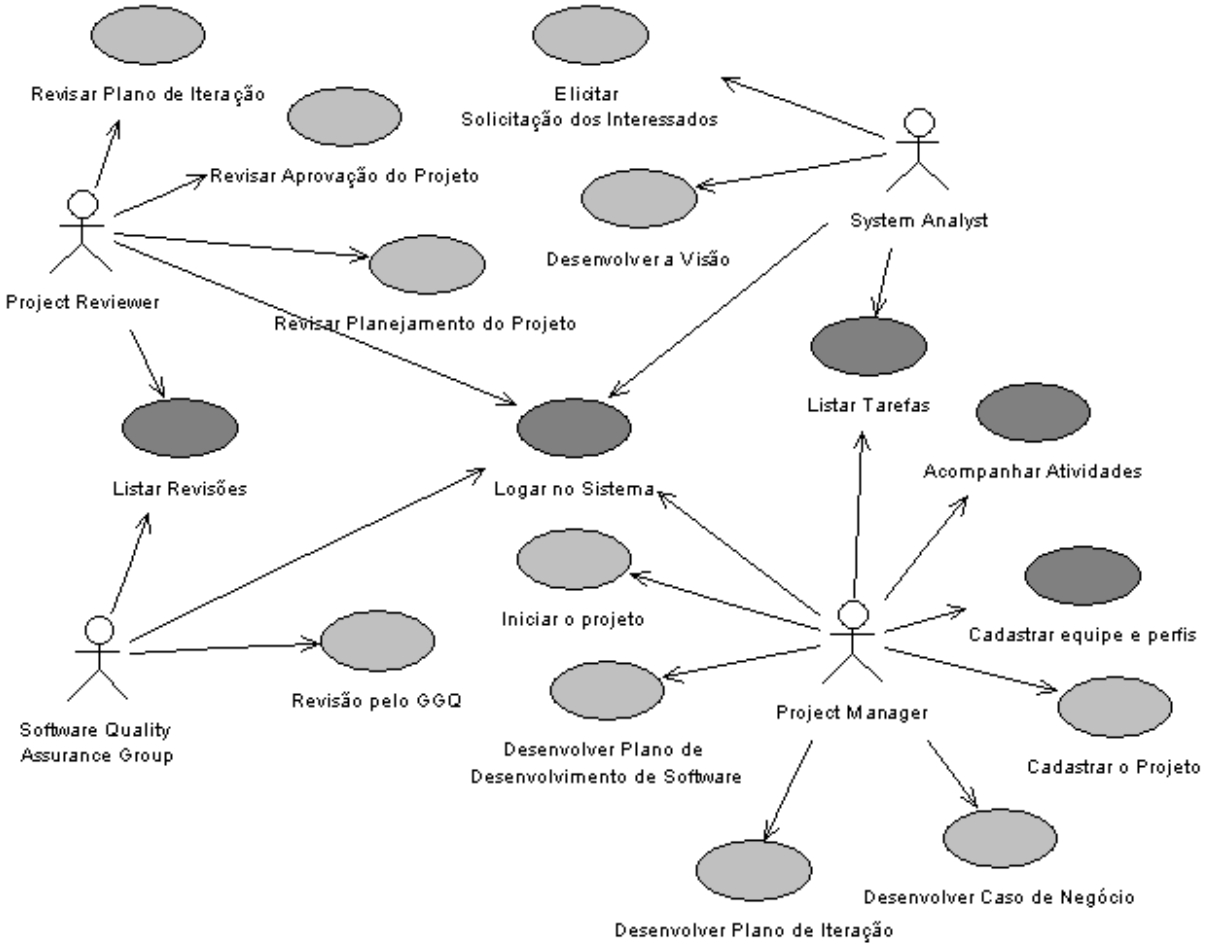


FIGURA 8.3 – Alguns Casos de Uso da Ferramenta de Gerenciamento de Projetos

Cada trabalhador cadastrado na ferramenta pode ser assinalado para um ou mais projetos, e as atividades são atribuídas de acordo com o perfil do usuário cadastrado no sistema.

Os casos de uso foram divididos em duas categorias, que são: casos de uso do processo, que causam uma transição de estado e os casos de uso de apoio, que não causam transição de estado.

Os casos de uso do processo têm que ser executados na seqüência definida pelo diagrama de atividades da macro-atividade de gerenciamento de projetos. Casos de uso de apoio podem ser executados a qualquer momento.

TABELA 8.1 – Descrição dos Casos de Uso de Gerenciamento de Projetos

Caso de Uso	Quem inicia a ação	Descrição
Cadastrar o Projeto	Gerente de Projeto	Cadastrar o projeto, descrevendo os responsáveis por cada atividade definida na macro-atividade de projeto.
Elicitar Solicitação dos Interessados	Analista de Sistemas	Desenvolver o artefato Solicitação dos Interessados, descrevendo as solicitações dos interessados do sistema.
Desenvolver Visão	Analista de Sistemas	Desenvolver o artefato Visão, definindo os problemas a serem resolvidos pelo sistema, os limites e as características principais do sistema.
Desenvolver Caso de Negócio	Gerente de Projeto	Desenvolver o Caso de Negócio, onde é elaborada uma justificativa econômica para o produto.
Revisar aprovação do Projeto	Revisor de Projeto	Determinar, do ponto de vista de negócio, se o projeto deve ser investigado ou não.
Desenvolver Plano de Desenvolvimento de Software	Gerente de Projeto	Desenvolver o Plano de Desenvolvimento de Software.
Revisar Planejamento do Projeto	Revisor de Projeto	Aprovar o Plano de Desenvolvimento de Software.
Desenvolver Plano de Iteração	Gerente de Projeto	Desenvolver um plano detalhado para a iteração, definindo atividades, responsáveis e critérios de avaliação.
Revisar Plano de Iteração	Revisor de Projeto	Aprovar ou não o Plano de Iteração.
Revisão pela APES	APES	Revisar as atividades de Planejamento de Projetos.

Os seguintes casos de uso descrevem opções de manutenção ou acompanhamento do sistema, não estando associados a macro-atividade de Gerenciamento de Projetos. Esses casos de uso podem ser executados a qualquer momento.

TABELA 8.2 – Descrição dos Casos de Uso de Apoio

Caso de Uso	Quem inicia a ação	Descrição
Logar no Sistema	Todos os Atores	Acessar a URL da ferramenta e inserir usuário e Senha. A ferramenta disponibiliza as opções de acordo com o perfil, e lista as atividades do usuário logado.
Listar Revisões	Revisor de Projeto APES	Lista todas as atividades de revisão que foram atribuídas ao usuário logado no sistema, separando-as em: a serem realizadas, concluídas e em andamento.
Listar Tarefas	Gerente de Projeto Analista de Sistemas	Lista as tarefas atribuídas ao usuário logado no sistema, separando as em concluídas e a serem realizadas.
Cadastrar Equipe e Perfil	Gerente de Projeto	Possibilita cadastrar os membros da equipe no projeto e associa-los aos papéis.
Acompanhar atividades	Gerente de Projeto	Lista as atividades do projeto, descrevendo o percentual concluído de cada atividade, com datas de início e de fim e o responsável.

8.4 Diagrama de Classes

Os dados persistentes do projeto, armazenados no *Web Storage System*, e as operações realizadas sobre esses dados, são descritos a seguir por meio de um diagrama de classes (Fig.7.4). Esses dados são definidos no *Web Storage System Schema* do Exchange 2000 Server.

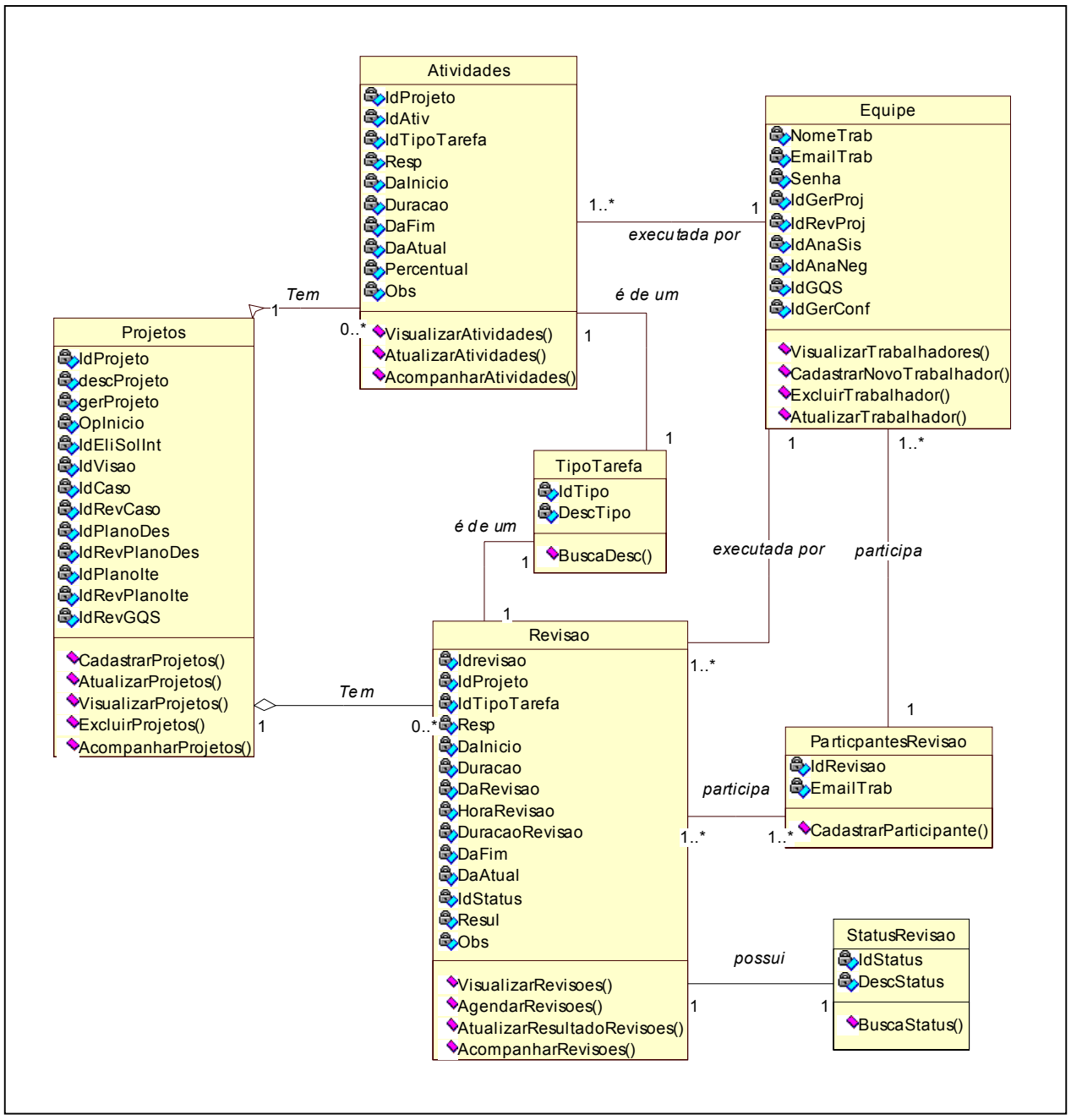


FIGURA 8.4– Diagrama de Classes da Ferramenta de Gerenciamento de Projetos

8.5 Diagrama de Estados

A Figura 8.5 mostra a seqüência de estados atribuídos ao projeto, durante a execução do processo de gerenciamento de projetos. Essa seqüência foi definida com base no *workflow* descrito na Figura 8.1 deste capítulo.

O processo inicia quando o gerente de projeto cadastra um novo projeto. Os dados desse projeto são gravados na pasta Projetos no WSS.

Quando cadastra o projeto, o gerente de projeto assinala o trabalhador responsável por cada atividade descrita na macro-atividade, com a data de início da atividade e o tempo estimado para execução da atividade. O gerente de projeto pode atribuir as responsabilidades a medida que as atividades forem sendo atribuídas, isto é, não precisa definir todos os responsáveis no cadastramento do projeto.

A seguir serão descritas as transições de estado que podem ocorrer, conforme mostra a Figura 8.5. Omitiu-se no diagrama a transição de estado para o próprio estado (transição interna), isto é, quando permanece no próprio estado, para deixar o diagrama mais claro.

O texto está organizado da seguinte forma: para cada estado são descritas as mudanças de estado que podem ocorrer partindo-se do estado em questão. Para cada mudança de estado serão descritos os eventos que ocasionam a mudança e as ações que serão realizadas como resultado da mudança de estado.

“Estado Inicial” (Projeto é cadastrado)

Muda para o estado “em planejamento”

O projeto é cadastrado e o gerente de projeto determina explicitamente que deseja salvar o projeto como rascunho (*checkbox*), o projeto passa do estado “inicial” para o estado “em planejamento”.

Evento:

- ✓ Gerente de projeto determina que o projeto deve ser salvo como rascunho.

Muda para o estado “em elicitação”

O projeto é cadastrado, o gerente de projeto determina explicitamente que o projeto deve ser iniciado (*checkbox*), e assinala o responsável por executar a próxima atividade.

Evento:

- ✓ Gerente de projeto determina o início do projeto e assinala responsável pela tarefa “Elicitar Solicitação dos Interessados”.

Ações:

- ✓ Informar o responsável pela execução da tarefa por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades do WSS.

Muda para o estado “planejado”

O projeto é cadastrado, o gerente de projeto determina explicitamente que o projeto deve ser iniciado (*checkbox*), mas não determina qual o responsável por executar a atividade “Elicitar Solicitação dos Interessados”.

Evento:

- ✓ Gerente de projeto determina o início do projeto, mas não assinala responsável pela tarefa “Elicitar Solicitação dos Interessados”.

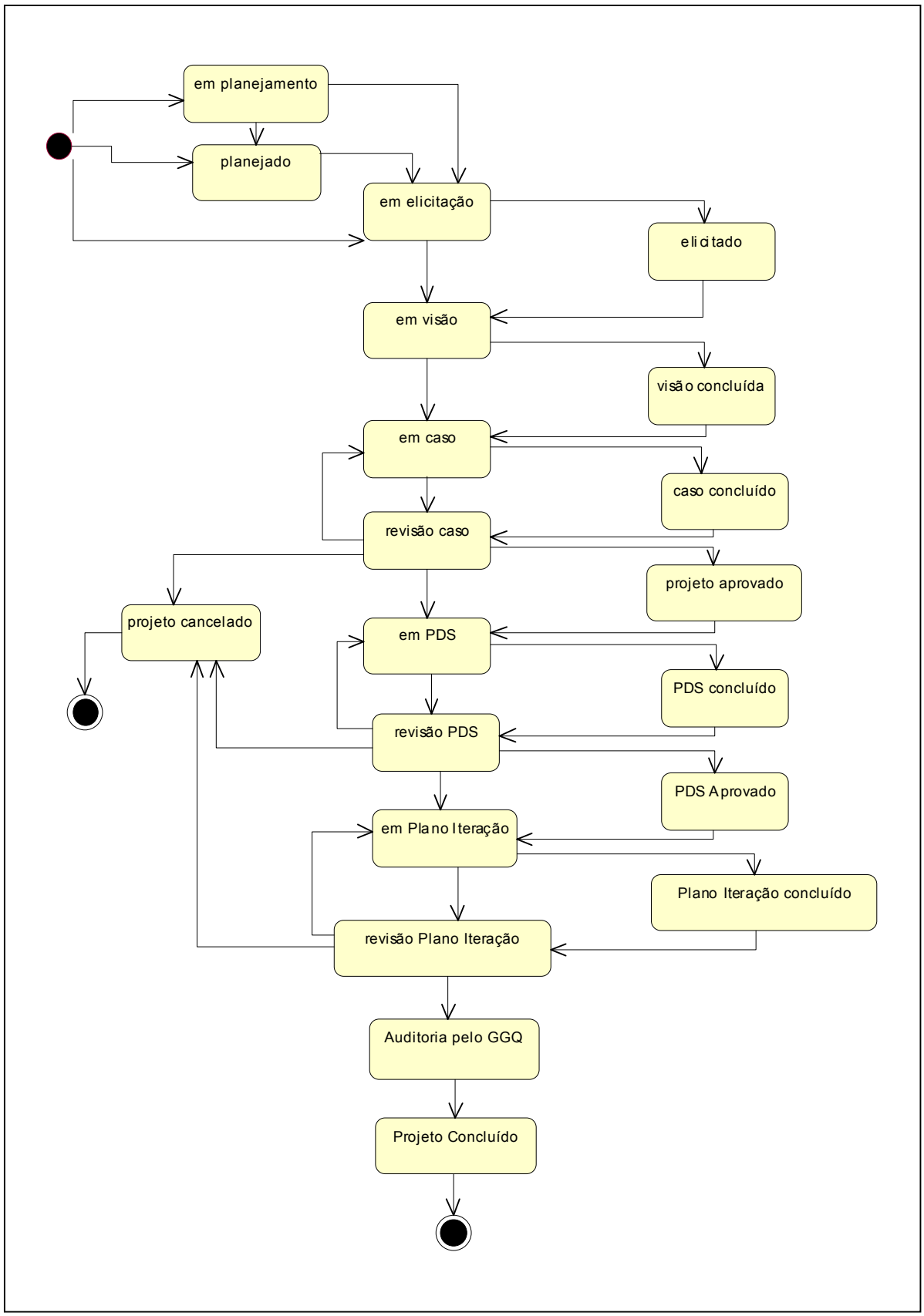


FIGURA 8.5 – Diagrama de Estados

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade "Elicitar Solicitação dos Interessados".

Estado "em planejamento":

Permanece no estado "em planejamento"

Evento:

- ✓ O gerente de projeto não determina o início do projeto.

Muda para o estado "planejado"

Evento:

- ✓ O gerente de projeto determina o início do projeto, mas não assinala o responsável pela tarefa "Elicitar Solicitação dos Interessados".

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade "Elicitar Solicitação dos Interessados".

Muda para o estado "em elicitação"

Evento:

- ✓ O gerente de projeto determina o início do projeto e assinala o responsável pela tarefa "Elicitar Solicitação dos Interessados".

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Estado "planejado":

Permanece no estado "planejado"

Evento:

- ✓ O gerente de projeto não assinala responsável pela atividade "Elicitar Solicitação dos Interessados".

Muda para o estado "em elicitação"

Evento:

- ✓ O gerente de projeto assinala responsável pela atividade "Elicitar Solicitação dos Interessados".

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.

- ✓ Inserir os dados da atividade na pasta atividades.

Estado "em elicitação":

Permanece no estado "em elicitação"

Evento:

- ✓ Atividade não concluída.

Muda para o estado "elicitado"

Evento:

- ✓ Atividade concluída, mas responsável pela atividade "Desenvolver Visão" não definido.

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade "Desenvolver Visão".

Muda para o estado "em visão"

Evento:

- ✓ Atividade concluída e responsável pela atividade "Desenvolver Visão" definido.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Estado "elicitado":

Permanece no estado "elicitado"

Evento:

- ✓ Responsável pela atividade "Desenvolver Visão" não definido.

Muda para o estado "em visão"

Evento:

- ✓ Gerente define responsável pela atividade "Desenvolver Visão".

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Estado "em visão":

Permanece no estado "em visão"

Evento:

- ✓ Atividade não concluída.

Muda para o estado "visão concluída"

Evento:

- ✓ Atividade concluída, mas responsável pela atividade "Desenvolver caso" não definido.

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade "Desenvolver Caso".

Muda para o estado "em caso"

Evento:

- ✓ Atividade concluída e responsável pela atividade "Desenvolver Caso" definido.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Estado "visão concluída":

Permanece no estado "visão concluída"

Evento:

- ✓ Responsável pela atividade "Desenvolver Caso" não definido.

Muda para o estado "em caso"

Evento:

- ✓ Gerente define responsável pela atividade "Desenvolver Caso".

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Estado "em caso":

Permanece no estado "em caso"

Evento:

- ✓ Atividade não concluída.

Muda para o estado "caso concluído"

Evento:

- ✓ Atividade concluída, mas responsável pela atividade "Revisar caso" não definido.

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade "Revisar Caso".

Muda para o estado "em revisão caso"

Evento:

- ✓ Atividade concluída e responsável pela atividade "Revisar Caso" definido.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta revisões.

Estado "caso concluído":

Permanece no estado "caso concluído"

Evento:

- ✓ Responsável pela atividade "Revisar Caso" não definido.

Muda para o estado "em revisão caso"

Evento:

- ✓ Gerente define responsável pela atividade "Revisar Caso".

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta revisões.

Estado "em revisão caso":

Permanece no estado "em revisão caso"

Evento:

- ✓ Atividade não concluída.

Muda para o estado "projeto aprovado"

Evento:

- ✓ Resultado da revisão é projeto aprovado e responsável pela próxima atividade não definido.

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade “Desenvolver PDS”.

Muda para o estado “em PDS (Plano de Desenvolvimento de Software)”

Evento:

- ✓ Resultado da revisão é projeto aprovado e responsável pela atividade “Desenvolver PDS” definido.

Ações:

- ✓ Informar o responsável pelo artefato por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Muda para o estado “em caso”

Evento:

- ✓ Resultado da Revisão é decisão adiada, o Caso de Negócio precisa ser melhorado.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Alterar os dados da atividade na pasta atividades.

Muda para o estado “projeto cancelado”

Evento:

- ✓ Resultado da revisão é projeto cancelado.

Estado “projeto aprovado”:

Permanece no estado “projeto aprovado”

Evento:

- ✓ Responsável pela atividade “Desenvolver PDS” não definido.

Muda para o estado “em PDS”

Evento:

- ✓ Gerente define o responsável pela atividade “Desenvolver PDS”.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Estado “em PDS”:

Permanece no estado “em PDS”

Evento:

- ✓ Atividade não concluída.

Muda para o estado “PDS concluído”

Evento:

- ✓ Atividade concluída, mas responsável pela atividade “Revisar PDS” não definido.

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade “Revisar PDS”.

Muda para o estado “em revisão PDS”

Evento:

- ✓ Atividade concluída e responsável pela atividade “Revisar PDS” definido.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta revisões.

Estado “PDS concluído”:

Permanece no estado “PDS concluído”

Evento:

- ✓ Responsável pela atividade “Revisar PDS” não definido.

Muda para o estado “em revisão PDS”

Evento:

- ✓ Gerente define responsável pela atividade “Revisar PDS”.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta revisões.

Estado “em revisão PDS”:

Permanece no estado “em revisão PDS”

Evento:

- ✓ Atividade não concluída.

Muda para o estado "PDS aprovado"

Evento:

- ✓ Resultado da revisão é PDS aprovado e responsável pela próxima atividade não definido.

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade "Desenvolver Plano de Iteração".

Muda para o estado "em Plano de Iteração"

Evento:

- ✓ Resultado da revisão é PDS aprovado e responsável pela atividade "Desenvolver Plano de Iteração" definido.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Muda para o estado "em PDS"

Evento:

- ✓ Resultado da Revisão é decisão adiada, o PDS precisa ser melhorado.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Alterar os dados da atividade na pasta atividades.

Muda para o estado "projeto cancelado"

Evento:

- ✓ Resultado da revisão é PDS cancelado.

Estado "PDS aprovado":

Permanece no estado "PDS aprovado"

Evento:

- ✓ Responsável pela atividade "Desenvolver Plano de Iteração" não definido.

Muda para o estado "em Plano de Iteração"

Evento:

- ✓ Gerente define o responsável pela atividade "Desenvolver Plano de Iteração".

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta atividades.

Estado "em Plano de Iteração":

Permanece no estado "em Plano de Iteração"

Evento:

- ✓ Atividade não concluída.

Muda para o estado "Plano de Iteração concluído"

Evento:

- ✓ Atividade concluída, mas responsável pela atividade "Revisar Plano de Iteração" não definido.

Ações:

- ✓ Informar o gerente por meio de e-mail que ele deve informar o responsável pela execução da atividade "Revisar Plano de Iteração".

Muda para o estado "em revisão Plano de Iteração"

Evento:

- ✓ Atividade concluída e responsável pela atividade "Revisar Plano de Iteração" definido.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta revisões.

Estado "Plano de Iteração concluído":

Permanece no estado "Plano de Iteração concluído"

Evento:

- ✓ Responsável pela atividade "Revisar Plano de Iteração" não definido.

Muda para o estado "em revisão Plano de Iteração"

Evento:

- ✓ Gerente define responsável pela atividade "Revisar Plano de Iteração".

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Inserir os dados da atividade na pasta revisões.

Estado "em revisão Plano de Iteração":

Permanece no estado "em revisão Plano de Iteração"

Evento:

- ✓ Atividade não concluída.

Muda para o estado "em auditoria pelo GGQ"

Evento:

- ✓ Resultado da revisão é plano de iteração aprovado.

Ações:

- ✓ Informar o GGQ por meio de um e-mail que a auditoria pode ser realizada.
- ✓ Inserir os dados da atividade na pasta revisões.

Muda para o estado "em Plano de Iteração"

Evento:

- ✓ Resultado da Revisão é decisão adiada, o Plano de Iteração precisa ser melhorado.

Ações:

- ✓ Informar o responsável por meio de um e-mail, descrevendo os dados da atividade a ser realizada, a descrição da atividade e os *templates* a serem utilizados.
- ✓ Alterar os dados da atividade na pasta atividades.

Muda para o estado "projeto cancelado"

Evento:

- ✓ Resultado da revisão é Plano de Iteração cancelado.

Estado "em Auditoria pelo GGQ":

Permanece no estado "em auditoria pelo GGQ"

Evento:

- ✓ Atividade não concluída.

Muda para o estado "Projeto Concluído"

Evento:

- ✓ Auditoria realizada.

Ações:

✓ Concluir o projeto.

Nos estados, projeto concluído e projeto cancelado,a única ação possível é excluir o projeto.

8.6 Protótipo Implementado

Nesta seção serão apresentadas algumas das interfaces da ferramenta desenvolvida. Inicialmente, é apresentada a página inicial do sistema, exibida após o usuário identificar-se e inserir sua senha. A partir da página inicial, apresentada na Figura 8.6 é possível acessar as opções do sistema disponibilizadas ao usuário, clicando no menu à esquerda da página.

FIGURA 8.6 – Página Inicial do Sistema

O Gerente de Projeto pode cadastrar ou atualizar dados de um projeto por meio da opção Cadastrar Projeto, mostrada na Figura 8.7. O acompanhamento pode ser realizado por meio da opção Monitorar Projeto.

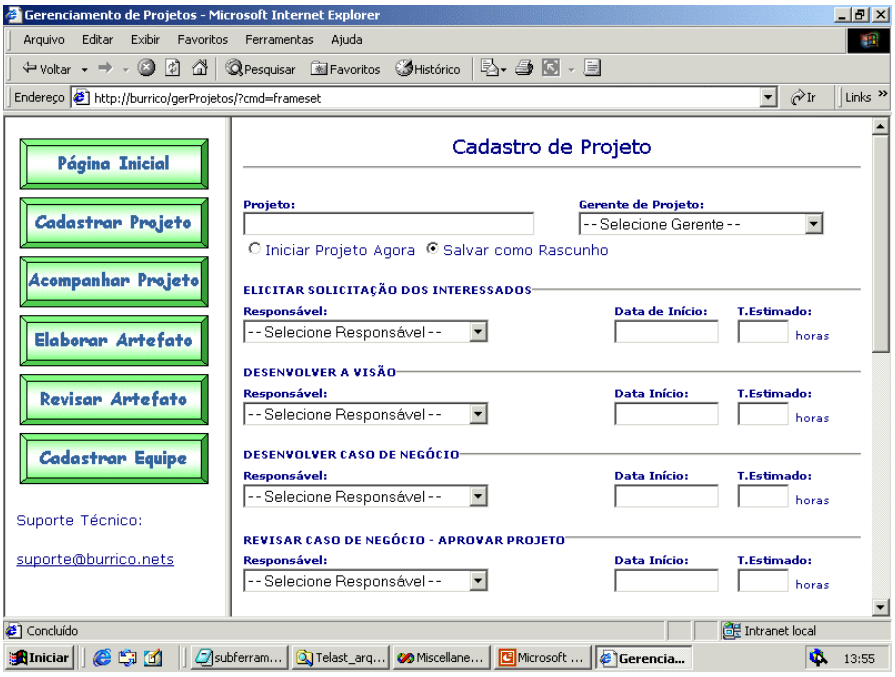


FIGURA 8.7 – Página Cadastrar Projeto

As atividades são separadas em duas categorias, que são: elaborar artefato e revisar artefato. Essa separação decorre do fato dessas atividades terem fluxos diferentes, como descrito no Diagramas de Estados.

A atividade de elaboração de um artefato depende apenas da conclusão da atividade e da existência do trabalhador responsável pela próxima atividade para determinar o próximo estado, enquanto que as atividades de revisão, dependem também do resultado da revisão. O resultado da revisão pode determinar um retorno a atividade anterior, objetivando melhorar o artefato que está sendo revisado; ou o cancelamento do projeto, ou então a aprovação do projeto.

Os dados dessas atividades também são diferentes, na atividade de revisão é necessário agendar a revisão, definindo participantes, hora e dia; e inserir o resultado da revisão. Na atividade de elaboração de artefato, é necessário apenas atualizar os dados da atividade.

A categoria elaborar artefato inclui as seguintes atividades: Elicitar Solicitação dos Interessados, Desenvolver a Visão, Desenvolver Caso de Negócio, Desenvolver PDS e Desenvolver Plano de Iteração. A categoria revisar artefato inclui as seguintes atividades: Revisar Caso de Negócio, Revisar Plano de Desenvolvimento de Software, Revisar Plano de Desenvolvimento de Software e Auditoria pelo GQS.

Quando o usuário seleciona a opção Elaborar Artefato, é apresentada a lista das atividades atribuídas a ele, como mostra a Figura 8.8.

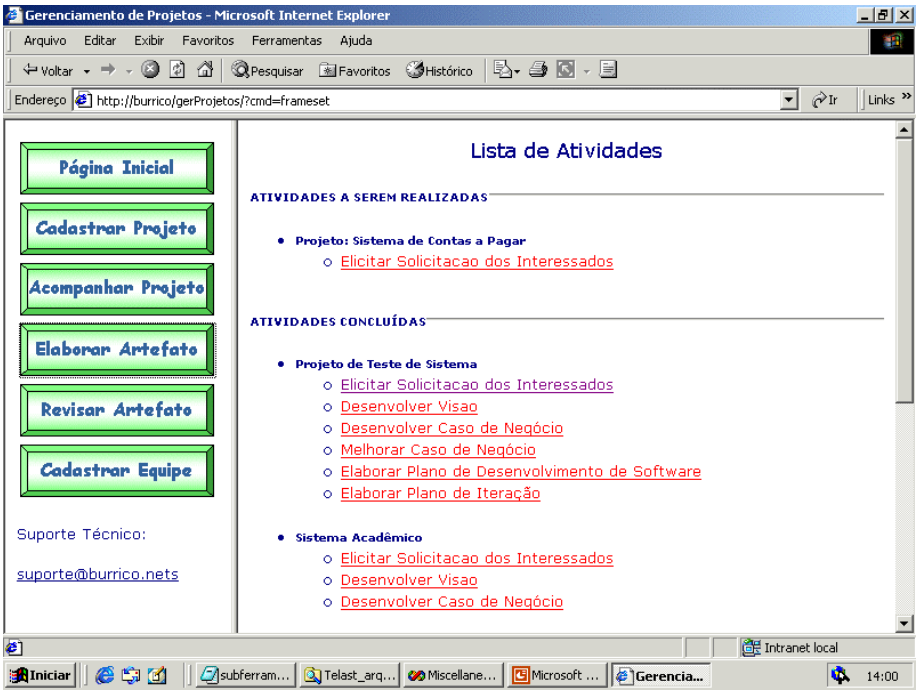


FIGURA 8.8 – Lista com as Atividades Atribuídas ao Usuário

Clicando na atividade é possível visualizar ou atualizar seus dados. Como mostra a Figura 8.9. Nesta mesma tela é possível visualizar a descrição da atividade ou o *template*, clicando nos *links* correspondentes a cada opção. A Figura 8.10 mostra a descrição da atividade e o *template* para a atividade *Elicitar Solicitação dos Interessados*.

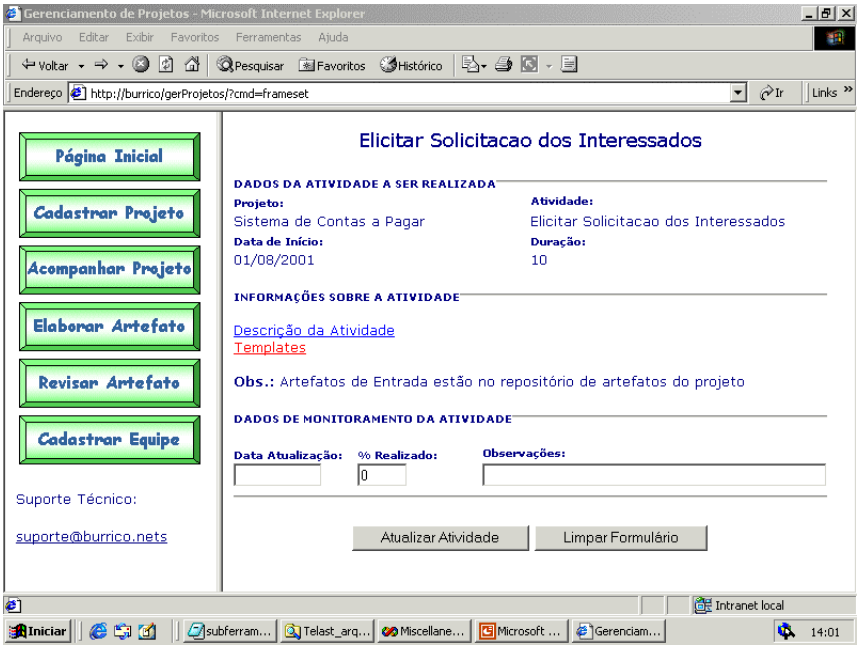


FIGURA 8.9 – Página de Atualização dos Dados das Atividades

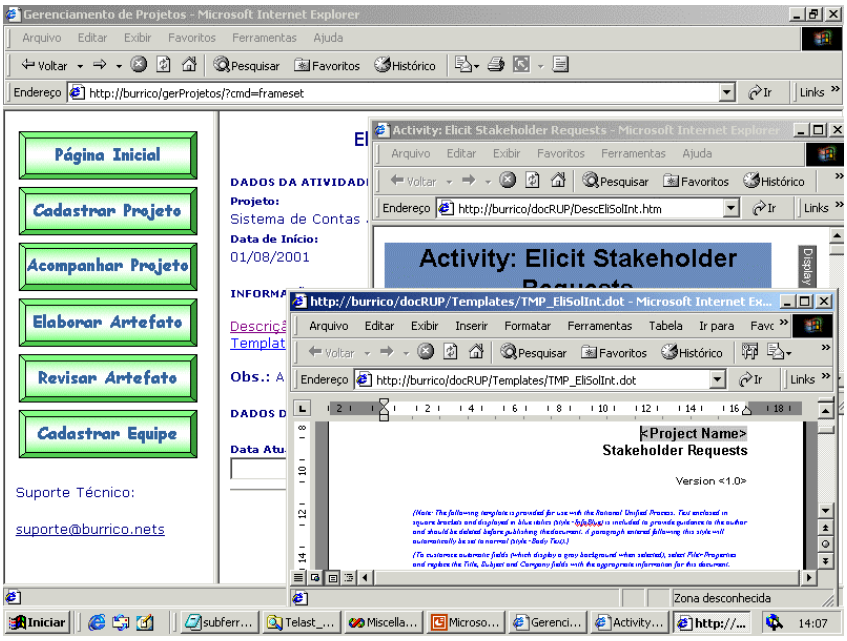


FIGURA 8.10 – Descrição e *Templates* associados a Atividade

A Figura 8.11 mostra a página onde o revisor do projeto deve agendar uma revisão, informando os participantes, a data e o horário da revisão. Quando o revisor do projeto confirma os dados agendados, os participantes são informados de sua participação na revisão por meio de e-mail.

FIGURA 8.11 – Página para Agendar Revisão

Após a realização da revisão, o revisor do projeto deve atualizar o resultado da revisão, para isto deve usar a página mostrada na FIGURA 8.12.

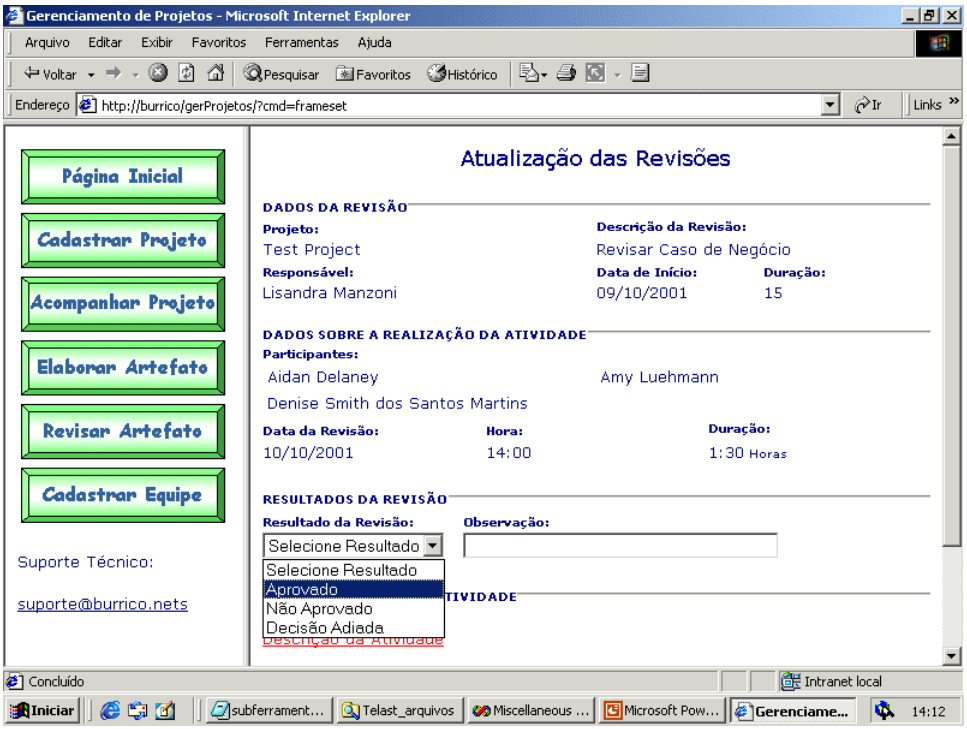


FIGURA 8.12 – Página para Registrar o Resultado da Revisão

8.7 Avaliação do Protótipo Implementado

O protótipo desenvolvido será avaliado com relação aos requisitos descritos na seção 8.7.1. A avaliação é descrita na seção 8.7.2.

8.7.1 Requisitos Desejados em um Ambiente de Suporte a Processos de Software

Um *Software Engineering Environment* (SEE) efetivo é uma coleção de capacidades integradas para suportar desenvolvedores e gerentes em seu trabalho [UCI 95]. Vários pesquisadores identificam os requisitos necessários a um ambiente para que ele suporte o processo de software [CHA 97, BAN 96, FIN 94, CHA 97a, UCI 95, CAL 96, CHA 99]. Nesta seção serão descritos alguns desses requisitos desejáveis, que servirão para avaliar o protótipo desenvolvido.

- ✓ **Suporte ao gerenciamento:** um ambiente deve fornecer visibilidade do projeto para suportar os gerentes nas tomadas de decisão e no controle sobre as atividades do projeto, deve também controlar o acesso aos dados do projeto e assinalar responsabilidades aos membros da equipe [UCI 95].

- ✓ **Suporte aos eventos do processo:** um SEE deve suportar eventos como notificar o usuário que é designado para executar determinada atividade do processo, disseminar as informações necessárias a execução da atividade, e coletar informações geradas como resultado da execução da atividade pelo usuário [UCI 95].
- ✓ **Flexibilidade:** as necessidades dos desenvolvedores e dos usuários são diversas variando de projeto para projeto e de tempos em tempos, com diferenças e alterações de cargos. É necessário que o ambiente possibilite alterações dinâmicas do processo [CHA 97].
- ✓ **Reusabilidade:** reusabilidade pode ser alcançada de muitas maneiras, como por exemplo: casos de projeto e padrões para as tarefas executadas constantemente, e modelos parametrizados e genéricos para as estruturas e comportamento que acontecem freqüentemente [CHA 97].
- ✓ **Colaboração:** o desenvolvimento de grandes sistemas de software requer a colaboração de diversos usuários, que não necessariamente estão localizados em um mesmo local. Um SEE deve suportar a interação e comunicação efetiva entre os usuários [CHA 97, BAN 96, UCI 95, CAL 99].
- ✓ **Fácil de Usar:** se a ferramenta é para apoiar os usuários, ela não deve se tornar um obstáculo. Os usuários devem entender as capacidades que a ferramenta oferece e como usá-la efetivamente [CAL 99].
- ✓ **Monitoramento:** monitoramento é crucial nas práticas de gerenciamento de hoje [CHA 97]. Coletar métricas e seguir um processo de desenvolvimento disciplinado são tarefas difíceis em qualquer projeto de software [CAL 99].
- ✓ **Consistência:** a consistência de dados compartilhados é importante em um ambiente de suporte a múltiplos usuários, onde os dados podem ser acessados de vários locais e em qualquer tempo [CHA 97].
- ✓ **Verificação:** a definição explícita do processo de software, permite que o mesmo seja revisto e melhorado.
- ✓ **Integração:** é a habilidade de ativar várias ferramentas que podem ser usadas no desenvolvimento de um produto de software, e capturar seus resultados [CHA 97, BAN 96].
- ✓ **Integração de dados:** refere-se a habilidade de ter em um único documento (hipertexto, hiperdocumento ou documento XML) todos os artefatos resultantes de uma seqüência de atividades do processo, permitindo a integração e o acesso fácil a esses artefatos, e a representação do conhecimento adquirido no processo de desenvolvimento. As diversas ferramentas usadas no processo poderiam compartilhar a informação e os artefatos armazenados em uma base de dados comum. Desde modo, a necessidade de um dicionário de dados, como proposto por Notari em [NOT 2000] e outros aparece. Esta ferramenta provê meios de armazenar dados de diferentes ferramentas em uma base de dados única (compartilhada por todas as ferramentas no ambiente de desenvolvimento).

8.7.2 Avaliando o Protótipo com Relação aos Requisitos Desejáveis

Para cada requisito será descrito como o protótipo o atende ou não.

Suporte ao gerenciamento

Usando a opção *Monitorar Projeto*, o gerente de projeto pode verificar o progresso das atividades (percentual realizado) e o estado atual do projeto, mas a ferramenta apresenta baixo suporte à tomada de decisão. Por exemplo, não fornece um relatório informando as atividades atrasadas, o impacto deste atraso no prazo final, etc. Isto significa que provavelmente o ambiente de desenvolvimento de software também precise integração com ferramentas de gerenciamento de projetos, como o Microsoft Project. Outros autores propõem esta integração entre sistemas de *workflow* e ferramentas de gerenciamento, como Bussler em [BUS 98]. A nova versão do protótipo incluirá esta integração.

O controle de acesso à ferramenta é feito por meio de *user-id* e senha. Quando o usuário acessa a ferramenta através da URL, o sistema mostra a tela de *Logon*, onde o usuário deve informar o *user-id* e a senha, cadastrados pelo gerente de projeto. O usuário pode acessar somente os dados as atividades atribuídas a ele, e são disponibilizadas as opções de acordo com os papéis que exerce.

O gerente usa a opção Planejar Projeto para assinalar responsabilidades pela execução de tarefas, e o servidor de colaboração envia e-mails informando os desenvolvedores quais as novas tarefas que foram atribuídas a eles, e mantém as listas de tarefas de todos os trabalhadores.

Suporte aos eventos do processo

Sempre que uma tarefa é atribuída a um trabalhador, uma mensagem contendo os dados da tarefa a ser executada, é enviada ao usuário. Estes dados incluem:

- ✓ Informações sobre a atividade: o projeto a que a atividade faz parte, descrição da atividade, data de início e tempo estimado para realização da atividade;
- ✓ Informações de suporte: *link* para o processo de desenvolvimento, onde são descritos os procedimentos a serem executados em cada atividade, guias de como executar a atividade e *tool mentors*, quando disponíveis; e *templates* a serem usados na execução da atividade.

Durante a realização da atividade, o trabalhador deve atualizar os dados da atividade que está executando. Estes dados incluem a data de atualização, o percentual realizado e observações sobre a atividade. Quando o percentual realizado atinge 100% o estado do processo de *workflow* é alterado. Para controlar efetivamente o processo de desenvolvimento é necessário coletar métricas, provendo o gerente de projeto com a visibilidade do projeto, e auxiliando-o na tomada de decisão.

Flexibilidade

Exchange 2000 Server permite atualizar o processo de *workflow* a qualquer momento, e as instâncias do processo seguirão este processo atualizado. O projeto dos dados pode ser ajustado conforme a aplicação evolui [MAR 2000].

O Exchange disponibiliza várias tecnologias de acesso a dados como ActiveX Data Objects (ADO), Collaboration Data Objects (CDO) for Exchange, and Extensible Markup

Language (XML) [MAR 2000, RIZ 2000], possibilitando ao desenvolvedor escolher a tecnologia que deseja utilizar.

O Microsoft Web Storage System é uma base de dados semi-estruturada que pode ser usada para armazenar, compartilhar, e gerenciar muitos tipos de dados, como por exemplo: mensagens eletrônicas, conteúdo *Web*, arquivos multimídia, documentos do Microsoft Office, etc [MAR 2000].

Exchange 2000 executa somente no sistema operacional Windows 2000, alguns dos serviços necessários ao Exchange, como o Active Directory são providos pelo Windows 2000 Server.

Reusabilidade

Os desenvolvedores podem criar páginas *Web* customizadas reutilizando componentes do Outlook Web Access, como por exemplo: caixa de entrada, calendário, contatos, etc. Estes componentes são embutidos em uma página *Web* [MAR 2000].

Desenvolvedores podem criar componentes COM em Visual Basic que são chamados nos scripts das páginas *Web*. Esses componentes podem interagir com os dados armazenados nos armazenamentos públicos e privados do Exchange, Active Directory e outros bancos de dados pelo uso de ADO e componentes COM CDOEX.

Cada tarefa está associada a um *template*, que é a estrutura pré-definida de um artefato. RUP associa com o artefato a descrição do *template* que pode ser usado para criar os artefatos correspondentes [RAT 2001].

Colaboração

O protótipo desenvolvido é uma ferramenta de suporte a processo baseada na *Web*, que oferece possibilidade de acessar informações compartilhadas através da Internet.

Acessando à ferramenta por meio de uma URL (*Universe Resource Locator*) específica, qualquer trabalhador pode realizar seu trabalho, seja atualizando os dados sobre as tarefas que está executando, cadastrando novos projetos, assinalando responsabilidades aos membros da equipe, programando revisões, etc.

Essa ferramenta não fornece suporte a colaboração por meio de *chat*, mostrando os usuário logados no sistema no momento..

Fácil de Usar

Essa ferramenta está estruturada como um *site* na *Web*, com *links* e páginas, facilitando seu uso.

Com o objetivo de auxiliar os trabalhadores na execução de suas atividades, a ferramenta associa cada tarefa com a descrição do procedimento a ser seguido para executar a atividade, associados a guias de trabalho e com os *templates* dos artefatos a serem desenvolvidos, facilitando a execução da atividade.

Guias de trabalho (*work guidelines*) apresentam as técnicas e conselhos práticos que são úteis ao trabalhador na execução do trabalho [RAT 2001]. *Templates* são modelos, ou protótipos, de artefatos [RAT 2001].

Monitoramento

Por meio da ferramenta é possível verificar a situação do projeto, em termos da execução das atividades. Na opção *Monitorar o Projeto* são listadas as atividades já concluídas, as não iniciadas e para as atividades em andamento é listado o percentual de conclusão da atividade atualizado pelo trabalhador responsável, juntamente com as datas previstas para conclusão.

É necessário automatizar a coleta de métricas para avaliar a qualidade do processo de desenvolvimento, e continuamente melhorar o processo de desenvolvimento de software.

Consistência

O controle dos dados é realizado pelo Web Storage System (WSS), os usuários interagem com o WSS, o qual, interage com os arquivos de dados, garantindo a consistência. WSS tem algumas das características de base de dados incluindo: transações, *backups*, indexação e armazenamento otimizado [MAR 2000].

Verificação

A modelagem do processo de software em um sistema de gerenciamento de *workflow* demanda que o processo seja explicitamente definido. Tendo um processo formalmente definido é possível verificar a consistência do processo e avaliar e melhorar este processo.

Integração

O protótipo desenvolvido está integrado com as ferramentas Microsoft Office. É necessário integrar com outras ferramentas, como: ferramentas para modelagem, ferramentas para gerenciamento de configuração, compiladores, ferramentas para teste, etc.

Integração de Dados

O protótipo desenvolvido não fornece integração de dados. Uma maneira simples de implementar esta característica é introduzir uma função no protótipo que cria um documento HTML ou XML, contendo *links* para os artefatos criados durante um projeto, e disponibilizando um banco de dados onde esses artefatos estão armazenados e são compartilhados pelos membros da equipe do projeto e pelas ferramentas.

A Tabela 8.3 mostra um resumo dos requisitos suportados e não suportados pelo protótipo desenvolvido.

TABELA 8.3 – Requisitos Suportados e Não-Suportados pelo Protótipo Desenvolvido

Requisitos	**Implementado por**	**Suportado**	**Não Suportado**
Suporte ao gerenciamento	AGP (Ambiente de Gerenciamento de Projetos)	-Monitorar projeto. -Controle de Acesso.	-Suporte a tomada de decisão.
Suporte aos eventos do processo	AGP	-Enviar e-mail com os dados da tarefa ao usuário. -Mudança de estado.	-Coletar métricas, e disparar eventos baseados na análise dos dados medidos.
Flexibilidade	Exchange 2000 Server	-Atualizar o processo modelado no WMS a qualquer momento. -Muitas tecnologias de acesso a dados disponível. -WSS é uma base de dados semi-estruturada.	-Exchange 2000 é dependente do sistema operacional Windows 2000 Server.
Reusabilidade	Exchange 2000 Server	-Reuse de componentes do Outlook Web Access e componentes COM.	
	AGP	-Use de modelos para criar os artefatos.	
Colaboração	Exchange 2000 Server	- Acesso através de URL.	
	AGP	- Compartilhamento de informação através da Internet.	-Suporte a colaboração por meio de *chat*. -Não suporta atividades cooperativas.
Fácil de usar	AGP	-Interface Intuitiva (*links* para páginas). -Tarefas são associadas com guias e modelos.	-Configuração da interface pelo usuário, definindo linguagem, cores, menus, fontes, etc.

Fácil de usar	Exchange 2000 Server	-Ferramenta gráfica para definição do processo.	
Monitoramento	AGP	-Mantém dados de acompanhamento das tarefas.	-Coleta automática de métricas.
Consistência	Exchange 2000 Server	- WSS controla transações, *backups*, indexação e armazenamento otimizado.	
Verificação	AGP	- Verificação e melhoria do processo, baseado em sua definição.	
Integração	Exchange 2000 Server	- Integração com ferramentas do Office.	- Integração com outras ferramentas, como ferramentas de modelagem, testes, gerenciamento de projetos, ferramentas da Rational, etc.
Integração de dados			- Prover um banco de dados compartilhado para os membros da equipe e ferramentas. - Criar documentos HTML ou XML que contenham *links* para os artefatos do projeto.

9 Conclusão e Trabalhos Futuros

Esta última seção apresenta as conclusões obtidas no desenvolvimento neste trabalho. Ao final deste capítulo são apresentadas algumas propostas de trabalhos futuros com base nos resultados obtidos neste trabalho.

9.1 Principais Propostas deste Trabalho

Este trabalho descreve a avaliação do Processo Unificado Rational (RUP) baseado no Modelo de Maturidade da Capacitação (CMM ou SW-CMM), a extensão do Processo Unificado para incluir funcionalidades não cobertas pelo RUP, e a utilização de um Sistema de Gerência de *Workflow* comercial para elaborar um protótipo de ambiente de apoio ao processo de desenvolvimento.

9.1.1 Avaliação do Processo Unificado

O Processo Unificado Rational suporta as melhores práticas para desenvolvimento de software, cobrindo todo o ciclo de vida do software. O Processo Unificado aumenta a produtividade das equipes de desenvolvimento, por fornecer aos membros da equipe acesso fácil a base de conhecimento por meio de guias e modelos, para as atividades mais críticas de desenvolvimento de software. Ele é um guia de como efetivamente utilizar a Linguagem de Modelagem Unificada (UML), e é suportado por ferramentas, que automatizam várias partes do processo [RAT 98].

CMM apresenta uma abordagem de melhoria de processo de desenvolvimento bem aceita pela comunidade de engenharia de software [PAU 96], inclusive influenciando várias abordagens subseqüentes a ele. Seus níveis de maturidade, áreas-chave de processo, metas, e práticas-chave têm sido extensivamente discutidos e revisados, dentro da comunidade de software. Estendendo RUP para satisfazer as práticas-chave do CMM significa usar o conhecimento adquirido pelo SEI, nestes anos de melhoria de processo para avaliar e estender um modelo de processo, que está cada vez mais interessando as empresas que desenvolvem software.

A avaliação do Processo Unificado Rational baseado no modelo CMM mostra que alguns dos aspectos do processo de software não são suportados pelo RUP.

O Processo Unificado apresenta uma abordagem bem-definida em gerenciamento de projetos e processos de engenharia de software, mas sua abordagem não se concentra em atividades de gerenciamento de sistemas. Ele apresenta lacunas em atividades envolvendo gerenciamento de recursos humanos, gerenciamento de custos e gerenciamento de aquisição.

Atividades relacionadas ao gerenciamento de subcontratação e programa de treinamento não são suportadas pelo RUP. Salienta-se que recursos bem-treinados e capazes de executar suas atividades são imprescindíveis ao bom andamento de um projeto. RUP também não descreve atividades para melhorar o processo de software da organização, apenas descreve uma avaliação do processo, mas não deixa claro como os

resultados da avaliação do processo da organização alimentam a melhoria contínua do processo.

As áreas-chave do CMM melhor suportadas pelo RUP pelo são *Gerenciamento de Configuração*, com uma cobertura de 85,7% das práticas-chave recomendadas, *Gerenciamento de Requisitos* com 83,3%, *Planejamento de Projetos de Software* com 80,0%, *Acompanhamento e Supervisão de Projetos* com 83,3%*, Definição dos Processos da Organização* com 81,8% e *Engenharia de Produto de Software* com 80%.

RUP oferece um bom suporte para as áreas-chave de *Gerenciamento Integrado de Software* suportando 78,9% das práticas-chave*,* e *Coordenação Intergrupos*, cobrindo 76,4% das práticas-chave descritas pelo CMM nesta área-chave de processo.

RUP oferece pouco suporte para as práticas-chave de *Garantia de Qualidade de Software* com 58,8%, *Foco nos Processos da Organização* com 43,7% e *Revisão por Pares* com 66,6%. Salienta-se que o baixo percentual alcançado nesta última área-chave de processo decorre do fato de muitas práticas-chave dessa área-chave encontrarem-se na característica-comum Habilidade de Executar, considerada fracamente atendida pelo RUP.

As áreas-chave de processo *Gerenciamento de Subcontratação* e *Programa de Treinamento* não são suportadas pelo Processo Unificado Rational.

A avaliação do RUP demonstrou que este modelo de processo suporta a maioria das práticas-chave descritas pelo CMM níveis 2 e 3, e identificou as práticas-chave não suportadas pelo RUP.

A identificação das atividades do Processo Unificado que satisfazem as práticas-chave do CMM pode servir como "alertas" no momento da customização do Processo Unificado para projetos específicos, isto é, as atividades consideradas importantes para atender o CMM e citadas durante a avaliação não deveriam ser eliminadas ou alteradas no momento dessa customização.

9.1.2 Propostas de Extensão do RUP para Alcançar Níveis 2 e 3 de CMM

Com o objetivo de compatibilizar o RUP com o CMM, e com base na avaliação realizado, foram elaboradas algumas propostas de extensão ao RUP. Essas propostas têm o objetivo de fornecer "dicas" de como as práticas-chave podem ser implementadas.

Cada proposta elaborada é descrita em termos das atividades que compõem a proposta. Para cada atividade são descritos os passos da atividade, o trabalhador que deve executá-la e os artefatos de entrada e saída, seguindo o modelo descrito pelo RUP.

Nas propostas de macro-atividade, o fluxo das atividades é descrito por meio de diagramas de atividades [BOO 98].

Em cada proposta são descritas referências bibliográficas que podem ser úteis para melhorar as propostas descritas deste trabalho, e talvez até elaborar modelos para os artefatos e guias associados às atividades.

9.1.3 Protótipo do Ambiente de Apoio ao Processo Implementado

Considerando que o desenvolvimento de aplicações de software raramente é executado por um único trabalhador, demandando a participação de equipes de pessoas, é importante prover uma ferramenta que facilite a colaboração entre os membros de uma equipe de desenvolvimento.

No capítulo 7 deste trabalho, discutiu-se a potencialidade do uso de sistemas de gerência de *workflow* para apoio a processos de software. Essa discussão resultou na implementação de um protótipo de um ambiente de gerenciamento de projetos.

O processo modelado no sistema de *workflow*, no caso o Microsoft Exchange 2000 Server, foi baseado no Processo Unificado estendido para contemplar nível 2 e 3 de maturidade do CMM.

A utilização de sistemas de *workflow* comerciais para construção de ambientes de apoio ao processo de software pode ser uma forma de facilitar a implementação desses ambientes. A implementação de um ambiente de apoio ao processo pode ser uma tarefa complexa, cara e demorada; visto que estes ambientes possuem características como: gerenciamento de processo, integração com ferramentas, atribuição de atividades e monitoramento de projetos. Utilizando-se um sistema de *workflow* comercial, várias dessas características já estão incorporadas na ferramenta, e podem ser utilizadas de forma transparente, possibilitando ao usuário se dedicar, somente, na definição do processo que deseja modelar.

O Exchange 2000 Server foi o servidor de colaboração escolhido para este experimento, porque algumas organizações já o utilizam como servidor de e-mail, e poderiam passar a utilizá-lo também como servidor de *workflow,* não necessitando adquirir um novo produto. Exchange é integrado com as ferramentas do Microsoft Office, e essas ferramentas são bastante utilizadas no processo definido para o experimento.

O protótipo desenvolvido foi avaliado baseado nos requisitos recomendados por alguns projetos [CHA 97, BAN 96, FIN 94, CHA 97a, UCI 95, CAL 96, CHA 99].

A ferramenta desenvolvida ajuda no suporte ao gerenciamento de projetos, provendo opções para monitorar o projeto, e suportar os eventos do processo, como mudanças de estado e comunicação entre os usuários, por meio das tarefas atribuídas a eles.

Exchange 2000 Server usa o Web Storage System, que é um banco de dados semi-estruturado, que possibilita o armazenamento de documentos, arquivos multimídia (vídeos e *voice mail*), contatos, etc.

AGP é uma ferramenta que pode ser acessada através da Internet, suportando a colaboração entre membros de uma equipe, e oferecendo os benefícios da Web, tais como: navegação fácil e intuitiva por meio de *links* e páginas, facilidade de acesso, etc.

O Exchange 2000 Server facilita a modelagem dos processos por disponibilizar uma ferramenta gráfica. Nessa ferramenta é possível especificar os estados do *workflow*, as transições de estados e as ações que devem ser disparadas a cada transição de estado.

Por outro lado, várias tarefas devem ser programadas, como a inserção e recuperação de dados do Web Storage System, envio de mensagens, ativação de ferramentas, etc.

Na próxima seção são descritos alguns trabalhos futuros, que visam melhorar o protótipo desenvolvido.

9.2 Trabalhos Futuros

O protótipo descrito neste trabalho deve ser melhorado para suportar os requisitos considerados importantes a um ambiente de apoio a processos de software que, segundo a avaliação realizada, não estão contemplados.

Dentre esses requisitos cita-se: coletar métricas, integrar ao ambiente outras ferramentas utilizadas durante a execução do processo de desenvolvimento, integrar os artefatos desenvolvidos em uma base de dados, possibilitando que diferentes ferramentas acessem esses dados, e criar documentos dinâmicos, usando HTML, ASP ou XML; com *links* para os artefatos do projeto.

A coleta automática de métricas representa um importante meio para o controle e o gerenciamento de projetos, por indicar o status do projeto. Uma avaliação do status do projeto é crítica para a realocação de recursos, para ajuste de cronograma e para medição da qualidade [CAL 96]. Os dados do projeto coletados podem ser usados em estimativas de projeto futuras. Para cada área-chave de processo do CMM, estão descritas uma ou mais práticas na característica-comum Medição e Análise. Essas práticas se referem à medição e análise de dados sobre a área-chave de processo em que está descrita.

Identifica-se a necessidade de integrar ao protótipo desenvolvido outras ferramentas utilizadas durante a execução do processo de desenvolvimento, tais como: ferramentas para modelagem, gerenciamento de configuração, teste, codificação, etc. Essa funcionalidade já está sendo implementada em um dos trabalhos do Projeto AMADEUS [http//www.inf.ufrgs.br/Amadeus/index.html], por Betemps.

Outra funcionalidade que pode ser implementada é o compartilhamento das informações e dos artefatos do projeto entre as diferentes ferramentas. Para isto, é necessário que os artefatos sejam armazenados em uma base de dados comum, e o acesso a esses dados pode ser realizado por meio de um dicionário de dados, como o proposto por Notari [NOT 2000].

A enciclopédia (ED-ADS), proposta por Notari [NOT 2000], fornece um conjunto de serviços para que as ferramentas em um ambiente de desenvolvimento de software possam definir metadados e armazenar seus dados.

O ED-ADS mantém dois repositórios: (1) um repositório de metadados (o próprio dicionário de dados) e (2) um repositório de dados (armazena os dados manipulados pelas ferramentas de um ambiente de desenvolvimento de software).

Extensible Markup Language (XML) pode ser uma alternativa para compartilhamento dos dados. Por ser uma linguagem mais flexível que HTML, possibilita ao usuário definir suas próprias *tags*. Isto significa que XML torna possível adaptar os

documentos XML para diferentes necessidades, e também usar XML para representar diferentes tipos de dados.

A utilização do Processo Unificado Estendido, em projetos reais de organização que desejam alcançar níveis 2 e 3 de CMM, pode validar a avaliação realizada e os resultados descritos neste trabalho, e pode possibilitar inclusive uma melhoria das propostas de extensão.

Outros trabalhos que podem ser realizados são a elaboração de modelos para os artefatos propostos e a descrição de guias para as atividades de extensão propostas.

10 Bibliografia

[ALC 99] ALCÂNTARA, Andréa A. et al. Uso do Processo RUP na Implantação da ISO 9000-3, In: WORKSHOP DE QUALIDADE DE SOFTWARE, 6., 1999, Florianópolis. **Anais…** Florianópolis: UFSC, 1999. p. 60-71.

[AMA 97] AMARAL, Vinícius L.; GRALA, Anderson S.; LIMA, José Valdeni de. Workflow e Gerência de Documentos. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 16., 1997. **Anais…** Brasília: UnB, 1997. p. 401-448.

[ARA 99] ARAUJO, Renata M.; BORGES, Marcos R. S. Sobre a Aplicabilidade de Workflow no Suporte a Processos de Software. In: WORKSHOP IBEROAMERICANO DE ENGENHARIA DE REQUISITOS E AMBIENTES DE SOFTWARE, 2., 1999. **Anais…**Costa Rica: TEC, 1999. p. 417-428.

[ARG 91] ARGYRIS, Chris. Teaching Smart People How to Learn. **Harvard Business Review**, Boston, p. 99-109, May/Jun. 1991.

[ART 00] ARTECH CONSULTORES S.R.L.. **Genexus Process Modeler:** User's Guide, Version 6.1. Chicago, 2000.

[AUS 96] AUSTIN, Robert D. **Measuring and Managing Performance in Organizations**. New York: Dorset House Publishing, 1996. ISBN: 0-932633-36-6

[BAN 96] BANDINELLI, Sergio; DI NITTO, Elisabetta; FUGGETTA, Alfonso. Supporting Cooperation in the SPADE-1 Environment**. IEEE Transaction on Software Engineering**, New York, v.22, n.12, p.841-865, Dec. 1996.

[BAR 00] BARNES, Anthony; GRAY, Jonathan. COTS, Workflow, and Software Process Management: An Exploration of Software Engineering Tool Development, In: AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE, 2000. **Proceedings…** New York: IEEE Computer Society Press, 1998.

[BAS 84] BASILI, Vitor R; WEISS, David M. A Methodology for Collecting Valid Software Engineering Data, **IEEE Transactions on Software Engineering**, New York, v. SE-10, n.6, p. 728-738, Nov. 1984..

[BEL 94] BELL CANADÁ. **Trillium:** Model for Telecom Product Development and Support Process Capability, release 3.0. Montreal, 1994.

[BOE 99] BOEGH, Jorgen et al. A Method For Software Quality Planning, Control, and Evaluation. **IEEE Software**, New York, v.16, n.2, p. 69-85, Mar. 1999.

[BOO 94] BOOCH, Grady. **Object-Oriented Analysis and Design with Applications**. 2nd ed. California: The Benjamin/Cummings Publishing Company, 1994.

[BOO 98] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Users Guide**. Massachusetts: Addison Wesley, 1998.

[BUC 84] BUCKLEY, Fletcher; POSTON, Robert. Software Quality Assurance**. IEEE Transactions on Software Engineering**, New York, v. SE-10, n. 1, p. 36-41, Jan. 1984.

[BUC 87] BUCKLEY, Fletcher. The Roles of a SQA Person. **ACM Software Engineering Notes**, New York, v. 12, n. 3, p. 42-44, Jul. 1987.

[BUS 98] BUSSLER, Christoph. Workflow Instance Scheduling with Project Management Tools. In: INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 9., 1998. **Proceedings…** New York: IEEE Computer Society Press, 1998.

[CAL 96] CALLAHAN, John; RAMAKRISHNAM, Sudhaka. Software Project Management and Measurement on the World-Wide-Web (WWW). In: WORKSHOP ON ENABLING TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISES, 5., 1996. **Proceedings…** New York: IEEE Computer Society Press, 1996.

[CAR 92] CARLETON, Anita D. et al. **Software Measurement for DoD Systems:** Recommendations for Initial Core Measures. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1992. (CMU/SEI-92-TR-19).

[CHA 97] CHAN, Daniel K. C.; LEUNG, Karl R. P. H. A Workflow Vista of the Software Process, In: INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 8., 1997. **Proceedings…** New York: IEEE Computer Society Press, 1997.

[CHA 97a] CHAN, Daniel K. C.; LEUNG, Karl R. P. H. Software Development as a Workflow Process, In: ASIA-PACIFIC SOFTWARE ENGINEERING AND INTERNATIONAL COMPUTER SCIENCE CONFERENCE, 4., 1997. **Proceedings…** New York: IEEE Computer Society Press, 1997.

[CHA 99] CHAN, Daniel K. C. Form Management in the Vicomte Workflow System, In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 32., 1999. **Proceedings…** New York: IEEE Computer Society Press, 1999.

[ELL 96] ELLMER, Ernst; MERKL, Dieter. Defining a Set of Criteria for the Assessment of Tool Support for CMM-based Software Process Improvement. In: INTERNATIONAL SYMPOSIUM ON ASSESSMENT OF SOFTWARE TOOLS, 4., 1996. **Proceedings…** New York: IEEE Computer Society Press, 1996.

[FER 96] FERGUSON, Jack et al. **Software Acquisition Capability Maturity Model (SA-CMM).** Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1996. (CMU/SEI-96-TR-020).

[FIN 94] FINKELSTEIN, Anthony; KRAMER, Jeff; NUSEIBECH, Bashar. **Software Process Moddeling and Technology**. New: York: John Wiley & Sons, 1994.

[FIT 99] FITZGERALD, Brian; O'KANE, Tom. A Longitudinal Study of Software Process Improvement. **IEEE Software**, New York, v.16, n.3 p. 37-45, May 1999.

[GAR 93] GARVIN, David A. Building a Learning Organization. **Harvard Business Review**, Boston, p. 78-91, Jul./Aug.1993.

[GEO 95] GEORGAKOPOULOS, Diimitrios; HORNICK, Mark; SHETH, Amit. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. In: DISTRIBUTED AND PARALLEL DATABASES, 3., 1995. **Proceedings…** Boston: Kluwer Academic Publishers, 1995.

[GIM 94] GIMENES, Itana M. S. O Processo de Engenharia de Software: Ambientes e Formalismos. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 14., 1994. **Proceedings…** Caxambu: SBC, 1994.

[GIM 2000] GIMENES, Itana M. S.; TANAKA, Sergio A.; OLIVEIRA, José P. M. de. An object Oriented Framework for Task Scheduling. In: TOOLS EUROPE, 2000, **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2000.

[GRA 92] GRADY, Robert B. **Practical Software Metrics For Project Management and Process Improvement**. Englewood Cliffs: Prentice Hall, 1992. ISBN: 0137203845.

[GRA 97] GRADY, Robert B. **Successful Software Process Improvement**. Englewood Cliffs: Prentice Hall, 1997. ISBN: 0136266231.

[GRE 96] GRESSE, Christiane; ROMBACH, Dieter; RUHE, Günther. A Practical Approach For Building GQM-Based Measurement. In: BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 10., 1996. São Carlos, 1996. **Proceedings…** São Carlos: UFSCAR, 1996.

[HAA 94] HAASE, Volkmar et al. Bootstrap: Fine-Tuning Process Assessment. **IEEE Software**, New York, v.11, n.4 p. 25-35, Jul. 1994.

[HEN 99] HENDERSON-SELLERS, Brian; DUÉ, Richard; GRAHAM, Ian. Third Generation OO Processes: A Critique of RUP and OPEN from a Project Management Perspective. In: TOOLS WORKSHOP ON PROJECT MANAGEMENT OF OBJECT-ORIENTED DEVELOPED SYSTEMS, 1999. **Proceedings…** [S.l.: s.n.], 1999.

[HEN 2000] HENDERSON-SELLERS, Brian. Object-Oriented Methods and Processes, Software Methods and Tools In: INTERNATIONAL CONFERENCE ON SOFTWARE METHODS AND TOOLS, 2000. **Proceedings…** New York: IEEE Computer Society Press, 2000.

[HER 94] HERBSLEB, James et al. **Benefits of CMM-Based Software Process Improvement Initial Results**. Software Engineering Institute, Carnegie Mellon University, 1994. (CMM/SEI-94-TR-13).

[HUM 90] HUMPHREY, Watts S. **Managing the Software Process**. [S.l.]: Addison Wesley, 1990.

[ISO 91] INTERNATIONAL STANDARDS ORGANIZATION. **ISO 9000-3 Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software**, ISO 9000-3. Geneva, 1991

[ISO 95] INTERNATIONAL STANDARDS ORGANIZATION. **ISO/IEC 12207 Information technology - Software life-cycle processes**, ISO/IEC 12207. Geneva, 1995.

[ISO 95a] INTERNATIONAL STANDARDS ORGANIZATION. **SPICE Software Process Assessment - Part 1: Concepts and Introductory Guide**, ISO 15504. Geneva, 1995.

[JAC 92] JACOBSON, Ivar et al. **Object-Oriented Software Engineering – a Use Case-Driven Approach**. Wokingham: Addison-Wesley, 1992.

[JAC 98] **Jacobson**, Ivar; BOOCH, Grady; RUMBAUGH, James. **The Unified Software Development Process**. [S.l.]: Addison Wesley, 1998.

[KRI 99] KRISHNAN, Mayuram S.; KELLNER, Marc. Measuring Process Consistency: Implications for Reducing Software Defects. **IEEE Transactions on Software Engineering**, New York, v. 25, n.6, p. 800-815, Nov. 99.

[KRU 2000] KRUCHTEN, Philippe. **The Rational Unified Process:** An Introduction. Massachusetts: Addison Wesley, 2000.

[LAV 98] LAVAZZA, Luigi. **Providing automated support for the GQM Measurement Process**. Milano: CEFRIEL, 1998. (RT 98005).

[MAN 96] MANGAN, Marco; ZIELINSKI, Jan; IOCHPE, Cirano; BRITTO. Eduardo. **First Report on Modelling *Software* Development through Project and Workflow Management with Integrated Human Interactions**. Porto Alegre: CPGCC/UFRGS, 1996. (RP 264).

[MAR 2000] MARTIN, Mindy. **Programming Collaborative Web Applications With Microsoft Exchange 2000 Server**. Redmond: Microsoft Press, 2000.

[MAM 2000] MARRA, Marci; WHITCOMB, Valerie; YODER, Doug. **Workflow Designer for Exchange: Automating Workflow on Exchange Folders, an End-to-End Developer Walkthrough**. [S.l.]: Microsoft Corporation, 2000.

[MAU 99] MAURER, Frank. et al. Software Process Support Over the Internet. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 21, 1999,. **Proceedings…** New York: IEEE Computer Society Press, 1999.

[NGU 97] NGUYEN, Minh N.; WANG, Alf I.; CONRADI, Reidar. Total Software Process Model Evolution in EPOS Experience Report. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 7., 1997, Boston. **Software Configuration Management**: proceedings. Berlin: Springer-Verlay, 1997. p. 17-23.

[NIE 96] NIELSEN, Jim; MILLER, Ann. Selecting Software Subcontractors. **IEEE Software**, New York, v. 13, n. 4, p. 104-109, Jul. 1996.

[NOT 2000] NOTARI, Daniel. **Uma Enciclopédia de Ambientes Distribuídos de Desenvolvimento de Software**. 2000. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[OCA 98] OCAMPO, Camilo; BOTELLA, Pere. **Some Reflections on Applying Workflow Technology to Software Process**. Barcelona: Departament de Lenguages i Sistemes Informàtics, 1998. (Report LSI-98-5-R)

[ORA 2001] ORACLE CORPORATION. **Oracle Workflow**. Em <ttp://technet.oracle.com/docs/products/oracle8i/doc_library/817_doc/ois.817/a85440/wf/wftop.htm>. Acesso em 26 jul. 2001.

[PAU 93] PAULK, Mark C. et. al. **Key Practices of the Capability Maturity Model**, Version 1.1. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1993. (CMU/SEI-93-TR-025).

[PAU 93a] PAULK, Mark C. **Capability Maturity Model for Software**, Version 1.1. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1993. (CMU/SEI-93-TR-024).

[PAU 96] PAULK, Mark C. Effective CMM-Based Process Improvement, In: INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, 16., 1996. **Proceedings…** New York: IEEE Computer Society Press, 1996.

[PAU 99] PAULK, Mark. Using the Software CMM With Good Judgment. **ASQ Software Quality Professional**, [S.l.], v.1, n. 3, p. 19-29, Jun. 99.

[PMI 2000] PROJECT MANAGEMENT INSTITUTE. **A Guide to the Project Management Body of Knowledge**. Pennsylvania, 2000.

[RAT 98] RATIONAL SOFTWARE CORPORATION. **Rational Unified Process Best Practices for Software Development Teams**. Cupertino, 1998.

[RAT 99] RATIONAL SOFTWARE CORPORATION. **Rational Unified Process,** Version 5.5. Cupertino, 1999.

[RAT 2001] RATIONAL SOFTWARE CORPORATION**. Rational Unified Process,** Version 2001.03.00. Cupertino, 2001.

[RAT 2001a] RATIONAL SOFTWARE CORPORATION. **Assessing the Rational Unified Process against ISO/IEC15504-5: Information Technology – Software Process Assessment Part 5: An Assessment Model and Indicator Guidance**. Em <http://www.rational.com/products/whitepapers/rup_iso.jsp>. Acesso em: 26 set. 2001.

[RIZ 2000] RIZZO, Thomas. **Programming Microsoft Outlook and Microsoft Exchange 2000 Server**. 2nd ed., Redmond: Microsoft Press, 2000.

[RUM 91] RUMBAUGH, James et al. **Object-Oriented Modeling and Design**, Upper Saddle River: Prentice-Hall, 1991.

[SEI 2000] SOFTWARE ENGINEERING INSTITUTE. **Capability Maturity Model.** Em <http://www.sei.cmu.edu>. Acesso em: 30 de jun. 2000.

[TSU 96] TSUKUMO, Alvaro N. et al. Modelos de Processo de Software: Visão Global e Análise Comparativa. In: CONFERÊNCIA INTERNACIONAL DE TECNOLOGIA DE SOFTWARE, 7., 1996. **Anais…** Curitiba: [S.n.], 1996.

[UCI 95] UC IRVINE INFORMATION AND COMPUTER SCIENCE DEPARTMENT. **The Arcadia Project. The Arcadia Research Project**. Em <http://www.ics.uci.edu/Arcadia/>. Acesso em: 26 set. 2001.

[ULT 98] ULTIMUS. **Open Workflow Architecture and Industry Standards**. North Carolina, 1998.

[ULT 98a] ULTIMUS. **Workflow, Groupware, and the role of Ultimus**. North Carolina, 1998.

[UNI 2000] UNIWAY. **Uniway Compete Usando Ferramenta Rational**. Em <http://www.novomilenio.inf.br/ano00/0003b018.htm>. Acesso em: 01 ago. 2001.

[VER 97] VERAART, V. A.; WRIGHT, S.L. Dukas: A Software Process Task Management Environment. In: AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE, 1997. **Proceedings…** New York: IEEE Computer Society Press, 1997.

[WFM 99] WORKFLOW MANAGEMENT COALITION. **Terminology and Glossary**. Winchester, 1999.

[WIG 90] WIGGENHORN, William. Motorola U: When Training Becomes an Education. **Harvard Business Review**, Boston, p. 71-83, Jul/Aug. 1990.