GERÊNCIA DE PROCESSOS DE SOFTWARE PARA EQUIPES GEOGRAFICAMENTE DISPERSAS

Carmen Lúcia Lodi Maidantchik

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profa. Ana Regina C. da Rocha, D.Sc.
COPPE/UFRJ

Prof. Geraldo B. Xexéo, D.Sc.
IM/UFRJ

Prof. Jano Moreira de Souza, Ph.D.
IM-COPPE/UFRJ

Prof. José Carlos Maldonado, D.Sc.
USP/SC

Prof. Walcélio Melo, D.Sc.
UCB

Prof. Zieli Dutra Thomé Filho, D.Sc.
COPPE/UFRJ

RIO DE JANEIRO, R.J. - BRASIL

JUNHO DE 1999

MAIDANTCHIK, CARMEN LÚCIA LODI

Gerência de Processos de Software para Equipes Geograficamente Dispersas [Rio de Janeiro] 1999

XVI, 213 p. 29.7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1999)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Processos de Software 2. Modelos de Maturidade 3. World-Wide Web

I. COPPE/UFRJ II. Título (série)

à minha mãe.

Agradecimentos

À orientadora e amiga Ana Regina pelo seu carinho, apoio e incentivo.

Ao Xexéo pelas discussões filosóficas e sua amizade.

Ao Zieli pelo seu trabalho junto à Colaboração entre a UFRJ e o CERN.

A Jano, Maldonado e Walcélio por me darem a satisfação de terem aceito participar da banca.

À COPPE/Oceânica e, principalmente, ao Protásio por apoiar e incentivar a minha carreira profissional.

Ao pessoal administrativo da COPPE/Sistemas, Ana Paula, Cláudia e Solange.

Ao Calôba, Marroquim e Seixas pelo excelente trabalho junto à Colaboração entre a UFRJ e o CERN.

Ao Peter Jenni, chefe do ATLAS, pelo seu constante apoio.

Aos meus amigos do doutorado e mestrado pelas conversas e trocas de informações.

À "rede internacional" Anna, Catherine, Giuliana, Jane e Luisella pelas nossas conversas eletrônicas.

Aos meus amigos que conheci no CERN e que, agora, nos encontramos geograficamente dispersos.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

GERÊNCIA DE PROCESSOS DE SOFTWARE PARA EQUIPES GEOGRAFICAMENTE DISPERSAS

Carmen Maidantchik

Junho de 1999

Orientadores:

Prof. Ana Regina Cavalcanti da Rocha

Prof. Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Esta tese apresenta um modelo de gerência de processos de software para equipes geograficamente dispersas. As metas do modelo proposto são: garantir que projetos distribuídos possam ser realizados com equipes de diferentes níveis de maturidade, melhorar a capacidade de cada grupo de trabalho e aprimorar o processo de software de uma organização. Para tanto, definiu-se um processo de software padrão específico para equipes geograficamente distribuídas, que é especializado em quatro outros processos, cada qual correspondendo a um nível de maturidade. A especialização do processo padrão utiliza um modelo de maturidade de referência, resultante da adaptação do CMM para atender às necessidades do desenvolvimento de software geograficamente distribuído. A correspondência entre os diferentes aspectos do processo padrão e as áreas-chave do modelo de maturidade permite identificar que categorias do processo necessitam de maior atenção. Desta forma, o processo de software de uma organização é aprimorado baseando-se na experiência da sua utilização. Para apoiar o acompanhamento do modelo de gerência e verificar a sua aplicabilidade foi desenvolvido o sistema HyperDev, utilizando a tecnologia WWW.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

SOFTWARE PROCESSES MANAGEMENT FOR GEOGRAPHICALLY DISPERSED WORKING GROUPS

Carmen Maidantchik

June, 1999

Advisors:

Prof. Ana Regina Cavalcanti da Rocha

Prof. Geraldo Bonorino Xexéo

Department: Systems and Computing Engineering

This thesis introduces a software process management model for geographically dispersed working groups. The proposed model goals are: guarantee that distributed software projects can be developed with unlike maturity level teams, improve each working group capability and enhance the organization software process. In order to achieve these objectives, a standard software process suitable for dispersed working groups was defined and, then specialized in four different processes each corresponding to a specific maturity level. The standard process specialization uses a capability maturity model, resulting from a CMM adaptation for geographically dispersed software development. The correspondence between the standard process aspects and the key process areas of the capability maturity model supports the identification of the process categories that need more attention. In this way, an organization software process is improved by the experience of its own use. In order to implement the monitoring level of the model and to verify its applicability the HyperDev system was developed using the WWW technology.

Conteúdo

Lista de Figuras

Lista de Tabelas

I. Introdução

Este capítulo descreve os fatores que motivaram a realização desta tese, define o objetivo da pesquisa e apresenta a sua organização.

I.1. Motivação

As colaborações geograficamente distribuídas estão surgindo em grande velocidade devido às mudanças sócio-econômicas e à grande utilização da Internet. Muitos projetos de software são formados por equipes que trabalham em diferentes localidades. Embora a rede de computadores permita uma comunicação rápida, confiável e eficiente, os profissionais utilizam diversos equipamentos e métodos e, portanto, as equipes são, muitas vezes, heterogêneas, cultural e tecnologicamente diferentes. Tais características implicam que a utilização de um único processo de software por todas as equipes envolvidas em um projeto é inadequada.

Esta tese foi desenvolvida no contexto do Projeto de Colaboração Internacional *"Física Experimental de Altas Energias e Tecnologias Associadas"* entre a UFRJ e o CERN[1] (*European Laboratory for Particle Physics*) (CERN, 1999). O CERN é um laboratório europeu, localizado na fronteira entre a Suíça e a França, que realiza experimentos envolvendo colisões de partículas, visando estudar os ínfimos constituintes da matéria e obter um melhor entendimento sobre a criação do Universo.

Neste laboratório, enormes aceleradores de partículas são construídos e utilizados para apoiar pesquisas cooperativas, permitindo que os físicos trabalhem de forma mais eficiente do que se cada país mantivesse uma pesquisa independente. O CERN é um dos maiores laboratórios científicos do mundo e, provavelmente, o melhor exemplo de colaboração internacional com 19 países membros e alguns países observadores. Tal característica contribuiu para a concepção, no próprio CERN, do World-Wide Web cujo objetivo inicial era permitir que os físicos de altas energias

[1]Organização Européia para Pesquisa Nuclear (a abreviação provém da designação original francesa: *Conseil Européen pour la Recherche Nucléaire).*

pudessem acessar em qualquer lugar os dados de maneira uniforme. O WWW é um sistema cliente-servidor para a Internet, projetado para colaborações numerosas, distribuídas e assíncronas (WWW, 1999). Por ser um sistema de informação multimídia, disperso, heterogêneo e cooperativo, atualmente, o seu escopo é muito maior do que a sua concepção inicial.

Em 1994, o CERN aprovou um novo projeto, o LHC (*Large Hadron Collider*), um acelerador de partículas que permitirá a colisão de prótons em altas energias (14 TeV), jamais alcançadas anteriormente. As pesquisas no LHC possibilitarão que cientistas penetrem na estrutura da matéria e recriem as condições iniciais do Universo apenas 10^{-12} segundos após o *Big-Bang*. O detetor ATLAS (*A Toroidal LHC ApparatuS*) está sendo projetado para respeitar as características do LHC (ATLAS COLLABORATION, 1994, ATLAS, 1999). A construção do detetor levará cinco anos e seu funcionamento começará somente no ano de 2005. O software deve ser projetado de maneira que possa ser adaptado de acordo com a evolução do experimento. O grupo de software do ATLAS necessita produzir programas de alta qualidade e, devido à longevidade do experimento, a introdução de novos mecanismos às aplicações existentes deve ser uma tarefa fácil. Além disto, deve-se considerar que mais de 1700 colaboradores de 144 universidades e centros de pesquisas utilizarão o software em diferentes plataformas espalhadas pelo mundo.

A complexidade dos sistemas computacionais para Física de Altas Energias e as características específicas do LHC indicam que os métodos tradicionais para desenvolvimento de software podem falhar, impossibilitando alcançar o nível de confiança desejado, dentro de estimativas de custo e prazo. Um grande problema é o fato de que os desenvolvedores estão espalhados pelo mundo. No caso do ATLAS, aproximadamente 1000 profissionais/ano serão necessários, sendo que 85% correspondem a pequenos grupos não localizados no CERN. Para fazer a melhor utilização do investimento humano e para garantir a alta qualidade do software, os sistemas têm que ser cuidadosamente projetados e o próprio projeto deve ser bem gerenciado. O progresso e a qualidade do código devem ser monitorados continuamente. É essencial definir processos de software que respeitem as características específicas deste

ambiente (CANDLIN, 1996).

I.2. Histórico da Pesquisa Realizada

O tema da presente tese foi identificado no CERN, no âmbito do Experimento ATLAS. No início do ano de 1993, o detetor ATLAS era apenas uma proposta e os pesquisadores colaboradores participavam das atividades de pesquisa e desenvolvimento (*R&D - Research and Development*). A proposta do futuro acelerador LHC e de seus experimentos seria avaliada no final de 1994. No decorrer destes quase dois anos, foi imprescindível produzir análises físicas, com alto grau de confiabilidade, a partir de dados gerados pelos projetos R&D e que demonstrassem a grande probabilidade de descobrir as novas partículas constituintes da matéria. Grande esforço foi dedicado para produzir uma descrição detalhada do detetor ATLAS a partir dos resultados dos testes do feixe de partículas, atividades realizadas com o apoio de diversos sistemas computacionais.

Durante 1993, o Grupo de Software do ATLAS estava especificando o software responsável pela reconstrução da trajetória das partículas após sua colisão, trabalho realizado por pesquisadores de diferentes universidades. Projetamos, neste momento, uma ferramenta para registrar os requisitos de tal sistema, implementando-a no WWW e utilizando o navegador Mosaic como interface. Esta ferramenta permitiu a comunicação entre os pesquisadores e facilitou o entendimento dos requisitos, organizados através de arquivos hipertextuais. O processo de construção de um software científico corresponde a um processo cíclico, onde os requisitos são sucessivamente refinados até que o resultado seja satisfatório. A representação das informações relativas ao processo de software através de hipertextos monstrou-se adequada, pois a alteração ou inclusão de alguns requisitos pode ser facilmente realizada, modificando ou criando nós. A ferramenta foi apresentada na conferência *Computing in High Energy Physics'94* (MAIDANTCHIK e ISAAC, 1994).

Com base nesta experiência, participamos da padronização das atividades de construção de software, realizada por equipes dispersas. Neste contexto, os diferentes

módulos das aplicações computacionais são implementados por diversos pesquisadores e, posteriormente, integrados e testados. Em ambientes com tal característica, é importante que o processo de construção do código respeite regras pré-definidas e que suas atividades sejam padronizadas. Assim, o projeto pode ser melhor controlado e as falhas da aplicação, mais facilmente detectadas. A padronização iniciou-se com a implementação de *macros*, que correspondem a conjuntos de comandos, reunidos em um único arquivo, para realizar desde modificações no código até a geração do programa executável, de maneira transparente ao desenvolvedor. Os procedimentos foram documentados em *The ATLAS Offline Software Manual* (ATLAS SOFTWARE COLLABORATION, 1996b).

No mesmo ano (1993), fomos responsável (*WebMaster*) pelo registro de informações do ATLAS no WWW (o ATLAS foi o primeiro grupo a escolher oficialmente o Web como principal repositório de informações). A atualização das informações era um trabalho repetitivo e implicava na modificação periódica de páginas estáticas. Em colaboração com o grupo *WWW Support* do CERN, participamos do desenvolvimento do programa *cgiparse*. A linguagem HTML permite a entrada de dados através de formulários hipertextuais; o WWW é responsável pelo seu envio e recebimento através da rede; o programa *cgiparse* é executado no servidor que decodifica os dados, exportando-os como variáveis do ambiente, os quais podem ser utilizadas pelos programas instalados no servidor, o que permite que os programas sejam executados na hora da consulta, apresentando informações dinâmicas através do WWW.

Ao longo da construção do software *online* do Calorímetro de Telhas do ATLAS (1994), a equipe de desenvolvimento enfrentou o problema de gerência das diversas versões. Entre e durante os testes do feixe de partículas, a configuração do sistema de aquisição de dados sofre freqüentes mudanças e, logicamente, o software *online* também teve que ser modificado. O WWW foi utilizado para documentar o processo de construção e as alterações ocorridas em cada versão do sistema. O navegador Netscape serviu como interface apresentando, de forma hierárquica, as informações do software. Esta experiência foi relatada no workshop *Software En-*

gineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics (MAIDANTCHIK *et al.*, 1995).

Após a aprovação da proposta do acelerador LHC (dezembro de 1994), iniciou-se a construção do sistema de aquisição de dados central do ATLAS. Participamos do desenvolvimento do sistema *Online Volume Bookkeeping* (1995), utilizando uma ferramenta CASE orientada a objetos. Tal sistema registra as informações de cada *run*[2] (data, hora, número de eventos adquiridos, identificação da fita magnética que registra os eventos, etc.) e dos equipamentos eletrônicos. A especificação do sistema foi definida de acordo com as necessidades dos diversos grupos do ATLAS – cada qual, responsável por uma parte do detetor – e pelas restrições do sistema de aquisição – velocidade e capacidade de armazenamento. Portanto, o projeto do software ensejou um conjunto de decisões – tomadas ao longo das reuniões entre os pesquisadores responsáveis pelo detetor – que influenciaram a funcionalidade do software e a apresentação das informações registradas. Para apoiar a documentação das decisões tomadas, utilizou-se o WWW. O Web também foi selecionado para exibir os dados manipulados pelo sistema *Bookkeeping*. O trabalho foi apresentado na conferência *Computing in High Energy Physics'95* (BEE *et al.*, 1995a, BEE *et al.*, 1995b).

A definição do processo de software do ATLAS iniciou-se em 1996. Como membro do Grupo *Offline Software* do ATLAS, participamos do planejamento do processo de software e a primeira ação foi identificar a situação corrente. As informações foram adquiridas através de reuniões e de análises dos procedimentos realizados durante a construção de aplicações. Esta primeira etapa serviu de base para identificar problemas e formalizar as atividades de desenvolvimento de software. Uma padronização inicial foi definida e aplicada em todos os projetos de software. Os padrões, baseados no ESA PSS-05 (ESA, 1991), definem roteiros ou recomendações, procedimentos, regras e a documentação associada ao software. O trabalho foi descrito nos documentos *ATLAS Computing Technical Proposal* (ATLAS SOFTWARE

[2]Um *run* corresponde a uma configuração do detetor, que é modificada periodicamente durante a aquisição de dados.

COLLABORATION, 1996a) e *Software Process Improvement in the ATLAS Experiment* (MAIDANTCHIK e DA ROCHA, 1996).

O padrão do ATLAS estabelece as atividades e tarefas a serem aplicadas durante o desenvolvimento, operação e manutenção do software, entretanto, não orienta a adaptação para sua utilização pelos diversos grupos de trabalho e nem indica diretrizes para melhorar a capacidade das equipes. A avaliação do processo é informal e não há coleta de informações para validar o padrão atual e apoiar futuras melhorias do próprio processo. Tais dificuldades motivaram o desenvolvimento de uma proposta mais adequada.

O conhecimento adquirido, através das pesquisas descritas anteriormente, serviu de base para a proposta de um modelo de gerência de processos de software para equipes de trabalho geograficamente dispersas, a qual foi apresentada pela primeira vez em 1996, na workshop "Qualidade de Software" (MAIDANTCHIK *et al.*, 1996). A evolução da pesquisa foi relatada periodicamente em conferências e workshops (MAIDANTCHIK *et al.*, 1997b, MAIDANTCHIK *et al.*, 1997a, MAIDANTCHIK *et al.*, 1998d, MAIDANTCHIK *et al.*, 1998a, MAIDANTCHIK, 1998a, MAIDANTCHIK *et al.*, 1998b, MAIDANTCHIK *et al.*, 1999).

I.3. Objetivos da Tese

Em função do contexto anteriormente descrito, o objetivo da pesquisa foi especificar um modelo de gerência para apoiar o desenvolvimento de software realizado por equipes de trabalho heterogêneas, geograficamente dispersas, cultural e tecnologicamente diferentes. Para definir tal modelo, realizou-se um estudo sobre modelagem e gerência de processos de software e ambientes centrados em processos. Avaliaram-se normas para padronizar e certificar processos de software e modelos para avaliar e melhorar a sua qualidade. A tecnologia WWW foi investigada para verificar a sua utilização em processos de software e pesquisou-se o estado da arte em sistemas hipertextuais de apoio à construção de software em redes.

As principais metas do modelo de gerência são: garantir que projetos distri-

buídos tenham êxito e possam ser realizadas mesmo com grupos de diferentes níveis de maturidade, melhorar a capacidade de cada equipe e aprimorar a capacidade da organização como um todo. Tais metas são alcançadas através da definição de especializações do processo de software em diferentes níveis de maturidade; da definição de instâncias do processo especializado; do acompanhamento de processos que registra as atividades de cada projeto e coleta dados; de avaliações, que identificam a maturidade de cada grupo de trabalho; da melhoria, que indica problemas comuns em todas as equipes e apontam os aspectos do processo que precisam de maior atenção, permitindo o aprimoramento do processo de software da organização.

Um protótipo foi implementado, utilizando o WWW, para apoiar o acompanhamento do modelo de gerência de processos de software. Através desta ferramenta, o projetista de software define os objetivos do sistema a ser construído. Os requisitos são capturados e registrados. A análise e projeto do sistema são produzidos por qualquer ferramenta CASE e os modelos resultantes e respectiva documentação são integrados à definição dos requisitos através de ligações hipertextuais. A ferramenta permite que o desenvolvimento de projetos seja monitorado de forma padronizada. O registro de informações, ao longo da construção de um software, é utilizado posteriormente, para avaliar a equipe, identificar falhas no processo e aprimorá-lo.

I.4. Organização da Tese

O capítulo II apresenta as pesquisas realizadas sobre modelagem, qualidade e gerência de processos e ambientes centrados em processos. As normas para a padronização e certificação de processos são apresentadas e modelos de maturidade de processos de software são discutidos.

O capítulo III, descreve-se o sistema World-Wide Web e seus mecanismos. O estado da arte em sistemas hipertextuais para apoiar a construção de software em rede é apresentado.

O capítulo IV apresenta as características e os problemas encontrados no desenvolvimento distribuído de software. Os requisitos a serem respeitados na definição

de um processo de software para equipes geograficamente dispersas são definidos. Ao final, apresenta-se o processo de software padrão que procura atender o escopo desta pesquisa.

O capítulo V apresenta o modelo de gerência de processos de software para equipes de trabalho geograficamente dispersas, seu objetivo, metas e critérios de definição, acompanhamento, avaliação e melhoria. Descreve-se as especializações do processo padrão, apresentado no capítulo anterior, para atender às diferentes características dos grupos de trabalho.

O capítulo VI apresenta o sistema HyperDev, que apóia o acompanhamento do modelo de gerência de processos de software para equipes geograficamente dispersas. São descritos seus objetivos, construção, funcionalidade e a tecnologia utilizada. Ao final, faz-se uma análise comparativa do sistema com os sistemas hipertextuais de apoio ao desenvolvimento de software em rede, descritos no capítulo III.

O último capítulo corresponde às conclusões da tese. São apresentadas as contribuições do trabalho e propostas para pesquisas futuras.

II. Processo de Software

Neste capítulo, apresentam-se a evolução histórica da pesquisa em processo de software, algumas definições e exemplos de linguagens para modelagem de processos. Descrevem-se aspectos relacionados à gerência do processo de software e os ambientes centrados em processo. Justifica-se a importância de aprimorar continuamente um processo de software, descrevendo normas para padronizar e certificar processos de software e modelos para avaliar e melhorar a qualidade de tais processos. Ao final, apresentam-se direções estratégicas em processos de software e descrevem-se os trabalhos relacionados ao desenvolvimento de software realizado por equipes distribuídas.

II.1. Evolução Histórica da Pesquisa sobre Processos de Software

A crise do software foi identificada no final dos anos 60. Para aumentar a produtividade dos processos de desenvolvimento de software e aprimorar a qualidade dos produtos, muito esforço foi investido, desde então, em métodos de desenvolvimento, modelos de ciclo de vida, técnicas, ferramentas CASE e linguagens de programação (CHENG, 1994). Muitas organizações investem na aquisição de novas tecnologias e no treinamento de profissionais buscando utilizar técnicas e métodos modernos. Porém, melhorias – sejam na produtividade, sejam na qualidade – não ocorrem apenas pela introdução de tecnologias (PERRY *et al.*, 1994, ARTHUR, 1997, ESPITI, 1999). O processo é um fator importante que afeta a produtividade das equipes de trabalho e a qualidade do software (CURTIS *et al.*, 1992, ELLMER, 1995, HUMPHREY, 1995, HALEY, 1996, CURTIS, 1997, PAULK *et al.*, 1997, EMAM *et al.*, 1998). De acordo com alguns autores (FOWLER e RIFKIN, 1990, BOLCER, 1996), é necessário definir processos de desenvolvimento de software consistentes e adequados e, só então, selecionar ferramentas e métodos para apoiar a execução das atividades contidas no processo.

O processo de software vem recebendo uma atenção cada vez maior, princi-

palmente no que diz respeito à sua definição, automatização, medição e melhoria (MADHAVJI e PENEDO, 1993, BANDINELLI *et al.*, 1995a, PRESSMAN, 1997, KAISER e KAPLAN, 1995). A comunidade de pesquisa interessada no processo de software se preocupa com a tecnologia para apoiar a definição e a execução de processos; a comunidade de engenharia pesquisa a integração e adaptação de processos utilizados por diferentes equipes de trabalho; a indústria e a comunidade de *workflow* se concentraram na reengenharia de processos para torná-los mais eficientes; os envolvidos em temas relacionados à qualidade de software direcionam a atenção na avaliação e medição de processos para identificar possíveis melhorias (MADHAVJI e PENEDO, 1993, GRUHN e URBAINCZYK, 1998, GRUNDY *et al.*, 1998). Todas estas perspectivas reconhecem que o processo de software se refere à estruturação de informações que devem ser gerenciadas (GRUHN e SCHNEIDER, 1998).

Uma grande discussão sobre processos de software ocorreu em 1987, quando Osterweil defendeu, no artigo *"Software Processes are Software Too"* (OSTERWEIL, 1987), a idéia de que da mesma forma que as aplicações, os processos são executados e, portanto, devem atender a seus requisitos, podem ser modelados e evoluem (LAI, 1993, OSTERWEIL, 1997). A habilidade de representar e executar processos de software é essencial para o seu entendimento, utilização e aprimoramento (SUTTON *et al.*, 1995). Embora tal idéia tenha sido apresentada muito antes por M. M. Lehman (1969) (OSTERWEIL, 1997), o primeiro evento reunindo exclusivamente pesquisas relacionadas a processos de software só ocorreu em 1984, o *1st. Software Process Workshop* (LEHMAN, 1997). Neste período, os engenheiros de software se preocupavam com a estruturação e modelagem de processos de software (RADICE *et al.*, 1985, HUMPHREY e KELLNER, 1989, KELLNER e HANSEN, 1989). Posteriormente, M. I. Kellner (KELLNER, 1989), D. Harel (HAREL *et al.*, 1990), W. Scacchi (SCACCHI e MI, 1993), G. Cugola (CUGOLA *et al.*, 1995) e outros (BARGHOUTI e KRISHNAMURTHY, 1993, LEONHARDT *et al.*, 1995, AVRILIONIS *et al.*, 1996a, AVRILIONIS *et al.*, 1996b) apresentaram trabalhos sobre automação e execução de processos de software.

A comunidade de pesquisa interessada em qualidade de software começou a in-

vestigar a influência da qualidade dos processos de software na qualidade dos produtos. O Instituto de Engenharia de Software (*Software Engineering Institute - SEI*) (SEI, 1999) publicou diversos trabalhos (FOWLER e RIFKIN, 1990, HUMPHREY, 1993, KELLNER e HANSEN, 1988) e desenvolveu um modelo de maturidade de processo de software, denominado Modelo de Maturidade da Capacidade (*Capability Maturity Model - CMM*) (PAULK *et al.*, 1993a, PAULK *et al.*, 1993b). Alguns pesquisadores compararam o modelo CMM com as normas de certificação ISO (PAULK, 1994, STELZER *et al.*, 1996, GRAHL *et al.*, 1997, CARVALHO *et al.*, 1997) e outros modelos de avaliação e melhoria de processos de software surgiram – BootStrap (ENGELMANN *et al.*, 1997, HAASE *et al.*, 1994) e Trillium (IRVING, 1999, COALLIER, 1996) – culminando com a busca de uma padronização internacional através do projeto SPICE (EMAM e GOLDENSON, 1996a, MARSHALL *et al.*, 1996, WOODMAN e HUNTER, 1996, SPICE, 1999).

W. S. Humphrey (HUMPHREY, 1990), D. J. Reifer (REIFER, 1993), V. R. Basili (BASILI, 1993) defenderam a gerência do processo como forma de garantir a qualidade dos produtos. Diversos trabalhos (JACCHERI e CONRADI, 1993, LAI, 1993, PERRY *et al.*, 1994, BANDINELLI *et al.*, 1995a) discutiram a evolução, amadurecimento e melhoria dos processos de software. Medições através de linguagens descritivas de processos indicam o seu tamanho e complexidade, permitem entender e melhorar processos existentes e prevêem o impacto com a introdução de novas tecnologias (BRÖCKERS *et al.*, 1996, HUMPHREY, 1996, PFLEEGER *et al.*, 1997). Já podem-se encontrar publicações que apresentam os resultados e benefícios alcançados através da aplicação de modelos de maturidade em organizações de software (HERBSLEB *et al.*, 1994, CATTANEO *et al.*, 1995, HALEY, 1996, HERBSLEB e GOLDENSON, 1996, DIAZ e SLIGO, 1997, PESSÔA *et al.*, 1997) ou da utilização das normas ISO juntamente com o modelo CMM (MACHADO *et al.*, 1997, ARRUDA *et al.*, 1998).

Atualmente, as pesquisas sobre processos de software se referem a: linguagens para modelar processos, ambientes centrados em processo para modelar e executar processos, melhoria de processos, modelos de processo, estudos empíricos e resultantes

de tributação, medição e evolução de processos, relações entre processos de software e outros tipos de processo (BASILI, 1996, GRADY, 1997, BACH, 1998, CURTIS, 1998).

II.2. Definição de Processo de Software

Radice (RADICE *et al.*, 1985) apresentou uma das primeiras definições de processo de software, composto por atividades pré-definidas que contém: uma lista de critérios de entrada, que devem ser satisfeitos antes de iniciar uma atividade; um conjunto de descrições das atividades, que indicam o que deve ser realizado; um procedimento de validação da qualidade dos trabalhos resultantes das atividades; uma lista de critérios de saída, que devem ser satisfeitos antes de finalizar as atividades.

Segundo M. L. Jaccheri e R. Conradi (JACCHERI e CONRADI, 1993), um processo de software corresponde ao conjunto de atividades de engenharia de software necessárias para transformar os requisitos do usuário em um sistema computacional e para evoluir tal conjunto. É composto por dois componentes: um processo de produção de software e um meta-processo para aprimorar e evoluir o processo.

Processos de software são compostos por atividades do ciclo de vida tais como análise de requisitos, projeto, código, teste, instalação, manutenção, etc. A maioria das atividades são compostas por outras atividades e poucas são atômicas. As atividades podem se comunicar entre si e operam sobre produtos de entrada para produzir produtos de saída (FOWLER e RIFKIN, 1990, JACCHERI e CONRADI, 1993). Os produtos de software consistem em todos os artefactos produzidos durante o ciclo de vida do software, tais como documentos resultantes da especificação, implementação e relatórios. Cada artefacto pode existir em várias versões (BANDINELLI *et al.*, 1995b).

Kitchenham e Pfleefer (1996) identificam os objetivos de um processo de desenvolvimento de software de qualidade: produzir sistemas computacionais de acordo com a sua especificação e melhorar a capacidade de uma organização em produzir tais sistemas. O ponto-chave corresponde às medidas que permitem verificar co-

mo as atividades levam à qualidade do software e como o processo de construção a influencia.

Outro aspecto a ser considerado ao definir um processo de software é a sua adequação ao domínio da aplicação e ao projeto, considerando suas características e restrições. A cada projeto, o processo de software deve ser ajustado às especificidades da aplicação, tecnologia a ser utilizada na sua construção, grupo de desenvolvimento e usuários finais (FALBO, 1996, WASSERMAN, 1996).

Um processo de software corresponde, de forma simplificada, a uma seqüência de passos a serem realizados para uma determinada proposta, definindo as ações para transformar entradas no produto final (PAULK *et al.*, 1997).

O reconhecimento de que a construção de um software corresponde a uma atividade de engenharia confirmou que um processo de software pode ser entendido como outros processos de engenharia (SOMMERVILLE, 1992). A evolução de processos de software requer a identificação de fragmentos do modelo do processo que precisam ser modificados. Deve-se, também, verificar quando e como a modificação deve ocorrer, analisar e orientar tais mudanças (JACCHERI e CONRADI, 1993, AVRILIONIS *et al.*, 1996c).

Os processos de software evoluem em quatro direções (RACCOON, 1997): (1) *inovação* – novos métodos e técnicas são introduzidos no processo de software, (2) *crescimento* – as inovações são utilizadas pela equipe de trabalho e suportes técnicos são implementados; nesta fase, torna-se possível padronizar o processo de software, (3) *maturidade* – o processo de software é entendido e dominado pelos profissionais envolvidos no projeto e (4) *melhorias* – as restrições do processo são identificadas e novos procedimentos para a construção do software são definidos para posterior introdução no processo atual.

II.3. Modelagem de Processo de Software

Alguns autores (KELLNER e HANSEN, 1988, CURTIS *et al.*, 1992, KRASNER *et al.*, 1992, BANDINELLI *et al.*, 1995a) identificam os objetivos para modelar pro-

cessos de software: facilitar a comunicação e entendimento efetivos sobre o processo; permitir a reutilização do processo; apoiar a evolução do processo; servir de base para automação do processo; facilitar a gerência do processo e permitir o seu aprimoramento. Para atingir tais objetivos, a modelagem deve incorporar capacidades de *representação*, que permitam descrever atividades correntes, atividades futuras e restrições do processo, e de *análise compreensiva*, que possibilite validar o modelo (FOWLER e RIFKIN, 1990, GIBSON, 1997).

Modelos de processo de software descrevem o conhecimento utilizado para gerenciar a realização de processos e verificar sua consistência e correção (CANALS *et al.*, 1995, SOMMERVILLE, 1996). Os modelos definem as estruturas que formam os processos e que são necessárias para apoiar uma estratégia de desenvolvimento de software (RACCOON, 1997). A modelagem de processos de software corresponde às técnicas utilizadas para definir e analisar os aspectos significantes dos processos (KRASNER *et al.*, 1992).

Segundo Visaggio (1994), um modelo de processo define que tarefas produzem quais objetos e quem é responsável pela sua realização. Quando propriamente projetado, um modelo orienta o trabalho dos engenheiros de software, definindo os objetivos gerais do processo e como decompô-los em metas, o que propicia uma organização hierárquica do processo de software (VISAGGIO, 1994, HUMPHREY, 1995, CANALS *et al.*, 1995). Bandinelli (BANDINELLI *et al.*, 1995a) considera que processos de software são entidades complexas e, portanto, precisam ser descritas e seguidas de maneira precisa e não ambígua através de modelos.

Um modelo de processo de software pode ser caracterizado como uma representação suficientemente geral para modelar vários processos de software e, suficientemente específica, para permitir a sua compreensão, evolução e adaptação (DA SILVA, 1993). Um modelo de processo de software típico abrange todo o seu ciclo de vida e o estrutura em fases, definindo as atividades a serem realizadas (CURTIS *et al.*, 1992). As atividades operam sobre produtos de software e produzem outros produtos. Os produtos de software correspondem a tudo que é produzido ao longo do seu ciclo de vida, tais como: requisitos, especificações, código, plano de gerência,

relatório de erros e documentação (JACCHERI e CONRADI, 1993). Portanto, um modelo pode ser descrito em termos de produtos intermediários, fluxo de trabalho e relações entre seus componentes, garantindo sua consistência e, ao mesmo tempo, oferecendo uma flexibilidade para permitir sua adaptação às características de um determinado projeto (JACCHERI e CONRADI, 1993, VISAGGIO, 1994, GRUNDY *et al.*, 1998).

Diagramas de fluxo de dados e de controle, máquinas de estado finito, orientação a objetos e redes de petri têm sido utilizados para modelar processos de software (CURTIS *et al.*, 1992, SHEPARD *et al.*, 1992, SUTTON *et al.*, 1995, OSTERWEIL, 1997). A modelagem de processos serve para apoiar o planejamento, monitoração e registro de processos, representam, analisam e compararam metodologias de desenvolvimento, simulam a execução de processos e oferecem suporte à sua automação (SUTTON *et al.*, 1995). Tem como benefícios (KELLNER e HANSEN, 1988, AOYAMA, 1998a): melhor entendimento das ações do processo por aqueles que devem executá-las e gerenciá-las e determinação dos objetivos gerais do processo.

II.3.1. Linguagens para Modelagem

A programação de processos de software é definida como a codificação de programas que representam e/ou apóiam a execução de tais processos. A programação de processos depende da existência de linguagens de programação de processos de software. De acordo com S. M. Sutton, S. Heimbigner e L. J. Osterweil (SUTTON *et al.*, 1995), tais linguagens devem respeitar as características dos processos e produtos de software. Processos de software devem ter uma estrutura flexível e dinâmica, respondendo às contingências que ocorrem durante o desenvolvimento de um produto e considerando a disponibilidade de recursos tecnológicos e humanos. Tipicamente, processos de software incluem um alto grau de concorrência, tais como a implementação de módulos independentes. Muitas atividades do processo de software são iniciadas com a chegada de um evento e ocasionam a execução de outras atividades e a propagação de mudanças nos produtos intermediários. Os produtos

de software se referem a diversos tipos de artefactos, tais como requisitos, código, plano de testes, documentação, etc. Os produtos também se relacionam entre si. As linguagens de programação de processos de software devem garantir a consistência dos processos desde que mudanças em uma de suas partes pode afetar outras partes.

Entre as linguagens que representam processos de software, as mais adequadas à sua especificação e projeto são: *Statecharts Language* do sistema STATEMATE (HAREL *et al.*, 1990, HAREL e NAAMAD, 1996), MVP-L (*Multiview Process Modeling Language*) (SUTTON *et al.*, 1995, BRÖCKERS *et al.*, 1996), LOTOS (*Language of Temporal Ordering Specification*) (SAEKI *et al.*, 1991) e Articulator (MI e SCACCHI, 1991, SCACCHI e MI, 1993). Algumas delas (Statecharts, LOTOS e Articulator) apóiam a simulação de processos e permitem a geração de representações executáveis. Entretanto, não apóiam a descrição detalhada de produtos. A ênfase da linguagem APPL/A (*Ada Process Programming Language based on Aspen*) (CURTIS *et al.*, 1992, SUTTON *et al.*, 1995) é na execução de processos, permitindo visualizar a implementação de modelos. Para tanto, APPL/A define um conjunto de mecanismos que representam processos concorrentes e reagem a eventos como, por exemplo, propagação de atualizações e envio de notificações em resposta a mudanças nos dados do processo.

Algumas linguagens permitem definir o fluxo de tarefas, associação de ferramentas e artefactos ao processo de software: Process Weaver (FERNSTRÖM, 1993, BOURDON, 1993, IPT GROUP, 1997) e ProSLCSE (*Process-oriented software life cycle support environment*) (SUTTON *et al.*, 1995). Ambas são gráficas e se baseiam em conceitos para modelagem de processos. VLP (*Visual Process Language*) (SHEPARD *et al.*, 1992) é uma linguagem formal projetada para representar graficamente um processo de software e permitir a execução automática de atividades. Outras linguagens gráficas, baseadas em redes de petri, tais como Melmac (GRUHN, 1991), SLANG (*SPADE Language*) (BANDINELLI *et al.*, 1993b, BANDINELLI *et al.*, 1995a, BANDINELLI *et al.*, 1996) e ainda Process Weaver, permitem a coordenação de ferramentas e dados do processo, e integram alguma forma de representação de produtos. Através da linguagem SLANG, as definições do processo de software são

organizadas hierarquicamente, de acordo com o paradigma de orientação a objetos. Cada atividade encapsula um conjunto de passos do processo e pode incluir chamadas a outras atividades.

Os sistemas baseados em regras representam o processo de software através da execução de linguagens naturais. Alguns exemplos são: Marvel Strategy Language (KAISER e BEN-SHAUL, 1993) e as regras tipo PROLOG, utilizadas pelo sistema Merlin (PEUSCHEL e SCHAFER, 1992). As regras permitem identificar inconsistências no processo de software (SUTTON *et al.*, 1995).

A linguagem SEL (*Oz's Shell Envelope Language*) corresponde a roteiros (*scripts*) Unix e construções, as quais são expandidas em comandos para manipular a interface entre ferramentas e o ambiente (VALETTO e KAISER, 1996).

Outras linguagens incorporam mais de um paradigma: SPELL (*Software Process Evolutionary Language*), do projeto EPOS (JACCHERI e CONRADI, 1993, CONRADI *et al.*, 1993), é uma linguagem orientada a objetos que apóia concorrência e, tecnicamente, corresponde a uma extensão de PROLOG; CSPL (*Concurrent Software Process Language*) (CHEN, 1997) integra uma sintaxe orientada a objetos, tipo Ada95, com a semântica do Unix *shell* para permitir a modelagem e a execução de processos. Finalmente, alguns sistemas oferecem mais de uma linguagem, como o SMART que combina uma linguagem orientada a objetos (baseada na linguagem Articulator (MI e SCACCHI, 1991)) para especificação de processo com outra linguagem (baseada na ferramenta SynerVision (PENEDO, 1995)) para execução de processos (GARG *et al.*, 1994, SUTTON *et al.*, 1995).

II.4. Ambientes Centrados em Processos

Ambientes de engenharia de software baseados em processos são utilizados para orientar desenvolvedores, automatizar as atividades de software e oferecer informações sobre o desempenho e o estado de processos de software (BANDINELLI *et al.*, 1995a). Um ambiente centrado em processo (*Process Centered Environments - PCE*) é composto por uma linguagem para modelagem de processos e várias fer-

ramentas de processo para apoiar a definição, instalação, evolução e execução de modelos de processo (JACCHERI e CONRADI, 1993, FALBO e ROCHA, 1996).

Embora os ambientes propostos tenham tido diferentes motivações e utilizem diversas técnicas, todos seguem o mesmo conceito: processos de software são entidades complexas que precisam ser descritas e acessadas de maneira precisa e não ambigüa (OSTERWEIL, 1987). Tal afirmação é necessária para apoiar a compreensão sobre processos e poder avaliar como melhorias podem ser alcançadas (BANDINELLI *et al.*, 1995a). Muitos desses sistemas propostos já foram utilizados em ambientes de produção, cujas experiências foram relatadas através de publicações (KELLNER, 1989, KRASNER *et al.*, 1992, SCACCHI e MI, 1993).

II.4.1. Pesquisas Acadêmicas

STATEMATE (HUMPHREY e KELLNER, 1989, HAREL *et al.*, 1990, HAREL e NAAMAD, 1996) é um sistema para modelar processos de software e objetiva facilitar o entendimento e a comunicação humana, apoiar a melhoria do processo e suportar a gerência de processos. O ambiente centrado em processo SPADE (*Software Process Analysis, Design and Enactment*) (BANDINELLI *et al.*, 1993a, BANDINELLI *et al.*, 1993b, BANDINELLI *et al.*, 1995a, BANDINELLI *et al.*, 1995c, BANDINELLI *et al.*, 1996) oferece uma linguagem específica ao domínio de aplicação (SLANG) para modelar processos de software através de uma rede de tarefas. SPMS (*Software Process Management System*) (KRASNER *et al.*, 1992, BANDINELLI *et al.*, 1995a) é um protótipo para suportar processos de software durante as tarefas de modelagem e evolução. SEPS (*Software-Engineering Process Simulation*) (LIN *et al.*, 1997) permite a simulação dinâmica do modelo de um processo de software e é composto por dois mecanismos: um de gerência (registro de decisões) e outro de engenharia (atividades de construção). Process Weaver (BOURDON, 1993, FERNSTRÖM, 1993, IPT GROUP, 1997) é uma ferramenta para modelar processos de software e apoiar a construção de produtos, permitindo o aprimoramento do nível de maturidade de processos de software através das seguintes funções: modelagem gráfica do processo, adaptação do modelos a diferentes projetos, rastreamento dos processos e

verificação se todas as atividades críticas do projeto foram realizadas. EPOS (*Expert system for Program and "Norwegian Og" System development*) (CONRADI *et al.*, 1993, JACCHERI e CONRADI, 1993) é um ambiente centrado em processo cujos objetivos são: facilitar a aquisição de informações sobre processos; permitir um aprimoramento através de uma modelagem específica; integrar aspectos organizacionais dos processos de software; apoiar a formalização e execução de processos; apoiar trabalhos cooperativos através de um banco de dados (EPOSDB) de transações; estruturar as versões e conectar mudanças; oferecer uma plataforma flexível para a integração de ferramentas. Merlin é um ambiente centrado em processo baseado em conhecimento, que permite modificar um conjunto de regras e fatos interpretados pelo executor do ambiente (PEUSCHEL e SCHAFER, 1992, BANDINELLI *et al.*, 1993b). Marvel é um ambiente de desenvolvimento de software, baseado em processo, multi-usuário, que oferece uma modelagem baseada em regras e apóia a execução de processos (CURTIS *et al.*, 1992). Oz (VALETTO e KAISER, 1996) foi desenvolvido na Universidade de Columbia como sucessor do Marvel. Melmac apóia a coordenação de ferramentas e dados do processo e oferece uma linguagem gráfica para codificar modelos de processo (SUTTON *et al.*, 1995).

Em ambientes centrados em processo, um processo de software pode ser modelado de várias formas através de linguagens procedurais, baseadas em regra, orientadas a objetos, etc (seção II.3.1.). Alguns ambientes oferecem tipos de objetos pré-definidos. Objetos correspondem a produtos intermediários, que são entradas ou resultantes de atividades (CHEN, 1997). SPADE, Marvel (KAISER e BENSHAUL, 1993) e EPOS permitem a definição de novos tipos de objetos.

SPADE apóia atividades cooperativas. Process Weaver oferece uma agenda para cada usuário e, ao executar uma atividade, a designação do trabalho é enviada à agenda do responsável (BOURDON, 1993). Em Merlin, todas as atividades a serem realizadas são apresentadas e o desenvolvedor pode escolhê-las (PEUSCHEL e SCHAFER, 1992). Este ambiente também apóia a representação de grupo de trabalho. O Process Weaver também permite a utilização com outras ferramentas que realizam tal função. (JACCHERI e CONRADI, 1993). Oz apóia cooperação síncrona

e assíncrona através de construções da linguagem que permitem descrever tarefas de gerência, as quais correspondem às regras no ambiente Marvel (BANDINELLI *et al.*, 1996).

No ambiente EPOS, a integração de novas ferramentas corresponde à instalação de uma ferramenta e a sua inserção no "envelope" de ferramentas (JACCHERI e CONRADI, 1993). Devido às suas arquiteturas abertas, SPADE e SPMS também permitem a introdução de novas ferramentas (BANDINELLI *et al.*, 1995c). O protótipo SPMS integra ferramentas de gerência com as ferramentas de apoio à construção do software (KRASNER *et al.*, 1992). Oz oferece diferentes níveis de integração de ferramentas, baseada no controle do comportamento das ferramentas, ou seja, o início de um processo requer que uma ferramenta execute alguma tarefa (BANDINELLI *et al.*, 1996, VALETTO e KAISER, 1996).

Muitos ambientes utilizam o seu próprio sistema para gerência de versões, tais como Marvel, EPOS. Outros, utilizam os sistemas existentes: IPSE 2.5 utiliza PS-Algol e O2 é utilizado em SPADE (JACCHERI e CONRADI, 1993). Oz utiliza um sistema de gerência de banco de dados orientado a objetos, com uma base de objetos para cada processo (VALETTO e KAISER, 1996).

No que se refere a planejamento, em EPOS e SPADE é possível construir, de forma incremental, a rede de tarefas de um processo de software (JACCHERI e CONRADI, 1993). Outros ambientes, como Marvel, indicam restrições que não podem ser violadas pela execução do processo (CURTIS *et al.*, 1992).

Quanto à simulação e validação de processos, em muitos ambientes, as relações podem ser utilizadas para controlar e propagar os impactos de uma mudança. Marvel, Merlin (PEUSCHEL e SCHAFER, 1992) e IPSE 2.5 oferecem algum apoio à verificação. Melmac (GRUHN, 1991) pode realizar simulações. EPOS não oferece funções para simular mudanças mas somente alguns mecanismos para verificação formal de mudanças (JACCHERI e CONRADI, 1993).

Alguns ambientes apóiam a evolução de processos, ou seja, quando o modelo de processo sofre modificações, o ambiente permite o seu refinamento antes da execução do processo, como acontece nos sistemas Process Weaver e Melmac (BAN-

DINELLI *et al.*, 1993b), ou após execução, como em Marvel, IPSE 2.5, SPADE e EPOS (JACCHERI e CONRADI, 1993). Os ambientes EPOS, IPSE 2.5, Marvel e SPADE oferecem um meta-processo para gerenciar a evolução do modelo (JACCHERI e CONRADI, 1993). STATEMATE não oferece nenhuma ferramenta de apoio à evolução, entretanto é possível identificar melhorias através da análise do processo (KELLNER e HANSEN, 1989). SPMP oferece um mecanismo para coleta de informações históricas a nível dos componentes do processo, o que facilita identificar inconsistências (KRASNER *et al.*, 1992).

II.4.2. Produtos Comerciais

Existem diversos tipos de ferramentas que orientam desde o planejamento de processo à definição e apoio a projetos de software (SURVEYER, 1997). Os protótipos e produtos comerciais de ambientes centrados em processo apóiam modelagem, análise, execução e evolução (BARGHOUTI e FEILER, 1993).

J. Surveyer (SURVEYER, 1997) categoriza as ferramentas comercializadas em quatro tipos: planejamento de processos, gerência de projetos, administração das equipes de trabalho e gerência total de processo e projeto. A grande maioria se referem à gerência de projetos para administrar sub-contratos, monitorar projetos, coordenar grupos de trabalho, identificar custos e riscos de projeto, controlar versões e visualizar projetos (GARVEY *et al.*, 1997, MOYNIHAN, 1997). Alguns exemplos são: Project Workbench PMW 3.2 da ABT Corp, CA-SuperProject for Windows da Computer Associates Int., Artemis Project View da Computer Science Corp., BB Project da Quality Decision Management Inc. Tais produtos foram desenvolvidos para ambiente Windows e Windows NT, oferecem uma configuração através de uma interface gráfica e alguns permitem a integração de outras ferramentas de projeto.

As ferramentas que apóiam o planejamento de processos são, por exemplo (SURVEYER, 1997): PE/Process Manager da LBMS, um programa para definir, planejar e medir processos de software, a serem utilizados para apoiar o planejamento de projetos; SHL Transform da MCI Systemhouse, um sistema de gerência de processos baseado em conhecimento que orienta práticas de projeto. Existem algumas ferra-

mentas para gerência total de processo e projeto: Firstcase 3.5 da AGS Management Systems Inc., um sistema de gerência de projeto que integra modelo de ciclo de vida, estimativa de projeto, administração e acompanhamento de projetos; Process Continuum, da Platinum Technology, oferece mecanismos para definir métodos, técnicas, papéis e tecnologia, além de realizar medições sobre a produtividade das equipes; SynerVision (DIAMANT, 1995), da Hewlet-Packard, uma ferramenta para execução de processos de software, onde o modelo é representado por uma hierarquia de tarefas associadas à equipe de trabalho.

Muitos dos ambientes centrados em processo citados na seção anterior ainda não são comerciais. Process Weaver se tornou um produto comercial, distribuído pela Cap Gemini Innovation, e está sendo utilizado na ESA (*European Space Agency*) e no CERN (BARGHOUTI e FEILER, 1993).

II.5. Gêrencia de Processos de Software

Gerência do processo de software é definida por Humphrey (HUMPHREY *et al.*, 1993) como a aplicação de conceitos, técnicas e práticas da engenharia de processo para explicitamente monitorar, controlar e melhorar processos de software.

Segundo Reifer (REIFER, 1993), gerência do processo de software refere-se ao desenvolvimento da infra-estrutura utilizada para administrar o processo, profissionais e produtos que populam projetos de software. No contexto de aprimorar a maturidade de organizações no desenvolvimento de software, a gerência do processo é definida como a aplicação de procedimentos para apoiar a construção de software de maneira uniforme, o que implica na definição, execução do processo, aquisição e análise de dados e controle do processo (KRASNER *et al.*, 1992, KRAUT e STREETER, 1995). Gerência do processo corresponde a um sistema para monitorar e melhorar continuamente cada método e procedimento do ciclo de vida do software (FOWLER e RIFKIN, 1990, ARTHUR, 1993, BLACKBURN *et al.*, 1996, HEIMANN *et al.*, 1996).

Gerência do processo inclui atividades de controle e manutenção dos itens que

são criados e utilizados durante o desenvolvimento de um software (WASSERMAN, 1996). De acordo com Arthur (ARTHUR, 1993), a gerência do processo reconhece que cada uma de suas atividades tenham *clientes internos* que determinam requisitos para o produto intermediário que recebem. A gerência do processo **define** o processo para realizar cada atividade e a maneira para **medir** e **garantir** que o processo está produzindo um produto que respeita as necessidades e expectativas dos *clientes externos*.

Um processo de software é gerenciado quando ambos, o processo e o produto, são entendidos e controlados através de medições detalhadas sobre a sua qualidade (HUMPHREY, 1995, MULLER, 1998). A principal tarefa da gerência do processo de software é garantir que os projetos alcancem seus objetivos (HUMPHREY, 1990), portanto, procedimentos de controle da qualidade devem ser aplicados ao longo do ciclo de vida de um software. Ao gerenciar adequadamente um processo de software, as organizações melhoram a capacidade de seus grupos de trabalho (HUMPHREY, 1995).

A gerência do processo de software se baseia na premissa que a qualidade de um produto de software é amplamente determinada pela qualidade do processo utilizado para desenvolvê-lo e mantê-lo (KELLNER e HANSEN, 1988, HUMPHREY, 1995, ELLMER, 1995, PAULK *et al.*, 1997). Cada engenheiro de software deve ser responsável pela qualidade de seu próprio produto. A gerência é responsável por mudanças no processo que irão prevenir defeitos e reduzir a variação da qualidade dos produtos de software (ARTHUR, 1993, RETTING e SIMONS, 1993, BOEHM, 1996). A busca da qualidade é tarefa de todos, mas responsabilidade do gerente (ARTHUR, 1993, WILLIAMS *et al.*, 1997).

Processos de software bem definidos e gerenciados permitem não só melhorar significativamente a qualidade dos produtos, como também aumentar a produtividade através de uma série de fatores (FOWLER e RIFKIN, 1990, HUMPHREY, 1995, FALBO, 1996): definição da seqüência das atividades do processo; suporte à gerência das atividades; suporte ao controle da qualidade dos produtos intermediários e coleta de informações para avaliações do processo de software.

II.6. Qualidade do Processo de Software

A capacidade de uma organização em produzir software é definida como a sua habilidade em desenvolver um novo produto – ou estender um produto existente – de maneira consistente, respeitando as expectativas dos usuários, com defeitos mínimos, em curto espaço de tempo e baixos custos (JOHNSON, 1996, PAULISH e CARLETON, 1997, IRVING, 1999). Aumentar a capacidade do processo de produção de uma organização permite: reduzir os custos de desenvolvimento e manutenção; diminuir o tempo de produção; respeitar as demandas dos usuários e do mercado; aumentar a satisfação dos usuários; melhorar a estimativa de esforços; analisar efetivamente o risco envolvido em um projeto; apurar a habilidade em respeitar cronogramas; e alcançar a qualidade desejada em todos os estágios do processo de desenvolvimento (YOURDON, 1996, MULLER, 1998).

Desde a década de 80, iniciaram-se esforços para melhorar processos de software, com o objetivo de melhorar a qualidade, aumentar a produtividade e diminuir os custos. Diferentes modelos são utilizados dependendo do mercado alvo das organizações de software (DEMIRÖRS *et al.*, 1998, WEBER e ROCHA, 1999).

Existem duas vertentes quanto à qualidade de processos de software: uma define normas para a padronização e certificação do processo e outra determina modelos para avaliar e aprimorar processos. Ambas se preocupam com a qualidade e a gerência do processo embora se diferenciem por suas filosofias de base (PAULK, 1994): os padrões identificam os requisitos mínimos para um sistema de qualidade enquanto que os modelos reforçam a necessidade de uma melhoria contínua do software.

II.6.1. Padronização e Certificação

A padronização do processo de software – através de roteiros, práticas ou procedimentos – define a maneira de realizar alguma atividade para atingir o resultado desejado (SOMMERVILLE, 1992). É necessária quando muitas pessoas, produtos ou ferramentas estão envolvidos para estabelecer ambientes comuns, integrar processamentos e conduzir os testes do sistema (SCHNEIDEWIND e FENTON, 1996, ANSI,

1999).

De acordo com Schneidewind e Fenton (1996), quando os padrões são utilizados em um processo de desenvolvimento de software, o produto final é de qualidade superior, pois a equipe de trabalho realiza as atividades com o apoio de um processo formal, documentado e disciplinado. Um processo de software padronizado define um processo comum a todos os projetos, descrevendo os elementos fundamentais que cada projeto de software deve incorporar. Uma série de fatores pode influenciar a aplicação de padrões (ESA, 1991): custo do projeto (nas fases de desenvolvimento e de operação); número de pessoas necessárias para desenvolver, operar e manter o software; número de usuários finais; número de linhas de código a serem produzidas; aplicações críticas (medidas pelas conseqüências de suas falhas); complexidade do software; e estabilidade dos requisitos. A gerência deve definir a padronização de um processo de software e sua respectiva documentação, que considere esses fatores.

Um exemplo de padronização de processo de software é o padrão PSS-05, definido pela Agência Espacial Européia (ESA, 1991) para satisfazer suas necessidades na produção de sistemas críticos, desenvolvidos na própria agência ou pela indústria de software. O padrão se aplica a programas, procedimentos, regras e documentação associados aos sistemas computacionais. A padronização se refere a subsistemas de um sistema mais complexo ou a sistemas independentes.

O padrão ISO 9000-3 (1991) – "Gerenciamento da Qualidade e Normas de Garantia da Qualidade Parte 3: Diretrizes para Aplicação da ISO 9001 em Desenvolvimento, Suprimento e Manutenção de Software" – define diretrizes para facilitar a aplicação do padrão ISO 9001 em organizações que desenvolvem, fornecem e mantêm software (ROTHERY, 1993, SCHMAUCH, 1994, ISO, 1999). Tais diretrizes destinam-se a descrever os controles e métodos sugeridos para a produção de software que atendam aos requisitos do comprador, evitando-se não conformidades em todos os estágios, desde o desenvolvimento até a manutenção (ABNT, 1993).

A norma ISO/IEC 12207 (1995) – "Processos de ciclo de vida de software" – estabelece uma estrutura de trabalho comum para processos de software através de atividades e tarefas que devem ser realizadas durante o ciclo de vida do soft-

ware (ISO/IEC 12207, 1995). A norma têm como objetivo definir, controlar e melhorar os processos de software, definindo os processos fundamentais (aquisição, fornecimento, desenvolvimento, operação e manutenção), de apoio (documentação, gerência da configuração, garantia da qualidade, verificação e validação, revisão conjunta, auditoria e resolução de problema) e organizacionais (gerência, infra-estrutura, melhoria e treinamento) (GRAHL *et al.*, 1997).

II.6.2. Avaliação e Melhoria

Com o objetivo de aprimorar continuamente o processo de software, é necessário avaliar com relativa regularidade a sua eficiência ou maturidade e, então, introduzir melhorias. Isto é denominado avaliação do processo de software e é composto por três partes (ESPITI, 1999): (1) uma estrutura de trabalho para classificar os níveis de maturidade do processo de software; (2) um algoritmo para calcular ou avaliar o atual nível de maturidade do processo de software; (3) um questionário para obter informações sobre o atual processo de engenharia de software de uma organização. As técnicas de avaliação e melhoria tipicamente reúnem um conjunto de objetivos a serem alcançados e uma lista para verificar o quanto uma organização atinge cada objetivo (SAKAMOTO *et al.*, 1998).

O conceito de maturidade de um processo de software está relacionado com a sua estrutura ou controle. À medida que um processo de software é melhor estruturado, certos problemas encontrados na construção de aplicações são resolvidos, indicando o amadurecimento do processo (GIBSON, 1997, SIMMONS *et al.*, 1998). Este amadurecimento permite que os gerentes de software se concentrem em outros problemas, aprimorando continuamente o processo de software (FALBO, 1996).

Humphrey (1990) considera um processo de software imaturo quando apresenta as seguintes características: prazos e orçamentos mal planejados; procedimentos para avaliar a qualidade do produto final inexistentes; e atividades para controlar a qualidade dos produtos intermediários não explícitas. Geralmente, os processos imaturos ou são improvisados durante o projeto de software ou não seguem nenhuma especificação, mesmo que definida previamente (PAULK *et al.*, 1997).

Em processos maduros, as atividades e responsabilidades de cada membro da equipe são bem determinadas e seguem um plano de trabalho. A qualidade dos produtos são continuamente monitoradas e os processos são atualizados, sempre que necessário, com base nos progressos obtidos (FALBO, 1996). As estimativas de prazos e custos são respeitadas, pois as restrições da organização são conhecidas, as metas de um projeto de software são bem definidas e as características do domínio da aplicação são identificadas.

O amadurecimento do processo de software traz, portanto, os seguintes benefícios para uma organização de software (HERBSLEB *et al.*, 1994, FOX e FRAKES, 1997, HERBSLEB *et al.*, 1997): redução de custos de desenvolvimento, alcançado através de uma alocação de recursos e tecnologia adequados à organização e ao sistema a ser construído; melhoria da previsão de cronogramas e orçamentos, uma vez que através da definição do processo de software, a equipe de trabalho sabe o que fazer, quando realizar determinada atividade e como utilizar uma informação; eliminação da duplicação de trabalho, pois os problemas podem ser rapidamente identificados e eliminados, evitando a sua propagação.

Uma estrutura de trabalho típica reúne cinco ou seis níveis de maturidade. Um processo de software é associado a um dos níveis de maturidade e então, é possível definir as melhorias necessárias para alcançar o próximo nível (ESPITI, 1999). Um processo que está no nível de maturidade mais baixo é considerado indefinido ou caótico. É impossível prever a qualidade de seus produtos e o tempo necessário para realizar as atividades (PAULK *et al.*, 1993b, EMAM *et al.*, 1998). Ao longo do tempo, a organização se familiariza com o domínio da aplicação e pode alcançar um desempenho consistente no que diz respeito a cronogramas e estimativas. Neste estágio, o processo depende das habilidades dos membros da equipe de desenvolvimento de software. Para melhorar, é imprescindível definir o processo de software e assegurar que ele seja seguido de maneira consistente, através de treinamento (HUTCHINGS *et al.*, 1993, CUSUMANO e SELBY, 1997). Atingido tal estágio, torna-se possível avaliar ou medir a qualidade dos produtos resultantes do processo de software e a eficiência do mesmo. Baseando-se nessa avaliação, o processo pode ser

otimizado (PAULK *et al.*, 1993a, PAULK *et al.*, 1997).

Os modelos de avaliação e melhoria da capacidade dos processos de software incorporam práticas que se aplicam a todas as atividades relacionadas, direta ou indiretamente, com o desenvolvimento de produtos de software (LOWE e COX, 1996, SIMMONS *et al.*, 1998, IRVING, 1999). Estes modelos devem permitir a medição da qualidade de maneira eficiente. As medições verificam se as técnicas realmente aprimoram a qualidade do software e como o processo de desenvolvimento afeta a qualidade dos produtos (KITCHENHAM e PFLEEFER, 1996). Também é importante identificar como o investimento em recursos levará à construção de produtos de alta qualidade.

O modelo CMM-SEI caracteriza a capacidade de uma organização em desenvolver software através de cinco níveis de maturidade, identificando as características e os problemas de uma organização em cada um dos níveis (PAULK *et al.*, 1993a, PAULK *et al.*, 1993b, PAULK *et al.*, 1997). Este modelo também especifica como atingir níveis mais altos, satisfazendo os requisitos de cada nível. Quando o nível de maturidade é conhecido, as ações necessárias para amadurecer o processo podem ser determinadas (PAULK *et al.*, 1993c, CURTIS, 1997).

Atualmente, exitem diversos modelos para apoiar a avaliação da maturidade de processos de software e identificar áreas-chave para permitir o seu aprimoramento (BANDINELLI *et al.*, 1995a, MULLER, 1998). Bootstrap baseia-se no modelo CMM, com algumas extensões e refinamentos, adaptando-o à indústria européia de software. O método Bootstrap é composto pelos seguintes elementos (HAASE *et al.*, 1994): uma hierarquia detalhada dos atributos de qualidade relacionados ao processo de software; uma versão aprimorada do algoritmo da SEI, que permite calcular o nível de maturidade para cada atributo de qualidade; uma versão melhorada do questionário da SEI para determinar a capacidade de uma organização em satisfazer cada um dos atributos de qualidade.

O modelo Trillium é baseado no CMM/SEI e as diferenças mais importantes entre os modelos são: a arquitetura do modelo é mapeada em áreas de capacidade, ao invés de ser baseada em áreas de processos; a perspectiva é dirigida ao produto

e não ao processo de software; os requisitos do usuário recebem maior importância; a maturidade tecnológica também é considerada (IRVING, 1999).

O projeto SPICE (*Software Process Improvement and Capability dEtermination*) (EMAM e GOLDENSON, 1996b, SPICE, 1999), tem como meta tornar-se um padrão internacional para avaliação e melhoria de processo de software. Para tanto, SPICE baseia-se nas características dos métodos e modelos de maturidade de processo de software existentes. SPICE oferece uma estrutura de trabalho para avaliar processos de software de organizações envolvidas no planejamento, gerência, monitoração e controle do desenvolvimento, operação, evolução, aquisição, fornecimento e suporte de software (PAULK *et al.*, 1997, EMAM *et al.*, 1998). O objetivo do SPICE é apoiar a determinação da capacidade e a melhoria dos processos de software em diferentes domínio de aplicação, desconsiderando estruturas organizacionais e utilizando critérios quantitativos. Os resultados obtidos são apresentados através de um perfil, ao invés de números ou expressões binárias (sucesso ou falha), podendo ser comparados com os resultados provenientes de outros modelos de avaliação semelhantes (EMAM *et al.*, 1998).

II.7. Desenvolvimento de Software por Equipes Distribuídas

Inicialmente, as pesquisas relacionadas à descentralização do desenvolvimento de software propuseram mecanismos para evitar que as equipes de trabalho tivessem que estar na mesma hora e local para realizar determinadas tarefas (STEIN *et al.*, 1997). O objetivo era decompor e distribuir o trabalho, apoiando a cooperação e a comunicação entre as equipes de software.

O conceito de *viewpoint* foi utilizado em modelos de processo descentralizados para orientar a verificação de consistência e a resolução de conflitos em ambientes de desenvolvimento de software (LEONHARDT *et al.*, 1995). Um *viewpoint* é um objeto fracamente acoplado e localmente gerenciado que encapsula parte do conhecimento sobre o sistema e seu domínio, especificado através de um esquema adequado

ao processo do projeto (FINKELSETIN *et al.*, 1992). Um *viewpoint* pode integrar um artefato do processo de software (a especificação do sistema, por exemplo) com o conjunto de atividades necessárias para construí-lo. O protótipo denominado *The Viewer* oferece um mecanismo de comunicação baseado em mensagens entre os *viewpoints* e, portanto, entre os modelos descentralizados de processo (LEONHARDT *et al.*, 1995). Expressões regulares também foram utilizadas para definir a comunicação entre processos de software, ou seja, a troca de informações entre as equipes de software (IIDA *et al.*, 1993). A identificação das dependências entre os produtos intermediários do software foi utilizada para evitar os conflitos que surgem com a mudança de artefatos desenvolvidos por diferentes grupos de trabalho (NARAYANASWAMY e GOLDMAN, 1992). O modelo *Actor-Dependency* foi utilizado para modelar e analisar uma organização de manutenção de software em grande escala (BRIAND *et al.*, 1995). O objeto de estudo foi o projeto *Fiht Dynamics Division* da NASA, onde 80 pessoas estavam envolvidas na manutenção de 3.5 milhões de linhas de código. Através desta pesquisa, pode-se concluir que um apoio computacional é essencial para realizar análises quantitativas, que envolvem uma grande quantidade de informações e profissionais.

Como os projetos de software estão se tornando cada vez maiores e mais complexos, a necessidade de serem realizados por equipes geograficamente dispersas se tornou real e concreta. O conhecimento e a capacidade imprescindíveis para o desenvolvimento de alguns projetos, muitas vezes, não se encontram em um mesmo local de trabalho, o que leva à distribuição de tarefas entre diversos departamentos de uma mesma organização ou, até mesmo, entre diferentes organizações (WET, 1999). Profissionais provenientes de diferentes áreas de conhecimento (economia, finanças, direito, computação, etc) trabalharão em diferentes lugares para atingir os objetivos de uma mesma empresa. A realização de trabalhos globalmente distribuídos envolve muitos problemas relacionados a diferentes línguas, diferenças culturais, diferentes horários, legislações, códigos de ética, equipamentos computacionais e software (MAURER e DELLEN, 1998). Embora tais problemas tenham causas sociais, alguns deles podem ser amenizados ou, até mesmo, solucionados, através de

tecnologias avançadas. O compartilhamento de informações, independentemente da sua localização, levou ao surgimento de um novo tipo de organização: as empresas virtuais. Uma empresa virtual pode ser formada por grandes corporações (constituindo vários grupos em diferentes localidades) ou por indivíduos que trabalham de forma independente a qualquer conexão corporativista (FIELDING *et al.*, 1998). O que é necessário para formar uma empresa virtual consiste em um objetivo comum, uma base de informações compartilhada, a coordenação das tarefas e profissionais.

As pesquisas realizadas anteriormente, embora direcionadas à cooperação e não tratando diretamente do problema de dispersão geográfica, apoiaram os trabalhos relacionados ao desenvolvimento distribuído de software. As abordagens mais recentes argumentam que novas técnicas sejam criadas para apoiar os aspectos mais críticos no desenvolvimento disperso de software, identificados por alguns autores, como sendo (PERPICH *et al.*, 1997, MAURER e KAISER, 1998, WET, 1999): a troca de informações, acesso ao conhecimento, realização de atividades cooperativas, coordenação de tarefas assíncronas e registro da experiência adquirida para posterior reutilização. Portanto, os ambientes de desenvolvimento distribuído de software devem apoiar as seguintes atividades (STEIN *et al.*, 1997): registro e disponibilização dos tópicos de discussão, compartilhamento de informação, rastreamento do raciocínio, ligações a documentos de apoio, tomada de decisões, coordenação e histórico do trabalho. Muitas pesquisas em engenharia de software para equipes distribuídas procuram resolver os aspectos técnicos do desenvolvimento de sistemas computacionais, tais como, inspeção de código (PERPICH *et al.*, 1997, STEIN *et al.*, 1997), estruturação de arquivos (BAENTSCH *et al.*, 1995), conexão entre os artefatos (KAISER *et al.*, 1997), autoria distribuída (WHITEHEAD e WIGGINS, 1998, WAIBA, 1999), mecanismos de busca (WAIBA, 1999), fluxo de trabalho (BOLCER e KAISER, 1999, METEOR, 1999), revisão de documentos (KONNO *et al.*, 1998) e gerência da configuração (CMS, 1997).

Pesquisas sobre bases de dados distribuídas ou remotas já foram bastante exploradas. A construção de repositórios para ambientes de desenvolvimento de software na Internet requer a integração de ferramentas convencionais (compiladores, depura-

dores de código, etc) com mecanismos de apoio à edição, comunicação e inserção de documentos (KAISER *et al.*, 1997). A troca de informações entre os diferentes desenvolvedores precisa ser gerenciada e coordenada através de mecanismos de acesso multi-usuário (WET, 1999). O protótipo EDEM (*Expectation-Driven Event Monitoring*) permite o registro de informações sobre a utilização de uma aplicação. Através do EDEM, podem-se registrar agentes e associar um conjunto de dados a um determinado agente, permitindo uma estruturação dinâmica das informações (HILBERT e REDMILES, 1998a). O sistema WebOQL, desenvolvido pela Universidade de Toronto, permite visualizar dados na Internet de maneira uniforme. O modelo conceitual do sistema é baseado na gerência de coleções de dados heterogêneas (AROCENA e MENDELZON, 1998). A mesma abstração é aplicada em projetos sobre sistemas virtuais para gerência de documentos (BALASUBRAMANIAN e BASHIAN, 1998).

Pesquisas sobre *groupware* procuram resolver os problemas que surgem quando atividades cooperativas, tais como reuniões, precisam ser realizadas por equipes geograficamente distribuídas (WET, 1999). O enfoque é o apoio computacional a tarefas síncronas e assíncronas em um espaço de trabalho (*workspace*) virtual. Entretanto, o apoio computacional não é suficiente pois as atividades assíncronas precisam ser coordenadas e, em muitos projetos de software, as relações entre os artefatos podem estar implícitas, podendo levar a inconsistências se os profissionais envolvidos não forem alertados previamente sobre os potenciais problemas (WET, 1999). Sistemas de apoio à modelagem e execução de processos podem apoiar a resolução de tais problemas ao explicitamente representar as relações entre os produtos intermediários e notificar as pessoas envolvidas quando as mudanças ocorrerem (ISAKOWITZ *et al.*, 1998, MAURER e KAISER, 1998).

Muitos trabalhos sobre modelagem, execução e melhoria de processos foram realizados na última década (descritos nas seções anteriores), mas a maioria não apóia explicitamente equipes geograficamente distribuídas (KAISER *et al.*, 1997). Algumas pesquisas nesta direção já começaram a surgir. O protótipo CoMo-Kit foi desenvolvido para resolver o problema de reunir diferentes conhecimentos em um único projeto de software ao permitir a representação do conhecimento sobre um processo

de software no próprio processo da organização. O protótipo apóia a modelagem e a execução de processos definidos através da linguagem MILOS (MAURER e DELLEN, 1998). O ambiente Serendipity-II apóia a modelagem e execução de processos síncronos e assíncronos na Internet, oferecendo diferentes visões sobre a descrição do processo de software e coordenando o fluxo de trabalho através de mecanismos de comunicação na Internet (GRUNDY *et al.*, 1998).

O surgimento do Web incentivou a extensão de alguns projetos sobre ambientes centrados em processos. É o caso do OzWeb (KAISER *et al.*, 1997), que reutiliza os mecanismos de gerência transacional entre as ferramentas integradas pelo ambiente Oz (VALETTO e KAISER, 1996) em uma arquitetura cliente/servidor. AISA (*Asynchronous Inspector of Software Artifacts*) é uma ferramenta para inspeção de software distribuída e assíncrona na Web que estende o protótipo CAIS (*Collaborative Asynchronous Inspector of Software*), o qual foi implementado em Lotus Notes (STEIN *et al.*, 1997). O sistema ASP (*Agile Software Process*) reúne diversas unidades de processo, permitindo que as equipes possam trabalhar, ao mesmo tempo, em diferentes versões de um software (AOYAMA, 1998a). O ASEE (*Agile Software Engineering Environment*) utiliza o mecanismo de compartilhamento de dados WAIN (*Wide-Area Information Network*) para apoiar o ASP na Internet. Atualmente, o ASEE está sendo integrado ao Web (AOYAMA, 1998b).

A tecnologia WWW oferece mecanismos para o desenvolvimento geograficamente distribuído de software, tais como compartilhamento de dados e apoio à coordenação através de mensagens eletrônicas. O Web oferece uma infra-estrutura para um ambiente de engenharia de software global, apoiando a evolução de um produto computacional desde a concepção até a sua implementação, não importando a localização ou o número de pessoas envolvidas (FIELDING *et al.*, 1998). A tecnologia Web está presente em diversos tipos de aplicações (intranets, comércio eletrônico, etc), entretanto, alguns mecanismos ainda precisam ser desenvolvidos (ISAKOWITZ *et al.*, 1998). No que se refere especificamente ao processo de software, devem-se identificar, explorar, desenvolver e aplicar técnicas mais apropriadas aos ambientes de construção de software (LEHMAN, 1997). A tecnologia de apoio às empresas vir-

tuais, que, atualmente, não se encontra na Web, são: relacionamento entre artefatos e processos, modelo de integração flexível entre os diferentes tipos de dados (código, especificação, diagramas, etc), anotação distribuída, visualização de artefatos, autoria distribuída, coordenação distribuída e gerência de mudanças (FIELDING *et al.*, 1998, WHITEHEAD e WIGGINS, 1998). Tais mecanismos são necessários tanto na engenharia de software como em outras áreas que envolvem o desenvolvimento geograficamente disperso de sistemas de informação complexos (MAURER e DELLEN, 1998).

II.8. Direções Estratégicas em Processo de Software

As atuais pesquisas sobre processo de software podem ser categorizadas em: visualização de processos de software, que oferecem visões gráficas de modelos e suas instâncias, permitindo uma navegação e edição interativa; execução de processos, através da execução de modelos de processo através da chegada de eventos (serviços) para determinar seu comportamento, permitindo uma análise dinâmica para identificar anomalias; monitoração e medição, que permite a coleta de dados do processo, documentando quais ações ocorrem e em que ordem; evolução de processos, que tratam do aprimoramento contínuo de processos e gerência do modelo (ATRIUM, 1997, WANG *et al.*, 1998).

Segundo C. Gunter (GUNTER *et al.*, 1996), as pesquisas sobre linguagens específicas ao domínio da aplicação, como por exemplo SLANG (BANDINELLI *et al.*, 1995a), permitirão acelerar o processo de desenvolvimento de software e reduzir custos da produção. Tais linguagens servem para especificar e projetar, através de notações específicas a um domínio de aplicação.

S. M. Sutton e L. J. Osterweil (SUTTON e OSTERWEIL, 1996) acreditam que os processos de software podem ser desenvolvidos e gerenciados em famílias, assim como os produtos de software. Uma família reúne processos ou projetos semelhantes, que se referem a um mesmo domínio do conhecimento. Uma família de processos oferece uma base para a aplicação sistemática de tecnologia para apoiar a aquisição

de requisitos, modelagem, execução, evolução e reutilização de processos. Esta perspectiva orientada a famílias permite que processos de software possam ser facilmente especificados, evoluídos e automatizados (GUNTER *et al.*, 1996).

Devido à grande variedade de ferramentas e técnicas que estão surgindo, L. Osterweil (OSTERWEIL *et al.*, 1996) e Reiss (REISS, 1996) acreditam que as pesquisas levem à construção de ambientes que combinem um alto grau de integração de ferramentas com uma flexibilidade permitindo a integração de novas tecnologias. Para tanto, torna-se necessário definir um processo de software que permita adquirir informações sobre a qualidade dos produtos intermediários e então, verificar o impacto de mudanças no processo de desenvolvimento. O aprimoramento de um processo de software pode ser alcançado através da análise do desempenho dos projetos (LEHMAN, 1995, WASSERMAN, 1996). Medição é um pré-requisito necessário para melhoria do processo de software (BRÖCKERS *et al.*, 1996, WEBER, 1997).

II.9. Considerações Finais

Este capítulo apresentou a evolução das pesquisas sobre processo de software. Inicialmente, a ênfase era na estruturação e modelagem de processos de software. As linguagens de modelagem de processos de software devem respeitar as características dos processos, definidos como um conjunto de atividades de engenharia de software necessárias para transformar os requisitos do usuário em um sistema computacional e para evoluir o próprio processo.

A automação e execução de processos de software facilita o seu entendimento, utilização e aprimoramento. Ambientes centrados em processos são utilizados para orientar desenvolvedores, automatizar as atividades de software e oferecer informações sobre o desempenho e o estado de processos de software. Algumas propostas de ambientes foram prototipadas e podem-se encontrar ferramentas comerciais que apóiam a gerência do processo de desenvolvimento de software.

A qualidade dos produtos de software e a produtividade das equipes de trabalho está diretamente relacionada à qualidade do processo de software utilizado para

construi-los. A busca da qualidade de processos de software estimulam a implantação de normas tais como a ISO/IEC 12207 e a aplicação de modelos que buscam a melhoria contínua do processo de software. O SEI desenvolveu o CMM e outros modelos foram propostos, culminando com uma padronização através do projeto internacional SPICE.

A produção de sistemas computacionais de qualidade a um custo razoável ainda requer novas pesquisas sobre linguagens de modelagem, famílias de processos, medições e análise de desempenho de projetos e integração de novas tecnologias, considerando as características e necessidades dos ambientes de desenvolvimento geograficamente disperso de software.

III. WWW e o Desenvolvimento de Software na Internet

Este capítulo relata a extensa utilização da Internet e a sua importância no desenvolvimento geograficamente distribuído de produtos de software. A tecnologia WWW, seus protocolos e linguagens de comunicação para a rede de computadores que foram utlizados neste trabalho são apresentados. Ao final, descreve-se o estado da arte em sistemas de apoio ao desenvolvimento de software em rede.

III.1. A Engenharia de Software na Era da Internet

A conexão entre diversos computadores através de uma rede mundial possibilita que as organizações e empresas de software enfrentem os atuais desafios do desenvolvimento de software. Com a Internet, a construção de sistemas computacionais pode envolver, mais facilmente, profissionais que não necessariamente se encontrem em um mesmo local, integrando especialistas de diferentes domímios de conhecimento (ISAKOWITZ *et al.*, 1998). Desta forma, a Internet apóia a existência de empresas virtuais para o desenvolvimento globalmente distribuído de software (MAURER e KAISER, 1998, PERRY *et al.*, 1998).

A Internet oferece uma infra-estrutura de comunicação, através da qual os membros de uma equipe de software podem facilmente trocar informações e os resultados possam ser rapidamente publicados (STEIN *et al.*, 1997, ISAKOWITZ *et al.*, 1998). Ferramentas de gerência de projetos, que integram modelagem e execução de processos, informam o andamento do trabalho e orientam as atividades a serem realizadas (MAURER e DELLEN, 1998). Sistemas que implementam o fluxo de trabalho na Web facilitam a coordenação de tarefas, a identificação de recursos necessários para a realização do projeto e a resolução de problemas (CIANCARINI *et al.*, 1996, KUDO *et al.*, 1998, MAURER e KAISER, 1998). Técnicas de trabalho cooperativo podem ser utilizadas para construir ambientes virtuais na Web, através

de salas de reunião e repositórios de produtos intermediários e documentos (SANDEWALL, 1996, VOSSEN *et al.*, 1996, KONNO *et al.*, 1998, MAIDANTCHIK, 1998b). Os artefatos de software podem ser rapidamente enviados aos membros do grupo, situados pelo mundo. As ferramentas de desenvolvimento de software podem ser utilizadas através de serviços Web ao invés de compradas e instaladas em computadores locais (CALLAHAN e RAMAKRISHNAN, 1998). A coleta e a interpretação de métricas podem ser realizadas através da Internet (CALLAHAN e RAMAKRISHNAN, 1998, CALLAHAN *et al.*, 1998). Repositórios contendo experiências adquiridas com projetos passados podem ser disponibilizados na Web, permitindo que os grupos de trabalho utilizem tal conhecimento para enfrentar novos problemas (CERI *et al.*, 1998, HILBERT e REDMILES, 1998b, MAIDANTCHIK *et al.*, 1998c).

A extensa utilização da Internet, não só na área de software mas também em todos os setores, foi impulsionada pelo World-Wide Web, que integra duas técnicas fundamentais – o protocolo HTTP (seção III.3.3.) e a linguagem hipertextual HTML (seção III.3.5.) – combinadas com as facilidades oferecidas pelas interfaces gráficas dos navegadores (RADA *et al.*, 1998).

Os mecanismos oferecidos pelo WWW influenciam como software é desenvolvido e permitem avançar o estado da arte em engenharia de software, redefinindo a maneira como um sistema computacional é especificado, projetado, implementado, mantido e aprimorado (AOYAMA, 1998b, FIELDING *et al.*, 1998, MULLER, 1998).

III.2. Hipertextos e Hipermídia

Os sistemas de hipermídia permitem a criação, associação e compartilhamento de uma grande variedade de informações, tais como documentos, gráficos, vídeos, filmes animados, sons e programas computacionais. Hipermídia consiste em um método não-linear de acesso a arquivos de maneira inovadora em relação aos tradicionais sistemas de documentação, os quais são seqüenciais por natureza (BALASUBRAMANIAN, 1999). Tais sistemas oferecem um acesso flexível às informações por

incorporar mecanismos de navegação, anotação e apresentação tutorial.

Hipertextos são definidos como um tipo de sistema de gerência de informação onde os dados são armazenados em uma rede de nós conectados entre eles. Os nós podem conter textos, imagens, códigos ou outros tipos de dados. Hipertextos com multimídia (som, vídeo, etc.) são chamados de hipermídia. A grande revolução dos sistemas de hipermídia é a capacidade de produzir grandes, complexos e inter-conectados grupos de informação.

Em 1965, Nelson criou a palavra hipertexto (texto não-linear) e definiu-o como "um conjunto de material inter-conectado, contendo suas relações e permindo que aqueles que o examinem possam fazer anotações e comentários" (NELSON, 1965). Entretanto, a idéia original de hipertextos foi descrita por Bush em 1945, através de um mecanismo denominado *memex*, no qual um indivíduo pode armazenar documentos, registros e discussões de maneira a permitir sua consulta de forma rápida e flexível (BALASUBRAMANIAN, 1999). Desta maneira, Bush descreveu a possibilidade de conectar diferentes tipos de itens.

A integração de hipertextos a aplicações computacionais, tais como os sistemas de ajuda, é muitas vezes definido como apenas um outro tipo de interface com o usuário. Entretanto, um hipertexto corresponde a um sistema híbrido que oferece uma base de dados e permite um acesso direto aos dados. Estes sistemas podem implementar um esquema para representação de informações, uma rede semântica, que integra textos informais com processos mecânicos e formais. O mecanismo essencial dos sistemas de hipertexto é a implementação de relações entre informações, que permite uma organização não linear de documentos. Concluindo, um sistema de hipertexto corresponde a um banco de dados que oferece um único método de acesso às informações, enquanto os banco de dados tradicionais incorporam alguma estrutura (NIELSEN, 1990). Em um sistema de hipertexto, o usuário tem a liberdade de explorar as informações de diferentes maneiras.

III.3. O Sistema World-Wide Web

O World-Wide Web é um sistema multimídia distribuído, heterogêneo e colaborativo. WWW é um sistema cliente-servidor para a Internet, projetado para colaborações numerosas, distribuídas e assíncronas (ANDREWS, 1996). Inicialmente, o WWW foi concebido no CERN para permitir que os físicos de altas energias pudessem acessar os dados em qualquer lugar de maneira uniforme. Atualmente, o seu escopo é muito maior (WWW, 1999).

Antes do WWW, para encontrar uma informação na rede de computadores, o usuário tinha que ter acesso a vários terminais, conectados a diferentes computadores e, conseqüentemente, aprender diversos programas para consultar os dados. O princípio de acesso universal, incorporado ao WWW, fundamenta-se no seguinte: desde que uma informação esteja disponível, o sistema deve permitir o seu acesso por qualquer tipo de computador, em qualquer país, e um usuário autorizado pode utilizar somente um único programa para resgatá-la. O usuário não precisa saber onde a informação está localizada ou aprender comandos complexos para acessá-la.

A Internet contém inúmeras informações sobre diversos domínios do conhecimento. No passado, a tarefa de encontrar e utilizar as informações não era trivial. Com o surgimento do WWW, os servidores oferecem as informações de maneira consistente e os clientes as apresentam de forma estruturada. O principal mecanismo do WWW corresponde às relações entre documentos. Tais relações hipertextuais podem apontar para qualquer informação armazenada em um computador conectado à Internet e transformam a rede em um *Web* de informações (DE BRA, 1996).

Poucos anos após sua concepção, o WWW se tornou um fenômeno social. Atualmente, o sistema divulga informações sobre o governos, universidades, empresas e indivíduos, apresentando a Internet a um público global e abrangente. Ao invés de acessar um único disco local, o Web utiliza o mundo inteiro como biblioteca.

III.3.1. Histórico e Revolução

Em 1989, Tim Berners-Lee propôs um sistema distribuído para o CERN baseado em técnicas de hipertextos, como uma maneira de relacionar pedaços de informações armazenadas nos computadores. Robert Cailliau integrou-se ao projeto, concentrando-se nos objetivos iniciais de desenvolver ferramentas para a comunidade dos físicos, enquanto Tim Berners-Lee continuou a desenvolver o WWW. Durante o ano de 1990, os primeiros *browser* e *server* foram desenvolvidos, embora limitados ao sistema operacional NeXTStep. Em 1991, um estudante técnico desenvolveu um navegador simples que pode ser utilizado em diferentes plataformas computacionais. Esta flexibilidade aos diversos computadores existentes despertou um grande interesse na comunidade científica, inicialmente, nos centros de pesquisa de física de altas energias: o diretório eletrônico do laboratório DESY (*Deutsches Elektronen-SYnchrotron*), na Alemanha, pôde ser consultado no FERMILAB (*Enrico Fermi Laboratory*), nos Estados Unidos; o banco de dados do SLAC (*Stanford Linear Accelerator Center*), nos Estados Unidos, pôde ser acessado no CERN, na Suíça. Tudo isso pôde ser realizado sem necessitar aprender os sistemas específicos dos computadores remotos.

O passo crucial veio, em 1993, quando o sistema de hipermídia distribuído Mosaic para X Window foi desenvolvido pelos pesquisadores Marc Andreessen e Eric Bina do *National Center for Supercomputing Applications* (NCSA), nos Estados Unidos. Este software permitiu que imagens coloridas fossem apresentadas através da Internet e outros formatos fossem facilmente acessados, através de uma interface altamente amigável para plataformas Unix. NCSA também desenvolveu outras versões para Apple/Macintosh e Microsoft/Windows, levando o WWW a um grande público.

O sistema se espalhou rapidamente na comunidade científica tornando-se indispensável. Desde então, o WWW cresceu exponencialmente muito além da comunidade científica. O WWW compreende milhares de servidores através dos quais milhões de usuários "navegam", aprendem e se comunicam. Raramente, uma nova tecnologia, desenvolvida para fins de pesquisa, produziu tão rápida e importante

aplicação em diversas áreas.

III.3.2. Os Navegadores

Os chamados navegadores são sistemas de hipermídia distribuídos, projetados para acessar informações na Internet. Os navegadores oferecem uma única interface aos diversos protocolos de comunicação e arquivos localizados na Internet, integrando poderosos métodos para procurar, utilizar e compartilhar informações (ANDREESSEN, 1993).

Os navegadores utilizam um modelo cliente/servidor para a distribuição de informação: o servidor funciona em uma máquina conectada à Internet atendendo às consultas encaminhadas pelos clientes, que podem localizar-se em qualquer lugar da Internet. As unidades de informação, enviadas pelos servidores, podem conter linhas de texto, textos formatados, gráficos, sons e outros tipos de dados e podem ser conectadas a outras unidades residentes na Internet. Os navegadores combinam mecanismos de hipermídia, informação distribuída, organização hierárquica e busca para recuperação de dados armazenados em qualquer lugar do mundo desde que possam ser acessados pela Internet (HARTMAN *et al.*, 1997, NETSCAPE, 1999).

III.3.3. O Protocolo de Comunicação HTTP

O HTTP (*HyperText Transfer Protocol*) é um protocolo de comunicação para a Internet, que pode ser utilizado em qualquer aplicação do tipo cliente/servidor (STALLINGS, 1997). Tem como características, além da alta velocidade, uma grande estabilidade e a facilidade de permitir extensões. Os protocolos através dos quais os clientes WWW podem comunicar-se são: FTP (*File Transfer Protocol*), WAIS (*Wide Area Information Servers*), Gopher e NNTP (*Network News Protocol*).

O protocolo HTTP é um sistema para transferência de arquivos em rede mais eficiente que o popular FTP pois permite que o WWW manipule os problemas referentes a diferentes tipos de informações utilizando uma negociação para representações de dados. Tal mecanismo possibilita que um conjunto extensível de métodos possa ser utilizado. Quando a representação de um determinado objeto é identificada, os

métodos indicam o procedimento a ser realizado.

O método POST do protocolo HTTP cria um novo objeto ligado e subordinado a um outro objeto. O conteúdo do novo objeto é encapsulado no corpo da consulta. Através do método GET, os dados são inseridos ao final do endereço do servidor. Ao realizar uma consulta, o servidor pode retornar arquivos hipertextuais, imagens ou outros tipos de dados que correspondem ao novo objeto. A relação e o objeto são dinâmicos, ou seja, o resultado pode variar de acordo com as informações enviadas pelo método ou com o momento da consulta como, por exemplo, quando um cliente quer verificar se uma informação está presente no banco de dados do servidor (BERNERS-LEE e OTHERS, 1996).

III.3.4. O Padrão URL

O padrão URL (*Uniform*[1] *Resource Locator*) corresponde a uma representação compacta da localização e método de acesso de um objeto – arquivos, documentos, imagens ou som – através da Internet (STALLINGS, 1997). Esta forma consistente de endereçamento permite que qualquer objeto, localizado em algum lugar na Internet, seja descrito, mesmo que este objeto seja acessado através de diferentes protocolos.

Os URLs podem apontar para documentos residentes em servidores FTP ou HTTP, artigos em servidores NNTP, acessos a banco de dados WAIS, mecanismos de busca a servidores Gopher ou praticamente qualquer tipo de dado disponível em algum lugar da rede que ofereça um mecanismo de acesso baseado em redes. O resultado corresponde a um processo completamente transparente para localização e recuperação de dados. O sistema realiza todas as ações necessárias, evitando que o usuário se envolva com os detalhes técnicos para navegar ou acessar dados na rede, permitindo que o cliente se concentre nas informações (ANDREESSEN, 1993).

[1] *A letra "U" em URL foi inicialmente designada a palavra* Universal.

III.3.5. A Linguagem HTML

A linguagem HTML (*HyperText Markup Language*) é definida em termos do padrão ISO *Standard Generalized Markup Language* (SGML) e possibilita a formatação de informações hipermídia (RAGGETT e OTHERS, 1997). O resultado é um simples documento, que permite a criação de textos estruturados com ligações hipertextuais. A escolha do SGML não foi feita devido a um mérito técnico específico, mas por permitir uma descrição da estrutura lógica do documento ao invés da sua formatação. Esta característica permite que a informação seja apresentada em diferentes plataformas utilizando várias convenções e fontes.

HTML é, portanto, uma linguagem lógica que utiliza marcadores (*markups* ou *tags*) para definir as diferentes partes de um hipertexto. A linguagem permite que o autor crie um documento, fazendo comentários na sua estrutura e não especificando como deverá ser apresentado ao leitor. Estas duas visões são referenciadas como física e lógica, definindo partes do texto como, por exemplo, *itálico*, no primeiro caso, ou *enfatizado*, no segundo. Logo, um arquivo em HTML pode ser considerado como a descrição lógica de um documento, utilizando comandos para descrever o conteúdo do mesmo. Esta descrição lógica sugere uma maneira de formatar o documento. Todavia, o verdadeiro formato é decidido pelo usuário que nageva através dos arquivos contectados.

A linguagem HTML contém vários elementos para uma entrada de dados pelo usuário. O mais simples é o elemento ISINDEX, que lê uma única linha de texto. O mais completo é o elemento FORM e seus sub-elementos INPUT, TEXTAREA e SELECT, que permitem entradas de dados mais sofisticadas através de formulários hipertextuais, tais como: selecionamento de botões (elemento INPUT), listas selecionáveis (elemento SELECT) e bloco de textos (elemento TEXTAREA). A entrada de dados em um formulário HTML é enviada ao servidor, cujo URL é especificado pelo atributo ACTION do elemento FORM. O elemento FORM também pode selecionar o método HTTP pelo qual os dados são enviados.

Quando o usuário submete uma entrada de dados através do formulário HTML, estas informações são codificadas e transmitidas ao servidor, onde serão interpreta-

das e processadas por um programa.

Os navegadores introduziram um novo modelo para os documentos HTML, os chamados *FRAMES*, que permitem especificar um página hipertextual em várias e independentes subjanelas de diferentes formatos, associadas a documentos hipertextuais distintos. Um FRAME HTML não contém o corpo de um documento hipertextual, mas um conjunto de elementos FRAMESET que definem o tamanho, a localização e outras dados sobre cada um dos FRAMES, cada qual apresentando informações distintas. Desta maneira, um FRAME pode ser utilizado para atualizar o conteúdo de outros FRAMES como, por exemplo, o cliente tem acesso ao sumário ao mesmo tempo que navega pelas seções de um hipertexto.

III.3.6. Programas CGI

CGI (*Common Gateway Interface*) é um padrão pelo qual programas externos interagem com seus servidores HTTP (COAR e OTHERS, 1998). Um documento HTML recuperado pelo WWW é estático, ou seja, o arquivo-texto, uma vez criado, não se modifica. Um programa CGI, por outro lado, é executado em tempo real, de maneira que possa apresentar informações dinâmicas. Os programas CGI atuam, portanto, como *gateways* entre o servidor HTTP e os dados ou programas locais. Um programa CGI pode manipular uma consulta a um banco de dados, recuperar o resultado e, a partir deste, gerar um documento HTML, retornando-o ao cliente.

Uma vez que um programa CGI pode ser executado, diz-se que esta estrutura é equivalente a permitir que o mundo execute um programa em qualquer sistema servidor. CGI utiliza variáveis do sistema (*environment variables*) para enviar os parâmetros – por exemplo, a entrada de dados de um formulário HTML – a um programa.

Os programas CGI podem retornar uma variedade de tipos de documentos, tais como: images, arquivos HTML, sons, referências a outros documentos, etc. Como o cliente deve saber que tipo de documento o servidor está enviando, o programa CGI deve informar ao servidor qual tipo MIME (*Multipurpose Internet Mail Extensions*) do documento que está sendo retornado.

III.3.7. As Linguagens Java e JavaScript

Inicialmente, a linguagem Java era destinada a controlar produtos eletrônicos domésticos por rede, uma das mais revolucionárias criações da Sun Microsystems. Com o WWW, a idéia era a de utilizar esta linguagem para criar programas executáveis que seriam enviados juntamente com uma página hipertextual, combinando animações, formulários interativos e programas. Java é conhecida como uma "plataforma computacional virtual" e suas aplicações (ou *applets*) se diferem das outras por residir em servidores na rede.

JavaScript é uma outra linguagem voltada para a WWW, desenvolvida pela Netscape. JavaScript é uma linguagem interpretada, compacta e baseada em objetos para desenvolvimento de aplicações cliente/servidor, similares aos programas CGI. De aprendizado mais fácil, o JavaScript colaborou para que a idéia dos programas executáveis distribuídos pela Internet prosperasse. Os navegadores podem interpretar aplicações JavaScript embutidas nas páginas HTML. Quando um cliente consulta uma destas páginas, o servidor envia o conteúdo completo do documento, que contém HTML e JavaScript, através da rede, para o cliente. Então, o navegador apresenta a página HTML e executa o programa JavaScript, produzindo resultados que o usuário pode visualizar, sem sobrecarregar a transmissão de dados pela rede, pois as tarefas são realizadas localmente no cliente.

Embora se pareçam, as linguagens Java e JavaScript apresentam diferenças fundamentais. O código do programa Java é transformado em uma estrutura chamada *bytecode*, que vai para o cliente juntamente com a página requisitada, sem, no entanto, fazer parte da página HTML. Java é uma linguagem de programação orientada a objetos, projetada para permitir uma execução rápida e para garantir a segurança de dados. JavaScript suporta muitas expressões sintáticas e construções básicas de controle de fluxo da linguagem Java. Entretanto, JavaScript descende das linguagens mais simples, que permitem uma estruturação dinâmica dos dados, tais como HyperTalk e dBASE. Embora de aprendizado mais fácil, a linguagem JavaScript é menos poderosa que a Java. Outra diferença entre as linguagens Java e JavaScript é que um programa Java pode ser compilado, podendo ser executado fora da rede,

enquanto que o JavaScript só permite construir aplicações para serem veiculadas em páginas WWW (JAVASCRIPT, 1997).

III.4. Sistemas de Apoio ao Desenvolvimento de Software em Rede

No início dos anos 70, as ferramentas CASE apoiavam a formatação do código-fonte, a programação estruturada e plano de testes (CYBULSKI e REED, 1992). Posteriormente, a ênfase passou para o desenvolvimento de ambientes de apoio interativo à construção de sistemas computacionais. Em projetos de software, a quantidade e a complexidade das informações dificultam a identificação de relacionamentos entre os documentos (BIEBER e KACMAR, 1995).

Vários trabalhos na aŕea de engenharia de software utilizam técnicas de hipertexto na construção de ambientes de desenvolvimento de software e ferramentas CASE. O mecanismo de navegação permite um acesso rápido aos diferentes níveis de detalhamento de um projeto (HILBERT e REDMILES, 1998b). A compreensão do usuário aumenta devido aos mecanismos oferecidos pela interface como, por exemplo, a agregação de textos e gráficos. Hipertextos podem facilmente integrar os vários aspectos de um processo de construção de software. O surgimento do WWW permitiu a utilização de ferramentas de apoio ao desenvolvimento de aplicações computacionais por equipes de trabalho geograficamente dispersas. CooMan (*Global Collaborative Project Management System*), desenvolvido na COPPE/UFRJ, é um sistema de apoio à comunicação, coordenação e compartilhamento de informações entre equipes dispersas através do WWW (SOUZA e MEDEIROS, 1995).

As seções a seguir descrevem alguns dos projetos que introduzem hipertextos e a tecnologia WWW em ambientes de software. Os projetos são apresentados em ordem cronológica, permitindo identificar a evolução das pesquisas. Os sistemas DHT e HyperCASE utilizam técnicas hipertextuais em uma plataforma cliente/servidor em uma rede restrita de computadores. Os demais projetos, desenvolvidos posteriormente, já utilizam o WWW para oferecer seus recursos através da Internet.

III.4.1. O Sistema Distribuído HyperText

O sistema distribuído HyperText (DHT) integra repositórios heterogêneos, cada qual sob uma administração autônoma e independente, com interfaces que manipulam tipos de dados diversos e oferecem diferentes modos de visualizar as relações entre os mesmos itens de informação (NOLL e SCACCHI, 1991). Os requisitos principais para integração de repositórios no sistema DHT são: organização, flexibilidade, transparência e autonomia. O solução para esta integração, respeitando os requisitos anteriormente citados, baseia-se na natureza híbrida dos hipertextos, combinando um acesso transparente aos dados a técnicas de organização de informações poderosas e flexíveis

A essência dos hipertextos combina os seguintes mecanismos: método de acesso às informações, seguindo as ligações para chegar aos nós; um esquema de representação para estruturar a informação; uma interface gráfica, permitindo uma interação com o usuário baseada na manipulação direta de ligações. O sistema DHT explora estes mecanismos para alcançar a integração desejada e preservando, ao mesmo tempo, a autonomia dos repositórios.

A arquitetura DHT é baseada no modelo cliente/servidor e inclui quatro componentes: um modelo de dados hipertextual; um protocolo de comunicação; servidores e aplicações para os clientes. O sistema requer um único *gateway* para cada tipo de repositório para converter as suas informações em um formato comum. Um banco de dados contém o meta-conhecimento sobre o conteúdo de cada repositório e, subseqüentemente, é mantido como informação no próprio repositório.

Para demonstrar como repositórios heterogêneos podem ser integrados através da arquitetura DHT, vários servidores já foram construídos, os quais permitem o acesso a uma variedade de tipos de repositórios, desde tabelas *hash*, sistema para gerenciamento de banco de dados relacional Ingres e banco de dados orientados a objetos. Os servidores de ligação e anotação permitem que os usuários associem comentários aos objetos do sistema e criem redes pessoais.

Os usuários do sistema DHT se envolvem em três atividades básicas: navegação entre os relacionamentos; visualização e edição de nós; e criação de novos nós, tarefas

realizadas através de um editor/navegador orientado a janelas. Os servidores apóiam estas atividades ao exportar objetos, oferecer descrição do tipo dos objetos e criar novos objetos através dos gerenciadores de armazenamento em resposta à consulta dos usuários.

A navegação pelos nós envolve as seguintes atividades: localizar os servidores apropriados; gerar uma lista de ligações associadas ao nó; selecionar as ligações presentes na lista de acordo com os atributos especificados pelo usuário; apresentar o resultado ao usuário. Os nós são apresentados, juntamente com o seu tipo, e o cliente envia uma mensagem ao servidor, após a edição do nó, para armazenar o seu novo conteúdo. Os servidores têm controle absoluto sob o processo de criação de novos objetos para garantir a consistência do repositório. Os usuários devem especificar o tipo de objeto a ser criado.

III.4.2. O Sistema HyperCASE

Cybulski e Reed desenvolveram um ambiente CASE baseado em hipertextos, denominado HyperCASE, para integrar ferramentas de apoio ao desenvolvimento de software (CYBULSKI e REED, 1992). Através do hipertexto, as informações são reunidas e conectadas permitindo uma navegação não seqüencial entre os diversos documentos do software. O projeto foi desenvolvido como parte do programa AAITP (*Amdahl Australian Intelligent Tools Program*).

O seu objetivo – assim como o de todas as ferramentas CASE – é oferecer uma plataforma de desenvolvimento de software integrada, poderosa e amigável que aumente a produtividade dos desenvolvedores. O HyperCASE apóia engenheiros de software na gerência do projeto, análise, projeto e implementação de aplicações computacionais. O sistema oferece um ambiente de desenvolvimento de software gráfico, integrado e configurável formado por ferramentas, fracamente acopladas, para apresentação de textos e diagramas referentes à construção da aplicação.

As ferramentas são reunidas através de uma interface hipertextual integrada a um repositório de documentos baseado no conhecimento. O sistema também inclui mecanismos para processar uma linguagem natural durante a especificação dos

requisitos do software. HyperCASE é composto por três subsistemas: HyperEdit, a interface gráfica implementada em X Windows; HyperBase, a base de conhecimento implementada em Prolog; e HyperDict, o dicionário de dados, projetado como um armazenamento de dados Prolog em Ingres. Esses subsistemas também funcionam individualmente, com o intuito de facilitar o desenvolvimento de novas ferramentas a serem integradas ao sistema.

O subsistema HyperEdit, composto por um gerenciador de interface, um sistema de autoria e um gerenciador de eventos, permite a construção, edição e apresentação de uma variedade de documentos, incluindo requisitos, diagramas de fluxo de dados, diagramas de transição de dados, código-fonte e plano de testes. Todas as ações realizadas no HyperEdit se refletem e estendem o conteúdo do HyperBase, que reúne os documentos do software. O subsistema HyperBase é composto por mecanismos e estruturas para organizar o sistema hipertextual genérico e por ferramentas CASE que garantem a consistência dos documentos e a reutilização dos componentes do projeto. As informações presentes no HyperDict permitem que as regras presentes no HyperBase verifiquem a integridade do banco de dados.

Os usuários podem relacionar artefatos do software às decisões do projeto. O sistema oferece mecanismos para recuperação de informações, procurando melhorar os métodos tradicionais para busca de dados em hipertextos. O sistema de navegação permite seguir o desenvolvimento de uma aplicação tanto em ordem cronológica quanto logicamente. Outros mecanismos que permitem um acesso rápido a documentação do software: recuperação de documentos por classificação, conteúdo ou relacionamento com outros documentos; navegação de documentos com *zoom*; navegação entre documentos pertencentes a uma lista histórica, mapas e marcadores.

HyperCASE combina processamento em linguagem natural, registro de decisões do programa e rastreamento do projeto para permitir uma reutilização do projeto e melhorar o controle sobre o mesmo. O sistema não apóia métodos específicos ou geradores automáticos de código. HyperCASE procura minimizar os esforços de manutenção e facilitar a manipulação de uma grande quantidade de documentos.

III.4.3. O Projeto HyperCode

O projeto HyperCode foi desenvolvido no Laboratório de Inteligência Artificial do MIT (BROWN *et al.*, 1994, KAISER *et al.*, 1997). HyperCode é uma ferramenta para compreensão do código, através de uma representação hipertextual de programas, onde as funções, tipos de dados, variáveis e macros são conectados com as suas definições. De maneira similar, as definições são conectadas aos dados. HyperCode é uma ferramenta para análise de programas, mas também apóia o compartilhamento e desenvolvimento de código.

O projeto apresenta um modelo de computação com as seguintes características: desenvolvimento cooperativo do código, rápida extensão e entrega de novos sistemas, organização do conhecimento compartilhado em software, disseminação de experiências e distribuição de código reutilizável.

Os servidores de código na Web atuam como poderosas fontes de informação e disseminação que mantém um banco de dados ativo, contendo pedaços de código compartilhados (PERPICH *et al.*, 1997). Os clientes podem consultar informações sobre aplicações existentes, facilitando a construção de futuras versões. As informações sobre a construção de programas podem ser automaticamente acessadas através de sistemas cliente/servidor em qualquer lugar do mundo.

Ao utilizar a ferramenta, um usuário pode determinar todas as dependências de uma dada função. Começando pela função principal, um usuário prossegue explorando a estrutura funcional do programa inteiro. HyperCode oferece ligações hipertextuais entre a declaração das variáveis e as definições da estrutura de dados. HyperCode permite que um usuário explore tanto as estruturas de dados quanto as funções de um programa. A ferramenta também cria ligações da definição das variáveis à lista de rastreamento, onde cada elemento da lista é relacionado à parte do código onde a variável é utilizada.

O banco de dados do HyperCode armazena informações sobre o código, suas relações e dados adicionais sobre o seu conteúdo do próprio banco. Desta maneira, o servidor WWW produz diferentes visões hipertextuais do conjunto de dados, tais como: documentação, comentários dos usuários, resultados e rastreamento das

execuções. Tais visões permitem que o usuário selecione o tipo de informação a ser apresentada.

As idéias básicas da ferramenta são aplicadas à linguagem de programação C, embora possam ser estendidas para qualquer linguagem de programação. As ligações hipertextuais são criadas durante a compilação. Os arquivos HTML podem ser acessados através de navegadores WWW, oferecendo um mecanismo poderoso para entender estruturas e fluxo de controle de programas complexos. O código e as ligações são armazenadas em um banco de dados. Informações adicionais sobre o código podem ser integradas através de ligações a documentos externos, comentários dos usuários e resultados da execução.

III.4.4. A Ferramenta LIGHT

A ferramenta LIGHT (*LIfe cycle Global HyperText*) (AIMAR *et al.*, 1995, AIMAR *et al.*, 1997, LIGHT, 1999) procura apoiar a resolução dos problemas enfrentados ao rastrear e entender grandes quantidades de tipos de documentos, desde o código até requisitos, análises, diagramas do projeto, manual do usuário, etc. Esta tarefa não é simples e é muito demorada devido ao número de documentos, extensão e relações entre eles. Em ambientes de desenvolvimento de software, onde as equipes estão distribuídas, e os documentos do software são produzidos em vários lugares, os problemas podem ser ainda maiores.

LIGHT foi desenvolvida no CERN e utiliza a tecnologia WWW para conectar os documentos do software. A seleção de uma ligação hipertextual, que representa uma chamada a rotina em um programa Fortran, apresenta a documentação da rotina. Acionando um elemento de dados o usuário tem acesso à definição de dados correspondente.

As três propriedades de uma documentação de software são: o grande volume de informações; a alta conexão entre todos os documentos gerados no ciclo de vida do software; e a distribuição dos dados. LIGHT aplicou este conceito aos experimentos do CERN, onde numerosas equipes de trabalho desenvolvem, utilizam, mantém e modificam código e respectiva documentação em diferentes localizações.

A idéia central do projeto é permitir que todos os documentos relativos a um sistema computacional estejam disponíveis e conectados. Os requisitos essenciais da ferramenta LIGHT são: extensão, a habilidade de facilmente adicionar novos programas conversores HTML; conexão automática, a habilidade de conectar automaticamente vários documentos através de ligações hipertextuais; configuração, a habilidade de configurar as conexões, dependendo do tipo de aplicação; atualização incremental, a habilidade de reconverter somente os documentos que foram alterados, sem produzir o hipertexto final de todos os documentos.

LIGHT permite a fácil integração de conversores HTML mantendo-os independentes, ou seja, os programas não produzem referências externas. Para oferecer uma conexão automática de documentos, a ferramenta segue uma conversão em duas etapas. Na primeira etapa, os documentos são analisados por *parsers* especializados, cada um por formato de documento, produzindo um banco de dados que contém a descrição de todas as possíveis ligações hipertextuais. Na segunda etapa, o programa lê o conteúdo do banco de dados e converte os documentos, definindo as ligações hipertextuais através de uma comparação entre os nós e as suas possíveis destinações, de acordo com o especificado no banco de dados. A conversão é orientada por um programa de configuração, que contém regras, as quais podem ser modificadas para configurar a conexão entre documentos.

Para reconverter somente os documentos que foram alterados, a ferramenta associa cada possível alvo hipertextual a um documento, o qual, por sua vez, é associado a um tipo (por exemplo, *rotina*) e a um nome. A combinação entre um documento, o seu tipo e nome identifica o alvo de maneira singular, o que é menos sujeito a mudanças, que um URL.

LIGHT ainda não é uma ferramenta de proposta geral porque para cada aplicação, novos conversores devem ser implementados e as regras de conversão, modificadas. A ferramenta foi utilizada para converter e publicar, no WWW, a documentação do programa de reconstrução JULIA do Experimento ALEPH do CERN. LIGHT está sendo modificada para apoiar a documentação das aplicações orientadas a objetos do Experimento ATLAS. Atualmente, a ferramenta incorpora módulos

para converter e conectar os seguintes formatos no Web: código em Fortran e C++, diagramas gerados através das ferramentas StP e Rational Rose, conversores Web de processadores e formatadores de texto (por exemplo, LaTeX, FrameMaker, outros), documentos gerados pelos *templates* do CERN e textos ou gráficos genéricos (LIGHT, 1999).

III.4.5. O Projeto ConceptTracker

O objetivo do projeto ConceptTracker, do Laboratório para Pesquisa em Software (LSR, 1999) (Departamento de Ciência da Computação da Universidade do Texas), nos Estados Unidos, é construir ambientes de desenvolvimento de software modernos para permitir que os conceitos de uma aplicação de software possam ser rastreados desde a sua concepção inicial até a implementação final (LEASE, 1998). ConceptTracker foi projetado para oferecer um protocolo aberto e extensível para armazenamento, indexação, procura e acesso aos artefatos criados durante o ciclo de vida de uma aplicação computacional. ConceptTracker permite a construção de ambientes de desenvolvimento de software modernos, onde os conceitos de uma aplicação possam ser rastreados desde a sua definição inicial até a sua implementação final. ConceptTracker utiliza o protocolo HTTP e a tecnologia WAIS.

ConceptTracker gerencia e conecta diferentes informações, relativas às metodologias tradicionais para desenvolvimento de software e ao domínio da aplicação. Outro objetivo deste projeto é demonstrar que o trabalho realizado por equipes cooperativas pode ser melhorado pela Internet, teleconferência e ferramentas computacionais.

O sistema oferece um protocolo para criação dinâmica de ligações hipertextuais intencionais entre documentos. Ligações hipertextuais intencionais são relações que não são armazenadas como parte do documento, mas que correspondem ao resultado de funções aplicadas à estrutura e ao conteúdo dos documentos. Estas ligações são geradas quando o usuário faz uma consulta a uma ligação hipertextual, como por exemplo: acesso a uma palavra em um documento e a sua definição no dicionário. As relações criadas conectam os conceitos presentes em um documento com outros documentos, que referenciam este conceito.

A maioria dos documentos mantidos nos bancos de dados hipertextuais, utilizados pelo protocolo do ConceptTracker, são estritamente relacionados ao desenvolvimento do software, ou seja, código-fonte e diagrama de fluxo de dados, embora permita manipular outros tipos de documentos, como o manual do usuário.

III.4.6. O Projeto WISE

O projeto WISE (*Web Integrated Software Environment*), coordenado pelo Laboratório de Pesquisa em Software do Departamento IV&V (*Independent Verification and Validation*) da Nasa, é o primeiro sistema para gerência de projetos de software baseado no WWW (CALLAHAN e RAMAKRISHNAN, 1998, CALLAHAN *et al.*, 1998, WISE, 1998). WISE é implementado como um serviço WWW que apóia a gerência e a medição de projetos de desenvolvimento de software baseados em análise dinâmica.

O WISE é um sistema computacional que manipula pedidos de mudanças, relatórios sobre os problema, descrição das atividades realizadas, oferendo uma estrutura para rastrear um projeto de software. O WISE é um sistema de gerência de mudanças que auxilia o rastreamento de problemas, mantém um registro das mudanças, problemas resolvidos, etc. O sistema apóia a coordenação do trabalho, distribuição de recursos, registro de informações e mantém o fluxo de dados entre as equipes de projeto. Inicialmente, o projeto começou como uma ferramenta para resgistrar os itens a serem realizados pelos programadores, evoluindo para um sistema que além de rastreiar problemas, possibilita que cada membro do grupo tenha acesso às questões pendentes.

Através de uma plataforma cliente-servidor, o usuário pode acessar um determinado tipo de informação, definir prioridades e visualizar os problemas existentes e os já resolvidos. WISE permite que programadores e gerentes acessem os relatórios sobre um determinado assunto e visualizem as métricas de projeto. Os formulários, relatórios, papéis, responsabilidades e métricas do WISE podem ser configurados para uma organização. WISE oferece um método para coordenar as atividades do projeto e permite que as equipes de desenvolvimento de software visualizem seu

progresso e desempenho. Portanto, o sistema permite:

- superar a barreira geográfica, pois um usuário pode acessar qualquer informação do projeto através do WWW e as equipes de trabalho podem acessar a ferramenta em qualquer lugar.
- superar a barreira da comunicação, incentivando a resolução de problemas em projetos cooperativos.
- resolver problemas, pois os gerentes de projeto podem rastrear, de maneira eficiente, o progresso do trabalho utilizando métricas incorporadas à ferramenta.
- permitir a consistência de informações, pois as mudanças efetuadas em um projeto são registradas no banco de dados, acessado pela equipe de trabalho.

Os mecanismos do WISE são: uma linguagem de programação para configuração do próprio sistema, um pacote de métricas que permite a integração de novas métricas e um banco de dados. A base do WISE corresponde a um banco de dados Oracle. Cada vez que o usuário seleciona uma opção, uma consulta é enviada à base do sistema através de um *gateway* que permite a conexão com o banco de dados. Os mecanismos do WISE relacionados à coleta e análise de dados foram baseados no projeto WebMe da Universidade de Maryland, Estados Unidos, apoiado pela NASA (TESORIERO e ZELKOWITZ, 1998).

III.4.7. O Projeto WHERE

O projeto WHERE (*A Web-based Hypertext Environment for Requirements Evolution*) está sendo realizado pela NASA e tem como objetivo apoiar o processo cooperativo da especificação de requisitos, através de ferramentas para gerenciar alterações incrementais (BARRY, 1994, EASTERBROOK e CALLAHAN, 1996, WHERE, 1999). Em grandes projetos de software, a gerência de especificações é muito complexa, uma vez que pequenas modificações em uma parte da especificação pode causar grandes impactos na documentação final. Ferramentas de rastreamento de

requisitos são de grande utilidade por registrar as ligações entre os requisitos em diferentes níveis de abstração. Entretanto, as ferramentas de rastreamento existentes apenas registram as ligações e não outras informações sobre o relacionamento expresso pelas ligações. Tais ferramentas não capturam o conhecimento sobre o método e as notações utilizadas e, portanto, não oferecem qualquer apoio ao desenvolvimento e à evolução de especificações.

WHERE é um ambiente que pode ser configurado para qualquer método e notação e utiliza a noção de *viewpoint* (descrito na seção II.7.) como método de estruturação. WHERE utiliza o WWW para apoiar a definição das relações entre os *viewpoints* na Internet.

O sistema, implementado em HTML e Java *applets*, integra um banco de dados para armazenar informações. As principais funcionalidades são oferecidas por três Java *applets*: o editor *viewpoint*, um editor de especificações configurável; o revisor de *viewpoints*, que permite a um usuário navegar e selecionar *viewpoints* criados por outras pessoas; o verificador de consistência, que identifica relacionamentos incorretos entre *viewpoints*. O editor Java rastreia um documento e o divide em *viewpoints*, de acordo com a sintaxe pré-determinada. Depois, o sistema permite que o usuário faça comentários aos *viewpoints*, associando-as a textos ou outras ligações. O projeto WISE e o método de estruturação *viewpoint* contribuíram para o projeto WHERE.

III.5. Considerações Finais

Este capítulo apresentou os recursos da Internet, utilizados neste trabalho, para o desenvolvimento de aplicações computacionais. A extensa utilização da Internet foi impulsionada pelo WWW, que popularizou o paradigma da hipermídia para uma audiência mundial. Os diversos mecanismos do WWW permitem um rápido acesso às informações localizadas na Internet, oferecendo uma interface altamente interativa.

As técnicas de hipermídia associadas à rede de computadores oferecem diversos mecanismos de apoio ao desenvolvimento de software, com o objetivo de aprimorar

a produtividade das equipes. A característica natural das ligações hipertextuais permite definir referências aos materiais resultantes de processo de desenvolvimento de software (KAISER *et al.*, 1997). Portanto, a tecnologia WWW oferece uma infra-estrutura oferece uma grande vantagem em ambientes de desenvolvimento de software (GUNTER *et al.*, 1996, OSTERWEIL *et al.*, 1996).

A tecnologia WWW possibilitou o surgimento de ferramentas de apoio à construção assíncrona de software realizada por equipes geograficamente distribuídas (HILBERT e REDMILES, 1998b). Os projetos apresentados permitem identificar a evolução das pesquisas nesta área. Os sistemas HyperText e HyperCASE apóiam a criação e o acesso aos componentes do software e suas respectivas relações. O objetivo de tais sistemas é permitir uma definição dinâmica das relações entres os diferentes componentes de uma aplicação computacional. A sua utilização é restrita a uma rede local de computadores. Os primeiros projetos a utilizar o WWW se concentraram no código e nos documentos do software, como descrito nas ferramentas HyperCode e Light. Embora restritas ao código e à documentação, tais projetos se preocuparam em divulgar, através da Internet, as informações do software a uma audiência globalmente dispersa. Os projetos posteriores incorporaram outros aspectos do processo de software. O sistema ConceptTracker armazena, indexa, procura e acessa os artefatos produzidos ao longo do processo de desenvolvimento de software. O projeto WISE apóia a gerência e a medição de projetos de software baseados em análise dinâmica. O projeto WHERE tem como objetivo apoiar o processo de especificação de requisitos.

Tais projetos demonstram a capacidade do Web em estruturar as informações do software e gerenciar suas mudanças dinâmicas, o que é extremamente útil em ambientes geograficamente dispersos. Várias pesquisas estão sendo realizadas para melhorar a funcionalidade do WWW, incorporanto, sistematicamente, os novos mecanismos aos navegadores (sistemas de comunicação, manipulação de imagens, nós dinâmicos, programas executáveis, etc). Entretanto, outras extensões são necessárias para capturar as informações do software, integrá-las e oferecer um apoio completo ao seu processo de construção.

IV. Processo Padrão para Equipes Geograficamente Dispersas

Neste capítulo, descrevem-se as características e identificam-se os problemas encontrados no desenvolvimento distribuído de software. Com base nesta identificação, determinam-se os requisitos a serem respeitados na definição de um processo de software para equipes geograficamente dispersas. Então, apresenta-se o processo padrão que procura atender o escopo desta pesquisa.

IV.1. Características dos Projetos Distribuídos de Software

O World-Wide Web conecta inúmeras informações de forma eficiente e amigável, localmente, através das intranets instaladas nas empresas, e globalmente, através da Internet. O compartilhamento de dados, independentemente da localização física dos profissionais, permitiu o surgimento de empresas de desenvolvimento de software virtuais (FIELDING *et al.*, 1998). Através do Web, os objetivos e as tarefas dos grupos de trabalho são identificados, mecanismos para armazenar e acessar informações são disponibilizados e a coordenação é realizada através de mensagens eletrônicas.

Os projetos de software que envolvem profissionais em lugares fisicamente distribuídos constituem um desafio tanto para gerentes quanto para desenvolvedores (AOYAMA, 1998b). Os problemas enfrentados em tais projetos envolvem diferentes aspectos da engenharia de software bem como questões técnicas relacionadas à Internet. Todas estas questões incentivaram pesquisadores e empresas de software a discutirem possíveis soluções em eventos internacionais, como por exemplo, o Workshop de Engenharia de Software na Internet, o primeiro dos quais foi realizado em 1998 (WSEI, 1998).

O desenvolvimento de software corresponde a um trabalho cooperativo, que utiliza intensivamente os meios de comunicação. As atividades de software podem ser classificadas como *atividades assíncronas*, quando os desenvolvedores trabalham em

diferentes horários com a mesma informação ou trocam mensagens e dados para coordenar suas tarefas; ou *atividades síncronas*, quando algumas tarefas, tais como reuniões, requerem discussões, em uma determinada hora, entre profissionais que podem ser responsáveis por diferentes aspectos do software. Como exemplo, os desenvolvedores compartilham, de forma assíncrona, partes do código, enquanto devem, de forma síncrona, resolver seus problemas.

A produção de sistemas envolve a participação de diversos e, muitas vezes, numerosos grupos de profissionais. A interação e a troca de informações entre os membros dos grupos pode determinar o sucesso ou fracasso de um projeto (ARNOLD, 1995). Devido aos recentes desenvolvimentos da tecnologia, as equipes de trabalho são formadas por profissionais provenientes de diferentes domínios do conhecimento. Os grupos multi-disciplinares devem compartilhar dados e informações que nem sempre são da sua área de trabalho.

A coordenação do desenvolvimento de software deve determinar a seqüência de atividades a serem realizadas, garantir que as tarefas foram corretamente executadas e verificar se outras tarefas, além das planejadas, devem ser feitas. Para uma coordenação efetiva, uma equipe de software deve rapidamente tornar disponíveis os resultados obtidos para que seja possível identificar pendências ou mudanças nos trabalhos realizados por outras equipes (BRIAND *et al.*, 1995). É responsabilidade da coordenação indicar, através do acompanhamento das atividades, as decisões críticas tomadas ao longo da construção de um sistema. Uma equipe deve ser seguida por um coordenador local, enquanto que a gerência do projeto deve supervisionar todo o processo de software.

Por trabalhar em diferentes localidades, as equipes de trabalho, muitas vezes, utilizam diferentes equipamentos, métodos, técnicas, ferramentas e padrões. O conhecimento e a experiência dos profissionais em computação e engenharia de software bem como o número de integrantes em cada equipe também pode variar (AOYAMA, 1998a). A capacidade econômica de determinados grupos para possíveis investimentos pode ser limitada. Devido ao fato que os recursos computacionais (hardware e software), humanos e econômicos são, geralmente, heterogêneos, e que as atividades

são diversas, o tempo de desenvolvimento difere de uma equipe a outra. Como a construção de um sistema é realizado por diferentes agentes, é necessário definir a capacidade de cada membro, alocar tarefas de forma apropriada e sincronizar as tarefas.

A engenharia de software envolve diferentes tipos de dados, tais como código, especificações, diagramas e resultados de testes. Durante as fases de desenvolvimento e manutenção, vários documentos, produtos intermediários e diferentes versões do software são criados e, posteriormente, integrados. Em muitos casos, uma única documentação é produzida por diversos profissionais e, portanto, as atividades de múltipla autoria devem ser gerenciadas de forma adequada (MAURER e DELLEN, 1998, WHITEHEAD e WIGGINS, 1998).

Durante o desenvolvimento de software, as tarefas realizadas por uma determinada equipe podem influenciar as tarefas de outras equipes. Portanto, é importante relatar tais dependências aos membros envolvidos e definir as responsabilidade de cada um. A resolução de problemas deve ser coordenada de forma efetiva pois, muitas vezes, é difícil encontrar a causa das falhas e investigações cooperativas são necessárias (KUDO *et al.*, 1998).

IV.1.1. Problemas no desenvolvimento distribuído de software

Considerando as características descritas acima, pode-se identificar os problemas presentes no desenvolvimento de software realizado por equipes geograficamente distribuídas. Algumas das dificuldades identificadas também estão presentes em outros contextos de construção de software. Entretanto, são agravadas quando os grupos de trabalho não se encontram em um mesmo lugar. Os problemas que se referem a determinados aspectos da construção do software são agrupados em classes. As classes de problemas presentes no desenvolvimento distribuído de software são:

- **Coordenação de equipes:** refere-se às relações entre os grupos dispersos e à sua supervisão considerando que as equipes podem ser multi-disciplinares e heterogêneas. Envolve a determinação da capacidade de cada equipe e a delegação de atividades, de acordo com suas características e restrições.

- **Coordenação das atividades:** refere-se ao controle do fluxo de trabalho, identificando atividades concorrentes, tarefas que exigem cooperação entre as equipes e como resolver os problemas de sincronismo.

- **Controle de artefatos:** envolve os problemas de integração de componentes, autoria distribuída, visibilidade dos resultados, notificação de mudanças e gerência de configuração.

- **Apoio à comunicação:** refere-se a todos os aspectos do desenvolvimento de software que requerem a troca de informações entre as equipes, de forma a permitir uma gerência efetiva do processo de software. Aplica-se à resolução de problemas, notificação de decisões, registro, acesso e compartilhamento de dados do software e publicação de informações que auxiliam o acompanhamento de atividades.

Tais problemas podem ser reduzidos ou, até mesmo, eliminados seguindo um processo de software bem definido e adequado às especificidades dos ambientes distribuídos.

IV.2. Requisitos de processos de software para equipes geograficamente dispersas

A qualidade dos produtos de software desenvolvidos por equipes geograficamente dispersas é fortemente dependente de uma comunicação efetiva, da coordenação dos grupos distribuídos, do rastreamento sistemático das atividades e artefatos, e da disponibilidade das informações do processo de software. Os requisitos de processos de software para equipes geograficamente dispersas são apresentados e discutidos a seguir (BANDINELLI *et al.*, 1996, KAISER *et al.*, 1997, PFLEEGER *et al.*, 1997, CALLAHAN *et al.*, 1998, EMAM *et al.*, 1998):

***Requisito 1*: Determinar a capacidade das equipes**

O processo deve incorporar atividades para identificar o conhecimento, experiência e

os recursos humanos, tecnológicos e econômicos de cada equipe. A determinação da capacidade dos grupos de trabalho deve ser executada antes de se iniciar um projeto de software, permitindo definir os papéis de cada membro da equipe e atribuir tarefas, de forma adequada, às diferentes equipes. É responsabilidade da organização atualizar periodicamente o cadastro dos grupos.

***Requisito 2*: Apoiar a coordenação distribuída**

Por trabalhar em diferentes localidades, cada equipe deve ser supervisionada, localmente, por um coordenador, o qual responde diretamente à gerência do projeto. É responsabilidade do coordenador determinar os requisitos das atividades a serem realizadas, identificar as responsabilidades de cada profissional, planejar o trabalho, acompanhar a realização de tarefas, garantir que as informações sobre o andamento do trabalho estejam disponíveis e divulgar eventuais mudanças, decisões e problemas. Tal requisito exige a rastreabilidade do processo e o acompanhamento das atividades técnicas pela coordenação local.

***Requisito 3*: Apoiar a gerência distribuída do projeto**

O processo deve incorporar atividades gerenciais para determinar os requisitos necessários à execução das atividades de cada equipe, verificar a sua adequação e integrar os resultados. É responsabilidade da gerência de uma organização monitorar a execução de todo o processo de software, através da supervisão das atividades relatadas pelos coordenadores locais. A gerência deve identificar tarefas concorrentes, determinar trabalhos cooperativos entre as diversas equipes e definir os pontos de controle do projeto.

***Requisito 4*: Apoiar o controle de artefatos**

O processo deve incorporar atividades para acompanhar o desenvolvimento de artefatos, garantir que a sua produção siga os procedimentos pré-definidos pela gerência do projeto e administrar a integração com outros artefatos produzidos pelas diversas equipes. O controle de artefatos requer o planejamento, monitoração e notificação

de resultados, mudanças, inconsistências e dependências.

***Requisito 5*: Apoiar a comunicação entre as equipes**

É responsabilidade da organização identificar os recursos disponíveis para transmitir e receber mensagens e implantar mecanismos tradicionais ou eletrônicos para permitir a comunicação entre as equipes de software. A comunicação refere-se a todos os aspectos do desenvolvimento de software que requerem troca de informações entre as equipes para a resolução de problemas, notificação de decisões e acompanhamento efetivo das atividades e da produção de artefatos (BRUEGGE e DUTOIT, 1997).

***Requisito 6*: Apoiar a publicação e o compartilhamento de informações**

O processo deve incorporar atividades para determinar que informações devem ser publicadas pelas equipes ao longo do ciclo de vida do software. É responsabilidade da organização instalar repositórios e mecanismos de acesso a todas as informações do software, tais como idéias, decisões, restrições e o estado do trabalho em execução. O processo deve incorporar procedimentos para informar mudanças realizadas por uma determinada equipe e que possam influenciar o trabalho de outras equipes.

***Requisito 7*: Apoiar a resolução de problemas**

A análise e a resolução de problemas em ambientes distribuídos requer que o processo de software incorpore atividades para notificar a existência de um problema, divulgar a sua resolução e identificar as possíveis implicações no trabalho das outras equipes. É responsabilidade da organização definir planos a serem executados quando um problema é detectado e elaborar relatórios sobre sua análise e acompanhamento do processo de resolução.

***Requisito 8*: Incorporar um repositório de terminologias**

É responsabilidade da organização identificar, padronizar e documentar um glossário sobre o domínio da aplicação do projeto, garantindo a compreensão única dos termos pelas várias equipes. O processo deve incorporar atividades para criar, manter

e publicar um repositório de terminologias.

***Requisito 9*: Apoiar a redefinição de atividades do software**

É responsabilidade da organização identificar, redefinir e controlar a execução das atividades do processo de software, considerando as características e restrições das equipes dispersas. A organização deve estabelecer os procedimentos e a infra-estrutura necessária para a realização de tarefas de forma distribuída, tais como gerência de configuração, especificação, projeto codificação e teste, autoria e documentação, validações, verificações e revisões.

IV.3. Processo de software padrão para equipes geograficamente dispersas

O processo padrão define uma estrutura única a ser seguida por todas as equipes distribuídas envolvidas em um projeto de software. O processo de software padrão definido para equipes geograficamente dispersas tem como objetivo atender os requisitos apresentados anteriormente. O processo cobre todo o ciclo de vida do software desde a concepção inicial, determinação de requisitos, construção, entrega, manutenção, até finalizar a sua utilização. A definição do processo teve como base a norma ISO/IEC 12207 (*Information technology - software life cicle processes*) (ISO/IEC 12207, 1995), a qual define as atividades do ciclo de vida do software através de cinco processos primários, oito processos de suporte e quatro processos organizacionais, estruturados da seguinte maneira:

- *Processos primários*, correspondem à aquisição, contratação, desenvolvimento, operação e manutenção do software.
- *Processos de apoio*, referem-se à documentação, gerência de configuração, controle da qualidade, verificação, validação, revisão e resolução de problemas.
- *Processos organizacionais*, correspondem à gerência, infra-estrutura, melhoria e treinamento.

Estes processos foram redefinidos e às três categorias, foram introduzidos outros processos e, aos processos, atividades específicas para o desenvolvimento distribuído de software para atender as necessidades de equipes de trabalho dispersas.

O processo proposto é destinado às organizações formadas por grupos geograficamente dispersos que desenvolvem software. Portanto, o processo de aquisição foi substituído por um processo de definição do problema e o processo de contratação, pelo processo de planejamento. O processo padrão para desenvolvimento distribuído de software também se baseou no SPICE (EMAM *et al.*, 1998) e em trabalhos anteriores realizados na COPPE (COPPE-SISTEMAS, 1998).

Respeitando os requisitos, anteriormente citados, para utilizar o processo padrão para desenvolvimento distribuído de software, é necessário definir um repositório com informações sobre as equipes de trabalho. Tal repositório contém avaliações das capacidades e características dos membros de cada grupo, resultantes de entrevistas, coleta de dados e observações de projetos anteriormente realizados. A organização também deve estabelecer procedimentos para a gerência do projeto, coordenação de atividades das equipes, controle de artefatos, comunicação entre os profissionais e instalar a infra-estrutura necessária à execução do processo de software.

A estrutura do processo proposto é apresentada na figura IV.1. Os processos primários definem as atividades relativas à construção do software. Os processos de apoio são executados ao longo da realização dos processos primários e são responsáveis pelo sucesso dos projetos e pela qualidade do software. Os processos organizacionais estabelecem a estrutura necessária para a realização das tarefas dos outros processos. Suas atividades são realizadas não necessariamente durante a realização de um projeto de software.

IV.3.1. Relacionamento entre os processos

Os processos se relacionam funcionalmente. Os engenheiros de software desenvolvem um sistema, cuja documentação e configuração são definidas e coordenadas pelos processos de apoio. Ao iniciar um projeto, os processos organizacionais determinam a infra-estrutura necessária à realização das atividades. A definição do

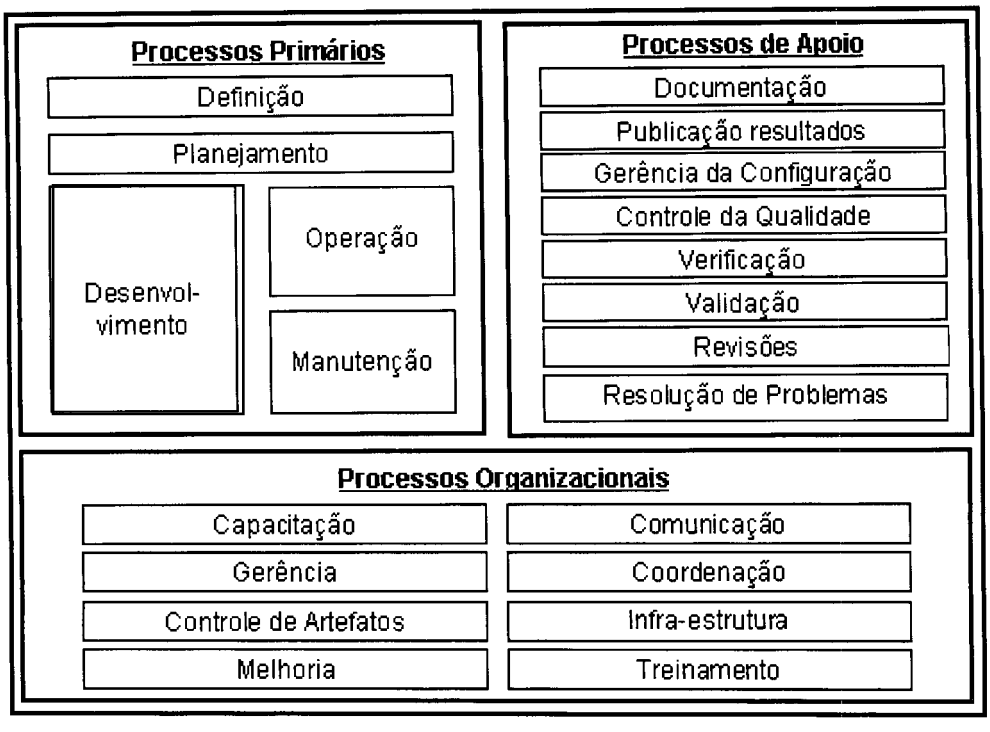


Figura IV.1: Estrutura do Processo Padrão para Equipes Distribuídas (adaptação da Norma ISO/IEC 12207).

processo é estática enquanto que a sua dinâmica somente é determinada quando o processo é aplicado a um projeto de software.

IV.3.2. Papéis no processo

Os profissionais que trabalham no processo de desenvolvimento de software podem ser associados aos seguintes papéis:

- *Gerente do Projeto*: responsável pelo projeto de software, monitoração do processo, distribuição de tarefas e integração dos resultados.
- *Coordenador de Grupo*: supervisiona um grupo de trabalho.
- *Administrador do sistema*: responsável pela gerência dos recursos computacionais utilizados no projeto.
- *Analista*: responsável pela especificação do sistema, determinação dos requisitos e validação do software.

- *Projetista*: responsável pelo projeto e implementação do software.
- *Programador*: realiza as atividades de codificação e testes do sistema.
- *Gerente de versões*: responsável pela manutenção das versões do sistema.
- *Operador*: responsável pela assistência aos usuários, durante a operação do sistema.
- *Mantenedor*: responsável pela manutenção (correção de falhas e implementação de melhorias) do software.

IV.3.3. Produtos do processo

Os produtos intermediários gerados ao longo do processo de desenvolvimento podem ser compostos por outros subprodutos e são classificados em tipos. A estruturação dos produtos permite ou deve permitir que cada um de seus componentes possa ser construído por um determinado grupo de trabalho, independente da sua localização. Posteriormente, os componentes são integrados. A determinação do tipo de um produto facilita a delegação de tarefas pois a sua construção será realizada pela equipe mais adequada, dependendo das suas capacidades e restrições.

IV.3.4. Atividades

As atividades são definidas segundo as categorias de processos às quais estão associadas. Portanto, as atividades podem ser classificadas em: **construtivas** (relacionadas à geração, publicação e manutenção de artefatos), **gerenciais** (responsáveis pela coordenação das equipes de trabalho, gerência do projeto de software e garantia da qualidade, gerência de configuração, medições e revisões) e **organizacionais** (realizadas pela organização para determinar a capacitação de equipes e oferecer o treinamento necessário, instalar a comunicação entre os profissionais, implementar a gerência e a infra-estrutura adequadas à realização das atividades de construção e coordenação, e permitir a melhoria contínua do próprio processo).

IV.3.5. Processos Primários

Os processos primários definem as atividades primárias do desenvolvimento de software, organizadas nos processos de definição, planejamento, desenvolvimento, operação e manutenção. As atividades que garantem a realização dos processos primários são determinadas pelos processos organizacionais. A qualidade dos artefatos produzidos é garantida pelos processos de apoio.

IV.3.5.1. Processo de Definição

O processo de definição inicia com a identificação das necessidades do cliente para desenvolver o software. Nesta fase, define-se o problema a ser solucionado e identificam-se os requisitos a serem respeitados. Faz-se uma análise dos recursos humanos, técnicos e econômicos necessários ao projeto. Determina-se o escopo do sistema a ser construído. Inicia-se a construção de um repositório de terminologias. As atividades deste processo são realizadas pela gerência do projeto, auxiliada pelos coordenadores das equipes. As tarefas são apoiadas por mecanismos de comunicação através da Internet, definidos pelos processos de comunicação e infra-estrutura.

A gerência das atividades do processo de definição e os recursos necessários para a realização das tarefas são determinados pelo processo de gerência e de infra-estrutura, respectivamente.

Atividades:

1. ***Definição do problema***
   Define-se o problema e a necessidade do cliente em adquirir, desenvolver ou aprimorar um sistema, produto ou serviço de software.

2. ***Definição inicial dos requisitos***
   Os requisitos do sistema são definidos, incluindo aspectos de segurança, confiabilidade e desempenho. Padrões e procedimentos a serem respeitados também devem ser identificados como requisitos.

3. ***Análise da viabilidade do projeto***
   Os recursos humanos, técnicos e econômicos necessários são identificados e faz-se uma análise de riscos, custos e benefícios de possíveis opções de desenvolvimento.

4. ***Determinação do escopo do sistema***
   Define-se uma lista de serviços que o sistema deverá realizar, definindo o seu

escopo. Nesta atividade, também, identificam-se produtos e serviços a serem adquiridos, ao invés de desenvolvidos, considerando a capacidade de cada equipe.

5. ***Construção do repositório de terminologias***
   Nesta atividade, inicia-se a construção de um repositório de terminologias relativo aos domínios de conhecimento do projeto em desenvolvimento. O repositório é instalado em um sistema de acesso via Internet, pelo qual todas as equipes possam ter acesso.

6. ***Documentação do processo de definição***
   Esta atividade tem como objetivo documentar os resultados das atividades anteriores, segundo o processo de documentação (processo de apoio). O documento resultante é publicado via Internet às equipes de software.

7. ***Revisão e aprovação***
   De acordo com os processos de apoio, os resultados das atividades do processo de definição, descritas na documentação correspondente, são revisados e aprovados pelos coordenadores locais e pela gerência do processo.

IV.3.5.2. Processo de Planejamento

No processo de planejamento, faz-se um plano dos processos de desenvolvimento, operação e manutenção e dos processos de apoio e organizacionais. Identificam-se os componentes do software, a partir de uma especificação inicial do sistema a ser desenvolvido. Determinam-se os padrões de documentos e as normas de codificação. As equipes de trabalho são organizadas, associando papéis a cada integrante dos grupos de trabalho. As atividades a serem desenvolvidas são alocadas a cada equipe de trabalho e, em conseqüência, a cada membro, de acordo com suas capacidades. Determinam-se as atividades cooperativas e concorrentes. Os pontos de controle, onde os trabalhos realizados pelas equipes são revisados e integrados, são determinados para permitir a coordenação do processo. Definem-se cronogramas locais e um cronograma geral. Faz-se uma análise dos riscos do projeto. O planejamento do projeto resultante é estruturado em dois níveis: um para cada equipe de software e um geral, que integra os resultados parciais.

O processo de planejamento é supervisionado segundo o processo de gerência. Os recursos e o treinamento necessários para a construção do sistema são definidos pelos processo de infra-estrutura e de treinamento (processos organizacionais),

respectivamente.

Atividades:

1. ***Especificação Inicial***
   Nesta atividade, o problema a ser tratado e os requisitos do sistema são analisados e faz-se uma modelagem inicial do sistema, utilizando um método de desenvolvimento, definindo o problema, o objetivo e as características do sistema proposto, usuários e as interfaces com outros sistemas. A especificação é realizada pela gerência do projeto, auxiliada pelos coordenadores das equipes, e apoiada por mecanismos de comunicação através da Internet.

2. ***Determinação dos componentes do sistema***
   A partir da especificação inicial, determinam-se os componentes do sistema, classificando-os de acordo com o conhecimento e recursos técnicos necessários para a sua construção. Esta tarefa é realizada em conjunto pela gerência do projeto e pelos coordenadores das equipes, apoiada por mecanismos de comunicação através da Internet.

3. ***Planejamento***
   Os requisitos definidos anteriormente determinam a estrutura de trabalho para gerenciar os processos de desenvolvimento, operação e manutenção e controlar a qualidade dos produtos. O planejamento é estruturado em dois níveis, sendo que o primeiro refere-se ao planejamento de cada equipe de trabalho e o segundo ao projeto de software, como um todo. A atividade de planejamento é composta pelas seguintes tarefas:

   - *Determinação de papéis*
     As responsabilidades de cada equipe e de seus membros são determinadas. Identifica-se a necessidade de treinamento (processos organizacionais). Tais informações são publicadas a todas as equipes através da Internet.
   - *Redefinição de atividades*
     Identificação das atividades que devem ser redefinidas para apoiar o desenvolvimento distribuído do software, tais como gerência de configuração, especificação, projeto codificação e teste, autoria e documentação, validações, verificações e revisões. A definição resultante das atividades é publicada a todas as equipes através da Internet.
   - *Alocação das atividades às equipes e a seus membros*
     De acordo com a capacidade de cada grupo de trabalho, identificada pelos processos organizacionais e, a partir da classificação dos componentes do sistema, as atividades dos processos de desenvolvimento, operação e manutenção são alocadas às equipes de software. Esta tarefa é realizada pela gerência do projeto e pelos coordenadores locais. As responsabilidades de cada profissional são publicadas através da Internet.
   - *Determinação da metodologia de trabalho*
     Definem-se os métodos e ferramentas a serem utilizados, a documentação

e os artefatos a serem gerados e determinam-se o ambiente de teste, bibliotecas, equipamentos, padrões e procedimentos para cada grupo de trabalho. Para as atividades de integração de resultados, definem-se a metodologia específica a estas tarefas.

- *Identificação de atividades concorrentes*
  As atividades construtivas que podem ser realizadas em paralelo por diferentes equipes são identificadas. Determinam-se que atividades dependem de artefatos produzidos por outras atividades. Define-se, então, as dependências entre as equipes e tais informações são publicadas na Internet.
- *Identificação de atividades cooperativas*
  Determinam-se quais são as atividades construtivas que envolvem a cooperação entre as equipes de desenvolvimento. Tais informações são publicadas na Internet.
- *Controle da qualidade*
  Define-se o planejamento das atividades relacionadas ao controle da qualidade do software ao longo dos processos primários. Determinam-se os resultados provenientes de outros processos de apoio (verificação, validação, revisões e resolução de problemas) que possam ser utilizados no controle da qualidade do software. Esta atividade é estruturada em dois níveis, sendo que um se refere às tarefas de controle da qualidade a serem realizadas por cada equipe e outro se refere ao controle da qualidade do projeto como um todo. Todas as atividades de controle da qualidade definidas são publicadas às equipes através da Internet.
- *Verificação e Validação*
  Determinam-se as atividades que verificam se os artefatos produzidos por atividades satisfazem os requisitos ou condições impostas por atividades anteriores. Identificam-se as atividades que validam se o software produzido atende às necessidades do usuário. Esta atividade é estruturada em dois níveis, sendo que um se refere às tarefas de verificação e validação a serem realizadas por cada equipe e outro se refere à verificação e validação do projeto como um todo. Todas as atividades de verificação e validação definidas são publicadas às equipes através da Internet.
- *Identificação de pontos de controle*
  Nesta tarefa, determinam-se as etapas do desenvolvimento onde os artefatos produzidos por cada equipe precisarão ser integrados e onde são realizadas as revisões (processos de apoio). Determinam-se os pontos de controle que auxiliam o acompanhamento das tarefas realizadas por cada equipe e de todo o projeto.
- *Determinação de cronogramas para cada equipe*
  Os cronogramas individuais de cada equipe de trabalho são definidos e publicados a todas as equipes através da Internet.

- *Determinação de um cronograma integrado*
  Um cronograma geral, que permite a coordenação dos trabalhos das diversas equipes, é definido pela gerência, a partir dos cronogramas de cada equipe. O resultado é publicado a todas as equipes através da Internet.
- *Identificação das medições*
  Os dados a serem coletados durante o projeto são identificados para permitirem medições e apoiar a melhoria do processo. As medições são definidas como locais (para cada equipe) e globais (para todo o projeto de software).
- *Análise de riscos*
  Faz-se uma análise das áreas do projeto que envolvem riscos potenciais referentes a questões técnicas, financeiras e de cronograma. Cada coordenador local identifica os riscos envolvidos na realização das tarefas atribuídas à sua equipe. A gerência do projeto reúne todos os riscos e determina as tarefas necessárias para evitar possíveis falhas no projeto.

4. ***Determinação de padrões de documentação***
   Os responsáveis pelos processos de apoio determinam os padrões de documentação a serem seguidos pelos processos primários e de apoio. Nesta fase determinam-se, também, as normas de programação. Os padrões e normas são únicos para todas as equipes e são publicados através da Internet.

5. ***Documentação do processo de planejamento***
   Esta atividade deve documentar os resultados das atividades anteriores, segundo o processo de documentação (processos de apoio). A documentação resultante é publicada a todas as equipes através da Internet.

6. ***Revisão e aprovação***
   De acordo com os processos de apoio, os resultados das atividades do processo de planejamento, descritas na documentação correspondente, são revisados e aprovados pelos coordenadores locais e pela gerência do processo.

IV.3.5.3. Processo de Desenvolvimento

Este processo determina as atividades dos desenvolvedores do software relativas à especificação, projeto, codificação, integração, testes, instalação e entrega, de acordo com o que foi determinado no processo de planejamento. Durante este processo, os grupos de trabalho devem periodicamente indicar o estado de seus trabalhos, publicar resultados, decisões tomadas e restrições encontradas (processos de comunicação e publicação de resultados). É necessário notificar mudanças e atualizar o repositório de terminologias. O processo de desenvolvimento é supervisionado,

localmente, pelo coordenador da equipe e, globalmente, pelo processo de gerência, coletando dados para a melhoria do próprio processo de desenvolvimento (processos organizacionais).

As atividades do processo de apoio são realizadas quando necessário e, portanto, em cada atividade, o desenvolvedor deve:

- Produzir a documentação do software, como definido pelo processo de documentação.
- Notificar alterações do projeto, através dos meios determinados previamente pelo processo de comunicação.
- Controlar as mudanças e administrar as versões do sistema, de acordo como o processo de gerência de configuração.
- Notificar os problemas encontrados e solucioná-los, de acordo com o processo de resolução de problemas.
- Realizar todas as atividades de apoio, de acordo com os processos de apoio.

Atividades:

1. ***Implantação do Processo***
   A infra-estrutura de trabalho necessária para a execução do processo é instalada e os mecanismos de comunicação são estabelecidos por cada equipe. Os métodos, ferramentas, documentos, artefatos, padrões e procedimentos, apropriados para cada equipe e definidos no processo de planejamento, são estabelecidos pela organização para realizar as atividades do processo de desenvolvimento.
2. ***Especificação***
   A especificação detalhada de cada componente do sistema é realizada de forma distribuída pelas equipes, baseada na especificação inicial, resultante do processo de definição. Esta atividade é realizada através das seguintes tarefas:
   - As funções, capacidades, estrutura de dados e interfaces entre os componentes são identificadas por cada equipe. As restrições de projeto, requisitos de qualidade e os requisitos para os processos de operação e manutenção são descritos.

- As especificações parciais, realizadas por cada grupo, são avaliadas pelos coordenadores locais.
- As especificações parciais são integradas.
- A especificação resultante é avaliada pela gerência do projeto, verificando a conformidade com os requisitos determinados previamente, a consistência com as necessidades identificadas no processo de definição e a possibilidade de implementar a solução de forma eficiente.
- Os resultados são publicados a todas as equipes através da Internet.

3. ***Projeto***
   Os componentes do sistema são projetados. Define-se a alternativa mais adequada para a implementação dos componentes, considerando os recursos de cada equipe. Pode-se fazer uma reavaliação da alocação de tarefas por equipes. Esta atividade é realizada através das tarefas:

   - Para cada componente, as equipes definem a sua arquitetura, identificando os itens de hardware e software. Verifica se todos os requisitos estão alocados nos itens presentes na arquitetura do componente.
   - Os projetos parciais, realizados por cada grupo, são avaliados pelos coordenadores locais.
   - Os projetos parciais são integrados.
   - O projeto final é avaliado pela gerência do projeto, fazendo uma validação do projeto integrado com a especificação.
   - Os resultados são publicados a todas as equipes através da Internet.

4. ***Implementação***
   Os componentes do sistema são codificados, de acordo com os padrões definidos no processo de planejamento. Esta atividade é realizada através das tarefas:

   - As equipes desenvolvem o código de cada componente (funções e estrutura de dados).
   - A documentação do código é elaborada e publicada aos grupos de trabalho.
   - Os procedimentos e os dados para teste são definidos.
   - Os artefatos resultantes são avaliados pelos coordenadores locais, garantindo que os requisitos foram satisfeitos.
   - Os resultados são publicados a todas as equipes através da Internet.

5. ***Testes individuais***
   O código referente a cada componente produzido é testado individualmente, de acordo com os procedimentos definidos anteriormente. Esta atividade é realizada nas seguintes etapas:

- As técnicas definidas no processo de planejamento são utilizadas para testar o componente e a interface com os outros componentes do sistema.
- Verifica-se se cada componente satisfaz os requisitos.
- Eventuais problemas são notificados e a sua resolução é acompanhada pelos coordenadores locais e pela gerência do projeto, de acordo com o processo de resolução de problemas.
- A documentação do software é atualizada, caso necessário.
- Os resultados são publicados a todas as equipes através da Internet.

6. ***Integração e testes do sistema***
   Os componentes do software são integrados e realiza-se o teste de todo o sistema. A integração e os testes são realizados por equipes definidas pelos coordenadores locais e pela gerência do projeto. Esta atividade é realizada através das tarefas:

   - As equipes elaboram um plano de integração, que define casos de teste (entradas e saídas), procedimentos, dados, responsabilidades e cronogramas.
   - Os componentes codificados são integrados, seguindo o plano.
   - Faz-se um teste de integração do software.
   - A documentação do software é atualizada, caso necessário.
   - Os resultados são publicados a todas as equipes através da Internet.

7. ***Validação do sistema***
   O sistema é validado, avaliando-se se os requisitos foram respeitados. As inconsistências encontradas são notificadas e a coordenação e a gerência do projeto determinam as mudanças a serem realizadas. Esta atividade é realizada através das tarefas:

   - O projeto, código, testes, resultado dos testes e documentação são avaliados.
   - Os resultados da validação são documentados, informando se os testes cobrem todos os requisitos e se o software atingiu os objetivos esperados.
   - Futuras melhorias do sistema são notificadas para futuras versões. A gerência do projeto, junto à coordenação das equipes, determina os aspectos do processo a serem aprimorados para cada grupo de trabalho, de acordo com o processo de melhoria.
   - Os resultados são publicados a todas as equipes através da Internet.

8. ***Instalação do sistema no ambiente operacional***
   O sistema é instalado no ambiente de operação por uma equipe definida pelos coordenadores locais e pela gerência do projeto. Esta atividade é realizada nas seguintes etapas:

- As equipes elaboram um plano de instalação, determinando os recursos e as informações necessárias para a implantação do software no ambiente de operação.
- O sistema é instalado, de acordo com o plano, e os dados necessários à execução do sistema são introduzidos.

9. ***Entrega do sistema***
   A documentação do usuário é entregue e realiza-se uma demonstração do sistema pela equipe definida pelos coordenadores locais e pela gerência do projeto. Os resultados da entrega do sistema são publicados às equipes através da Internet.

10. ***Homologação do sistema***
    Durante esta atividade, o sistema é submetido à avaliação final para homologação, por um grupo de usuários finais. A avaliação e os problemas encontrados são documentados e publicados a todas as equipes. Quando o sistema é aprovado e já não necessita modificações, os desenvolvedores oferecem um treinamento aos usuários.

11. ***Atualização do repositório de terminologias***
    Durante o processo de desenvolvimento, o repositório de terminologias deve ser atualizado periodicamente. Cabe lembrar que o sistema que implementa o repositório de terminologias deve permitir a sua atualização através de uma rede de computadores e oferecer mecanismos, tais como: controle de versões e gerência da concorrência.

12. ***Documentação do processo de desenvolvimento***
    Esta atividade deve documentar os resultados das atividades anteriores, segundo o processo de documentação (processos de apoio). A documentação resultante é publicada a todas as equipes através da Internet.

13. ***Revisão e aprovação***
    De acordo com os processos de apoio, os resultados das atividades do processo de desenvolvimento, descritos na documentação correspondente, são revisados e aprovados pelos coordenadores locais e pela gerência do processo.

IV.3.5.4. Processo de Operação

O processo de operação contém as atividades dos operadores, de acordo com o que foi determinado no processo de planejamento. A equipe de operadores é formada por integrantes de uma ou várias equipes, cuja escolha é determinada pela coordenação local e pelo gerente do projeto. O processo se refere à operação do software e suporte operacional aos usuários. O processo de operação é coordenado

pelo processo de gerência, seguindo o processos de infra-estrutura e coletando dados para a melhoria do próprio processo de desenvolvimento (processos organizacionais).

As atividades do processo de apoio são realizadas quando necessário e, portanto, em cada atividade, o operador deve:

- Produzir a documentação correspondente, como definido pelo processo de documentação.
- Publicar os defeitos encontrados e as modificações do software, através dos meios determinados previamente pelo processo de comunicação.
- Controlar as mudanças e administrar as versões do sistema, de acordo como o processo de gerência de configuração.
- Notificar os problemas encontrados e solucioná-los, de acordo com o processo de resolução de problemas.
- Realizar todas as atividades de apoio, de acordo com os processos de apoio.

Atividades:

1. ***Definição de procedimentos***
   Os procedimentos para registro, resolução e rastreamento de problemas são definidos.
2. ***Assistência ao usuário***
   O operador deve dar assistência aos usuários, documentando suas dúvidas e dificuldades. Tais informações são publicadas através da Internet. As equipes devem ser informadas da presença de uma nova dúvida ou problema e, caso seja responsável pela parte do software em questão, iniciar os procedimentos para solucionar a dificuldade.
3. ***Identificação de erros e melhorias***
   Ao detectar problemas ou possíveis melhorias no software, o operador deve reportá-los ao processo de manutenção e publicá-los através da Internet. As equipes devem ser informadas da presença de um novo erro e, caso seja responsável pela parte do software em questão, iniciar os procedimentos para consertar o erro.
4. ***Documentação do processo de operação***
   Esta atividade deve documentar os resultados das atividades anteriores, segundo o processo de documentação (processos de apoio). A documentação resultante é publicada a todas as equipes através da Internet.

5. ***Revisão e aprovação***
   De acordo com os processos de apoio, os resultados das atividades do processo de operação, descritos na documentação correspondente, são revisados e aprovados pelos coordenadores locais e pela gerência do processo.

IV.3.5.5. Processo de Manutenção

O processo de manutenção contém as atividades dos mantenedores, de acordo com o que foi determinado no processo de planejamento. A equipe de mantenedores é formada por integrantes de uma ou várias equipes, cuja escolha é determinada pela coordenação local e pelo gerente do projeto. A fase de manutenção consiste na correção de erros e implementação de melhorias no software. As mudanças realizadas devem ser documentadas e, então, determina-se que equipe irá realizá-las. Os documentos gerados durante a fase de desenvolvimento devem ser atualizados. O processo de manutenção é iniciado quando o software é modificado e termina quando este não é mais utilizado. As atividades seguem as mesmas atividades definidas no processo de desenvolvimento. O processo de manutenção é coordenado pelo processo de gerência, coletando dados para a melhoria do próprio processo de software (processos organizacionais).

As atividades dos processos de apoio são realizadas, quando necessário e, portanto, em cada atividade, o mantenedor deve:

- Produzir a documentação correspondente, como definido pelo processo de documentação.
- Notificar as mudanças realizadas no software, através dos meios determinados previamente pelo processo de comunicação.
- Controlar as mudanças e administrar as versões do sistema, de acordo como o processo de gerência de configuração.
- Notificar os problemas encontrados e solucioná-los, de acordo com o processo de resolução de problemas.
- Realizar todas as atividades de apoio, de acordo com os processos de apoio.

Atividades:

1. ***Definição de procedimentos***
   Os procedimentos para conduzir a manutenção do software são definidos pela gerência do projeto e pelos coordenadores locais. Os pedidos de modificação, as tarefas realizadas e os resultados obtidos são documentados. Como as equipes de trabalho se encontram em diferentes localidades, torna-se essencial oferecer um sistema de registro de pedidos de modificações na Internet, de fácil acesso por todas as equipes.

2. ***Identificação da equipe responsável***
   A gerência do projeto e os coordenadores locais determinam a(s) equipe(s) que farão a modificação do software, para cada requisição do usuário.

3. ***Análise do problema e modificação do software***
   A equipe de mantenedores analisa o problema ou a requisição de mudanças, identificando:

   - Tipo de trabalho: correção, melhoria, adaptação ou preveção.
   - Escopo: tamanho da mudança, custo e tempo.
   - Aspectos críticos: impacto no desempenho e segurança.

4. ***Implementação da modificação***
   O mantenedor deve analisar o pedido de mudança, verificando que componentes do software e documentação devem ser modificados. As atividades subseqüentes correspondem àquelas do processo de desenvolvimento. O pedido de mudança deve ser publicado a todas as equipes através de um sistema de controle de mudanças na Internet.

5. ***Avaliação das mudanças***
   Após a realização da mudança, a equipe de mantenedores deve verificar se a nova versão do software satisfaz a requisição de mudanças.

6. ***Manutenção do repositório de terminologias***
   O repositório de terminologias deve ser mantido através de um sistema na Internet.

7. ***Documentação do processo de manutenção***
   Esta atividade deve documentar os resultados das atividades anteriores, segundo o processo de documentação (processos de apoio). A documentação resultante é publicada a todas as equipes através da Internet.

8. ***Revisão e aprovação***
   De acordo com os processos de apoio, os resultados das atividades do processo de manutenção, descritos na documentação correspondente, são revisados e aprovados pelos coordenadores locais e pela gerência do processo.

IV.3.6. Processos de Apoio

Os processos de apoio definem as atividades de suporte aos outros processos do ciclo de vida do software, organizadas nos processos de documentação, publicação de resultados, gerência da configuração, controle da qualidade, verificação, validação, revisão e resolução de problemas. As atividades e tarefas dos processos de apoio são de responsabilidade da organização de software (processos organizacionais), a qual deve estabelecer os pré-requisitos necessários para a sua existência e garantir a sua funcionalidade.

IV.3.6.1. Processo de Documentação

O processo de documentação reúne as atividades de registro de informações produzidas ao longo do ciclo de vida do software. Este processo contém um conjunto de atividades que planejam, projetam, produzem e mantém os documentos necessários para todos os profissionais do software, tais como gerentes, engenheiros, programadores e usuários. Devido à distribuição das equipes de trabalho, o processo de documentação deve ser apoiado por um sistema de acesso via Internet, que integre mecanismos de controle de versões e de concorrência.

Atividades:

1. ***Implantação do Processo***
   Nesta atividade, define-se um plano que identifica todos os documentos a serem produzidos durante o ciclo de vida do software. Para cada documento, os seguintes itens devem ser identificados:

   - Título do documento
   - Objetivo
   - Audiência
   - Procedimentos e responsabilidades para a sua produção, revisão, modificação, aprovação, produção, armazenamento, manutenção e gerência de configuração.

2. ***Projeto***
   Cada documento identificado na atividade anterior é definido, seguindo os padrões de formatação. As fontes para obtenção de informações são determinadas e o processo de infra-estrutura (processo organizacional) identifica ferramentas

a serem utilizadas. Tais ferramentas devem permitir a construção cooperativa de documentos via Internet. Ao final, os documentos projetados devem ser revisados pelo processo de revisões, considerando seu formato e conteúdo técnico.

3. ***Produção***
   Os documentos são produzidos, de acordo com o plano de documentação, e armazenados seguindo os requisitos de segurança, manutenção e backup. Os documentos devem poder ser acessados por todas as equipes através da rede de computadores.

4. ***Manutenção***
   As tarefas a serem realizadas quando um documento é modificado são realizadas, de acordo com o processo de manutenção. As mudanças são gerenciadas de acordo com o processo de gerência de configuração e apoiadas por um sistema na Internet.

IV.3.6.2. Processo de Publicação de Resultados

O processo de publicação de resultados se refere às atividades para publicação de resultados das atividades do ciclo de vida do software. Este processo planeja, projeta, e distribui informações sobre os artefatos produzidos e o estado do processo através da Internet.

Atividades:

1. ***Implantação do Processo***
   Esta atividade define um plano para a publicação de resultados, identificando todas que informações sobre os artefatos e as atividades do processo que serão publicadas, tais como o estado do projeto, decisões, restrições, mudanças, etc. Determina-se, também, em que etapas do ciclo de vida do software tais informações serão disponibilizadas às equipes de trabalho. O plano deve ser documentado, segundo o próprio processo de documentação.

2. ***Projeto***
   O processo de infra-estrutura (processo organizacional) identifica as ferramentas a serem utilizadas na publicação de resultados através da Internet.

3. ***Distribuição***
   Os resultados são publicados por eletrônico através da Internet.

IV.3.6.3. Processo de Gerência da Configuração

Este processo está relacionado aos procedimentos administrativos e técnicos para identificar os itens do software, controlar modificações e as versões dos itens, registrar

o estado de cada item, garantir a consistência e correção dos itens e controlar o armazenamento, manipulação e entrega dos itens. A gerência da configuração é realizada em duas etapas: uma que se refere aos itens criados por cada equipe de software e outra que diz respeito aos artefatos resultantes da integração de todos os itens.

Atividades:

1. ***Implantação do Processo***
   O plano para a gerência de configuração é produzido, determinando as tarefas para gerenciar a configuração do software, cronogramas e procedimentos para realizar as tarefas, as responsabilidades das equipes de trabalho e as relações com os processos de desenvolvimento e manutenção. Inicialmente, um plano é definido para cada equipe e, então, os planos são integrados formando o plano para a gerência de configuração para todo o projeto. O plano deve ser documentado, segundo o processo de documentação e publicado a todas as equipes através da Internet.

2. ***Identificação da configuração***
   Definem-se os itens de software e suas versões a serem controlados durante o projeto. Os itens de software referem-se tanto aos componentes construídos por cada equipe como os artefatos resultantes da integração dos componentes. Determinam-se os números das versões ou outro tipo de identificação a ser utilizado.

3. ***Determinação de responsabilidades***
   Para cada configuração, identificada na atividade anterior, determina-se a equipe que será responsável por seu controle, de acordo com o processo de capacitação. A equipe pode ser local ou do projeto como um todo.

4. ***Controle da configuração***
   Esta atividade se refere à identificação e registro de pedidos de mudanças, análise e avaliação das mudanças, aprovação do pedido e implementação, verificação e entrega do item modificado. É importante oferecer um sistema de apoio ao controle da configuração através da Internet e de fácil acesso a todas as equipes.

5. ***Determinação do estado da configuração***
   Gerência e registro de relatórios sobre o estado atual e histórico de um item do software, incluindo identificação das versões, número de versões e diferenças entre as versões. Utilização do sistema de apoio ao controle da configuração através da Internet.

6. ***Avaliação da configuração***
   Avalia-se o aspecto funcional de um item, verificando se foram respeitadas

as mudanças requisitadas. Verifica-se, também, se a nova versão respeita os requisitos definidos anteriormente e se o projeto e documentação estão atualizados. Tal atividade é realizada pelos coordenadores locais e pela gerência do projeto.

7. ***Entrega e gerência de configuração***
   A entrega dos itens deve ser controlada, verificando-se se o código e a documentação foram armazenados, empacotados e entregues de acordo com o que foi planejado. A entrega deve ser publicada a todas as equipes através da Internet.

IV.3.6.4. Processo de Controle da Qualidade

O processo de controle da qualidade reúne atividades para garantir que os produtos e os processos de software respeitam os requisitos e são coerentes com os respectivos planos. O controle da qualidade utiliza resultados dos outros processos de apoio. Para que o controle da qualidade seja efetivo, é necessário que os requisitos e os planos estejam disponíveis a todas as equipes de software.

Atividades:

1. ***Implantação do processo***
   Os objetivos do processo de qualidade são identificados. Determinam-se as relações com os processos de verificação, validação e revisão. Define-se um plano para conduzir as atividades de controle da qualidade, o qual inclui padrões, métodos, procedimentos, recursos, cronogramas e informações a serem coletadas. O plano deve ser documentado, segundo o processo de documentação. O plano é estruturado em dois níveis: um que se refere às tarefas de controle da qualidade a serem realizadas por cada equipe e outro às tarefas de controle da qualidade dos artefatos integrados.

2. ***Determinação de responsabilidades***
   Determina-se a equipe ou integrantes de cada equipe que será responsável pelo controle da qualidade, de acordo com o processo de capacitação. A equipe pode ser local ou do projeto como um todo.

3. ***Projeto***
   O processo de infra-estrutura (processo organizacional) identifica as ferramentas a serem utilizadas na execução das atividades de controle da qualidade. É importante oferecer um sistema de apoio ao controle da qualidade através da Internet e de fácil acesso a todas as equipes.

4. ***Controle de produtos***
   Deve-se garantir que todos os artefatos do software e respectiva documentação seguiram os planos e satisfazem os requisitos. Eventuais problemas são

notificados, de acordo com o processo de publicação de resultados, e a sua resolução é acompanhada pelos coordenadores locais e pela gerência do projeto, de acordo com o processo de resolução de problemas.

5. ***Controle de processos***
   Deve-se garantir que os processos (primários e de apoio) seguiram os planos e respeitam os padrões. Deve-se garantir que as equipes têm habilidades e conhecimento para a realização das tarefas. Eventuais problemas são notificados, de acordo com o processo de publicação de resultados, e a sua resolução é acompanhada pelos coordenadores locais e pela gerência do projeto, de acordo com o processo de resolução de problemas.

IV.3.6.5. Processo de Verificação

O processo de verificação determina se os artefatos de uma atividade respeitam os requisitos ou condições impostos em atividades anteriores. Este processo inclui análises, revisões e testes. O processo de verificação é estruturado em dois níveis: um se refere à verificação dos artefatos produzidos por cada equipe de software e outro diz respeito à verificação dos artefatos resultantes da integração de resultados parciais.

Atividades:

1. ***Implantação do processo***
   Os requisitos do projeto devem ser analisados, identificando-se os riscos envolvidos caso existam erros, as restrições da tecnologia escolhida e a disponibilidade de recursos. Um plano de verificação é elaborado, identificando as tarefas de verificação a serem realizadas por cada equipe e os recursos necessários para todo o processo. O plano deve ser documentado, segundo o processo de documentação e publicado a todas as equipes através da Internet.

2. ***Determinação de responsabilidades***
   Determina-se a equipe ou integrantes de cada equipe que será responsável pelas atividades de verificação, considerando os riscos identificados anteriormente e seguindo o processo de capacitação. A equipe pode ser local ou do projeto como um todo.

3. ***Projeto***
   O processo de infra-estrutura (processo organizacional) identifica as ferramentas a serem utilizadas na execução das atividades de verificação. É importante oferecer um sistema de apoio à verificação de produtos através da Internet e de fácil acesso a todas as equipes.

4. ***Verificação de produtos***
   Reúne as tarefas para verificar o projeto, código, integração e documentação. Os problemas são notificados, de acordo com o processo de publicação de resultados, e a resolução é acompanhada pelos coordenadores locais e pela gerência do projeto, de acordo com o processo de resolução de problemas.

5. ***Verificação de processos***
   Consiste nas tarefas cujo objetivo é verificar se os planos, a implantação dos processos, padrões e equipes são adequados. Os problemas são notificados, de acordo com o processo de publicação de resultados, e a resolução é acompanhada pelos coordenadores locais e pela gerência do projeto, de acordo com o processo de resolução de problemas.

IV.3.6.6. Processo de Validação

O processo de validação determina se o produto final atingiu seus requisitos.

Atividades:

1. ***Implantação do processo***
   Um plano de validação é elaborado, identificando os itens a serem validados, as tarefas de validação a serem realizadas, os recursos necessários para o processo de validação. O plano deve ser documentado, segundo o processo de documentação.

2. ***Determinação de responsabilidades***
   Determina-se a equipe, ou integrantes de cada equipe, que será responsável pelas atividades de validação de acordo com o processo de capacitação. A equipe de validação é responsável por todo o projeto.

3. ***Projeto***
   O processo de infra-estrutura (processo organizacional) identifica as ferramentas a serem utilizadas na execução das atividades de validação. É importante oferecer um sistema de apoio à validação de produtos através da Internet e de fácil acesso a todas as equipes.

4. ***Validação de produtos***
   Análise de resultados dos testes de acordo com a especificação do software. Os problemas são notificados, de acordo com o processo de publicação de resultados, e a resolução é acompanhada pelos coordenadores locais e pela gerência do projeto, de acordo com o processo de resolução de problemas.

IV.3.6.7. Processo de Revisões

O processo de revisão avalia o estado e os artefatos de uma atividade do projeto, envolvendo aspectos técnicos e gerenciais.

Atividades:

1. ***Implantação do processo***
   Identificação dos itens a serem revistos e dos recursos necessários para realizar as revisões. Elaboração do plano de revisões que é documentado, segundo o processo de documentação. Tais informações devem ser publicadas a todas as equipes através da Internet.

2. ***Determinação de responsabilidades***
   Determina-se a equipe, ou integrantes de cada equipe, que será responsável pelas revisões, de acordo com o processo de capacitação. A equipe pode ser local ou do projeto como um todo.

3. ***Projeto***
   O processo de infra-estrutura (processo organizacional) identifica as ferramentas a serem utilizadas na execução das atividades de revisões. É importante oferecer um sistema de apoio às revisões de produtos através da Internet e de fácil acesso a todas as equipes.

4. ***Revisões gerenciais***
   estado do projeto deve ser avaliado segundo o que foi determinado nos planos. O progresso das atividades deve ser notificado, segundo o processo de publicação de resultados. Os problemas são notificados, de acordo com o processo de publicação de resultados, e a resolução é acompanhada pelos coordenadores locais e pela gerência do projeto, de acordo com o processo de resolução de problemas.

5. ***Revisões técnicas***
   Os artefatos e serviços são avaliados, verificando-se se estão completos, respeitam os padrões e especificações e estão prontos para se poder executar a próxima atividade. Os problemas são notificados, de acordo com o processo de publicação de resultados, e a resolução é acompanhada pelos coordenadores locais e pela gerência do projeto, de acordo com o processo de resolução de problemas.

IV.3.6.8. Processo de Resolução de Problemas

Este processo refere-se à análise e resolução de problemas encontrados durante a execução dos outros processos. Como as equipes se encontram em diferentes localidades, os problemas devem ser publicados e o processo de resolução deve ser acompanhado por um sistema em rede de computadores. É responsabilidade da gerência estruturar o problema, identificar qual equipe é responsável por que parte

do problema e garantir a sua solução. É responsabilidade dos coordenadores locais administrar a comunicação entre as equipes para a resolução dos problemas.

Atividades:

1. ***Implantação do processo***
   Elaboração do plano de resolução de problemas, garantindo que após sua detecção, os problemas são rapidamente notificados e as causas são identificadas, analisadas e, se possível, eliminadas. Cada problema deve ser classificado por uma categoria e priorizado. A resolução de problemas deve ser acompanhada e publicada, segundo o processo de publicação de resultados. O plano é documentado, segundo o processo de documentação.

2. ***Determinação de responsabilidades***
   Determina-se a equipe ou integrantes de cada equipe que será responsável pela resolução dos problemas, de acordo com o processo de capacitação. A equipe pode ser local ou do projeto como um todo.

3. ***Resolução de problemas***
   Quando um problema é identificado, um relatório é preparado, descrevendo-o. Este documento é atualizado durante as tarefas de investigação do problema, análise e resolução. É importante oferecer um sistema de apoio à resolução de problemas através da Internet de fácil acesso por todas as equipes.

IV.3.7. Processos Organizacionais

Os processos organizacionais estão relacionados às características, capacitação, recursos e infra-estrutura de empresas ou instituições que desenvolvem software. Estabelecem a estrutura necessária à realização das tarefas dos outros processos e definem atividades para a melhoria dos recursos humanos, através de treinamentos, e do próprio processo de desenvolvimento de software.

IV.3.7.1. Processo de Capacitação

O processo de avaliação das equipes determina a capacidade de cada grupo de trabalho, através da identificação de seus recursos humanos, computacionais e econômicos.

Atividades:

1. ***Implantação do processo***
   Elaboração do plano para obter informações sobre as equipes de trabalho,

através de entrevistas, observações e coleta de dados. O plano é documentado, segundo o processo de documentação.

2. ***Coleta de dados***
   Nesta atividade, as seguintes informações identificadas anteriormente, são coletadas:

   - número de equipes de trabalho
   - quantidade de membros por equipe
   - formação e experiência de cada membro de uma equipe
   - lista dos recursos computacionais (hardware e software)
   - disponibilidade econômica
   - capacidade técnica de cada equipe
   - capacidade gerencial de cada equipe

3. ***Determinação de capacidades***
   A partir das informações obtidas, determina-se a capacidade de cada grupo de software e associam-se perfis para cada membro. Esta tarefa é realizada pela coordenação local.

IV.3.7.2. Processo de Comunicação

O processo de comunicação refere-se a todos os aspectos do desenvolvimento de software que requerem a troca de informações entre as equipes para a resolução de problemas, notificação de decisões, registro, acesso e compartilhamento de dados do software e publicação de informações que permitam o acompanhamento de atividades.

Atividades:

1. ***Implantação do processo***
   Elaboração do plano para determinar as informações a serem comunicadas e identificar os recursos disponíveis para transmitir e receber mensagens. O plano é documentado, segundo o processo de documentação.

2. ***Determinação dos meios e protocolos de comunicação***
   Identificação de métodos e procedimentos, meios tradicionais ou eletrônicos para a troca de informações, considerando os recursos técnicos dos grupos. Os protocolos de comunicação são definidos e implementados.

IV.3.7.3. Processo de Gerência

O processo de gerência contém atividades genéricas que podem ser aplicadas para gerenciar um determinado processo. Um gerente é responsável por um produto, projeto ou tarefa.

Atividades:

1. ***Início e definição do escopo***
   O processo de gerência é iniciado com a determinação dos requisitos dos processos a serem executados. A gerência deve identificar a possibilidade de executar o processo, determinando os recursos das equipes e avaliando a sua adequação.

2. ***Planejamento***
   O gerente deve preparar um plano para a execução de um processo, contendo a descrição das atividades e dos produtos de software a serem realizados. O plano deve conter cronogramas, estimativas de esforço, recursos necessários, alocação de tarefas, determinação de responsabilidades, quantificação dos riscos associados a cada tarefa, medições e custos.

3. ***Execução e controle***
   O gerente deve iniciar a implantação do plano e monitorar a execução do processo, informando o progresso das atividades, de acordo com o processo de publicação de resultados. O gerente deve investigar, analisar e solucionar problemas, de acordo com o processo de resolução de problemas. É responsabilidade do gerente garantir que eventuais mudanças não introduzam inconsistências em outras tarefas.

IV.3.7.4. Processo de Coordenação

O processo de coordenação reúne atividades para coordenar as equipes de trabalho. Um coordenador é responsável por um produto ou tarefa realizada por uma determinada equipe.

Atividades:

1. ***Início e definição do escopo***
   O processo de coordenação é iniciado com a determinação dos requisitos das atividades a serem executadas e a identificação das responsabilidades dos grupos de trabalho. A gerência deve identificar a possibilidade de realizar a tarefa, determinando os recursos da equipe e avaliando a sua adequação.

2. ***Planejamento***
   O coordenador deve preparar um plano para a execução de uma atividade, contendo a sua descrição e os produtos a serem gerados. O plano deve conter cronogramas, estimativas de esforço, recursos necessários, medições e custos.

3. ***Execução e controle***
   O coordenador deve iniciar a implantação do plano e monitorar a execução da tarefa, informando o seu progresso, de acordo com o processo de publicação de resultados. O coordenador deve investigar, analisar e solucionar problemas de cada equipe, de acordo com o processo de resolução de problemas.

IV.3.7.5. Processo de Controle de Artefatos

O processo de controle de artefatos se refere às atividades para controlar a produção de artefatos, realizados pelas equipes de software, e integrá-los.

Atividades:

1. ***Início e definição do escopo***aa O processo é iniciado com a determinação dos requisitos das atividades a serem executadas e respectivos artefatos a serem produzidos por cada equipe.

2. ***Planejamento***
   Define-se um plano para a realização de artefatos, identificando as interfaces com outros artefatos, produzidos pelas diversas equipes de software. O plano deve ser documentado, de acordo com o processo de documentação.

3. ***Execução e controle***
   plano deve ser implantado, a geração e a integração dos artefatos deve ser monitorada, informando o seu progresso, de acordo com o processo de publicação de resultados. Os problemas e sua solução são notificados, de acordo com o processo de resolução de problemas. É importante utilizar um sistema de apoio à execução e controle de artefatos através da Internet de fácil acesso por todas as equipes de software.

IV.3.7.6. Processo de Infra-estrutura

Este processo estabelece e mantém a infra-estrutura necessária para os outros processos. A infra-estrutura se refere a equipamentos, sistemas computacionais, ferramentas, técnicas e padrões. Como as equipes de software são dispersas, as ferramentas a serem utilizadas devem apoiar tarefas a serem realizadas através da Internet e oferecer mecanismos, tais como controle de versões e gerência da concorrência.

Atividades:

1. ***Implantação do processo***
   A infra-estrutura necessária deve ser definida e o planejamento para a sua instalação é documentado, segundo o processo de documentação. Aspectos como

funcionalidade, desempenho, segurança, equipamentos, custos e restrições devem ser considerados.

2. ***Instalação da infra-estrutura***
   A infra-estrutura é instalada para a execução dos processos e apoio a atividades distribuídas, como identificadas no processo de planejamento.

3. ***Manutenção da infra-estrutura***
   A infra-estrutura deve ser mantida, monitorada e modificada, caso necessário para garantir que os requisitos de cada processo sejam respeitados.

IV.3.7.7. Processo de Melhoria

Este processo se refere às medições, controle e melhoria dos processos do ciclo de vida do software.

Atividades:

1. ***Determinação do processo***
   Um mecanismo de controle é definido para monitorar, controlar e melhorar o processo utilizado por cada equipe de software e o processo que envolve todo o projeto.

2. ***Avaliação do processo***
   Um procedimento de avaliação do processo deve ser planejado, documentado e executado. Dados devem ser coletados e analisados para permitir a identificação de inconsistências nos processos. A avaliação também deve determinar a adequação e eficiência dos processos do ciclo de vida do software. A avaliação é realizada em duas etapas: uma refere-se à avaliação do processo utilizado por cada equipe e outro diz respeito ao processo padrão definido pela organização.

3. ***Melhoria do processo***
   A organização deve sugerir mudanças e identificar fraquezas para melhorar tanto os processos seguidos pelas equipe quanto o processo padrão da organização. A documentação dos processos deve ser atualizada, refletindo as melhorias e publicada a todas as equipes, através da Internet.

IV.3.7.8. Processo de Treinamento

O processo de treinamento se refere à melhoria do conhecimento das equipes de trabalho, no que se refere às habilidades gerenciais e técnicas.

Atividades:

1. ***Implantação do processo***
   Realiza-se a revisão dos requisitos para determinar os recursos e conhecimentos necessários para a realização das tarefas. De acordo com o processo de capacitação, identificam-se as equipes que precisam ser treinadas e os tópicos de treinamento. Define-se um plano de treinamento para cada equipe, que é documentado, de acordo com o processo de documentação.

2. ***Treinamento***
   Elaboração do material e treinamento de cada equipes.

IV.4. Considerações Finais

Este capítulo descreveu as características de projetos de software realizados por equipes geograficamente dispersas. Através desta descrição, identificaram-se os problemas e determinaram-se os requisitos a serem respeitados por processos de software específicos ao desenvolvimento distribuído de sistemas computacionais. A definição do processo de software padrão para equipes geograficamente dispersas respeitou tais requisitos, adaptando-se a norma ISO/IEC 12207.

A tabela IV.1 relaciona as classes de problemas detectados no desenvolvimento de software realizado por equipes geograficamente distribuídas com os requisitos que um processo deve respeitar para solucionar tais problemas e as categorias de processos que atendem aos requisitos.

Algumas das dificuldades encontradas no desenvolvimento distribuído de software também estão presentes em outros contextos. Entretanto, são agravadas quando os grupos de trabalho não se encontram em um mesmo lugar pois, muito provavelmente, as equipes são heterogêneas. Neste caso, a gerência do processo de software deve colocar uma maior atenção na coordenação das equipes e respectivas atividades, controle dos artefatos e na comunicação entre os profissionais envolvidos no projeto.

Classes de Problemas	Requisitos	Categorias do Processo
Coordenação de equipes	R1: Determinar a capacidade das equipes	Processo de Capacitação Processo de Planejamento
	R2: Apoiar a coordenação distribuída	Processo de Coordenação
	R3: Apoiar a gerência distribuída do projeto	Processo de Gerência
	R5: Apoiar a comunicação entre as equipes	Processo de Comunicação Processo de Infra-estrutura
	R8: Incorporar um repositório de terminologias	Processo de Definição Processo de Desenvolvimento Processo de Manutenção
Coordenação das atividades	R2: Apoiar a coordenação distribuída	Processo de Coordenação
	R3: Apoiar a gerência distribuída do projeto	Processo de Gerência
	R6: Apoiar a publicação e o compartilhamento de informações	Processo de Documentação Processo de Publicação de Resultados
	R7: Apoiar a resolução de problemas	Processo de Resolução de Problemas
	R9: Apoiar a redefinição de atividades do software	Processo de Planejamento
Controle de artefatos	R4: Apoiar o controle de artefatos	Processo de Controle de Artefatos Processo de Gerência da Configuração
	R6: Apoiar a publicação e o compartilhamento de informações	Processo de Documentação Processo de Publicação de Resultados
Comunicação	R5: Apoiar a comunicação entre as equipes	Processo de Comunicação Processo de Infra-estrutura
	R6: Apoiar a publicação e o compartilhamento de informações	Processo de Publicação de Resultados

Tabela IV.1: Quadro comparativo entre os problemas de equipes dispersas, requisitos e aspectos do processo padrão.

V. Modelo de Gerência de Processos para Equipes Geograficamente Dispersas

Neste capítulo, o modelo de gerência de processos de software para equipes de trabalho geograficamente dispersas é apresentado. O objetivo primário do modelo proposto é apoiar a produção, realizada por grupos distribuídos em diferentes localizações, de sistemas computacionais de qualidade, melhorando sistematicamente a capacidade de uma organização para produzir tais sistemas. Tal meta é alcançada através de quatro níveis de gerência de processos: definição, acompanhamento, avaliação e melhoria.

V.1. Introdução

No capítulo II, a evolução das pesquisas sobre processo de software foi apresentada. Definiu-se gerência do processo de software e justificou-se a importância de aprimorar continuamente um processo de software. O ponto-chave para a melhoria dos processos de software corresponde à coleta de informações que permitem verificar como as atividades levam à qualidade do software e como o processo de desenvolvimento influencia o produto final. Outro aspecto a ser considerado para aprimorar um processo de software é a sua adequação ao domínio da aplicação e às equipes de trabalho, considerando suas características e restrições. A cada projeto, o processo de software deve ser ajustado às especificidades da aplicação, à tecnologia a ser utilizada na sua construção, ao grupo de desenvolvimento e aos usuários finais (FALBO, 1996, WASSERMAN, 1996).

Atualmente, a Internet é globalmente utilizada e oferece a possibilidade de desenvolver projetos cooperativos com a participação de equipes de trabalho distribuídas. A comunicação é apoiada eletronicamente e diversos profissionais, provenientes de vários domínios de aplicação, podem realizar suas atividades, trocando informações e conhecimento. Entretanto, por trabalhar em diferentes localizações, os profissionais utilizam diversos equipamentos e métodos e dificilmente as equipes são homogêneas.

Tudo isso torna difícil e inadequada a determinação de um único processo de software (MAURER e KAISER, 1998). As vantagens oferecidas pela rede de computadores e as mudanças sócio-econômicas mundiais exigem que o desenvolvimento de projetos de software seja repensado.

Desta forma, torna-se evidente a necessidade de definir um modelo de gerência de processos, o qual apóie as tarefas de padronizar um processo de software, configurar o processo padrão conforme as características dos diversos grupos, acompanhar a sua utilização, reunindo dados que permitam a avaliação do processo de software, e, finalmente, oferecer procedimentos para aprimorar a qualidade do próprio processo de software. Tais tarefas não poderiam ser facilmente realizadas em ambientes dispersos sem o apoio de uma tecnologia adequada. Projetos de software geram inúmeras informações, em diferentes níveis de abstração e formatos. Quando realizados por grupos heterogêneos, a complexidade de integrar os diversos componentes do software é agravada pela incompatibilidade dos equipamentos.

O capítulo III apresenta a tecnologia WWW, seus protocolos e linguagens de comunicação para a rede de computadores e o estado da arte em sistemas Web de apoio à construção de software. Sistemas de hipermídia podem modificar a maneira como o software é desenvolvido, mantido, testado e otimizado. Entretanto, torna-se importante salientar que a grande restrição encontrada em projetos geograficamente dispersos não corresponde somente à descentralização do trabalho, mas sim à diversidade cultural e ao conhecimento diferenciado entre os profissionais. Portanto, a contribuição da tecnologia WWW é possibilitar o desenvolvimento de ferramentas de apoio à construção assíncrona de software realizada por equipes distribuídas (HILBERT e REDMILES, 1998b).

Neste capítulo, apresentamos uma proposta de solução para o desenvolvimento distribuído de software, que contribui ao estado da arte em processos de software. Tal proposta baseia-se na premissa de que a qualidade dos produtos de software e a produtividade das equipes de trabalho está diretamente relacionada à qualidade e à adequação do processo de software utilizado para construí-los (CURTIS *et al.*, 1992). A mudança necessária na engenharia de software para acompanhar a evolução mun-

dial, disparada através da globalização e apoiada pela Internet, é justamente retirar a responsabilidade do sucesso de projetos de software exclusivamente das mãos dos profissionais, e delegá-la aos processos de software das organizações. Além disto, a qualidade dos produtos de software não corresponde a um único obstáculo a ser superado através de equipamentos modernos, ferramentas computacionais avançadas ou técnicas revolucionárias. A qualidade do software corresponde a um esforço que envolve profissionais, organizações e tecnologia. Tal afirmação, no contexto filosófico da pesquisa, exige, portanto, uma busca contínua do aprimoramento de processos de software.

V.2. Descrição Geral do Modelo de Gerência

O modelo de gerência de processos de software tem por objetivo apoiar o desenvolvimento de software realizado por equipes de trabalho heterogêneas, geograficamente dispersas, cultural e tecnologicamente diferentes. Suas metas são: garantir que projetos distribuídos sejam viáveis com grupos de diferentes níveis de maturidade, melhorar a capacidade de cada equipe e aprimorar a capacidade da organização como um todo. Tais metas são alcançadas através das seguintes atividades:

- definir um processo padrão para equipes geograficamente distribuídas;
- especializar o processo de software padrão em diferentes níveis de maturidade;
- identificar a capacidade de cada grupo de trabalho;
- instanciar o processo de software para cada projeto e acompanhar a sua utilização;
- avaliar como cada grupo de trabalho utilizou o processo de software;
- identificar procedimentos para melhorar a capacidade de cada grupo;
- aprimorar o processo de software padrão, baseando-se na experiência da sua utilização.

O processo padrão tem como principal objetivo fornecer uma estrutura única para que as equipes distribuídas envolvidas em um projeto de software utilizem uma linguagem comum. Para que o processo possa ser adaptado à necessidade do produto a ser construído e às características dos grupos de trabalho, a sua estrutura foi baseada em dois princípios básicos: modularidade e responsabilidade. O processo é organizado em um conjunto de sub-processos, cada qual possui uma única função no ciclo de vida. Define-se, então, responsabilidades para executar cada sub-processo, facilitando a aplicação do processo padrão em um projeto (MACHADO *et al.*, 1997).

Especialização corresponde ao processo no qual uma abstração é representada de forma específica a um determinado contexto, através da inclusão, modificação ou, até mesmo, encapsulamento de alguns de seus atributos. Para atender às diferentes características dos grupos de trabalho, o processo padrão para equipes geograficamente distribuídas é especializado em quatro processos, cada qual correspondendo a um nível de maturidade. Para especializar o processo de software, é necessário ter como referência um modelo que determine os diferentes níveis de maturidade de um processo e como alcançá-los. O modelo de maturidade de processos escolhido foi o CMM por já ter sido utilizado, referenciado e avaliado em diversos trabalhos (PAULK, 1994, HERBSLEB e GOLDENSON, 1996, GRAHL *et al.*, 1997, HERBSLEB *et al.*, 1997).

O modelo de maturidade foi adaptado para atender às necessidades do desenvolvimento de software geograficamente distribuído. Novas áreas-chave para cada nível de maturidade foram incluídas e as existentes foram adaptadas. O primeiro processo especializado corresponde ao segundo nível (repetitivo) e apresenta as atividades referentes às áreas-chave de processo para este nível. Cada nível mais alto, engloba outras atividades relativas às áreas-chave de processo específicas ao nível de maturidade correspondente. As especializações do processo incorporam a capacidade necessária para o desenvolvimento de software em um determinado nível de maturidade, através da descrição de atividades específicas àquele nível. A execução de uma especialização do processo é que vai determinar a sua maturidade, ou seja, até que ponto o processo é definido, gerenciado, medido, controlado e eficiente.

Através de uma avaliação, a capacidade de cada grupo é identificada e escolhe-se a especialização mais adequada. A cada projeto, uma instância da especialização escolhida é gerada. Durante o desenvolvimento do projeto, coletam-se informações que serão utilizadas para verificar se a equipe utilizou a especialização do processo de software de forma correta. Ao final, define-se uma lista de ações a serem seguidas pela equipe com o intuito de aprimorar a sua capacidade. A cada novo projeto, identifica-se se a capacidade de desenvolvimento de software da equipe foi aprimorada e, em caso afirmativo, a equipe pode utilizar uma especialização do processo de nível mais alto.

A utilização e avaliação das especializações do processo padrão também permitem identificar possíveis falhas e deficiências do processo padrão, o que permite a sua melhoria. A figura V.1 apresenta uma representação esquemática do modelo de gerência.

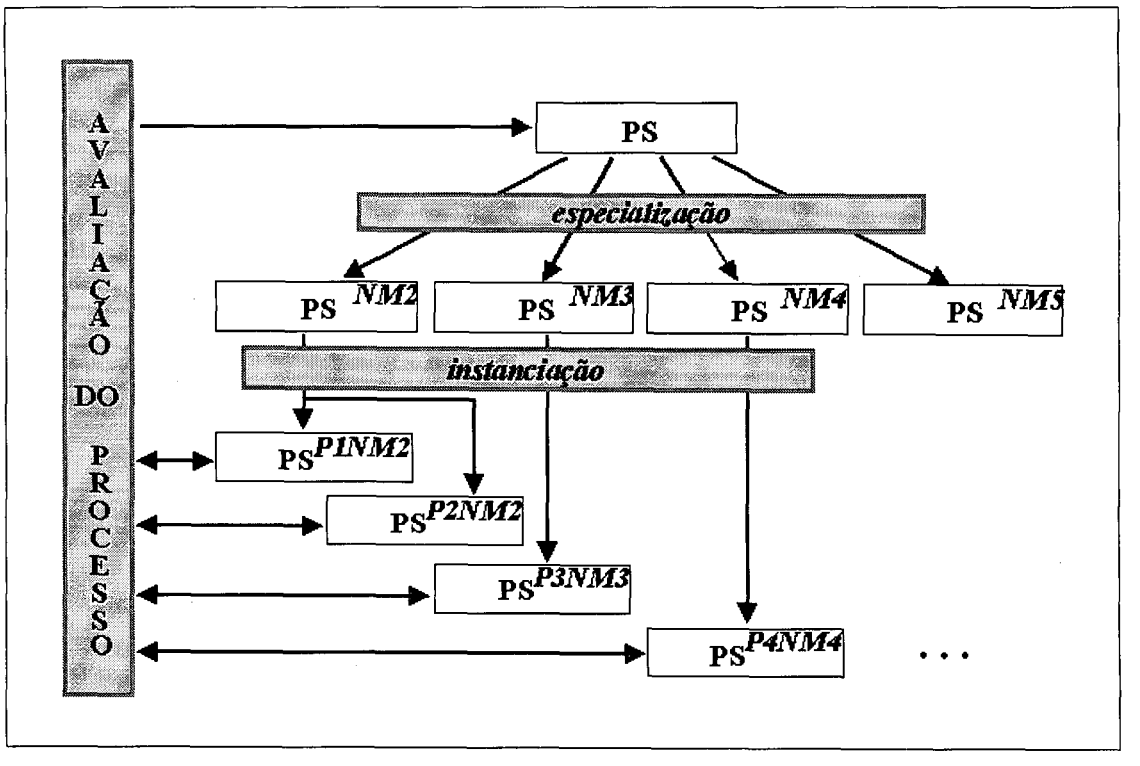


Figura V.1: Representação do Modelo de Gerência de Processos de Software para Equipes Geograficamente Dispersas.

O modelo de gerência apoia a melhoria contínua do processo de software, que é alcançada através dos seguintes níveis de gerência: **definição** (especializa o pro-

cesso padrão em diferentes níveis de maturidade), **acompanhamento** (define a instância do processo especializado, registra as atividades de cada projeto e coleta dados), **avaliação** (realizada através das atividades de medição, análise e crítica) e **melhoria** (apoiada pela definição de um plano de ações).

V.3. Modelo de Maturidade de Processos para Equipes Geograficamente Dispersas

O modelo de maturidade de processos para equipes geograficamente dispersas permite que o processo de software utilizado por uma organização possa ser analisado, com base em um conjunto de critérios. A análise identifica a capacidade do processo em produzir software, respeitando os requisitos de qualidade, dentro dos prazos e orçamentos estabelecidos. Além disto, pode-se também identificar as dificuldades que levam à produção de produtos de software de baixa qualidade ou que não respeitem prazos e orçamentos.

Em ambiente geograficamente distribuídos, cada grupo de trabalho pode ter diferentes características. Na linguagem de "processos", pode-se dizer que cada equipe pode ter diferentes capacidades. A proposta do modelo de gerência é permitir que cada grupo de software utilize um processo com nível de maturidade correspondente à sua capacidade. A coordenação local deve garantir que o processo a ser seguido seja adequado e que a sua execução seja corretamente realizada. A gerência das equipes deve garantir que o produto de software seja corretamente produzido mesmo que os componentes sejam desenvolvidos por equipes com diferentes capacidades. É, também, responsabilidade da coordenação e da gerência estimular a melhoria contínua da capacidade de cada grupo de trabalho e, por conseqüência, de toda a organização.

V.3.1. Conceitos Primários do Modelo de Maturidade

O modelo de maturidade de processos para equipes geograficamente distribuídas foi elaborado para determinar a capacidade e apoiar a melhoria destes processos. O

modelo pode ser aplicado para qualquer domínio de aplicação e é independente da estrutura organizacional e dos métodos e tecnologias utilizados.

V.3.2. Estrutura do Modelo de Maturidade

O processo padrão corresponde ao ponto inicial para a aplicação do modelo de gerência. Definem-se, então, todas as tarefas e práticas envolvidas na produção e evolução de um produto de software a serem realizadas por equipes com capacidades diferentes.

No nível repetitivo (2^o nível de maturidade) do CMM, existem seis áreas-chave, totalizando sessenta e duas atividades, relacionadas à gerência dos requisitos, planejamento, acompanhamento do projeto, garantia da qualidade e gerência da configuração do software. No modelo proposto, o nível de maturidade repetitivo engloba oito áreas-chave, totalizando noventa e cinco recomendações. As áreas-chave "Gerência da Comunicação " e "Gerência da Capacidade das Equipes" foram inseridas e a área-chave "Gerência da Sub-contratação do Software" foi substituída por "Gerência da Delegação de Atividades do Software", para apoiar equipes geograficamente dispersas. No nível definido, as áreas-chave "Definição das Especializações do Processo Organizacional" e "Coordenação Distribuída do Software" foram inseridas no modelo. A área-chave "Coordenação Quantitativa das Especializações" determina novas recomendações a serem seguidas no nível de maturidade gerenciado. A área-chave "Gerência da Mudança das Especializações" completa o nível de maturidade otimizado para o desenvolvimento distribuído de software.

A tabela V.1, apresenta as diferenças entre as áreas-chave do CMM e do Modelo de Maturidade para equipes geograficamente dispersas.

Nível	CMM	Modelo de Maturidade para Equipes Geograficamente Dispersas
2	Gerência de Requisitos	Gerência Distribuída de Requisitos
	Planejamento do Projeto de Software	Planejamento Distribuído do Projeto de Software
	Acompanhamento e Supervisão do Projeto	Acompanhamento e Supervisão Distribuídos do Projeto
	Gerenciamento da Sub-contratação de Software	Gerência da Delegação de Atividades do Software
	Garantia da Qualidade de Software	Garantia da Qualidade de Software
	Gerência de Configuração de Software	Gerência Distribuída de Configuração de Software
		Gerência da Comunicação
		Gerência da Capacidade das Equipes
3	Foco nos Processos da Organização	Foco nos Processos da Organização
	Definição do Processo da Organização	Definição do Processo Padrão da Organização
		Definição das Especializações do Processo Organizacional
	Programa de Treinamento	Programa de Treinamento
	Gerência do Software Integrado	Gerência do Software Integrado
		Coordenação Distribuída do Software
	Engenharia do Produto de Software	Engenharia Distribuída do Produto de Software
	Coordenação Inter-Grupos	Coordenação Inter-Grupos
	Revisões	Revisões Distribuídas
4	Gerência Quantitativa do Processo	Gerência Quantitativa do Processo
		Coordenação Quantitativa das Especializações
	Gerência da Qualidade de Software	Gerência da Qualidade de Software
5	Prevenção de Defeito	Prevenção de Defeitos
	Gerência da Mudança Tecnológica	Gerência da Mudança Tecnológica
	Gerência da Mudança do Processo	Gerência da Mudança do Processo
		Gerência da Mudança das Especializações

Tabela V.1: Quadro Comparativo entre o CMM e o Modelo de Maturidade de Processos Geograficamente Distribuídos.

A especialização de cada nível de maturidade determina recomendações que devem ser respeitadas durante os projetos de software. A seguir, são apresentadas tais recomendações, organizadas por área-chave para cada nível de maturidade:

1. **Nível Inicial**

1.1. **Área-chave:** Não há áreas-chave associadas a este nível.

2. Nível Repetitivo

2.1. Área-chave: Gerência Distribuída de Requisitos

Objetivo: definir e garantir que os requisitos do software sejam respeitados ao longo de um projeto de software por todas as equipes de desenvolvimento. Garantir que o produto resultante da integração dos componentes do software, construídos em diferentes localizações, respeite os requisitos do software.

Recomendações:

- revisão dos requisitos pelos engenheiros de software antes de incorporá-los ao projeto de software.
- definição dos requisitos a serem respeitados por cada componente do software, identificando sua adequação às características da equipe de desenvolvimento de software.
- publicação dos requisitos gerais e específicos a todas as equipes de software.
- utilização dos requisitos como base para o planejamento, construção dos produtos intermediários e realização de atividades para cada equipe de software.
- utilização dos requisitos como base para o planejamento, realização de atividades e integração dos componentes do software para o projeto como um todo.
- revisão das mudanças nos requisitos e identificação de possíveis modificações no planejamento, produtos intermediários e atividades de cada equipe de software.
- revisão das mudanças nos requisitos e identificação de possíveis modificações no projeto de software.
- publicação das mudanças nos requisitos a todas as equipes de software.

2.2. Área-chave: Planejamento Distribuído do Projeto de Software

Objetivo: definir planos para a realização de tarefas técnicas e gerenciais de cada equipe de software e das atividades de integração e coordenação dos trabalhos realizados pelas equipes.

Recomendações:

- participação de engenheiros de software na elaboração da proposta do projeto.
- publicação da proposta do projeto a todas as equipes de software.
- elaboração do planejamento do projeto no estágio inicial do projeto.

- elaboração de planejamentos distintos para cada equipe de software e de um planejamento integrado para o projeto como um todo.
- publicação dos planejamentos parciais e geral a todas as equipes de software.
- participação de engenheiros de software e de outros grupos no planejamento do projeto ao longo de seu desenvolvimento.
- revisão das tarefas delegadas a cada equipe de software, de acordo com um procedimento documentado.
- definição do modelo de ciclo de vida do software com estágios pré-definidos, identificando os pontos de controle para integração dos resultados de cada equipe de software.
- desenvolvimento das atividades dispersas do projeto de acordo com um procedimento documentado.
- documentação do planejamento do projeto de software
- publicação do documento resultante do planejamento a todas as equipes de software.
- identificação de ferramentas necessárias para realizar e controlar as atividades de cada equipe e para integrar os resultados parciais produzidos.
- determinação do tamanho de cada componente do software e do produto final a partir de um procedimento documentado.
- estimativa do esforço e custo para cada equipe de software e para o projeto como um todo a partir de um procedimento documentado.
- estimativa de recursos computacionais para cada equipe de software a partir de um procedimento documentado.
- determinação de cronogramas parciais e integrado a partir de um procedimento documentado.
- identificação e documentação de riscos associados a custos, recursos, cronogramas e aspectos técnicos para cada equipe de software.
- registro de informações sobre o planejamento, organizado por equipe de software.

2.3. Área-chave: Acompanhamento e Supervisão Distribuídos do Projeto

Objetivo: apresentar dados para acompanhar o progresso das atividades realizadas por cada equipe de software, permitir a tomada de decisões e prevenir e resolver os problemas encontrados ao longo do projeto.

Recomendações:

- acompanhamento das atividades de cada equipe de software de acordo com um plano de desenvolvimento documentado.
- acompanhamento da comunicação entre equipes e publicação de informações de acordo com um plano de desenvolvimento documentado.
- revisão dos planos de desenvolvimento das equipes de software e do projeto como um todo de acordo com um procedimento documentado.
- delegação de tarefas às equipes de software, de acordo com um procedimento documentado.
- comunicação das mudanças aprovadas a todas as equipes de software.
- acompanhamento do tamanho dos produtos de software (ou tamanho das mudanças dos produtos) realizadas por cada equipe de software.
- acompanhamento de esforços e custos relacionados a cada equipe de software, realizando ações corretivas, caso necessário.
- identificação da necessidade de recursos computacionais de cada equipe de software.
- acompanhamento dos cronogramas parciais.
- acompanhamento de atividades técnicas realizadas por cada equipe de software.
- acompanhamento dos riscos, associados com custos, recursos, cronograma e aspectos técnicos de cada equipe de software.
- registro de métricas do projeto de software, por cada equipe de software.
- realização de revisões periódicas para supervisionar o progresso técnico, planejamento e desempenho de cada equipe de software em relação ao plano de desenvolvimento.
- realização de revisões formais para validar os resultados parciais obtidos por cada equipe, de acordo com um procedimento documentado.

2.4. Área-chave: Gerência da Delegação de Atividades do Software

Objetivo: selecionar profissionais aptos para realizar os diversos aspectos de um projeto de software, identificando que equipe é mais adequada para desenvolver determinado componente do software.

Recomendações:

- definição e planejamento do trabalho a ser delegado.
- seleção das equipes de software, baseada na avaliação da sua habilidade em realizar o trabalho de acordo com um procedimento documentado.
- utilização de planejamentos parciais para coordenar as atividades delegadas a cada equipe de trabalho.
- revisão e aprovação do plano de desenvolvimento por cada equipe de software.
- utilização do plano de desenvolvimento documentado e aprovado para acompanhar as atividades de cada equipe de software.
- resolução de mudanças no planejamento para cada equipe, de acordo com um procedimento documentado.
- realização de revisões periódicas das atividades pela gerência do projeto e pela coordenação local.
- realização de revisões técnicas por cada equipe de software e sua respectiva coordenação.
- realização de revisões formais por cada equipe de software para verificar se os resultados esperados foram obtidos.
- publicação dos resultados parciais a todas as equipes de software.
- monitoração das atividades de controle da qualidade realizada pelo coordenador de cada equipe de software.
- publicação da realização do controle da qualidade a todas as equipes de software.
- monitoração das atividades de gerência da configuração realizada pelo coordenador de cada equipe de software.
- publicação da realização das atividades de gerência da configuração a todas as equipes de software.
- realização de testes de aceitação na entrega dos produtos gerados por cada equipe de software.
- publicação dos resultados dos testes de aceitação a todas as equipes de software.
- avaliação do desempenho de cada equipe de software.

2.5. Área-chave: Garantia da Qualidade de Software

Objetivo: verificar se os produtos parciais construídos por cada equipe de software e o produto final respeitam os padrões de qualidade estabelecidos.

Recomendações:

- preparação de planos para controle da qualidade para cada equipe de software e um plano para controle da qualidade geral de acordo com um procedimento documentado.
- realização das atividades de controle da qualidade por cada equipe de software, de acordo com o plano de qualidade.
- participação do grupo de qualidade no planejamento, padronização e determinação dos procedimentos de cada equipe de software do projeto.
- revisão das atividades técnicas de cada equipe de software pelo grupo de controle da qualidade.
- avaliação dos produtos de software gerados por cada equipe de software pelo grupo de controle da qualidade.
- avaliação do produto de software integrado pelo grupo de controle da qualidade.
- publicação periódica dos resultados das atividades de controle da qualidade a todas as equipes de software.
- documentação de mudanças nas atividades e produtos de software de cada equipe de software, de acordo com um procedimento documentado.
- publicação das mudanças em atividades ou produtos para todas as equipes de software.
- revisão das atividades do grupo de qualidade com os usuários, quando apropriado.

2.6. Área-chave: Gerência Distribuída de Configuração de Software

Objetivo: estabelece e garante a integridade dos componentes do software e do produto final durante todo o ciclo de vida do software.

Recomendações:

- preparação de um plano de gerência da configuração para cada componente de acordo com um procedimento documentado.
- preparação de um plano de gerência da configuração do produto de software integrado de acordo com um procedimento documentado.
- utilização de um plano de gerência da configuração documentado e aprovado para realizar as tarefas relativas à gerência da configuração.

- determinação de uma biblioteca de configurações como um repositório do software.
- identificação dos itens do software, cuja configuração deve ser gerenciada.
- publicação dos itens do software a todas as equipes.
- registro, revisão, aprovação e acompanhamento dos itens do software produzidos por cada equipe de acordo com um procedimento documentado.
- registro, revisão, aprovação e acompanhamento da integração dos itens do software de acordo com um procedimento documentado.
- controle de mudanças de acordo com um procedimento documentado.
- criação e controle de produtos a partir da biblioteca de configurações de acordo com um procedimento documentado.
- registro do estado da configuração de itens de acordo com um procedimento documentado.
- publicação do estado da configuração dos itens a todas as equipes de software.
- produção de relatórios padronizados sobre as atividades da gerência da configuração.
- publicação dos relatórios a todas as equipes de software.
- revisão dos itens de software sob configuração de acordo com um procedimento documentado.

2.7. Área-chave: Gerência da Comunicação

Objetivo: definir e garantir que a comunicação entre as equipes seja realizada.

Recomendações:

- determinação de meios de comunicação efetivos para cada equipe de software.
- especificação das informações a serem comunicadas por cada equipe de software.
- identificação de ferramentas necessárias para realizar a comunicação entre as equipes de software.
- determinação de um repositório de informações que possa ser acessado por todas as equipes de software.
- revisão e aprovação das informações a serem comunicadas antes da sua publicação.

- verificação de que todas as informações a serem comunicadas foram publicadas.

2.8. Área-chave: Gerência da Capacidade das Equipes

Objetivo: definir e garantir que as atividades a serem delegadas a cada equipe de software sejam adequadas à sua capacidade, através da determinação das habilidades e restrições dos grupos de trabalho.

Recomendações:

- determinação da capacidade de cada equipe de software, de acordo com um procedimento documentado.
- identificação dos recursos necessários ao desenvolvimento de cada componente do software, de acordo com um procedimento documentado.
- atribuição de tarefas a cada equipe de software, considerando suas capacidades.
- determinação de um repositório de informações sobre as habilidades e restrições de cada equipe de software.
- manutenção e atualização do repositório de informações das equipes de software.
- publicação das habilidades das equipes de software.

3. Nível Definido

3.1. Área-chave: Foco nos Processos da Organização

Objetivo: definir as capacidades dos processos de software especializados e coordenar as atividades de melhoria do processo padrão da organização.

Recomendações:

- avaliação periódica dos processos de software especializados, através da avaliação de suas instâncias e determinação de um plano de ações para melhorá-los.
- elaboração e manutenção de um plano para implantação e melhoria do processo de software especializado para cada equipe de software.
- elaboração e manutenção de um plano para implantação e melhoria do processo de software padrão da organização.
- coordenação das atividades de implantação e melhoria do processo especializado de cada equipe de software.
- publicação do resultado das atividades de implantação e melhoria dos processos a todas as equipes de software.

- utilização de um banco de dados para coletar informações sobre os processos de software de cada equipe de software e seus resultados.
- monitoração, avaliação e aplicação de novos processos, métodos e ferramentas.
- treinamento para utilização do processo de software.
- divulgação das atividades do projeto ao grupo responsável pela implantação e melhoria do processo e às equipes de software.

3.2. Área-chave: Definição do Processo Padrão da Organização

Objetivo: desenvolver e manter um processo padrão; coletar informações sobre como especializar o processo padrão e utilizá-lo em projetos de software.

Recomendações:

- definição e manutenção do processo de software padrão de acordo com um procedimento documentado.
- documentação do processo de software padrão de acordo com os padrões definidos pela organização.
- publicação da documentação do processo de software padrão.
- documentação e manutenção dos projetos realizados e aprovados, descrevendo os processos de software utilizados.
- elaboração e manutenção de roteiros, critérios e recomendações para o processo de software padrão.
- publicação de roteiros, critérios e recomendações para o processo de software padrão.
- determinação e manutenção de um banco de dados de informações sobre processos de software.
- determinação e manutenção de uma biblioteca de documentos relativos ao processo de software.

3.3. Área-chave: Definição das Especializações do Processo Organizacional

Objetivo: desenvolver e manter as especializações do processo padrão da organização para cada nível de maturidade.

Recomendações:

- definição e manutenção das especializações do processo de software padrão, de acordo com um procedimento documentado.

- documentação das especializações do processo de software padrão, de acordo com os padrões definidos pela organização.
- publicação da documentação das especializações do processo de software padrão.
- documentação e manutenção dos projetos realizados e aprovados, descrevendo as especializações do processo de software utilizadas.
- elaboração e manutenção de roteiros, critérios e recomendações para especialização do processo de software padrão em diferentes níveis de maturidade.
- publicação de roteiros, critérios e recomendações para especialização do processo de software padrão em diferentes níveis de maturidade.
- determinação e manutenção de um banco de dados de especializações de processos de software.
- determinação e manutenção de uma biblioteca de documentos relativos às especializações do processo de software.

3.4. Área-chave: Programa de Treinamento

Objetivo: desenvolver a capacidade e o conhecimento das equipes de software de uma organização, considerando suas habilidades e restrições.

Recomendações:

- elaboração e manutenção de um plano de treinamento para cada equipe de software.
- desenvolvimento e revisão do plano de treinamento de acordo com um procedimento documentado.
- publicação do plano de treinamento a todas as equipes de software.
- realização do treinamento de acordo com o plano de treinamento da organização.
- elaboração e manutenção de cursos de treinamento de acordo com os padrões da organização.
- determinação da capacidade de cada equipe de software para verificar se necessitam de um treinamento.
- manutenção de informações sobre os treinamentos.
- publicação de informações resultantes dos treinamentos.

3.5. Área-chave: Gerência do Software Integrado

Objetivo: integrar atividades técnicas e gerenciais em um modelo coerente de processo de software, que corresponde ao processo padrão da organização.

Recomendações:

- determinação do processo de software de um projeto através da especificação do processo de software padrão da organização, de acordo com um procedimento documentado.
- revisão do processo de software para um determinado projeto, de acordo com um procedimento documentado.
- publicação do processo de software do projeto a todas as equipes de software.
- elaboração e revisão de um plano de desenvolvimento de software para o projeto, descrevendo a sua utilização.
- gerência do projeto de software de acordo com o processo de software definido para o projeto.
- utilização do banco de dados de processos de software para planejamento do projeto.
- gerência do tamanho do produto de software a ser desenvolvido, de acordo com um procedimento documentado.
- gerência do esforço e custos do projeto, de acordo com um procedimento documentado.
- gerência dos recursos computacionais, de acordo com um procedimento documentado.
- gerência dos aspectos críticos do cronograma do projeto, de acordo com um procedimento documentado.
- identificação, avaliação, documentação e gerência dos riscos de um projeto, de acordo com um procedimento documentado.
- realização periódica de revisões do projeto de software para determinar as ações necessárias para obter os resultados e atingir o desempenho esperado.

3.6. Área-chave: Coordenação Distribuída do Software

Objetivo: coordenar as atividades técnicas e gerenciais de cada equipe de software, seguindo o processo padrão da organização.

Recomendações:

- determinação do processo de software de cada equipe de software a partir do processo de software padrão da organização, de acordo com um procedimento documentado.
- revisão do processo de software das equipes de desenvolvimento, de acordo com um procedimento documentado.
- elaboração e revisão de um plano de desenvolvimento de software para cada equipe de software, descrevendo a sua utilização.
- coordenação das atividades das equipes de software de acordo com o processo de software definido.
- utilização do banco de dados de processos de software para planejamento e estimativa das atividades de cada equipe de software.
- coordenação do tamanho dos componentes de software (ou das mudanças dos componentes) a serem produzidos por cada equipe de software, de acordo com um procedimento documentado.
- coordenação do esforço e custos das atividades de cada equipe de software, de acordo com um procedimento documentado.
- coordenação dos recursos computacionais de cada equipe de software, de acordo com um procedimento documentado.
- coordenação dos aspectos críticos do cronograma das equipes de software, de acordo com um procedimento documentado.
- identificação, avaliação, documentação e coordenação dos riscos das atividades de cada equipe de software, de acordo com um procedimento documentado.
- realização periódica de revisões das atividades das equipes de software para determinar as ações necessárias para obter os resultados e atingir o desempenho esperado.

3.7. Área-chave: Engenharia Distribuída do Produto de Software

Objetivo: definir, integrar e realizar tarefas de engenharia de software para desenvolver os produtos de software.

Recomendações:

- integração de métodos e ferramentas de engenharia de software às especializações do processo de software do projeto.
- determinação, manutenção, documentação e verificação dos requisitos do software de acordo com o processo de software do projeto.

- determinação, manutenção, documentação e verificação dos requisitos de cada componente do software de acordo com a especificação do processo de software do projeto.

- determinação, manutenção, documentação e verificação do projeto de cada componente do software de acordo com a especialização do processo de software do projeto, respeitando os requisitos do software e identificando a estrutura de trabalho necessária à codificação.

- determinação, manutenção, documentação e verificação do código de acordo com a especialização do processo de software do projeto, respeitando os requisitos e o projeto.

- realização dos testes de cada componente do software, de acordo com a especialização do processo de software definido para o projeto.

- planejamento e realização dos testes de integração dos componentes do software de acordo com o processo de software definido para o projeto.

- planejamento e realização de testes de aceitação para demonstrar que o software integrado satisfaz os requisitos.

- publicação de todos os documentos descrevendo os requisitos, projeto, codificação, testes individuais, de integração e de aceitação a todas as equipes de software.

- elaboração e manutenção da documentação que será usada na operação e manutenção do software de acordo com o processo de software definido para o projeto.

- coleta e análise de problemas identificados nas revisões e testes de acordo com o processo de software definido para o projeto.

- publicação dos problemas identificados nas revisões e testes a todas as equipes de software.

- garantia da consistência entre os componentes de software, incluindo planos de software, descrição do processo, requisitos, projeto, código, planos e procedimentos de testes.

3.8. Área-chave: Coordenação Inter-Grupos

Objetivo: coordenar diferentes equipes de trabalho, apresentando informações sobre o trabalho em andamento.

Recomendações:

- determinação dos requisitos do software pelos engenheiros de software, usuários e outros profissionais de todas as equipes de software envolvidas no projeto.

- monitoração e coordenação de atividades técnicas por representantes das equipes de software.
- utilização de um plano de documentação para comunicar decisões e coordenar o trabalho realizado.
- identificação e negociação de dependências críticas entre os grupos de engenheiros de software, de acordo com um procedimento documentado.
- revisão de resultados produzidos por um determinado grupo pelos profissionais que os utilizarão para verificar se correspondem às suas necessidades.
- resolução de incompatibilidades entre os diferentes grupos seguem procedimentos documentados.
- publicação das incompatibilidades identificadas a todas as equipes de software.
- revisões técnicas periódicas dos trabalhos são realizadas por representantes dos grupos de engenheiros de software.
- publicação dos resultados das revisões técnicas às equipes de software.

3.9. Área-chave: Revisões Distribuídas

Objetivo: examinar sistematicamente os produtos de software para prevenir possíveis falhas.

Recomendações:

- planejamento das revisões e documentação do planejamento.
- publicação do planejamento das revisões a todas as equipes de software.
- realização de revisões, de acordo com um procedimento documentado.
- registro dos resultados das revisões.
- publicação dos resultados das revisões.

4. Nível Gerenciado

4.1. Área-chave: Gerência Quantitativa do Processo

Objetivo: controlar o desempenho do processo de software, de forma quantitativa.

Recomendações:

- elaboração de um plano para a gerência quantitativa do processo, de acordo com um procedimento documentado.

- publicação do plano para a gerência quantitativa do processo a todas as equipes de software.
- realização das atividades de gerência quantitativa do processo para um determinado projeto de software de acordo com o plano de gerência quantitativa do processo da organização.
- publicação dos resultados das atividades de gerência quantitativa do processo.
- determinação da estratégia para coleta de dados e análise quantitativa de acordo com o processo de software definido para o projeto.
- coleta de métricas para controle de projeto de acordo com um procedimento documentado.
- publicação das métricas coletadas a todas as equipes de software.
- análise e controle quantitativo do processo de software definido para um projeto de acordo com um procedimento documentado.
- preparação e distribuição de relatórios sobre a gerência quantitativa de um processo de software definido para o projeto.
- determinação e manutenção de uma base de dados sobre a capacidade do processo padrão da organização.

4.2. Área-chave: Coordenação Quantitativa das Especializações

Objetivo: controlar o desempenho das especializações do processo de software, de forma quantitativa.

Recomendações:

- elaboração de um plano para a coordenação quantitativa das especializações do processo, de acordo com um procedimento documentado.
- publicação do plano para a coordenação quantitativa das especializações do processo a todas as equipes de software.
- realização das atividades de coordenação quantitativa das especializações do processo para um determinado projeto de software de acordo com o plano de coordenação quantitativa do processo da organização.
- publicação dos resultados das atividades de coordenação quantitativa das especializações do processo.
- determinação da estratégia para coleta de dados e análise quantitativa de acordo com as especializações do processo de software definido para o projeto.

- coleta de métricas para controle das atividades das equipes de software de acordo com um procedimento documentado.
- publicação das métricas coletadas a todas as equipes de software.
- análise e controle quantitativo das especializações do processo de software definido para um projeto de acordo com um procedimento documentado.
- preparação e distribuição de relatórios sobre a coordenação quantitativa de uma especialização do processo de software definido para o projeto.
- determinação e manutenção de uma base de dados sobre a capacidade das especializações do processo.

4.3. Área-chave: Gerência da Qualidade de Software

Objetivo: definir os atributos de qualidade a serem respeitados e realizar uma análise tanto dos componentes do software quanto do produto de software.

Recomendações:

- elaboração e manutenção de um plano de qualidade de cada componente do software de acordo com um procedimento documentado.
- publicação do plano de qualidade a todas as equipes de software.
- utilização do plano de qualidade como base para as atividades de gerência da qualidade de software.
- definição, monitoração e revisão dos fatores de qualidade dos componentes de software ao longo de seu ciclo de vida.
- medição e análise da qualidade dos componentes de software e comparação com os objetivos de qualidade definidos.
- medição e análise da qualidade do produto integrado de software e comparação com os objetivos de qualidade definidos.

5. Nível Otimizado

5.1. Área-chave: Prevenção de Defeitos

Objetivo: identificar possíveis causas de defeitos, modificar as especializações do processo para prevenir erros e mudar o processo padrão para evitar falhas em projetos futuros.

Recomendações:

- elaboração e manutenção de um plano para prevenção de defeitos.

- publicação do plano para prevenção de defeitos a todas as equipes de software.
- preparação das atividades de prevenção de defeitos ao início de cada tarefa.
- realização de reuniões para analisar causas de defeitos, de acordo com um procedimento documentado.
- publicação das possíveis causas de defeitos identificadas nas reuniões às equipes de software.
- implantação de um plano de ações para prevenção de defeitos, resultante das reuniões de análise das causas dos defeitos.
- documentação e acompanhamento de dados relacionados à prevenção de defeitos.
- revisão do processo de software padrão da organização incorporando ações para prevenção de defeitos, de acordo com um procedimento documentado.
- revisão das especializações do processo de software de um projeto incorporando ações para prevenção de defeitos, de acordo com um procedimento documentado.
- divulgação dos resultados das atividades para previsão de defeitos a todas as equipes de software.

5.2. Área-chave: Gerência de Mudanças Tecnológicas

Objetivo: identificar os benefícios tecnológicos e integrá-los às especializações do processo padrão da organização.

Recomendações:

- elaboração e manutenção de um plano para gerência de mudanças tecnológicas.
- publicação do plano para gerência de mudanças tecnológicas.
- identificação das mudanças tecnológicas pelo grupo responsável acompanhando as atividades de cada equipe de software.
- identificação dos diferentes aspectos das especializações do processo de software que necessitam ou seriam beneficiados por mudanças tecnológicas.
- seleção e aquisição de tecnologias para cada equipe de software, de acordo com um procedimento documentado.
- implantação de mudanças tecnológicas em cada equipe de software.
- incorporação de novas tecnologias ao processo de software padrão, de acordo com um procedimento documentado.

- incorporação de novas tecnologias às especializações do processo de software, de acordo com um procedimento documentado.

5.3. Área-chave: Gerência da Mudança do Processo

Objetivo: melhorar continuamente o processo de software da organização.

Recomendações:

- determinação de um programa de melhoria do processo envolvendo os membros da organização.
- publicação do programa de melhoria do processo a todas as equipes de software.
- coordenação das atividades de melhoria do processo pelo grupo responsável pelas atividades do software.
- elaboração e manutenção de um plano para melhoria do processo, de acordo com um procedimento documentado.
- publicação do plano para melhoria do processo a todas as equipes de software.
- realização das atividades de melhoria do processo de acordo com o plano de melhoria do processo.
- coordenação das propostas para melhoria do processo de acordo com um procedimento documentado.
- participação ativa dos membros da organização para a melhoria do processo.
- implantação inicial das melhorias do processo para determinar seus benefícios antes de serem introduzidas no processo padrão da organização.
- implantação das melhorias no processo padrão da organização de acordo com um procedimento documentado.
- publicação das melhorias no processo padrão a todas as equipes de software.
- manutenção de registros sobre as atividades de melhoria dos processos.
- divulgação dos resultados obtidos com a melhoria do processo às equipes de software.

5.4. Área-chave: Gerência de Mudanças nas Especializações

Objetivo: melhorar continuamente as especializações do processo de software da organização.

Recomendações:

- determinação de um programa de melhoria das especializações do processo envolvendo os membros das equipes de software.
- publicação do programa de melhoria das especializações do processo a todas as equipes de software.
- coordenação das atividades de melhoria das especializações do processo pelo grupo responsável pelas atividades do software.
- elaboração e manutenção de um plano para melhoria das especializações do processo, de acordo com um procedimento documentado.
- publicação do plano para melhoria das especializações do processo a todas as equipes de software.
- realização das atividades de melhoria das especializações do processo, de acordo com o plano de melhoria do processo.
- coordenação das propostas para melhoria das especializações do processo de acordo com um procedimento documentado.
- participação ativa dos membros da organização para a melhoria das especializações do processo.
- implantação inicial das melhorias das especializações do processo para determinar seus benefícios antes de serem introduzidas no processo padrão da organização.
- implantação das melhorias nas especializações do processo padrão da organização, de acordo com um procedimento documentado.
- publicação das melhorias das especializações do processo padrão a todas as equipes de software.
- manutenção de registros sobre as atividades de melhoria das especializações do processo.
- divulgação dos resultados obtidos com a melhoria das especializações do processo às equipes de software.

V.4. Aplicação do Modelo

Nesta seção, descrevem-se as etapas para a aplicação do modelo de gerência de processos de software para equipes geograficamente distribuídas: determinação da capacidade, instanciação e acompanhamento do processo especializado, avaliação dos

processos especializados, melhoria da capacidade dos grupos de trabalho e melhoria do processo padrão.

V.4.1. Determinação da Capacidade

O primeiro processo de avaliação objetiva identificar o nível de maturidade de cada equipe de software. Através desta identificação, é possível determinar a especialização de processo mais adequada ao grupo. A etapa de determinar a capacidade de um grupo de trabalho só é realizada na primeira vez que o modelo de gerência de processos é aplicado ou quando um novo grupo de trabalho é formado.

A primeira atividade corresponde à formação de uma equipe que irá analisar os dados de projetos anteriores, avaliar documentos, aplicar os questionários, e entrevistar gerentes, líderes de projeto e profissionais, de acordo com o método CBA-IPI (*CMMSM-Based Appraisal for Internal Process Improvement*) da SEI (DUNAWAY e MASTERS, 1996). A equipe é composta por integrantes da área gerencial, os quais são escolhidos a partir do modelo organizacional, e de profissionais de software do grupo a ser avaliado.

Os objetivos desta etapa são explicitados e o plano de avaliação é elaborado. Cronogramas detalhados e a lista de documentos a serem examinados também fazem parte do plano de avaliação. Um conjunto inicial de documentos sobre a equipe e seus projetos passados são revisados para entender o seu processo de software, identificar pontos a serem melhorados e mapear os dados às especializações de processos da organização.

A avaliação é realizada a partir de dados coletados em projetos anteriores e através de questionários, revisões de documentos, apresentações e entrevistas com o grupo de software (FERGUSON *et al.*, 1997). O questionário é organizado pelas áreas-chave de processo do CMM. O questionário é respondido por profissionais previamente selecionados, que de preferência, devem estar relacionados a diferentes atividades e responsabilidades do processo de software. As respostas podem ser: *sim*, *não*, *não se aplica*, *não sei*. É também importante que comentários sejam feitos. As informações coletadas são utilizadas para se obter uma visão geral da

capacidade de uma equipe para desenvolver sistemas computacionais.

Posteriormente, em conjunto com outros dados adquiridos através de entrevistas e da revisão de documentos, é que a capacidade poderá ser determinada. A proposta da entrevista é elucidar eventuais dúvidas, entender como o trabalho é realizado, o processo utilizado e identificar as áreas que podem ser melhoradas. Ao final, o resultado da avaliação é documentado e apresentado.

V.4.2. Especialização do Processo Padrão

O processo de especialização inicia-se com a determinação da correspondência entre os diferentes aspectos do processo padrão e as áreas-chave do modelo de maturidade de referência. O objetivo é identificar que aspectos do processo de software devem ter maior atenção para atender a determinadas áreas-chave de um nível de maturidade. Por exemplo, a área-chave "Acompanhamento e Supervisão do Projeto" do nível Repetitivo (2º nível de maturidade) requer que a organização introduza esforços nos processos de Gerência e de Coordenação dos processos organizacionais do processo padrão. Além disso, para que esta área-chave seja atendida, também é importante que a organização tenha uma atenção maior em determinadas atividades dos processos primários e organizacionais.

A principal meta do processo de especialização é permitir que os gerentes e coordenadores das equipes distribuídas se abstraiam de determinados aspectos do processo padrão, concentrando-se nos aspectos específicos que permitirão atender as área-chave de um nível de maturidade. Desta forma, os aspectos do processo para atender todas as áreas-chave de um nível do modelo de maturidade são encapsulados para determinar a correspondência com os níveis de maturidade. A figura V.2 representa tal conceito.

A seguir, apresentamos a relação de cada área-chave do modelo de maturidade de processos para equipes geograficamente distribuídas com as atividades dos processos do processo padrão que melhor atendem os seus objetivos:

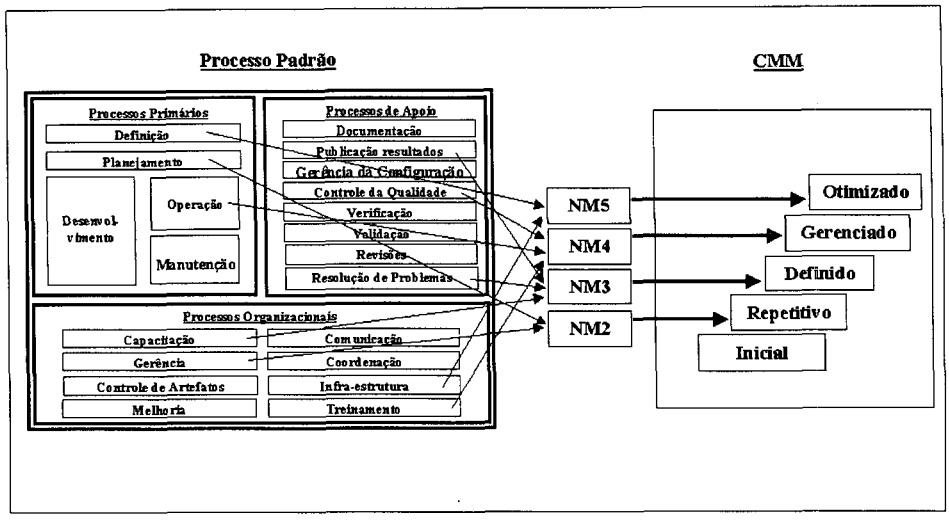


Figura V.2: Representação Esquemática da Especialização do Processo Padrão.

1. **Nível Inicial**

Não há áreas-chave.

2. **Nível Repetitivo**

2.1. **Gerência Distribuída de Requisitos:** os processos de definição e de planejamento (primários) contém atividades referentes à definição e análise dos requisitos. O processo de controle da qualidade (apoio) reúne atividades para garantir, ao longo do seu desenvolvimento, que os produtos de software respeitam os requisitos.

2.2. **Planejamento Distribuído do Projeto de Software:** o processo de planejamento (primário) se refere diretamente a esta área-chave. O processo de gerência (organizacional) determina que a gerência do projeto prepare um plano para a execução de um processo, contendo a descrição das atividades e dos produtos de software a serem realizados.

2.3. **Acompanhamento e Supervisão Distribuídos do Projeto:** as atividades de execução e controle dos processo de gerência e coordenação (organizacionais) determinam a execução do processo e o seu acompanhamento. Os processos de desenvolvimento, operação e manutenção definem as atividades a serem realizadas no projeto. Para completar tal atividade, necessita-se incorporar as tarefas descritas nos processos de publicação de resultados e resolução de problemas.

2.4. **Gerência da Delegação de Atividades do Software:** a atividade denominada "alocação das atividades às equipes e membros do processo de planejamento" (primário) se refere diretamente a esta área-chave. O processo de gerência (organizacional) determina que a gerência do projeto prepare um plano contendo a alocação de tarefas e a determinação de responsabilidades.

2.5. **Garantia da Qualidade de Software:** o processo padrão possui um processo específico para garantir que os produtos e os processos de software respeitam os requisitos e são coerentes com os respectivos planos. O controle da qualidade utiliza resultados dos processos de verificação, validação e resolução de problemas (processos de apoio).

2.6. **Gerência Distribuída de Configuração de Software:** o processo padrão possui um processo específico para gerenciar, de forma distribuída, a configuração do software. No processo de planejamento (primário), identificam-se os componentes do software, a partir de uma especificação inicial do sistema a ser desenvolvido. No processo de controle de artefatos, controla-se a produção e a integração de artefatos.

2.7. **Gerência da Comunicação:** o processo de comunicação (organizacional) se refere diretamente a esta área-chave. A gerência da comunicação apóia os processos de resolução de problemas e de publicação dos resultados e é apoiada pelo processo de documentação (apoio).

2.8. **Gerência da Capacidade das Equipes:** o processo de capacitação (organizacional) se refere diretamente a esta área-chave. O processo de planejamento (primário) contém atividades para delegar atividades com base na capacitação de cada equipe. Os processos de gerência e coordenação (organizacionais) reúnem atividades para gerenciar um determinado processo e coordenar as equipes de trabalho.

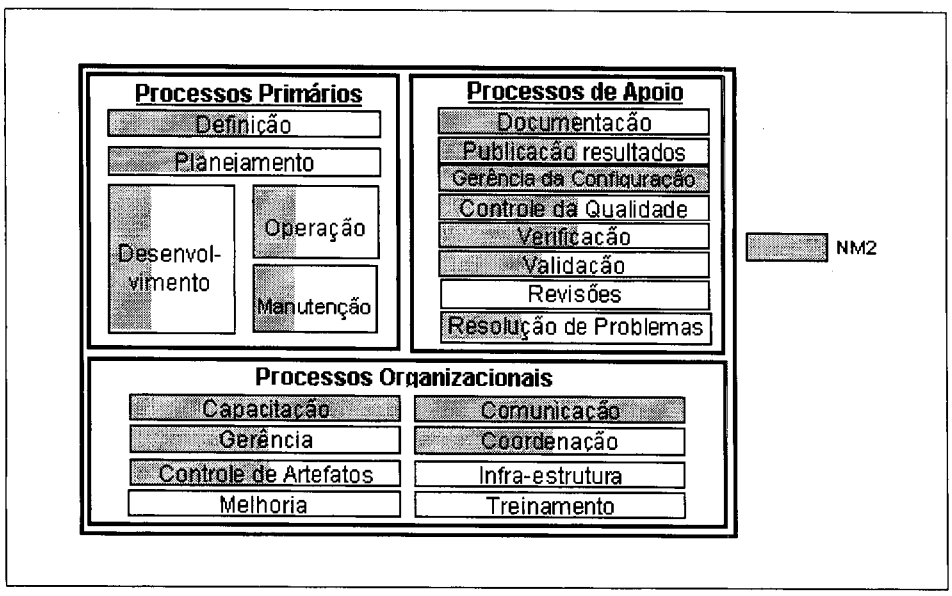


Figura V.3: Modelo de Referência do Processo Padrão para o 2º Nível de Maturidade.

3. **Nível Definido**

3.1. **Foco nos Processos da Organização:** o processo de melhoria (organizacional) contém atividades para definir, avaliar e melhorar o processo de software de uma organização.

3.2. **Definição do Processo Padrão da Organização:** o processo de melhoria (organizacional) apresenta a atividade para determinar o processo da organização. Os processos de desenvolvimento, operação e manutenção (primários) realizam as atividades definidas na definição do processo padrão.

3.3. **Definição das Especializações do Processo Organizacional:** nenhuma atividade tem correspondência direta com esta área-chave. Entretanto, a definição das especializações do processo organizacional é definida através do modelo de gerência de processos para equipes geograficamente dispersas.

3.4. **Programa de Treinamento:** o processo padrão possui um processo específico para aprimorar o conhecimento das equipes de trabalho. O processo de planejamento (primário) determina o treinamento necessário para a construção do sistema.

3.5. **Gerência do Software Integrado:** o processo de desenvolvimento (primário) contém atividades referentes à integração de artefatos. Os processos de verificação e validação (apoio) contém atividades para verificar se os artefatos de uma atividade respeitam os requisitos ou condições impostos em atividades anteriores e para validar se o produto final (integrado) atingiu seus objetivos. O processo de controle de artefatos (organizacional) reúne atividades para controlar a produção e a integração dos artefatos do software.

3.6. **Coordenação Distribuída do Software:** os processos de gerência e coordenação (apoio) reúnem atividades para gerenciar um projeto de software e coordenar localmente cada equipe de trabalho.

3.7. **Engenharia Distribuída do Produto de Software:** os processos de desenvolvimento, operação e manutenção (primários) contém atividades que definem as tarefas de engenharia de software, incluindo especificação, projeto, implementação, testes e aprovação de um produto de software.

3.8. **Coordenação Inter-Grupos:** os processos de desenvolvimento (primário) e resolução de problemas (apoio) reúnem atividades conjuntas entre as equipes de software. Esta área-chave é apoiada pelo processo de publicação de resultados (apoio).

3.9. **Revisões Distribuídas:** o processo de revisão (apoio) se refere diretamente a esta área-chave. Os processos de definição, planejamento, desenvolvimento, operação e manutenção (primários) contém atividades de revisões dos produtos gerados por estes processos.

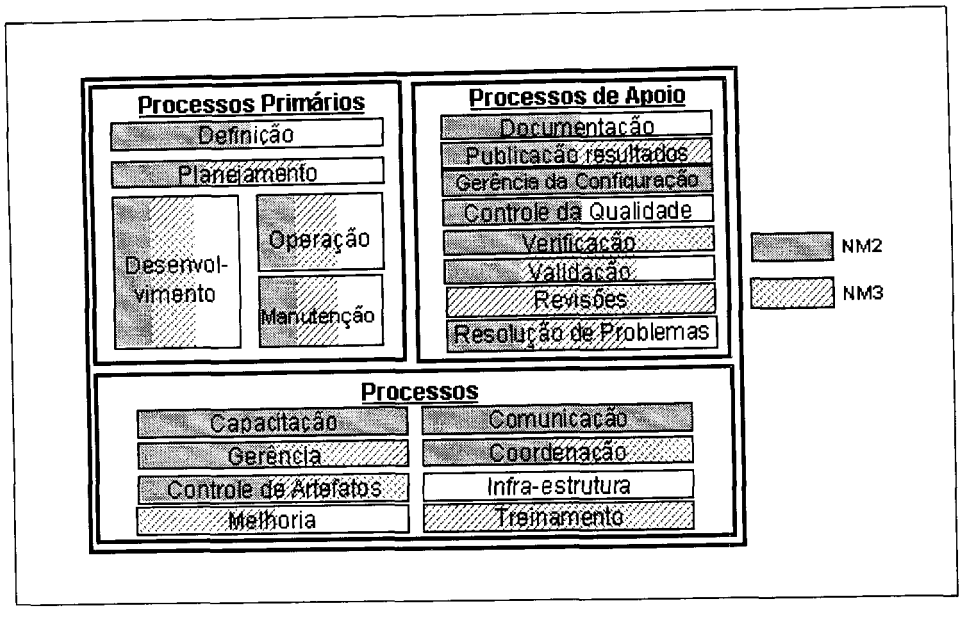


Figura V.4: Modelo de Referência do Processo Padrão para o 3° Nível de Maturidade.

4. **Nível Gerenciado**

4.1. **Gerência Quantitativa do Processo:** o processo de planejamento (primário) e controle da qualidade (apoio) contém atividades para definir os dados a serem coletados durante um projeto de software. Durante os processos de desenvolvimento, operação e manutenção (primários) os dados são coletados. As informações são documentadas pelo processo de documentação (apoio). O processo de melhoria (organizacional) utiliza os dados coletados anteriormente para fazer a análise do processo de software.

4.2. **Coordenação Quantitativa das Especializações:** é apoiada pelos mesmos processos descritos na área-chave "Gerência Quantitativa do Processo", sendo que os dados coletados referem-se às especificações do processo padrão.

4.3. **Gerência da Qualidade de Software:** os processos de controle da qualidade e validação (apoio) contém atividades para gerenciar a qualidade do software. O processo de melhoria (organizacional) reúne atividades para aprimorar o processo de software e, portanto, do produto de software.

5. **Nível Otimizado**

5.1. **Prevenção de Defeitos:** os processos de resolução de problemas (apoio) e de melhoria (organizacional) contém atividades para identificar as causas de imperfeições no software.

5.2. **Gerência da Mudança Tecnológica:** o processos de infra-estrutura (organizacional) estabelece e mantém equipamentos, sistemas computacionais, ferramentas, técnicas e padrões necessários para o desenvolvimento de software.

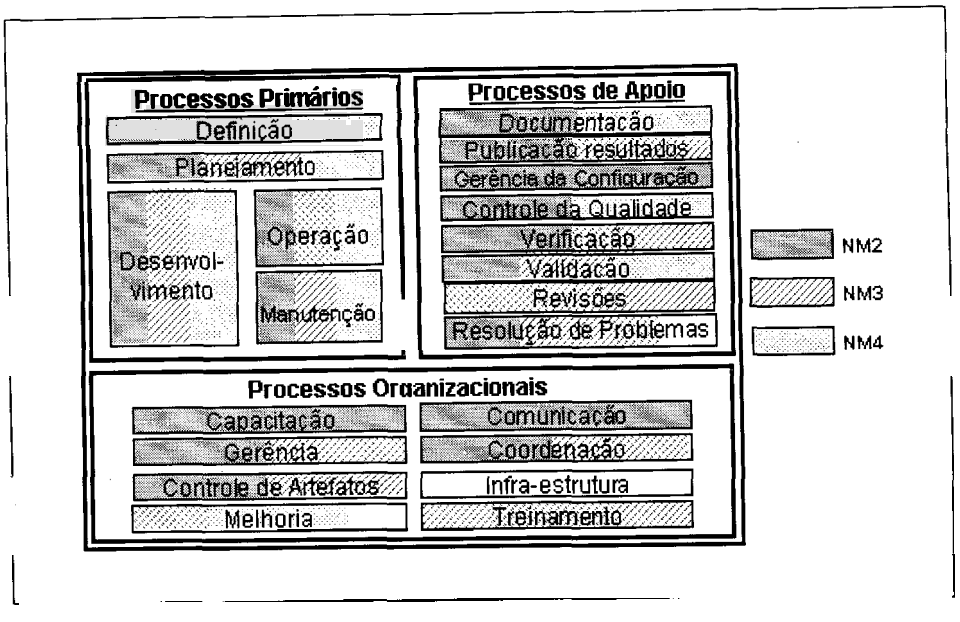


Figura V.5: Modelo de Referência do Processo Padrão para o 4° Nível de Maturidade.

O processo de melhoria (organizacional) contém atividades para aprimorar todos os aspectos do processo, inclusive o processo de infra-estrutura.

5.3. **Gerência da Mudança do Processo:** o processo de melhoria (organizacional) contém atividades referentes ao aprimoramento do processo padrão, o que implica em mudanças na sua definição.

5.4. **Gerência da Mudança das Especializações:** é apoiada pelo mesmo processo descrito na área-chave "Gerência da Mudança do Processo", sendo que as mudanças referem-se às especificações do processo padrão.

V.4.3. Geração das Especializações

A especialização do processo padrão em um determinado nível de maturidade é gerada a partir da exclusão das atividades relacionadas aos níveis de maturidade superiores. Por exemplo, a especialização do segundo nível de maturidade não contém as atividades que se referem ao terceiro, quarto e quinto níveis de maturidade. Portanto, a especialização do segundo nível de maturidade incorpora exclusivamente as atividades referentes a todas as áreas-chave do nível repetitivo. A especialização do terceiro nível de maturidade incorpora tanto as atividades referentes a todas as áreas-chave do nível repetitivo quanto do nível definido. A especialização do quarto nível de maturidade incorpora as atividades referentes a todas as áreas-chave

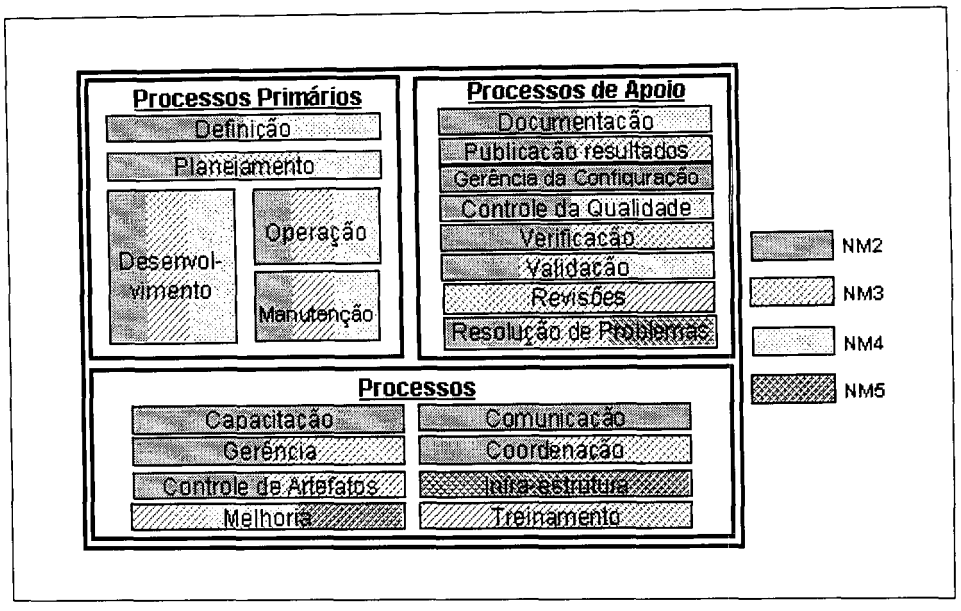


Figura V.6: Modelo de Referência do Processo Padrão para o 5^o Nível de Maturidade.

dos níveis repetitivo, definido e gerenciado. Pode-se, então, concluir que o processo padrão incorpora todas as atividades necessárias para o nível otimizado.

As especializações determinam, através da identificação de atividades, a capacidade mínima que um processo deve ter para respeitar os requisitos de um determinado nível de maturidade. A execução da especialização do processo padrão é que vai determinar a sua maturidade.

V.4.4. Instanciação e Acompanhamento do Processo Especializado

As especializações do processo padrão orientam a construção de sistemas computacionais, definindo suas atividades e produtos intermediários. O processo de instanciação se refere à alocação de métodos, técnicas de engenharia de software e recursos humanos e tecnológicos para cada atividade. A coordenação local identifica as restrições dos grupos de trabalho e aloca profissionais para cada atividade do projeto. A gerência do projeto deve garantir que todas as atividades sejam realizadas, que as recomendações sejam cumpridas e que os produtos intermediários sejam corretamente produzidos.

Durante o acompanhamento do projeto, as informações manipuladas ao longo do processo de desenvolvimento devem ser devidamente coordenadas e estruturadas. A

comunicação entre as equipes e entre integrantes de uma mesma equipe deve ser apoiada e registrada. As tarefas realizadas e as decisões tomadas são documentadas e facilmente acessadas por todos aqueles envolvidos no projeto de software. Alguns dados são coletados, armazenados e organizados para posterior análise e, então, identificação de falhas no processo. Define-se uma estrutura de trabalho que realize a referência cruzada dos documentos gerados em todas as atividades de um projeto. O documento que descreve os requisitos do usuário é conectado com o projeto do sistema; o código, com a especificação. Assim, o engenheiro de software pode navegar através dos diferentes níveis de abstração de um componente da aplicação em desenvolvimento. Neste contexto, a representação de um componente do software pode ser textual ou gráfica, horizontal (mesmo níveis de abstração) e vertical (da especificação até sua implementação). Seguindo as conexões horizontais, pode-se verificar as interações de um componente com outros. Ao seguir as conexões verticais, pode-se seguir desde a especificação inicial do sistema, até a implementação de um determinado componente.

O processo de acompanhamento deve ser apoiado por um conjunto de mecanismos que, ao integrar os resultados de cada etapa do projeto de software, permite um melhor entendimento dos componentes do sistema em construção. O engenheiro de software pode percorrer as informações do sistema certificando-se que as interações, em um determinado nível de abstração, respeitam os requisitos e restrições impostos em um nível de abstração mais elevado. Durante a manutenção do sistema, pode-se verificar se a mudança em um componente influi em outros. O gerente de projeto pode acessar as informações interligadas para confirmar se todos os requisitos do software foram respeitados. As informações inter-conectadas de uma aplicação podem ser utilizadas como exemplo durante o desenvolvimento de outras aplicações, ou seja, os projetistas podem acessar todas as referências de um componente, identificando possíveis candidatos à reutilização. Nos estágios iniciais de um projeto, as restrições, decisões e a estratégia de desenvolvimento devem ser lembradas nas etapas seguintes.

O acompanhamento de um projeto pode ser estruturado nas seguintes etapas:

definição e execução. A definição do projeto se refere à escolha da equipe de trabalho, dos seus coordenadores e à identificação das responsabilidades. Na execução do projeto, as atividades são realizadas. A seguir, os principais mecanismos de apoio ao acompanhamento de projetos de software realizados por equipes geograficamente dispersas são apresentados (GRUNDY *et al.*, 1995, AVRILIONIS *et al.*, 1996c, INGHAM *et al.*, 1997, SAKAMOTO *et al.*, 1998, MAIDANTCHIK *et al.*, 1998b, MAURER e KAISER, 1998, GRUNDY *et al.*, 1998).

Configuração

Durante a configuração do processo, as fases, atividades e produtos do projeto de software são descritos, seguindo o processo especializado escolhido. Determinam-se os métodos, técnicas e ferramentas a serem utilizados.

Registro de Projetos

A atividade inicial de um projeto corresponde ao registro de suas informações: nome do projeto, descrição, data e identificação das equipes, gerência, coordenadores e membros das equipes. Define-se a estrutura necessária para iniciar o projeto, determinando as bases de dados para armazenamento das informações do projeto. O registro do projeto é realizado pela gerência.

Membros da Equipe de Trabalho

O desenvolvimento disperso de aplicações computacionais é realizado através de esforços cooperativos entre as equipes de trabalho, que devem manter uma base de informações ativa com o objetivo de melhorar a sua comunicação e de coordenar suas atividades. Para cada membro das equipes de software, deve-se registrar seu nome, endereço eletrônico (para apoiar uma comunicação entre os profissionais), responsabilidades e tarefas designadas.

Registro das Atividades do Projeto

Durante o projeto do software, a execução de todas as atividades deve ser registrada.

As informações e respectivas associações são organizadas para facilitar a sua procura, atualização e apresentação. A organização das informações sobre a execução das atividades deve respeitar a configuração do projeto do software.

Componentes do Sistema

Um componente do sistema corresponde a uma frase em linguagem natural no documento da especificação do software, a um gráfico resultante do projeto e linhas de código, durante a implementação. Cada componente é associado a sua descrição, tipo e representações. A descrição do componente facilita a sua procura. O tipo (texto, gráfico, som, etc.) permite a associação do componente com o mecanismo adequado para sua apresentação. Cada representação identifica a atividade do projeto que gerou o componente. Os componentes, sua descrição e representações podem ser reutilizados em projetos futuros. As associações entre os componentes também facilitam a manutenção do software.

Decisões e Restrições do Projeto

As decisões e restrições do projeto devem ser comunicadas às equipes de trabalho com o objetivo de coordenar suas atividades. Mecanismos que registram tais informações devem ser estruturados e indexados de forma que qualquer dado possa ser facilmente recuperado e entendido por profissionais, mesmo aqueles que não participaram do desenvolvimento do software. As decisões e restrições do projeto podem apoiar futuras manutenções.

Controle de Acesso

O controle de acesso às informações de um projeto de software é necessário para definir os dados que podem ser acessados ou modificados somente por determinados profissionais ou classes de profissionais. Em ambientes multi-autoria, deve-se implementar sistema um controle de concorrência, evitando que mais de um autor possa modificar um mesmo documento simultaneamente.

Gerência das Informações

Em cada documento do software, podem existir diferentes representações de uma mesma informação. Também podem existir diferentes níveis semânticos de uma mesma representação. Portanto, os diferentes tipos de dados podem ser associados a uma identificação lógica do seu conteúdo e a uma descrição mais detalhada da sua representação. Diversos mecanismos são importantes para oferecer às equipes um amplo controle no processo de criação, associação e busca às informações relacionadas ao projeto de software, tais como: controle de versões, permitindo que anotações sejam criadas e recuperadas de acordo com a versão apropriada dos documentos aos quais se referem, sistema de aviso, que informam mudanças efetuadas ou apenas as anotações de interesse a determinados usuários.

Apresentação das Informações

O processo de busca de determinada informação pode ser muito dispersivo, impedindo que o profissional se concentre no entendimento da informação. Para minimizar tal problema, devem-se oferecer mecanismos para a apresentação das informações, tais como mostrar diferentes níveis de detalhe de uma informação e permitir o acesso às diferentes visões de um mesmo conjunto de dados. Por exemplo, a cada etapa do projeto de software, mais detalhes sobre uma mesma informação são apresentados. No que se refere à apresentação dos diferentes aspectos de um dado, as mensagens eletrônicas armazenadas podem ser recuperadas através da ordem cronológica, na qual foram enviadas, ou do nome do membro da equipe de trabalho que a enviou. Tais mecanismos oferecem uma maneira mais eficiente de indexar informações pois enfatizam relações semânticas ou invés de relações físicas.

Sistema de Busca

Cada componente ou documento do software pode estar associado a uma descrição, contendo palavras-chave. O sistema de busca deve integrar mecanismos de recuperação e de navegação dos componentes de um projeto de software, permitindo que o profissional encontre a informação desejada através de refinamentos sucessivos. Ao

procurar por uma determinada informação, o sistema de busca deve recuperar tanto o dado desejado, como todos os documentos que se referem à informação desejada. A implementação de sistemas de busca envolvem duas fases: indexação e recuperação de informações. Os índices possibilitam a procura a uma determinada informação em servidores indexados e remotos através da combinação de palavras-chave.

Sistema de Comunicação

Para apoiar atividades cooperativas de maneira eficiente, deve-se permitir a comunicação entre os membros e a conexão aos documentos referenciados na discussão. Correio-eletrônico e sistemas de notícias são ferramentas de suporte à comunicação assíncrona, mas não oferecem um mecanismo para discussões e acesso a documentos compartilhados. O sistema de comunicação deve suportar a criação, apresentação, visualização e controle de informações, que são acessadas em conjunto com os documentos correspondentes, mas que podem ser armazenadas separadamente. O sistema de comunicação de um projeto de software deve permitir o armazenamento de mensagens eletrônicas e a sua conexão com o registro de produtos do software.

Gerência da Configuração

Os itens do software, cuja configuração deve ser controlada, podem ser organizados em uma coleção hierárquica de diretórios e arquivos, os quais podem ser combinados para formar uma versão do software. Uma cópia dos arquivos é mantida em um repositório principal e mantém uma cópia independente para cada membro da equipe de software. É importante separar a versão de trabalho e a versão pública de cada item do software. Desta forma, os desenvolvedores podem trabalhar com suas próprias cópias. Antes de inserir uma determinada cópia do trabalho no repositório principal, alterando a versão atual, o desenvolvedor pode comparar e concatenar os seus arquivos com os arquivos armazenados no repositório principal. Este mecanismo permite um paralelismo das atividades de desenvolvimento do software.

Controle de Inconsistências

Ao longo de um projeto de software, algumas inconsistências podem surgir, tais como atividades não cadastradas, componentes do sistemas sem representações em todas as etapas do desenvolvimento e informações não registradas. A gerência e os coordenadores das equipes devem identificar periodicamente as inconsistências do projeto e, então, executar tarefas, evitando a sua propagação.

Geração da Documentação do Projeto

A gerência de um projeto de software é responsável pela produção e controle de vários documentos produzidos por diferentes profissionais, trabalhando em vários lugares e utilizando variadas ferramentas computacionais. A utilização efetiva destes documentos é dificultada pela sua quantidade e tamanho, complexidade e produção dispersa. Portanto, é necessário gerar uma única documentação referente a todos os aspectos do projeto de software.

V.4.5. Avaliação dos Processos Especializados

A avaliação do modelo de gerência define todos os procedimentos a serem seguidos para identificar melhorias e fraquezas tanto nos processos de software de cada grupo de trabalho quanto no processo de software padrão. A avaliação é baseada no CBA-IPI (DUNAWAY e MASTERS, 1996, FERGUSON *et al.*, 1997). Suas metas são: identificar o nível de maturidade de cada grupo de trabalho, apontar deficiências e indicar atividades para a melhoria de cada grupo de trabalho, e, aprimorar o processo de software padrão.

Os requisitos mínimos para o processo de avaliação correspondem à identificação da equipe de avaliação, plano de avaliação, coleção de dados a serem avaliados, validação dos dados, classificação e divulgação dos resultados. A avaliação compreende as seguintes etapas:

1. Planejar a avaliação: organizar a equipe, formular o questionário, identificar os objetivos, definir o procedimento de avaliação, oferecer um guia a ser seguido pela equipe.

2. Conduzir a avaliação: questionário, entrevistas, medições e observações do acompanhamento, publicação dos resultados para consolidar os dados.

3. Relatar os resultados: diagnóstico através de diagramas e determinação das recomendações para aprimorar o processo de cada grupo e do processo padrão.

A avaliação é realizada de forma distribuída, ou seja, uma avaliação para cada grupo de trabalho e, posteriormente, as informações são integradas para permitir a avaliação do processo de software da organização.

V.4.6. Melhoria da Capacidade dos Grupos de Trabalho

O desempenho de cada projeto é avaliado, atribuindo uma nota (0-4) para cada área-chave de processo. O conceito de uma determinada KPA (C_{KPA}) corresponde à média de todas as suas notas: $\frac{\sum_{i=1}^{N}}{N}$.

Uma média abaixo de 3 é considerada insatisfatória. O nível de maturidade é definido como o nível onde todas as KPA correspondentes são consideradas satisfatórias, ou seja, a média deve ser igual ou superior a 3.

O formulário de avaliação, apresentado na tabela V.2, permite identificar, para cada equipe de trabalho, o grau de atendimento das áreas-chave para cada nível de maturidade do processo de software.

O formulário de avaliação é preenchido pela equipe de avaliação, após cada projeto, observando as informações registradas durante a etapa de acompanhamento. Os objetivos são: avaliar se a instância da especialização do processo padrão foi corretamente utilizada, identificar áreas-chave deficientes e confirmar a melhoria da capacidade de determinadas equipes de trabalho. Atividades incompletas ou realizadas de forma incorreta indicam potenciais problemas durante a execução da instância do processo. Médias abaixo de 3 indicam as KPAs que necessitam ter maior atenção. Caso todas as médias sejam superiores a 3, a equipe de trabalho pode passar para a especialização de nível de maturidade seguinte.

Nível CMM	Áreas-Chave de Processo CMM	Grau de Atendimento				
		Ruim	Baixo	Regular	Bom	Ótimo
2	Gerência Distribuída de Requisitos					
2	Planejamento Distribuído do Projeto de Software					
2	Acompanhamento e Supervisão Distribuídos do Projeto					
2	Gerência da Delegação de Atividades do Software					
2	Garantia da Qualidade de Software					
2	Gerência Distribuída de Configuração de Software					
2	Gerência da Comunicação					
2	Gerência da Capacidade das Equipes					
3	Foco nos Processos da Organização					
3	Definição do Processo Padrão da Organização					
3	Definição das Especializações do Processo Organizacional					
3	Programa de Treinamento					
3	Gerência do Software Integrado					
3	Coordenação Distribuída do Software					
3	Engenharia Distribuída do Produto de Software					
3	Coordenação Inter-Grupos					
3	Revisões Distribuídas					
4	Gerência Quantitativa do Processo					
4	Coordenação Quantitativa das Especializações					
4	Gerência da Qualidade de Software					
5	Prevenção de Defeitos					
5	Gerência da Mudança Tecnológica					
5	Gerência da Mudança do Processo					
5	Gerência da Mudança das Especializações					

Tabela V.2: Formulário de Avaliação do Grau de Atendimento para cada Área-Chave.

A correspondência entre os diferentes aspectos do processo padrão e as áreas-chave do modelo de maturidade, definidos através da especialização (seção V.4.2.), permite identificar as categorias do processo padrão mais deficientes a partir da avaliação realizada. Os resultados são representados graficamente através de um diagramas de barra, onde as mais baixas indicam as categorias do processo padrão que necessitam de maior atenção (DIAZ e SLIGO, 1997, PAULK *et al.*, 1997, HERBSLEB *et al.*, 1997). A figura V.7 apresenta um exemplo do resultado de uma análise qualitativa dos processos primários.

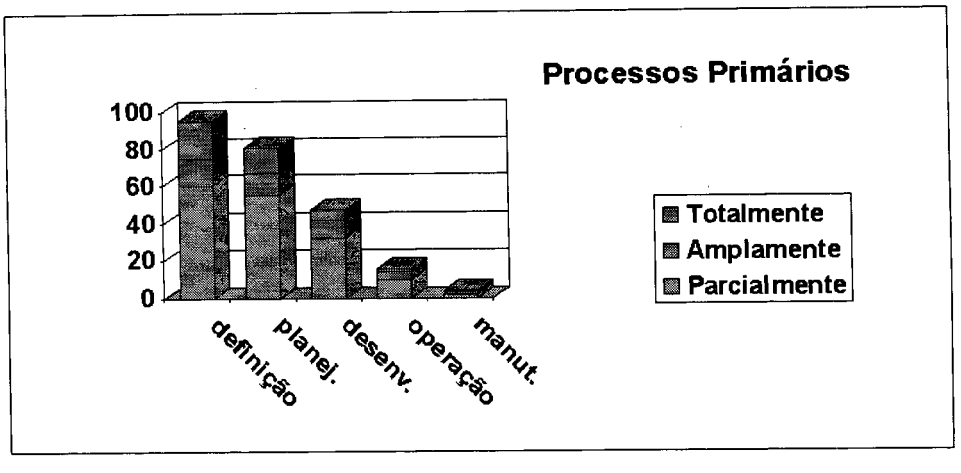


Figura V.7: Análise Qualitativa dos Processos Primários.

O eixo-x representa os processos primários, que correspondem à aquisição, contratação, desenvolvimento, operação e manutenção do software. O eixo-y representa o percentual de adequação para cada aspecto do processo primário, categorizado em totalmente, amplamente ou parcialmente satisfatório. O resultado final da análise para melhorar a capacidade dos grupos de trabalho apresenta três gráficos, que correspondem aos processos primários, organizacionais e de apoio.

V.4.7. Melhoria do Processo Padrão

Após avaliar todas as equipes de software, podem-se identificar problemas comuns em todos os projetos. Ou seja, KPAs de valores baixos presentes em todos os grupos indicam que o próprio processo padrão e, por conseqüência, as especializações precisam ser aprimoradas. Tal deficiência pode ser resultante de diversos fatores: falta de entendimento da área-chave, necessidade de treinamento ou suporte técnico inadequado.

O progresso de uma organização, ao implementar um nível CMM, pode ser representado por diagramas de Kiviat (DASKALANTONAKIS, 1997, HOLLENBACH *et al.*, 1997), como exemplificado na figura V.8. Cada reta que se inicia no centro do círculo corresponde a uma área-chave de processo a um determinado nível. O exemplo indica o progresso alcançado durante um período para evoluir do nível 1 até o nível 2. O diagrama também indica tanto as áreas-chave de processo no nível 2 que precisam de maior atenção, como também aquelas cujo esforço impresso já tenha

sido recompensado.

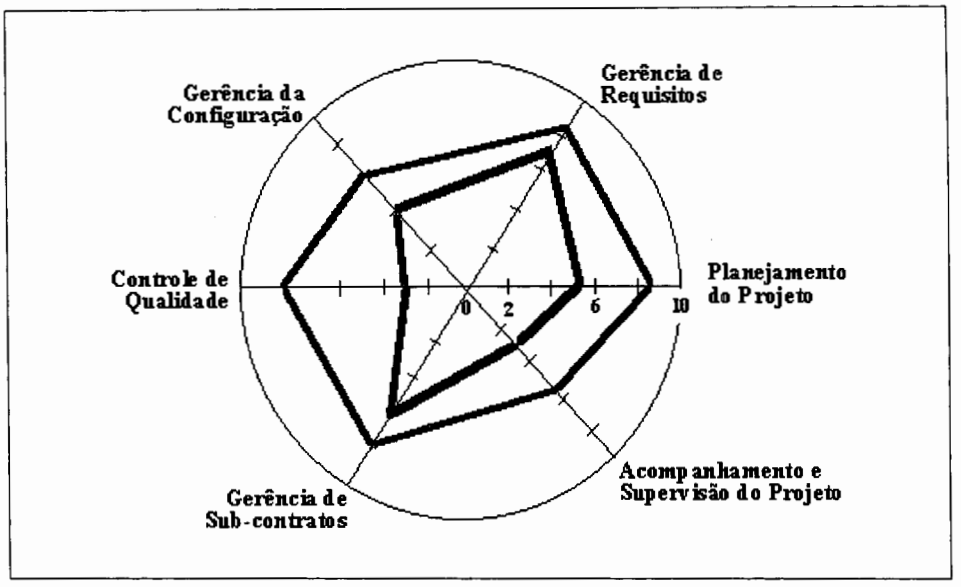


Figura V.8: Progresso de uma Organização no 2° Nível de Maturidade.

Uma organização somente será madura, quando o processo padrão corresponder ao quinto nível de maturidade. Inicialmente, o processo padrão corresponde ao nível de maturidade 1, especializando-o para os níveis 2, 3, 4 e 5. Depois, não existirão grupos no nível de maturidade inicial e o processo padrão corresponderá ao nível de maturidade 2, especializando-o para os níveis 3, 4 5. E assim, sucessivamente, até que a única especialização corresponda ao próprio processo padrão.

V.5. Considerações Finais

Este capítulo apresentou o modelo de gerência de processos de software para equipes geograficamente dispersas, cujo objetivo é apoiar a produção, realizada por grupos de trabalho distribuídos em diferentes localizações, de sistemas computacionais de qualidade, melhorando sistematicamente a capacidade de uma organização em produzir tais sistemas. O modelo propõe que o processo de software padrão, baseado nas normas ISO/IEC 12207 e adequado às equipes geograficamente dispersas, seja especializado em diferentes níveis de maturidade. O objetivo é permitir que as equipes de software desenvolvam suas atividades obedecendo ao processo da organização mesmo que possuam diferentes capacidades e, portanto, cada equipe

utilizará a especialização do processo padrão mais adequada às suas características. A especialização do processo padrão necessita de um modelo de maturidade de referência. O modelo escolhido foi o CMM, o qual foi adaptado para as necessidades do desenvolvimento disperso de software. O processo padrão é, então, especializado com base no modelo de maturidade apresentado.

A primeira fase da aplicação do modelo de gerência corresponde à determinação da capacidade de cada grupo de trabalho, que permite identificar a especialização do processo padrão a ser utilizada. Posteriormente, recursos humanos e tecnológicos devem ser alocados às atividades do processo especializado, o que é denominado instanciação do processo especializado. A utilização da instância da especialização deve ser acompanhada, estruturando e registrando as informações técnicas e gerenciais relacionadas à construção do software. Os dados coletados são utilizados, durante a fase de avaliação, para identificar as áreas-chave que precisam de maior atenção. Como o modelo de gerência define a correspondência entre as áreas-chave do modelo de maturidade e as categorias do processo padrão, a avaliação também permite identificar que aspectos do processo podem ser aperfeiçoados. Portanto, o resultado da avaliação apóia tanto a melhoria da capacidade de cada equipe como a melhoria do próprio processo padrão.

O modelo proposto promove uma melhoria contínua de processos de software para equipes geograficamente dispersas através dos quatro níveis de gerência apresentados: definição, acompanhamento, avaliação e melhoria. O último nível (melhoria) implica na redefinição do processo padrão, que é, então, monitorado, avaliado e aprimorado, formando uma sucessão de estados cíclicos e contínuos.

VI. HyperDev: Acompanhamento do Modelo de Gerência

Este capítulo apresenta o sistema HyperDev, que apóia o acompanhamento do modelo de gerência de processos de software para equipes geograficamente dispersas. São apresentados seus objetivos e detalhes de sua construção. Descreve-se a sua funcionalidade e, ao final, compara-se o sistema com outros sistemas hipertextuais de apoio ao desenvolvimento de software.

VI.1. Objetivos Gerais

O HyperDev foi implementado para apoiar o acompanhamento do modelo de gerência de processos de software para equipes geograficamente dispersas. O sistema permite a configuração do processo de software, de acordo com o nível de maturidade dos grupos de trabalho. O HyperDev oferece diferentes mecanismos para acompanhar a utilização do processo de software configurado, através do registro das informações do software. Através do HyperDev, o desenvolvimento do software pode ser monitorado, de forma padronizada. O sistema utiliza a tecnologia WWW e a sua construção teve como objetivos: validar a aplicação do modelo proposto, verificar a adequação da tecnologia utilizada e identificar restrições e possíveis melhorias.

HyperDev integra diversos mecanismos: configuração do processo de software, orientação à seqüência das atividades a serem realizadas, apoio à geração de documentos, suporte à comunicação entre os membros da equipe de trabalho, registro das tarefas realizadas, integração de documentos, auxílio à busca de informações do software, registro e visualização de diferentes abstrações dos componentes do software, registro das decisões tomadas e verificação da consistência da documentação produzida. Portanto, através de tais mecanismos, o sistema apóia a padronização do processo de construção do software, gerência das atividades de software, distri-

buição de tarefas entre os membros das equipes, geração automática de documentos, facilita o entendimento sobre os componentes do software e serve como uma base para a manutenção e futuras extensões.

A implementação foi realizada utilizando o WWW para apoiar a construção de software em ambientes distribuídos. HyperDev possui um alto grau de flexibilidade, pois oferece roteiros hipertextuais, que podem ser adaptados a diferentes projetos, para orientar a execução de tarefas e a produção de documentos. Através da Internet, a comunicação entre equipes de trabalho é facilitada e as informações podem ser acessadas rapidamente. O protocolo HTTP permite a integração de diferentes formatos de documentos. O registro de informações, ao longo da construção de um software, é acessado posteriormente, na etapa de avaliação do processo especializado do modelo proposto.

O servidor HyperDev pode ser acessado por vários clientes, formando uma subrede de informações, classificadas em: atividades do processo de software, informações do software e dados de controle do software. O primeiro grupo de informações corresponde à configuração do processo de software, que determina as atividades a serem realizadas e a seqüência de execução. A configuração é, então, registrada e, ao longo da construção do software, acessada através do sistema para orientar os desenvolvedores. O segundo tipo de informação corresponde àquelas geradas a cada atividade do processo de software. O conjunto destas informações é utilizado para a geração dos documentos do software. O seu registro também permite um maior controle sobre o processo: os requisitos podem ser verificados a cada etapa. Todas as informações do software são armazenadas de forma padronizada. O terceiro grupo de informações reúne os dados de controle do software. Dados de controle correspondem ao conjunto de informações que não se referem diretamente à construção do software: nome e endereço eletrônico dos membros das equipes, datas de início e término das atividades, decisões tomadas, etc. Estas informações estão relacionadas à gerência pois coordenam a comunicação entre os membros da equipe de trabalho, organizam e ordenam as tarefas a serem realizadas, permitem identificar restrições e contribuem como base para futuros projetos.

VI.2. Construção

A construção de HyperDev foi realizada através de protótipos evolutivos. Inicialmente, foram definidos os requisitos do sistema, os quais foram refinados a cada versão do sistema através da investigação das opções, determinação das decisões tomadas e identificação das restrições impostas pela tecnologia utilizada e pelo ambiente de desenvolvimento.

O sistema foi desenvolvido de forma modular para facilitar a integração dos diversos mecanismos e permitir a introdução de novas funcionalidades. O cerne de HyperDev corresponde aos módulos de configuração e apoio ao registro de atividades do processo de software. Cada mecanismo foi, inicialmente, implementado e testado separadamente, e então, integrado aos módulos principais através de ligações hipertextuais.

No processo de construção de HyperDev, utilizou-se o próprio Web para definir as atividades do seu desenvolvimento, registrar as decisões de projeto e as atividades de construção, como descrito a seguir:

- arquivo de mensagens eletrônicas: registra a comunicação entre os desenvolvedores, informando as próximas tarefas a serem executadas. O navegador Netscape foi utilizado para enviar e armazenar mensagens;

- documentação dos módulos: descreve a implementação, realizada por diferentes profissionais, facilitando o processo de integração dos módulos. Corresponde a um arquivo HTML que contém ligações hipertextuais a programas que executam separadamente os módulos;

- registro das decisões do projeto: descreve as deliberações correspondentes à versão atual do sistema e documenta idéias para futuras versões;

- cadastro de atividades: registra as tarefas já realizadas, a serem realizadas na versão atual e em futuras versões. Este documento facilita a coordenação das atividades, determina a prioridade das tarefas e permite o planejamento da construção da próxima versão do sistema.

A utilização de tais mecanismos de registro das atividades e informações sobre a construção do sistema apoiou o processo de implementação. A meta foi obter, o mais rápido possível, uma versão operacional do HyperDev. Através da sua execução, pôde-se identificar restrições e opções. Nas versões sucessivas, tais informações foram consideradas para permitir a produção de um sistema mais completo e eficiente.

VI.2.1. Requisitos

Os principais requisitos do modelo de gerência de processos de software a serem implementados no HyperDev são:

Requisitos Gerais

- **permitir a distribuição de tarefas pela rede de computadores**: tornar possível a paralelização das atividades realizadas por diferentes profissionais, independentemente de sua localização;
- **gerenciar a concorrência de atividades**: evitar conflitos que possam surgir durante a realização simultânea de tarefas;
- **permitir a sua utilização independentemente do hardware**: evitar limitações quanto à utilização de plataformas e sistemas computacionais.

Requisitos Funcionais

- **apoiar a configuração do processo de software**: oferecer diversas configurações padrão, de acordo com o modelo de maturidade de referência, permitindo a adequação do processo de software a diferentes projetos;
- **registrar projetos de software**: permitir o cadastramento de informações que identificam um projeto, possibilitando sua futura recuperação e acesso;
- **oferecer um mecanismo de comunicação entre os membros das equipes de trabalho**: possibilitar a coordenação de tarefas e a troca de informações em ambientes cooperativos;

- **registrar as mensagens enviadas aos membros das equipes de trabalho**: cadastrar a discussão entre os desenvolvedores, permitindo o acesso às informações;
- **registrar as atividades do processo de software**: possibilitar a coordenação de tarefas e servir de base para a produção dos documentos de software;
- **gerar os documentos do software**: facilitar a produção dos documentos que descrevem o processo de construção da aplicação;
- **permitir a integração de documentos externos ao sistema**: possibilitar que informações existentes ou produzidas por outros sistemas sejam inseridas no processo de construção da aplicação;
- **registrar as decisões de projeto**: apoiar a identificação de prioridades e o próprio processo de construção da aplicação;
- **registrar tarefas realizadas**: permitir que todos os usuários possam ter acesso ao trabalho realizado por um determinado membro da equipe de trabalho;
- **registrar os componentes da aplicação em construção**: possibilitar uma melhor compreensão da aplicação;
- **permitir a associação entre os componentes da aplicação em construção**: permitir a visualização de diferentes níveis de abstração dos componentes;
- **oferecer um mecanismo de busca às informações**: possibilitar a procura de dados cadastrados;
- **relatar as inconsistências da aplicação em construção**: identificar informações incompletas cadastradas pelos usuários.

Requisitos da Interface com o Usuário

- **oferecer uma interface amigável**: possibilitar que o usuário se concentre nas tarefas a serem realizadas, evitando esforços adicionais resultantes da interação com a interface do sistema;
- **permitir o acesso concomitante a diferentes mecanismos**: disponibilizar várias funções do sistema, evitando que o usuário interrompa o trabalho em curso;
- **validar a entrada de dados**: evitar que dados incorretos sejam cadastrados e minimizar o tempo de resposta do sistema, uma vez que as informações são validadas antes de serem processadas;
- **oferecer um sistema de ajuda**: oferecer um apoio adicional quanto à utilização do sistema.

Após a identificação dos requisitos, realizou-se a análise e projeto detalhado do HyperDev. Identificaram-se os módulos componentes e as funções de cada módulo foram, então, estruturadas e implementadas.

VI.2.2. Análise e Projeto

A figura VI.1 apresenta a arquitetura do sistema HyperDev. A base do sistema corresponde a um conjunto de *templates* que orientam o usuário ao longo do processo de construção do software, definindo todas as atividades e documentos a serem produzidos. Os *templates* integram as informações fornecidas pelos desenvolvedores. Os *templates* são tipados, ou seja, cada um corresponde a um tipo de informação: eventos do sistema, serviços do sistema, atividades do projeto de software, etc. As informações sobre a aplicação em construção são organizadas através de nós hipertextuais.

HyperDev é composto por diferentes bases de dados: **projetos**, informações sobre os projetos de software; **equipes**, registo do nome e endereço eletrônico dos

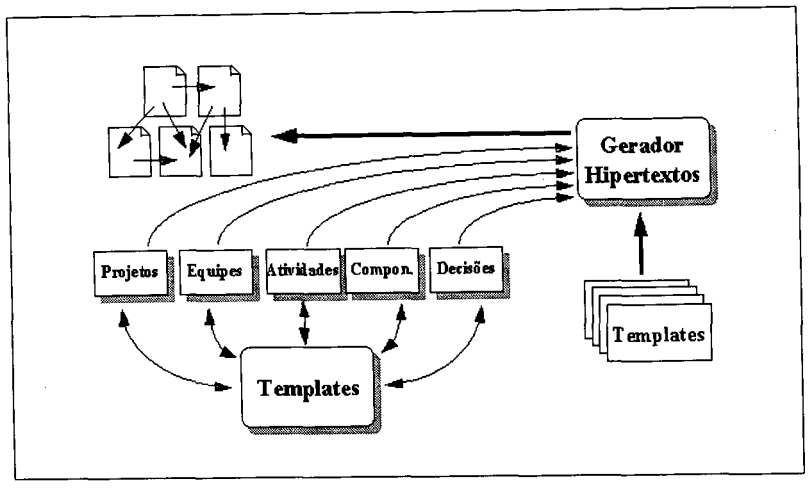


Figura VI.1: Arquitetura do Sistema HyperDev.

membros da equipe de trabalho; **atividades**, conjunto de informações sobre as atividades realizadas durante o processo de software; **componentes**, descrição dos componentes do software; e **decisões**, registro das determinações tomadas, modificações previstas e restrições identificadas durante o processo de software.

Durante a construção do software, documentos externos, gerados por ferramentas CASE ou editores de texto – contendo gráficos ou textos formatados – podem ser facilmente integrados às informações registradas nas bases de dados através de ligações hipertextuais. Todas as informações cadastradas pelo sistema são, então, integradas, gerando o hiperdocumento correspondente a todo o processo de software.

A figura VI.2 apresenta o diagrama resultante da análise do HyperDev. A modelagem do sistema é expressa em termos de objetos e suas relações. O objeto **projeto** determina um processo de software (**PDS**) e é formado por uma equipe de trabalho (**membros**). As **atividades** são compostas por **atividades técnicas** e **atividades gerenciais**. As **atividades técnicas** produzem **documentos** e o **software**, composto por diversos **componentes**. As **atividades gerenciais** produzem **decisões** . Os **membros** se comunicam entre si, registrando as **mensagens** enviadas durante o processo de software. **Documentos externos**, **mensagens**, **decisões**, **software** e respectivos **documentos** são convertidos em **hipertextos**, compostos por outros **hipertextos**.

Na fase de projeto, o HyperDev foi decomposto em módulos, cada um corres-

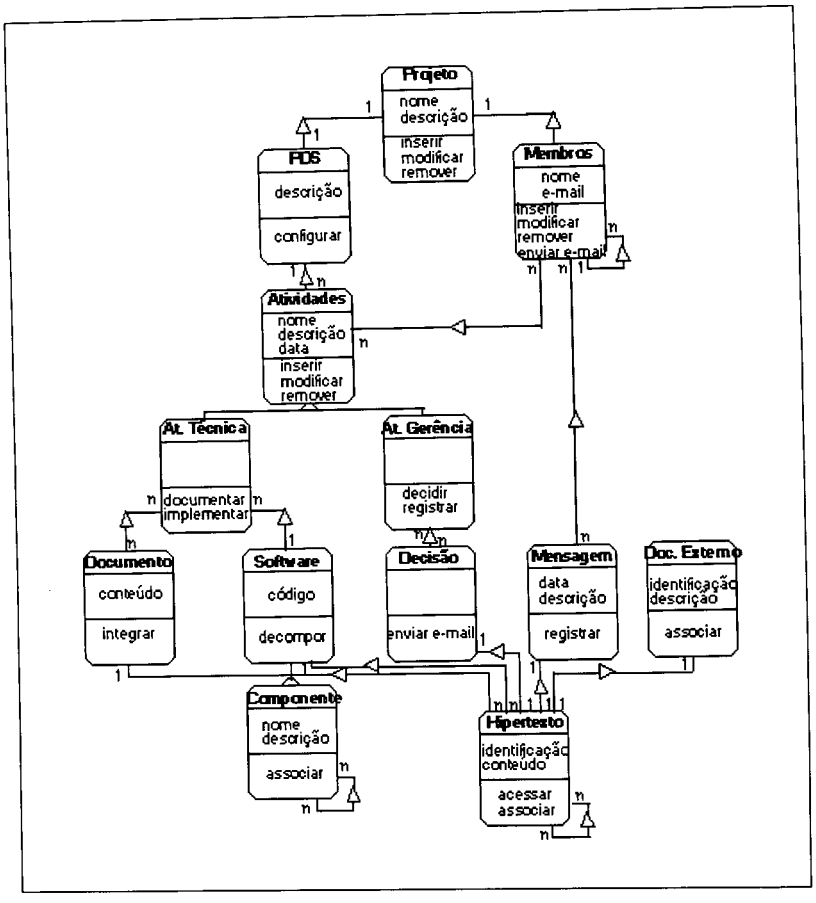


Figura VI.2: Modelagem do HyperDev.

pondendo a um conjunto de requisitos funcionais. Posteriormente, alocaram-se operações aos módulos e determinaram-se as interfaces com os usuários e bases de dados. A figura VI.3 apresenta os módulos do HyperDev. O módulo principal corresponde ao **Registro de Atividades** do processo de software. Este módulo é responsável pela apresentação dos *templates*, permitindo a inserção de informações do software, respeitando a configuração definida através do mecanismo de **Configuração do PDS**. Os módulos **Registro de Projetos** e **Cadastro de Membros** registram a descrição dos projetos de software e dos membros da equipe de trabalho, respectivamente. O módulo **Geração de Documentos** produz arquivos hipertextuais correspondentes ao processo de software, a partir do módulo **Integração de Documentos Externos** das atividades de software registradas. O módulo **Relatório de Inconsistências** apresenta dados incompletos e associações entre componentes do software inexistentes. Os módulos **Registro de Decisões**, **Registro de Tarefas**,

Registro de Componentes do Software, Comunicação e **Busca a Informações** oferecem mecanismos para coordenar e apoiar todas as demais tarefas relacionadas ao processo de software.

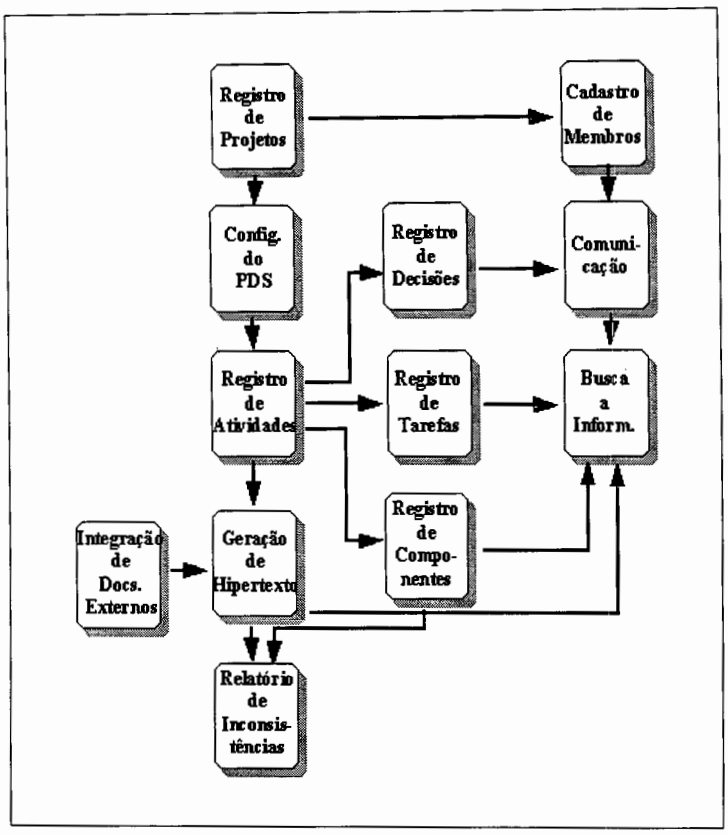


Figura VI.3: Módulos do HyperDev.

VI.2.3. Tecnologia Utilizada

HyperDev é baseado em uma arquitetura cliente/servidor permitindo que os usuários, trabalhando em diferentes localizações, o acessem através da rede de computadores. O HyperDev pode ser acessado por qualquer cliente WWW através de navegadores que suportam *FRAMES*. Esta extensão à linguagem HTML foi selecionada e utilizada na construção do HyperDev, pois permite dividir a janela de um navegador em várias subjanelas independentes, cada qual associada a um hipertexto distinto. A entrada de dados é realizada através do elemento *FORM* da linguagem HTML, oferecendo formulários hipertextuais.

A validação dos dados é realizada no próprio cliente através de programas Java-

Script evitando, portanto, sobrecarregar o tráfico de dados na rede de computadores: as informações não precisam ser enviadas ao servidor para serem validadas. Programas CGI atuam como elementos de ligação entre o servidor HTTP e as aplicações locais, implementadas em linguagem C e *shell scripts*. Tal estrutura permite a execução dos módulos HyperDev em tempo real, apresentando informações dinâmicas aos clientes.

Com o objetivo de validar a tecnologia a ser utilizada, algumas aplicações cliente/servidor foram desenvolvidas. Um sistema de cadastro de visitantes às páginas do projeto permitiu avaliar a eficiência e identificar as restrições da linguagem JavaScript e de programas CGI. Tal aplicação foi integrada ao sistema, constituindo o módulo para cadastro dos membros da equipe de trabalho.

VI.2.4. Ambiente de Operação

Todo o processamento se realiza no sistema operacional AIX da IBM, que está permanentemente ligada à Internet através da rede da UFRJ. A URL de acesso do HyperDev é: `http://www.lacc.ufrj.br/ hyperdev`.

No servidor Unix, é criado um usuário – denominado *hyperdev* – onde todos os programas e arquivos do sistema são armazenados. O servidor administra os eventos (execução de programas e apresentação da página hipertextual acessada pelo cliente) e o usuário *hyperdev* coordena o armazenamento e acesso de informações em uma estrutura hierárquica de diretórios. A figura VI.4 representa a seqüência de eventos relacionada à operação de busca e seleção de recursos, descritas a seguir:

1. Requisição de um recurso pelo cliente: o usuário requer um recurso do HyperDev por meio do seu programa navegador.

2. Identificação e busca do recurso pelo servidor: o servidor HTTP, o qual roda continuamente na Máquina 1, recebe o pedido de recurso através da Internet e o busca na Máquina 2, que armazena o código do HyperDev. Se este recurso for um documento HTML, então a seqüência segue diretamente para o evento 4; caso contrário, quando for um programa executável, segue para o evento 3.

3. Execução do programa: o programa é executado pela Máquina 1. Este programa pode acessar os bancos de dados do HyperDev, manipular e formatar dados para visualização pelo navegador, como também pode chamar outros programas. Normalmente, a saída de um programa é um texto contendo informações relacionadas, formatado e identificado como um documento HTML, para que possa ser interpretado pelo navegador do cliente.

4. O recurso é remetido pelo servidor de http: O resultado da busca de recurso é enviado através da Internet para a máquina cliente que o requisitou.

5. O navegador interpreta o recurso: o recurso chega ao cliente como um documento HTML, que é interpretado pelo navegador, fornecendo as informações requisitadas.

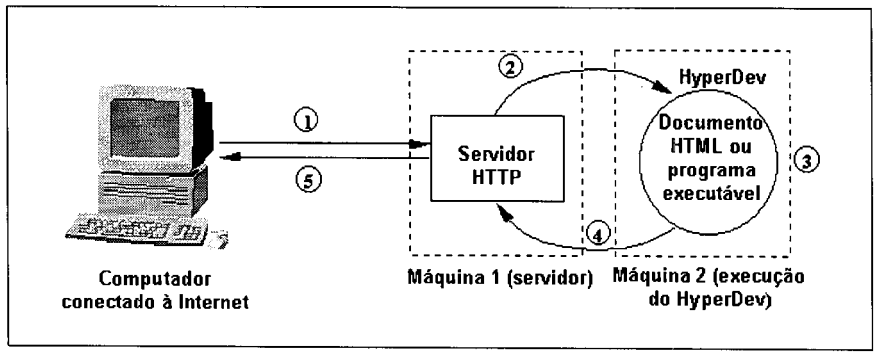


Figura VI.4: Seqüência de eventos para a busca por um recurso do HyperDev.

O cliente requisita os recursos do HyperDev como usuário remoto e, portanto, todos os recursos já estão previamente configurados para este tipo de acesso. Assim, os programas e documentos HTML devem ser de domínio público. A maneira como isso se traduz em termos computacionais está demonstrada na figura VI.5, que apresenta uma árvore de diretórios a partir da raiz da conta *hyperdev*. No diretório *public_html* ficam gravados os arquivos a que o servidor HTTP tem acesso. Os programas executáveis são acessados através do diretório *cgi-bin*, onde estão armazenados os ponteiros para os programas originais, armazenados nos diretórios correspondentes aos módulos aos quais pertencem.

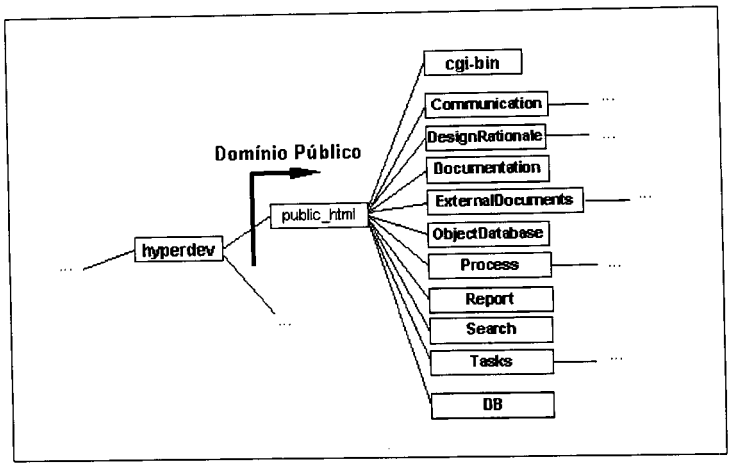


Figura VI.5: Árvore de subdiretórios da conta *hyperdev*.

VI.2.4.1. Complexidade do Sistema

O HyperDev é composto basicamente de cinco tipos de arquivo: documentos HTML, arquivos fonte em linguagem C, arquivos executáveis, *shell scripts* e arquivos texto. A tabela VI.1 apresenta como estes arquivos estão distribuídos pelos módulos do HyperDev. Ainda existem as imagens e os arquivos de dados temporários. São ao todo trinta e sete imagens armazenadas nos formatos GIF e JPEG. Os arquivos temporários são necessários para algumas funções do HyperDev. A tabela VI.2 apresenta a complexidade do sistema em termos de linhas de código (LOC).

Módulos do HyperDev	**Documentos HTML**	**Arquivos fonte em C**	**Arquivos Executáveis**	***Shell Scripts***	**Arquivos Texto**
Tela principal	12	17	15	19	4
Software Process	0	7	6	4	1/atividade/projeto
Object	0	8	6	22	2/projeto
Design Rationale	0	12	10	20	4/projeto
Communication	([1])	2	2	5	0
Tasks	0	3	2	4	1/usuário/projeto
Search	0	0	0	2	0
Documentation	0	3	2	2	0
External Documents	0	10	9	17	1/projeto+docs. externos
Report	0	7	1	2	0
TOTAL	**12**	**69**	**53**	**97**	**variável**

Tabela VI.1: Complexidade do sistema (em quantidade de arquivos por módulo).

Módulos do HyperDev	Documentos HTML	Arquivos fonte em C	*Shell Scripts*
Tela principal	290	1870	1254
Software Process	0	799	285
Object	0	592	1152
Design Rationale	0	2204	1157
Communication	(1	205	702
Tasks	0	131	221
Search	0	0	113
Documentation	0	1113	60
External Documents	0	933	816
Report	0	2010	16
TOTAL	**290**	**9857**	**5776**

Tabela VI.2: Complexidade do sistema (em linhas de código).

VI.3. Utilização do Sistema

Ao utilizar HyperDev, é preciso, inicialmente, cadastrar um projeto de software e configurar o processo de software, determinando as fases, as atividades e sub-atividades a serem realizadas. Após estas etapas, os integrantes dos grupos de trabalho têm acesso ao ambiente principal de trabalho, mediante suas senhas individuais. O ambiente de trabalho é composto pelos módulos do HyperDev. A figura VI.6 representa a seqüência das tarefas descritas.

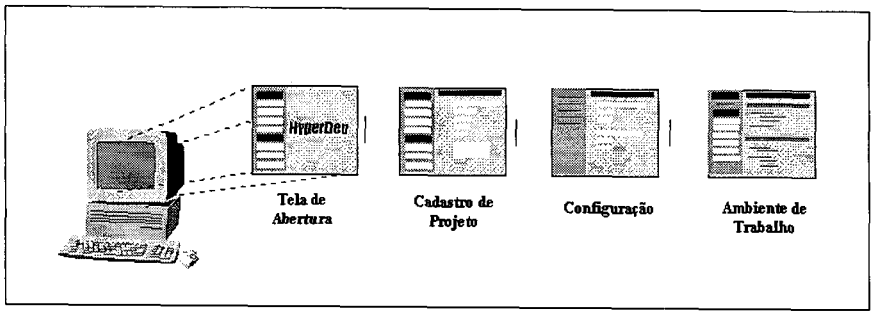


Figura VI.6: Utilização do HyperDev.

O usuário tem acesso direto à área de trabalho através da sua senha. Todas as ações realizadas são devidamente atribuídas ao usuário em questão. A estrutura cliente-servidor do HyperDev permite que vários projetos diferentes sejam utilizados

ao mesmo tempo por diferentes usuários espalhados pela Internet, ou que várias pessoas trabalhem simultaneamente no mesmo projeto em computadores diferentes.

VI.3.1. Tela Principal

A figura VI.7 apresenta a tela principal do HyperDev. As referências à esquerda apresentam resumidamente a função de cada um dos botões. Estas funções iniciais estão relacionadas com o cadastramento e a remoção de projetos e com a consulta, modificação e busca de informações de projeto e de pessoal. O acesso à área de trabalho dos projetos registrados é também realizado a partir desta tela. Todas estas funções são apresentadas a seguir, na ordem em que estão dispostas na tela.

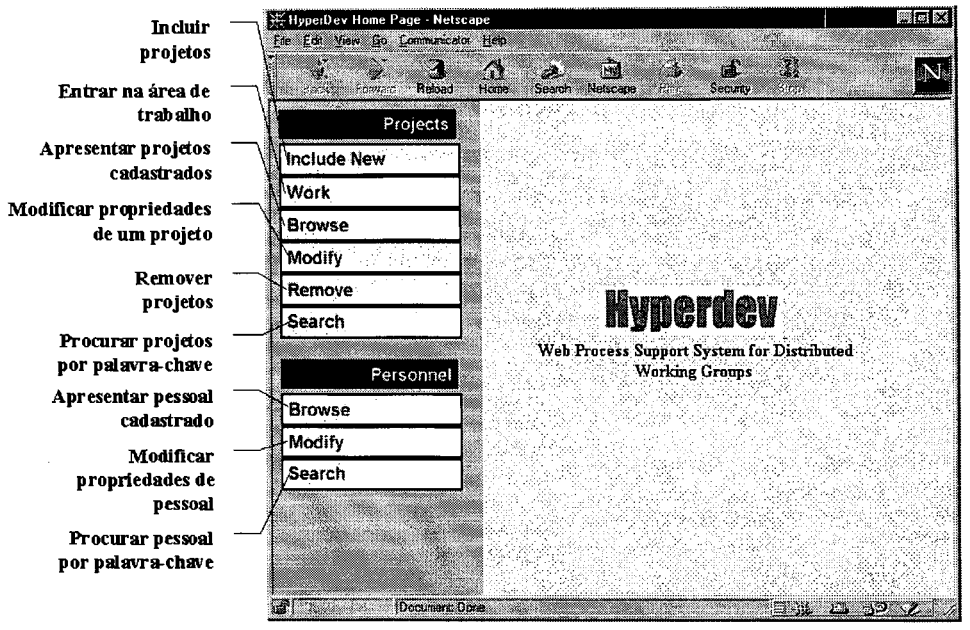


Figura VI.7: Janela de abertura do HyperDev.

VI.3.1.1. Incluir Projetos

Esta função, que corresponde ao botão *Include New*, permite o cadastramento de um novo projeto. Ao ativar o botão, um formulário (figura VI.8) é apresentado com os campos que devem ser necessariamente preenchidos: nome, descrição, data, número da versão, nome dos membros das equipes e do responsável pelo projeto e senha. O responsável pelo projeto é também considerado um integrante do grupo

de trabalho que retém os direitos de alteração de propriedades gerais e remoção do projeto, através da senha do projeto.

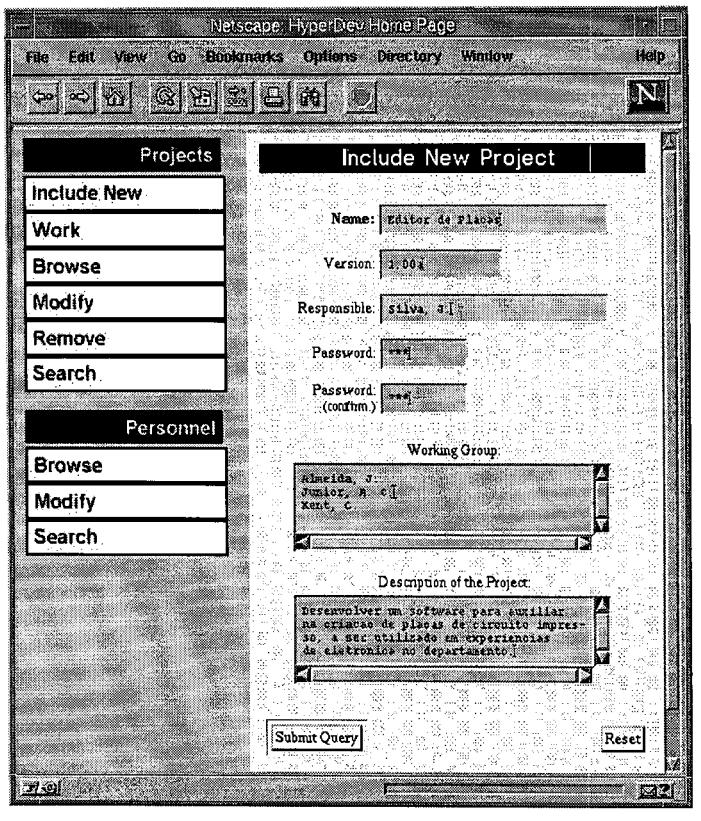


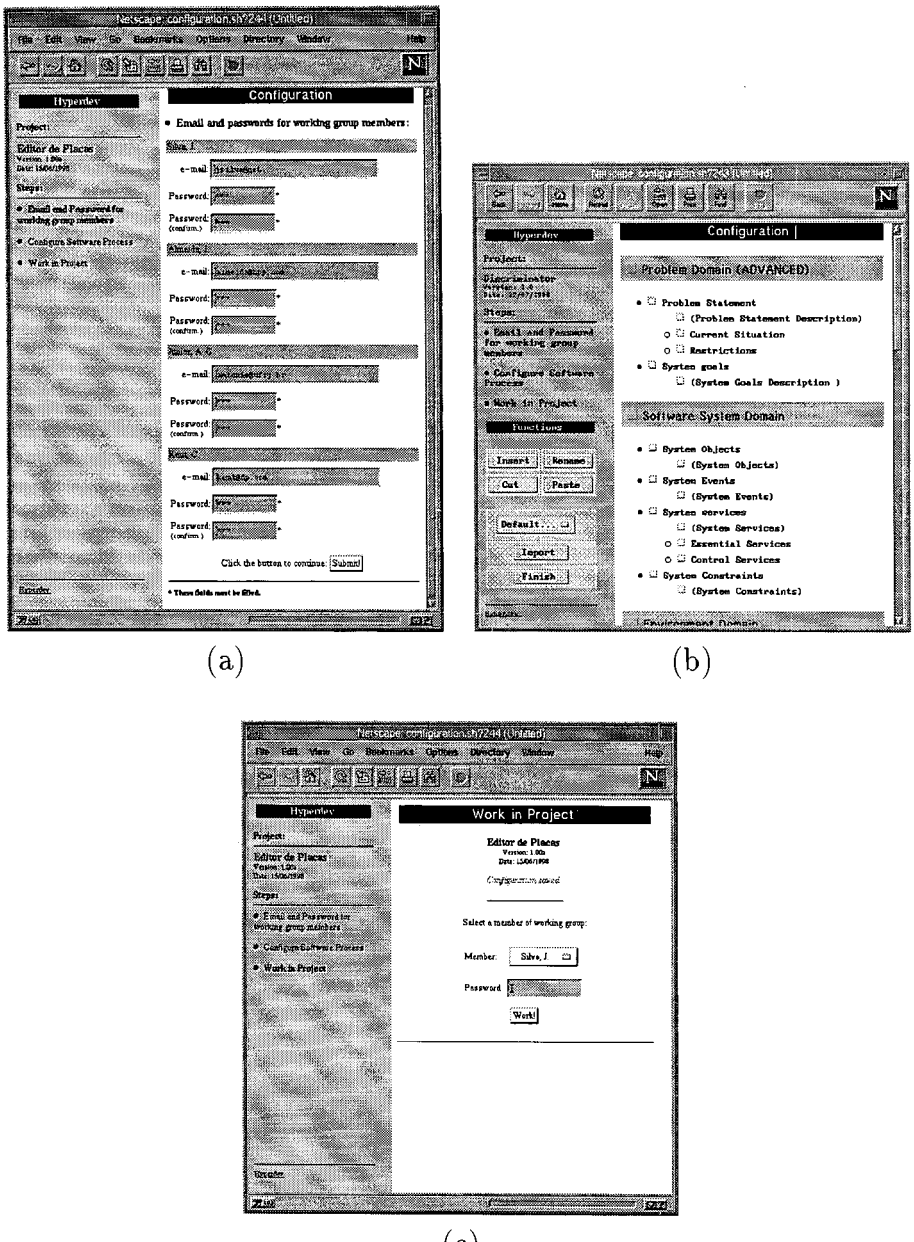
Figura VI.8: Inclusão de um novo projeto.

Após o cadastro inicial, cada integrante do projeto informa o seu endereço eletrônico e senha, como apresentado na figura VI.9 (a). As senhas restringem o acesso dos usuários às suas respectivas áreas de trabalho.

Na etapa seguinte, o processo de software é configurado, como apresentado na figura VI.9 (b). O HyperDev oferece a configuração *default* e quatro especializações do processo de software através de uma interface hipertextual (botão *Default*). As configurações dos processos de projetos cadastrados também podem ser selecionadas através do botão *Import*. O responsável ou gerente do projeto escolhe uma das opções, podendo modificar a configuração do processo para adequá-la ao projeto a ser desenvolvido.

Após a configuração, o usuário tem acesso à área de trabalho, selecionando o seu nome em um menu *pull-down* e digitando a sua senha, como apresenta a figu-

ra VI.9 (c). O usuário deve, então, ativar o botão *Work!*.



(a) (b)

(c)

Figura VI.9: (a) Atribuição de *emails* e de senhas individuais; (b) configuração do processo de software e (c) seleção do usuário para acessar a área de trabalho.

VI.3.1.2. Trabalhar em um Projeto

A função *Work* permite o acesso à área de trabalho. O usuário deve informar o nome do projeto, selecionar o seu nome e informar a sua senha. Caso o HyperDev não encontre nenhum projeto, cujo nome corresponda exatamente à seqüência de

caracteres digitada, o sistema apresenta a lista dos projetos, cujos nomes contenham a seqüência de caracteres. A figura VI.10 apresenta todos os passos desta operação: na figura VI.10 (a), o usuário digita "cam"; como não existe nenhum projeto com este nome, o HyperDev apresenta o nome de um provável projeto alvo, chamando "Camlk" (figura VI.10 (b)). Na janela seguinte (figura VI.10 (c)), o usuário seleciona o seu nome e digita a sua senha.

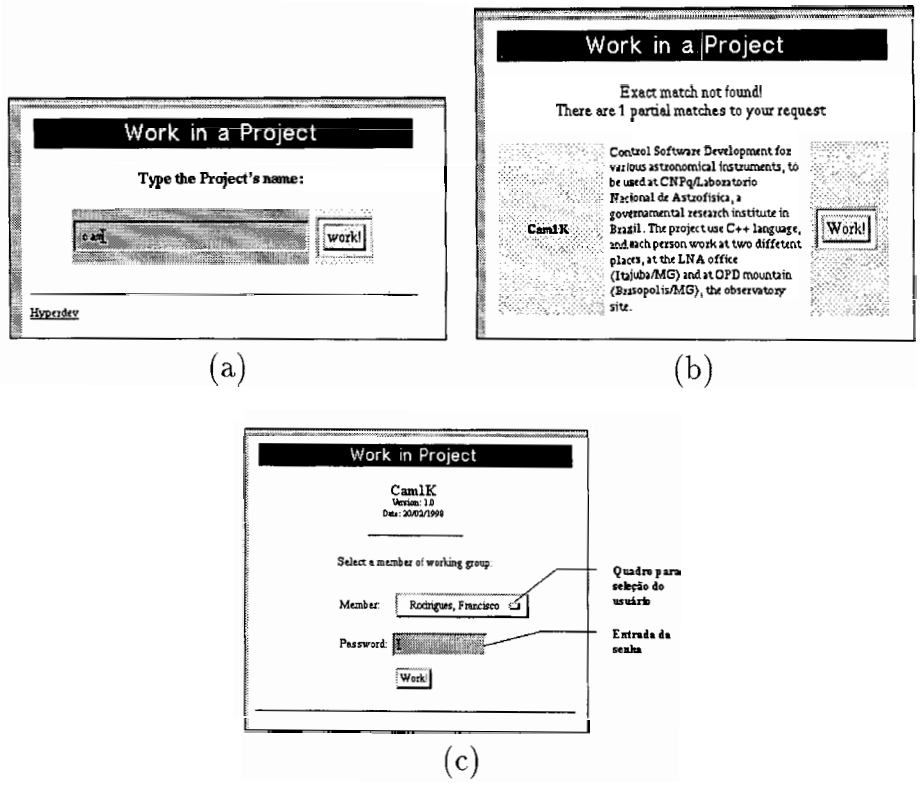


Figura VI.10: Função *Work*: (a) entrada do nome do projeto; (b) lista de projetos e (c) janela de seleção de usuário e entrada da senha.

VI.3.1.3. Acessar Projetos Cadastrados

Esta função, ativada pelo botão *Browse*, apresenta todos os projetos cadastrados, em ordem alfabética. Um projeto é selecionado através do botão *Work*. (figura VI.11).

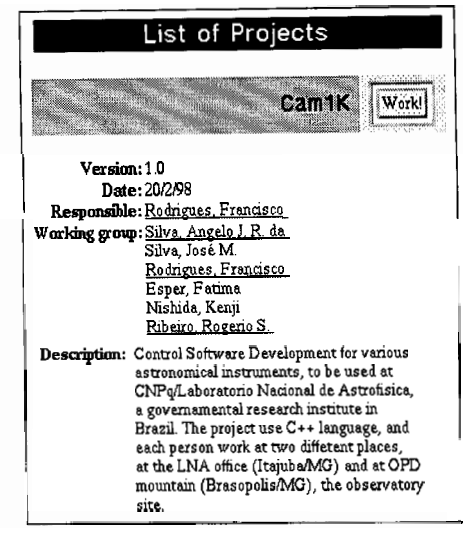
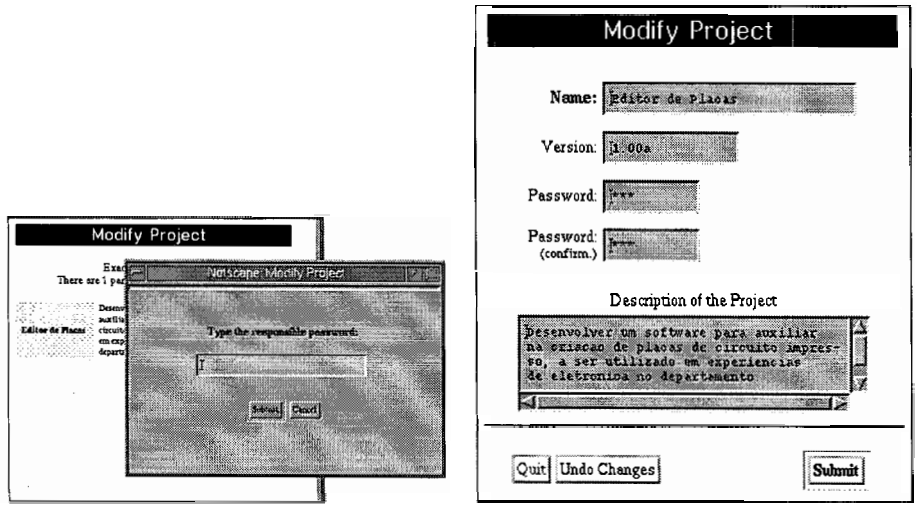


Figura VI.11: Função *Browse*.

VI.3.1.4. Modificar o Cadastro de um Projeto

Por meio desta função, acionada pelo botão *Modify*, é possível modificar os dados cadastrais de um projeto. O usuário precisa informar o nome completo, ou em parte, do projeto a ser modificado, como na função *Work*. É necessário informar a senha do projeto para ter acesso à tela de modificação. A figura VI.12 apresenta um exemplo do uso desta função.



(a) (b)

Figura VI.12: Função *Modify*: (a) seleção do projeto e entrada de senha e (b) janela de modificação.

VI.3.1.5. Remover um Projeto

Esta função, ativada pelo botão *Remove*, remove um projeto cadastrado. O usuário informa o nome completo, ou em parte, do projeto e depois a senha, como na função *Modify*.

VI.3.1.6. Procurar por um Projeto

A função *Search* realiza buscas, através de palavras-chave, a projetos pesquisando os dados cadastrais de todos os projetos. O usuário informa a palavra-chave e as opções de busca, que selecionam quais campos devem ser pesquisados. O resultado da busca é apresentado como na função *Browse* (figura VI.11). A figura VI.13 apresenta o formulário de busca.

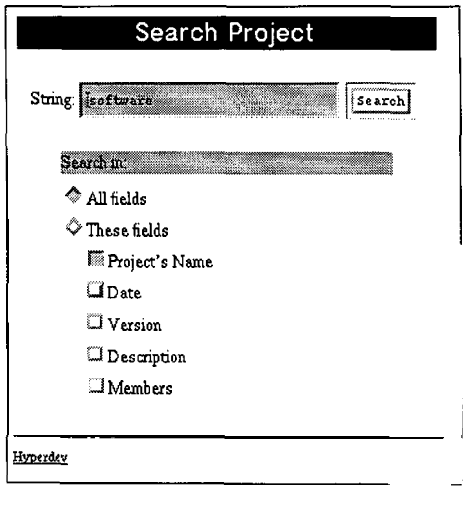


Figura VI.13: Função *Search*.

VI.3.1.7. Acessar o Pessoal Cadastrado

A função *Browse* (*Personnel*) apresenta, em ordem alfabética, todos os membros cadastrados, incluindo o *email* e o nome do projeto do qual fazem parte. Os *emails* correspondem a ligações hipertextuais para a janela de edição de mensagens. As informações listadas pela função *Browse* (*Personnel*) são apresentadas na figura VI.14.

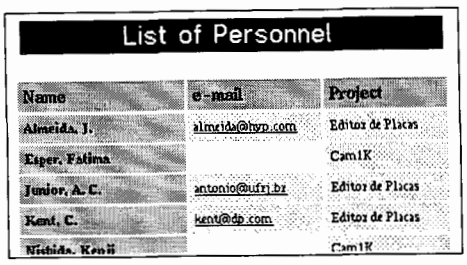
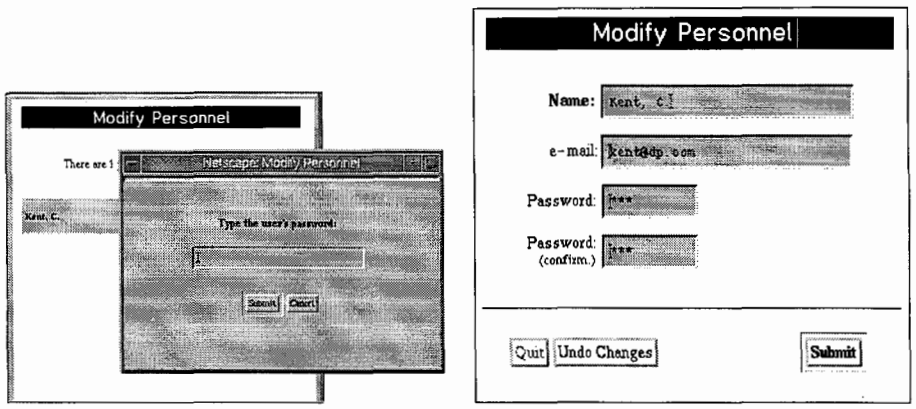


Figura VI.14: Função *Browse (Personnel)*.

VI.3.1.8. Modificar Informações Pessoais

A função *Browse (Personnel)* permite a alteração dos dados do pessoal cadastrado. O usuário deve informar o seu nome e senha. A figura VI.15 apresenta um exemplo desta função.



(a) (b)

Figura VI.15: Função *Modify (Personnel)*: (a) Confirmação do nome do usuário e entrada de senha individual e (b) janela de modificação.

VI.3.1.9. Procurar por um Membro

A função *Search (Personnel)* realiza uma busca, por palavra-chave, no cadastro de membros das equipes de trabalho. O resultado da busca é apresentado como na função *Browse (Personnel)*. A figura VI.16 apresenta o formulário de busca.

VI.3.2. Área de Trabalho

A área de trabalho reúne os módulos de apoio ao desenvolvimento de software. A figura VI.17 apresenta a tela principal da área de trabalho, que é divida em dois

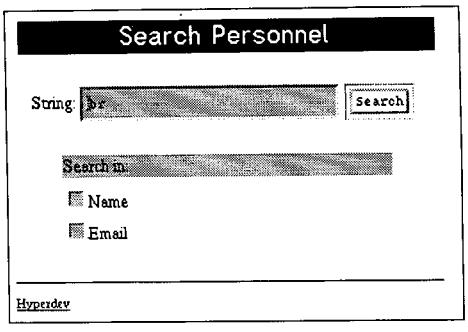


Figura VI.16: Função *Search (Personnel)*.

quadros. No quadro da esquerda estão informações sobre o projeto e membros da equipe, juntamente com os botões que acionam os módulos. À direita, apresenta-se o processo de software. O nome das fases é destacado e as atividades, e respectivas sub-atividades, correspondem a *links* para as páginas hipertextuais, onde suas informações são inseridas.

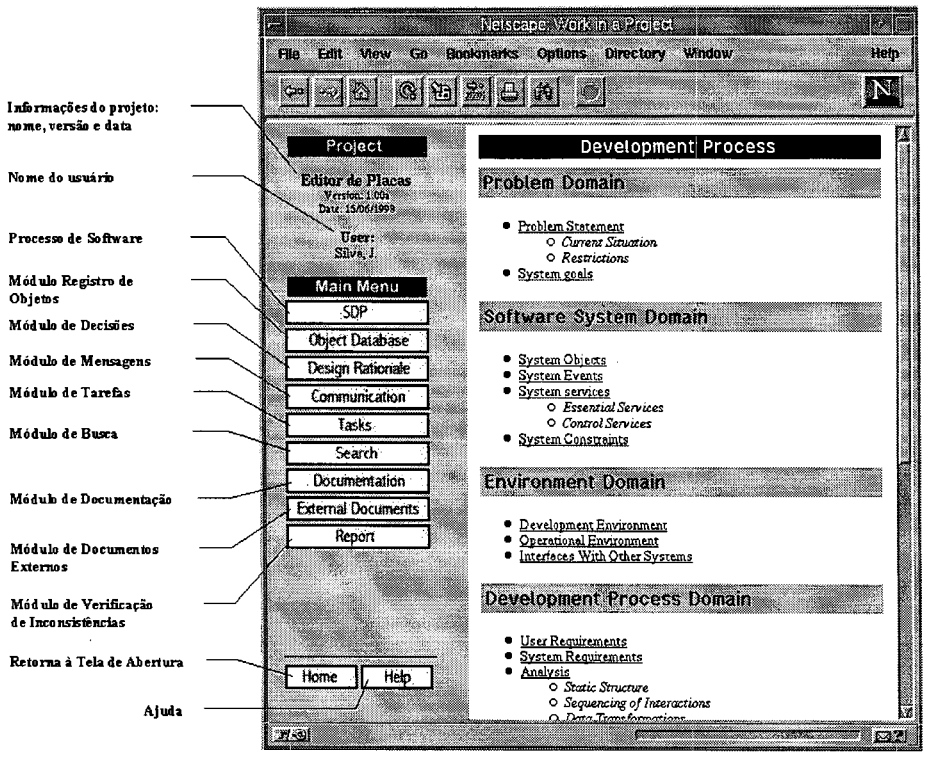


Figura VI.17: Tela principal da área de trabalho.

O processo de software é apresentado na janela principal de trabalho, enquanto os outros módulos são apresentados em janelas de trabalho separadas.

VI.3.3. Módulo Processo de Software

O módulo processo de software ou *SDP* (do inglês, *Software Development Process*) é composto por um conjunto de formulários, onde os integrantes das equipes inserem as informações correspondentes ao trabalho realizado. Um formulário de uma atividade apresenta campos de texto referentes à descrição de uma atividade e suas sub-atividade. As informações das atividades descrevem objetivos ou resultados. A figura VI.18 apresenta um exemplo. A atividade "Problem Statement" é selecionada, por meio do seu *link*. O quadro ampliado apresenta o campo de decrição da atividade preenchido com informações de projeto.

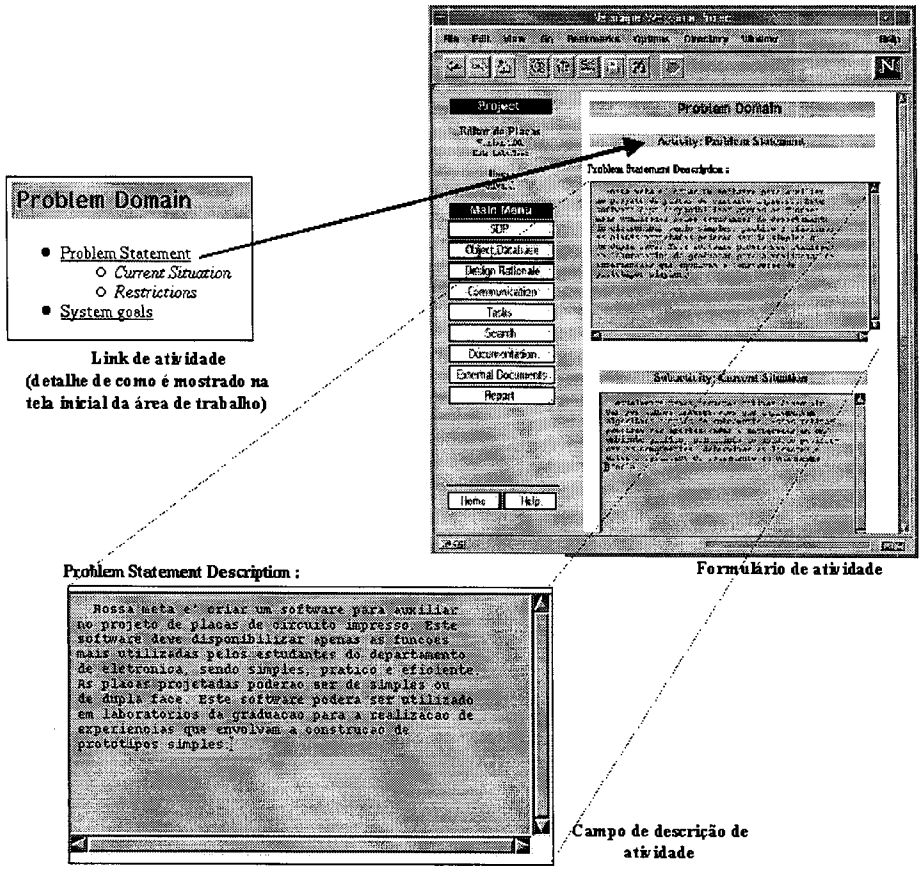


Figura VI.18: Trabalho em uma atividade.

Após terminar o trabalho em uma atividade, o usuário pode salvar o formulário, mesmo que alguns campos estejam incompletos, como apresentado na figura VI.19. As informações são armazenadas no repositório do projeto e o formulário da atividade subseqüente é automaticamente apresentado. O botão "SDP" apresenta o

processo de software completo (figura VI.17). Os campos que foram deixados em branco podem ser preenchidos em uma próxima sessão de trabalho.

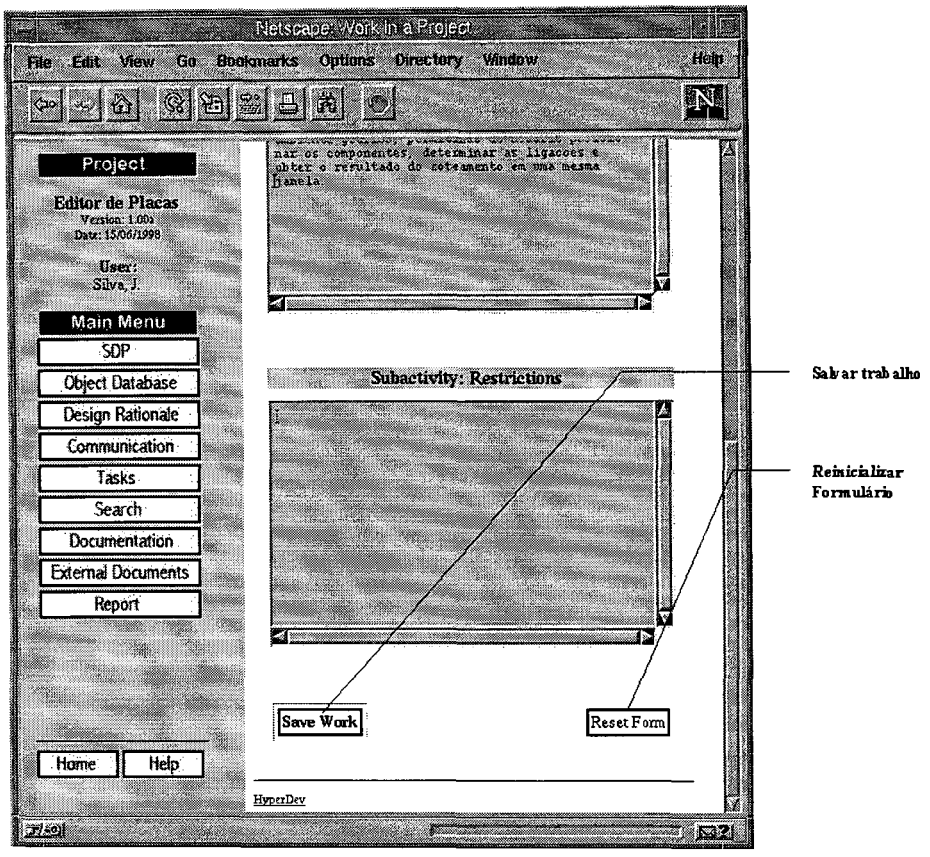


Figura VI.19: Término do trabalho em uma atividade.

Além das informações inseridas pelo usuário, o HyperDev registra um histórico das modificações realizadas. O autor e a data das modificações são armazenados juntamente com o conteúdo anterior à modificação dos campos de uma atividade. Tais informações são apresentadas quando a atividade for novamente selecionada.

Ao acionar os demais módulos do HyperDev, o nome da atividade acessada é passado como parâmetro. Portanto, as operações realizadas através dos módulos registram o nome da atividade, criando uma ligação hipertextual com o processo de software.

VI.3.4. Registro de Objetos

Quando o usuário ativa o botão *Object Database*, uma outra janela é aberta exclusivamente para apresentar este módulo. A figura VI.20 apresenta a tela prin-

cipal deste módulo, que é composta por dois quadros. À esquerda estão algumas informações, como os nomes do projeto e do usuário, a atividade na qual o módulo foi acionado e os botões que acionam as funções relacionadas ao módulo. Todos os outros módulos do HyperDev apresentam tais informações neste mesmo formato. No quadro da direita, são apresentadas as informações sobre os objetos componentes do software.

As quatro funções básicas do módulo são: *List*, *Create*, *Modify* e *Delete*. A função *List* apresenta os objetos e suas propriedades: nome do objeto, nome do usuário que o criou, data de criação, uma URL de referência, a descrição do objeto e os de suas ligações hipertextuais. A figura VI.20 apresenta tais informações. Os *links* dos objetos apontam para informações contidas no módulo processo de software e são formados pelo nome da atividade seguido do nome de um dos campos de texto presentes no formulário da atividade. Ao acionar um *link*, o usuário tem acesso, em uma janela separada, ao conteúdo do campo de texto correspondente. O nome do autor de cada *link* e a data de criação também são apresentados.

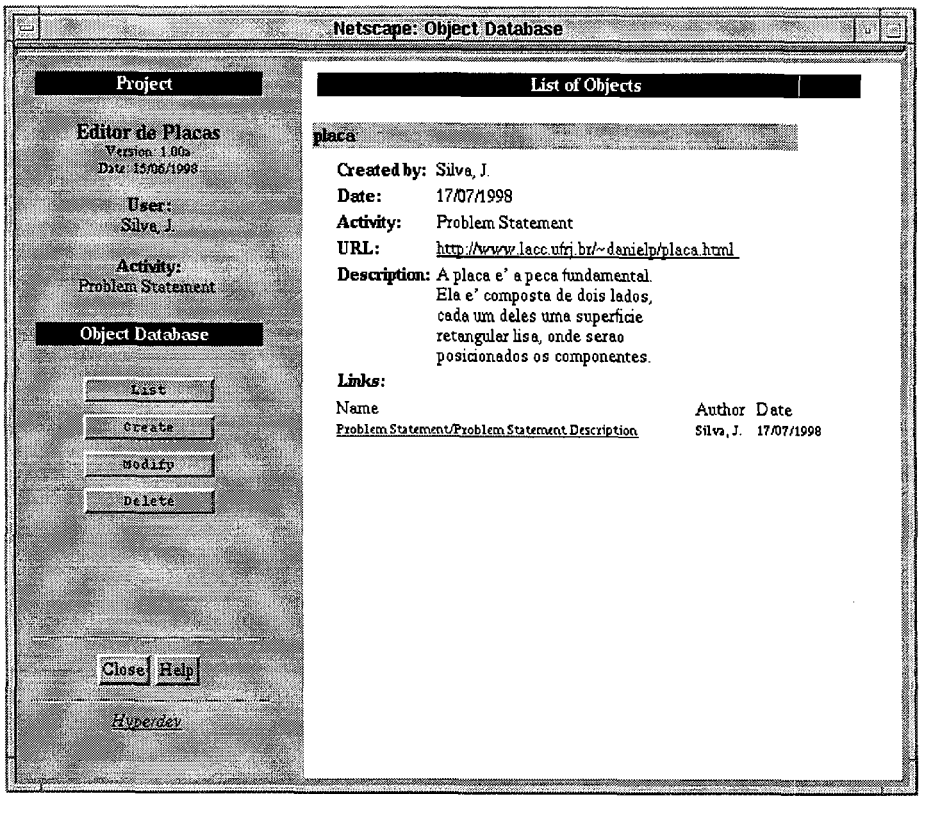


Figura VI.20: Módulo *Object Database*: Tela principal / Função *List*.

A função *Create* permite criar um novo objeto. O formulário de criação, que aparece no quadro direito do módulo, é apresentado na figura VI.21. O usuário informa o nome, a URL de referência e a descrição do objeto, além de selecionar uma atividade do processo de software relacionada ao objeto. Uma função JavaScript atualiza a lista de sub-atividades, de acordo com a atividade selecionada. Duas funções auxiliam a criação do objeto: *View Link* e *Test URL*. A primeira permite visualizar as informações sobre a sub-atividade selecionada e a segunda verifica se a URL de referência existe. Após a inserção dos dados, o objeto é registrado através do botão *Create Object*. O nome do usuário, a data de criação e a atividade corrente são adicionados automaticamente no registro do objeto.

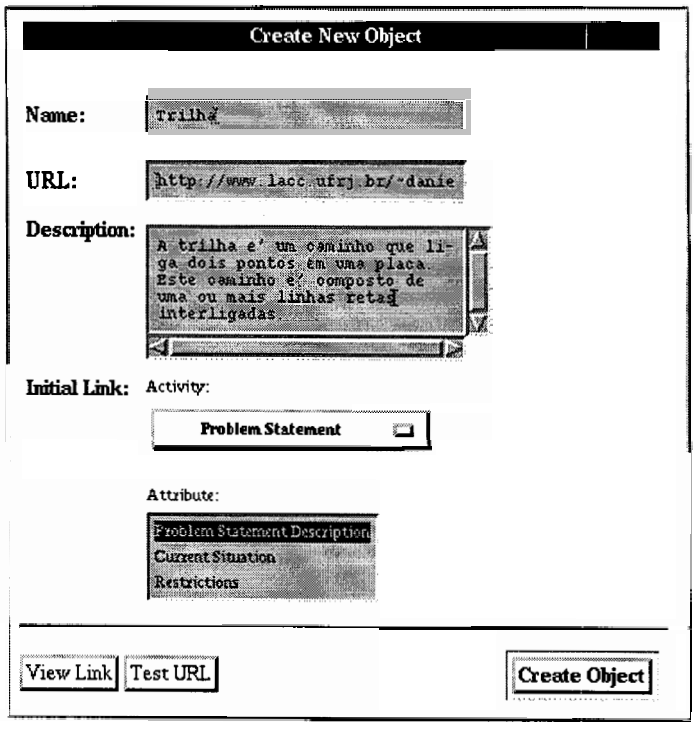


Figura VI.21: Módulo *Object Database*: Função *Create*.

A função *Modify* permite modificar os dados de um objeto registrado, tais como: alterar as propriedades de identificação do objeto (nome, URL de referência e descrição), modificar ou remover um *link* existente e criar um novo *link*. A figura VI.22 (a) apresenta a tela onde um objeto é selecionado (por meio de um menu *pull-down*) e o tipo de modificação é escolhido. O botão *Modify Description* aciona a tela de modificação de propriedades do objeto (figura VI.22 (b)). Tal operação

está restrita ao autor do objeto. O botão *Change/Delete Link* aciona uma tela que apresenta as propriedades do objeto selecionado. Para cada *link* existente são apresentados dois botões: *Change* e *Delete*, como apresentado na figura VI.22 (c). A tela de modificação de um *link*, acionada através do botão *Change*, é apresentada na figura VI.22 (d). Esta tela apresenta as propriedades do objeto selecionado, o nome do *link* a ser modificado e dois menus para a seleção do novo *link*, exatamente como na função *Create*. Através da função *Delete*, um objeto pode ser removido mas, somente por seu autor. A função *Create Link* aciona uma tela para a criação de um novo *link*. Esta função pode ser realizada por qualquer membro cadastrado do projeto, mesmo não sendo o autor do objeto.

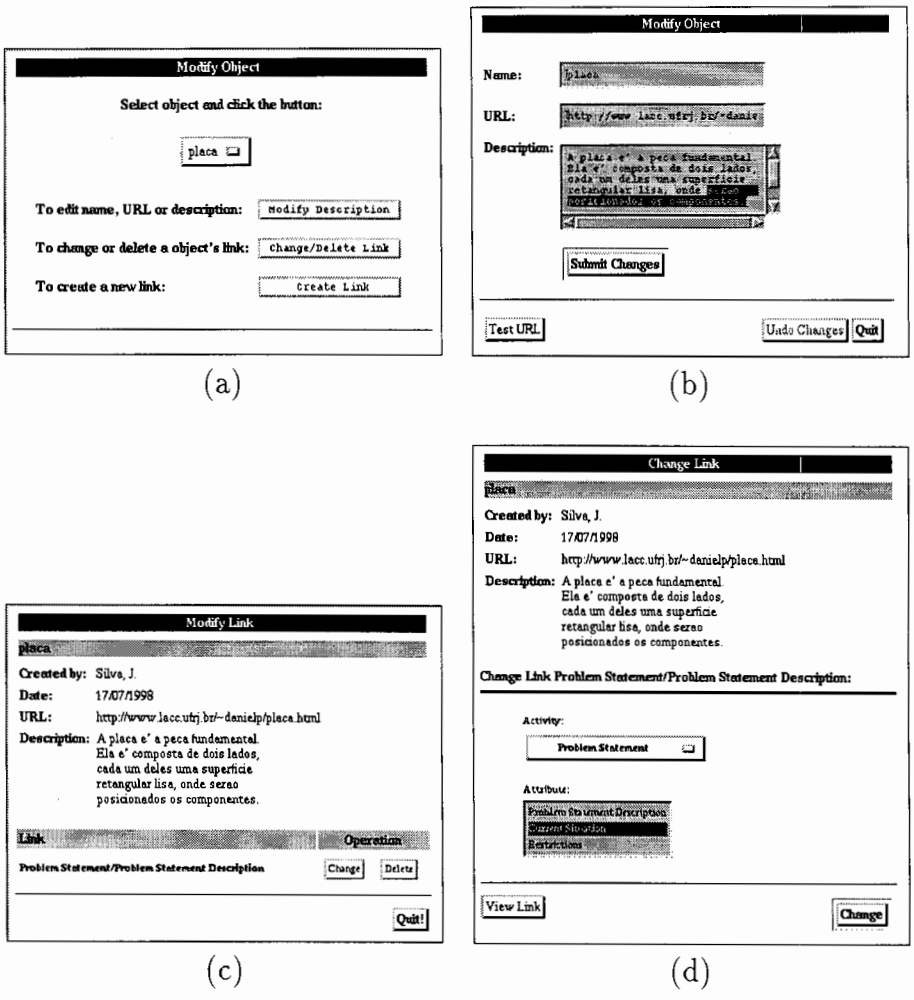


Figura VI.22: Módulo *Object Database*. Função *Modify*: (a) seleção do objeto e da modificação a ser realizada; (b) modificação das propriedades do objeto; (c) seleção de um *link* e (d) modificação de um *link* do objeto.

A função *Delete* remove um objeto do banco de dados de objeto. A operação de remoção é permitida apenas ao usuário que criou o objeto. Para ativar esta função, o usuário seleciona o objeto por meio de um menu *pull-down* (como na função *Modify*) e pressiona o botão *Delete*, ambos no quadro da direita da tela.

VI.3.5. Registro de Decisões e Implementações

Este módulo mantém um registro das decisões e das implementações do projeto. O acesso ao módulo é realizado através do botão *Design Rationale* na área principal de trabalho. A figura VI.23 apresenta a tela correspondente ao módulo. Existem quatro tipos de dados que podem ser manipulados: as decisões, as implementações já realizadas, as implementações previstas para a versão atual e as previstas para futuras versões. Sobre estes dados operam um mesmo conjunto de funções básicas: *List, Insert, Modify* e *Delete*. O usuário seleciona, através de menu *pull-down*, o tipo de dado com o qual deseja trabalhar. O conteúdo do quadro da direita é atualizado automaticamente ao modificar a seleção do tipo de dado.

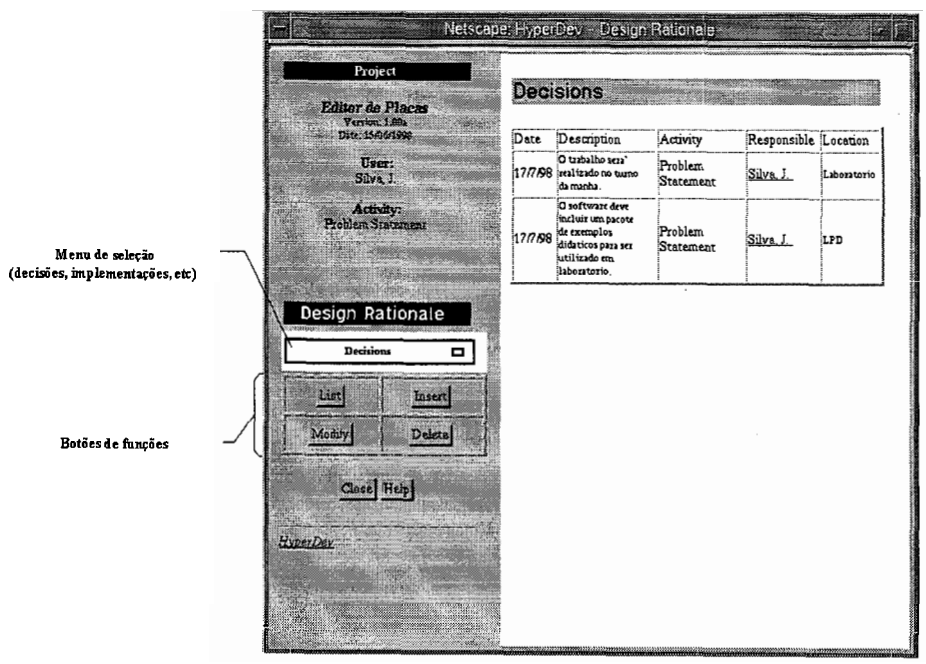


Figura VI.23: Módulo *Design Rationale*: janela principal / função *List*.

A função *List* apresenta as informações cadastradas. Para as *decisões*, as informações apresentadas são: a data de inserção, descrição, a atividade corrente, o nome

do usuário responsável e o local onde a decisão foi tomada. Para as *implementações já realizadas*, as informações apresentadas são as mesmas descritas anteriormente, com exceção da data, que se divide em dois tipos: a data em que a implementação foi registrada e a data da sua realização. No registro de *implementações previstas para esta versão*, apresenta-se o botão *Done* ao lado de cada item, permitindo mover o item para *implementações já realizadas*. O registro de *implementações previstas para futuras versões* requer as mesmas informações das decisões.

A função *Insert* permite cadastrar novas informações, como apresentado na figura VI.24. O nome do usuário, data e atividade corrente são adicionadas automaticamente pelo sistema no cadastro.

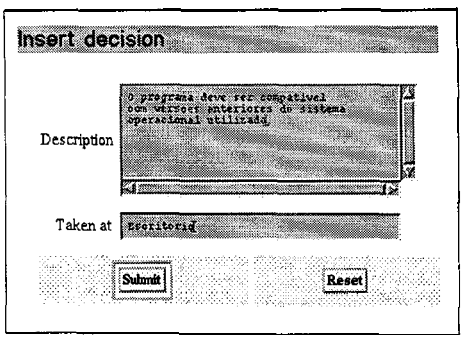


Figura VI.24: Módulo *Design Rationale*: função *Insert*.

A função *Modify* apresenta as informações de todos os itens do tipo de cadastro selecionado. Para cada item, se o usuário for o autor, o botão *Modify* é apresentado. Caso contrário, a mensagem *"not owner"* é apresentada. Tal mensagem indica que o usuário não tem direito a realizar a operação solicitada. A figura VI.25 apresenta um exemplo da função *Modify*.

A função *Delete* remove um item do tipo de cadastro selecionado. Assim como na função *Modify*, as informações de todos os itens são apresentadas e o usuário só poderá remover aquelas que ele próprio tenha inserido. A figura VI.26 apresenta um exemplo desta função.

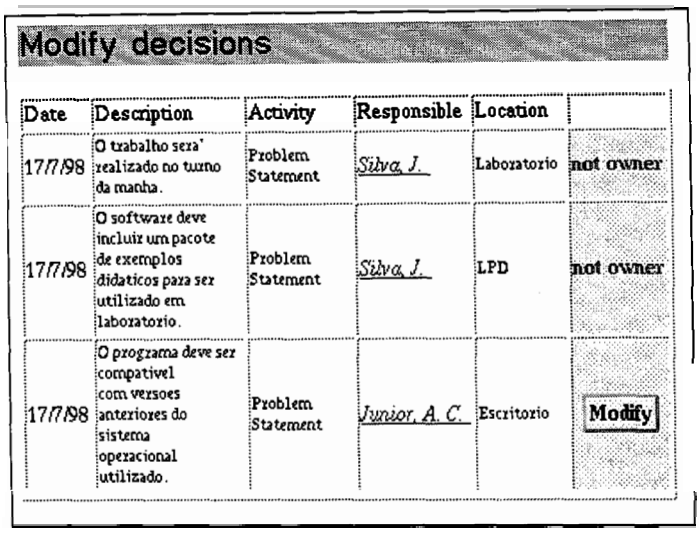


Figura VI.25: Módulo *Design Rationale*: função *Modify*.

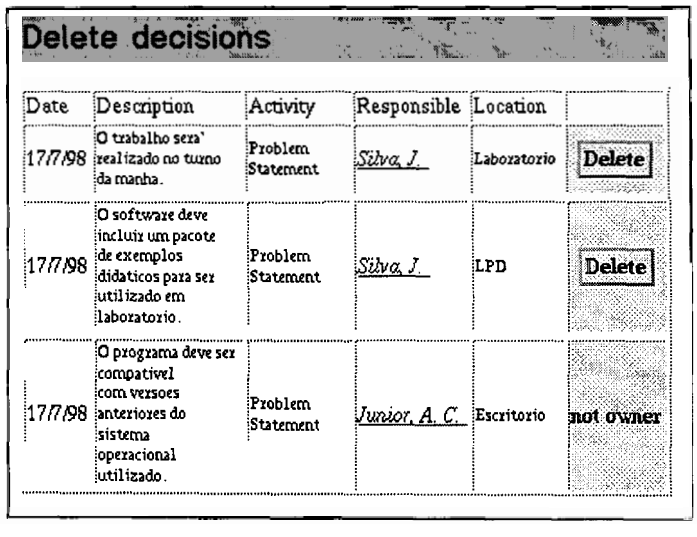
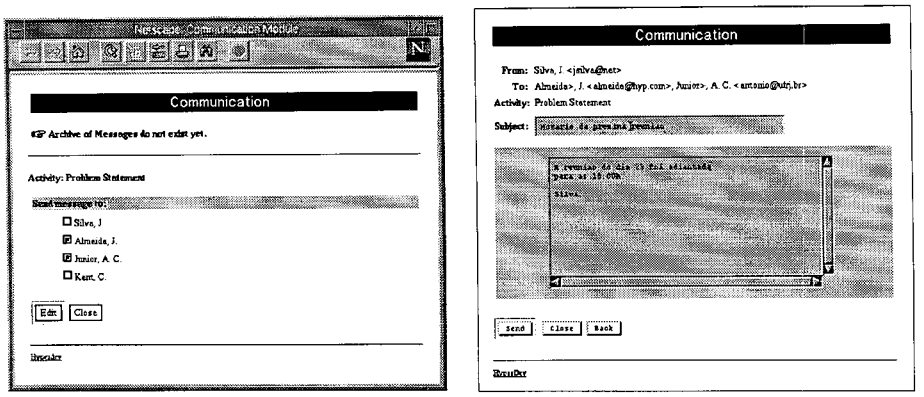


Figura VI.26: Módulo *Design Rationale*: função *Delete*.

VI.3.6. Módulo de Comunicação

O módulo de comunicação é acionado quando o usuário ativa o botão *Communication*, na área principal de trabalho. Este módulo permite enviar mensagens eletrônicas através da Internet para os membros de um projeto, sendo que uma cópia fica registrada no repositório de mensagens. Este repositório pode ser acessado por todos os integrantes do projeto. A tela principal deste módulo é apresentada na figura VI.27 (a). A partir desta tela, o usuário pode enviar uma mensagem ou consultar o repositório de mensagens.

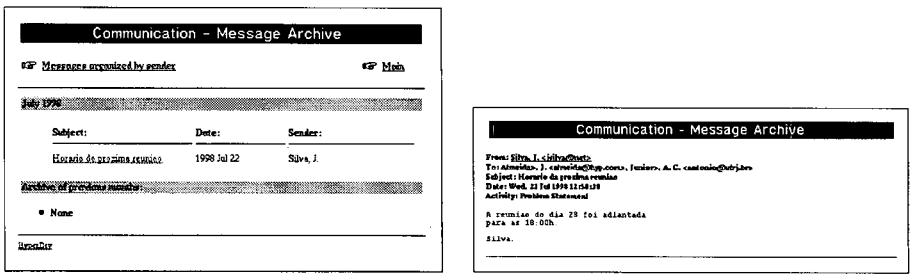
Para enviar uma nova mensagem, o usuário seleciona o nome dos destinatários dentre os integrantes dos grupos de trabalho. A tela de edição de mensagem (figura VI.27 (b)) é ativada através do botão *Edit*, onde o usuário insere o assunto e o texto da mensagem. O nome da atividade corrente é automaticamente inserida na mensagem.



(a) (b)

Figura VI.27: Módulo *Communication*: (a) tela principal; (b) edição de uma nova mensagem.

O repositório de mensagens as apresenta em ordem cronológica, organizadas por mês (figura VI.28 (a)). O sistema sempre apresenta as mensagens do mês atual. Para consultar as mensagens dos meses anteriores, o usuário deve selecionar o *link* correspondente ao mês. O usuário também pode consultar a lista de mensagens organizada por remetente. Os assuntos das mensagens correspondem a *links* de acesso ao texto da mensagem (figura VI.28 (b)).



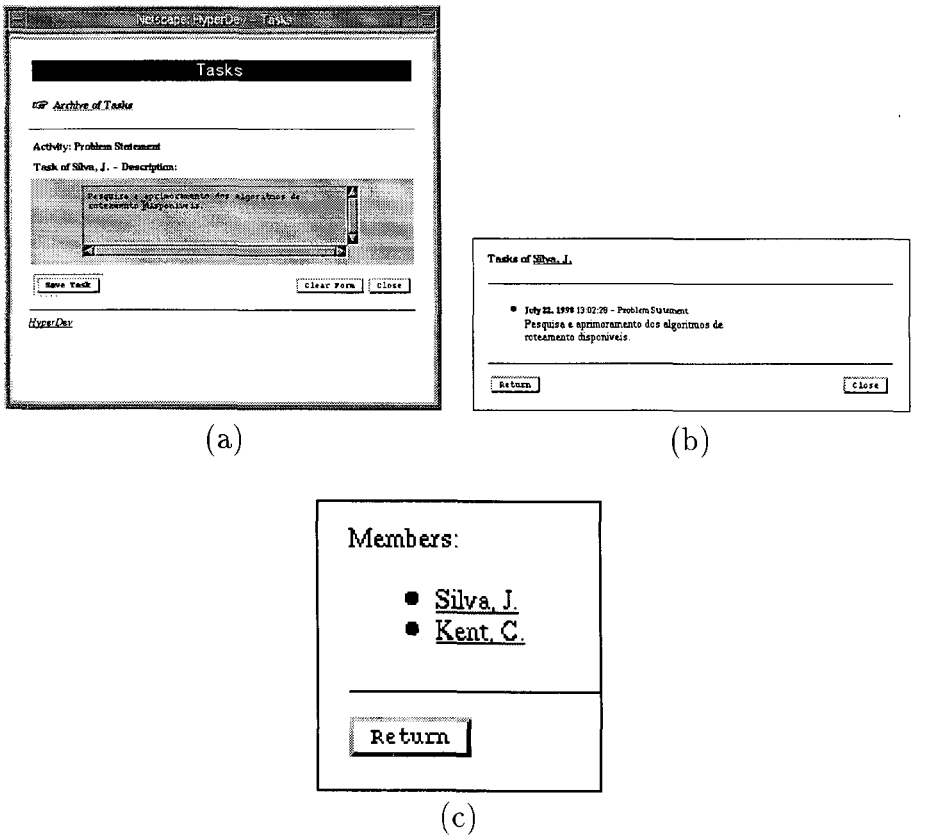
(a) (b)

Figura VI.28: Módulo *Communication*: (a) mensagens organizadas por data; (b) corpo da mensagem.

VI.3.7. Módulo de Tarefas

O módulo de tarefas mantém um registro das tarefas e é acionado através do botão *Tasks*, a partir da área principal de trabalho. A figura VI.29 (a) apresenta um formulário hipertextual através do qual um usuário pode inserir a descrição de uma tarefa. O botão *Save Task* grava a descrição da tarefa no repositório do sistema. O nome da atividade corrente, a data e hora são incluídos automaticamente no registro.

As tarefas individuais registradas podem ser consultadas por todos os integrantes do projeto. O sistema apresenta os nomes dos membros que já inseriram suas tarefas. As tarefas são apresentadas por ordem de data e horário, como apresentado na figura VI.29 (b). As tarefas de um determinado membro são apresentadas através da seleção do seu nome (figura VI.29 (c)).



(a) (b)

(c)

Figura VI.29: Módulo *Tasks*: (a) tela principal; (b) lista de tarefas do membro selecionado; (c) seleção do membro da equipe.

VI.3.8. Módulo de Busca de Informações

Este módulo é ativado pelo botão *Search*, na área principal de trabalho. A busca por palavra-chave (figura VI.30) é realizada em todas as informações cadastradas sobre as atividades do processo de software.

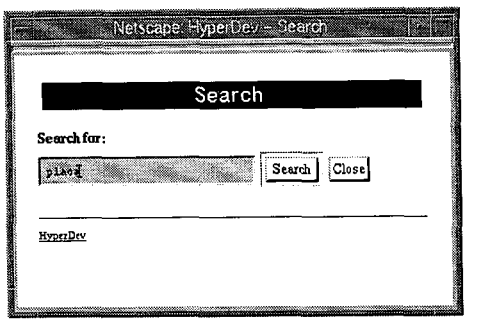
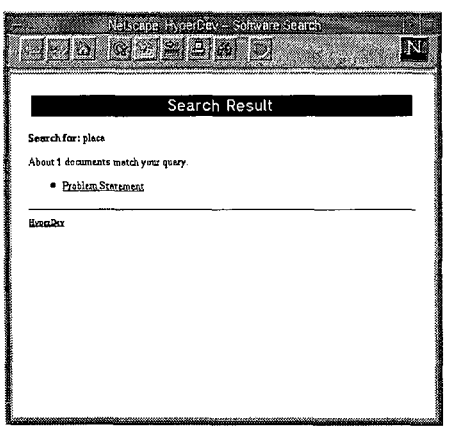
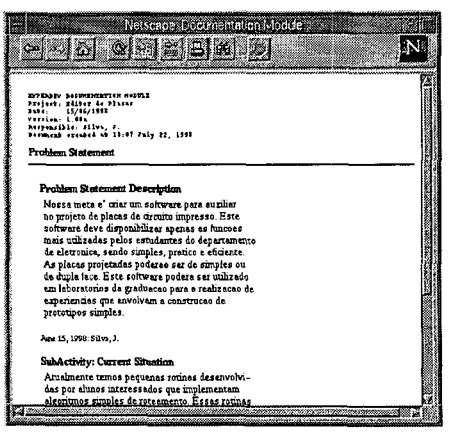


Figura VI.30: Módulo *Search*: tela principal.

Na figura VI.31 (a), o módulo apresenta o resultado da procura: a lista de atividades cujas informações contenham a palavra-chave. As informações de uma atividade do processo de software são apresentadas através da seleção de seu respectivo *link* (figura VI.31 (b)).



(a)



(b)

Figura VI.31: Módulo *Search*: (a) lista de atividades; (b) conteúdo da atividade.

VI.3.9. Módulo de Documentos Externos

Este módulo é ativado pelo botão *External Documents*, na área principal de trabalho. O módulo permite que o processo de software possa ser complementado

com fontes de informação externas ao projeto. Essas informações correspondem a qualquer arquivo disponível através da Internet. A tela principal deste módulo é apresentada na figura VI.32. No quadro à esquerda estão quatro botões que ativam as funções *List, Insert, Modify* e *Delete*.

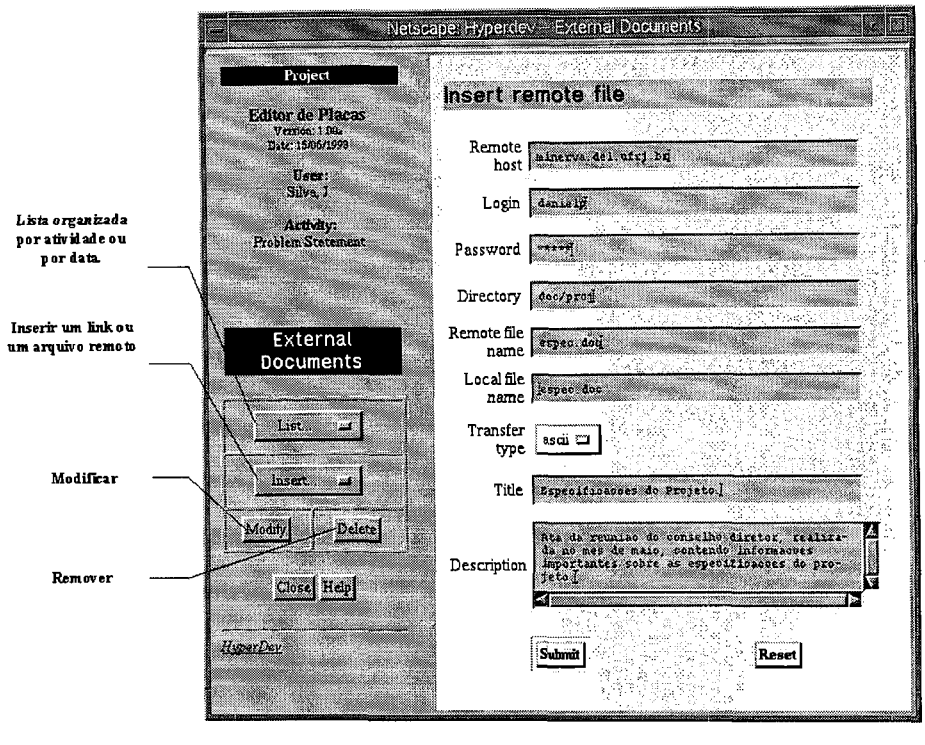


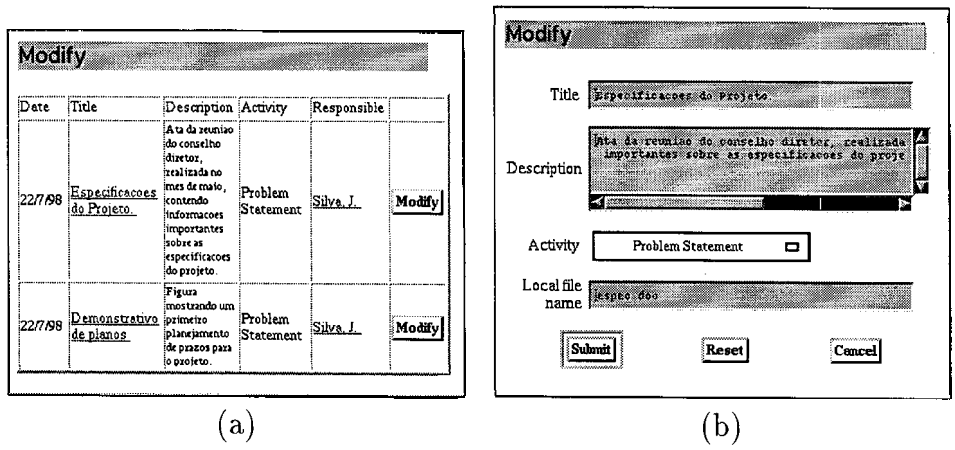
Figura VI.32: Módulo *External Documents*: tela principal / função *Insert*.

A função *List* apresenta os documentos registrados através deste módulo, organizados pela atividade à qual se associam ou pela data na qual foram inseridos no cadastro. A primeira opção apresenta as atividades do processo de software e, para cada uma, a lista dos documentos externos associados, juntamente com a sua descrição. O conteúdo do documento externo pode ser visualizado ou transferido para o cliente ao acionar o *link* correspondente. A segunda opção apresenta as seguintes informações: data, nome do documento, descrição, atividade associada e usuário responsável. O nome do documento, também neste caso, corresponde a uma ligação hipertextual para o seu conteúdo.

A função *Insert* permite inserir um documento externo no cadastro. O documento inserido é associado à atividade corrente do processo de software. Os documentos externos podem ser *links* ou *arquivos remotos*. Um *link* funciona como um indicador

para um arquivo na Web. Para visualizar o conteúdo deste tipo de documento, o navegador irá acessá-lo através de seu endereço original (URL). Um *arquivo remoto* é copiado, via *ftp*, para um repositório de documentos do projeto. O formulário para inserção de um documento do tipo *link* requer a URL correspondente, enquanto que para um documento do tipo *arquivo remoto*, é necessário fornecer informações que permitam a sua transferência via Internet. Para ambos os casos são necessários o nome e uma descrição, que servem para identificar o documento.

A função *Modify* permite modificar as propriedades de um documento cadastrado. Quando o usuário aciona esta função, o sistema apresenta os documentos externos registrados (figura VI.33 (a)). Ao lado de cada documento criado pelo usuário aparece o botão *Modify*, que dá acesso à tela de modificação (figura VI.33 (b)). Caso o usuário não seja aquele que inseriu o documento externo, a mensagem *"not owner"* aparecerá. As modificações possíveis são: alterar o nome do documento externo, a sua descrição e a atividade associada. Pode-se, também, modificar o URL, se o documento for do tipo *link*, ou o nome do arquivo armazenado no repositório do projeto, se o documento for do tipo *arquivo remoto*.



(a) (b)

Figura VI.33: Módulo *External Documents*. Função Modify: (a) seleção do documento externo e (b) tela de modificação.

A função *Delete* remove um documento externo do cadastro. Assim como na função *Modify*, os documentos registrados são apresentados e, ao lado de cada documento inserido pelo usuário aparecerá o botão *Delete*. Caso o usuário não seja aquele que inseriu o documento externo, a mensagem *"not owner"* aparecerá.

VI.3.10. Módulo de Documentação

O módulo de documentação é ativado através do botão *Documentation*. A tela principal deste módulo (figura VI.34) apresenta oito botões que determinam o tipo de documentação a ser gerada.

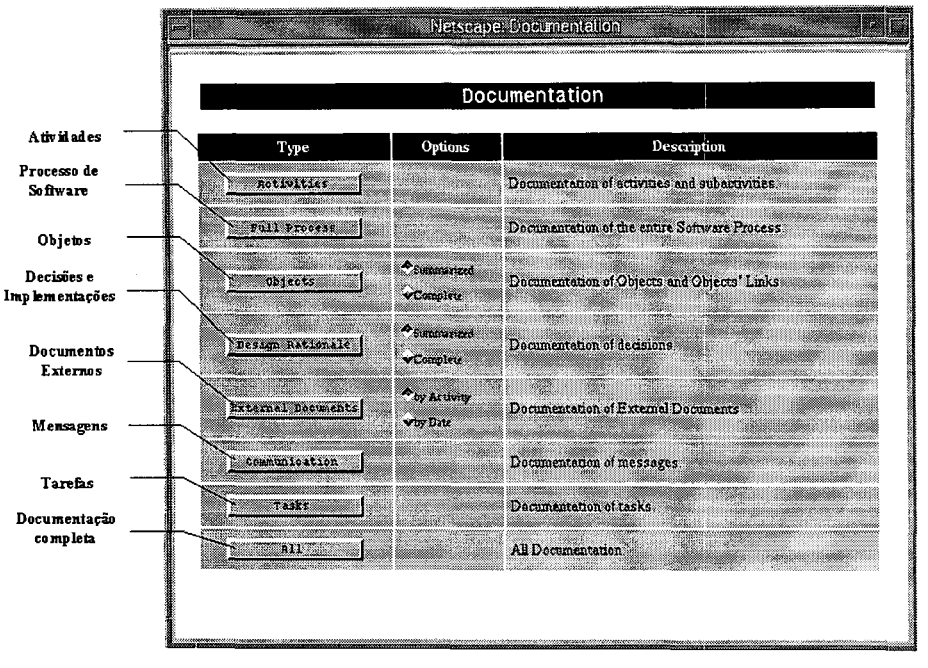


Figura VI.34: Módulo *Documentation*: tela principal.

A opção *Activities* permite a seleção da atividade do processo de software a ser documentada. A opção *Full Process* documenta todas as informações das atividades do processo de software. A opção *Object* documenta, de forma resumida ou completa, os objetos cadastrados. A documentação das *Decisões e Implementações* apresenta, de forma resumida ou completa, as informações sobre as decisões, implementações já realizadas, implementações previstas para a versão atual e implementações previstas para futuras versões registradas ao longo da construção do software. A documentação dos *Documentos Externos* apresenta os documentos registrados, classificados por atividade ou por data. A documentação de *Mensagens* apresenta as mensagens eletrônicas registradas através do módulo de comunicação, organizadas por remetente. A documentação de *Tarefas* apresenta as tarefas registradas, organizadas por data e por membro do grupo de trabalho. a opção *All* apresenta toda a documentação geradas pelas opções anteriores. A documentação

resultante é apresentada em uma tela separada (figura VI.35).

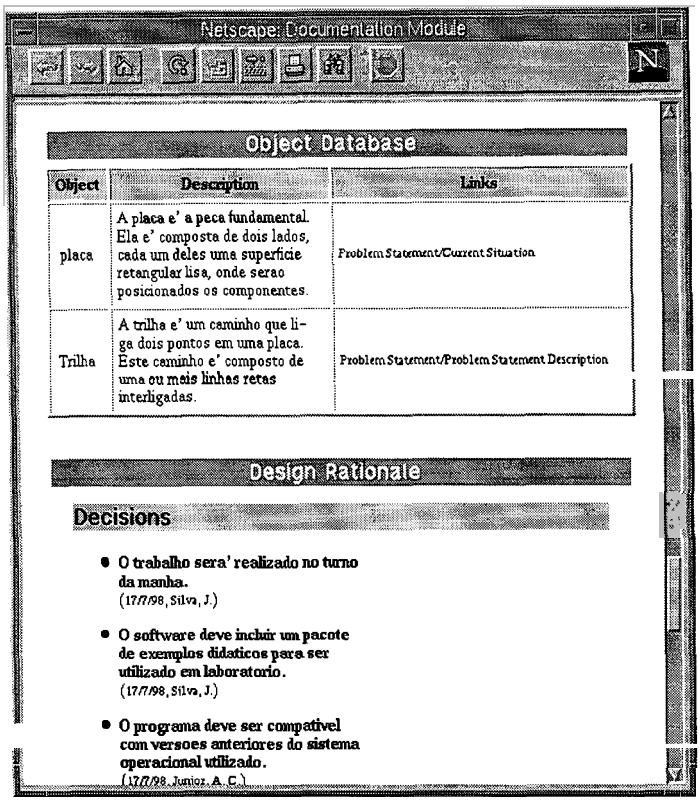


Figura VI.35: Módulo *Documentation*: documentação resultante.

VI.3.11. Verificação de Inconsistências

Este módulo apresenta possíveis inconsistências identificadas nos demais módulos de HyperDev. Ao ativar o botão *Report*, na área principal de trabalho, surge uma janela onde o resultado da validação é apresentado. Os tipos de inconsistências variam de acordo com cada módulo, conforme apresentado na tabela VI.3.

O sistema apresenta, inicialmente, um resumo das inconsistências, informando o número de problemas encontrados em cada módulo. Posteriormente, para cada módulo os problemas são detalhados. As inconsistências podem ser classificadas como aviso, erro ou grave. Um aviso é considerado algo que não compromete o funcionamento do projeto, como, por exemplo, deixar uma atividade incompleta. Uma URL de um documento externo inexistente corresponde a um erro. Problemas com o registro do projeto são classificados como graves. A figura VI.36 apresenta um exemplo do relatório de inconsistências.

Módulo	Inconsistências
Registro do Projeto	– Usuário sem email – Problemas com o registro do projeto
Processo de Software	– Atividades incompletas
Registro de Objetos	– Objetos sem links – *Links* vazios (sem nenhuma informação)
Registro de Decisões e Implementações	– Nenhuma decisão registrada – Nenhuma implementação registrada – Implementação classificada como prevista que ainda não tenha sido realizada
Documentos Externos	– URL inexistente – Arquivo inexistente no repositório
Registro de Tarefas	– usuários que não tenham registrado tarefas

Tabela VI.3: Módulo *Report*: lista das inconsistências.

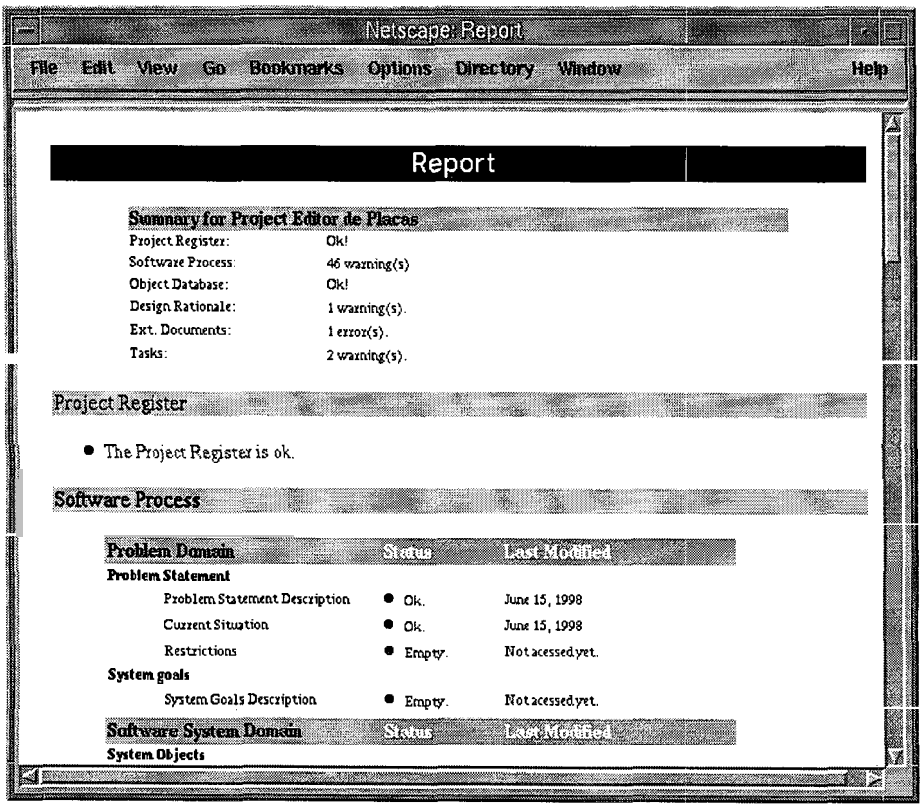


Figura VI.36: Módulo *Report*: lista das inconsistências.

VI.4. Comparação de HyperDev com outros Sistemas de Apoio ao Desenvolvimento de Software em Rede

Nesta seção, faz-se uma comparação entre HyperDev e os sistemas de apoio ao desenvolvimento de software em rede, apresentados na seção III.4.

O sistema DHT (seção III.4.1.) integra repositórios heterogêneos, permitindo uma conexão entre as fases de um processo de software, sendo que cada fase gera um repositório. Por manter o meta-conhecimento sobre o conteúdo de cada repositório, a busca aos componentes da aplicação é mais eficiente do que em HyperDev. O sistema HyperCASE (seção III.4.2.), assim como HyperDev, apóia engenheiros de software nas atividades de construção de aplicações computacionais. A meta do HyperCASE é conectar os documentos do software através da integração de ferramentas CASE. Embora eficiente, este sistema restringe os usuários a utilizarem as ferramentas CASE disponíveis através do próprio sistema. Os sistemas HyperText e HyperCASE utilizam técnicas de hipertextos mas se restringem a redes locais.

HyperCode (seção III.4.3.) produz uma representação hipertextual do código de programas. Além de permitir um melhor entendimento dos programas, este projeto também apóia equipes de trabalho cooperativas, oferecendo mecanismos para controle do compartilhamento e desenvolvimento de código.

A ferramenta LIGHT (seção III.4.4.) integra todos os documentos e subprodutos de um software gerando um único hipertexto. Pode ser utilizada após a geração dos documentos do software e, portanto, não apóia as atividades de um processo de software. LIGHT promove um melhor entendimento da aplicação em desenvolvimento ao conectar os mesmos conceitos presentes em diferentes documentos. Um outro ponto é que, a cada projeto de software, novos conversores devem ser implementados e as regras de conversão, modificadas. Assim como o sistema DHT, LIGHT só pode ser utilizado após a geração dos documentos do software.

O projeto ConceptTracker (seção III.4.5.) é o que mais se aproxima ao HyperDev por oferecer um protocolo aberto e extensível para registro, indexação, busca e acesso aos produtos do software. Porém, este projeto não coordena as atividades a

serem realizadas ao longo de um processo de software. ConceptTracker concentra-se nas tarefas técnicas da construção de uma aplicação computacional e não integra mecanismos de gerência do processo.

O projeto WISE (seção III.4.6.) apóia as atividades de gerência de um projeto de software através do rastreamento de problemas e do registro de mudanças. WHERE (seção III.4.7.) concentra-se somente na atividade de especificação dos requisitos. Tais projetos apóiam somente algumas fases do processo de desenvolvimento de software embora de forma mais completa que HyperDev (TESORIERO e ZELKOWITZ, 1998).

A tabela VI.4 apresenta uma comparação funcional entre os sistemas anteriores, descrevendo os principais mecanismos de cada um e as funções que são oferecidas pelos demais para apoiar os diferentes aspectos de um desenvolvimento de software realizado por equipes geograficamente dispersas.

O desenvolvimento de HyperDev considerou as características dos sistemas e projetos descritos anteriormente. As restrições de tais sistemas foram consideradas e possíveis integrações, analisadas. A ferramenta LIGHT, por exemplo, pode ser integrada a HyperDev: a produção dos subprodutos do software é apoiada pelo HyperDev e a sua conexão é realizada por LIGHT.

A comparação com os sistemas hipertextuais de apoio ao desenvolvimento de software permitiu verficar que muitos dos sistemas existentes são destinados a aspectos específicos do processo de software. Enquanto que tais sistemas procuram ser completos e destinados a determinadas tarefas, o HyperDev oferece uma maior flexibilidade e demonstra ser mais adequado às tarefas gerenciais. Uma demonstração de tal afirmação refere-se à documentação gerada pelo sistema que permite que a gerência do processo identifique falhas nos processos especializados, apoiando, então, a melhoria contínua da capacidade de cada equipe e da maturidade do processo padrão da organização, como um todo.

	Integração de Repositórios	Integração de Ferramentas	Codificação Distribuída	Integração de Documentos	Integração de Artefatos	Gerência de Mudanças	Rastrea-mento de Requisitos
HyperDev	ligações hipertextuais documentos	-	ligações hipertextuais componentes	ligações hipertextuais documentos	ligações hipertextuais componentes	registro de decisões	ligações hipertextuais componentes
DHT	apoio completo	-	nós e ligações hipertextuais	integração repositórios heterogêneos	integração repositórios heterogêneos	nós e ligações hipertextuais	nós e ligações hipertextuais
HyperCase	repositório único de documentos	apoio completo	ligações hipertextuais documentos	repositório único de documentos	repositório único de documentos	integração CASE para mudanças	integração CASE para rastreamento
HyperCode	-	-	apoio completo	-	-	ligações hipertextuais de versões	-
LIGHT	ligações hipertextuais documentos	-	ligações hipertextuais documentos	apoio completo	ligações hipertextuais documentos	ligações hipertextuais documentos	ligações hipertextuais documentos
Concept Tracker	ligações hipertextuais documentos	-	ligações hipertextuais documentos	ligações hipertextuais documentos	apoio completo	-	ligações hipertextuais documentos
WISE	-	-	-	-	-	apoio completo	-
WHERE	-	-	-	-	-		apoio completo

Tabela VI.4: Comparação dos Sistema de Apoio ao Desenvolvimento de Software em Rede.

VI.5. Considerações Finais

Este capítulo apresentou o sistema HyperDev que apóia a utilização do modelo de gerência de processos de software para equipes geograficamente dispersas. O sistema facilita a padronização do processo de software ao permitir a sua definição de forma adequada a cada equipe de trabalho. Através de HyperDev, todas as atividades do desenvolvimento do software podem ser definidas e monitoradas pela Internet. Tal facilidade permite a gerência e a distribuição de tarefas entre os membros das equipes do projeto. O sistema também possibilita a integração dos diversos documentos produzidos ao longo da construção do software, gerando a documentação de todo

o processo de software. O registro de informações do software, decisões tomadas, itens a serem implementados e mensagens encaminhadas aos integrantes dos grupos de trabalho facilita o entendimento e a gerência do processo de software, como um todo, além de servir de base para futuras manutenções e extensões da aplicação em desenvolvimento.

O HyperDev foi utilizado no ambiente acadêmico, o que permitiu a sua validação. A meta do projeto "Cam1K" do Laboratório Nacional de Astrofísica é desenvolver um software de controle para instrumentos astronômicos, envolvendo seis profissionais que realizam suas atividades em diferentes localizações nas cidades de Itajubá e Brasópolis, em Minas Gerais. Alunos do curso de engenharia eletrônica da UFRJ definiram dois projetos a serem apoiados pelo sistema. O objetivo do projeto "Editor de Placas" é desenvolver um software para a edição de placas de circuito impresso no computador. O projeto "Laboratório de Eletrônica I" está relacionado à área educacional e o seu escopo é utilizar o HyperDev como ferramenta de apoio ao ensino, determinando as atividades de cada aluno e monitorando seus resultados. Finalmente, o HyperDev foi utilizado pelo grupo de aquisição de dados do ATLAS para desenvolver o sistema *on-line* do detetor.

A utilização do HyperDev nos projetos citados anteriormente permitiu verificar que os objetivos primários do sistema foram atingidos e que os requisitos gerais, funcionais e de interface com o usuário foram respeitados. As deficiências e possíveis melhorias também foram identificadas e serão apresentadas no capítulo das conclusões da tese.

O HyperDev permitiu verificar que a aplicação do modelo de gerência proposto é viável e pode ser facilitada com a utilização de ferramentas e tecnologia adequadas.

VII. Conclusões

Neste capítulo final, as conclusões da pesquisa realizada são apresentadas. Descrevem-se suas contribuições e propostas para trabalhos futuros.

VII.1. Visão Geral da Pesquisa Realizada

A qualidade dos produtos de software está diretamente relacionada à qualidade e adequação do processo de software utilizado para construi-los (CURTIS *et al.*, 1992, ELLMER, 1995, HALEY, 1996, CURTIS, 1997, HUMPHREY, 1995, PAULK *et al.*, 1997, EMAM *et al.*, 1998). Portanto, a melhoria contínua do processo de software corresponde a um dos maiores objetivos das empresas de software. Muitas empresas investem na aquisição de novas tecnologias e no treinamento de profissionais. Entretanto, ferramentas CASE e equipamentos sofisticados não garantem o aprimoramento de processos de software. A grande maioria dos gerentes não procura entender as restrições do ambiente de desenvolvimento e tampouco considera o conhecimento e a experiência das equipes de trabalho antes de definir métodos e processos adequados.

As colaborações geograficamente distribuídas estão surgindo em grande número devido à Internet, que permite a realização de trabalhos por equipes que não necessariamente se encontram em um mesmo lugar. A alta conexão mundial entre computadores permitiu que os programas pudessem ser facilmente distribuídos e amplamente utilizados. Entretanto, por trabalharem em diferentes localizações, os profissionais utilizam diversos equipamentos e métodos e, dificilmente, as equipes são homogêneas. Tudo isso torna difícil e inadequada a determinação para uso de um único processo de software.

Gerência do processo de software é definida como a aplicação de conceitos, técnicas e práticas da engenharia de software para explicitamente monitorar, controlar e melhorar processos de software. Processos de software bem definidos e

gerenciados permitem melhorar significativamente a qualidade dos seus produtos. Os modelos de maturidade caracterizam a capacidade de uma organização em desenvolver software. Quando o nível de maturidade é conhecido, as ações necessárias para amadurecer o processo podem ser determinadas. Atualmente, existem diversos modelos para avaliar a maturidade de processos de software e identificar áreas-chave para permitir o seu aprimoramento. Cada um dos modelos contribui, de alguma forma, para aprimorar a qualidade de processos de software. Todos os modelos foram fortemente influenciados pelos mesmos princípios de qualidade e os primeiros modelos influenciaram os mais recentes. Os pontos em comum entre estes modelos e padrões são: inexistência de um processo de software adequado a qualquer projeto, a importância das medições, a necessidade de avaliações e a determinação da maturidade dos processos como um mecanismo para definir planos de ação para o seu aprimoramento.

Para assegurar a construção do software realizada por uma colaboração mundial, é imprescindível definir processos de software adequados. Um processo de software é essencial para que um profissional participe do desenvolvimento ordenado de um sistema computacional, definindo a maneira pela qual uma aplicação deve ser construída. No caso específico do experimento ATLAS, o processo de software especifica os requisitos mínimos mas não indica diretrizes para melhorá-lo. Apesar de ter uma estrutura consistente, o processo de software não permite ajustes às necessidades específicas de cada equipe de software.

Um processo de software bem definido e gerenciado traz os seguintes benefícios: redução dos custos de desenvolvimento, eliminação de trabalhos duplicados, previsão de cronogramas e orçamentos mais apurada e geração de produtos de software de melhor qualidade. A melhoria contínua dos processos de software é uma meta que deve estar presente nas organizações de software devido às demandas do mercado atual.

VII.2. Contribuições da Pesquisa

A pesquisa apresentada nesta tese enfocou a importância dos processos de software na qualidade dos produtos. Muitos trabalhos foram realizados sobre definição, modelagem, integração, execução, avaliação, medição e melhoria de processos. Aprimorar o controle de processos e a sua qualidade parece ser uma boa idéia em qualquer ambiente, entretanto, deve-se, em primeiro lugar, verificar se o processo de software é apropriado para o grupo de trabalho que deverá utilizá-lo.

O processo de software padrão para equipes geograficamente dispersas atende os requisitos que foram determinados através da identificação das características de tais ambientes. A pesquisa realizada permitiu identificar nove requisitos básicos a serem considerados na definição de qualquer processo de software a ser utilizado por equipes distribuídas. As dificuldades presentes no desenvolvimento disperso de software foram agrupadas, definindo as seguintes classes de problemas: coordenação de equipes, coordenação das atividades, controle de artefatos e apoio à comunicação. Embora possam estar presentes em outros contextos, os problemas são agravados quando as equipes são heterogêneas, ou seja, quando os grupos de trabalho dispõem de diferentes recursos econômicos, humanos e tecnológicos. O processo de software padrão proposto seguiu a norma ISO/IEC 12207, amplamente utilizada, redefinindo as suas categorias de processos e introduzindo outras categorias para atender as necessidades específicas das equipes de trabalho dispersas. Os problemas, requisitos e categorias do processo de software foram relacionados, permitindo verificar se todos os requisitos foram atendidos por um ou mais aspectos do processo padrão e se todas as classes de problemas foram consideradas na determinação dos requisitos.

O modelo de gerência de processos de software para equipes geograficamente dispersas tem por objetivo apoiar o desenvolvimento de sistemas computacionais realizado por grupos de trabalho heterogêneos, geograficamente dispersos, cultural e tecnologicamente diferentes. As principais metas são: garantir que projetos distribuídos possam se realizar e tenham êxito mesmo que as equipes tenham diferentes níveis de maturidade, melhorar a capacidade de cada grupo de trabalho e aprimorar

o processo de software de uma organização. A pesquisa realizada baseou-se na premissa de é que melhor adaptar processos de software às equipes do que impô-los aos profissionais. Esta adaptação corresponde à especialização do processo de software padrão em quatro processos, cada qual correspondendo a um nível de maturidade. Para especializar o processo de software padrão, foi necessário determinar um modelo de maturidade de referência e, então, o CMM foi adaptado para atender às necessidades do desenvolvimento de software geograficamente disperso. A correspondência entre as categorias do processo padrão e as áreas-chave do modelo de maturidade de referência foram identificadas com o intuito de determinar que aspectos do processo de software devem ter maior atenção para atender as áreas-chave de um nível de maturidade.

No contexto filosófico da pesquisa, a qualidade dos produtos de software é fortemente influenciada pela qualidade dos processos utilizados para construí-los. O modelo de maturidade identifica as áreas-chave que levam ao amadurecimento das equipes de trabalho, ou seja, permitem determinar como aprimorar a sua capacidade (ou habilidade) em desenvolver um produto de software. Através das relações entre as áreas-chave e as categorias do processo padrãop, tal identificação também permite determinar que aspectos do processo de software necessitam de maior atenção (controle ou investimento) para melhorar a capacidade da própria organização. A etapa de especialização do processo padrão possibilita que a proposta de melhoria contínua seja aplicada a equipes heterogêneas. Além disso, o processo de software de uma organização é aprimorado baseando-se na experiência da sua própria utilização.

O sistema Hyperdev, que implementa o acompanhamento do modelo de gerência, apóia a monitoração de projetos de software realizado por equipes dispersas. Através da Internet, as informações do software podem ser facilmente registradas e acessadas, e, posteriormente, utilizadas durante o processo de avaliação. O sistema oferece os seguintes mecanismos: orientação da seqüência das atividades a serem realizadas, padronização do processo de software, flexibilidade quanto à escolha de ferramentas, independência da plataforma de desenvolvimento, distribuição de tarefas pela rede de computadores e integração das atividades técnicas ao trabalho administrativo e

gerencial. O HyperDev permitiu verificar que a aplicação do modelo de gerência de processos de software para equipes geograficamente dispersas é viável e pode ser facilitada com a utilização de ferramentas e tecnologia adequadas.

VII.3. Direções Futuras

A aplicação do modelo de gerência em outros contextos pode exigir adaptações em alguns de seus aspectos. Algumas práticas do modelo devem ser julgadas para que sejam realizadas adequadamente, considerando as especificidades do ambiente de desenvolvimento de software. A utilização do modelo de gerência de forma eficaz e correta requer um entendimento de como o modelo é estruturado para ser aplicado em diferentes propósitos. Tais considerações podem contribuir para futuras extensões do modelo de gerência de processos de software, ou ainda, possíveis especializações. Provavelmente, a aplicação do modelo em domínios de conhecimento específicos irá requerer uma adaptação do processo padrão e do modelo de maturidade de referência.

Os procedimentos do modelo de gerência para aprimorar a qualidade de processos de software foram baseados no modelo CMM do SEI. A escolha deste modelo de maturidade foi motivada por já ter sido utilizado, referenciado e avaliado em diversos trabalhos. Entretanto, outros modelos de maturidades podem ser utilizados.

A instanciação de especializações do processo pode ser apoiada através de linguagens para modelar e executar processos. Fragmentos das instâncias podem ser reutilizados, após uma avaliação da sua qualidade em determinado nível de maturidade e da sua adequação ao domínio da aplicação.

O sistema HyperDev pode ser estendido para apoiar a especialização de processos, a determinação da capacidade de cada grupo de trabalho e apontar deficiências, indicando as atividades para melhorar a capacidade de cada grupo de trabalho e aprimorar o processo de software padrão.

Os profissionais poderão encontrar dificuldades durante a aplicação do modelo de gerência de processos de software para equipes geograficamente dispersas. Futu-

ras pesquisas permitirão identificar alguns problemas, suas implicações e possíveis soluções.

O estudo e a utilização de diversas tecnologias – tanto na construção do HyperDev quanto em outros projetos de pesquisa – permitiu verificar que o desenvolvimento e a integração de novos mecanismos ao HyperDev é viável. A seguir, descrevem-se algumas funções que podem ser implementadas nas futuras versões do sistema.

Contole de Versões: o sistema também pode coordenar a manipulação das diferentes versões dos documentos, organizando-os em um repositório principal e mantendo uma cópia independente para cada usuário. CVS (*Concurrent Versions System*) é uma ferramenta Unix para manipular versões do software e realizar o controle das entregas em ambientes multi-usuário. CVS oferece a funcionalidade necessária para gerenciar trabalhos concorrentes, permite recuperar as versões precedentes através da sua data ou identificações simbólicas, oferece um ambiente de edição aberto, utilizando algoritmos para resolução de conflitos.

Banco de Dados Relacional: um sistema de gerência de banco de dados relacional pode ser integrado ao sistema para armazenamento, recuperação e manipulação de informações. Comandos SQL (*Structured Query Language*) podem ser executados no servidor, oferecendo uma interface entre os usuários e as bases de dados do HyperDev. A entrada de dados pode ser realizada através de formulários HTML e o resultado é apresentado em páginas hipertextuais.

Ferramentas Gráficas: o sistema pode ser estendido para integrar ferramentas que produzam informações gráficas. Os diagramas gerados por ferramentas CASE, durante as atividades de especificação e projeto, contém pouco texto, mas cada elemento gráfico contém importantes conceitos que podem ser associados com outros documentos no módulo "Registro de Objetos" do HyperDev.

Geração de Hipertextos: conversores automáticos facilitam a identificação de nós e ligações, classificando-os por tipos e capturando sua semântica. O HyperDev pode ser integrado à ferramenta LIGHT (seção III.4.4.), que oferece regras configuráveis para mapear o conteúdo de documentos em arquivos hipertextuais.

Linearização de Hipertextos: o problema inverso à conversão de textos em hipertextos corresponde à linearização de documentos hipertextuais. Documentos impressos se diferem dos hipertextos e, portanto, ferramentas para a produção de ambos os tipos devem considerar as suas diferenças. A formatação das páginas ou nós (cabeçalhos, numerações, etc.) oferece informação semântica adicional. Os desenvolvedores podem definir e registrar vários formatos e associá-los aos *templates* do HyperDev. Desta maneira, diferentes ações podem ser realizadas nos processos de geração de hipertextos e impressão dos documentos.

Bibliografia

ABNT, 1993, *Normas de gestão da qualidade e garantia da qualidade – Parte 3: Diretrizes para a aplicação da NBR 19001 ao desenvolvimento, fornecimento e manutenção de "software"*, Novembro. Associação Brasileira de Normas Técnicas.

AIMAR, A., AIMAR, M., CATTANEO, M., ILLAS, P. C., KHODABANDEH, A., PALAZZI, P., ROUSSEAU, B., RUGGIER, M., 1995, "Using WWW to Improve Software Development and Maintenance: Application of the LIGHT System to ALEPH Programs". In: *Proceedings of the Computing in High Energy Physics'95 Conference*, Rio de Janeiro, Brasil, Setembro.

AIMAR, A., BARONE, L. M., KRYSIUK, P., P. PALAZZI, ROUSSEAU, B., 1997, "Light/OO: an Extension of Light to OO Development". In: *Proceedings of the Computing in High Energy Physics'97 Conference*, Berlim, Alemanha, Abril.

ANDREESSEN, M., 1993, *Getting Started with NCSA Mosaic.* In: Report, National Center for Supercomputing Applications (NCSA), Maio.

ANDREWS, K., 1996, "Applying Hypermedia Research to the World-Wide Web". In: *Proceedings of the Hypermedia Research and the World-Wide Web Workshop*, ACM Hypertext'96, Washington, DC, EUA, Março.

`http://www.cs.bgsu.edu/hrweb/papers/andrews.html`

ANSI, 1999, *American National Standards Institute (ANSI).*

`http://www.ansi.org/broch1.html`

AOYAMA, M., 1998a, "Agile Software Process and Its Experience". In: *Proceedings of the 20^{th} International Conference on Software Engineering*, pp. 3–12, Kyoto, Japão, Abril.

AOYAMA, M., 1998b, "Web-Based Agile Software Development", *IEEE Software*, v. 15, n. 6, pp. 56–65, Novembro/Dezembro.

ARNOLD, J. E., 1995, "Toward Collaborative Software Process". In: *Proceedings of the 17th International Conference on Software Engineering*, pp. 107–109, Seattle, Washington, EUA, Abril.

AROCENA, G. O., MENDELZON, A. O., 1998, "Viewing WISs as Database Applications", *Communications of the ACM*, v. 41, n. 7, pp. 101–102, Julho.

ARRUDA, M. N. P., VILLAS-BOAS, A., CÔRTES, M. L., MARTINI, M. R. B., 1998, "Avaliação CMM – A Experiência CPqD Telebrás". In: *Proceedings of the Workshop de Qualidade de Software*, pp. 57–66, Maringá, Paraná, Brasil, Outubro.

ARTHUR, L. J., 1993, *Improving Software Quality: An Insider's Guide to TQM.* John Wiley & Sons, Inc.

ARTHUR, L. J., 1997, "Quantum Improvements in Software System Quality", *Communications of the ACM*, v. 40, n. 6, pp. 46–52, Junho.

ATLAS, 1999, *ATLAS Home Page.*

`http://www.cern.ch/Atlas/`

ATLAS COLLABORATION, 1994, *ATLAS Technical Proposal.* In: Report CERN/LHCC/94-43, CERN, Genebra, Suíça, Dezembro.

ATLAS SOFTWARE COLLABORATION, 1996a, *ATLAS Computing Technical Proposal.* In: Report CERN/LHCC/96-43, CERN, Genebra, Suíça, Dezembro.

ATLAS SOFTWARE COLLABORATION, 1996b. *The ATLAS Offline Software Manual.* CERN, Genebra, Suíça.

`http://www.cern.ch/Atlas/GROUPS/SOFTWARE/DOCUMENTS/document.html`

ATRIUM, 1997, *The USC ATRIUM Laboratory.* University of Southern California.

`http://www.usc.edu/dept/ATRIUM/`

AVRILIONIS, D., BELKHATIR, N., CUNIN, P., 1996a, "Improving software process modelling and enactment techniques". In: *Proceedings of the 5th European Workshop on Software Process Technology (EWSPT'5)*, Nancy, França, Outu-

bro.

http://www-lsr.imag.fr/Les.Personnes/Denis.Avrilionis/PUBLICATIONS/

AVRILIONIS, D., CUNIN, P., BELKHATIR, N., 1996b, "A Unified Framework for Software Process Enactement and Improvement". In: *Proceedings of the 4th International Conference on Software Process (ICSP'4)*, Brighton, UK, Dezembro.

http://www-lsr.imag.fr/Les.Personnes/Denis.Avrilionis/PUBLICATIONS/

AVRILIONIS, D., CUNIN, P., FERNSTRÖM, C., 1996c, "OPSIS: A View Mechanism for Software Process which Supports their Evolution and Reuse". In: *Proceedings of the 18th International Conference on Software Engineering*, pp. 38–47, Berlim, Alemanha, Março.

http://www-lsr.imag.fr/Les.Personnes/Denis.Avrilionis/PUBLICATIONS/

BACH, J., 1998, "Microdynamics of Process Evolution", *IEEE Computer*, pp. 111–113, Fevereiro.

BAENTSCH, M., MOLTER, G., STURM, P., 1995, "WebMake: Integrating Distributed Software Development in a Structure-Enhanced Web". In: *Proceedings of the 3rd International WorldWide Web Conference*, Darmstadt, Alemanha, Abril.

BALASUBRAMANIAN, V., 1999, *State of Art Review on Hypermedia Issues and Applications.*

http://www.isg.sfu.ca/~duchier/misc/hypertext_review/index.html

BALASUBRAMANIAN, V., BASHIAN, A., 1998, "Document Management and Web Technologies: Alice Marries the Mad Hatter", *Communications of the ACM*, v. 41, n. 7, pp. 107–115, Julho.

BANDINELLI, S., *ET AL.*, 1995a, "Modeling and Improving an Industrial Software Process", *IEEE Transactions on Software Engineering*, v. 21, n. 5, pp. 440–454, Maio.

BANDINELLI, S., BARESI, L., FUGGETTA, A., LAVAZZA, L., 1993a, "Requirements And Early Experiences In The Implementation Of The SPADE Repository". In: *Proceedings of the 8th International Software Process Workshop*, pp. 30–32, Wadern, Alemanha, Março.

BANDINELLI, S., DINITTO, E., FUGGETTA, A., 1996, "Supporting Cooperation in the SPADE-1 Environment", *IEEE Transactions on Software Engineering*, v. 22, n. 12, Dezembro.

`ftp-se.elet.polimi.it/dist/Papers/ProcessModeling/`

BANDINELLI, S., DINITTO, E., FUGGETTA, A., LAVAZZA, L., 1995b, "Coupled vs Decoupled User Interaction Environments in PSEEs". In: *Proceedings of the 9th International Software Process Workshop*, pp. 50–52, Airlie, VA, Outubro.

BANDINELLI, S., FUGGETTA, A., GHEZZI, C., 1993b, "Software Process Model Evolution in the SPADE Environment", *IEEE Transactions on Software Engineering*, v. 19, n. 12, pp. 1128–1144, Dezembro.

`ftp-se.elet.polimi.it/dist/Papers/ProcessModeling/`

BANDINELLI, S., FUGGETTA, A., LAVAZZA, L., PICCO, G. P., 1995c, "Combining Control and Data Integration in the SPADE-1 Process-centered Software Engineering Environment". In: *Proceedings of the 9th International Software Process Workshop*, pp. 96–99, Airlie, VA, Outubro.

BARGHOUTI, N. S., FEILER, P. H., 1993, "Demonstration Experience Report Session Summary". In: *Proceedings of the 8th International Software Process Workshop*, pp. 2–5, Wadern, Alemanha, Março.

BARGHOUTI, N. S., KRISHNAMURTHY, B., 1993, "An Open Environment for Process Modeling and Enactment". In: *Proceedings of the 8th International Software Process Workshop*, pp. 33–35, Wadern, Alemanha, Março.

BARRY, A., 1994, "A WWW-Based Information Management System for NASA Projects". In: *Proceedings of the Second International World-Wide Web Conference: Mosaic and the Web*, Chicago, Illinois, EUA, Outubro.

`href="http://research.ivv.nasa.gov/docs/techreports/`

BASILI, V. R., 1993, "The Future Engineering of Software: A Management Perspective". In: Reifer, D. J. (ed), *Software Management*, 4 ed., IEEE Computer Society Press.

BASILI, V. R., 1996, "The Role of Experimentation in Software Engineering: Past, Current, and Future". In: *Proceedings of the 18th International Conference on Software Engineering*, pp. 442–449, Berlim, Alemanha, Março.

BEE, C., MAIDANTCHIK, C., *ET AL.*, 1995a, "Applications of an OO Methodology and CASE to a DAQ system". In: *Proceedings of the Computing in High Energy Physics'95 Conference*, Rio de Janeiro, Brasil, Setembro.

BEE, C., MAIDANTCHIK, C., *ET AL.*, 1995b, "Experience Using a Distributed Object Oriented Database for a DAQ system". In: *Proceedings of the Computing in High Energy Physics'95 Conference*, Rio de Janeiro, Brasil, Setembro.

BERNERS-LEE, T., OTHERS, 1996, *Hypertext Transfer Protocol – HTTP 1.0.* In: Report Network Working Group, W3C.

BIEBER, M., KACMAR, C., 1995, "Designing Hypertext Support for Computational Applications", *Communications of the ACM*, Agosto.

BLACKBURN, J. D., SCUDDER, G. D., WASSENHOVE, L. N. V., 1996, "Improving Speed and Productivity of Software Development: A Global Survey of Software Developers", *IEEE Transactions on Software Engineering*, v. 22, n. 12, pp. 875–884, Dezembro.

BOEHM, B., 1996, "Anchoring the Software Process", *IEEE Software*, pp. 73–82, Julho.

BOLCER, G., KAISER, G., 1999, "SWAP: Leveraging the Web to Manage Workflow", *IEEE Internet Computing*, v. 3, n. 1, pp. 22–24, Janeiro/Fevereiro.

BOLCER, G. A., 1996, "Endeavors: A Process System Integration Infrastructure". In: *Proceedings of the 4th International Conference on Software Process (ICSP'4)*, Brighton, UK, Dezembro.
`http://www.ics.uci.edu/~gbolcer/`

BOURDON, M., 1993, "Building Process Models using Process Weaver: a Progressive Approach". In: *Proceedings of the 8th International Software Process Workshop*, pp. 40–42, Wadern, Alemanha, Março.

BRIAND, L., MELO, W., SEAMAN, C., BASILI, V., 1995, "Characterizing and Assessing a Large-Scale Software Maintenance Organization". In: *Proceedings of the 17th International Conference on Software Engineering*, pp. 133–143, Seattle, Washington, EUA, Abril.

BRÖCKERS, A., DIFFERDING, C., THREIN, G., 1996, "The Role of Software Process Modeling in Planning Industrial Measurement Programs". In: *Proceedings of the The 3rd International Software Metrics Symposium*, Berlim, Alemanha, Março.

BROWN, J., HARRIS, J., KARBINER, L., POLETTO, M., DEHON, A., ESLICK, I., KNIGHT, T. F., 1994, "HyperCode, Global Cooperative Computing". In: *Proceedings of the Second International World-Wide Web Conference: Mosaic and the Web*, Chicago, Illinois, EUA, Outubro.

`http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/`

BRUEGGE, B., DUTOIT, A. H., 1997, "Communication Metrics for Software Development". In: *Proceedings of the 19th International Conference on Software Engineering*, pp. 271–281, Boston, Massachusetts, EUA, Maio.

CALLAHAN, J., RAMAKRISHNAN, S., 1998, *Software Project Management and Measurement on the World-Wide Web (WWW)*. In: Report NASA-IVV-96-005, NASA/WVU Software IV&V Facility, National Aeronautics and Space Administration (NASA), Março.

`http://research.ivv.nasa.gov/projects/WISE/index.html`

CALLAHAN, J. R., KHATSURIYA, R. R., HEFNER, R., 1998, "Web-Based Issue Tracking for Large Software Projects", *IEEE Internet Computing*, v. 2, n. 5, pp. 25–33, Setembro/Outubro.

CANALS, G., CHAROY, F., GODART, C., MOLLI, P., 1995, "P-Root & COO: Building a Cooperative Software Development Environment". In: *Proceedings of the Software Engineering Environments Conference*, IEEE Computer Society, pp. 2–10, Nordwijkerhout, Holanda, Abril.

`http://gille.loria.fr:7000/see95/see95.html`

CANDLIN, R., 1996, *An Introduction to the ATLAS Software Process.* ATLAS Internal Note SOFT-NO-026, CERN, Genebra, Suíça, Junho.

CARVALHO, M. B., FROSSARD, R. S., PADILHA, A. M., 1997, "Certificação X Melhoria". In: *Proceedings of the Workshop Qualidade de Software - XI Simpósio Brasileiro de Engenharia de Software*, pp. 1–6, Fortaleza, Ceará, Brasil, Outubro.

CATTANEO, F., FUGGETTA, A., LAVAZZA, L., 1995, "An Experience in Process Assessment". In: *Proceedings of the 17th International Conference on Software Engineering*, pp. 115–121, Seattle, Washington, EUA, Abril.

CERI, S., FRATERNALI, P., GEVINTI, S., PARABOSCHI, S., 1998, "Building a Database Design Laboratory on the Internet", *IEEE Internet Computing*, v. 2, n. 5, pp. 41–52, Setembro/Outubro.

CERN, 1999, *CERN Home Page.*

`http://www.cern.ch`

CHEN, J. J., 1997, "CSPL: An Ada95-Like, Unix-Based Process Environment", *IEEE Transactions on Software Engineering*, v. 23, n. 3, pp. 171–184, Março.

CHENG, J., 1994, "A Reusability-Based Software Development Environment", *ACM Software Engineering Notes*, v. 19, n. 2, pp. 57–62, Abril.

CIANCARINI, P., KNOCHE, A., TOLKSDORF, R., VITALI, F., 1996, "PageSpace: An Architecture to Coordinate Distributed Applications on the Web". In: *Proceedings of the Fifth International World-Wide Web Conference*, Paris, França, Maio.

`http://http://www5conf.inria.fr/fich_html/papers/P5/Overview.html`

CMS, 1997, *CONFIGURATION MANAGEMENT SYSTEM (CMS).* Electronic Warfare Associates, Inc., 1000 Technology Drive, Suite 3210, Fairmont, West Virginia 26554, EUA.

`http://wv.ewa.com/`

COALLIER, F., 1996, "The Assessment and Management of Software Products's Procurement Risks". In: *Proceedings of the VII CITS, Conferência Internacio-*

nal de Tecnologia de Software: Qualidade de Software, Centro Internacional de Tecnologia de Software (CITS), pp. 3–16, Curitiba, Paraná, Brasil, Junho.

COAR, K. A. L., OTHERS, 1998, *The Common Gateway Interface Version 1.1.* In: Report, W3C.

CONRADI, R., JACCHERI, M. L., MAZZI, C., 1993, "Variability of Process Models, and the EPOS Solution". In: *Proceedings of the 8th International Software Process Workshop*, pp. 43–46, Wadern, Alemanha, Março.

COPPE-SISTEMAS, 1998, *Processo Padrão para Desenvolvimento de Software na Fundação Bahiana de Cardiologia.* In: Report, COPPE/Programa de Engenharia de Sistemas e Computação, Rio de Janeiro, Brasil.

CUGOLA, G., DINITTO, E., GHEZZI, C., MANTIONE, M., 1995, "How to Deal With Deviations During Process Model Enactment". In: *Proceedings of the 17th International Conference on Software Engineering*, pp. 265–273, Seattle, Washington, EUA, Abril.

CURTIS, B., 1997, "Software Process Improvement: Methods and Lessons Learned". In: *Proceedings of the 19th International Conference on Software Engineering*, pp. 624–625, Boston, Massachusetts, EUA, Maio.

CURTIS, B., 1998, "Which Comes First, the Organization or Its Process?", *IEEE Software*, v. 15, n. 6, pp. 10–13, Novembro/Dezembro.

CURTIS, B., KELLNER, M. I., OVER, J., 1992, "Process Modelling", *Communications of the ACM*, v. 35, n. 9, pp. 75–90, Setembro.

CUSUMANO, M. A., SELBY, R. W., 1997, "How Microsoft Builds Software", *Communications of the ACM*, v. 40, n. 6, pp. 53–62, Junho.

CYBULSKI, J., REED, K., 1992, "A Hypertext-Based Software Engineering Environment", *IEEE Software*, pp. 62–68, Março.

`http://www.cs.latrobe.edu.au/research/se-3.html`

DA SILVA, J. L. M., 1993, *Modelagem do Processo de Desenvolvimento de Software*. Tese de M.Sc., COPPE/UFRJ - Programa de Engenharia de Sistemas e Computação, Rio de Janeiro, Brasil.

DASKALANTONAKIS, M. K., 1997, "Achieving Higher SEI Levels". In: Oman, P. W., Pfleeger, S. L. (eds), *Applying Software Metrics*, IEEE Computer Society Press.

DE BRA, P., 1996, "A Web of Objects". In: *Proceedings of the Hypermedia Research and the World-Wide Web Workshop*, ACM Hypertext'96, Washington, D.C., EUA, Março.

`http://www.cs.bgsu.edu/hrweb/papers/debra.html`

DEMIRÖRS, E., DEMIRÖRS, O., DIKENELLI, O., KESKIN, B., 1998, "Process Improvement Towards ISO 9001 Certification in a Small Software Organization". In: *Proceedings of the 20^{th} International Conference on Software Engineering*, pp. 435–438, Kyoto, Japão, Abril.

DIAMANT, J., 1995, "Human Interactions Support in HP SynerVision for SoftBench". In: *Proceedings of the 9^{th} International Software Process Workshop*, pp. 44–46, Airlie, VA, Outubro.

DIAZ, M., SLIGO, J., 1997, "How Software Process Improvement helped Motorola", *IEEE Software*, pp. 75–81, Setembro/Outubro.

DUNAWAY, D. K., MASTERS, S., 1996, *CMM^{SM}-Based Appraisal for Internal Process Improvement (CBA IPI): Method Description.* In: Report CMU/SEI-96-TR-007, Software Engineering Institute, Pittsburgh, Pennsylvania, EUA, Abril.

EASTERBROOK, S., CALLAHAN, J., 1996, "Independent Validation of Specifications: A Coordination Headache". In: *Proceedings of the IEEE Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'96) – Workshop on Requirements Engineering in and for Networked Enterprises*, Stanford, CA, EUA, Junho.

`http://sern.cpsc.ucalgary.ca/~maurer/ICSE98WS/ICSEWSubmissions.html`

ELLMER, E., 1995, "Improving Software Process". In: *Proceedings of the Software Engineering Environments Conference*, IEEE Computer Society, pp. 74–83, Nordwijkerhout, Holanda, Abril.

EMAM, K. E., DROUIN, J., MELO, W., 1998, *SPICE – The Theory and Practice of Software Process Improvement and Capability Determination.* IEEE Computer Society Press.

EMAM, K. E., GOLDENSON, D. R., 1996a, "Some Initial Results from the International SPICE Trials", *Software Process Newsletter*, v. spring, n. 6, pp. 1–5.

EMAM, K. E., GOLDENSON, D. R., 1996b, "Some Results from the SPICE Phase One Trials Assessments". In: *Proceedings of the VII CITS - Conferência Internacional de Tecnologia de Software: Qualidade de Software*, Centro Internacional de Tecnologia de Software (CITS), pp. 51–71, Curitiba, Paraná, Brasil, Junho.

ENGELMANN, F., STIENEN, H., LEBSANFT, E., 1997, "Bootstrap: Four Years of Assessment Experience". In: *Proceedings of the 19th International Conference on Software Engineering*, pp. 568–569, Boston, Massachusetts, EUA, Maio.

ESA, 1991, *ESA Software Engineering Standards PSS-05-0 Issue 2.* In: Report, European Space Agency, 8-10, Mario-Nikis, F-75738, Paris Cedex, Fevereiro.

ESPITI, 1999, *European Software Process Improvement Training Initiative (ESPITI).*

`http://www.sea.uni-linz.ac.at/espiti/home.html`

FALBO, R. A., 1996, "Automatização do Processo de Desenvolvimento de Software". In: da Rocha, A. R. C., Weber, K. C. (eds), *Qualidade de Software - Seleção de Textos*, Centro Internacional de Tecnologia de Software (CITS), Gráfica e Editora Linarth Ltda., Curitiba, Paraná, Brasil.

FALBO, R. A., ROCHA, A. R. C., 1996, "Requisitos de Ambientes de Desenvolvimento para suportar Processos de Software". In: *Proceedings of the VII CITS, Conferência Internacional de Tecnologia de Software: Qualidade de Software*, Centro Internacional de Tecnologia de Software (CITS), pp. 107–121, Curitiba, Paraná, Brasil, Junho.

FERGUSON, J., *ET AL.*, 1997, *Software Acquisition Process Maturity Questionnaire.* In: Report CMU/SEI-97-SR-013, Software Engineering Institute, Pittsburgh, Pennsylvania, EUA.

FERNSTRÖM, C., 1993, "Process WEAVER: Adding process support to UNIX". In: *Proceedings of the 2nd International Conference on Software Process (ICSP'2)*, pp. 12–26, Berlim, Alemanha, Março.

FIELDING, R. T., WHITEHEAD-JR., E. J., ANDERSON, K. M., BOLCER, G. A., OREIZY, P., TAYLOR, R. N., 1998, "Web-Based Development of Complex Information Products", *Communications of the ACM*, v. 41, n. 8, pp. 84–92, Agosto.

FINKELSETIN, A., *ET AL.*, 1992, "Viewpoints: a Framework for Integration Multiple Perspectives in System Development", *International Journal of Software Engineering and Knowledge Engineering*, pp. 31–57, Março.

FOWLER, P., RIFKIN, S., 1990, *Software Engineering Process Group Guide.* In: Report CMU/SEI-90-TR-24, Software Engineering Institute, Pittsburgh, Pennsylvania, EUA.

FOX, C., FRAKES, W., 1997, "The Quality Approach: Is It Delivering?", *Communications of the ACM*, v. 40, n. 6, pp. 24–29, Junho.

GARG, P. K., MI, P., PHAM, T., SCACCHI, W., THUNQUEST, G., 1994, "The SMART Approach to Software Process Engineering". In: *Proceedings of the 16th International Conference on Software Engineering*, pp. 341–352.

GARVEY, P. R., PHAIR, D. J., WILSON, J. A., 1997, "An Information Achitecture for Risk Assessment and Management", *IEEE Software*, pp. 25–34, Maio/Junho.

GIBSON, R., 1997, "Software Process Modeling and Assessment". In: *Proceedings of the 9th International Conference on Software Engineering & Knowledge (SEKE'97)*, pp. 105–110, Madri, Espanha, Junho.

GRADY, R. B., 1997, "Successfully Applying Software Metrics". In: Oman, P. W., Pfleeger, S. L. (eds), *Applying Software Metrics*, IEEE Computer Society Press.

GRAHL, E. A., HUGO, M., COLOMBO, R. M. T., FERNANDES, R. A., 1997, "Um Comparativo entre o Modelo CMM-SEI e a Norma ISO/IEC 12207". In: *Proceedings of the Workshop Qualidade de Software - XI Simpósio Brasileiro de Engenharia de Software*, pp. 32–39, Fortaleza, Ceará, Brasil, Outubro.

GRUHN, V., 1991, "The Software Process Management Environment MELMAC". In: *Proceedings of the 1st European Workshop on Software Process Technology (EWSPT'1)*, pp. 191–201, Milão, Itália, Maio.

GRUHN, V., SCHNEIDER, M., 1998, "Workflow Management based on Process Model Repositories". In: *Proceedings of the 20th International Conference on Software Engineering*, pp. 379–388, Kyoto, Japão, Abril.

GRUHN, V., URBAINCZYK, J., 1998, "Software Process Modeling and Enactment: An Experience Report Related to Problem Tracking in an Industrial Project". In: *Proceedings of the 20th International Conference on Software Engineering*, pp. 13–21, Kyoto, Japão, Abril.

GRUNDY, J. C., APPERLEY, M. D., HOSKING, J. G., MUGRIDGE, W. B., 1998, "A Descentralized Architecture for Software Process Modeling and Enactment", *IEEE Internet Computing*, v. 2, n. 5, pp. 53–62, Setembro/Outubro.

GRUNDY, J. C., MUGRIDGE, W. B., HOSKING, J. G., AMOR, R. W., 1995, "Support for Collaborative, Integrated Software Development". In: *Proceedings of the Software Engineering Environments Conference*, IEEE Computer Society, pp. 84–93, Nordwijkerhout, Holanda, Abril.
`http://gille.loria.fr:7000/see95/see95.html`

GUNTER, C., MITCHELL, J., NOTKIN, D., 1996, "Strategic Directions in Software Engineering and Programming Languages", *ACM Computing Surveys*, v. 28, n. 4, pp. 727–737, Dezembro.

HAASE, V., MESSNARZ, R., KNOCH, G., KUGLER, H. J., DECRINIS, P., 1994, "Bootstrap: Fine-Tuning Process Assessment", *IEEE Software*, pp. 25–35, Julho.

HALEY, T. J., 1996, "Software Process Improvement at Raytheon", *IEEE Software*, pp. 33–41, Novembro.

HAREL, D., *ET AL.*, 1990, "STATEMATE: A Working Environment for the Development of Complex Reactive Systems", *IEEE Transactions on Software Engineering*, v. 16, n. 4, pp. 403–414, Abril.

HAREL, D., NAAMAD, A., 1996, "The STATEMATE Semantics of Statecharts", *ACM Transactions on Software Engineering and Methodology*, v. 5, n. 4, pp. 293–333, Outubro.

HARTMAN, J. H., PROEBSTING, T. A., SUNDARAM, R., 1997, "Index-Based Hyperlinks". In: *Proceedings of the Sixth International World-Wide Web Conference: Everyone, Everything Connected*, Santa Clara, Califórnia, EUA, Abril. `http://www6.nttlabs.com/HyperNews/get/PAPER248.html`

HEIMANN, P., JOERIS, G., KRAPP, C., WESTFECHTEL, B., 1996, "DYNAMITE: Dynamic Task Nets for Software Process Management". In: *Proceedings of the 18th International Conference on Software Engineering*, pp. 331–341, Berlim, Alemanha, Março.

HERBSLEB, J., *ET AL.*, 1997, "Software Quality and the Capability Maturity Model", *Communications of the ACM*, v. 40, n. 6, pp. 30–40, Junho.

HERBSLEB, J., CARLETON, A., ROZUM, J., SIEGEL, J., ZUBROW, D., 1994, *Benefits of CMM-Based Software Process Improvement: Initial Results.* In: Report CMU/SEI-94-TR-13, Software Engineering Institute, Pittsburgh, Pennsylvania, EUA.

HERBSLEB, J. D., GOLDENSON, D. R., 1996, "A Systemativ Survey of CMM Experience and Results". In: *Proceedings of the 18th International Conference on Software Engineering*, pp. 323–330, Berlim, Alemanha, Março.

HILBERT, D. M., REDMILES, D. F., 1998a, "An Approach to Large-Scale Collection of Application Usage Data Over the Internet". In: *Proceedings of the 20th International Conference on Software Engineering*, pp. 136–145, Kyoto, Japão, Abril.

HILBERT, D. M., REDMILES, D. F., 1998b, "An Approach to Large-Scale Collection of Application Usage Data Over the Internet". In: *Proceedings of the 20th*

International Conference on Software Engineering, pp. 136–145, Kyoto, Japão, Abril.

HOLLENBACH, C., YOUNG, R., PFLUGRAD, A., SMITH, D., 1997, "Combining Quality and Software Improvement", *Communications of the ACM*, v. 40, n. 6, pp. 41–45, Junho.

HUMPHREY, W. S., 1990, *Managing the Software Process.* Addison-Wesley Publishing Company.

HUMPHREY, W. S., 1993, *Introduction to Software Process Improvement.* In: Report CMU/SEI-92-TR-7, Software Engineering Institute, Pittsburgh, Pennsylvania, EUA.

HUMPHREY, W. S., 1995, *A Discipline for Software Engineering.* Addison-Wesley Publishing Company Inc.

HUMPHREY, W. S., 1996, "Using a Defined and Measured Personal Software Process", *IEEE Software*, pp. 77–88, Maio.

HUMPHREY, W. S., KELLNER, M. I., 1989, "Software Process Modeling: Principles of Entity Process Models". In: *Proceedings of the 11th International Conference on Software Engineering*, pp. 331–342, Pittsburgh, Pennsylvania, EUA, Maio.

HUMPHREY, W. S., KITSON, D. H., KASSE, T. C., 1993, "The State of Software Engineering Practice: A Preliminary Report". In: Reifer, D. J. (ed), *Software Management*, 4 ed., IEEE Computer Society Press.

HUTCHINGS, T., *ET AL.*, 1993, "Process Improvement that Lasts: an Integrated Training and Consulting Method", *Communications of the ACM*, v. 36, n. 10, pp. 105–113, Outubro.

IIDA, H., MIMURA, K., INOUE, K., TORII, K., 1993, "Monitor and Navigation System for Cooperative Development Based on Activity Sequence Model". In: *Proceedings of the 2nd International Conference on Software Process (ICSP'2)*, pp. 64–74, Berlin, Alemanha, Março.

INGHAM, D. B., CAUGHEY, S. J., LITTLE, M. C., 1997, "Supporting Highly Manageable Web Services". In: *Proceedings of the Sixth International World-Wide Web Conference: Everyone, Everything Connected*, Santa Clara, Califórnia, EUA, Abril.

`http://www6.nttlabs.com/HyperNews/get/PAPER27.html`

IPT GROUP, 1997, *Process Weaver.* CERN – Information and Programming Technology Group, Genebra, Suíça.

`http://www1.cern.ch/PTTOOL/ProcessWeaver/Detail.html`

IRVING, C. W., 1999, *The Trillium Model.*

`http://ricis.cl.uh.edu/trillium/trillium.html`

ISAKOWITZ, T., BIEBER, M., VITALI, F., 1998, "Web Information Systems", *Communications of the ACM*, v. 41, n. 7, pp. 78–80, Julho.

ISO, 1999, *International Organization for Standardization (ISO).*

`http://www.iso.ch/`

ISO/IEC 12207, 1995, *ISO/IEC 12207: Information technology – Software Life Cycle Processes.* ISO/IEC Copyright Office – Case Postale 56 – CH-1211 Genève 20 – Suíça.

JACCHERI, M. L., CONRADI, R., 1993, "Techniques for Process Model Evolution in EPOS", *IEEE Transactions on Software Engineering*, pp. 1145–1156, Dezembro.

`http://www.idt.unit.no/~epos/`

JAVASCRIPT, 1997, *JavaScript Reference.* In: Report, Netscape Communications Corp.

`http://home.netscape.com`

JOHNSON, D. M., 1996, "The Systems Engineer and the Software Crisis", *ACM Software Engineering Notes*, v. 21, n. 2, pp. 64–73, Março.

KAISER, G. E., BEN-SHAUL, I. Z., 1993, "Process Evolution in the Marvel Environment". In: *Proceedings of the 8th International Software Process Workshop*, pp. 104–106, Wadern, Alemanha, Março.

KAISER, G. E., DOSSICK, S. E., JIANG, W., YANG, J. J., 1997, "An Architecture for WWW-based Hypercode Environments". In: *Proceedings of the 19th International Conference on Software Engineering*, pp. 3–13, Boston, Massachusetts, EUA, Maio.

KAISER, G. E., KAPLAN, S. M., 1995, "CSCW and Software Process". In: *Proceedings of the 9th International Software Process Workshop*, pp. 9–11, Airlie, VA, Outubro.

KELLNER, M. I., HANSEN, G. A., 1988, *Software Process Modeling.* In: Report CMU/SEI-88-TR-9, Software Engineering Institute, Pittsburgh, Pennsylvania, EUA.

KELLNER, M. I., HANSEN, G. A., 1989, "Software Process Modeling: A Case Study". In: *Proceedings of the 22nd Hawaii International Conference on System Sciences*, Hawaii, EUA, Janeiro.

KELLNER, M. J., 1989, "Experience with Enactable Software Process Models". In: *Proceedings of the 5th International Software Process Workshop: Experience with Software Process Models*, pp. 10–13, Outubro.

KITCHENHAM, B., PFLEEFER, S. L., 1996, "Software Quality: the Elusive Target", *IEEE Software*, pp. 12–21, Janeiro.

KONNO, S., TANAKA, T., OHNO, T., 1998, "A Web-based Document Review System". In: *Proceedings of the ICSE 98 Workshop on Software Engineering over the Internet.*

`http://sern.cpsc.ucalgary.ca/~maurer/ICSE98WS/ICSEWSubmissions.html`

KRASNER, H., *ET AL.*, 1992, "Lessons Learned from a Software Process Modeling System", *Communications of the ACM*, v. 35, n. 9, pp. 91–100, Setembro.

KRAUT, R. E., STREETER, L., 1995, "Coordination in Software Development", *Communications of the ACM*, v. 38, n. 3, pp. 69–81, Março.

KUDO, Y., KOIZUMI, S., OHNO, O., KAWABE, H., FURUHATA, Y., 1998, "Problem Management System for Distributed Software Development". In: *Proceedings of the ICSE 98 Workshop on Software Engineering over the Internet.*
`http://sern.cpsc.ucalgary.ca/~maurer/ICSE98WS/ICSEWSubmissions.html`

LAI, R., 1993, "The Move to Mature Processes", *IEEE Software*, pp. 14–17, Julho.

LEASE, M., 1998, *ConceptTracker.* Laboratory for Software Research.
`http://www.cs.tamu.edu/research/LSR/`

LEHMAN, M. M., 1995, "Software Process Improvement – The Way Forward". In: *Proceedings of the VII International Conference on Advanced Information Systems Engineering*, CAiSE*95, pp. 1–11, Jyvaskyla, Finlândia.
`http://www.informatik.uni-trier.de/~ley/db/conf/caise/caise95.html`

LEHMAN, M. M., 1997, "Process Modelling – Where Next". In: *Proceedings of the 19th International Conference on Software Engineering*, pp. 549–569, Boston, Massachusetts, EUA, Maio.

LEONHARDT, U., KRAMER, J., NUSEIBEH, B., 1995, "Decentralised Process Enactment in a Multi-Perspective Development Environment". In: *Proceedings of the 17th International Conference on Software Engineering*, pp. 255–264, Seattle, Washington, EUA, Abril.

LIGHT, 1999, *LIGHT, LIfe cycle Global HyperText.* Information and Programming Technology Group, CERN, Genebra, Suíça.
`http://light.cern.ch/`

LIN, C. Y., ABDEL-HAMID, T., SHERIF, J. S., 1997, "Software-Engineering Process Simulation Model (SEPS)", *The Journal of Systems and Software*, v. 28, n. 3, pp. 263–277, Setembro.

LOWE, D. E., COX, G. M., 1996, "Implementing the Capability Maturity Model for Software Development", *Hewlett-Packard Journal*, Agosto.

LSR, 1999, *Laboratory for Software Research.*

`http://www.cs.tamu.edu/research/LSR/`

MACHADO, C. A. F., OLIVEIRA, L. C. A., FERNANDES, R. A., 1997, "Utilização da Norma ISO/IEC 12207 e do Modelo CMM-SEI na Celepar". In: *Proceedings of the Workshop Qualidade de Software - XI Simpósio Brasileiro de Engenharia de Software*, pp. 40–48, Fortaleza, Ceará, Brasil, Outubro.

MADHAVJI, N. H., PENEDO, M. H., 1993, "Special Section on the Evolution of Software Process", *IEEE Transactions on Software Engineering*, v. 19, n. 12, pp. 1125–1127, Dezembro.

MAIDANTCHIK, C., 1998a, "Gerência do Processo de Software para Equipes Globalmente Distribuídas". In: *Proceedings of the II Concurso de Monografias - Qualidade e Produtividade em Software*, Brasil.

MAIDANTCHIK, C., 1998b, "Web Documentation Systems in a Worldwide Collaboration". In: *Proceedings of the International Seminar on Document Management (ISDM'98)*, Curitiba, Paraná, Brasil, Novembro.

MAIDANTCHIK, C., DA ROCHA, A. R. C., 1996, *Software Process Improvement in the ATLAS Experiment.* ATLAS Internal Note SOFT-NO-025, CERN, Genebra, Suíça, Março.

MAIDANTCHIK, C., DA ROCHA, A. R. C., WERNER, C., 1995, "Quality Assurance on Coupling DAQ Software Modules". In: *Proceedings of the Fourth International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics*, Pisa, Itália, Abril.

MAIDANTCHIK, C., DA ROCHA, A. R. C., XEXÉO, G. B., 1996, "Melhoria do Processo de Desenvolvimento de Software no Experimento ATLAS do CERN". In: *Proceedings of the X Simpósio Brasileiro de Engenharia de Software (SBES'96) - Workshop Qualidade e Produtividade de Software*, São Carlos, São Paulo, Brasil, Outubro.

MAIDANTCHIK, C., DA ROCHA, A. R. C., XEXÉO, G. B., 1997a, "Processo de Desenvolvimento de Software para Equipes de Trabalho Cooperativas e Dis-

tribuídas". In: *Proceedings of the Workshop Qualidade de Software - COPPE-Sistemas/UFRJ e PBQP/Sub-comite de Software*, Rio de Janeiro, Brasil, Maio.

MAIDANTCHIK, C., DA ROCHA, A. R. C., XEXÉO, G. B., 1997b, "A WWW Software Development Environment to Support Cooperative and Spread Working Groups". In: *Proceedings of the Computing in High Energy Physics'97 Conference*, Berlim, Alemanha, Abril.

MAIDANTCHIK, C., DA ROCHA, A. R. C., XEXÉO, G. B., 1998a, "Gerência do Processo de Software para Equipes Globalmente Distribuídas". In: *Proceedings of the Workshop de Qualidade de Software*, pp. 27–32, Maringá, Paraná, Brasil, Outubro.

MAIDANTCHIK, C., DA ROCHA, A. R. C., XEXÉO, G. B., 1998b, "A Web Process Support Group System for Distributed Working Groups". In: *Proceedings of the WebNet World Conference of the WWW, Internet & Intranet*, Orlando, Flórida, EUA, Novembro.

MAIDANTCHIK, C., DA ROCHA, A. R. C., XEXÉO, G. B., 1999, "Software Process Standardization for Distributed Working Groups". In: *Proceedings of the 4^{th} IEEE International Software Engineering Standards Symposioum*, Curitiba, Paraná, Brasil, Maio.

MAIDANTCHIK, C., ISAAC, M. C. P., 1994, "HyperDev: Hypertext Tool to Support Object-Oriented Software Development". In: *Proceedings of the Computing in High Energy Physics'94 Conference*, São Francisco, EUA, Abril.

MAIDANTCHIK, C., PETERSON, D., SEIXAS, J. M., 1998c, "Sistema WWW de Apoio ao Processo de Desenvolvimento de Projetos Cooperativos e Distribuídos". In: *Proceedings of the XXV Congresso Brasileiro de Ensino de Engenharia (COBENGE)*, São Paulo, Brasil, Outubro.

MAIDANTCHIK, C., ROCHA, A. R. C., XEXÉO, G., 1998d, "A WWW Software Development Environment to Support Cooperative and Spread Working Groups", *Computer Physics Communications*, v. 110, n. 1-3, pp. 84–92, Maio.

MARSHALL, P., MACLENNAN, F., TOBIN, M., 1996, "Analysis of Observation and Problem Reports from Phase One of the SPICE Trials", *Software Process Newsletter*, v. spring, n. 6, pp. 10–12.

MAURER, F., DELLEN, B., 1998, "An Internet Based Software Process Management Environment". In: *Proceedings of the ICSE 98 Workshop on Software Engineering over the Internet.*

`http://sern.cpsc.ucalgary.ca/~maurer/ICSE98WS/ICSEWSubmissions.html`

MAURER, F., KAISER, G., 1998, "Software Engineering in the Internet Age", *IEEE Internet Computing*, v. 2, n. 5, pp. 22–24, Setembro/Outubro.

METEOR, 1999, *METEOR: Managing End-To-End OpeRations.* Large Scale Distributed Information Systems - Department of Computer Science, University of Georgia, EUA.

`http://lsdis.cs.uga.edu/proj/meteor/meteor.html`

MI, P., SCACCHI, W., 1991, "Modeling Articulation Work In Software Engineering Process". In: *Proceedings of the 1st International Conference on Software Process (ICSP'1)*, pp. 188–201, Redonto Beach, Califórnia, EUA, Outubro.

MOYNIHAN, T., 1997, "How Experienced Project Managers Assess Risk", *IEEE Software*, pp. 35–41, Maio/Junho.

MULLER, R. J., 1998, *Productive Objects: An Applied Software Project Management Framework.* São Francisco, Califórnia, EUA, Morgan Kaufmann Publishers, Inc.

NARAYANASWAMY, K., GOLDMAN, N., 1992, ""Lazy" Consistency: A Basis for Cooperative Software Development". In: *Proceedings of the CSCW'92*, pp. 257–264, Toronto, Canadá.

NELSON, T., 1965, "A File Strucutre for the Complex, The Changing and The Indeterminate". In: *Proceedings of the ACM 20th National Conference.*

NETSCAPE, 1999, *Netscape Communications Corp.*

`http://home.netscape.com`

NIELSEN, J., 1990, *Hypertext/Hypermedia.* Academic Press.

NOLL, J., SCACCHI, W., 1991, "Integrating Diverse Information Repositories: A Distributed Hypertext Approach", *IEEE Computer*, pp. 38–45, Dezembro.

OSTERWEIL, L., *ET AL.*, 1996, "Strategic Directions in Software Quality", *ACM Computing Surveys*, v. 28, n. 4, pp. 738–750, Dezembro.

OSTERWEIL, L. J., 1987, "Software Process Are Software Too". In: *Proceedings of the 9^{th} International Conference on Software Engineering*, pp. 2–13, Monterey, CA, EUA, Março.

OSTERWEIL, L. J., 1997, "Software Process Are Software Too, Revisited: An Invited Talk on the Most Influential Paper of ICSE 9". In: *Proceedings of the 19^{th} International Conference on Software Engineering*, pp. 540–548, Boston, Massachusetts, EUA, Maio.

PAULISH, D. J., CARLETON, A. D., 1997, "Case Studies of Software Process Improvement Measurement". In: Oman, P. W., Pfleeger, S. L. (eds), *Applying Software Metrics*, IEEE Computer Society Press.

PAULK, M., CURTIS, B., CHRISSIS, M., WEBER, C., 1993a, *Cabability Maturity Model for Software.* In: Report CMU/SEI-93-TR-24, V1.1, Software Engineering Institute, Pittsburgh, Pennsylvania, EUA.

PAULK, M., WEBER, C., GARCIA, S., CHRISSIS, M., BUSH, M., 1993b, *Key Practices of the Capability Maturity Model.* In: Report CMU/SEI-93-TR-25, V1.1, Software Engineering Institute, Pittsburgh, Pennsylvania, EUA.

PAULK, M. C., 1994, "How ISO 9001 Compares with the CMM", *IEEE Software*, pp. 74–83, Janeiro.

PAULK, M. C., CURTIS, B., CHRISSIS, M. B., 1993c, "Capability Maturity Model, Version 1.1", *IEEE Software*, pp. 18–27, Julho.

Paulk, M. C., Weber, C. V., Curtis, B., Chrissis, M. B. (eds), 1997, *The Capability Maturity Model: Guidelines for Improving the Software Process.* Carnegie Mellon University, Software Engineering Institute, Addison-Wesley Longman, Inc.

PENEDO, M. H., 1995, "ISPW9 Process Demonstrations – Summary". In: *Proceedings of the 9th International Software Process Workshop*, pp. 19–32, Airlie, VA, Outubro.

PERPICH, J. M., *ET AL.*, 1997, "Anywhere, Anytime Code Inspections: Using the Web to Remove Inspection Bottlenecks in Large-Scale Software Development". In: *Proceedings of the 19th International Conference on Software Engineering*, pp. 14–21, Boston, Massachusetts, EUA, Maio.

PERRY, D., *ET AL.*, 1994, "People, Organizations, and Process Improvement", *IEEE Software*, pp. 36–45, Julho.

PERRY, D. E., SIY, H. P., VOTTA, L. G., 1998, "Parallel Changes in Large Scale Software Development: An Observational Case Study". In: *Proceedings of the 20th International Conference on Software Engineering*, pp. 251–260, Kyoto, Japão, Abril.

PESSÔA, M., SPíNOLA, M., VOLPE, R. L. D., 1997, "Uma Experiência Prática na Implantação do Modelo CMM". In: *Proceedings of the Workshop Qualidade de Software - XI Simpósio Brasileiro de Engenharia de Software*, pp. 49–57, Fortaleza, Ceará, Brasil, Outubro.

PEUSCHEL, B., SCHAFER, W., 1992, "Concepts and Implementation of Rule-Based Process Engine". In: *Proceedings of the 14th International Conference on Software Engineering*, pp. 262–279.

PFLEEGER, S. L., JEFFERY, R., CURTIS, B., KITCHENHAM, B., 1997, "Status Report on Software Measurement", *IEEE Software*, pp. 33–43, Março/Abril.

PRESSMAN, R. S., 1997, *Software Engineering: a Practitioner's Approach.* 4 ed., McGraw-Hill, Inc., New York.

RACCOON, L. B. S., 1997, "Fifty Years of Progress in Software Engineering", *ACM Software Engineering Notes*, v. 22, n. 1, pp. 88–104, Janeiro.

RADA, R., CARGILL, C., KLENSIN, J., 1998, "Consensus and the Web", *Communications of the ACM*, v. 41, n. 7, pp. 17–22, Julho.

RADICE, R. A., ROTH, N. K., JR., A. C. O., CIARFELLA, W. A., 1985, "A programming process architecture", *IBM Systems Journal*, v. 24, n. 2, pp. 79–90.

RAGGETT, D., OTHERS, 1997, *HTML 4.0 Specification.* In: Report, W3C.

REIFER, D. J., 1993, "Managing the Three P's: The Key to Success in Software Management". In: Reifer, D. J. (ed), *Software Management*, 4 ed., IEEE Computer Society Press.

REISS, S. P., 1996, "Software Tools and Environments", *ACM Computing Surveys*, v. 28, n. 1, pp. 281–284, Março.

RETTING, M., SIMONS, G., 1993, "A Project Planning and Development Process for Small Teams", *Communications of the ACM*, v. 36, n. 10, pp. 45–55, Outubro.

ROTHERY, B., 1993, *ISO 9000.* Makron Books do Brasil Editora Ltda.

SAEKI, M., KENOKO, T., SAKAMOTO, M., 1991, "A Method for Software Process Modeling and Description Using LOTOS". In: *Proceedings of the 1st International Conference on Software Process (ICSP'1)*, pp. 90–104, Redonto Beach, Califórnia, EUA, Outubro.

SAKAMOTO, K., NAKAKOJI, K., TAKAGI, Y., NIIHARA, N., 1998, "Toward Computational Support for Software Process Improvement Activities". In: *Proceedings of the 20th International Conference on Software Engineering*, pp. 22–31, Kyoto, Japão, Abril.

SANDEWALL, E., 1996, "Towards a World-Wide Data Base". In: *Proceedings of the Fifth International World-Wide Web Conference*, Paris, França, Maio.
`http://http://www5conf.inria.fr/fich_html/papers/P54/Overview.html`

SCACCHI, W., MI, P., 1993, "Experiences in the Modeling, Analysis, and Simulation of Formalized Software Processes". In: *Proceedings of the 8th International Software Process Workshop*, pp. 135–138, Wadern, Alemanha, Março.

SCHMAUCH, C. H., 1994, *ISO 9000 for Software Developers.* Milwaukee, Wisconsin 53202, ASQC Quality Press.

SCHNEIDEWIND, N. F., FENTON, N., 1996, "Do Standards Improve Quality?", *IEEE Software*, pp. 22–24, Janeiro.

SEI, 1999, *Software Engineering Institute (SEI)*. Carnegie Mellon University. `http://www.sei.cmu.edu/`

SHEPARD, T., SIBBALD, S., WORTLEY, C., 1992, "A Visual Software Process Language", *Communications of the ACM*, v. 35, n. 4, pp. 37–44, Abril.

SIMMONS, S. B., ELLIS, N. C., FUJIHARA, H., KUO, W., 1998, *Software Measurement: A Visualization Toolkit for Project Control and Process Improvement.* Prentice-Hall, Inc., New Jersey 07458, EUA.

SOMMERVILLE, I., 1992, *Software Engineering.* Addison-Wesley Publishing Company Inc.

SOMMERVILLE, I., 1996, "Software Process Models", *ACM Computing Surveys*, v. 28, n. 1, pp. 269–271, Março.

SOUZA, J., MEDEIROS, S., 1995, "CooMan - A Global Collaborative Project Management System". In: *Proceedings of the CRIWG'95*, Lisboa, Portugal, Julho. `http://www.cos.ufrj.br/~palma/`

SPICE, 1999, *Software Process Improvement & Capability dEtermination.* `http://www-sqi.cit.gu.edu.au/spice/general/spiceint.html`

STALLINGS, W., 1997, *Data and Computer Communications.* 5 ed., Prentice-Hall, Inc., New Jersey 07458, EUA.

STEIN, M., RIEDL, J., HARNER, S. J., MASHAYEKHI, V., 1997, "A Case Study of Distributed, Asynchronous Software Inspection". In: *Proceedings of the 19^{th} International Conference on Software Engineering*, pp. 107–117, Boston, Massachusetts, EUA, Maio.

STELZER, D., MELLIS, W., HERZWURM, G., 1996, "Software Process Improvement via ISO 9000? Results of Two Surveys among European Software Houses". In: *Proceedings of the 29^{th} Hawaii International Conference on System Sciences*,

Wailea, Hawaii, EUA, Janeiro.

http://www.informatik.uni-koeln.de/winfo/prof.mellis/publications/

SURVEYER, J., 1997, "Project and Process Management", *Software Development*, pp. 59–63, Novembro.

http://www.sdmagazine.com

SUTTON, S. M., HEIMBIGNER, S., OSTERWEIL, L. J., 1995, "APPL/A: A Language for Software Process Programming", *ACM Transactions on Software Engineering and Methodology*, v. 4, n. 3, pp. 221–286, Julho.

SUTTON, S. M., OSTERWEIL, L. J., 1996, "Product Families and Process Families". In: *Proceedings of the 10^{th} International Software Process Workshop*, Ventron, França, Junho.

TESORIERO, R., ZELKOWITZ, M., 1998, "A Web-Based Tool for Data Analysis and Presentation", *IEEE Internet Computing*, v. 2, n. 5, pp. 63–69, Setembro/Outubro.

VALETTO, G., KAISER, G. E., 1996, "Enveloping Sophisticated Tools into Process-Centered Environments", *Automated Software Engineering Notes*, v. 3, n. 3/4, pp. 309–345, Agosto.

VISAGGIO, G., 1994, "Process Improvement Through Data Reuse", *IEEE Software*, pp. 76–85, Julho.

VOSSEN, G., WASKE, M., WITTKOWSKI, G., 1996, *Dynamic Workflow Management on the Web*. In: Report 24/96-I, Lehrstuhl für Informatik, Universität Münster, Grevener Strasse 91, D-48159 Münster, Alemanha, Novembro.

http://wwwmath.uni-muenster.de/math/inst/info/u/dbis/Wesle/Common/

WAIBA, 1999, *Intelligent Browsing Assistant for the World Wide Web and GroupWare for the Web*. OSF Research Institute.

http://www.camb.opengroup.org/www/waiba/index.html

WANG, Y., KING, G., DORLING, A., PATEL, D., COURT, I., STAPLES, G., ROSS, M., 1998, "A Worldwide Survey of Base Process Activities Towards Soft-

ware Engineering Process Excellence". In: *Proceedings of the 20th International Conference on Software Engineering*, pp. 439–442, Kyoto, Japão, Abril.

WASSERMAN, A. I., 1996, "Toward a Discipline of Software Engineering", *IEEE Software*, pp. 23–31, Novembro.

WEBER, K. C., 1997, "Medições em Empresas de Software". In: *Proceedings of the Workshop Qualidade de Software - XI Simpósio Brasileiro de Engenharia de Software*, pp. 15–23, Fortaleza, Ceará, Brasil, Outubro.

WEBER, K. C., ROCHA, A. R. C., 1999, *Qualidade e Produtividade em Software*. R. Tabapuã, 1348, Itaim-Bibi, São Paulo, Brasil, Makron Books do Brasil Editora Ltda.

WET, 1999, *2nd Workshop on Coordinating Distributed Software Development Projects*. IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.
`http://131.246.244.166:8880/~koetting/WETICE99/`

WHERE, 1999, *WHERE: A Web-based Hypertext Environment for Requirements Evolution*. National Aeronautics and Space Administration (NASA).
`http://research.ivv.nasa.gov/projects/WHERE/`

WHITEHEAD, E. J., WIGGINS, M., 1998, "WEBDAV: IETF Standard for Collaborative Authoring on the Web", *IEEE Internet Computing*, v. 2, n. 5, pp. 34–40, Setembro/Outubro.

WILLIAMS, R. C., WALKER, J. A., DOROFEE, A. J., 1997, "Putting Risk Management into Practice", *IEEE Software*, pp. 75–82, Maio/Junho.

WISE, 1998, *The WISE Project Management System*.
`http://research.ivv.nasa.gov/projects/WISE/index.html`

WOODMAN, I., HUNTER, R., 1996, "Analysis of Assessment Data from Phase One of the SPICE trials", *Software Process Newsletter*, v. spring, n. 6, pp. 5–10.

WSEI, 1998, *ICSE 98 Workshop on Software Engineering over the Internet*.
`http://sern.cpsc.ucalgary.ca/~maurer/ICSE98WS/ICSWWSSubmissions.html`

WWW, 1999, *World Wide Web Consortium.*

`http://www.w3.org/hypertext/WWW/TheProject`

YOURDON, E., 1996, *Rise & Ressurrection of the American Programmer.* Prentice-Hall, Inc., New Jersey 07458, EUA.