AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE ORIENTADO A ORGANIZAÇÃO APLICADO À INSTRUMENTAÇÃO VIRTUAL

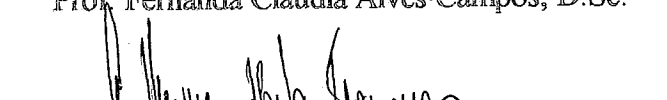
Fábio Renato Silva Martins

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 MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

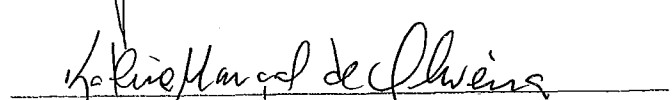
Aprovada por:

Profª. Ana Regina Cavalcanti da Rocha, D.Sc.

Prof. Fernanda Claudia Alves Campos, D.Sc.



Prof. Guilherme Horta Travassos, D.Sc.



Profª. Káthia Marçal de Oliveira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2004

MARTINS, FABIO RENATO SILVA

Ambiente de Desenvolvimento de Software Orientado a Organização Aplicado à Instrumentação Virtual [Rio de Janeiro] 2004

VIII, 140 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2003)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Processo de Software

2. Ambientes de Desenvolvimento de Software Orientados à Organização

3. Instrumentação Virtual

I. COPPE/UFRJ II. Título (série)

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

AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE ORIENTADO A ORGANIZAÇÃO APLICADO À INSTRUMENTAÇÃO VIRTUAL

Fábio Renato Silva Martins

Março/2004

Orientadora: Ana Regina Cavalcanti da Rocha

Programa: Engenharia de Sistemas e Computação

Esta dissertação propõe uma forma de particularização e aplicação das técnicas, métodos e ferramentas existentes na engenharia de software à área de instrumentação virtual. Instrumentação virtual é um tipo de software aplicado à área de automação e medição industrial que tem características próprias, por exemplo, linguagem de programação gráfica e interatividade com hardware e equipamentos. Como solução, propõe-se a configuração de um ambiente de desenvolvimento de software orientado a organização que objetiva apoiar projetos de desenvolvimento de instrumentação virtual.

Este ambiente foi configurado através da Estação TABA, um meta-ambiente capaz de gerar, através de instanciação e configuração, ambientes de desenvolvimento de software (ADS) adequados às particularidades de processos de desenvolvimento e projetos específicos, às particularidades do domínio e da organização. Para alcançar tal objetivo, além da definição de um processo de desenvolvimento para instrumentação virtual, foi necessária a adequação da Estação TABA a alguns requisitos oriundos das particularidades de sua aplicação a instrumentação virtual. Estes requisitos englobam a criação de um processo de documentação e uma infra-estrutura computacional de apoio ao mesmo no contexto da Estação TABA. A proposta foi aplicada em uma empresa de desenvolvimento de instrumentação virtual para aprimoramento do trabalho.

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

ENTERPRISE-ORIENTED SOFTWARE DEVELOPMENT ENVIRONMENT APPLIED TO VIRTUAL INSTRUMENTATION

Fábio Renato Silva Martins

March/2004

Advisor: Ana Regina Cavalcanti da Rocha

Department: System and Computing Engineering

This thesis proposes a way of particularization and application of software engineering techniques, methods and tools to the virtual instrumentation field. Virtual instrumentation is a software type, which is applied to the automation, and industrial measurement field that has proper characteristics, as, for example, graphical programming language and interactivity with hardware and equipment. As solution, it is proposed a configuration of an Enterprise-Oriented Software Development Environment that objective support virtual instrumentation development projects. This environment was configured through TABA workstation, a meta-environment, which through instantiation and configuration is capable of generating software development environments (SDE) adequate to the particularities of development processes and specific projects, to the particularities of the domain and the organization. To reach such objective, beyond the definition of a virtual instrumentation development process, it was necessary the adequation of workstation TABA to some requirements derived of the particularities of its application to the virtual instrumentation. These requirements are the creation of a documentation process and its support computational infrastructure in the workstation TABA context. The proposal was applied in a company of virtual instrumentation development for the improvement of the work.

AGRADECIMENTOS

Gostaria de agradecer a todos que me ajudaram direta ou indiretamente na realização desta tese. Gostaria de agradecer a minha família, mãe, irmã e sobrinha, sempre presente em todos os momentos importantes de minha vida. Agradecer aos amigos da graduação, que sempre, de alguma forma, contribuíram para minha formação: Adriano, Cristiano Eduardo, Leonardo, Marcus, Mauro, Paulo, Renato, Rodrigo, Silvio e Thomas. Aos amigos do Laboratório de Engenharia de Software, ao Somulo, por suas contribuições e ajuda, ao Diogo, ao David, Mariano, aos demais amigos do mestrado e a Karina.

Agradecimentos em especial ao Gleison pela paciência e atenção que sempre mostrou, ao Sávio, pelas contribuições e disposição em ajudar a qualquer momento, característica existente apenas nos grandes amigos. Ao Professor Ângelo, pelas contribuições. A Hélio Ricardo, mais do que um companheiro de trabalho, um amigo, suas contribuições foram além da minha formação e me ajudaram a ser uma pessoa melhor.

Agradeço a toda a equipe da M&D, Sanderson, Leandro, Fernando e Kelly pelas contribuições ao trabalho. Ao pessoal administrativo da COPPE/Sistemas, à Ana Paula Prata, e aos professores da linha de Engenharia Software, Guilherme e Cláudia. Às Professoras Fernanda e Káthia por participarem da minha banca.

À Ana Regina, minha orientadora, pela amizade, orientação e incentivo, por acreditar em mim e no meu trabalho, por ser a pessoa especial que é, sua presença, bom humor e felicidade alegram qualquer pessoa ou ambiente e tornam qualquer trabalho mais fácil.

Ao meu primeiro orientador Julio Salek Aude, por ter contribuído de forma irrevogável na minha formação e caráter, onde estiver estará feliz por este trabalho, saudades.

De certo é que as pessoas, e as coisas, não são de verdade! Mire e veja: o mais importante e bonito, do mundo, é isto: que as pessoas não estão sempre iguais, ainda não foram terminadas - mas que elas vão sempre mudando. Afinam ou desafinam. Verdade maior. É o que a vida me ensinou. Isso que me alegra, montão.

João Guimarães Rosa, *Grande sertão: veredas*

INDICE

CAPÍTULO I
INTRODUÇÃO

Neste capítulo são apresentadas as questões que motivaram a realização deste trabalho, seu objetivo principal e a forma como esta dissertação está organizada.

Introdução

Este capítulo pretende, na motivação, caracterizar o problema e levantar hipóteses para sua ocorrência. A seção Objetivo da Tese apresenta a solução e a razão pela qual acredita-se que esta seja efetiva. A Organização da Tese ilustra como o trabalho está apresentado.

1.1 Motivação

Atualmente, a grande evolução dos microprocessadores está levando a uma revolução nos sistemas de automação industrial. Até então, somente sistemas centrados em hardware eram confiáveis o suficiente para aplicações industriais. Com a evolução e o aumento da confiabilidade dos sistemas microprocessados baseados em arquitetura PC (IBM PC), soluções que utilizam esta tecnologia tornaram-se mais atraentes, devido a sua maior flexibilidade e baixo custo. O software tem um papel fundamental dentro desta nova abordagem de automação. Sua demanda e complexidade acompanharam o crescimento do mercado de aplicações na indústria.

No início dos anos 90 o conceito de instrumento virtual (VXIPLUG&PLAY SYSTEM ALLIANCE, 1994, GOLDBERG, 2000) foi introduzido aos sistemas industriais e amplamente adotado por instituições acadêmicas, laboratórios, centros de pesquisa de áreas tais como, automobilismo, aeroespacial, médica e biomédica (SPOELDER *et al.*, 1997, BRIGNELL *et al.*, 1997,RATNER et al, 2000, WANG *et al.*, 2000). A chave do sucesso dos sistemas de instrumentação virtual está em sua maior flexibilidade com relação a soluções de hardware (GOLDBERG, 2000).

Um instrumento virtual é, em princípio, um sistema baseado em computação para testes, medições e controle que utiliza o poder computacional de um microprocessador genérico para cumprir suas tarefas tais como: controle do fluxo de dados provenientes dos periféricos, sensores e interfaces; aquisição, processamento e análise dos dados; gerenciamento de informações, armazenamento, geração de relatórios, gráficos e outros itens (NATIONAL INSTRUMENTS, 2000). Uma definição mais geral é dada por GOLDBERG (2000), segundo este autor um instrumento virtual é composto por sub-unidades especializadas, computadores de utilidade genérica, software e conhecimento especialista.

Os softwares para desenvolvimento nesta nova abordagem de automação são geralmente baseados em tecnologias de programação gráfica, pois estas sempre foram populares na comunidade de automação e controle. Estas tecnologias de programação gráfica são denominadas Linguagens G (HILS, 1992, SHU, 1988) ou linguagens de programação gráfico-diagramáticas (*Diagramatic-Graphical Programming Language*) (BRAGG *et al.*, 1994).

As linguagens de instrumentação virtual mais populares no mercado atualmente são HP VEE®, *Hewlett Packard*™ (HP-VEE-ENGINE, 1997), e *Labview*®, *National Instruments*™ (NATIONAL INSTRUMENTS, 2000).

Quanto às características da programação importantes em uma análise do ponto de vista de engenharia de software e desenvolvimento, pode-se dizer que, por se tratar de uma tecnologia recente, as linguagens de instrumentação virtual são fortemente baseadas na experiência dos desenvolvedores. O domínio de automação possui grande complexidade e as soluções são fortemente baseadas na experiência pessoal do profissional envolvido.

A evolução da qualidade no desenvolvimento de software nas empresas brasileiras vem sendo acompanhada a partir de pesquisas bienais realizadas pela Secretaria de Política de Informática e Automação (SEPIN, 2001) do Ministério da Ciência e Tecnologia (MCT). Esta pesquisa, entre outros objetivos, fornece um panorama das práticas, métodos e ferramentas de engenharia de software adotados pelas empresas do setor de informática do país.

No âmbito deste trabalho, foi realizada uma pesquisa direcionada a empresas que desenvolvem instrumentos virtuais sobre o uso de práticas de engenharia de software, bem

como no conhecimento da existência de normas, ferramentas e demais elementos associados à engenharia de software, que será referenciada como Pesquisa sobre Instrumentação Virtual (PIV). Esta pesquisa é detalhada no capítulo II. Apesar de ter sido realizada com um número pequeno de empresas (oito organizações), esta pesquisa confirma seu objetivo ao ser comparada à pesquisa do SEPIN (2001): Empresas de desenvolvimento de soluções de automação baseadas em instrumentação virtual desconhecem ou utilizam menos as práticas, técnicas, metodologias e ferramentas de Engenharia de Software. O objetivo da pesquisa não foi determinar o panorama da aplicação de engenharia de software em empresas de desenvolvimento baseado em instrumentação virtual, de forma similar ao que o SEPIN realiza com empresas de desenvolvimento diverso, e sim ter um subsídio da particularidade da área de instrumentação virtual. As Tabelas 1.1 e 2.1 ilustram tal comparação.

A tabela 1.1 indica um possível desconhecimento maior sobre modelos de maturidade e normas relacionadas a qualidade de software para as empresas de desenvolvimento baseado em instrumentação virtual, assim como a tabela 1.2 mostra que tais empresas também utilizam em menor monta alguns tipos de ferramentas. É necessário observar que este trabalho não discute a utilidade ou a eficácia de tais métodos e ferramentas no aumento da produtividade e da qualidade do produto, admitindo-se que, de uma forma geral, tal correlação exista. Uma discussão mais ampla de tal correlação, ou seja, práticas e ferramentas de engenharia de software com qualidade do produto, pode ser encontrada em ODDO (2003).

Tabela 1.1 – Comparação do percentual de conhecimento de normas e modelos de maturidade da pesquisa realizada (PIV) com relação à pesquisa da SEPIN (PBQP)

Grau de Conhecimento	Não Conhece		Conhece mas não usa		Conhece e está começando a usar		Conhece e usa	
Norma/Modelo	PBQP	PIV	PBQP	PIV	PBQP	PIV	PBQP	PIV
ISO/IEC 12207	32,7	75	55,1	25	8,3	0	3,9	0
CMM	25,3	62,5	53,7	37,5	17,1	0	3,9	0
SPICE	39,1	87,5	56,7	12,5	3,2	0	1,0	0
ISO/IEC 9126	34,2	87,5	54,4	12,5	7,5	0	3,9	0
ISO/IEC 12119	36,3	75	56,0	25	5,4	0	2,4	0

Descrição das normas e modelos de maturidade	
ISO/IEC 12207	Information Technology-Software Life Cycle Processes
CMM	Capability Maturity Model/SEI
SPICE	(ISO/IEC TR 15504)- Software Process Improvement and Capability Determination
ISO/IEC 9126	(NBR 13596) - Information Technology- Software Quality Caracteristics and Metrics
ISO/IEC 12119	Information Technology-Software packages – Quality requeriments and testing.

Tabela 2.1 – Comparação do percentual de uso de algumas práticas e ferramentas de engenharia de software da pesquisa realizada (PIV) com relação à da SEPIN (PBQP)

Prática de engenharia de software adotada	PBQP	PIV
Análise critica conjunta	43	25
Gerência de configuração	23,4	12,5
Gerência de requisitos	24,4	12,5
Medições de qualidade (métricas)	17,4	12,5
Métodos estruturados	40,1	25
Métodos orientados a objetos	53,8	12,5
Processo de software definido e documentado	84,7	12,5
Planejamento formal de testes	37,8	12,5
Modelagem de dados	70,1	50
Ferramentas	PBQP	PIV
Analisador de código	12,8	0
Gerador de código-fonte	34	0
Gerador de GUI	24,6	0
Otimizador	7,6	0
Gerenciador de configuração	20,1	12,5

Entre os motivos plausíveis para originar tal situação nesta área em particular, tem-se:

- A aparente facilidade de programação das linguagens gráficas utilizadas na instrumentação virtual leva a uma ilusão de simplicidade de código. No entanto, fica evidente, principalmente em sistemas de complexidade média ou superior, que instrumentos virtuais programados em tais linguagens gráficas são tão difíceis de serem mantidos e entendidos como programas em linguagens textuais.

- As peculiaridades dos sistemas de instrumentação virtual tornam a aplicação direta das soluções tradicionais de engenharia de software não imediata ou simples, devido à falta de metodologia, processos de software, ferramentas e outras técnicas adaptadas a esta área.
- As soluções desta área são fortemente baseadas na experiência dos profissionais envolvidos, não existindo nenhum consenso de processo que possa ser traduzido em uma metodologia para desenvolvimento de instrumentação virtual.
- A distância entre as comunidades de engenharia de software e de medição e automação. Geralmente, os desenvolvedores de sistemas de instrumentação virtual são engenheiros da área de medição e automação, desconhecendo métodos e técnicas de engenharia de software.

Desta forma, é possível acreditar que qualquer solução que procure obter sucesso na aplicação das práticas de engenharia de software à área de instrumentação virtual deva lidar com estes possíveis motivos que levam o setor desta área a este quadro particular. Uma possível linha de ação para uma solução é procurar tornar as práticas de engenharia de software de fácil aplicação para não especialistas nessa área, como por exemplo:

- Todos os recursos devem estar acessíveis ao desenvolvedor de forma simples (documentos, modelos, ferramentas e outros), se possível, de forma integrada;
- O desenvolvedor deve saber o que fazer e em que momento, e ter alguma forma de ajuda ou orientação na execução das atividades e na produção dos artefatos;
- As atividades e procedimentos envolvidos no processo de desenvolvimento devem estar no menor nível de abstração possível, facilitando o entendimento e aplicação às particularidades de instrumentação virtual tal como a programação gráfica;
- O conhecimento no desenvolvimento de sistemas de instrumentação virtual, bem como o conhecimento dos domínios freqüentes de sua aplicação (automação, medição, etc), não devem ser ignorados, pois são fatores chaves no sucesso de projetos nessa área.

1.2 Objetivo da Tese

Este trabalho tem como objetivo apresentar alternativas que pretendem amenizar os problemas no desenvolvimento de software para empresas que trabalhem com esta nova

linha de automação industrial, através de uma abordagem de engenharia para o desenvolvimento destes sistemas que visa tornar tal desenvolvimento disciplinado e repetitível.

Pretende-se alcançar estes objetivos utilizando-se boas práticas de engenharia de software e adaptando-se as mesmas às necessidades específicas destes sistemas, de forma a abordar os requisitos levantados na seção anterior.

A solução proposta será a configuração de um ambiente de desenvolvimento de software orientado à organização através da Estação TABA (VILLELA *et al.*, 2000) (VILLELA, 2004) para empresas de desenvolvimento baseado em instrumentação virtual. A Estação TABA é um meta-ambiente capaz de gerar, através de instanciação e configuração, ambientes de desenvolvimento de software (ADS) adequados às particularidades de processos de desenvolvimento e projetos específicos, as particularidades do domínio e da organização (VILLELA, 2004).

No entanto, a Estação TABA não atende de forma completa todos os requisitos considerados importantes de serem abordados por qualquer aplicação das práticas de engenharia de software à área de instrumentação virtual definidos ao final da seção 1.1.

Portanto, faz parte do escopo deste trabalho, além da configuração e definição de um ambiente de desenvolvimento de software para organizações de instrumentação virtual, a adequação da Estação TABA aos novos requisitos provenientes do que acredita-se necessário para uma solução efetiva do problema. Esta adequação diz respeito principalmente ao que deve ser produzido ao longo do processo de desenvolvimento e de que maneira. A Estação TABA não possuía, ainda, nenhum suporte à documentação para orientar os desenvolvedores sobre que documentos devem ser produzidos em cada atividade envolvida no desenvolvimento, em que momento e de que forma.

Esta adequação levou a inclusão do seguinte tópico no escopo deste trabalho:

- A definição de um processo de documentação a partir da norma ISO/IEC 12207 (NBR ISO/IEC 12207, 1997) para ser utilizado no contexto da Estação TABA com a criação de um apoio computacional específico que inclui uma ferramenta de apoio ao planejamento da documentação, *DocPlan*;.

Os demais tópicos que fazem parte do escopo deste trabalho são proveniente de forma direta ou indireta das necessidades levantadas pela execução do processo de

configuração de ambientes apresentado no anexo II (VILELLA, 2004), que descreve todas as atividades necessárias para a geração de um ambiente configurado para uma organização:

- A definição de um processo de desenvolvimento de software para instrumentação virtual, definindo-se além das atividades/sub-atividades do processo, os artefatos produzidos e roteiros para produção de documentos;
- Configuração de um ambiente para uma organização de desenvolvimento baseado em instrumentação virtual;
- A instanciação de um ambiente de desenvolvimento de software orientado a organização para um projeto de instrumentação virtual.

Em uma análise, preliminar, sobre a efetividade da solução aqui proposta, acredita-se que o ambiente de desenvolvimento de instrumentação virtual, produto final deste trabalho, irá atender os requisitos levantados ao final da seção 1.1 da seguinte maneira:

- O processo de software e o de documentação existentes orientam o desenvolvedor quanto ao que deve ser feito e quando (ROCHA *et al.*, 1990).
- A Estação TABA permite a retenção do conhecimento sobre domínios de aplicação, através da possibilidade de inserção de modelos de ontologias (OLIVEIRA *et al.*, 1999a).
- O conhecimento dos repositórios dos ambientes configurados TABA orienta na execução das atividades do processo de desenvolvimento e serve como memória organizacional para as lições aprendidas ao longo de projetos de desenvolvimento de instrumentação virtual (VILELLA *et al.*, 2000) (VILELLA, 2004).
- O conjunto de ferramentas disponíveis para os ambientes configurados oferece suporte computacional às principais atividades envolvidas no planejamento de projetos de software, facilitando que usuários não especialistas em engenharia de software possam se beneficiar de tais técnicas.
- O ambiente de desenvolvimento integra as diversas ferramentas envolvidas no processo de software, facilitando o acesso e utilização das mesmas (TRAVASSOS, 1994).

Será realizada a configuração de um ambiente para a empresa de desenvolvimento baseado em instrumentação virtual M&D Monitoração e Diagnose Ltda. Esta empresa aplica instrumentação virtual principalmente no desenvolvimento de sistemas inteligentes de apoio à decisão em manutenção preditiva de equipamentos considerados críticos num processo produtivo. Como exemplo, o sistema MDM, constituído de um sistema especialista fuzzy para o diagnóstico de falhas em hidrogeradores, acrescido de um conjunto de ferramentas de análise de vibração para dados oriundos dos sistemas de monitoramento e supervisão.

O ambiente configurado denominado MDM ADS será utilizado nos novos projetos de desenvolvimento de instrumentação virtual da M&D, sendo o primeiro o desenvolvimento de um sistema de instrumentação virtual para supervisão, controle e automação de um laboratório de ensaios em motores elétricos, que dará origem ao Ambiente de Desenvolvimento de Software Orientado a Organização LASEEE.

Embora não mais no escopo do presente trabalho, pretende-se, a partir dos resultados da aplicação dos ambientes MDM ADS e LASEEE, determinar a eficiência da proposta, suas limitações e possíveis melhorias, permitindo posterior aplicação em outras organizações.

1.3 Organização da Tese

Esta tese contém, além desta introdução, mais cinco capítulos e dois anexos.

No segundo capítulo são introduzidos de forma breve os conceitos envolvidos com instrumentação virtual, a motivação de sua existência, metodologias de desenvolvimento empregadas, exemplos de aplicação e o resultado da pesquisa realizada junto a empresas de instrumentação virtual sobre o uso das práticas de engenharia de software.

No terceiro capítulo é apresentada a revisão bibliográfica sobre ambientes de Desenvolvimento de Software Orientados à Organização e a Estação TABA, discutindo-se seus objetivos, requisitos e infra-estrutura de conhecimento, além da revisão sobre processo de software e documentação, conceitos necessários para o entendimento das adequações necessárias a Estação TABA.

No quarto capítulo descreve-se em mais detalhes o trabalho de adequação da Estação TABA para atender as necessidades do desenvolvimento de instrumentação virtual, ou seja, o processo de documentação junto de seu apoio computacional.

O quinto capítulo discute hierarquias de definição de processos de software fora e dentro do contexto da Estação TABA, apresenta o processo de desenvolvimento de instrumentação virtual, o ambiente configurado MDM ADS e o ambiente instanciado para um projeto específico, o ADSOrg LASEEE.

No sexto capítulo são apresentadas as considerações finais deste trabalho, ressaltando suas contribuições, limitações e perspectivas futuras.

O anexo I apresenta o questionário utilizado na pesquisa sobre instrumentação virtual. O anexo II apresenta a processo de configuração de ambientes da estação TABA.

CAPÍTULO II
INSTRUMENTAÇÃO VIRTUAL

Neste capítulo são discutidos os conceitos que envolvem a instrumentação virtual, sua definição, motivação, evolução, metodologias de aplicação e sua relação com as práticas de engenharia de software.

2.1 A Revolução da Medição

Atualmente, a grande evolução dos microprocessadores está levando a uma revolução nos sistemas de automação industrial. Até então, somente sistemas centrados em hardware, por exemplo, CLP's (Controlador Lógico Programável), eram confiáveis o suficiente para aplicações industriais. Com a evolução e o aumento da confiabilidade dos sistemas microprocessados baseados em arquitetura PC (IBM PC), soluções que utilizam esta tecnologia tornaram-se mais atraentes, devido a sua maior flexibilidade e baixo custo. Desta forma, diversos padrões industriais foram criados para suportar periféricos e interfaces para instrumentação baseada em computadores.

Atualmente percebe-se um crescimento da aplicação integrada destes sistemas microprocessados na indústria, inclusive na substituição de sistemas operantes centrados em tecnologias de hardware. Tais sistemas baseados em arquitetura PC, inicialmente simples, foram aumentando seu tamanho e complexidade, à medida que a tecnologia de microprocessadores aumentava sua gama de aplicações em sistemas industriais, tais como sistemas de medição, teste, controle, monitoração, supervisão, entre outros. Estas soluções baseadas em PC, são compostas geralmente de: um sistema de hardware PC genérico (microcomputador PC ou um PC industrial), hardware específico para a aplicação (placas de aquisição, controle, módulos de hardware externos, sensores, etc) e o software de automação (NATIONAL INSTRUMENTS, 2000).

O software tem um papel fundamental dentro desta nova abordagem de automação. Sua demanda e complexidade acompanharam o crescimento do mercado de aplicações na indústria.

2.2 Definição de Instrumentação Virtual

No início dos anos 90, o conceito de instrumento virtual foi introduzido nos sistemas industriais e amplamente adotado por instituições acadêmicas, laboratórios, centros de pesquisa em áreas tais como: automobilismo, aeroespacial, médica e biomédica (SPOELDER *et al.*, 1997, BRIGNELL *et al.*, 1997,RATNER et al, 2000, WANG *et al.*, 2000). A chave do sucesso dos sistemas de instrumentação virtual está em sua maior flexibilidade com relação a soluções de hardware.

Um instrumento virtual é, em princípio, um sistema baseado em computação para testes, medições e controle, que se utiliza de padrões de interfaces de hardware (tais como, PCI, RS-232, IEEE 488) e do poder computacional de uma CPU genérica para cumprir tarefas tais como: controle do fluxo de dados provenientes de periféricos, sensores e interfaces; aquisição, processamento e análise dos dados; gerenciamento das informações, armazenamento, geração de relatórios, gráficos e outros itens (NATIONAL INSTRUMENTS, 2000). Uma definição mais geral é dada por GOLDBERG (2000). Segundo este autor um instrumento virtual é composto por sub-unidades especializadas, computadores de utilidade genérica, software e conhecimento especialista.

Os softwares para desenvolvimento dentro deste novo paradigma de automação são geralmente baseados em tecnologias de programação gráfica. Linguagens de programação gráficas sempre foram populares na comunidade de automação e controle, como por exemplo, programação para CLP (ISO/IEC 61131-3, 1993). Estas tecnologias de programação gráfica são denominadas linguagens G (HILS, 1992, SHU, 1988) ou linguagens de programação gráfico-diagramáticas (*Diagramatic-Graphical Programming Language*) (BRAGG *et al.*, 1994).

GOLDBERG (2000) discute o que um sistema de instrumentação virtual ou sistema de controle deve conter em sua essência. Segundo o mesmo, é necessário um sensor responsável pela percepção do fenômeno a ser medido. Se o parâmetro medido não for elétrico, o sensor deverá conter um transdutor para transformar a informação em um sinal elétrico. Caso este sinal elétrico não esteja em uma condição apropriada, deverá existir um circuito de condicionamento para levar o sinal a um nível em que o mesmo possa ser medido e usado. Este condicionamento pode incluir amplificadores, filtros, linearizadores, e

até mesmo retificadores. Finalmente, deve existir um conversor analógico digital para permitir que o dado seja transformado em um formato digital para posterior manipulação em sistemas computacionais.

Uma vez que o dado esteja em um formato digital, o mesmo pode ser armazenado, processado, alterado, comparado ou manipulado como necessário. Então, pode-se, por exemplo, exibi-lo ou convertê-lo de volta em um sinal analógico para controle de processos. Todas estas operações podem ser realizadas através de um instrumento virtual e processadas em um microcomputador PC padrão. A Figura 1 sintetiza o diagrama de blocos de um sistema de instrumentação virtual.

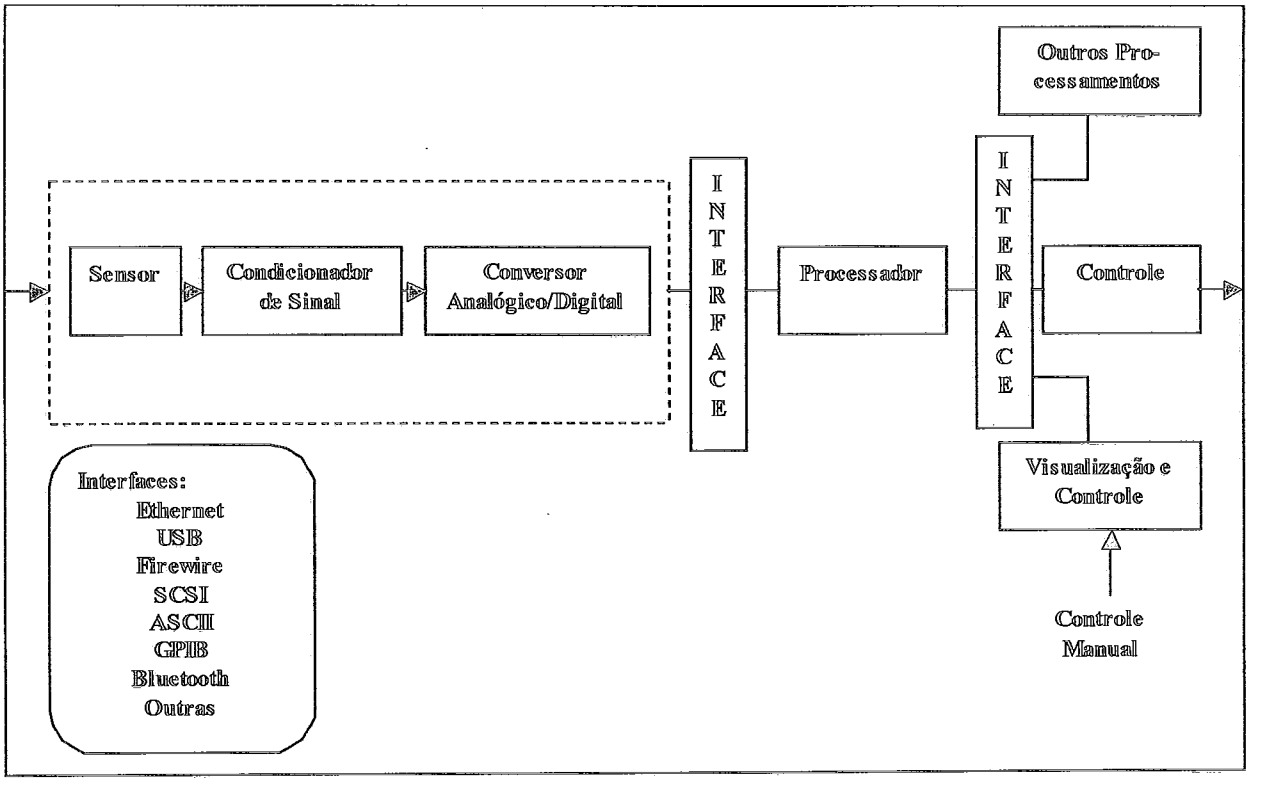


Figura 2.1 –Diagrama de blocos de um sistema de instrumentação virtual.

A tecnologia de instrumentação virtual, quando aplicada ao desenvolvimento de sistemas no escopo de automação, pode aumentar a produtividade de duas a dez vezes com relação a outras tecnologias (NATIONAL INSTRUMENTS, 2000). Entre os benefícios responsáveis por este aumento de produtividade, é possível citar:

- Comunicação com hardware e equipamentos incorporada a linguagem;
- A existência de bibliotecas de rotinas e protocolos comumente necessários em sistemas de automação;
- O uso da tecnologia de programação gráfica.

O uso da programação gráfica é considerado um dos fatores chaves para o aumento da produtividade devido a vantagens tais como:

- Maior facilidade de uso e aprendizado devido a ser mais amigável que linguagens textuais;
- Maior clareza e simplicidade da lógica do programa, pois representa mais fielmente a forma como as pessoas modelam seus problemas. Modelos mentais são comumente representados em formas gráficas, tais como, árvores, gráficos e diagramas de blocos.

Outro fator importante nas linguagens de programação gráfica é que diagramas de blocos de função permitem uma verificação formal mais simples e econômica, por isso é a melhor opção para sistemas de automação críticos, por exemplo, sistemas que em caso de falha levem a riscos de perdas humanas (WANG, 2001).

As linguagens de instrumentação virtual mais populares no mercado atualmente são *HP VEE, Hewlett Packard* (HP-VEE-ENGINE, 1997), e *Labview, National Instruments* (NATIONAL INSTRUMENTS, 2000).

2.3 Programação em Linguagem G

Como já citado na seção anterior, as linguagens de instrumentação virtual são baseadas em programação gráfica, ou linguagens G. A programação gráfica é realizada através da criação de diagramas de blocos. As figuras 2.1 e 2.2 ilustram o diagrama em blocos (código fonte) de aplicações desenvolvidas *em HP VEE* e *Labiew* respectivamente.

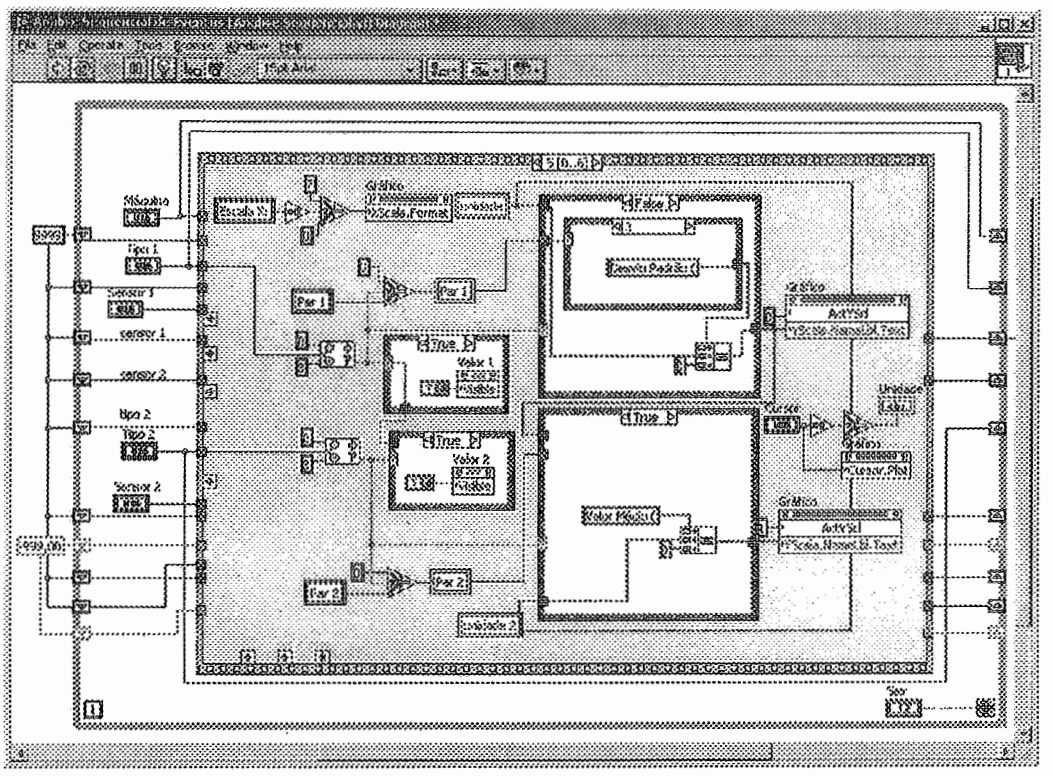


Figura 2.1– Diagrama em Blocos (Código Fonte) de uma aplicação em Labivew®

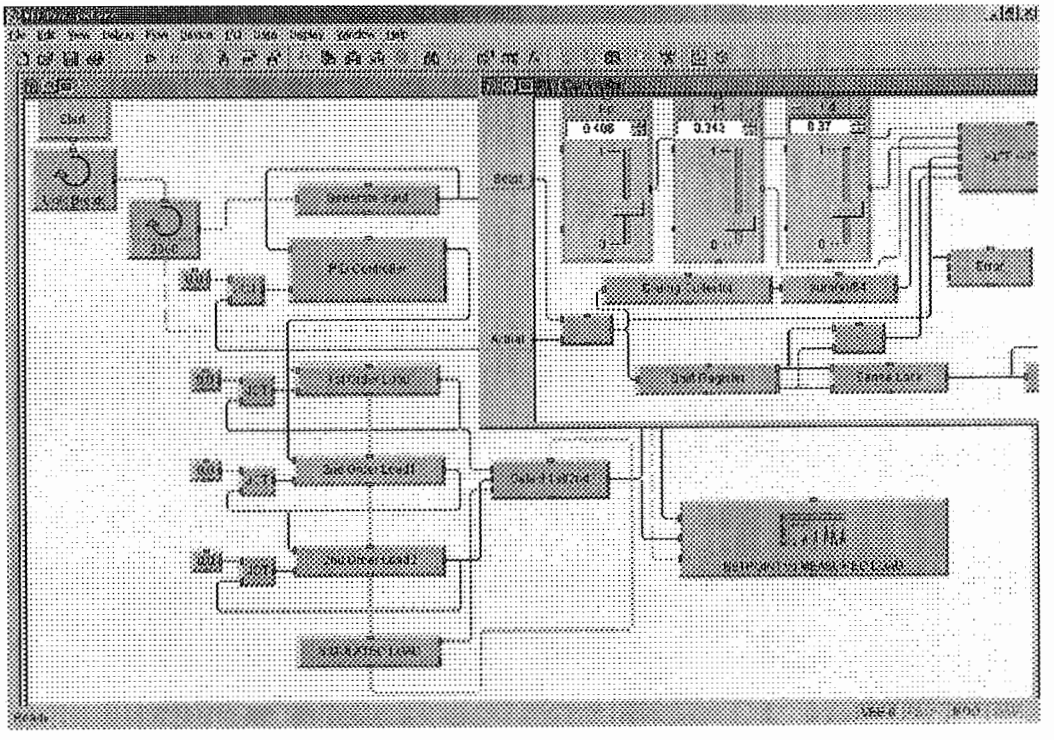


Figura 2.2– Diagrama em Blocos (Código Fonte) de uma aplicação em HP VEE®

Programar em diagramas de blocos é conceitualmente popular entre engenheiros, e a interface visual é semanticamente simples, portanto há menos barreiras para o aprendizado do que uma nova linguagem textual (WANG, 2001). Isto se deve, em parte, a uma simples metáfora: módulos de software podem ser interconectados por canais de software da mesma forma que unidades de hardware podem ser conectadas por cabos carregando sinais analógicos e digitais.

Em um sistema de diagrama de blocos, o programador ou engenheiro coloca na tela ícones que representam módulos de processamento ou blocos. Conexões entre portas de entradas e saídas representam um fluxo de dados de um bloco para outro. Blocos especiais, sem portas de entrada, representam canais de entrada e blocos sem portas de saída representam canais de saída. A maioria dos sistemas suporta construções de redes hierárquicas: um determinado trecho do código pode ser nomeado e representado por um ícone, contendo todas as entradas e saídas necessárias à rede interna como um todo. Isto seria o equivalente a abstração procedural nas linguagens de programação estruturadas.

Apesar de vários blocos comumente utilizados possuírem um alto nível de abstração e grande quantidade de rotinas embutidas, tais linguagens também fornecem o conjunto de primitivas básicas existentes nas linguagens de programação comuns a partir do qual blocos mais complexos podem ser construídos. As linguagens G também permitem a inclusão de blocos construídos em outras linguagens, tais como C ou Pascal, ou a interação com outros aplicativos através das interfaces mais comuns (bibliotecas dinâmicas ou componentes).

O modelo computacional no qual tais sistemas são baseados é denominado "*pipeline data flow*" (HILS, 1992). A combinação entre interface visual e este modelo computacional é bem estabelecida em vários campos, incluindo processamento de sinais (LEE *et al.*, 1994, BARRERA *et al.*, 1991, LAUWEREINS, 1994), processamento e visualização de imagens (RASURE *et al.*, 1992, KONSTANTINIDES *et al.*, 1994), instrumentação (KODOSKY *et al.*, 1991) e linguagens de programação visual genéricas (NAJORK *et al.*, 1990,HILS, 1992).

Repare que no que refere-se a linguagens G, não é possível falar em linhas de código. Assim, é necessário utilizar um outro tipo de métrica para calcular o tamanho de um sistema de instrumentação virtual. O mais simples e comumente empregado é o conceito de nó ou item no diagrama. Qualquer elemento no diagrama é considerado um nó.

Examinando-se o diagrama da figura 2.3, nota-se que ele apresenta 8 nós: 2 varáveis de entrada, 2 operadores, uma variável de saída, uma constante, o laço condicional e sua variável de controle.

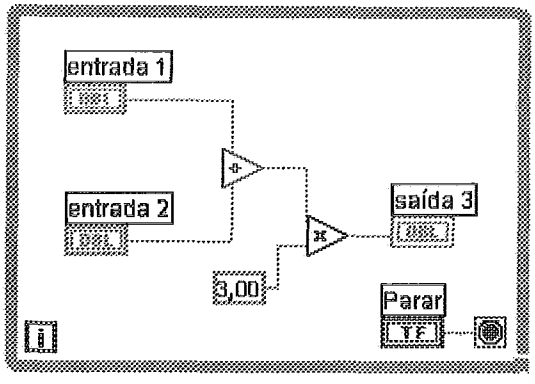


Figura 2.3 – Diagrama de um programa simples em Labview

As métricas para estimativa do tamanho e de complexidade de um instrumento virtual desenvolvido em *Labview* estão listadas na tabela 2.1.

2.4 Uma Sub-Classe Especial em Automação e Medição

Uma classe especial de programação dentro de medidas e automação é chamada de processamento de sinais em tempo real. Este foi um dos campos que mais evoluíram nos últimos tempos, apoiado tradicionalmente por microprocessadores especializados, devido a sua grande necessidade computacional e de tempo de resposta. Atualmente, a revolução dos processadores já permite que várias dessas funcionalidades, antes só possíveis em hardware especializado, sejam executadas por software, ou pelo menos que o hardware especializado seja programado em uma linguagem de alto nível.

Processamento de sinais em tempo real é um campo único e tem sua classe própria de aplicações – aplicações que demandam extremo desempenho computacional e tempos de resposta muito rápidos.

Geralmente, essa necessidade de desempenho acarreta a perda de generalidade, manutenibilidade, portabilidade e outras características de boas práticas de engenharia de software comumente aceitas. Isto implica em que sistemas de processamento de sinais em tempo real devem contar com técnicas e ferramentas de engenharia de software que, além

de se preocuparem com os problemas tradicionais, permitissem uma melhor otimização do código sem a perda de legibilidade e manutenibilidade, ou seja, encontrar o compromisso entre grandezas que geralmente são antagônicas.

Tabela 2.1 – Métricas de tamanho e complexidade de um programa em Labview

Nome	Descrição
Número de nós	São os elementos do diagrama. Isto inclui funções, operações, sub-rotinas, estruturas condicionais e de repetição, variáveis, constantes e etc.
Número de estruturas	Número de estruturas condicionais, de repetição e de seqüenciamento.
Número de diagramas	Número de blocos de diagrama. Cada módulo possui um único diagrama de bloco principal. Em adicional, Existe um sub-diagrama para cada laço, estrutura condicional ou seqüencial.
Profundidade máxima do diagrama	Ultimo nível de profundidade entre os diagramas, se o programa não possui estruturas sua profundidade é 0.
Largura do diagrama (*pixels*)	Largura do diagrama principal em *pixels*.
Altura do diagrama (*pixels*)	Altura do diagrama principal em *pixels*.
Número de fontes de fios de ligação	É o total de fontes no módulo. Cada fio possui uma única fonte, mas pode se dividir para vários destinos.
Número de controles	Número de controles no painel (interface).
Número de indicadores	Número de indicadores no painel (interface).
Número de leituras de propriedades	Número de propriedades de controles ou indicadores lidos no diagrama.
Número de escritas de propriedades	Número de propriedades de controles ou indicadores escritas no diagrama.
Número de leituras de variáveis globais	Número de leituras de variáveis globais no diagrama.
Número de escritas em variáveis globais	Número de escritas em variáveis globais no diagrama.
Número de leituras de variáveis locais	Número de leituras de variáveis locais no diagrama.
Número de escritas em variáveis locais	Número de escritas em variáveis locais no diagrama.
Número de CINs	Número de interfaces com programas escritos em outras linguagens.
Número de chamadas a bibliotecas dinâmicas	Número de chamadas a programas externos.
Número de conectores de entrada	Número de parâmetros de entrada na função que representa o método.
Número de conectores de saída	Número de parâmetros de saída na função que representa o método.

Embora processamento de sinais seja uma aplicação muito comum para sistemas de instrumentação virtual, o presente trabalho não terá nenhuma proposta de solução em termos de engenharia de software para os problemas particulares desta aplicação, apenas levanta a questão com o objetivo de expor ao leitor todas as características da área de instrumentação virtual.

2.5 Engenharia de Software e Instrumentação Virtual

Alguns trabalhos isolados tentam aplicar os conceitos de engenharia de software a instrumentação virtual.

PITTMAN e MILLER (1997) ressaltaram a falta de métricas para linguagens de programação não textuais e particularizaram a complexidade ciclomática de McCabe para aplicação nas estruturas hierárquicas de módulos da linguagem de programação *Labview*. Um módulo constitui-se em um "Instrumento virtual" que pode ser considerado uma sub-rotina.

RABE e MILLER (1997) recomendam algumas práticas a serem realizadas no contexto do desenvolvimento de instrumentos virtuais para sistemas de teste de forma a abordar os requisitos necessários para se alcançar os diferentes níveis do *Capability Maturity Model* (CMM) (ABRAN, 2001).

Atualmente, não existe um paradigma de desenvolvimento adotado pela maioria dos desenvolvedores em linguagens G. Um possível paradigma seria a Análise Estruturada ou Essencial (YOURDON,1989). Através de um diagrama de Fluxo de Dados (DFD) é possível representar os módulos ou instrumentos virtuais e a troca de informações entre os mesmos. Com o uso de um diagrama de Entidade-Relacionamento é possível representar os dados manipulados pelos instrumentos. O diagrama de Transição de Estados permite explicitar a visão do comportamento do instrumento virtual e suas funções de controle.

Por outro lado, embora não seja de aplicação imediata, o desenvolvimento Orientado a Objetos (OO) possui as já conhecidas vantagens em relação ao desenvolvimento estruturado que leva a uma melhor facilidade de manutenção, escalabilidade e maior reuso (YOURDON *et al.*, 1991)

A metodologia OO se torna muito útil para resolver um problema comum em programação gráfica:

- Os projetos são tradicionalmente realizados de forma *top-down*, na qual a programação hierárquica de módulos é realizada de forma a apenas diminuir o tamanho dos módulos, não refletindo o domínio do problema. Isto leva a uma forte dependência entre os módulos devido a não existência de encapsulamento.

No entanto, é necessária a criação de uma camada de tecnologia dependente da linguagem em questão para a aplicação da metodologia OO nessas linguagens gráficas. Esta camada serve para fornecer alguns elementos necessários a um sistema OO, tais como repositórios de objetos e mecanismos de polimorfismo. A metodologia OO aplicada a linguagens G é denominada GOOP *Graphical Object-Oriented Programming* (NATIONAL INSTRUMENTS, 2003). NATIONAL INSTRUMENTS (2003) implementou a infra-estrutura necessária para utilização do GOOP no contexto da linguagem comercial *Labview*®.

A desvantagem da utilização do GOOP está na perda de tempo de processamento que é ocasionada pela execução das rotinas associadas ao repositório de objetos, uma vez que na aplicação de instrumentação virtual para sistemas de controle em tempo real, qualquer perda de tempo de resposta computacional é significativa.

Apesar desta desvantagem assume-se que ela possa ser contornada pelo crescente aumento de desempenho dos processadores, e define-se o paradigma do desenvolvimento Orientado a Objetos como base para o processo de desenvolvimento de instrumentação virtual. Isto se deve principalmente ao fato de que a separação de conceitos e modularidade da Orientação a Objetos são de grande valor na aplicação aos domínios relacionados à instrumentação virtual. Estes domínios possuem um grande número de conceitos com diversas particularidades em comum e um alto grau de reaproveitamento destes conceitos entre aplicações.

2.5.1 Pesquisa sobre Engenharia de Software e Instrumentação Virtual

A evolução da qualidade no desenvolvimento de software nas empresas do Brasil vem sendo acompanhada desde 1993 a partir de pesquisas bienais realizadas pela Secretaria de Política de Informática e Automação (SEPIN) do Ministério da Ciência e Tecnologia (MCT) no âmbito do Subcomitê Setorial da Qualidade e Produtividade em Software, do Programa Brasileiro da Qualidade e Produtividade SSQP/SW-PBPQ. Esta pesquisa, entre outros objetivos, fornece um panorama das práticas, métodos e ferramentas de engenharia de software adotados pelas empresas do setor de informática do país. A população alvo da pesquisa consiste em empresas desenvolvedoras de software, quer seja pacote de software para comercialização, software sob encomenda para terceiros, software embarcado ou software para Internet, além das empresas distribuidoras ou editoras de software de terceiros.

No âmbito deste trabalho, foi realizada uma pesquisa semelhante à pesquisa do SEPIN, mas direcionada a empresas desenvolvedoras de instrumentos virtuais (PIV). A pesquisa teve como objetivo levantar neste setor o uso das práticas de engenharia de software, bem como o conhecimento da existência de normas, ferramentas e demais elementos associados à engenharia de software. Desta forma pretende-se comparar os resultados obtidos por esta pesquisa aos resultados da pesquisa que abrange todo os setores da informática para confirmar, ou então pelo menos ter um indício, que a aparente pouca aplicação das práticas de engenharia de software à instrumentação virtual seja também verificada a partir de um procedimento mais científico.

2.5.1.1 Metodologia da Pesquisa

Objetivo: Ter uma indicação quantitativa da relação do uso das práticas de engenharia de software, bem como o conhecimento da existência de normas, ferramentas e demais elementos associados à engenharia de software, entre as empresas de desenvolvimento de instrumentação virtual e as empresas de software de uma forma geral.

Instrumentação: O instrumento de coleta utilizado foi um questionário (anexo I) elaborado a partir de um resumo das questões existentes nos questionários do SEPIN de 1999 e 2001 (SEPIN, 1999, 2001). Além das perguntas referentes ao conhecimento e ao uso de práticas de engenharia de software, o questionário possui um glossário de termos baseado no SEPIN (2001) e no trabalho de ODDO (2003). O questionário foi enviado por e-mail para as empresas participantes.

Seleção de contexto e Indivíduos: Os questionários foram enviados por e-mail e preenchidos sem acompanhamento e sem obrigação de identificação, portanto, o preenchimento foi voluntário e realizado no tempo e ambiente escolhidos pelo participante. Os indivíduos foram selecionados por conveniência e disponibilidade, entre um grupo aleatório de empresas distribuídas em diversas regiões do país, que anunciam entre suas capacitações, desenvolvimento ou manutenção de sistemas escritos em *Labview* ou *HP VEE*. Estas empresas foram levantadas a partir da lista de empresas oficialmente habilitadas pelos fabricantes, tanto do *Labview*, quanto do *HP VEE*, como desenvolvedores oficiais de soluções utilizando seus produtos, além de outras empresas que anunciam em seus serviços este tipo de desenvolvimento. Responderam a pesquisa três empresas do Rio de Janeiro, duas empresas de São Paulo e três não identificadas.

2.5.1.2 Consolidação dos Resultados

A tabela 2.2, em 100% dos questionamentos, indica um maior desconhecimento sobre modelos de maturidade e normas relacionadas à qualidade de software para as empresas de desenvolvimento baseado em instrumentação virtual. A tabela 2.3 mostra que 90% das 20 práticas de engenharia de software listadas e 80% das 21 ferramentas são menos utilizadas em empresas de instrumentação virtual, ou seja, as empresas de software de aplicação geral possuem um percentual de uso maior de tais técnicas e ferramentas. O mesmo ocorre com 65% dos tipos de documentos listados.

Tabela 2.2 – Comparação do percentual de conhecimento de normas e modelos de maturidade da pesquisa realizada (PIV) com relação à pesquisa da SEPIN (PBQP)

Grau de conhecimento	Não conhece		Conhece mas não usa		Conhece e está começando a usar		Conhece e usa	
Norma/Modelo	PBQP	PIV	PBQP	PIV	PBQP	PIV	PBQP	PIV
ISO/IEC 12207	32,7	75	55,1	25	8,3	0	3,9	0
CMM	25,3	62,5	53,7	37,5	17,1	0	3,9	0
SPICE	39,1	87,5	56,7	12,5	3,2	0	1,0	0
ISO/IEC 9126	34,2	87,5	54,4	12,5	7,5	0	3,9	0
ISO/IEC 12119	36,3	75	56,0	25	5,4	0	2,4	0
Descrição das normas e modelos de maturidade								
ISO/IEC 12207	Information Technology-Software Life Cycle Processes							
CMM	Capability Maturity Model/SEI							
SPICE	(ISO/IEC TR 15504)- Software Process Improvement and Capability Determination							
ISO/IEC 9126	(NBR 13596) - Information Technology- Software Quality Caracteristics and Metrics							
ISO/IEC 12119	Information Technology-Software packages – Quality requeriments and testing.							

Mesmo com um número pequeno de empresas (oito organizações), a pesquisa confirma seu objetivo quando comparada a SEPIN (2001): empresas de desenvolvimento de instrumentação virtual desconhecem ou/e utilizam menos as práticas, técnicas, metodologias e ferramentas de engenharia de software. Note que o objetivo da pesquisa não foi determinar o panorama da aplicação de engenharia de software em empresas de desenvolvimento baseado em instrumentação virtual, de forma similar ao que o SEPIN realiza com empresas de desenvolvimento diverso, e sim servir como mais uma referência da particularidade da área de instrumentação virtual. Embora o intervalo de confiança desta pesquisa seja grande devido ao reduzido número de amostras, a grande disparidade de valores obtidos por ambas as pesquisas mostra que mesmo que os valores não sejam exatamente esses, fica visível que, em média, as empresas de instrumentação virtual aplicam menos práticas, ferramentas e documentação associados à engenharia de software.

Tabela 2.3 – Comparação do percentual de uso de algumas práticas e ferramentas de engenharia de software da pesquisa realizada (PIV) com relação à da SEPIN (PBQP)

Prática de engenharia de software adotada	PBQP	PIV	Ferramentas	PBQP	PIV
Auditoria	22,6	12,5	Analisador de código	16,3	0
Análise crítica conjunta	43	25	CASE Lower	22,5	0
Controles de versão de produto	69,1	50	CASE Upper	22,7	0
Engenharia da informação	21,3	0	Depurador interativo	39	50
Gerência de configuração	23,4	12,5	Distribuição de software	22,7	25
Gerência de requisitos	24,4	12,5	Documentador	27	37,5
Gerência de projetos	64,7	75	Driver de teste	11,1	0
Inspeção Formal	16,3	0	Gerador de código-fonte	34	0
Medições de qualidade (métricas)	17,4	12,5	Gerador de dados de teste	10,9	0
Processo de software definido e documentado	84,7	12,5	Gerador de entrada de dados	10,9	0
Gestão de mudança	10,4	0	Gerador de gráficos	20,6	0
Joint Application Design – JAD	8,1	0	Gerador de GUI	24,6	12,5
Métodos estruturados	40,1	37,5	Gerador de relatórios	49,2	12,5
Métodos orientados a objetos	53,8	12,5	Gerador de telas	31,7	12,5
Modelagem de dados	70,1	50	Gerenciador de bibliotecas de módulos	23,4	12,5
Normas e padrões da organização	40,1	75	Gerenciador de configuração	20,1	25
Planejamento formal de testes	37,8	12,5	Gerenciador de conteúdo	7,6	0
Projeto da interface com o usuário	56,6	50	Gerenciador de documentos	19,4	0
Prototipação	51	50	Gerenciador de projetos	38,5	25
Walkthrough estruturado	16,3	0	Otimizador	7,6	0
			Prototipador	13,5	12,5

Este trabalho não discute a utilidade e eficácia de tais métodos e ferramentas para o aumento da produtividade e da qualidade do produto, embora admita-se que, de uma forma geral, tal correlação existe. Uma discussão mais ampla de tal correlação (práticas e ferramentas de engenharia de software com qualidade do produto) pode ser encontrada em ODDO (2003), onde se relata uma pesquisa com especialistas para determinar quais práticas, métodos, ferramentas e documentos listados no questionário do SEPIN têm associação com a qualidade do produto final.

Tabela 2.4 – Comparação do percentual da adoção de tipos de documentação da pesquisa realizada (PIV) com relação à da SEPIN (PBQP)

Documentação adotada	*PBQP*	*PIV*	Documentação Adotada	*PBQP*	*PIV*
Contratos e acordos	68	87,5	Help on-line	65,4	50
Documentação de marketing	37,1	37,5	Manual de treinamento	47,5	25
Documentação de programas	62,9	75	Manual do usuário	68,7	75
Documentação do processo de software	35,5	25	Plano de controle da qualidade	15	0
Documentação no código	62,7	37,5	Plano de testes	35,3	12,5
Descrição do produto p/ comercialização	53,2	75	Projeto do sistema	46,5	37,5
Especificação do sistema	61,5	50	Registro formal de revisões e testes	30,2	12,5
Guia de instalação	57,4	62,5			

2.6 Considerações Finais

Este capítulo apresentou os conceitos e definições que norteiam instrumentação virtual, assim como sua utilidade, domínios de aplicação e particularidades. Foi também discutido o conceito de linguagem G e sua estreita relação com instrumentação virtual. Foi descrito como a engenharia de software é aplicada a este tipo de software e apresentada uma breve revisão bibliográfica sobre o assunto. A pesquisa sobre engenharia de software e instrumentação virtual foi detalhada, tornando possível identificar o menor uso de práticas e ferramentas de engenharia de software pelas organizações desenvolvedoras deste tipo de software.

Uma vez já introduzidos o conceito e a utilidade de instrumentação virtual, o próximo capítulo discutirá os conceitos de ambiente de desenvolvimento de software e da Estação TABA. Assim, será possível entender como a Estação TABA poderá atender o desenvolvimento de instrumentação virtual e o que é preciso para isso, o que, em suma, constitui-se no objetivo dessa tese.

CAPÍTULO III
ESTAÇÃO TABA E DOCUMENTAÇÃO DE SOFTWARE

Este capítulo apresenta uma breve revisão bibliográfica dos conceitos relacionados a Estação TABA que são importantes para o seu entendimento, bem como dos conceitos relacionados aos requisitos ainda não atendidos e que são importantes para a instrumentação virtual.

Introdução

Na era da informação, conhecimento tem sido considerado como o patrimônio mais importante de uma organização, sendo fator estratégico e de influência decisiva em sua competitividade (WINCH, 1999; O'LEARY, 1998; LEE *et al.*, 2001; ABECKER *et al*, 1998, RUS e LINDVALL, 2002) e, por isso, sua formalização, captura e reutilização devem ser incentivadas. Desta forma, a introdução do conceito de gerência do conhecimento no contexto de desenvolvimento de software é fundamental para as organizações envolvidas neste tipo de atividade.

Ambiente de Desenvolvimento de Software (ADS) é um sistema computacional que provê suporte para o desenvolvimento e manutenção de software e para o gerenciamento e controle destas atividades, contendo uma base de dados central e um conjunto de ferramentas de apoio.

O estudo de ADS teve início na década de 50 e evoluiu rapidamente ao longo dos anos. As pesquisas iniciais em ADS visavam o desenvolvimento de ferramentas de automação do processo de desenvolvimento de software. Atualmente, os estudos em ADS exploram o desenvolvimento de ferramentas integradas para apoiar o desenvolvedor de software na execução das atividades do processo de desenvolvimento. Neste contexto, o Grupo de Engenharia de Software da COPPE/UFRJ iniciou na década de 90 estudos em ambientes de desenvolvimento de software com o objetivo de definir e criar a Estação TABA, um meta-ambiente de desenvolvimento de software capaz de gerar, através de instanciação, outros ADSs (ROCHA *et al.*, 1990). O conceito de ADS evoluiu para a

definição de ADS orientado a domínio (ADSOD), no qual é provido suporte à utilização do conhecimento do domínio de aplicação durante o desenvolvimento (OLIVEIRA *et al*, 1999a).

O conceito de Ambiente de desenvolvimento de Software Orientados a Organização (ADSOrg) surge então através da união dos conceitos de gerência de conhecimento com ADS. Um ADSOrg tem por objetivo dar apoio ao processo de gerência do conhecimento em organizações que desenvolvem produtos de software, provendo uma infra-estrutura de apoio aos desenvolvedores de forma a disponibilizar o conhecimento obtido e evoluído pela organização ao longo do tempo (VILLELA *et al.*, 2001b). Para acompanhar essa evolução, a Estação TABA passou a instanciar, também, ambientes de desenvolvimento de software orientados a organização (ADSOrg).

Pela sua própria definição, um ADSOrg tem como objetivo o aumento da qualidade do processo de desenvolvimento de software através de suas características de agente facilitador do processo. No capítulo I, propõe-se a configuração de um ambiente de desenvolvimento de software orientado a organização para diminuir as dificuldades no desenvolvimento de instrumentos virtuais. Características particulares desta área, tal como a ausência de sólidos conhecimentos em engenharia de software por parte dos participantes do processo de desenvolvimento, levam a crer que o aumento da eficiência ocasionado pela utilização de um ambiente será maior quando aplicado a instrumentação virtual, uma vez que a lacuna de deficiência de conhecimento é maior. Vale ressaltar que não está sendo dito que o ambiente será utilizado de forma mais eficiente, mas que o ganho ocasionado pelo seu uso poderá ser maior.

Todavia, para que um ADSOrg concretize este papel de agente facilitador dos aspectos importantes envolvidos no desenvolvimento de software, o que foi considerado fator chave na adoção do mesmo como proposta de solução para os problemas da área de instrumentação virtual, é essencial dispor-se de apoio à documentação.

È bem conhecido e aceito pela industria de software o fato de que os custos de manutenção dos produtos de software geralmente respondem por pelo menos 50% do custo total de seu ciclo de vida. Muitas evidências de ordem prática também sugerem que a maior parte do tempo dos profissionais de informática seja gasta no entendimento de sistemas existentes e não efetivamente no desenvolvimento de novos. Estes fatos ressaltam a

importância da documentação no processo de desenvolvimento de software como agente facilitador do processo inevitável de manutenção.

Por isso, o processo de documentação é fundamental dentro do conceito de ambiente de desenvolvimento de software e portanto, de instrumentação virtual. O processo de documentação apoiado computacionalmente dentro de um ambiente permitirá que esse realize de forma mais eficiente seu papel de agente facilitador do desenvolvimento.

Este requisito não atendido pela Estação TABA: apoio ao planejamento e produção de documentação, será abordado por este trabalho. Uma breve discussão bibliográfica sobre o assunto é apresentada neste capítulo e a solução adotada na Estação TABA para este dois problema é o alvo do próximo capítulo.

Na próxima seção apresentaremos o objetivo, as características e os requisitos da Estação TABA. As seções 3.1.1 e 3.1.2 comentam, respectivamente, sobre a evolução dos ambientes de desenvolvimento de software com a utilização de conhecimento sobre o domínio durante o processo de desenvolvimento e com a inclusão de conhecimento organizacional. A seção 3.1.5 aborda o modelo atual e a implementação da Estação TABA, além de algumas das ferramentas integradas já implementadas por ocasião da definição dos ADSOD e dos ADSOrg. A seção 3.2 discute o requisito não tratado pela Estação TABA considerado importante para a instrumentação virtual: documentação. A mesma apresenta, de forma breve, a importância de documentação e alguns aspectos existentes a seu respeito na literatura. Esta seção também discute processo de software, que é a espinha dorsal de um ADSOrg, além de apresentar a Norma ISO 12207 cujos processos servem de base para os processos de desenvolvimento, manutenção e documentação definidos neste trabalho. A seção 3.5 apresenta algumas considerações finais.

3.1 Estação TABA e Ambientes de Desenvolvimento de Software

3.1.1 Ambientes de Desenvolvimento de Software

Um Ambiente de Desenvolvimento de Software (ADS) é definido como sendo um sistema computacional que provê suporte para o desenvolvimento, manutenção e melhorias em software e para o gerenciamento e controle dessas atividades (MOURA e ROCHA, 1992). Para alcançar tal objetivo o ADS deve conter um repositório com todas as

informações relacionadas com o projeto de software ao longo do seu ciclo de vida, além de possibilitar o desenvolvimento e integração de ferramentas de apoio à execução das atividades do processo de desenvolvimento de software.

TRAVASSOS (1994) enfatiza que os ADS devem se preocupar com o apoio às atividades individuais e ao trabalho em grupo, o gerenciamento de projeto, o aumento da qualidade geral dos produtos e o aumento da produtividade, permite ao engenheiro de software acompanhar o projeto e medir a evolução dos trabalhos através de informações obtidas ao longo do desenvolvimento.

ADSs centrados em processo (GARG e JAZAERI, 1995) baseiam-se na idéia de integração e incorporação de ferramentas ao processo de software para amenizar os problemas de desenvolvimento *ad hoc*. Entende-se por processo de desenvolvimento de software um conjunto bem definido e ordenado de atividades, somado aos recursos utilizados e produzidos e ao conjunto de ferramentas e técnicas para apoio à realização destas atividades (PFLEEGER, 2001).

Pesquisas recentes têm mostrado a necessidade de padronizar a forma como é desenvolvido software dentro de uma organização com o objetivo de aumentar o controle e possibilitar melhorias dos processos de desenvolvimento de software (EMAN *et al.*, 1998) (NBR ISO/IEC 12207, 1997) (MAIDANTCHIK, 1999). Isto implica em um novo requisito para qualquer ADS, permitir que o processo de desenvolvimento de software seja realizado segundo os padrões organizacionais.

A padronização dos processos de desenvolvimento de software pode ser obtida através da definição de um processo padrão, ou seja, um processo básico que guia o estabelecimento de um processo comum dentro da organização (EMAN *et al.*, 1998). A partir deste processo, pode-se definir processos de desenvolvimento de software especializados específicos para o tipo de software (tecnologia e paradigma utilizado) e as características do desenvolvimento. O processo especializado por sua vez, pode ser instanciado considerando-se as particularidades do projeto de software. Desta forma, um ADS deve ser desenvolvido de forma a guiar os desenvolvedores de software na execução do processo de desenvolvimento instanciado para um projeto de software.

3.1.2 A Estação TABA

A Estação TABA (ROCHA *et al.*, 1990), é um meta-ambiente capaz de gerar, através de instanciação, ambientes de desenvolvimento de software adequados às particularidades de processos de desenvolvimento e de projetos específicos. ROCHA *et al.* (1990) definem meta-ambiente como um ambiente que abriga um conjunto de programas que interagem com os usuários para definir interfaces, selecionar ferramentas e estabelecer os tipos de objetos que irão compor o ambiente de desenvolvimento específico.

O projeto TABA foi criado a partir da constatação de que domínios de aplicação diferentes possuem características distintas e que estas devem incidir nos ambientes de desenvolvimento através dos quais os desenvolvedores de software desenvolvem aplicações. Desta forma, a Estação TABA tem por objetivo auxiliar na definição, implementação e execução de ADS adequados a contextos específicos. Com o intuito de atender a este objetivo, quatro funções foram definidas originalmente para a Estação TABA (TRAVASSOS, 1994):

(i) Auxiliar o engenheiro de software na especificação e instanciação do ambiente mais adequado ao desenvolvimento de um produto específico a partir do processo de software e/ou de uma definição do domínio de aplicação;

(ii) Auxiliar o engenheiro de software na implementação das ferramentas necessárias ao ambiente definido;

(iii)Permitir aos desenvolvedores do produto de software a utilização da estação através do ambiente desenvolvido;

(iv)Permitir a execução do software na estação configurada para o seu desenvolvimento.

A Estação TABA, em sua primeira concepção, permitia a instanciação de um ADS a partir da definição do processo de desenvolvimento de software.

3.1.3 Ambientes de Desenvolvimento de Software Orientados a Domínio

Entre as atividades envolvidas no processo de desenvolvimento de software, o entendimento correto e completo do problema ao qual o software pretende resolver é

considerado fator chave de sucesso. A dificuldade em tal entendimento provem de vários fatores, entre os quais destaca-se que problemas complexos exigem mais conhecimento do que uma só pessoa possa ter, o que faz com que a comunicação e a colaboração tomem-se fundamentais. Esta atividade é ainda mais difícil quando os desenvolvedores não conhecem o domínio ou não tem nenhuma experiência em desenvolver software para o mesmo (OLIVEIRA *et al.*, 1999a). Entende-se por domínio uma área de aplicação na qual vários produtos de software serão desenvolvidos. Além do mais, muito do conhecimento do domínio adquirido durante o desenvolvimento de um software é perdido devido à alta rotatividade em projetos de desenvolvimento de software, o que acarreta repetidas investigações do mesmo domínio a cada ciclo de desenvolvimento de um software na organização.

Uma possível solução para este problema está no entendimento compartilhado do conhecimento do domínio durante todo o processo de desenvolvimento, o que facilitaria a compreensão do domínio e, conseqüentemente, melhoraria a produtividade. Assim, surgiram, no contexto da Estação TABA, os ambientes de desenvolvimento de software orientados a domínio (ADSOD), que buscam integrar o conhecimento do domínio aos ambientes de desenvolvimento de software (OLIVEIRA *et al.*, 1999) e que podem ser vistos como uma evolução dos ambientes de desenvolvimento de software tradicionais. Desta forma, o aspecto central dos ADSOD é a introdução e uso do conhecimento do domínio no ADS.

3.1.4 Ambientes de Desenvolvimento de Software Orientados a Organização

ADSOD estende o conceito de ADS para permitir a inserção, entendimento e recuperação do conhecimento sobre o domínio. No entanto, o conhecimento necessário para uma melhoria do desenvolvimento de software vai além do conhecimento do domínio. A incorporação de conhecimento organizacional, como normas, diretrizes organizacionais, melhores práticas e relatos de experiências, motivou a evolução dos ADSOD para Ambientes de Desenvolvimento de Software Orientados a Organização (ADSOrg). O objetivo destes ambientes é apoiar o gerenciamento do conhecimento requerido em uma atividade de engenharia de software, evitando que este conhecimento fique disperso ao

longo da estrutura organizacional e, conseqüentemente, sujeito a dificuldades de acesso e, mesmo, a perdas (VILLELA, 2004).

ADSOrgs possuem os seguintes objetivos (VILLELA, 2004):

- Prover para os desenvolvedores de software o conhecimento acumulado pela organização e relevante no contexto do desenvolvimento de software;
- Apoiar o aprendizado organizacional neste contexto.

As funções originais da Estação TABA foram revistas e ampliadas para satisfazer essas necessidades. Suas funções atuais são (VILLELA, 2004):

(i) Auxiliar o engenheiro de software na configuração do ambiente mais adequado para apoiar o desenvolvimento e a manutenção de software em uma organização específica (Ambiente Configurado), considerando seu processo de software e a gerência do conhecimento organizacional relevante neste contexto;

(ii) Auxiliar o engenheiro de software na instanciação de ambientes de desenvolvimento de software para projetos específicos (caso a configuração de um ambiente para organização não seja possível ou considerada adequada);

(iii) Auxiliar os gerentes de projeto na instanciação de ambientes de desenvolvimento de software para projetos específicos a partir de um ambiente configurado;

(iv) Auxiliar o engenheiro de software de empresas cujo negócio é o desenvolvimento e a manutenção de software para diversos clientes na especialização de processos da sua empresa de acordo com as particularidades de um cliente específico;

(v) Auxiliar o engenheiro de software a implementar ferramentas necessárias aos ambientes;

(vi) Apoiar, através dos ambientes instanciados, o desenvolvimento e a manutenção de software, bem como a gerência destas atividades;

(vii) Permitir a execução do software na própria Estação, pelo menos para fins de teste.

A figura 3.1 ilustra o esquema de instanciação e configuração de ADS na Estação TABA.

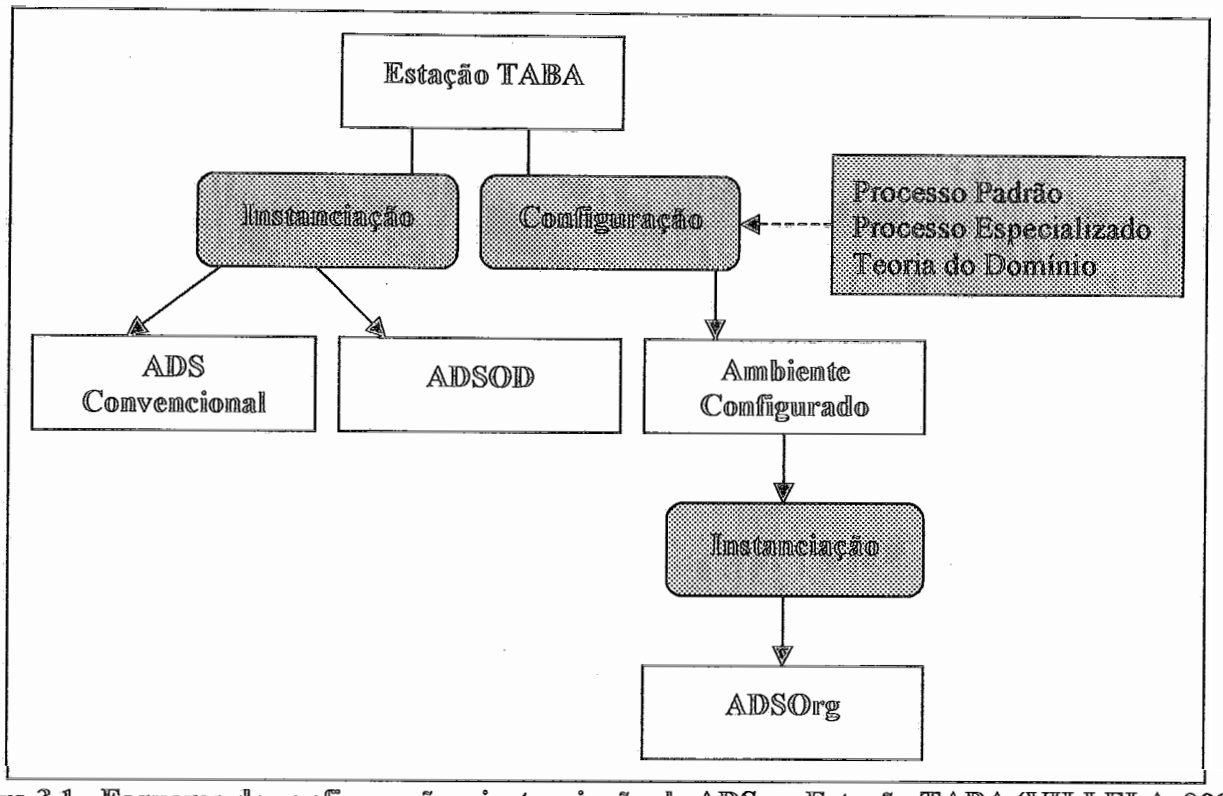


Figura 3.1 - Esquema de configuração e instanciação de ADS na Estação TABA (VILLELA, 2004).

A instanciação de ADSs convencionais e orientados a domínio continua sendo possível diretamente a partir da Estação TABA. A inclusão de novos requisitos na Estação TABA tornou possível sua configuração para uma determinada organização através da definição de seus processos padrões e especializados, bem como de teorias de domínio de interesse. O ambiente configurado então permite a instanciação de ADSOrg para os projetos de software da organização.

3.1.5 Implementação Atual

A implementação da Estação TABA foi iniciada em 1994 a partir do trabalho de TRAVASSOS (1994). No início, foi desenvolvida utilizando-se a linguagem Eiffel numa estação de trabalho da *Sun Microsystems*®. Com o passar do tempo, percebeu-se que o poder e a robustez da plataforma, apesar de adequada ao ambiente de pesquisas, dificultava a experimentação das idéias em outros ambientes devido à falta de portabilidade do código para plataformas mais acessíveis e comumente utilizadas. Com o intuito de solucionar este

problema, foi realizada uma re-implementação na qual se optou pela plataforma de microcomputadores e pelo uso da linguagem C++ (OLIVEIRA, 1999) (SANTOS e ZLOT, 1999).

Com a re-implementação, o modelo de dados foi revisto e avaliado de forma a limitar o trabalho de implementação a funcionalidades básicas, suficientes para a especificação, instanciação e execução de ambientes de desenvolvimento de software. A esse subconjunto foram acrescentadas novas características necessárias à definição e instanciação de ADSOD (OLIVEIRA, 1999).

Os novos requisitos introduzidos pela criação do ADSOrg levaram a uma nova revisão do modelo para adequá-lo ao esquema de configuração/instanciação de ambientes já discutido na seção anterior, além da criação de novas ferramentas. Por exemplo, foi desenvolvida a ferramenta *Config* (VILLELA, 2004), para atender à configuração de ambientes para organizações e a ferramenta *AdaptPro* (BERGER, 2003) para apoiar a instanciação do processo de desenvolvimento de software específico para um projeto, o ADSOrg. A tabela 3.1 apresenta os serviços e as ferramentas atualmente integradas aos ambientes TABA que auxiliam os desenvolvedores de software na definição de processos de desenvolvimento e na execução das atividades de desenvolvimento de software.

Tabela 3.1 - Ferramentas dispomíveis nos ambientes TABA.

FERRAMENTA	DESCRIÇÃO	REFERÊNCIAS
(serviço interno da Estação TABA)	Definição e instanciação de ADS e ADSOD	(OLIVEIRA, 1999) (SANTOS E ZLOT, 1999)
AssistPro	Definição, especialização e instanciação de processos em ADS.	(FALBO *et al.*, 1999) (FALBO, 1998)
EditPro	Definição, especialização e instanciação de processos em ADSOD.	(SANTOS e ZLOT, 1999)
DefPro	Definição, especialização e instanciação de processos em ADSOD.	(MACHADO, 2000) (MACHADO *et al.*, 2000a) (MACHADO *et al.*, 2000b)
Config	Configuração de ambientes para organizações.	(VILLELA, 2004)
EdiTeD	Definição de teorias de domínio.	(OLIVEIRA *et al.*, 2000)
EdiTar	Definição de teorias de tarefas.	(OLIVEIRA, 1999) (ZLOT, 2002) (ZLOT *et al.*, 2002)

GENESIS	Apoio à atividade de Investigação do Domínio.	(GALOTTA, 2000)
NAVEGUE	Apoio à atividade de Investigação do Domínio.	(GALOTTA, 2000)
REGCON	Apoio à atividade de Investigação do Domínio.	(GALOTTA, 2000)
(serviço interno da Estação TABA)	Avaliação de artefatos e de processos de software.	(GALOTTA, 2000)
RiscPlan ou RiscManager (nome em nova versão)	Apoio ao planejamento da gerência de riscos do projeto.	(FARIAS *et al.*, 2001) (FARIAS, 2002)
CustPlan ou CustManager (nome em nova versão) TempPlan ou TempManager (nome em nova versão)	Apoio ao planejamento de tempo e custos do projeto.	(BARCELLOS, 2003)
RHPlan ou RHManager (nome em nova versão)	Apoio ao planejamento de recursos humanos do projeto.	(SCHNAIDER, 2003)
Sapiens	Descrição do conhecimento organizacional.	(VILLELA *et al.*, 2001c) (SANTOS *et al.*, 2002) (VILLELA *et al.*, 2001a)
AdaptPro	Apoio à instanciação de processos de desenvolvimento de software específicos para um projeto (ADSOrg).	(BERGER, 2003)
Acknowledge	Apoio ao processo de gerência de conhecimento	(MONTONI, 2003)
GConf	Apoio ao planejamento de gerência de configuração	
QualityPlan	Apoio ao planejamento da qualidade	
ControlPlan	Apoio ao planejamento do acompanhamento e controle	
OrgPlan	Apoio ao planejamento da organização do projeto	
Metrics	Coleta de métricas baseado no plano de medição	
MedPlan	Gera o plano de medição da organização e do projeto	
ProjectStatus	Provê o status dos projetos em desenvolvimento da organização	
ActionPlanManager	Criação de planos de ação ao longo do projeto	
ControlManager	Apoio ao controle e planejamento do processo	
ProcView	Visualização de processos da organização e do projeto	

QFuzzy	Ferramenta para avaliação de qualidade	
QualityPlan	Planejamento do controle de qualidade do projeto	
GConf	Gerência de configuração	(FIGUEIREDO, 2004)

3.2 Requisitos para Ambiente de Instrumentação Virtual Não Atendidos pela Estação TABA

O capítulo I discutiu, justificou e apresentou alguns requisitos que acredita-se necessários para se obter sucesso na utilização das práticas de engenharia de software por desenvolvedores de instrumentação virtual, tais como:

- O desenvolvedor deve saber o que fazer e em que momento, e ter alguma forma de ajuda ou orientação na execução das atividades e na produção dos artefatos;
- O conhecimento no desenvolvimento de sistemas de instrumentação virtual, bem como o conhecimento dos domínios freqüentes de sua aplicação (automação, medição, etc), não devem ser ignorados, pois são fatores chaves no sucesso de projetos nessa área.

É possível concluir que tais objetivos seriam contemplados configurando-se um meta-ambiente de desenvolvimento de software orientado à organização a partir da Estação TABA. No entanto, a Estação TABA ainda não atendia a estes requisitos de forma satisfatória.

Um processo de apoio à documentação permitiria ao desenvolvedor saber que documentos devem ser produzidos, em que momento e de que forma para cada atividade.

A Estação TABA baseia-se na idéia de integração e incorporação de ferramentas ao processo de software. Portanto, um processo de software definido de acordo com as necessidades do desenvolvimento constitui-se fator chave para que esta alcance seus objetivos.

Estas 2 questões: documentação e processo de software para organizações que desenvolvem instrumentação virtual, foram trabalhadas no contexto da Estação TABA e são produto deste trabalho. Nas próximas seções será realizada uma breve discussão bibliográfica sobre documentação e processos de software, com o objetivo de

contextualizar o leitor a respeito de tais conceitos, tornando-o apto a entender as adequações necessárias a Estação TABA.

3.2.1 Documentação de Software

Os problemas relacionados à manutenção de software e já discutidos neste capítulo ressaltam a importância da documentação no processo de desenvolvimento de software como agente facilitador do processo de manutenção.

Em uma análise mais profunda pode-se notar que o problema não está somente na produção dos documentos, mas também na forma e com qual qualidade estes são feitos. Estudos empíricos têm mostrado que uma documentação pobre, de má qualidade, desatualizada e incompleta é a maior causa de defeitos no desenvolvimento e manutenção de software (COOK e VISCONT, 1996, PENCE e HON, 1993). Sendo assim, documentação é um componente chave para a qualidade do software e melhorar o processo de documentação implica em considerável impacto na qualidade do produto final, o software.

Quando se fala sobre documentação, talvez as questões mais em aberto, e portanto, suscetíveis a discussões sejam:

- Quanto de documentação é necessário?
- Qual é a utilidade de dado documento? Ou talvez, quais documentos possuem o custo de produção inferior ao benefício gerado pelo mesmo?
- Que nível de detalhamento e precisão é necessário para um dado documento?
- Como manter a documentação atualizada e consistente?

Percebe-se que estas perguntas têm um alto grau de correlação, sobrepondo-se umas às outras e é difícil discutir qualquer uma delas de forma isolada. A literatura apresenta inúmeras propostas para se amenizar uma ou mais destas questões. O que pode-se dizer é que estas questões não podem ser investigadas em um alto nível de generalização e são sensíveis ao contexto no qual a documentação está sendo aplicada (BRIAND , 2003).

Dentre os fatores que mais impactam na quantidade de documentação necessária pode-se destacar o tamanho do projeto, a criticidade do sistema em desenvolvimento, ou

seja, os riscos associados em caso de falha e a necessidade de manutenção futura entre outros.

SMITH (2003) discute o quanto de processo é necessário, e por conseqüência documentação, na aplicação de grandes *framework* de processos. Para isto, o autor faz uma comparação da necessidade de documentação da metodologia RUP (KRUCHTEN,1999) e do método ágil XP (BECK, 2000), no caso em que ambos são aplicados a pequenos projetos de software.

Mesmo uma documentação incompleta e desatualizada permanece útil em algumas circunstancias. LETHBRIDGE *et al.* (2003) conduziram uma pesquisa com diferentes empresas e concluíram que quanto mais alto o grau de abstração de um documento menor é a correlação entre sua acurácia e sua freqüência de uso. Isto é, documentos relacionados a código e projeto detalhado, por exemplo, são considerados úteis por engenheiros de software se, e somente se, atualizados freqüentemente, enquanto que por exemplo, especificações e descrições arquiteturais são úteis e utilizadas mesmo que desatualizadas. A tabela 3.2 resume essa idéia.

Tabela 3.2 – A correlação entre a acurácia percebida
de um tipo de documento e sua freqüência de consulta

Tipo de documento	Correlação (acurácia x uso)
Testes ou Qualidade	0,67
Projeto de baixo nível	0,58
Requisitos	0,43
Arquitetural	0,41
Projeto detalhado	0,39
Especificação	0,03

Quanto à questão de quanto de documentação é necessário e com que freqüência esta deve ser atualizada, LETHBRIDGE *et al.* (2003) ressaltam que talvez o caminho não seja forçar os engenheiros de software a atualizarem a documentação de forma meticulosa, e sim, procurar tornar a documentação mais simples e poderosa. Os engenheiros de software devem procurar elaborar documentos de fácil atualização, se possível automatizada, e entender melhor os papéis dos diversos tipos de documento, de forma a

melhor incorporá-los às necessidades particulares do processo de desenvolvimento da empresa.

ODDO (2003) realizou uma pesquisa de campo com especialistas em software para identificar quais práticas, ferramentas e documentos relacionados à engenharia de software são consideradas capazes de influenciar positivamente a qualidade do processo ou do produto de software. Uma lista de documentos de software foi submetida à votação dos especialistas, e, após cálculos de fatores de influência, como por exemplo, experiência do especialista, pode-se gerar a tabela 3.3 com a votação proporcional de cada documento. Pelos critérios da pesquisa, valores inferiores a 0,5 são considerados como documentos não relevantes.

COOK e VISCONT (2002) influenciados pelo CMM (ABRAN, 2001) desenvolveram um modelo de maturidade de processo de documentação de software (DPMM). DPMM é uma descrição da maturidade do processo de documentação, capacidades e práticas que caracterizam organizações que geram documentos de alta qualidade. Os autores desenvolveram um modelo de maturidade de processo de documentação em quatro níveis e um procedimento de avaliação baseado em questionários. Assim como o CMM, cada nível do DPMM é caracterizado por áreas-chave, indicadores e objetivos que precisam ser atingidos para que uma empresa seja avaliada naquele nível. COOK e VISCONT avaliaram segundo o DPMM 91 projetos em 41 companhias, o que serviu de base para o refinamento do modelo e a conclusão de que 72% das empresas se encontram no nível 1 e o restante no nível 2.

KYLMÄKOSKI (2003) desenvolveu na *Nokia®* um método denominado *RaPiD7*, produção rápida de documentos em sete passos. Este método objetiva diminuir o tempo de produção e aumentar a qualidade dos documentos produzidos ao longo de um processo de software. Para alcançar tal objetivo, o método procura colocar o mais cedo possível o maior número de pessoas interessadas no resultado do documento envolvidas com a sua produção. Isto ocorre através de uma série de Workshops que visam recolher e analisar idéias, projetar o documento e tomar todas as decisões necessárias. A metodologia fornece um conjunto de etapas e procedimentos que devem ser seguidos para a autoria eficiente de um documento. Em alguns aspectos, O *RaPiD7* pode ser comparado a métodos ágeis ou a técnica JAD (MCCONNELL, 1996).

Tabela 3.3 – Peso de cada tipo de documento com relação à qualidade do processo/ produto de software segundo especialistas.

Documento	Peso	Documento	Peso
Especificação do sistema	0,86	Plano de controle da qualidade	0,61
Documentação no código	0,79	Acompanhamento de custos	0,58
Documentação do processo de software	0,78	Guia de instalação	0,58
Documentação de programas	0,77	Relatório de teste	0,56
Plano de testes	0,76	Contratos e acordos	0,55
Acompanhamento de prazos	0,73	Plano de contingência	0,55
Manual do usuário	0,72	Manual de treinamento	0,53
Help on-line	0,71	Histórico do projeto	0,51
Projeto do sistema	0,70	Identificação de risco	0,47
Manual do sistema	0,69	Descrição do produto para comercialização	0,32
Registro formal de revisões e testes	0,62	Documentação de marketing	0,16

3.2.1.1 Ferramentas

Embora o presente trabalho forneça o suporte computacional ao planejamento da documentação, e não à sua produção, serão citados alguns trabalhos que procuram amenizar os problemas da documentação de software discutidos anteriormente e que possuem como produto final ferramentas de apoio à produção de documentos.

CHIUEH *et all* (2000) desenvolveram VARIORUM, um sistema de documentação que provê um mecanismo simples e poderoso para anotação e compreensão de programas, baseado em hipertextos e tecnologia multimídia. O uso da tecnologia de hipertexto na organização de documentos de software é encontrado em diversos projetos de pesquisas tais como LIGHT (BARONE e ROUSSEAU, 1996) e CASE (CYBULSKI, 1992).

ANTONIOL *et all* (1997) descrevem um ambiente para compreensão e manutenção de programas denominado CANTO, que integra informação de baixa granularidade com visões arquiteturais retiradas do código fonte.

FRENCH *et all* (1997) mantêm a documentação de software como hipertextos que possuem diferentes *links* para diferentes necessidades dos usuários. Estes *links* são gerados pelo próprio sistema e mantidos atualizados mesmo com a atualização dos documentos.

TOLEMAN *et all* (2001) desenvolveram UQ*, que constitui-se de ambiente baseado em uma linguagem formal genérica para a manipulação de documentos estruturados. Este ambiente tem por objetivo capturar relações semânticas e sintáticas de um ou entre documentos e suportar a interação do usuário através de visões textuais ou diagramáticas. Esta especificação das relações interdocumentos prove, por exemplo, os requisitos básicos para implementação de ferramentas de rastreamento.

3.2.1.2 Padronização

A padronização da documentação de software poderia ajudar na redução de custos e esforço no desenvolvimento e manutenção de sistemas, aumentar a portabilidade dos mesmos e ajudar usuários e desenvolvedores a entender mais facilmente softwares existentes (PHOHA, 1997). Mas não existe um padrão de documentação de software universalmente reconhecido. Em parte isso se deve ao fato de que estilos e conteúdos de documentação diferem entre desenvolvedores e até para um mesmo desenvolvedor em circunstâncias diferentes. Em adição, linguagens de programação, metodologias de desenvolvimento, *frameworks* de processos de software e a natureza e domínio da aplicação podem ditar um estilo particular de documentação.

No entanto, existe uma padronização para documentação de software de engenharia e científico (ANSI/ANS 10.3-1995) que PHOHA (1997) sugere como ponto inicial para a documentação da maioria dos sistemas, mesmo os mais complexos. A norma ANSI/ANS 10.3-1995 não é um conjunto de especificações rígidas, e sim uma linha geral para uma documentação bem organizada e clara, seja esta documentação de objetivo externo ou interno a organização que a adota. A norma provê, de fato, um *framework* robusto e flexível para as necessidades de documentação.

DELANGHE (2000) desenvolveu um interessante trabalho quanto à aplicação de estilos de aprendizado (*Learning Styles*) na documentação de software e como o uso adequado destes podem melhorar a qualidade e legibilidade de um documento. DELAGHE se baseia na tória psicológica de KOLB (1984), conhecida como Aprendizado Experimental (*Experiential Learning*), que classifica as pessoas em quatro grandes classes de estilos de Aprendizado. As pessoas pertencentes a um determinado estilo têm sua forma própria de

escrever e somente um texto dentro do seu estilo aguça sua atenção. Desta forma, DELAGHE estuda formas de se elaborar documentos de software que atendam, dentro do possível, leitores pertencentes a qualquer estilo de aprendizado, para que o documento seja de fácil leitura para qualquer pessoa.

3.2.2 Processo de Software e de Documentação

Processo de software é o conjunto de atividades, métodos e práticas usadas na produção e evolução de software (ABRAN, 2001). De fato, o estabelecimento de um processo de software tem por objetivo aplicar uma abordagem repetitiva, disciplinada, sistemática e quantificável no desenvolvimento e manutenção de software, de forma a obter um produto de alta qualidade no custo estimado.

A idéia de olhar o processo pelo qual um produto é feito para garantir sua qualidade final não é nova em outras indústrias. Na verdade, a idéia de garantir a qualidade ao longo do desenvolvimento vem como uma evolução natural dos métodos de controle da qualidade da saída do processo (do produto final), que apenas evitavam que produtos de baixa qualidade chegassem ao cliente final, sem combater de forma eficiente a causa das falhas.

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

Um processo de software possui quatro papéis (KRUCHTEN, 1999):

(i) Guiar a ordem de atividades da equipe;

(ii) Especificar quais artefatos devem ser produzidos e quando;

(iii) Dirigir as tarefas de desenvolvedores individuais e da equipe como um todo;

(iv) Oferecer critérios de monitoração e medição dos produtos e atividades do projeto.

Um processo de software explora os seguintes conceitos (FUGGETTA, 2000):

- Tecnologia de desenvolvimento de software: suporte tecnológico usado no processo. Para realizar as atividades de desenvolvimento de software são necessárias ferramentas, infra-estrutura e ambientes. A tecnologia adequada permite a produção de software de alta complexidade de maneira financeiramente viável.

- Métodos e técnicas de desenvolvimento de software: diretrizes para uso de tecnologia e realização das atividades de desenvolvimento de software. O suporte metodológico é essencial para explorar efetivamente a tecnologia.
- Comportamento organizacional: a ciência de organizações e pessoas. Software é geralmente desenvolvido por uma equipe que deve ser coordenada e gerenciada dentro de uma estrutura organizacional.
- Mercado e economia: como qualquer outro produto, o software deve atender às necessidades reais dos clientes em uma situação de mercado específica.

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

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

Com o crescente interesse na aplicação de processos de software, surgiram diversos padrões e normas com o objetivo de homogenizar e facilitar a aplicação desta base de conhecimento ao desenvolvimento de software. Para que sejam de aplicação geral a diversos tipos de software, metodologias e ferramentas, estes processos devem ter um substancial grau de abstração, o que permite sua customização para as particularidades de cada caso. Em outras palavras, os processos devem se preocupar com o que deve existir para assegurar a qualidade pretendida, e não em de que forma isto será realizado. Na verdade, estes padrões não tratam apenas do processo de desenvolvimento em si, mas de um conjunto (*framework*) de processos interdependentes. Dentre os diversos padrões, um dos mais utilizados e a norma ISO/IEC 12207 (ISO/IEC 12207, 1997).

A norma internacional ISO/IEC 12207 – Tecnologia da Informação – Processos de Ciclo de Vida de Software (ISO/IEC 12207, 1997) tem por objetivo auxiliar os envolvidos com a produção de software na definição de seus papéis, através de processos bem definidos, e desta forma proporcionar para as organizações que a utilizam um melhor entendimento das atividades a serem executadas nas operações que envolvem, de alguma forma, o software (ROCHA *et al.*, 2001).

A norma classifica tais processos em quatro grandes grupos de acordo com sua finalidade: Processos Fundamentais, Processos de Apoio, Processos Organizacionais e

Processos de Adaptação. Cada processo é definido em termos de suas próprias atividades e cada atividade é adicionalmente definida em termos de suas tarefas. Uma organização pode selecionar um subconjunto apropriado de processos, de acordo com seus objetivos, pois a Norma foi projetada para ser adaptada a uma organização, projeto ou aplicação específica através da aplicação do processo de adaptação.

3.2.2.1 Processos Fundamentais

Os processos fundamentais constituem um conjunto de cinco processos que atendem as partes (pessoa ou organização) fundamentais durante o ciclo de vida do software. Uma parte fundamental é a que inicia ou executa o desenvolvimento, operação ou manutenção dos produtos de software. Os processos fundamentais são: Processo de Aquisição, Processo de Fornecimento, Processo de Desenvolvimento, Processo de Operação e Processo de Manutenção.

O processo de desenvolvimento define as atividades do desenvolvedor, ou seja, da organização que especifica e desenvolve o produto de software. Este processo consiste nas seguintes atividades: (i) implementação do processo; (ii) análise dos requisitos do sistema; (iii) projeto da arquitetura do sistema; (iv) análise dos requisitos do software; (v) projeto da arquitetura do software; (vi) projeto detalhado do software; (vii) codificação e testes do software; (viii) integração do software; (ix) teste de qualificação do software; (x) integração do sistema; (xi) teste de qualificação do sistema; (xii) instalação do software; (xiii) apoio à aceitação do software.

3.2.2.2 Processos de Apoio

Os processos de apoio constituem um conjunto de oito processos. Um processo de apoio auxilia um outro processo como parte integrante, com um propósito distinto, que contribui para o sucesso e qualidade do projeto de software. Um processo de apoio é empregado por outro processo, quando necessário. São eles: Processo de Documentação, Processo de Gerência de Configuração, Processo de Garantia da Qualidade, Processo de

Verificação, Processo de Validação, Processo de Revisão Conjunta, Processo de Auditoria e Processo de Resolução de Problemas.

3.2.2.3 Processos Organizacionais

Constituem um conjunto de quatro processos que são empregados por uma organização para melhorar continuamente a sua estrutura e os seus processos. São empregados tipicamente fora do domínio de projetos e contratos específicos; entretanto, ensinamentos desses projetos e contratos contribuem para a melhoria da organização. São eles: Processo de Gerência, Processo de Infra-estrutura, Processo de Melhoria e Processo de Treinamento .

3.2.2.4 O Processo de Documentação

O processo de documentação é um dos processos de apoio de ciclo de vida segundo a norma ISO/IEC 12207. Este tem por objetivo registrar informações por um processo ou atividade do ciclo de vida. O processo contém o conjunto de atividades que planeja, projeta, desenvolve, produz, edita, distribui e mantém os documentos necessários a todos os interessados, tais como gerentes, engenheiros e usuários do sistema ou produto de software.

A forma e conjunto de atividades deste processo é a base do processo de documentação definido para a Estação TABA e que será apresentado no próximo capítulo.

3.2.2.5 Processo de Adaptação

A Seção 3.1.1 discutiu a necessidade da padronização dos processos de software para cada organização e como estes podem ser especializados em processos de software específicos para o tipo de software (tecnologia e paradigma utilizado) e as características do desenvolvimento, mantendo-se os elementos básicos do processo padrão. O processo especializado, por sua vez, pode ser instanciado considerando-se as características do projeto específico, devendo-se considerar o tamanho e a complexidade do produto, as

características da equipe de desenvolvimento, a expectativa de vida útil do software e demais características do projeto.

A norma ISO/IEC 12207 reconhece esta necessidade no processo de adequação do processo a organizações e projetos específicos e possui no seu escopo o Processo de Adaptação. Esta norma é, portanto, flexível para diversas abordagens de engenharia de software, utilizável com qualquer modelo de ciclo de vida, qualquer método ou técnica de engenharia de software e qualquer linguagem de programação. Estas questões são muito dependentes do projeto e do estado da arte da tecnologia. Logo, estas escolhas são deixadas a critério dos usuários da norma (ROCHA *et al.*, 2001).

Um detalhamento maior sobre processo padrão, processos especializados e processos instanciados será realizado no capítulo V, no momento da definição do Processo de Desenvolvimento de Instrumentação Virtual.

3.3 Considerações Finais

Neste capítulo, foram apresentados os conceitos de Ambientes de Desenvolvimento de Software, Ambientes de Desenvolvimento de Software Orientados a Domínio e Ambientes de Desenvolvimento de Software Orientados a Organização. Foram descritos também a infra-estrutura, características e requisitos da Estação TABA e sua evolução para a instanciação de ambientes orientados a domínio e orientados a organização.

Os requisitos não tratados pela Estação TABA considerados importantes para a configuração de um ambiente de instrumentação virtual foram revistos e uma breve discussão bibliográfica foi realizada sobre documentação e processo de software.

Nos próximos capítulos descreve-se o trabalho realizado no âmbito desta tese, que constitui-se da definição de um processo de documentação e inserção de todo o apoio computacional ao planejamento de documentação na estação TABA, além da configuração de um ADSOrg para uma empresa de desenvolvimento de instrumentação virtual através da definição dos processos de software necessários.

CAPÍTULO IV

ADEQUAÇÕES NECESSÁRIAS À ESTAÇÃO TABA

Este capítulo apresenta as adequações necessárias à Estação TABA para torná-la apta a atender os requisitos de sistemas de instrumentação, para tanto, um processo de documentação é definido e seu apoio computacional apresentado.

Introdução

Este capítulo discute as extensões e melhorias necessárias à Estação TABA, de modo a permitir que esta seja configurada e gere um ambiente de desenvolvimento de software para organizações que desenvolvam instrumentação virtual. Isto implica na definição de um processo de documentação e na elaboração do apoio computacional ao planejamento da documentação.

A próxima seção descreve o processo de documentação que foi definido no escopo deste trabalho e que leva em consideração a norma ISO/IEC 12207 (NBR ISO/IEC 12207, 1997) e as características da Estação TABA. Documentos são produzidos ao longo de todo o processo de desenvolvimento e manutenção do sistema. Um processo para guiar quais documentos devem ser produzidos, em que momento e de que forma se torna fundamental para que o desenvolvedor saiba o que produzir e de que maneira. A seção 4.2 descreve o suporte computacional criado para apoiar o planejamento da documentação na Estação TABA.

4.1 Processo de Documentação da Estação TABA

O Processo de Documentação é responsável pelo registro das informações produzidas pelas atividades dos processos do Ciclo de Vida, planejando a documentação, definindo que documentos serão gerados, em que momento e por quem. Tal processo define roteiros para cada documento, além de orientar a produção, edição, distribuição e manutenção dos mesmos.

Na Estação TABA este processo se realiza em quatro momentos:

1. No meta-ambiente TABA, ao se introduzir conhecimento sobre processos de software. Neste momento são introduzidos roteiros de documentos que posteriormente serão adaptados as organizações e a projetos específicos. (atividade 1).
2. No meta-ambiente TABA, ao se configurar um ambiente para uma organização específica, quando os roteiros de documentos anteriormente introduzidos e disponíveis no meta-ambiente são adaptados para atender a organização para a qual o ambiente está sendo configurado. (atividade 2).
3. No ambiente instanciado (ADSOrg) a partir do ambiente configurado, ao se planejar a documentação para um projeto específico. (atividade 3 e 4).
4. No ADSOrg, durante a execução de um projeto. (atividade 5).

A seguir, há a descrição de cada uma das atividades do processo de documentação:

1. Definir Roteiros de Documentos

O objetivo desta atividade é definir e inserir roteiros de documentos no meta-ambiente da Estação TABA. Entende-se por roteiro de documento um guia de que conteúdo um documento deve conter, assim como de que forma este deve ser apresentado. Para cada atividade inserida no meta-ambiente devem ser identificados os possíveis documentos e respectivos roteiros, bem como as associações entre estes elementos. Neste momento os roteiros definidos não são específicos para organizações ou projetos. Portanto, são elaborados em um nível alto de abstração e generalidade, tornando sua aplicação e adaptação viável para qualquer empresa ou projeto.

Sub-atividades:

1.1 Elaborar Roteiros de Documentos:

Para cada atividade dos processos são identificados os possíveis artefatos produzidos, e, para cada artefato que seja um documento, deve-se projetar um roteiro (modelo). Este roteiro deve especificar o titulo, propósito, público-alvo, padrões de formatação, descrição de conteúdo, paginação, localização de figuras/tabelas, marcas de propriedade/segurança, empacotamento e outros itens de apresentação.

Produto: *roteiros de documentos do processo definidos.*

Responsável: *engenheiro de software do meta-ambiente..*

1.2 Cadastrar Roteiros de Documentos:

Os artefatos identificados em 1.1 devem ser cadastrados no meta-ambiente, inserindo-se nome, descrição e classificação e associados à sua atividade produtora. Os roteiros previamente definidos devem ser associados aos respectivos artefatos do tipo documento que estes definem.

Produto: b*ase de dados de atividades povoada com os artefatos e roteiros de documentos.*

Responsável: e*ngenheiro de software do meta-ambiente..*

2. Adaptar Roteiros para uma Organização

O objetivo desta atividade é adaptar os roteiros disponíveis na base do meta-ambiente associados às atividades dos processos de uma organização. Esta atividade também faz parte do Processo de Configuração de Ambientes. Pelo menos as seguintes características dos roteiros devem ser revistas: público-alvo, setor da empresa responsável, marcas de propriedade/segurança, inclusão e exclusão de seções do documento. Nesta atividade pode-se também excluir roteiros não aplicáveis à organização, ou então definir um novo roteiro para atividades específicas da organização não previstas no nível de abstração do meta-ambiente.

Produto: r*oteiros de documentos adaptados para a organização.*

Responsável: e*ngenheiro de software do meta-ambiente.*

3. Adaptar Roteiros para um Projeto

O objetivo desta atividade é adaptar os roteiros associados às atividades definidas para o ambiente configurado para projetos específicos. Ao se instanciar um ADSOrg podem ser feitas inclusões ou exclusões de seções dos documentos, excluir roteiros não aplicáveis ao projeto ou então definir roteiros para atividades não previstas no ambiente configurado, mas que foram acrescentadas ao ADSOrg durante a instanciação.

Produto: ADSOrg com r*oteiros adequados para o projeto.*

Responsável: e*ngenheiro de software do ambiente configurado.*

4. Elaborar Plano de Documentação para um Projeto.

O Plano de Documentação faz parte do planejamento do projeto e deve ser gerado no início do projeto. Para cada documento a ser produzido durante o desenvolvimento deve-se definir procedimentos para avaliação, responsabilidades pelas entradas, desenvolvimento, destinatários, revisão, alteração, aprovação, produção e distribuição.
Produto: p*lano de documentação do projeto.*
Responsável: gerente do projeto.

5. Produzir Documentos.

Os documentos devem ser produzidos de acordo com os roteiros definidos e fornecidos conforme o plano de documentação. Pode-se utilizar ferramentas na produção dos documentos desde que gerem os documentos segundo o roteiro. Uma vez que um documento foi produzido e aprovado, sua alteração implica na execução do processo de gerência de configuração suportado pela Estação TABA. Uma vez que um documento é aprovado, seu controle, armazenamento e segurança estão sujeitos ao processo de gerência da configuração.
Produto: d*ocumentos do projeto.*
Responsável: gerente do projeto, analistas, desenvolvedores, etc.

4.2 Apoio ao Planejamento de Documentos na Estação TABA

As próximas seções apresentam o suporte computacional implementado na Estação TABA para apoiar o processo de documentação descrito na seção anterior. Este apoio computacional constitui-se de:

(i) Uma ferramenta de planejamento de documentação *DocPlan*;

(ii) Uma infra-estrutura de gerência e armazenamento de roteiros de documentos;

(iii) Um grupo de atividades criadas e inseridas em ferramentas existentes nos diferentes níveis da Estação TABA.

Como apresentado no capítulo III, a Estação TABA apresenta diferentes níveis de abstração durante a instanciação de um ambiente, como ilustrado na figura 3.1. O processo de documentação é executado através dos diferentes níveis de abstração da Estação TABA. Isto é, ao longo dos processos de configuração e instanciação de um ADS, as atividades do processo de documentação são executadas em diferentes momentos, e com o apoio de diferentes ferramentas. A figura 4.1 ilustra esta relação entre a instanciação de um ADS e o processo de documentação.

Desta forma, foram necessárias alterações nos processos de ferramentas existentes na Estação TABA para acomodar as atividades do processo de documentação.

No nível do meta-ambiente, se tornou necessária a expansão do editor de conhecimentos para abranger a edição de roteiros e a associação deste com os artefatos que representam o documento que o mesmo define. Também foi necessária a inserção de novas atividades na ferramenta *Config* (VILLELA, 2004), responsável pela configuração de ambientes para uma organização.

No ambiente configurado, houve a necessidade da alteração do processo de instanciação de ambientes apoiado pela ferramenta *AdaptPro* (BERGER, 2003), a partir da inserção de novas atividades responsáveis pela particularização dos roteiros para as características da organização.

No ambiente instanciado (ADSOrg), uma nova ferramenta denominada *DocPlan* foi criada para dar apoio ao planejamento da documentação do projeto. Também foram necessárias alterações no controle e na produção de artefatos para apoiar a inicialização de documentos a partir de seus roteiros.

Nas próximas seções discute-se estas alterações em detalhes para cada um dos níveis de abstração da Estação TABA: meta-ambiente, ambiente configurado e ADSOrg.

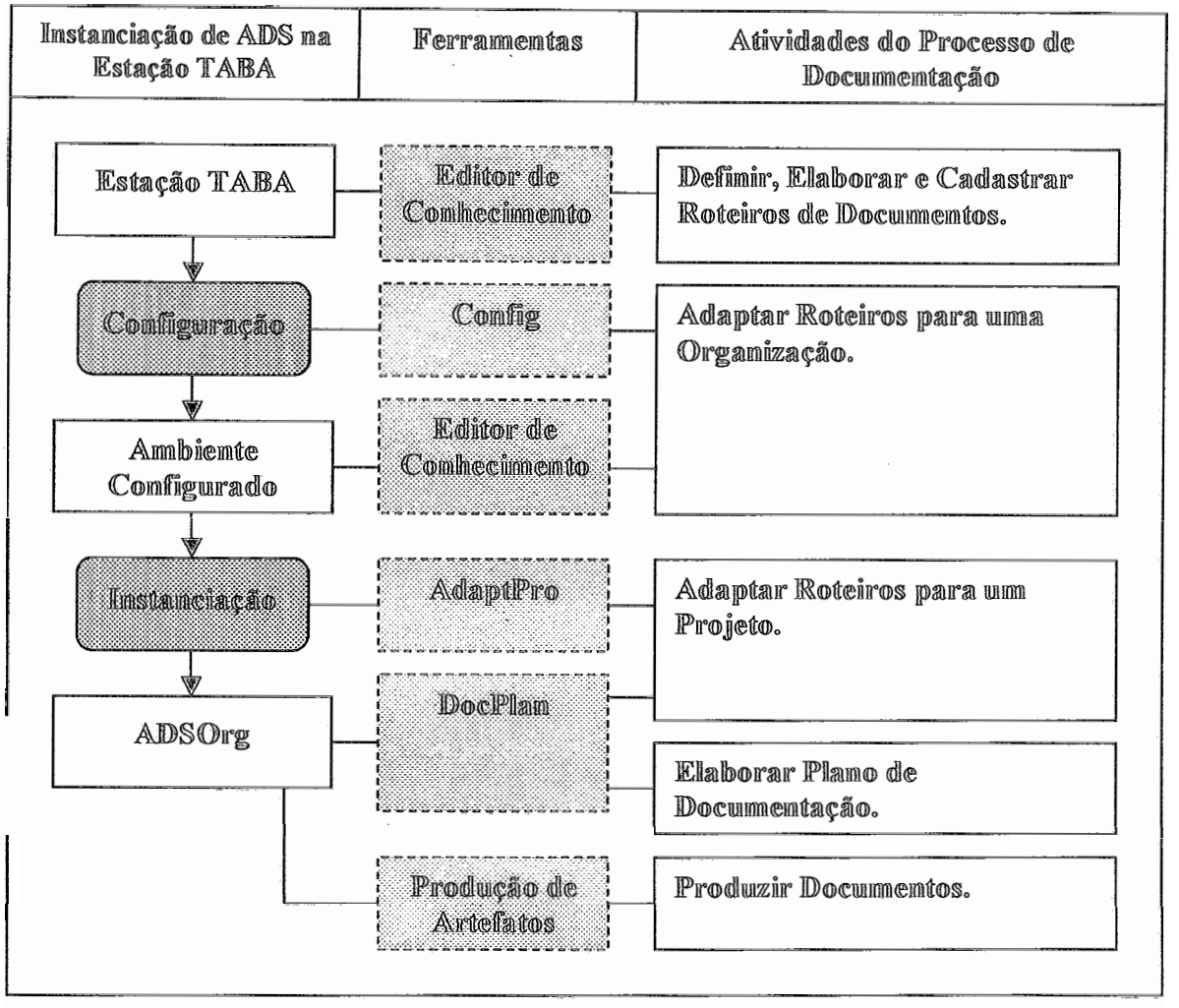


Figura 4.1– Processo de Documentação na instanciação de um ADS

4.2.1 Alterações no Meta-Ambiemte da Estação TABA

- Conhecimento da Estação TABA sobre artefatos de software

O meta-ambiente da Estação TABA possui, no que se refere a processos de software, conhecimento sobre as atividades comumente realizadas em projetos de software. Estas atividades são baseadas na norma ISO/IEC 12207 (NBR ISO/IEC 12207, 1997), em modelos de maturidades como o SPICE (ISO/IEC TR 15504, 1998), o CMM (ABRAN, 2001), o CMMI (CHRISSIS *et al.*, 2003) e na experiência dos pesquisadores da área engenharia de software da COPPE/UFRJ. A figura 4.2 ilustra o conhecimento sobre atividades no meta-ambiente TABA.

Cada atividade possui, associada a ela, uma lista dos artefatos de software que pode produzir. A figura 4.3 ilustra como essa associação é realizada no editor de conhecimentos de processo de software do meta-ambiente da Estação TABA.

Através do Editor de Conhecimento a atividade 1: *Definir Roteiros de Documentos* do Processo de Documentação definido, será realizada.

Entende-se por artefato produtos de software produzidos ou consumidos por atividades durante a sua realização. São exemplos de artefatos: manuais de qualidade, diagramas de fluxos de dados, diagramas de objetos, código fonte, etc. Algumas classificações presentes para um artefato na Estação TABA são, por exemplo, componente de software, componente de hardware, equipamento e documento. A edição do item de conhecimento artefato e a forma de associação com roteiros apoiada pela Estação TABA é ilustrada na figura 4.4.

Os artefatos do tipo documento são os que interessam ao processo de documentação. Estes possuem a possibilidade de associação com um ou mais roteiros que definem a forma e o conteúdo do documento. Vale ressaltar que um mesmo artefato, de mesma definição geral, em diferentes contextos pode apresentar diferentes formatos e conteúdos. Por exemplo, o artefato *Especificação de Requisitos* produzido pela atividade *Especificar Requisitos de Software* possui diferentes roteiros associados em virtude do paradigma de desenvolvimento adotado, como por exemplo, um roteiro para uma especificação de requisitos orientada a objetos e outro para uma especificação estruturada. Em um nível maior de reutilização do conhecimento adquirido pela organização, pode até mesmo já existir cadastrado no meta-ambiente roteiros que levam em consideração características específicas de classes de projeto. Estes seriam selecionados no momento da instanciação. Por exemplo, para projetos pequenos não seria necessário um grande formalismo, já podendo existir roteiros com seções suprimidas e níveis de exigência menores.

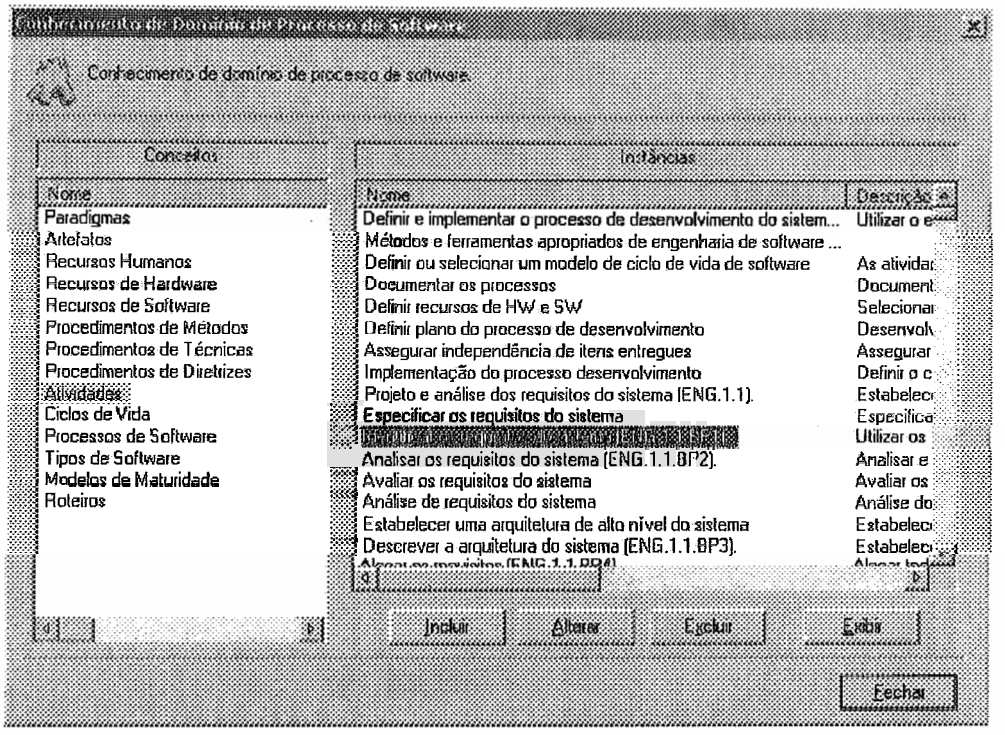


Figura 4 .2 – Conhecimento sobre atividades no meta-ambiente TABA

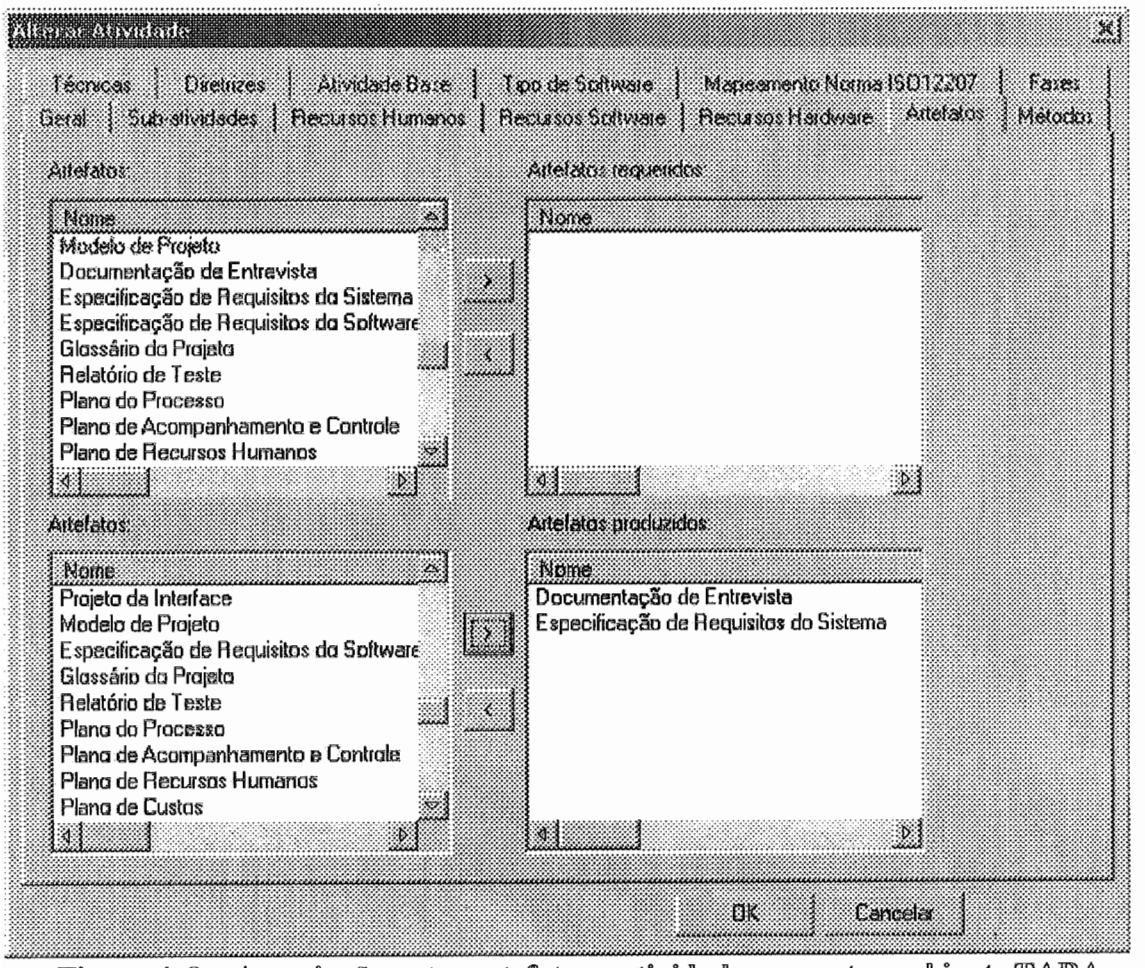


Figura 4 .3 – Associação entre artefatos e atividades no meta-ambiente TABA

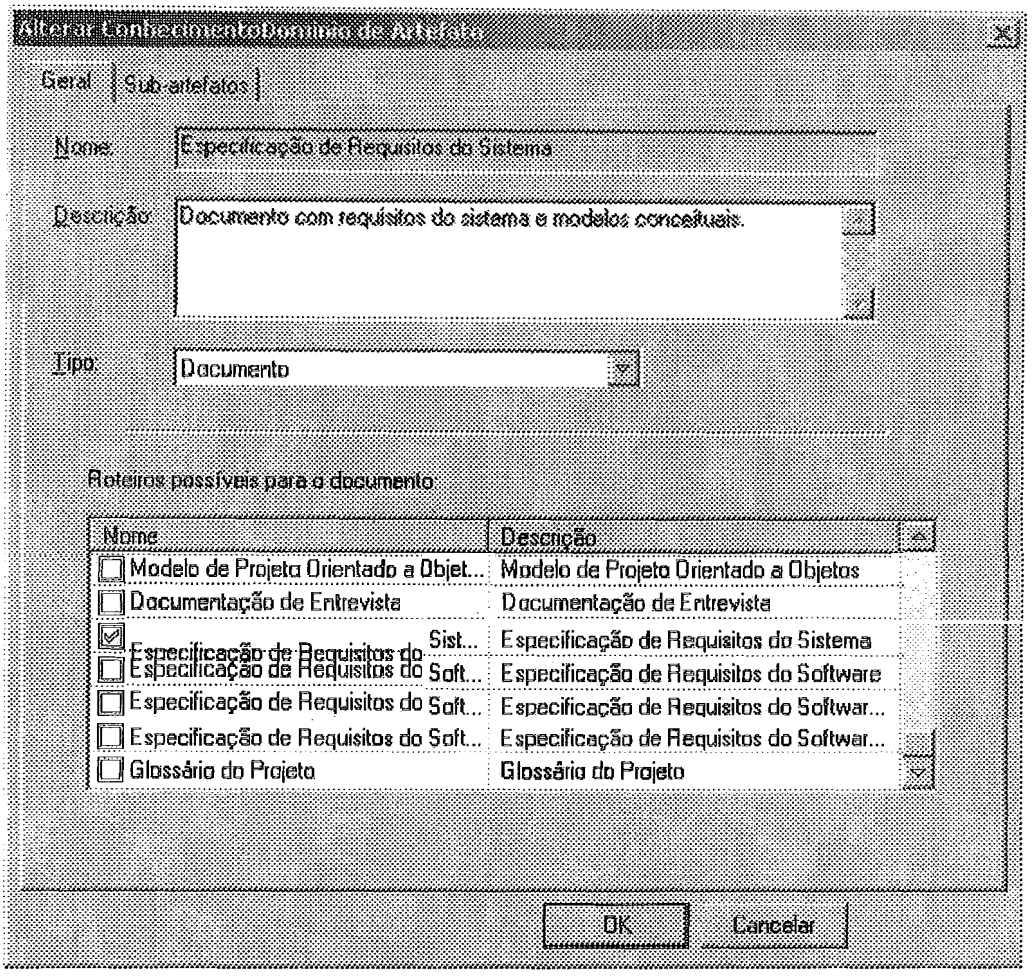


Figura 4.4 – Edição de artefato no meta-ambiente TABA

A figura 4.5 ilustra como um item de conhecimento do tipo roteiro é editado na Estação TABA. Detalhando esta funcionalidade, cada roteiro possui um arquivo externo associado onde é armazenado o conteúdo do roteiro para facilitar a manipulação do mesmo. Este arquivo pode estar em qualquer padrão, como exemplo, HTML, arquivo texto ASCII (TXT) ou Rich Text Format (RTF). Os roteiros originais da Estação TABA estão salvos no formato Modelo de Documento do Microsoft Word (.dot). Estes arquivos externos são transparentes ao usuário, tendo sua gerência e armazenamento executados pela Estação TABA. A edição dos roteiros se dá através de um editor externo acionado automaticamente no momento em que o usuário clica no botão "Editar".

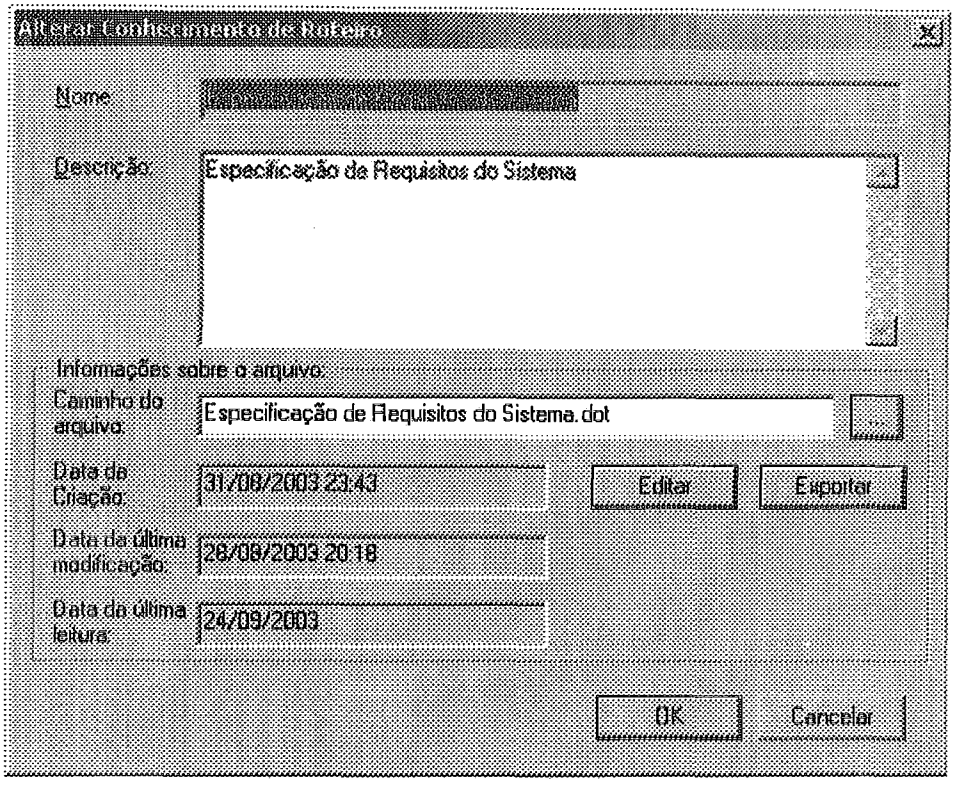


Figura 4.5 – Edição de roteiro no meta-ambiente TABA

- Planejamento de documentação durante a configuração de um ambiente

A ferramenta *Config* reside no meta-ambiente da Estação TABA e tem como objetivo configurar um ambiente para uma organização considerando as suas características (VILLELA, 2004). *Config* fornece apoio automatizado ao processo de configuração de ambientes e, entre suas atribuições, é responsável pela definição do processo padrão e especializados de uma organização, conforme apresentado no capítulo anterior. Um maior detalhamento sobre hierarquia de processos é realizado no próximo capítulo, no momento em que define-se o processo de desenvolvimento de instrumentação virtual.

O processo de configuração de ambientes foi expandido para atender as atividades contidas no Processo de Documentação definido neste trabalho, suportando computacionalmente a atividade 2: *Adaptar Roteiros para uma Organização*. A tabela 4.1 ilustra as atividades originais da ferramenta *Config* e, em destaque, encontram-se as novas atividades oriundas do Processo de Documentação. Maiores detalhes sobre cada atividade do Processo de Configuração de Ambientes podem se encontrados em VILLELA (2004).

Tabela 4.1 – Atividades do Processo de Configuração de Ambientes

Nome da Atividade	Descrição e Nome das Sub-atividades
1. Contextualizar Configuração	Contextualizar a configuração, indicando as características gerais da organização, a sua cultura na área de software e quais são os objetivos almejados com a configuração, de forma a fornecer os subsídios necessários à elaboração da proposta de configuração
	Sub-atividades:
	1. Realizar Entrevistas;
	2. Caracterizar a Organização;
	3. Identificar a Cultura Organizacional na Área de Software;
	4. Identificar Objetivos para a Configuração do Ambiente;
2. Elaborar Proposta de Configuração do Ambiente	Elaborar a proposta de configuração de um ambiente para a organização, de acordo com as informações previamente obtidas, o que implica em especificar o conteúdo e as ferramentas que serão disponibilizadas no Ambiente Configurado e em elaborar planos de organização e custos, além de cronograma para a configuração.
	Sub-atividades:
	1. Identificar Teorias de Domínio;
	2. Identificar Processos;
	3. Identificar Ferramentas;
	4. Definir Equipes e Responsabilidades;
	5. Definir Cronograma;
	6. Definir Custos;
	7. Enviar Proposta;
3. Registrar Parecer sobre a Proposta	objetivo registrar o parecer dos Representantes da Organização sobre a proposta de configuração, indicando a sua aprovação, a necessidade de modificações e ajustes, ou a não aprovação da proposta.
4. Definir Processo Padrão	Definir o processo padrão da organização para desenvolvimento de software.
	Sub-atividades:
	1. Caracterizar Processo Padrão;
	2. Incluir Atividades do Processo Padrão Base (quando pertinente);
	3. Incluir Atividades ISO/IEC 12207;
	4. Incluir Atividades do Tipo de Software;
	5. Incluir Atividades Próprias da Organização;
	6. Incluir Atividades Orientadas a Domínio;
	7. Determinar Atividades Obrigatórias;
	8. Adaptar Roteiros de Documentos para a Organização;

5. Definir Processo Especializado	Definir, a partir do processo padrão, processos especializados para os paradigmas de desenvolvimento utilizados na organização e, opcionalmente, para diferentes tipos de software.
	Sub-atividades:
	1. Especializar Atividades do Processo Padrão;
	2. Incluir Atividades do Tipo de Software;
	3. Determinar Atividades Obrigatórias;
	4. Especializar Roteiros de Documentos;
6. Definir Teoria do Domínio	Definir uma Teoria do Domínio para um determinado domínio de conhecimento.
	Sub-atividades:
	1. Revisar Propósito;
	2. Especificar Requisitos;
	3. Capturar Teoria do Domínio;
	4. Conceituar Teoria do Domínio;
	5. Formalizar Teoria do Domínio;
	6. Identificar Tarefas;
	7. Avaliar Teoria do Domínio;
	8. Implementar Teoria do Domínio;
7. Descrever Tarefa	Descrever uma tarefa genérica que tenha sido identificada como pertinente ao contexto de uma Subteoria do Domínio.
	Sub-atividades:
	1. Descrever Tarefa em Linguagem Natural;
	2. Definir Estratégia para Solução da Tarefa;
	3. Capturar Conceitos e Relações da Tarefa;
	4. Descrever Fluxo de Controle da Tarefa;
	5. Formalizar Descrição da Tarefa;
	6. Avaliar Descrição da Tarefa;
	7. Implementar Descrição da Tarefa;
8. Incluir Novas Ferramentas nas Descrições dos Processos	Ferramentas que tenham sido construídas após a definição dos processos padrão e especializados precisam ser incluídas na descrição das atividades que as utilizam, para que, no momento da configuração, possam ser integradas ao Ambiente Configurado.
9. Criar Ambiente Configurado	Gerar o Ambiente Configurado para a organização a partir da seleção do processo padrão e das Teorias do Domínio que definem o ambiente.
	Sub-atividades:
	1. Definir Ambiente Configurado;
	2. Gerar Ambiente Configurado;
	3. Testar Ambiente Configurado;

Adaptar Roteiros para o Processo Padrão da Organização

Uma das atividades do processo de configuração é *Definir Processo*, na qual, ao seu final, deve ser possível adaptar os roteiros previamente definidos e disponíveis no meta-ambiente TABA para as características da organização, conforme a atividade 2, *Adaptar Roteiros para uma Organização*, do Processo de Documentação. Para isso, a atividade *Adaptar Roteiros de Documentos para a Organização* foi adicionada ao final das sub-atividades que compõem *Definir Processo Padrão*. Exemplos desta particularização ou adaptação de roteiros para uma organização são: inclusão de logomarcas e marcas de propriedade/segurança, inclusão de cabeçalhos de identificação de documentos, retirada, inclusão ou alteração de seções específicas para procedimentos próprios da organização e alteração de outros itens de apresentação. A tela da ferramenta *Config* referente a esta atividade é ilustrada na figura 4.6.

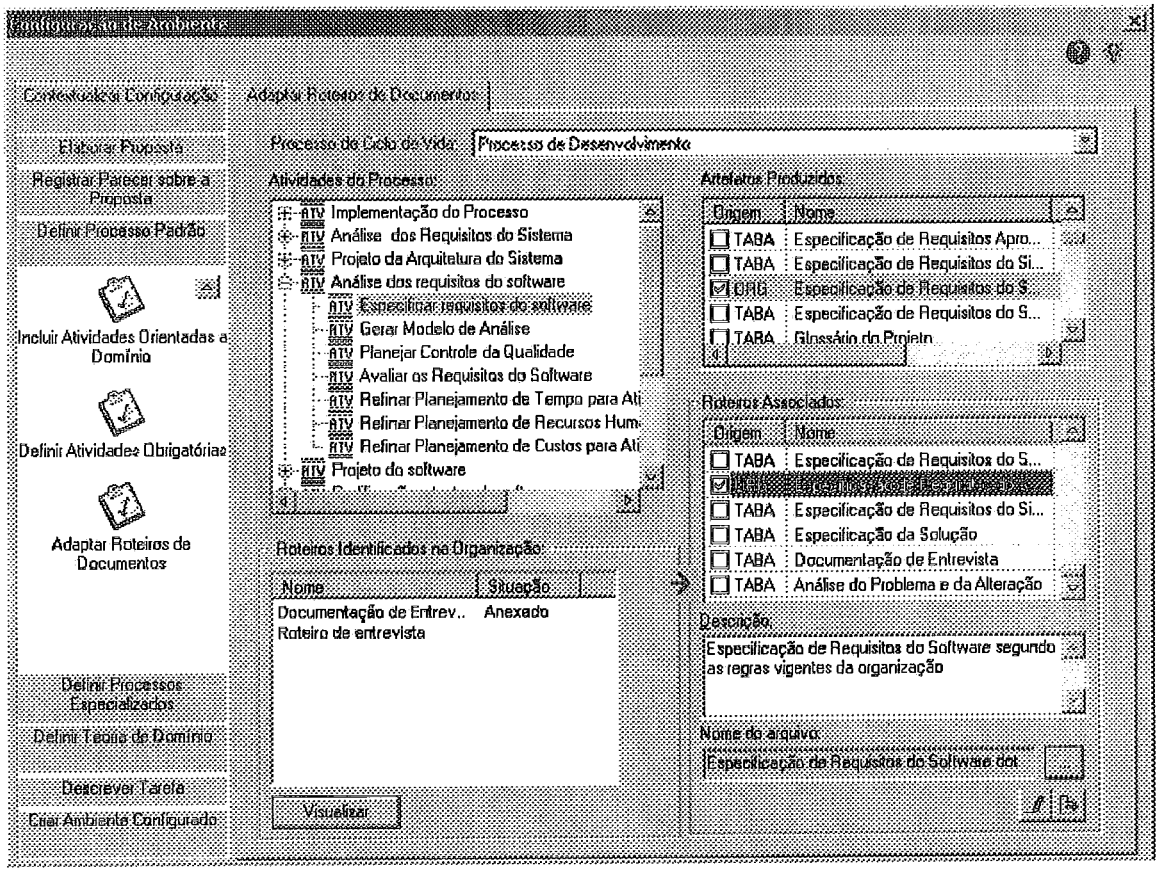


Figura 4.6 - Config – Adaptar Roteiro de Documentos para a Organização

Desta forma, através da ferramenta *Config*, o Engenheiro de Software responsável pela configuração do ambiente para a organização, adapta os roteiros de interesse. Selecionando-se um nó na árvore que representa as atividades do processo à esquerda da tela, os artefatos do tipo documento produzidos pela atividade são selecionados automaticamente na lista ao lado. É possível alterar a descrição do artefato ou seu nome, associar novos artefatos à atividade ou remover outras associações.

Selecionando-se um artefato, todos os roteiros associados ao mesmo são selecionados automaticamente na lista de roteiros disponíveis no meta-ambiente exibida no canto inferior direito da tela, sendo então possível incluir ou retirar associações entre roteiros e artefatos. Selecionando-se um roteiro, pode-se alterar seu nome, descrição e arquivo associado. A interface permite editar ou substituir o arquivo externo relacionado ao roteiro. A Estação TABA gerencia estes arquivos de forma transparente, copiando para locais de armazenamento interno. A qualquer momento é possível inserir novos roteiros e artefatos não existentes na Estação TABA.

Uma vez que um artefato é modificado (seu nome ou descrição), a sua definição alterada passa a existir somente naquele processo, desvinculando-se do conhecimento original da Estação TABA do qual foi derivado. Ou seja, as alterações realizadas em um artefato não repercutem no conhecimento de processo de software existente na Estação TABA, bem como em qualquer outro processo previamente definido. O mesmo ocorre para as alterações realizadas em um roteiro (modificações em seu nome, descrição ou conteúdo de arquivo externo). Vale ressaltar que quando um artefato é modificado, suas associações com roteiros continuam existindo e referem-se às definições originais da Estação TABA. Somente quando um roteiro é modificado esse passará a existir apenas no processo sob edição, e neste caso, o artefato associado ao mesmo também passará a existir apenas no processo de forma automática. Isto é necessário porque o artefato é o elo de ligação entre o roteiro e a atividade.

Quando um dos fatores discutidos acima ocorre, levando a definição de um artefato ou roteiro a existir somente no processo sendo definido, a coluna "Origem", existente nas listas de artefatos e roteiros, passa a exibir o valor "ORG" em substituição a "TABA". Isto fornece uma forma imediata de se saber quais artefatos e roteiros já sofreram algum tipo de

modificação ou quais ainda se referem à definição original da Estação TABA, que é caracterizada pelo valor "TABA" na coluna "Origem".

Adaptar Roteiros para o Processo Especializado da Organização

Durante a especialização do processo padrão novas atividades podem ser incluídas e estas podem possuir artefatos não existentes no processo padrão. Desta forma, a atividade de *Especializar Roteiros de Documentos* foi inserida como uma das sub-atividades que compõem *Definir Processo Especializado* com o objetivo de particularizar novos roteiros ou então existentes, mas que só agora foram inseridos ao processo. Na verdade, esta atividade é a continuação da atividade 2: *Adaptar Roteiros para uma Organização*, do Processo de Documentação. A figura 4.7 apresenta a tela responsável por *Especializar Roteiros de Documentos* para o processo sendo especializado.

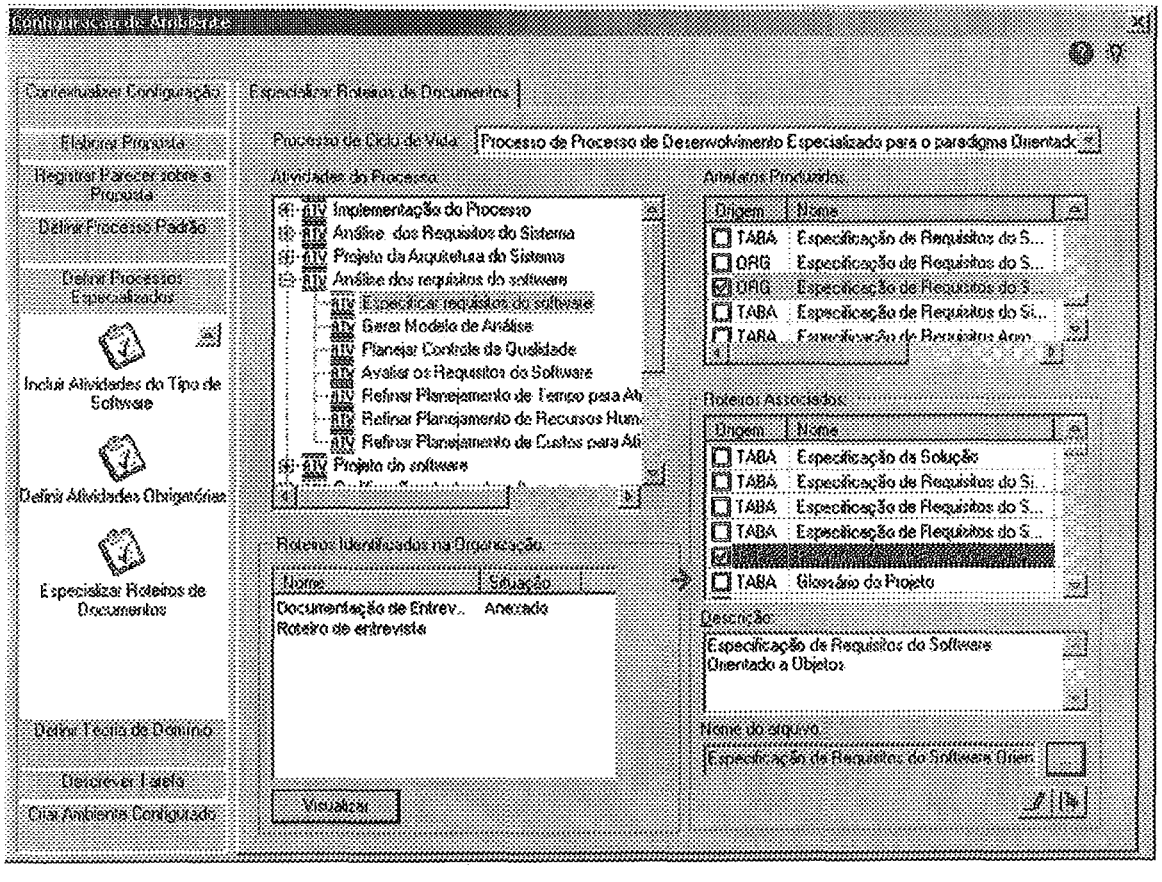


Figura 4.7 - Config – Especializar Roteiros de Documentos

Esta atividade tem o mesmo objetivo e, portanto, as mesmas funcionalidades da adaptação de roteiros para o processo padrão, já descritas acima. A única mudança está no foco da aplicação desta funcionalidade, que antes era na adequação dos artefatos de todo o processo, e agora trata principalmente das atividades exclusivas do processo especializado em questão.

4.2.2 Alterações no Ambiente Configurado da Estação TABA

- **Planejamento de documentação durante a instanciação de um ambiente**

O outro momento em que um roteiro necessita ter suas características adaptadas é durante a instanciação de um ambiente de desenvolvimento de software (ADSOrg) para um projeto específico, conforme atividade 3: *Adaptar Roteiros para um Projeto,* do Processo de Documentação. O processo de instanciação de ambientes é apoiado pela ferramenta *AdaptPro* (BERGER, 2003), existente no ambiente configurado para uma organização. O ambiente configurado pode ser interpretado como um gerador de ambientes de desenvolvimento de software adequados às necessidades da organização e de seus projetos. A ferramenta *AdaptPro* apóia o planejamento do processo, orientando na adequação de um processo especializado às características do projeto, a seleção do modelo de ciclo de vida, a geração de um ambiente (ADSOrg) e demais atividades necessárias.

O Processo de Instanciação de Ambientes foi expandido para atender as atividades contidas no Processo de Documentação definido neste trabalho, suportando computacionalmente a atividade 3: *Adaptar Roteiros para um Projeto*. As atividades do Processo de Instanciação de Ambientes estão descritas na tabela 4.2, na qual, em negrito sublinhado encontram-se as atividades que foram acrescidas para dar suporte ao Processo de Documentação. Maiores detalhes sobre cada atividade do Processo de Instanciação de Ambientes podem se encontrados em BERGER (2003).

Tabela 4.2 – Atividades do Processo de Instanciação de Ambientes

Nome da Atividade	Descrição e Nome das Sub-atividades
1. Caracterizar Projeto	Definir as características do projeto e do software a ser desenvolvido
	Sub-atividades:
	1. Definir Projeto
	2. Identificar Características do Projeto
2. Planejar Processo	Planejar o processo de desenvolvimento, suas atividades, modelo de ciclo de vida e iterações.
	Sub-atividades:
	1. Incluir Atividades do Tipo de Software
	2. Adaptar Roteiros
	3. Definir Modelo de Ciclo de Vida
	4. Realizar Mapeamento
	5. Incluir/Excluir Atividades
	6. Revisar Roteiros
	7. Selecionar Técnicas
	8. Selecionar Ferramentas
	9. Visualizar Processo
3. Instanciar ADSOrg	Gerar um ambiente de desenvolvimento de software para um novo projeto.

Adaptar Roteiros do Processo Especializado para o Projeto

A atividade *Adaptar Roteiros* foi incluída como uma das sub-atividades de *Planejamento do Processo,* permitindo que o usuário realize a adequação dos roteiros às necessidades do projeto sendo instanciado, conforme a atividade 3, *Adaptar Roteiros para um Projeto,* do Processo de Documentação. Esta sub-atividade foi inserida no momento em que o Processo Especializado está pronto para ser mapeado no modelo de ciclo de vida. Características do projeto tais como: tamanho, duração, nível de formalismo de documentação e nível de danos em falha do software, podem influenciar no conteúdo dos roteiros de documentos. A tela para adaptação de roteiros do *AdaptPro* é ilustrada na figura 4.8.

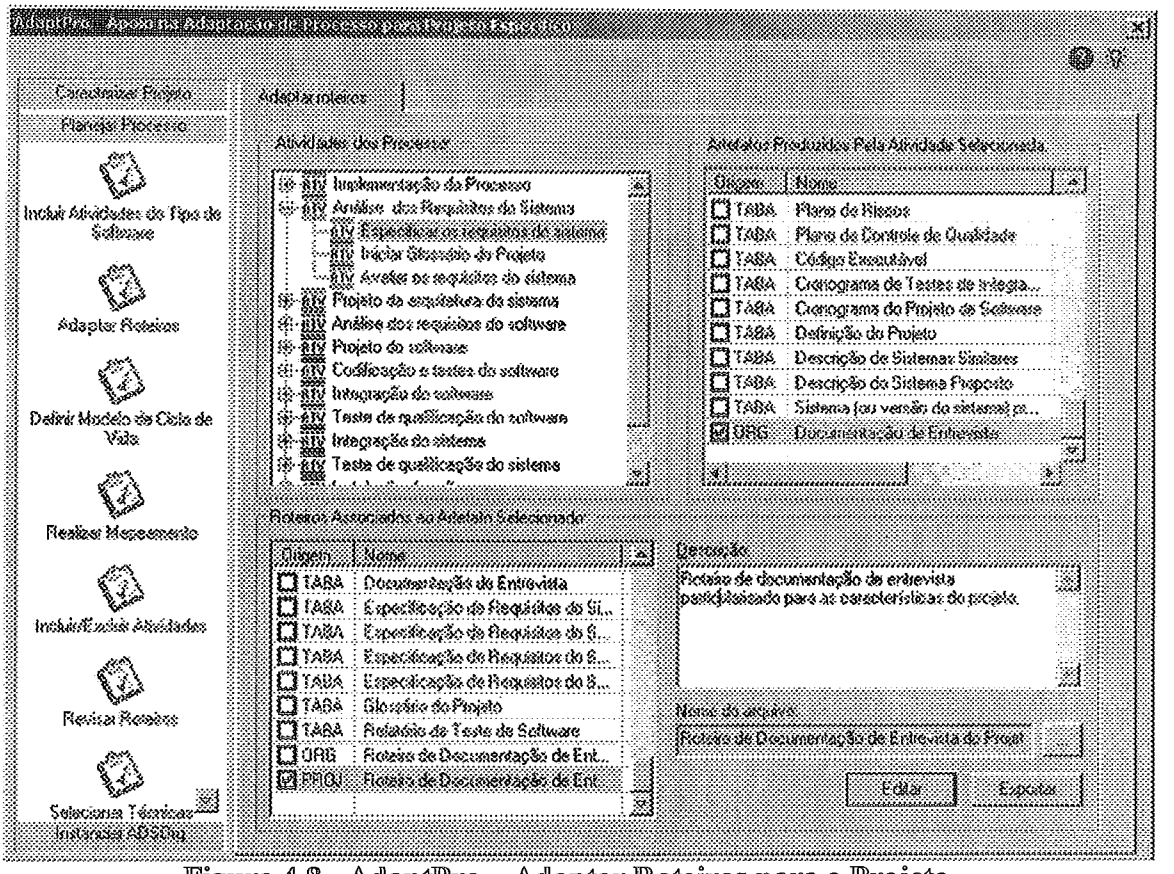


Figura 4.8 - AdaptPro – Adaptar Roteiros para o Projeto

As funcionalidades acessíveis ao usuário são as mesmas já descritas na seção anterior para a atividade de *Adaptar Roteiros*. A única diferença reside no fato de que se um roteiro ou artefato é modificado, seu conceito passa não mais a ser associado exclusivamente à organização, como ocorre na ferramenta Config, e sim ao projeto. Isto é, qualquer artefato ou roteiro uma vez modificado, seja oriundo da Estação TABA ou da organização, passa a ser identificado com o rótulo "PROJ" no campo origem, indicando sua exclusividade ao projeto sendo instanciado.

Adaptar Roteiros do Processo Instanciado para o Projeto

A atividade *Revisar Roteiros* foi incluída como uma das sub-atividades de *Planejamento do Processo,* permitindo que o usuário continue a adaptação/particularização dos roteiros às necessidades do projeto sendo instanciado, conforme atividade 3, *Adaptar Roteiros para um Projeto,* do Processo de Documentação.

A atividade *Revisar Roteiros* foi inserida após a definição do modelo de ciclo de vida e do mapeamento necessário para adequação do processo ao modelo selecionado. Neste momento, atividades são repetidas ao longo de diferentes fases, iterações ou ciclos de desenvolvimento, bem como novas atividades podem ser inseridas ou modificadas. Neste contexto, um usuário pode alterar os artefatos ou roteiros associados com atividades novas ou modificadas, bem como particularizar os artefatos ou roteiros para atividades agora residentes em diferentes ciclos, fases ou iterações. Isto é, o usuário pode, por exemplo, determinar que o documento de *Especificação de Requisitos* do primeiro ciclo de desenvolvimento seja diferente do segundo, possuindo seções diferentes, a partir da modificação do roteiro associado ao artefato *Especificação de Requisitos* produzido pela atividade *Especificar Requisitos de Software* contida no segundo ciclo de desenvolvimento.

Caso nenhum roteiro ou artefato seja modificado, as atividades repetidas nas diferentes fases, ciclos e iterações referem-se ao mesmo artefato produzido. A figura 4.9 ilustra a tela associada à Revisão de Roteiros na ferramenta *AdaptPro*.

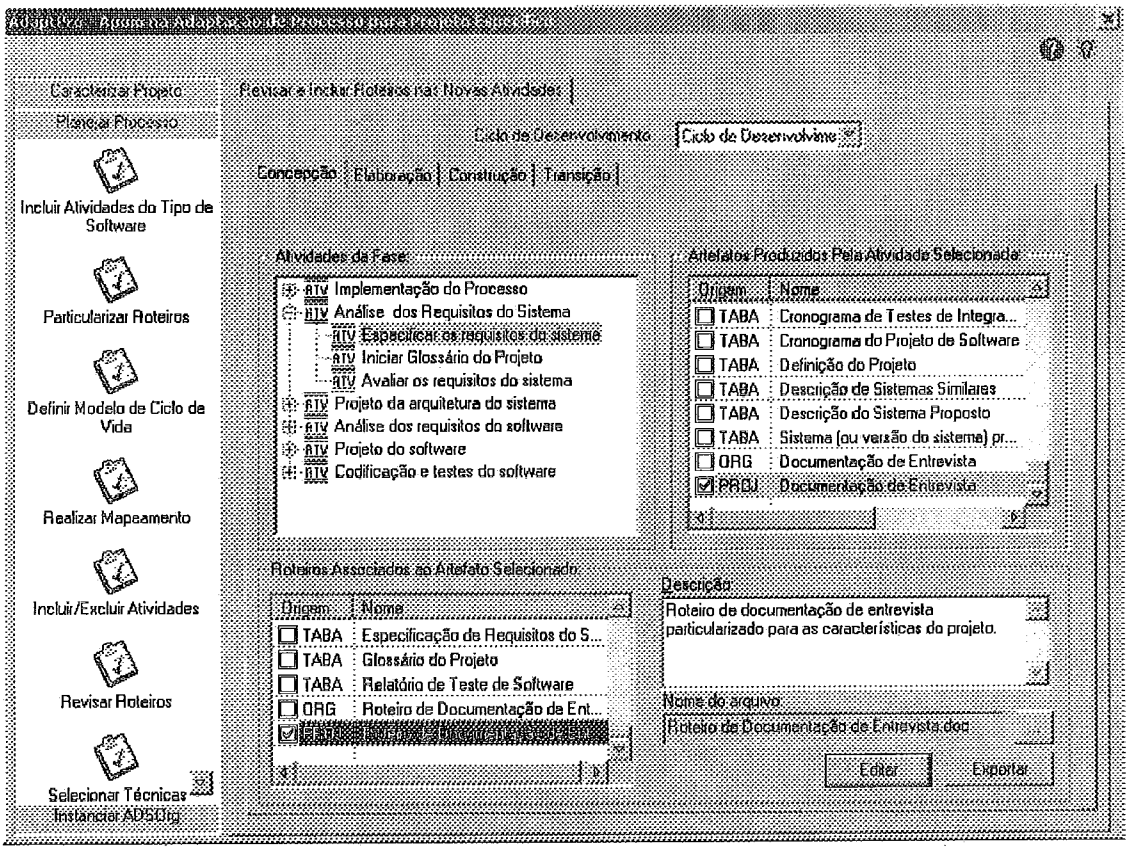


Figura 4.9 – AdaptPro – Revisão de Roteiros Adaptados

As funcionalidades são as mesmas descritas sobre as telas anteriores, somente sendo a apresentação modificada para representar a organização das atividades ao longo do modelo de ciclo de vida selecionado. No exemplo ilustrado pela figura 4.9, as atividades estão dispostas segundo o modelo de ciclo de vida RUP – Rational Unified Process (KRUCHTEN, 1999).

Ao final da execução das atividades referentes a *Adaptar Roteiros para um Projeto*, deverá existir, no máximo, um roteiro associado a cada artefato do tipo documento produzido ao longo do processo, com o objetivo de evitar conflitos na produção dos documentos.

4.2.3 Alterações no Ambiente Instanciado da Estação TABA (ADSOrg)

O ambiente instanciado da Estação TABA (ADSOrg) é um ambiente de desenvolvimento de software (ADS) potencializado pelo conhecimento da organização em desenvolvimento de software e dos domínios para os quais desenvolve. No contexto do Processo de Documentação definido, será no ADSOrg que serão realizadas as atividades 4 e 5: *Elaborar Plano de Documentação para um Projeto*, através da ferramenta *DocPlan*, e *Produzir Documentos*.

A ferramenta DocPlan

A ferramenta *DocPlan* é responsável pelo apoio ao *Processo de Planejamento da Documentação em Projetos de Software*, portanto, responsável pela execução da atividade 4: *Elaborar Plano de Documentação para um Projeto*, do Processo de Documentação definido. A atividade *Planejar Documentação* é sub-atividade obrigatória de *Implementação do Processo,* atividade pertencente ao Processo de Desenvolvimento de Software segundo a norma ISO/IEC 12207 (NBR ISO/IEC 12207, 1997), cujas atividades servem de base para os processos definidos na Estação TABA. Desta forma, qualquer processo de desenvolvimento definido na Estação TABA contará com a atividade *Planejar Documentação*, que possui a associação com a *ferramenta DocPlan*. O acesso à ferramenta é ilustrado através da figura 4.10, que apresenta a interface básica de um ADSOrg.

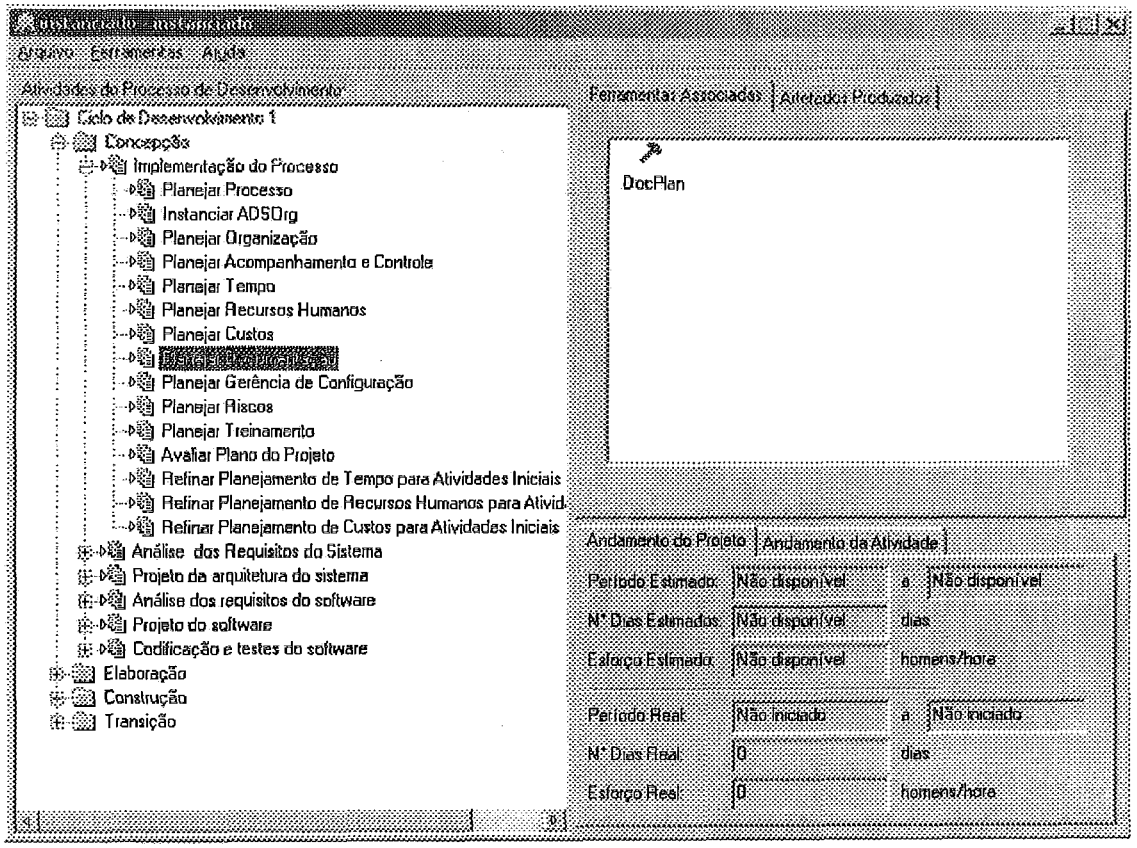


Figura 4.10 – ADSOrg – Acesso ao DocPlan

A interface básica da ferramenta *DocPlan* é ilustrada na figura 4.11. No lado esquerdo pode-se identificar o processo guia de utilização da ferramenta e no lado direito a tela da atividade sendo executada pelo usuário. A estrutura desta interface é a mesma em todas as ferramentas de apoio ao planejamento do projeto do ADSOrg, facilitando o treinamento nas mesmas. Os ícones localizados abaixo da barra de título permitem a busca e o registro de conhecimento e são acessados através de interfaces com a ferramenta de aquisição de conhecimento *Acknowledge* (MONTONI, 2003).

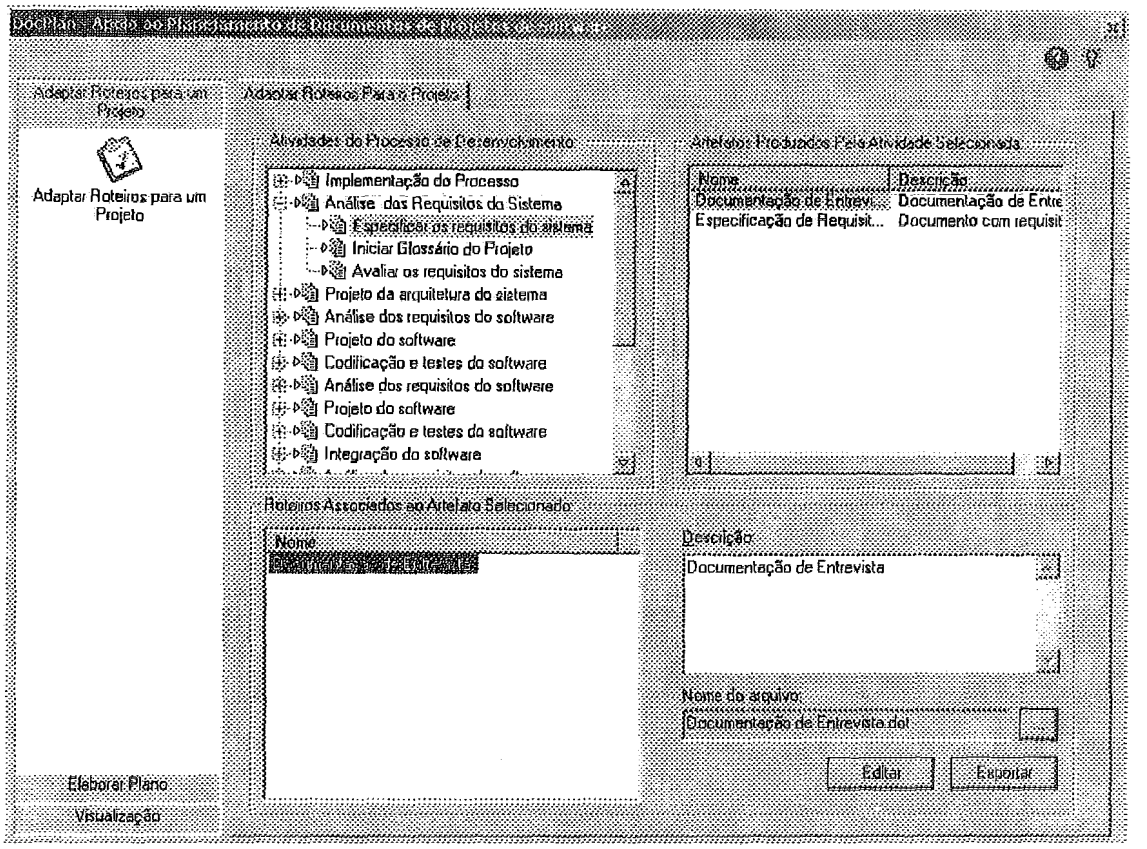


Figura 4.11 – DocPlan – Adaptar Roteiros para Projeto

Selecionando o botão *Registrar Conhecimento,* o usuário pode registrar o conhecimento adquirido relacionado à atividade sendo executada. A figura 4.12 ilustra o registro de conhecimento.

De maneira semelhante, o usuário pode consultar o conhecimento sobre adaptação de roteiros adquirido em projetos anteriores e armazenado no repositório da organização selecionando o botão *Consultar Conhecimento.* A figura 4.13 ilustra a tela de consulta de conhecimento.

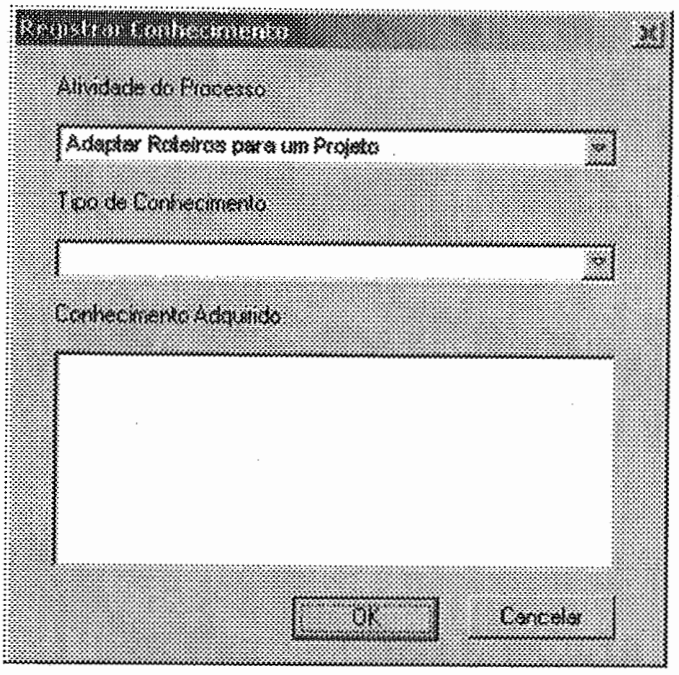


Figura 4.12 – Interface com a ferramenta *Acknowledge*: Registrar Conhecimento

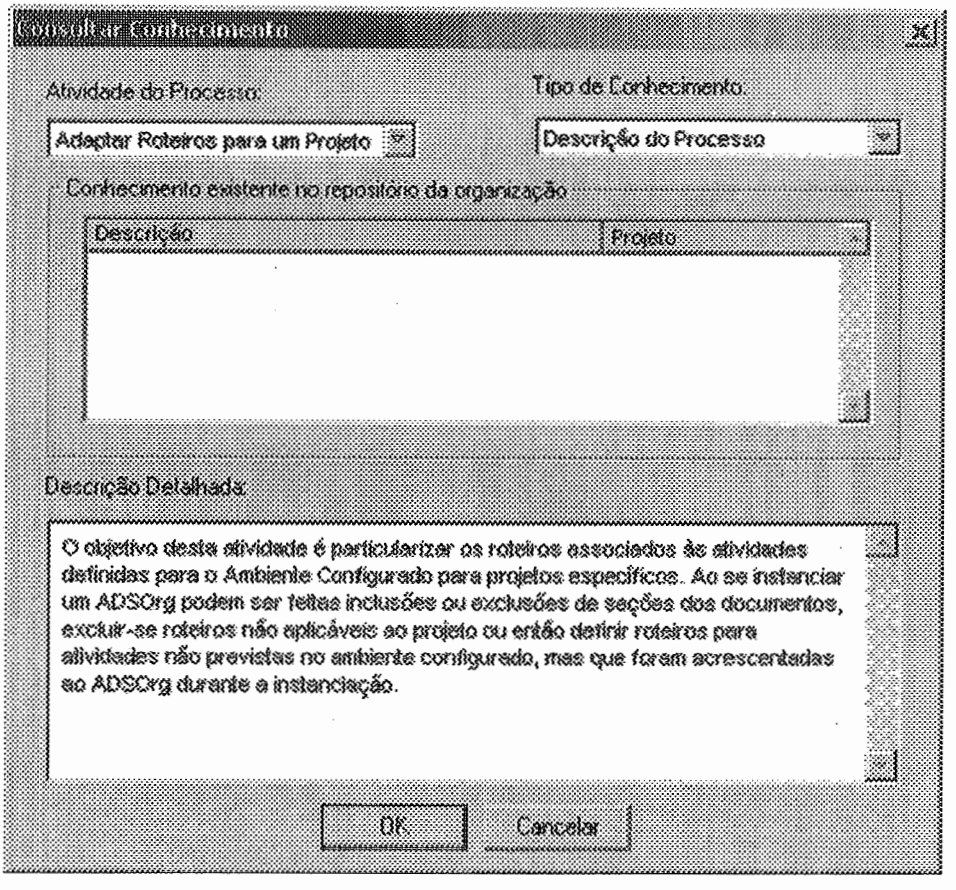


Figura 4.13 – Interface com a ferramenta *Acknowledge*: Consultar Conhecimento

Adaptar Roteiros para o Projeto

A atividade *Adaptar Roteiros para um Projeto* permite a adequação dos roteiros e artefatos do processo de desenvolvimento às peculiaridades do projeto em andamento e é ilustrada na figura 4.11.

Embora esta atividade já tenha sido realizada durante o planejamento do processo através da ferramenta *AdaptPro* BERGER (2003), esta mesma atividade, inserida neste novo contexto, oferece ao usuário a possibilidade de adequar roteiros e artefatos a novas necessidades levantadas durante o detalhamento e planejamento do processo. Suas funcionalidades são as mesmas já detalhadas na atividade similar existente na ferramenta *AdaptPro*.

Elaborar o Plano do Projeto

A atividade *Elaborar Plano* é sub-dividida em três atividades que consistem em definir o público-alvo, responsáveis pela produção e aprovação de cada artefato.

A figura 4.14 ilustra a interface para definição do público-alvo de um artefato.

Selecionando-se, à esquerda, uma atividade na árvore representativa do processo, à direita, são exibidos os artefatos produzidos pela atividade. Uma vez selecionado um artefato, abaixo são exibidas as categorias de profissionais associadas como passíveis de interesse pelo documento. As categorias de profissionais exibidas como passíveis de seleção são, na verdade, todos os papéis estimados como necessários ou efetivamente alocados para execução de qualquer atividade do processo e não apenas da atividade selecionada, pois o público-alvo de um dado documento pode englobar papeis que não necessariamente participam de sua produção.

Entende-se por papel uma função que alguém realiza no contexto de uma determinada atividade para alcançar os objetivos da mesma. As atividades referentes à alocação de recursos humanos são realizadas com o apoio da ferramenta *RHPlan* (SCHNAIDER, 2003), através da atividade do Processo de Desenvolvimento *Planejar Recursos Humanos,* sub-atividade integrante de *Implementação do Processo*.

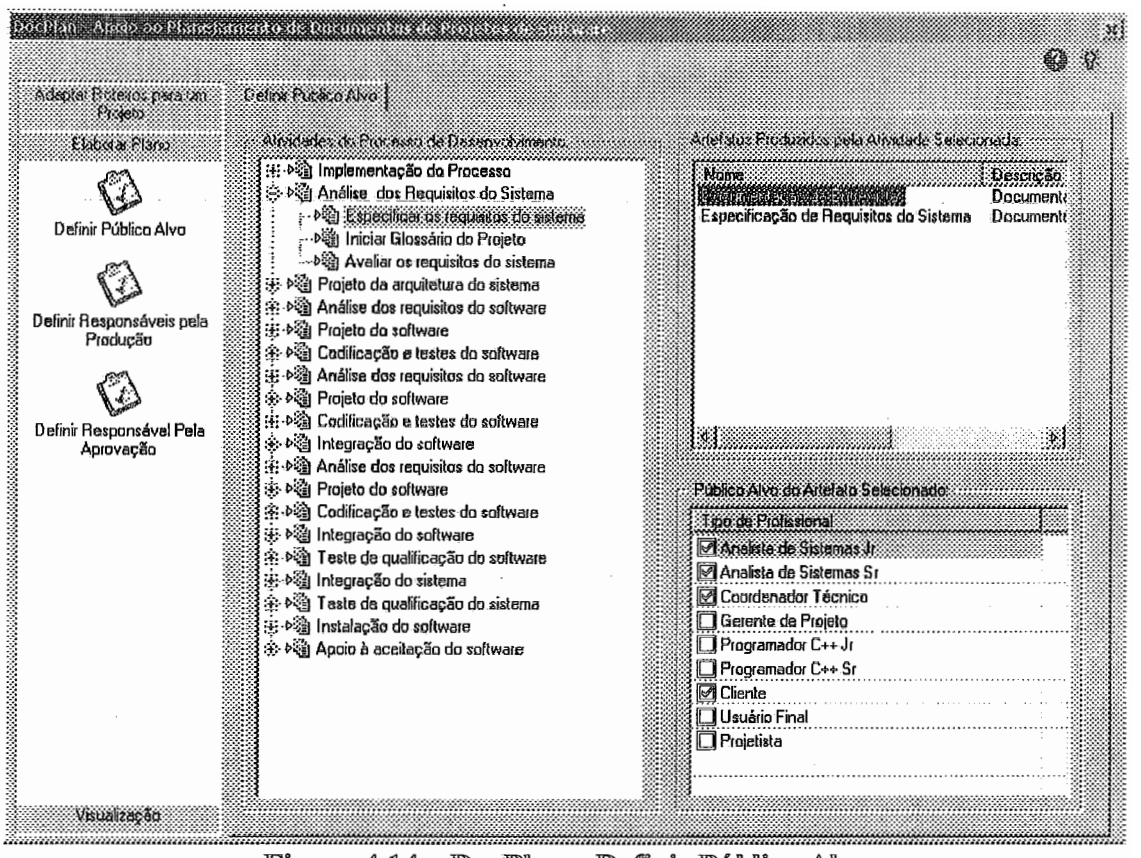


Figura 4.14 – DocPlan – Definir Público Alvo

De forma semelhante, o usuário define os responsáveis pela produção de um artefato através da atividade *Definir Responsáveis pela Produção.* A figura 4.15 ilustra esta atividade.

Para cada artefato, o usuário seleciona os responsáveis pela produção a partir de uma lista de pessoas estimadas como necessárias, ou efetivamente alocadas na realização da atividade selecionada, ou de suas macro-atividades. A necessidade de incluir na lista de pessoas passíveis de seleção, as estimadas para realização das macro-atividades, reside no fato de que no planejamento inicial da alocação de recursos humanos, em um primeiro momento, apenas cobre as macro-atividades, para posterior refinamento. A inclusão destas pessoas permite então um planejamento também inicial de responsabilidades pela produção de documentos. O planejamento e alocação de recursos humanos ocorre através da interação entre as ferramentas *CustPlan* (BARCELLOS, 2003) e ***RHPlan*** **(SCHNAIDER,** 2003).

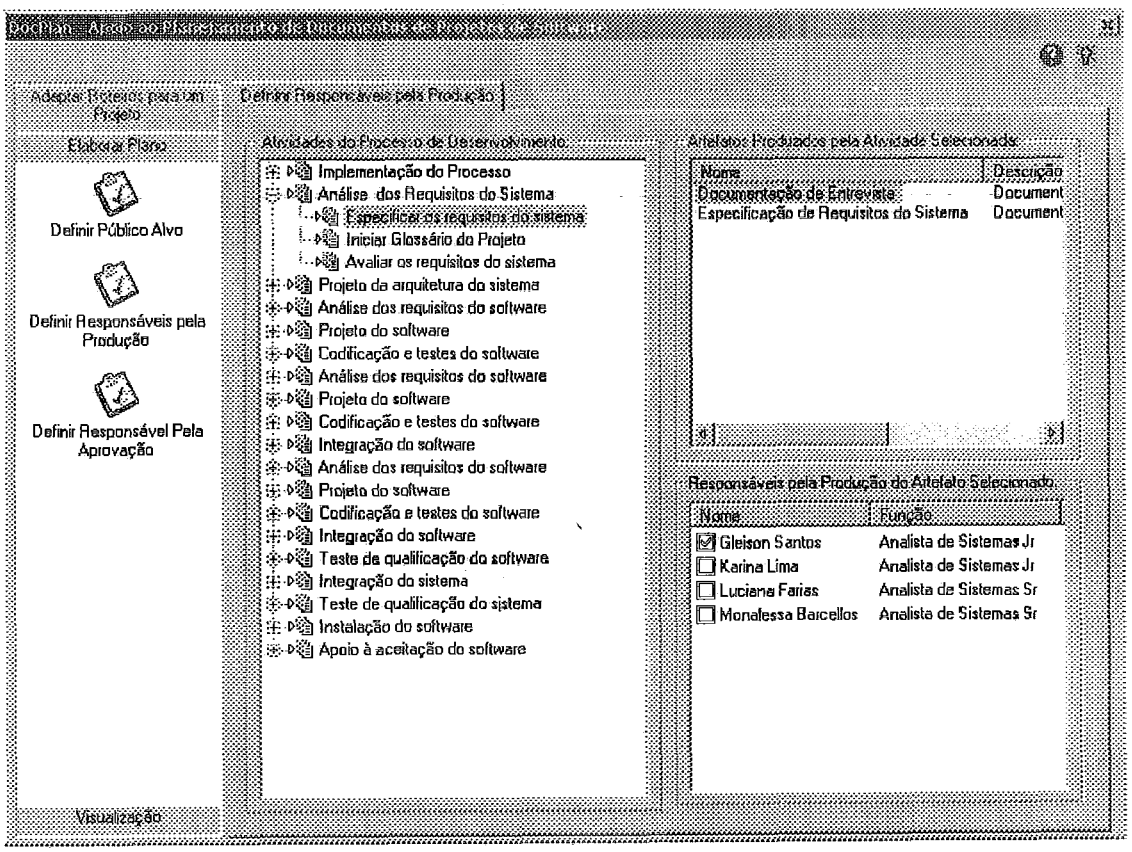


Figura 4.15 – DocPlam – Definir Responsáveis pela Produção

A atividade *Definir Responsáveis pela Aprovação* possui funcionalidade similar às demais descritas anteriormente e é ilustrada pela figura 4.16

Nesta atividade, o usuário define o responsável ou responsáveis pela aprovação de um artefato. As pessoas responsáveis pela aprovação são os membros da organização e não apenas do projeto, para abranger os casos em que o procedimento de aprovação de documentos da organização seja único entre vários projetos concorrentes.

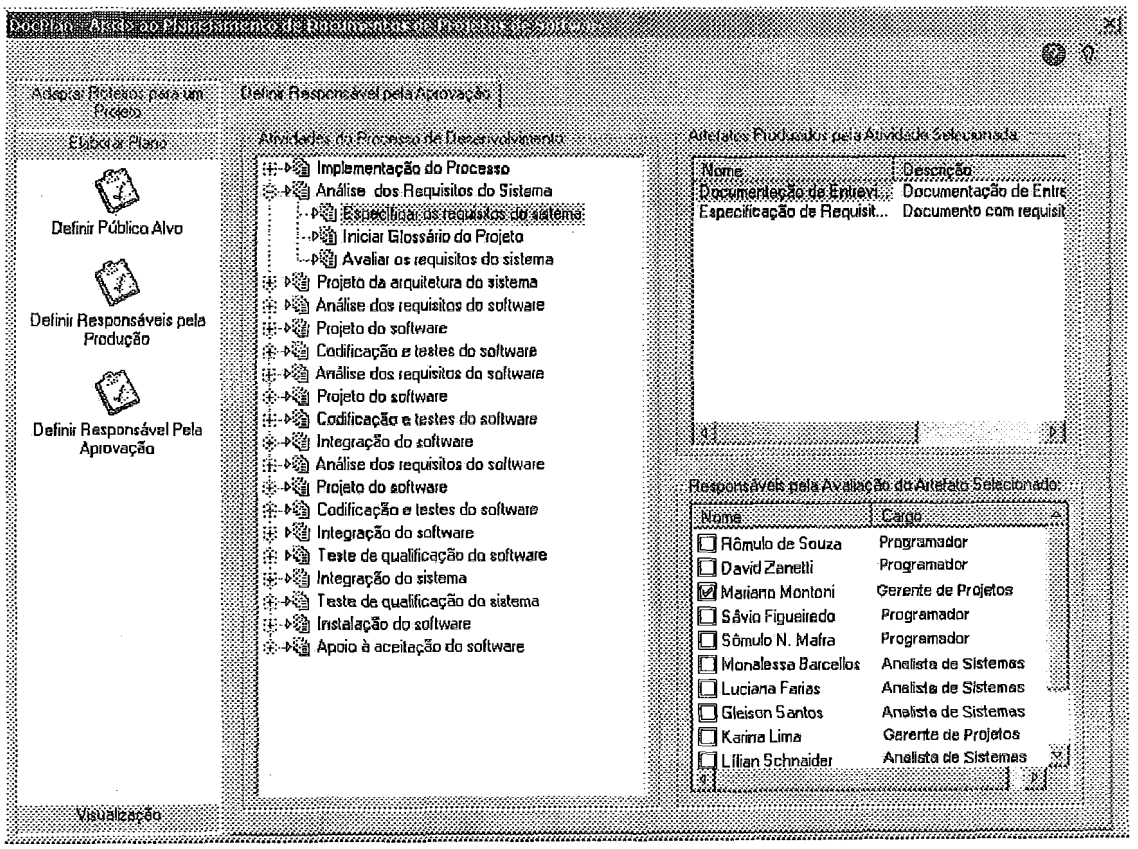


Figura 4.16 – DocPlan – Selecionar Responsáveis pela Aprovação

Visualização do Plano de Documentação

A atividade Visualizar Plano de Documentação, ilustrada pela figura 4.17, permite que o usuário visualize o plano de documentação elaborado ao longo das atividades anteriores. O plano constitui-se num sinóptico das atividades do processo, indicando datas de início e término de cada e o público-alvo, responsáveis pela produção e aprovação dos artefatos.

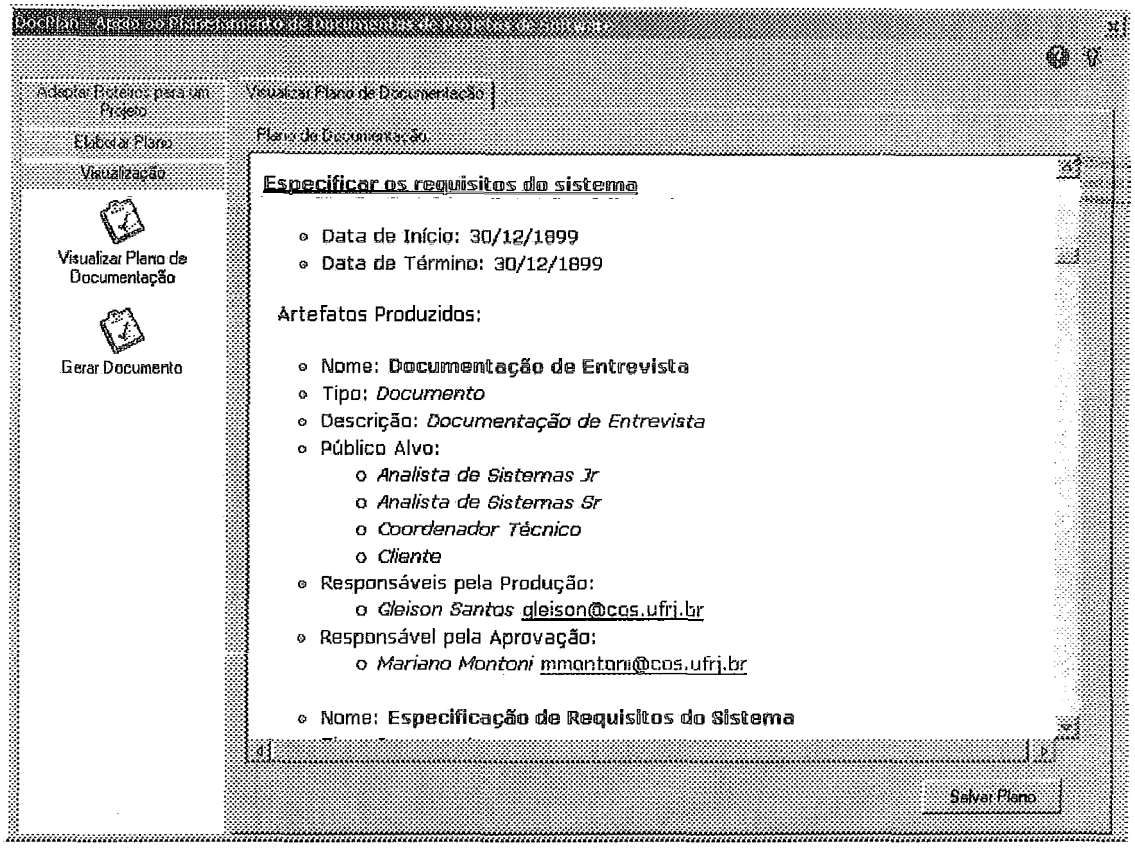


Figura 4.17 – DocPlan – Visualizar Plano de Documentação

Geração de Documentos

A Estação TABA permite que um artefato seja definido como sendo uma agregação de outros artefatos (sub-artefatos). No caso de artefatos compostos do tipo documento é possível a composição do documento resultante a partir dos seus sub-artefatos. Esta funcionalidade é provida através da atividade *Gerar Documentos* cuja interface é ilustrada pela figura 4.18.

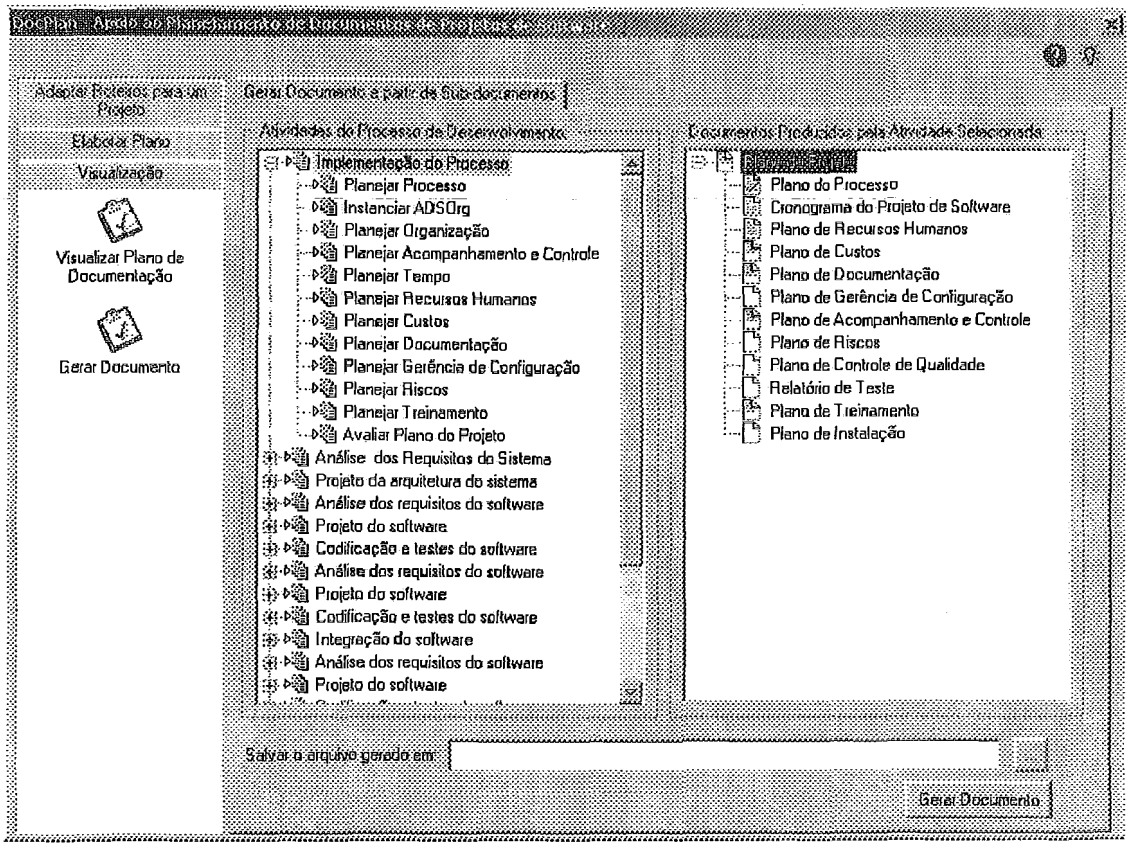


Figura 4.16 – DocPlan – Gerar Documentos a partir de sub-artefatos

Nesta atividade, uma vez selecionada uma atividade na árvore representativa do processo, é possível visualizar à direita uma árvore contendo todos os artefatos produzidos pela atividade e suas relações de composição. Selecionando-se um artefato composto e clicando no botão *Gerar Documento,* pode-se visualizar o documento resultante, composto pela concatenação de todos os documentos associados a cada sub-artefato, através do editor apropriado ao tipo de arquivo do documento. Por exemplo, selecionando-se o Plano do Projeto, é possível produzi-lo a partir da concatenação dos diversos sub-planos (plano de documentação, custos, tempo, riscos, etc).

Produção e Armazenamento de Documentos

A *Produção de Documentos* pode ser divida em três etapas, que são: início de um documento, desenvolvimento e conclusão. Uma vez concluído e aprovado um documento, este passa a estar sob responsabilidade do processo de gerência de configuração, que será

responsável pela gerência de alterações e distribuição do mesmo, caso este seja um item de configuração. A figura 4.19 ilustra a tela básica de entrada de um ADS instanciado.

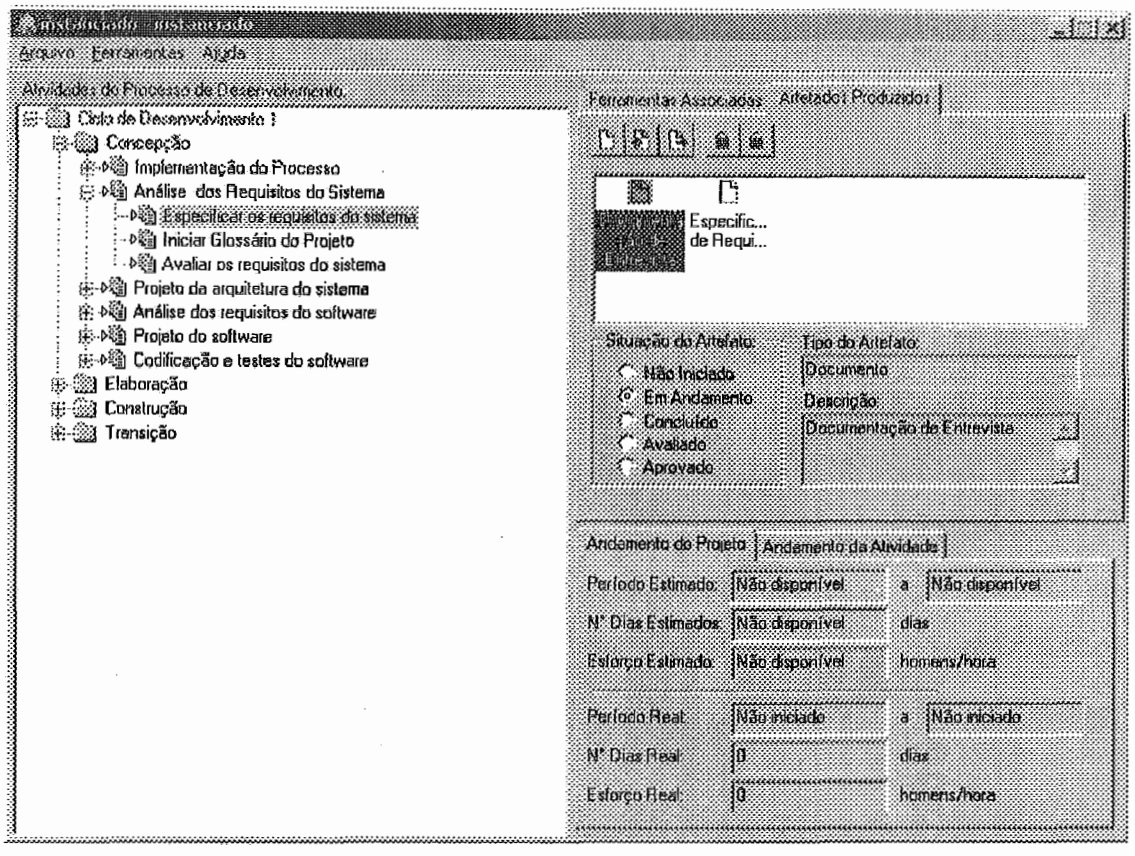


Figura 4.19 – ADS Instanciado – Artefatos Produzidos pelas Atividades

O exemplo ilustrado pela figura 4.19 indica como um artefato do tipo documento pode ser iniciado. Selecionando-se uma das atividades da árvore do processo, ilustrada à esquerda, é exibida, à direita, a lista de artefatos que devem ser produzidos pela atividade.

Acima da listagem dos artefatos produzidos existe uma barra de ferramentas com funções que podem ser executadas sobre o artefato selecionado: *Iniciar um Novo Documento* (função habilitada apenas se o artefato possuir roteiro associado); *Importar o Arquivo para o Repositório do ADSOrg*; *Obter Arquivo do Repositório do ADSOrg*; *Obter Uso Exclusivo do Arquivo*; *Liberar o Uso Exclusivo do Arquivo*. As últimas funcionalidades referem-se ao gerenciamento e armazenamento do arquivo que contém o documento em desenvolvimento representado pelo artefato selecionado. Desta forma, um

documento pode ser iniciado a partir de seu roteiro através do primeiro botão da barra de ferramentas.

Uma vez que a situação do artefato passa para a condição de aprovado, não é mais possível importar um arquivo ou iniciar o artefato através da barra de ferramentas. Qualquer alteração sobre artefatos aprovados será através da ferramenta *Gconf* (FIGUEIREDO, 2004) responsável pelo processo de gerência de configuração do ADSOrg, caso o artefato seja um item de configuração

4.3 Considerações Finais

O presente capítulo discutiu o trabalho de adequação da Estação TABA aos requisitos de instrumentação virtual, que também em parte atendem aos outros tipos de software. Um processo de documentação foi definido e todo o suporte computacional necessário à sua automação foi desenvolvido e inserido nos diferentes níveis de abstração da Estação TABA, além da criação de uma ferramenta específica para o planejamento da documentação, a *DocPlan*.

O próximo capítulo, após detalhar os conceitos envolvidos em níveis de abstração de processos, apresenta o processo de desenvolvimento de instrumentação virtual, que serve de base para o processo de software utilizado na empresa exemplo, sendo útil também para qualquer outra organização. Além disso, o ambiente configurado MDM ADS para empresa M&D Monitoração e Diagnose LTDA é apresentado, assim como um exemplo de sua utilização em um projeto específico, resultando no ADSOrg LASEEE.

CAPÍTULO V

AMBIENTE CONFIGURADO PARA ORGANIZAÇÕES DE DESENVOLVIMENTO DE INSTRUMENTAÇÃO VIRTUAL

Este capítulo apresenta os processos de software definidos como linha base para o desenvolvimento de instrumentos virtuais, um ambiente configurado para uma organização de desenvolvimento de instrumentação virtual, o MDM ADS, e um exemplo de sua aplicação em um projeto específico, o ADSOrg LASEEE.

Introdução

O capítulo anterior apresentou as adequações consideradas necessárias à Estação TABA para que essa pudesse configurar um ambiente com as características levantadas como importantes para instrumentação virtual. Finalmente, o capítulo V apresenta um ambiente configurado para desenvolvimento de instrumentação virtual. Para se chegar a este ambiente é necessária a execução das atividades do processo de configuração. Portanto, a definição do processo de software possui papel fundamental neste contexto.

No capítulo III discutiu-se o conceito de processo de software e a sua necessidade de padronização a organizações e projetos específicos. Na próxima seção serão apresentados, de forma objetiva, os conceitos envolvidos na definição de processo padrão, especializado e instanciado, para então discutir-se como os mesmos são abordados pela Estação TABA na seção 5.2. A seção 5.3 apresenta o processo especializado definido para o desenvolvimento de instrumentação virtual, que embora tenha sido elaborado para uma empresa específica, sua base pode ser utilizadas, a princípio, por empresas que trabalhem com este tipo de software. Por fim apresenta-se o ambiente configurado MDM ADS no contexto da empresa M&D Monitoração e Diagnose Ltda e um exemplo de sua aplicação em um projeto de desenvolvimento de instrumentação virtual, o projeto LASEEE.

5.1 Processo Padrão, Especializado e Instanciado

Processo de software pode ser definido como um conjunto de atividades, métodos, práticas e transformações que são usados para desenvolver e manter software e seus produtos associados, que podem incluir planos de projeto, documentos de projeto, código, casos de teste e manuais do usuário. À medida que uma organização amadurece, o processo de software se torna melhor definido e mais consistentemente implementado por toda a organização (PAULK *et al.*, 1997; RAMAN, 2000).

Pesquisas recentes têm mostrado a necessidade de padronizar a forma como software é desenvolvido dentro de uma organização com o objetivo de aumentar o controle e possibilitar melhorias dos processos de desenvolvimento de software (EMAN *et al.*, 1998) (NBR ISO/IEC 12207, 1997) (MAIDANTCHIK, 1999).

Esta padronização pode ser alcançada através da definição de um processo padrão que descreve as atividades que devem ser realizadas no desenvolvimento de sistemas de software em todos os projetos da organização. O processo padrão descreve os elementos fundamentais que devem ser incorporados em qualquer processo definido e as relações entre esses elementos O uso de modelos de processo genéricos como base para o planejamento do processo de software específico para um projeto permite aos gerentes definir planos em conformidade com os padrões de qualidade e procedimentos da organização (MAURER *et al.*, 1999; HENNINGER, 2001).

OLIVEIRA *et al.* (2000) observam que, a partir do processo padrão definido para a organização, diferentes processos de desenvolvimento de software podem ser especializados de acordo com as características de cada tipo de software. Um tipo de software refere-se a um conjunto de sistemas de software que utiliza uma determinada tecnologia de desenvolvimento (como, por exemplo, sistemas baseados em conhecimento, sistemas de informação, etc), seguindo um paradigma específico (como, por exemplo, orientado a objetos, estruturado, etc.). MACHADO (2000) acrescenta que, além do tipo de software, deve-se observar características do desenvolvimento. Nesse momento, novas atividades podem, também, ser definidas e incluídas no processo especializado. No entanto, todos os elementos básicos definidos no processo padrão deverão sempre estar presentes nos processos especializados.

Para ser utilizado em um projeto específico, o processo especializado mais adequado para um determinado tipo de software deve ser instanciado para atender às características do projeto. Na instanciação para projetos, deve-se considerar, por exemplo, o tamanho e a complexidade do produto, as características da equipe de desenvolvimento, a expectativa de vida útil do software e demais características do projeto, além do modelo de ciclo de vida mais apropriado.

Esta abordagem de definição de processos é ilustrada pela figura 5.1, que exibe as atividades envolvidas na definição dos processos e o que é considerado em cada momento de definição.

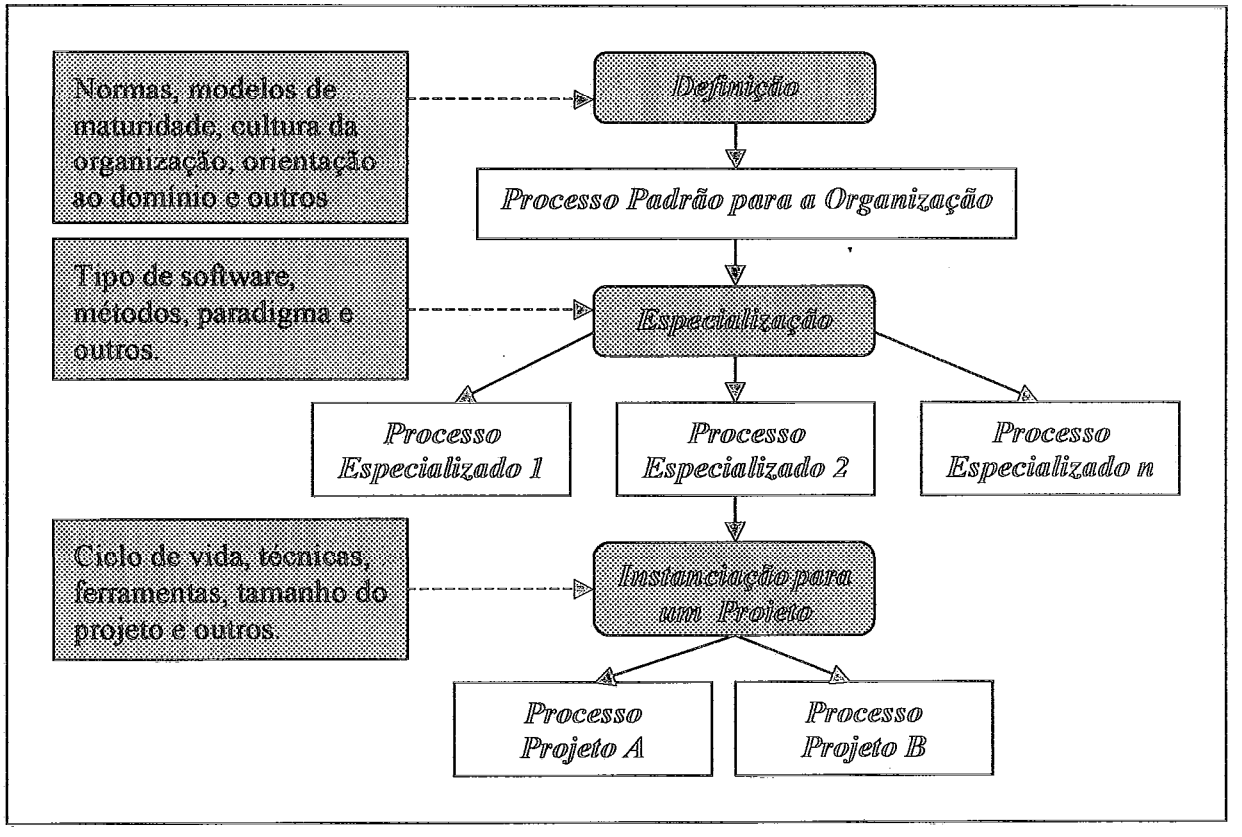


Figura 5.1 – Esquema de Definição de Processos de Software

5.2 Processo de Software na Estação TABA

A Estação TABA utiliza a abordagem em 3 níveis de abstração apresentada na seção anterior para a definição do processo de desenvolvimento que guia a execução de um ADSOrg. Esta seção discute como este modelo de hierarquia de processos ocorre no contexto da Estação TABA.

Nos capítulos anteriores foram discutidos os níveis de abstração da Estação TABA: meta-ambiente, ambiente configurado e ADSOrg, além das ferramentas que apóiam os processos de configuração (*Config*) e instanciação (*AdaptPro*) responsáveis pela passagem de um nível para o outro. Através destes ambientes e ferramentas é que também ocorre a definição dos processos de software. A figura 5.2 ilustra o mapeamento da figura 5.1 no contexto dos ambientes e ferramentas da Estação TABA.

No meta-ambiente, a definição dos processos de software ocorre através da ferramenta *Config* (VILLELA, 2004), responsável pela configuração de ambientes para uma organização. Esta ferramenta, entre suas funções, guia o engenheiro de software na definição do processo padrão da organização e permite que este se beneficie do conhecimento disponível no meta-ambiente da Estação TABA sobre atividades de processos de software. Estas atividades incluem, por exemplo, atividades previstas na norma ISO/IEC 12207 (NBR ISO/IEC 12207, 1997) e nos modelos de maturidade CMM (ABRAN, 2001), CMMI (CHRISSIS *et al.*, 2003) e SPICE (ISO/IEC TR 15504, 1998).

Através da atividade *Definir Processo Padrão* da ferramenta *Config*, o engenheiro de software pode selecionar atividades existentes em normas, modelos de maturidade e capacidade, na organização ou orientadas a domínio. Existe ainda a possibilidade de selecionar atividades do tipo de software para inclusão no processo padrão, caso a empresa desenvolva somente um tipo de software.

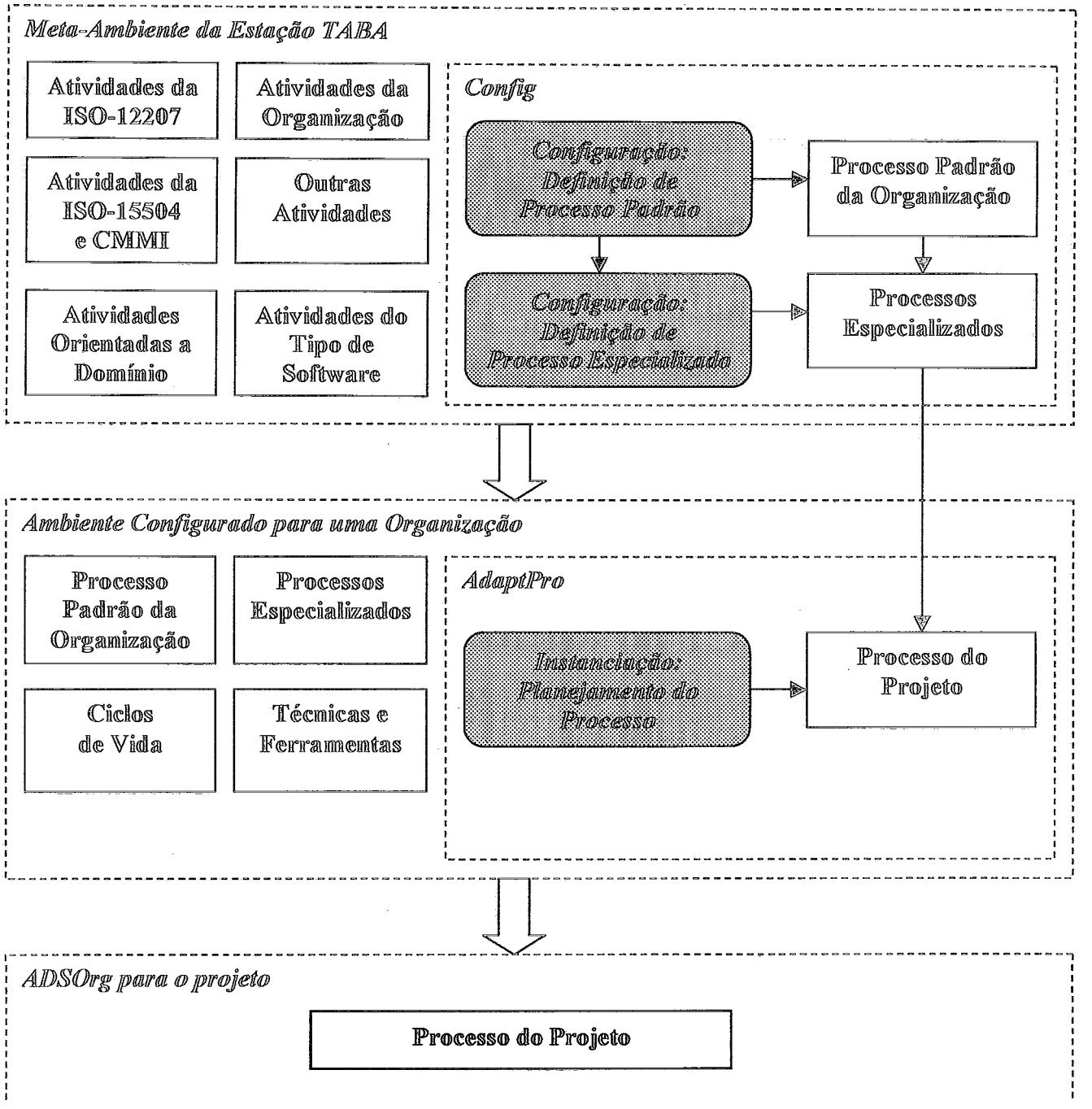


Figura 5.2 – Esquema de Definição de Processos de Software na Estação TABA

A atividade *Definir Processo Especializado* permite que um processo padrão seja particularizado para um determinado tipo de software e paradigma de desenvolvimento. A ferramenta automaticamente substitui as atividades de desenvolvimento pertinentes por específicas ao paradigma. Estas atividades especializadas contêm conhecimento sobre o

desenvolvimento no paradigma adotado. Por exemplo, a atividade *Especificar Requisitos de Software* possui na sua especialização para o paradigma Orientado a Objetos roteiro para elaboração de Casos de Uso.

Uma vez definidos os processos especializados, pode-se então, através do ambiente configurado e sua ferramenta *AdaptPro* (BERGER, 2003), instanciar um processo de software para um projeto específico. A ferramenta *AdaptPro* apóia o processo de instanciação de ambiente e também no planejamento do processo, orientando na sua adequação às características do projeto, na seleção do modelo de ciclo de vida e demais atividades necessárias.

A atividade *Planejar Processo* pertencente à ferramenta *AdaptPro* permite moldar um processo especializado a um determinado modelo de ciclo de vida, selecionar técnicas de avaliação da qualidade de acordo com requisitos previamente estabelecidos, além da seleção de ferramentas para cada atividade do processo.

Ao final destas atividades, realizadas no meta-ambiente e no ambiente configurado, é gerado como produto final o ambiente instanciado (ADSOrg) com o processo pronto para guiar todo o ciclo de desenvolvimento do software.

5.3 Processo de Desenvolvimento de Instrumentação Virtual

O processo de desenvolvimento de instrumentação virtual pode ser considerado, conforme discussão nas seções anteriores, como um processo especializado para um tipo de software. Embora este processo seja a base do ambiente configurado MDM ADS, apresentado na próxima seção, ele não contempla atividades específicas da organização. O objetivo da ausência de tais atividades é que o processo sirva de base para qualquer outra empresa que decida adotar a abordagem de desenvolvimento de instrumentação virtual apresentada neste trabalho.

A definição do processo especializado de desenvolvimento de instrumentação virtual leva em consideração três pontos:

- As atividades pertinentes do processo fundamental de desenvolvimento da norma NBR ISO/IEC 12207 (1997);

- As atividades específicas do tipo de software instrumentação virtual como, por exemplo, *Especificar Hardwares e Equipamentos*;
- Atividades do paradigma de desenvolvimento Orientado a Objetos.

A inclusão do paradigma na definição do processo de desenvolvimento de instrumentação virtual tem a desvantagem de atar à adoção da Orientação a Objetos ao desenvolvimento de instrumentação virtual. No entanto, esta adoção está de acordo com as necessidades e particularidades levantadas para a área de instrumentação virtual ao longo deste trabalho. O capítulo II apresentou as vantagens da adoção deste paradigma no desenvolvimento de instrumentação virtual. O capítulo I argumenta que quanto mais específicas forem as soluções adotadas, sem a perda da generalidade de suas aplicações, melhor seria, pois é importante que não especialistas em engenharia de software utilizem a abordagem aqui proposta. Portanto, quanto maior o nível de detalhe do processo de software, mais fácil será a sua aplicação e a adoção do paradigma de desenvolvimento contribui de forma substancial para este detalhamento.

A tabela 5.1 lista todas as atividades do processo de desenvolvimento de instrumentação virtual. A seguir detalha-se cada uma das atividades.

Tabela 5.1 – Processo de Desenvolvimento de Instrumentação Virtual

Implementação do Processo	
1. Planejar Processo	7. Planejar Custos
2. Instanciar ADSOrg	8. Planejar Documentação
3. Planejar Organização do Projeto	9. Planejar Gerência de Configuração
4. Planejar Acompanhamento e Controle	10. Planejar Riscos
5. Planejar Tempo	11. Planejar Treinamento
6. Planejar Recursos Humanos	12. Avaliar Plano do Projeto
Refinar/detalhar planejamento das próximas atividades	
13. Refinar/detalhar Planejamento de Tempo para próximas atividades	15. Refinar/detalhar Planejamento de Custos detalhado para próximas atividades
14. Refinar/detalhar Planejamento de **Recursos** Humanos para próximas atividades	
Análise dos requisitos do sistema	
16. Especificar os requisitos do sistema	18. Especificar hardwares e equipamentos
17. Iniciar Glossário do Projeto	19. Avaliar os requisitos do sistema

Projeto da arquitetura do sistema	
20. Especificar uma arquitetura de alto nível do sistema	21. Avaliar a arquitetura do sistema
Análise dos requisitos do Instrumento Virtual	
22. Elicitar e especificar requisitos do Instrumento Virtual	24. Avaliar os requisitos do Instrumento Virtual
23. Planejar Controle da Qualidade	
Projeto do Instrumento Virtual	
25. Projeto da aplicação	27. Projeto da base de dados
26. Projeto da interface	28. Avaliar o Modelo de Projeto
Codificação e testes do Instrumento Virtual	
29. Codificar Unidades	33. Planejar Teste do Instrumento Virtual
30. Planejar testes de Unidades	34. Integrar as unidades do Instrumento Virtual
31. Testar unidades	35. Planejar testes de qualificação do Instrumento Virtual
32. Avaliar o código	36. Avaliar plano teste de qualificação do Instrumento Virtual e testes realizados
Teste de qualificação do Instrumento Virtual	
37. Realizar testes de qualificação do Instrumento Virtual	39. Elaborar versão preliminar da documentação do usuário
38. Avaliar o Instrumento Virtual	40. Preparar o produto de software a ser entregue
Integração do sistema	
41. Integrar sistema	43. Avaliar Sistema Integrado
42. Planejar testes de qualificação do sistema	
Teste de qualificação do sistema	
44. Realizar homologação interna	45. Realizar homologação externa
Instalação do Instrumento Virtual	
46. Planejar a instalação do sistema	49. Efetuar a carga inicial de dados
47. Completar a documentação do sistema	50. Instalar o sistema
48. Completar documentação do usuário	
Apoio à aceitação do Instrumento Virtual	
51. Apoiar a avaliação do usuário	53. Realizar treinamento
52. Realizar entrega do sistema	
Encerramento do projeto	
54. Realizar Avaliação post-mortem	

Implementação do Processo
Descrição: Planejar o processo de **desenvolvimento** e demais **planos** para realizar as atividades do processo de desenvolvimento.

Artefato: Plano do Projeto (Plano do Processo + Cronograma + Plano de Recursos Humanos + Plano de Custos + Plano de Documentação + Plano de Gerência de Configuração + Plano de Acompanhamento e Controle + Plano de Riscos + Plano de controle da Qualidade + Plano de Testes + Plano de Treinamento + Plano de Instalação).
Sub-atividades
Nome: Planejar Processo *Descrição*: Gerar do Plano do Processo para o novo projeto, que é o processo instanciado. *Responsabilidade:* gerente do projeto *Artefato:* Plano do Processo *Ferramenta: AdaptPro*
Nome: Instanciar ADSOrg *Descrição*: Esta atividade tem como objetivo a instanciação de um Ambiente de Desenvolvimento de Software (ADSOrg) para a realização do projeto, a partir do processo instanciado. *Artefato:* ADSOrg Responsável: gerente do projeto Ferramenta: *AdaptPro*
Nome: Planejar Organização do Projeto *Descrição*: Realizar o planejamento da organização do projeto, identificando para cada parte envolvida, os responsáveis e os contatos. *Responsabilidade:* gerente do projeto *Artefato:* Plano de Organização *Ferramenta: OrgPlan*
Nome: Planejar Acompanhamento e Controle *Descrição*: Gerar o planejamento do acompanhamento e controle do projeto, identificando para cada marco e ponto de controle, os métodos de acompanhamento (apresentações orais, reuniões, relatórios, planilhas, acordos de aceite, etc), os responsáveis e os participantes em cada atividade de acompanhamento. *Responsabilidade:* gerente do projeto *Artefato:* Plano de Acompanhamento e Controle *Ferramenta: ControlPlan*
Nome: Planejar Tempo *Descrição*: Gerar o planejamento inicial de tempo do projeto *Responsabilidade:* gerente do projeto *Artefato:* Cronograma *Ferramenta:* TempPlan1
Nome: Planejar Recursos Humanos *Descrição*: Gerar o planejamento de recursos humanos do projeto *Responsabilidade:* gerente do projeto *Artefato:* Plano de Recursos Humanos *Ferramenta:* RHPlan1

Nome: Planejar Custos *Descrição*: Gerar o planejamento inicial de custos do projeto *Responsabilidade:* gerente do projeto *Artefato:* Plano de Custos *Ferramenta:* CustPlan1
Nome: Planejar Documentação *Descrição*: Gerar o planejamento da documentação, identificando os produtos a serem gerados durante o desenvolvimento do projeto. Para cada documento deve ser definido: público-alvo; responsabilidades pelo desenvolvimento e aprovação. *Responsabilidade:* gerente do projeto *Artefato:* Plano de Documentação *Ferramenta: DocPlan*
Nome: Planejar Gerência de Configuração ***Descrição*: Gerar o planejamento da Gerência** de configuração para o projeto. O plano deve descrever as atividades da gerência de configuração, procedimentos e cronograma para executar as atividades, os responsáveis pela execução das atividades. *Responsabilidade:* gerente do projeto *Artefato:* Plano de Gerência de Configuração e itens de software sob gerência de configuração Ferramenta: *GConf*
Nome: Planejar Riscos *Descrição*: Gerar o planejamento de riscos do projeto *Responsabilidade:* gerente do projeto *Artefato:* Plano de Riscos *Ferramenta: RiscPlan*
Nome: Planejar Treinamento *Descrição*: Gerar o planejamento dos diferentes tipos de treinamento necessários para o desenvolvimento, implantação, operação e manutenção do produto. *Responsabilidade:* gerente do projeto *Artefato:* Plano de Treinamento *Ferramenta: Word*
Nome: Avaliar Plano do Projeto *Descrição*: O conjunto de Planos gerados devem ser avaliados considerando-se os seguintes critérios: viabilidade econômica, viabilidade financeira, viabilidade de cronograma, viabilidade de mão de obra, viabilidade tecnológica, viabilidade legal. *Artefato:* Plano do Projeto aprovado, Laudo de Preparação Individual de Avaliação, Laudo de Avaliação Final. Responsável: gerente do projeto e patrocinador

Refinar/detalhar planejamento das próximas atividades
Descrição: Refinar ou detalhar planejamentos existentes a partir das novas informações existentes sobre o sistema e/ou projeto.
Artefato: Plano do Projeto revisto e modificado
Sub-atividades

Nome: Refinar/detalhar Planejamento de Tempo para próximas atividades
Descrição: Atualizar planejamento de tempo do projeto, refinando e detalhando o anterior para as próximas atividades que o gerente julgou, baseado do nível de informação existente, como passíveis de tal detalhamento.
Responsabilidade: gerente do projeto
Artefato: Cronograma revisto
Ferramenta: TempPlan2

Nome: Refinar/detalhar Planejamento de Recursos Humanos para próximas atividades
Descrição: Atualizar planejamento de Recursos Humanos do projeto, refinando e detalhando o anterior para as próximas atividades que o gerente julgou, baseado do nível de informação existente, como passíveis de tal detalhamento.
Responsabilidade: gerente do projeto
Artefato: Plano de Recursos Humanos Revisto
Ferramenta: RHPlan2

Nome: Refinar/detalhar Planejamento de Custos para próximas atividades
Descrição: Atualizar planejamento de Custos do projeto, refinando e detalhando o anterior para as próximas atividades que o gerente julgou, baseado do nível de informação existente, como passíveis de tal detalhamento.
Responsabilidade: gerente do projeto
Artefato: Plano de Custos Revisto
Ferramenta: CustPlan2

Análise dos Requisitos do Sistema

Descrição: Análise do uso pretendido do sistema para especificar seus requisitos gerais, funcionalidades e restrições, se aplicável.

Artefato: Especificação de Requisitos do Sistema, Glossário, Especificação de Hardware e Equipamentos, Laudo de Preparação Individual para Avaliação dos Requisitos do Sistema, Laudo Final da Avaliação dos Requisitos do Sistema, Relatório Histórico, Plano de Ação.

Sub-atividades

Nome: Especificar os requisitos do sistema
Descrição: Especificar funções e capacidades do sistema, requisitos de negócio, organizacionais e de usuários, requisitos de proteção e de segurança, de fatores humanos, interface e manutenção. Neste momento deve-se levantar todos os drives necessários para a comunicação com outros sistemas e/ou equipamentos. Deve-se também definir restrições, limitações do sistema e suposições que afetam o produto.
Artefato: Documentação de Entrevistas, Especificação de Requisitos do Sistema, Relatório Histórico.
Responsabilidades: analistas de sistemas, usuários
Ferramentas de Software: Word, Navegue

Nome: Iniciar Glossário do Projeto *Descrição*: Se pertinente deve-se iniciar o glossário do projeto *Artefato*: Glossário do Projeto, Relatório Histórico. *Responsabilidades:* analistas de sistemas *Ferramentas de Software: Word*, Navegue
Nome: Especificar hardware e equipamentos. *Descrição*: Especificar hardware e equipamentos necessários ao sistema de instrumentação virtual, tais como placas de aquisição, sistemas de condicionamento, equipamentos de controle, etc. *Artefato*: Especificação de hardware e equipamentos, Relatório Histórico. *Responsabilidades:* Engenheiro de Aplicação *Ferramentas de Software: Word*
Nome: Avaliar os requisitos do sistema *Descrição*: Avaliar os requisitos segundo os seguintes critérios: Rastreabilidade para as necessidades de aquisição, consistência com as necessidades de aquisição, testabilidade, viabilidade do projeto da arquitetura do sistema, viabilidade de operação e manutenção. *Artefato*: Laudo de Preparação Individual para Avaliação dos Requisitos do Sistema, Laudo Final da Avaliação dos Requisitos do Sistema, Especificação de Requisitos aprovada, Relatório Histórico, Plano de Ação Responsabilidades: gerente do projeto e usuários. *Ferramentas de Software: Word,* Navegue

Projeto da arquitetura do sistema
Descrição: Definição de uma arquitetura geral do sistema identificando itens de hardware, software e operações manuais, se aplicável
Artefato: Projeto da Arquitetura do Sistema, Relatório Histórico, Laudo de Preparação Individual para Avaliação da Arquitetura do Sistema, Laudo Final da Avaliação da Arquitetura do Sistema, Plano de Ação.
Sub-atividades
Nome: Especificar uma arquitetura de alto nível do sistema *Descrição*: Estabelecer uma arquitetura de alto nível do sistema (que identifique itens de hardware, software e operações manuais) e assegurar que todos os requisitos do sistema sejam alocados. Especificar como cada módulo do sistema irá interagir com os diversos equipamentos e itens de hardware para alcançar o objetivo do sistema. *Artefato*: Projeto da Arquitetura do Sistema, Relatório Histórico *Responsabilidades:* Usuários, Analista de Sistemas *Ferramentas de Software: Word*, Rose ou System Architect.

Nome: Avaliar a arquitetura do sistema
Descrição: Avaliar a arquitetura de alto nível considerando os seguintes critérios: Rastreabilidade e consistência com os requisito do sistema, adequação dos métodos e padrões de projetos utilizados, viabilidade de os itens de software atenderem seus requisitos alocados, viabilidade de operação e manutenção.
Artefato: Laudo de Preparação Individual para Avaliação da Arquitetura do Sistema, Laudo Final da Avaliação da Arquitetura do Sistema, Projeto da Arquitetura do Sistema aprovado, Relatório Histórico, Plano de Ação.
Responsabilidades: Gerente de Projeto, Usuários
Ferramentas de Software: Word

Análise dos requisitos do Instrumento Virtual

Descrição: Analisar os requisitos do Instrumento Virtual a ser construído tanto no que se refere a requisitos funcionais como a requisitos não funcionais.

Artefato: Especificação de Requisitos do Instrumento Virtual, Documentação de Entrevistas, Laudo de Preparação Individual para Avaliação dos Requisitos do Instrumento Virtual, Laudo Final de Avaliação dos Requisitos do Instrumento Virtual, Relatório Histórico, Plano de Ação, Plano de Controle da Qualidade.

Sub-atividades

Nome: Especificar requisitos do Instrumento Virtual
Descrição: Estabelecer e documentar requisitos funcionais e não funcionais (requisitos de interface, requisitos de qualidade, requisitos tecnológicos, requisitos legais, requisitos de proteção, requisitos de segurança, definição de dados e requisitos de bases de dados, requisitos de instalação, requisitos do usuário para execução e operação, requisitos do usuário para manutenção. A modelagem a ser realizada depende do método de desenvolvimento escolhido. A especificação dos requisitos de qualidade é feita considerando-se as características de qualidade da ISO 9126, podendo ser acrescentadas outras características de qualidade, caso pertinente.
Artefatos: Especificação de Requisitos do Instrumento Virtual, Documentação de Entrevistas, Relatório Histórico
Responsabilidades: Usuário e analistas de sistemas
Ferramentas de Software: Word, Rose ou System Architect; Navegue, Q-Fuzzy

Nome: Planejar Controle da Qualidade
Descrição: Gerar o planejamento do controle da qualidade do projeto. O plano deve indicar os artefatos de software sujeitos a controle da qualidade, os critérios de avaliação a serem considerados a partir dos requisitos de qualidade identificados para o produto, as técnicas de avaliação a serem utilizadas, as responsabilidades e cronograma associados.
Responsabilidade: gerente do projeto
Artefato: Plano de Controle da Qualidade
Ferramenta: QualPlan

Nome: Avaliar os requisitos do Instrumento Virtual
Descrição: Avaliar os requisitos segundo os seguintes critérios: rastreabilidade para os requisitos do sistema e projeto do sistema, consistência externa, testabilidade, viabilidade do projeto de software, viabilidade da operação e manutenção.
Artefatos: Especificação de Requisitos do Instrumento Virtual aprovada, Laudo de Preparação Individual para Avaliação dos Requisitos do Instrumento Virtual, Laudo Final de Avaliação dos Requisitos do Instrumento Virtual, Relatório Histórico, Plano de Ação.
Responsabilidades: Gerente de Projeto e Usuário.
Ferramentas: Word

Projeto do Instrumento Virtual

Descrição: Esta atividade tem como objetivo a transformação do Modelo de Análise no Modelo de Projeto, utilizando-se os procedimentos do método de desenvolvimento escolhido. O modelo de projeto deve representar as três dimensões: funções (Projeto da Aplicação), dados (Projeto do Banco de Dados) e interface (Projeto da Interface)

Artefato: Modelo de Projeto (Projeto da Aplicação + Projeto da Interface + Projeto do Banco de Dados).

Sub-atividades

Nome: Projeto da aplicação
Descrição: O Projeto da Aplicação tem como objetivo mapear o Modelo de Análise numa arquitetura técnica. Dado que são possíveis várias arquiteturas, o objetivo é escolher a melhor configuração considerando os requisitos e a disponibilidade de tecnologia. Determinar a arquitetura correta envolve determinar o tamanho da capacidade computacional para o problema.
Artefatos: Projeto da Aplicação, Relatório Histórico
Responsabilidades: analistas
Ferramentas de Software: Word, Rose.

Nome: Projeto da interface
Descrição: O projeto da Interface especifica a parte do produto que é visível para o usuário. Pode-se, inicialmente, fazer um protótipo da interface (com foco na principal tarefa do usuário) e avaliá-lo com o usuário. O projeto da interface deve determinar: mapa de navegação (descreve que janelas estão disponíveis e os possíveis caminhos de navegação entre elas), layout de janelas, descrição das janelas (descreve a função e características da janela), mini-especificação das janelas (define o comportamento para abrir e fechar a janela e da execução de cada botão, controle e item do menu) e especificação de campos (define os campos para cada elemento de dados que aparece na janela, como os dados serão obtidos, lista de nomes nas tabelas e colunas, indicação de como unir tabelas e descrição de como aplicar critérios de seleção).
Artefatos: Projeto da Interface, Relatório Histórico
Responsabilidades: Analista de Sistemas
Ferramentas de Software: Word

Nome: Projeto da base de dados *Descrição*: Desenvolver e documentar um projeto de alto nível para as bases de dados. As atividades a serem realizadas dependem do método de desenvolvimento escolhido. *Artefatos:* Projeto do Banco de Dados, Relatório Histórico *Responsabilidades:* Analista de Sistemas *Ferramentas de Software:* Word, Er-Win
Nome: Avaliar o Modelo de Projeto *Descrição*: Avaliar o modelo de projeto segundo os seguintes critérios: Rastreabilidade para os requisitos do item de software, consistência externa com os requisitos do item de software consistência interna, adequação dos métodos e padrões de projeto utilizados, viabilidade da operação e manutenção. *Artefatos*: Projeto Detalhado do Instrumento Virtual aprovado, Requisitos de Teste aprovado, Laudo de Preparação Individual para Avaliação do Projeto Detalhado e Requisitos de Teste, Laudo Final da Avaliação do Projeto Detalhado e Requisitos de Teste, Relatório Histórico, Plano de Ação. *Responsabilidades:* Gerente de Projeto. *Ferramentas de Software:* Word.

Codificação e testes do Instrumento Virtual
Descrição: Codificação e testes das unidades do sistema.
Artefato: Código, Plano de Testes de Unidades, Cronograma de Testes de Integração, Relatório de Testes, Laudo de Preparação Individual para Avaliação do Código, Laudo Final da Avaliação do Código, plano de ação, Plano de Teste de Qualificação do Instrumento Virtual, Laudo de Preparação Individual para Avaliação do Plano de Teste de Qualificação do Instrumento Virtual, Laudo Final de Avaliação do Plano de Teste de Qualificação do Instrumento Virtual, Relatório Histórico.
Sub-atividades
Nome: Codificar unidades *Descrição*: Desenvolver e documentar cada unidade de software e base de dados e os procedimentos de teste e dados para testar cada unidade de software e a base de dados. *Artefatos:* Código, Relatório Histórico *Responsabilidade*: programadores
Nome: Planejar testes de Unidades *Descrição*: Definir e documentar os requisitos de teste e o cronograma para testar as unidades. Atualizar os requisitos de teste e o cronograma para a integração do instrumento virtual. *Artefato:* Plano de Testes de Unidades, Cronograma de Testes do Instrumento Virtual, Relatório Histórico *Responsabilidades:* Gerente de Projeto *Ferramentas de Software:* Word

Nome: Testar unidades *Descrição*: Testar cada unidade de software e base de dados, garantindo que sejam atendidos os seus requisitos. Os resultados dos testes devem ser documentados. *Responsabilidade*: programadores *Artefatos:* unidades testadas, Relatório de Testes, Relatório Histórico *Ferramentas de Software:* Word
Nome: Avaliar o código *Descrição*: Avaliar o código das unidades e os resultados dos testes, considerando os seguintes critérios: rastreabilidade para os requisitos e projeto, consistência externa com os requisitos e projeto, consistência interna entre os requisitos da unidade, cobertura de teste das unidades, adequação dos métodos e padrões de codificação utilizados, viabilidade da integração e testes do software, viabilidade da operação e manutenção. Artefato: Código de unidades aprovado, Laudo de Preparação Individual para Avaliação do Código, Laudo Final da Avaliação do Código, Relatório Histórico, Plano de Ação *Responsabilidade*: analistas *Ferramentas*: Word
Nome: Planejar Teste do Instrumento Virtual *Descrição*: Desenvolver um plano de teste de integração para integrar as unidades do instrumento virtual,componentes de software e hardware. Planejar testes para aferir a correta integração entre hardware e software. O plano deve conter: requisitos de teste, procedimentos, dados, responsbilidades e cronograma. *Artefato*: Plano de Teste, Relatório Histórico *Responsabilidades:* Gerente de Projeto e analistas *Ferramenta de software:* Word
Nome: Integrar as unidades do Instrumento Virtual *Descrição*: O desenvolvedor deve integrar as unidades e componentes do instrumento virtual (software e hardware) e testar estas agregações à medida que forem sendo integradas. Os resultados da integração e dos testes devem ser documentados. Deve-se garantir que o item de software integrado está pronto para o teste de qualificação do Instrumento Virtual. *Artefato*: unidades integradas, Relatório de Testes, Relatório Histórico *Responsabilidades:* analistas e programadores
Nome: Planejar testes de qualificação do Instrumento Virtual *Descrição*: Desenvolver e documentar para cada requisito de qualificação do item de software, um conjunto de testes, casos de teste (entradas, saídas e critérios de teste) e procedimentos de teste. *Artefatos Produzidos:* Plano de Teste de Qualificação do Instrumento Virtual, Relatório Histórico *Responsabilidades:* Gerente de Projeto, analistas *Ferramentas de Software:* Word

Nome: Avaliar plano teste de qualificação do Instrumento Virtual e testes realizados
Descrição: Avaliar o plano do teste de qualificação do Instrumento Virtual, projeto, código, testes e resultados dos testes considerando os seguintes critérios: rastreabilidade para os requisitos do sistema, consistência externa com os requisitos do sistema, consistência interna, cobertura de teste dos requisitos de software, adequação dos métodos e padrões de teste utilizados, conformidade com os resultados esperados, viabilidade do teste de qualificação do software, viabilidade de operação e manutenção.
Artefatos: Software integrado, Plano de Teste de Qualificação do Instrumento Virtual aprovado, Laudo de Preparação Individual para Avaliação do Plano de Teste de Qualificação do Instrumento Virtual e do Plano de Integração, Laudo Final de Avaliação do Plano de Teste de Qualificação do Instrumento Virtual e do Plano de Integração, Relatório Histórico, Plano de Ação
Responsabilidades: **Gerente de Projeto, Usuário**
Ferramenta: Word

Teste de qualificação do Instrumento Virtual

Descrição: Realizar o teste do Instrumento Virtual desenvolvido no que se refere aos requisitos estabelecidos

Artefato: Relatórios de teste, Laudo de Preparação Individual para Avaliação do Instrumento Virtual, Laudo Final da Avaliação do Instrumento Virtual, Relatório Histórico, Plano de Ação, Manual do Usuário

Sub-atividades

Nome: Realizar testes de qualificação do Instrumento Virtual
Descrição: Conduzir e documentar o teste de qualificação para o Instrumento Virtual. Deve ser garantido que a implementação de cada requisito do software seja testada para conformidade. Os resultados do teste devem ser documentados.
Artefatos: Instrumento Virtual com teste de qualificação realizado, Relatórios de teste, Relatório Histórico
Responsabilidades: Gerente de Projeto, Usuário
Ferramentas: Word

Nome: Avaliar o Instrumento Virtual
Descrição: Avaliar o projeto, código, testes, resultados dos testes e a documentação do usuário considerando os seguintes critérios: cobertura de teste dos requisitos do item de software, conformidade com os resultados esperados, viabilidade da integração e testes do sistema, se conduzidos, viabilidade da operação e manutenção.
Artefato: Instrumento Virtual aprovado, Laudo de Preparação Individual para Avaliação do Instrumento Virtual, Laudo Final da Avaliação do Instrumento Virtual, Relatório Histórico, Plano de Ação
Responsabilidades: Gerente de Projeto, Usuário
Ferramentas: Word

Nome: Elaborar versão preliminar da documentação do usuário *Descrição*: Desenvolver e documentar versões preliminares da documentação do usuário. *Artefato:* Manual do Usuário, Relatório Histórico *Responsabilidades:* Analista de Sistemas *Ferramentas de Software:* Word
Nome: Preparar o produto de software a ser entregue *Descrição*: Atualizar e preparar o produto de software a ser entregue para a integração do sistema, teste de qualificação do sistema, instalação do Instrumento Virtual ou apoio à aceitação do Instrumento Virtual, quando aplicável. Estabelecer uma linha básica (*baseline*) para o projeto e código do item de software. *Artefatos:* Instrumento Virtual pronto, Relatório Histórico *Responsabilidades:* analistas *Ferramentas:* Word

Integração do sistema
Descrição: Realizar a integração do sistema com os procedimentos manuais, hardware e outros sistemas.
Artefato:. Relatório de Testes, Relatório Histórico, Laudo de Preparação Individual para Avaliação do Sistema Integrado, Laudo Final da Avaliação do Sistema Integrado, plano de ação.
Sub-atividades
Nome: Integrar sistema *Descrição*: Integrar os itens de configuração de software com hardware, operações manuais e outros sistemas e testar as agregações de acordo com seus requisitos. A integração e os resultados do testes devem ser testados. *Artefatos:* Sistema integrado pronto para o teste de qualificação do sistema, Relatório de Testes, Relatório Histórico *Responsabilidades*: gerente do projeto, outros fornecedores quando aplicável *Ferramentas:* Word
Nome: Planejar testes de qualificação do sistema *Descrição*: Desenvolver e documentar um conjunto de testes, casos de teste e procedimentos de testes para conduzir o teste de qualificação do sistema. *Artefato*: Plano de Teste de qualificação do Sistema, Relatório Histórico *Responsabilidades:* Gerente de Projeto, usuários, outros fornecedores quando aplicável *Ferramentas de Software:* Word

Nome: Avaliar o Sistema Integrado
Descrição: Avaliar o sistema integrado segundo os seguintes critérios: Cobertura dos testes dos requisitos do sistema, adequação dos métodos e padrões de teste utilizados, conformidade com os resultados esperados, viabilidade do teste de qualificação do sistema, viabilidade da operação e manutenção. O desenvolvedor deve garantir que o sistema integrado está pronto para o teste de qualificação do sistema.
Artefato: Sistema integrado aprovado, Laudo de Preparação Individual para Avaliação do Sistema Integrado, Laudo Final da Avaliação do Sistema Integrado, Relatório Histórico. Plano de Ação
Responsabilidades: Gerente de Projeto,
Ferramentas de Software: Word

Teste de qualificação do sistema

Descrição: Teste do sistema visando evidenciar que o sistema está pronto para ser entregue.

Artefato: Relatórios de teste, Relatório Histórico

Sub-atividades

Nome: Realizar homologação interna
Descrição: Realizar o teste do sistema de forma a garantir que a especificação de cada requisito seja testada, para conformidade, e que o sistema está pronto para ser entregue. Os resultados do teste devem ser documentados.
Artefatos: Sistema com homologação interna, Relatórios de teste, Relatório Histórico
Responsabilidades: Gerente de Projeto, usuário, outros fornecedores quando aplicável
Ferramentas de Software: Word

Nome: Realizar homologação externa
Descrição: Após os testes de homologação interna, o sistema é encaminhado ao cliente para a realização do teste para homologação necessário para a avaliação do serviço executado.. Os resultados do teste devem ser documentados.
Artefatos. Sistema homologado, Relatórios de teste, Relatório Histórico
Responsabilidades: Gerente de Projeto, usuário, outros fornecedores quando aplicável
Ferramentas de Software: Word

Instalação do Instrumento Virtual

Descrição: Planejar e instalar o sistema no ambiente do usuário.

Artefato: Plano de Instalação, Manual do Sistema, Manual do Usuário, Relatório Histórico.

Sub-atividades

Nome: Planejar a instalação do sistema
Descrição: Desenvolver um plano para instalar o produto no ambiente alvo definindo recursos e informações necessárias.
Artefato: Plano de Instalação, Relatório Histórico
Responsabilidades: Gerente de Projeto
Ferramenta: word

Nome: Completar a documentação do sistema *Descrição*: Finalizar a documentação do sistema (se um Case foi utilizado de maneira adequada, estes manuais constituem o conteúdo de seu repositório). *Artefato*: Manual do Sistema *Responsabilidade*: analistas *Ferramentas*: Word, Rose ou System Architect; Navegue.
Nome: Completar a documentação do usuário *Descrição*: Completar a documentação do usuário, se necessário *Artefato:* Manual do Usuário, Relatório Histórico *Responsabilidade:* analistas de sistemas *Ferramentas:* Word;
Nome: Efetuar a carga inicial de dados Descrição: Preparar os dados de entrada para tornar possível a inicialização do sistema. *Artefato*: sistema com dados iniciais *Responsabilidade*: analistas *Ferramentas*:
Nome: Instalar o sistema *Descrição*: Instalar o produto de acordo com o plano de instalação. *Artefato:* Software instalado, Relatório Histórico *Responsabilidades:* analistas *Ferramenta: Word*

Apoio à aceitação do Instrumento Virtual
Descrição: Avaliar o sistema em uso, treinar o usuário e obter a aceitação do sistema.
Artefato: Relatório de Testes, Relatório Histórico.
Sub-atividades
Nome: Apoiar a avaliação do usuário *Descrição*: Realizar os testes com os usuários para aceitação final do sistema. *Artefato*: sistema aceito, Relatório de Testes, Relatório Histórico *Responsabilidades:* analistas, Usuário *Ferramenta: Word*
Nome: Realizar entrega do sistema *Descrição*: Entrega oficial do sistema segundo o contrato estabelecido. *Artefato*: sistema entregue, Relatório Histórico *Responsabilidades:* Gerente de Projeto *Ferramenta: Word*
Nome: Realizar treinamento *Descrição*: Realizar treinamento inicial e contínuo dos usuários. Artefatos: Relatório Histórico *Responsabilidades:* analistas *Ferramenta: Word*

Encerramento do projeto
Descrição: Realizar as atividades de encerramento do projeto.

Artefato: Relatório Avaliação Post Mortem.
Sub-atividades
Nome: Realizar Avaliação post-mortem. *Descrição*: Esta atividade tem como objetivo realizar uma avaliação post-mortem ao final do projeto e documentar seus resultados *Artefato:* Relatório Avaliação Post Mortem. Responsável: gerente do projeto

5.4 MDM ADS, um Ambiente de Desenvolvimento de Software para Instrumentação Virtual.

Uma vez definido o processo de software e realizada as adequações necessárias a Estação TABA, tem-se todos os requisitos para a configuração de um ambiente, permitindo que uma empresa o utilize no desenvolvimento de instrumentação virtual.

A ferramenta *Config* (VILLELA, 2004) apóia a configuração do ambiente baseando-se nas atividades listadas na tabela 4.1. Entre as diversas questões que alimentam o processo de configuração, destaca-se nos próximos parágrafos, como um exemplo de como ocorre o processo de configuração, algumas entradas pertinentes às atividades de *Caracterizar a Organização*, *Identificar Cultura Organizacional na Área de Software* e *Identificar Objetivos da Configuração do Ambiente.*

A empresa selecionada é a M&D Monitoração e Diagnose Ltda. A mesma busca atender as necessidades tecnológicas da indústria quanto à modernização e automação dos sistemas de produção. Esta empresa aplica instrumentação virtual principalmente no desenvolvimento de sistemas inteligentes de apoio à decisão em manutenção preditiva de equipamentos considerados críticos em um processo produtivo. Como exemplo, o sistema MDM, que constitui-se num sistema especialista fuzzy para o diagnóstico de falhas em hidrogeradores acrescido de um conjunto de ferramentas de análise de vibração para dados oriundos de sistemas de monitoramento e supervisão dos equipamentos.

A M&D é uma microempresa (até 9 funcionários), que desenvolve projetos médios (<17000 homem/hora) de média complexidade no setor de serviços de engenharia e pesquisa. Atua nos domínios de automação industrial, geração elétrica, análise de vibrações

mecânicas entre outros. Desenvolve tanto pacotes pra comercialização quanto software sob encomenda. Desenvolve com Orientação a Objetos e especificação em UML, além de sistemas baseados em conhecimento. A M&D pretende, com a configuração do ambiente, melhorar suas estimativas e controle do desenvolvimento, para torná-lo disciplinado e repetitível.

A figura 5.3 ilustra uma das telas envolvidas no processo de caracterização da organização.

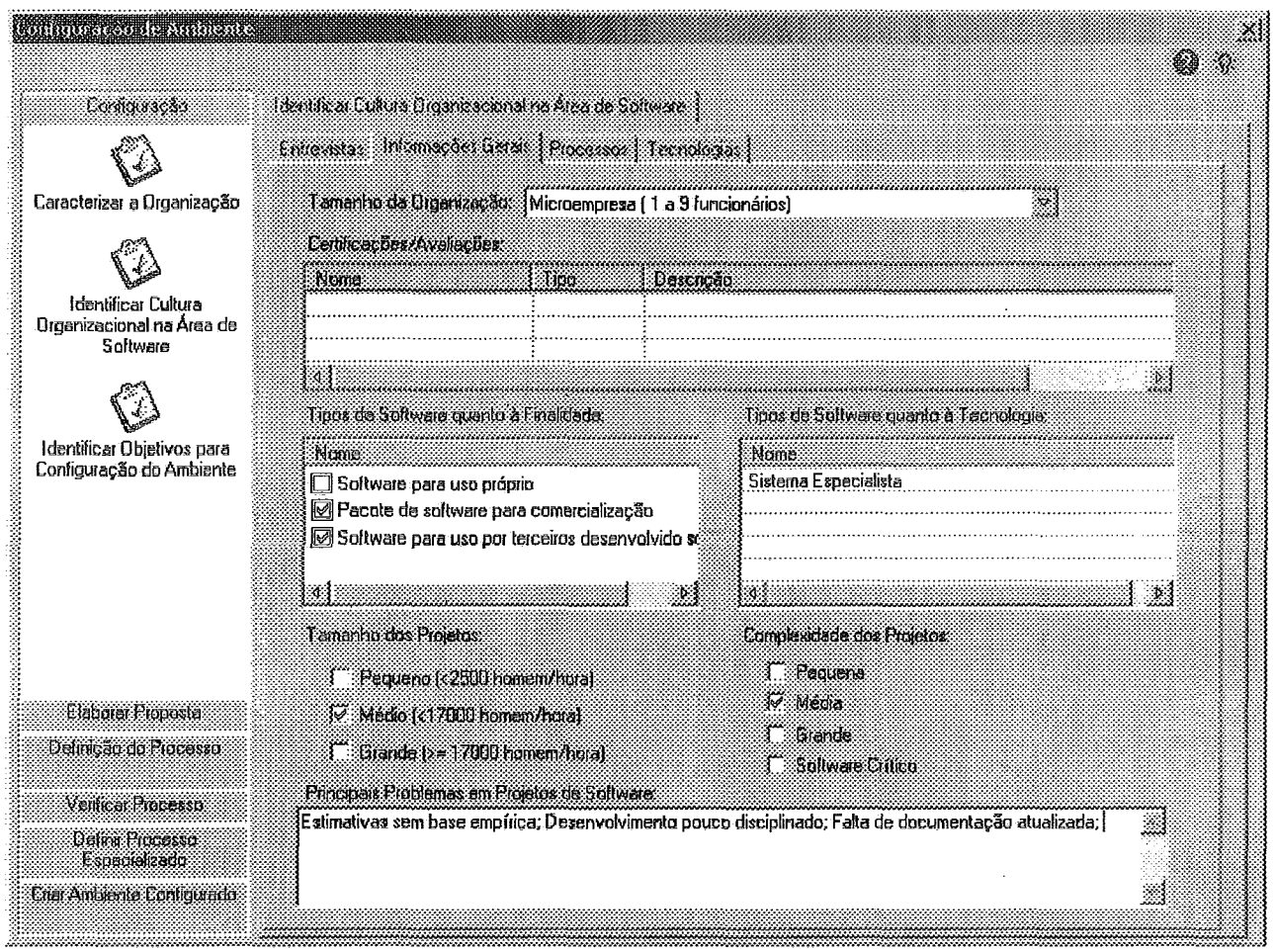


Figura 5.3 – Ferramenta *Config:* etapa de configuração do ADS MDM

As atividades da ferramenta *Config: Definição do Processo* e *Definição do Processo Especializado,* foram discutidas na seção 5.2 e são responsáveis pela definição do processo apresentado na seção 5.3, que é o mesmo utilizado pelo processo de configuração para a M&D.

Maior detalhamento sobre as atividades apresentadas aqui e demais envolvidas no processo de configuração pode ser encontrado em VILELLA (2004).

Após estas atividades surge como resultado o ambiente configurado MDM ADS, que é apresentado na figura 5.4

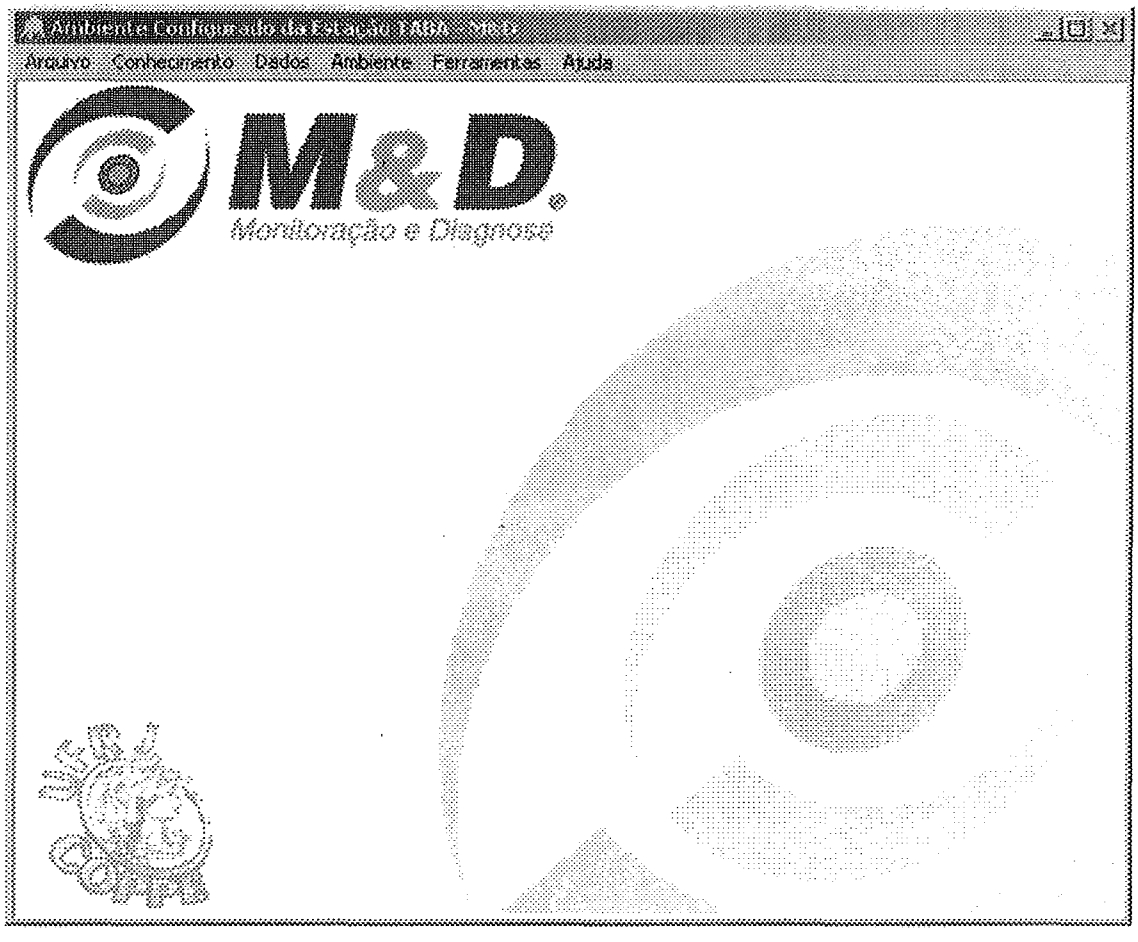


Figura 5.4 – MDM ADS

Entre as ferramentas existentes no ambiente configurado, o Sapiens (SANTOS, 2003) é uma das ferramentas utilizadas para descrição do conhecimento organizacional. O *Sapiens* permite a representação da estrutura organizacional com as habilidades, conhecimentos e experiências requeridos ao longo dessa estrutura, além de permitir a alocação de pessoal, incluindo os conhecimentos, habilidades e experiências que cada profissional possui, contém, ainda, mecanismos para busca e navegação. Desta forma, no ambiente configurado é possível visualizar o conhecimento organizacional da empresa. As figuras 5.5 e 5.6 ilustram a ferramenta *Sapiens*, executada a partir do MDM ADS, representando a estrutura organizacional da empresa M&D.

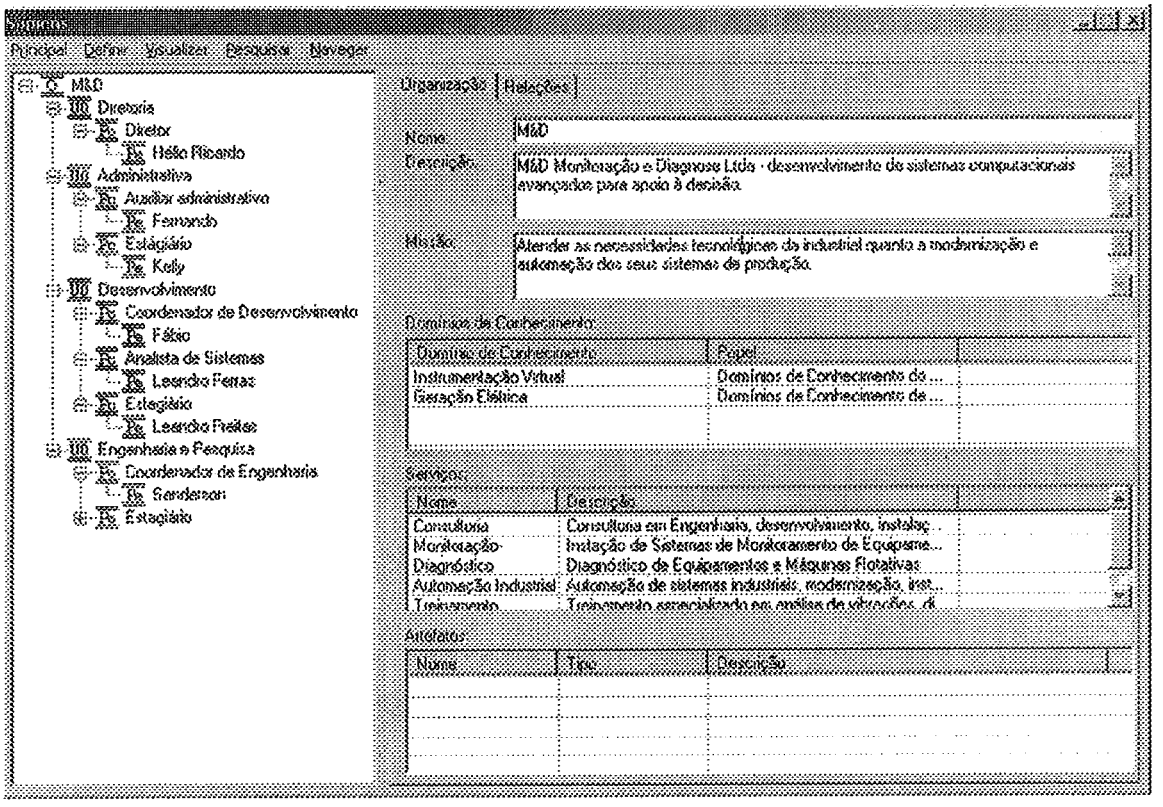


Figura 5.5 – Ferramenta *Sapiens:* representação da estrutura organizacional da M&D

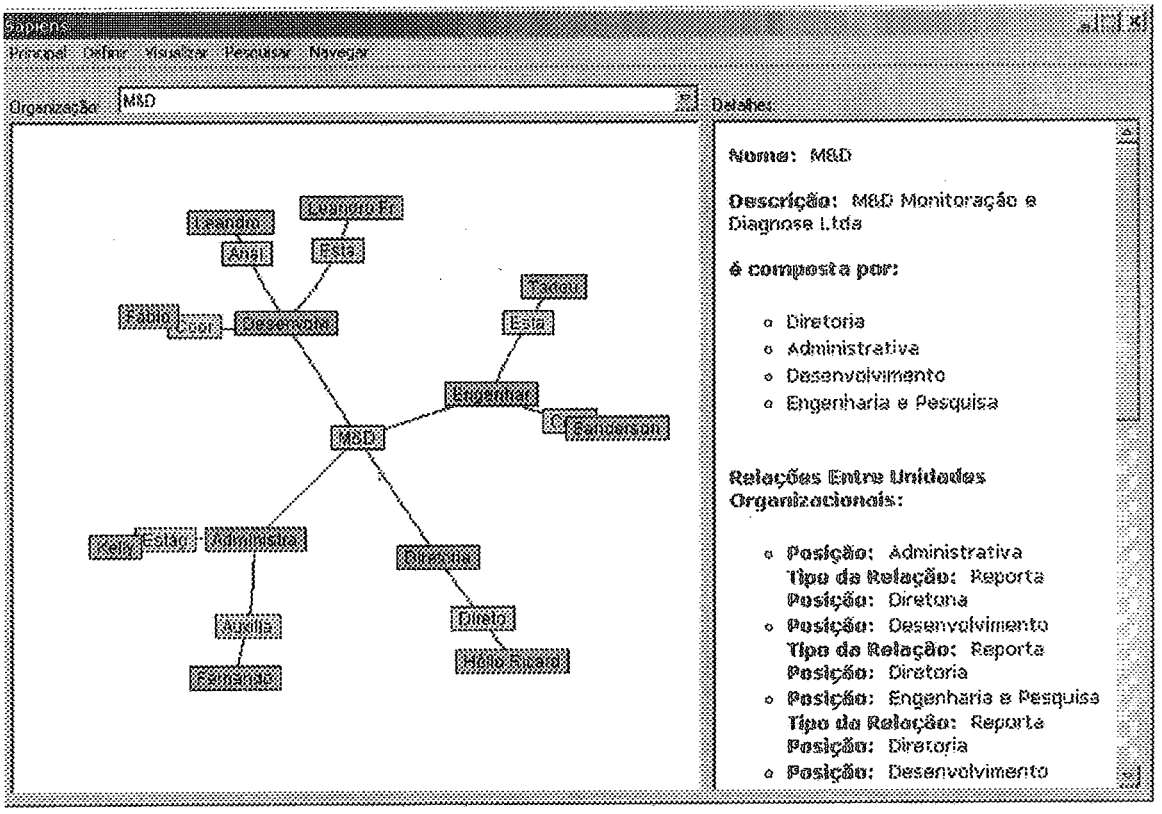


Figura 5.6 – Ferramenta *Sapiens:* representação gráfica da estrutura organizacional da M&D

5.4.1 ADSOrg LASEEE, um Ambiente Instanciado para um Projeto

Com o objetivo de avaliar a abordagem proposta (ambiente, processos, ferramentas criadas e alteradas), foi instanciado a partir do MDM ADS um ADSOrg para um projeto específico, denominado projeto LASEEE. O projeto constitui-se no desenvolvimento sob encomenda de um sistema de instrumentação virtual para automação do Laboratório de Acionamento e Segurança em Equipamentos Eletro-Eletrônico do Centro de Pesquisas de Energia Elétrica CEPEL/ELETROBRAS. O sistema, que será desenvolvido na linguagem *Labview* 6.1, visa a automação de todas as rotinas de ensaios de motores elétricos e transformadores do laboratório. Esta automação inclui desenvolvimento das rotinas de aquisição dos dados monitorados, controle dos equipamentos, gerenciamento dos recursos, controle de acesso entre outros. O projeto tem duração de seis meses e a equipe de desenvolvimento é composta por oito profissionais, sendo alguns deles de dedicação parcial.

Foi definido como modelo de ciclo de vida do projeto o relacionado ao RUP (KRUCHTEN, 1999) e definido por BERGER (2003) como principal modelo de ciclo de vida para ao paradigma Orientado a Objeto dentro do contexto da ferramenta *AdaptPro*. O modelo RUP é interativo e incremental e divide o ciclo de desenvolvimento em quatro grandes fases: Concepção, Elaboração, Construção e Transição.

Para o projeto em questão a fase de Concepção é responsável pelo planejamento do projeto, definição e especificação dos requisitos do sistema, definição da arquitetura e outras atividades referentes ao *startup* do projeto. A fase de Elaboração é responsável pela implementação dos principais módulos da arquitetura, dos módulos de acesso ao sistema e aos recursos compartilhados. A Fase de Construção é dividida em quatro iterações, cada uma responsável pela *release* de um conjunto de instrumentos virtuais responsáveis pela execução de um conjunto de ensaios de equipamentos.

A figura 5.7 ilustra o ADSOrg LASEEE, onde, à esquerda, encontra-se a árvore do processo instanciado com as atividades agrupadas pelas fases e iterações. A figura 5.8 ilustra o módulo WEB da Estação TABA, que permite que desenvolvedores registrem o tempo gasto em cada atividade do processo para acompanhamento gerencial do andamento do projeto.

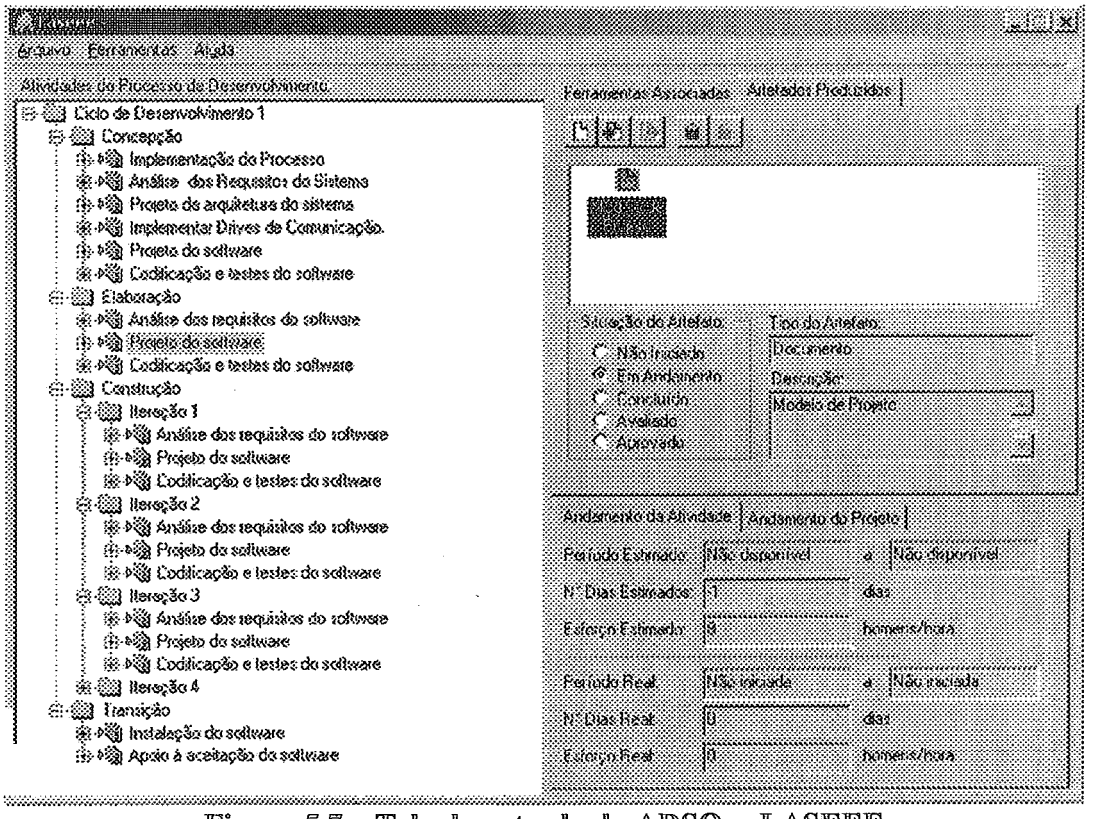


Figura 5.7 – Tela de entrada do ADSOrg LASEEE

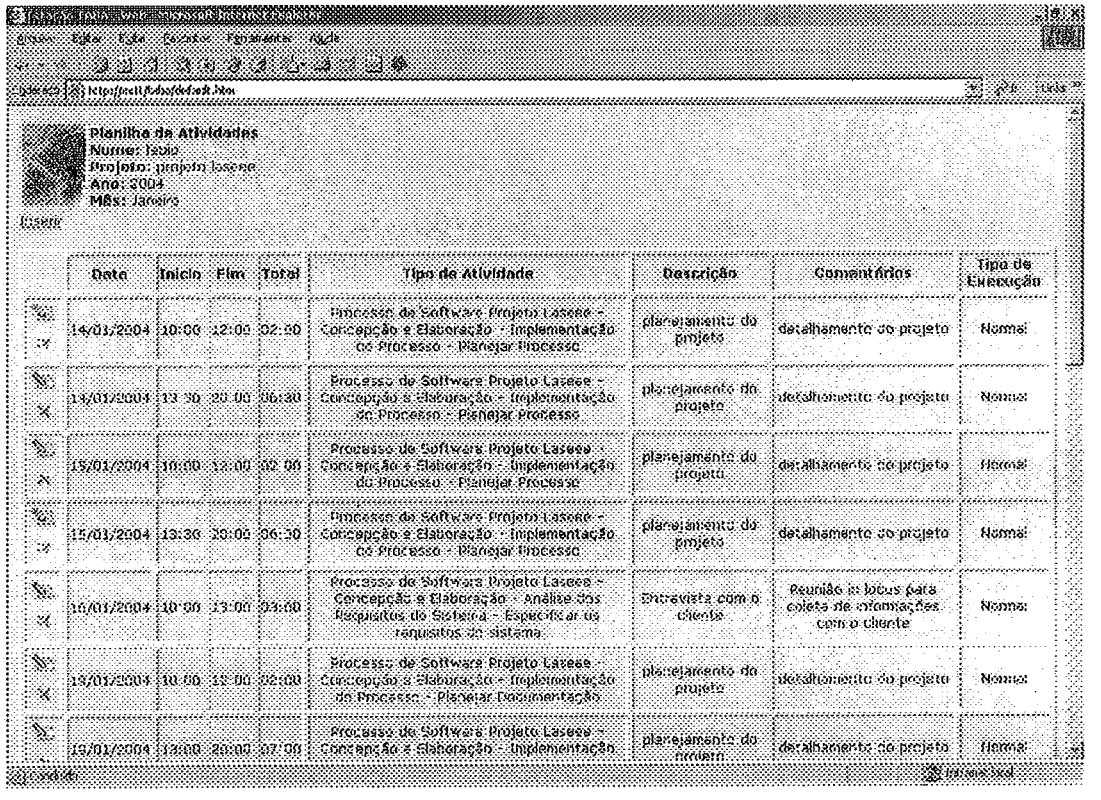


Figura 5.8 – Módulo Web do ADSOrg LASEEE

5.5 Considerações Finais

Discutiu-se no presente capítulo o ambiente configurado MDM ADS e sua instanciação para aplicação no projeto LASEEE no contexto da empresa M&D Monitoração e Diagnose Ltda. Para configurar tal ambiente foi necessário antes rever os conceitos de hierarquia de processos (processo padrão, especializado e instanciado), para compreensão de como a Estação TABA permite a definição de diferentes níveis de abstração de processos e como esta hieraquização é útil para uma organização com diferentes linhas de desenvolvimento.

Um processo especializado foi definido para desenvolvimento de instrumentação virtual. Este processo e outras informações necessárias, como as características da organização, serviram de entrada para a ferramenta *Config* que apoiou o processo de configuração do MDM ADS, utilizado pela empresa M&D. Finalmente, através do MDM ADS, um ADSOrg foi instanciado para um projeto apresentado, o projeto LASEEE.

O próximo capítulo conclui o presente trabalho e discute as contribuições desta proposta, perspectivas futuras, possíveis melhorias e extensões deste trabalho.

CAPÍTULO VI
CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as contribuições, principais conclusões e perspectivas futuras para continuidade deste trabalho.

Introdução

Este trabalho apresenta uma abordagem para o desenvolvimento de instrumentação virtual baseada na Estação TABA, que acredita-se contribuir para a melhoria do desenvolvimento deste tipo peculiar de software. Para construir uma fundamentação teórica sob os argumentos aqui explanados, foi realizada uma revisão da literatura sobre instrumentação virtual, Estação TABA, Ambientes de Desenvolvimento de Software Orientados a Organização, documentação e processos de software. Ainda para resguardar os argumentos, foi realizada uma pesquisa sobre engenharia de software e instrumentação virtual. O capítulo IV apresentou todas as modificações da Estação TABA consideradas importantes para que o Ambiente de Desenvolvimento de Instrumentação Virtual apresentado no capítulo V fossem de aplicação efetiva. Finalmente, este capítulo apresenta as considerações finais e perspectivas futuras.

6.1 Conclusão e Contribuições

Este trabalho teve como objetivo propor possíveis caminhos para se amenizar os problemas no desenvolvimento de instrumentação virtual para empresas que utilizem este tipo de software, definindo uma abordagem de engenharia para o desenvolvimento destes sistemas, buscando tornar tal desenvolvimento disciplinado e repetitível.

A solução proposta para o problema foi a configuração de um ambiente de desenvolvimento de software orientado a organização que embora seja, a princípio, voltado somente para uma organização, também atende uma classe de organizações: Organizações desenvolvedoras de instrumentação virtual.

No entanto, para que o ambiente atendesse o desenvolvimento de instrumentos virtuais, foram levantadas algumas premissas consideradas como possíveis de influenciar

no sucesso da aplicação da solução. Estas premissas foram responsáveis pela inclusão de um novo requisito na Estação TABA: documentação.

Um processo de documentação para a Estação TABA foi definido baseado na norma ISO/IEC 12207 (NBR ISO/IEC 12207, 1997) e nas particularidades da Estação TABA. Este processo ocorre através dos diversos níveis de abstração da Estação TABA e interage com processos existentes, como por exemplo, os processos de configuração e instanciação de ambientes. Atividades e seus respectivos suportes computacionais foram inseridos nos diferentes processos e níveis de abstração da Estação TABA.

Uma ferramenta para apoio ao planejamento de documentação foi desenvolvida e implementada, a *DocPlan*. Esta ferramenta produz um plano de documentação condizente com o sugerido pela norma ISO/IEC 12207 (NBR ISO/IEC 12207, 1997).

Um processo de software baseado nos requisitos da norma ISO/IEC 12207 (NBR ISO/IEC 12207, 1997) e nas particularidades da instrumentação virtual foi definido para ser o processo base do ambiente de desenvolvimento de instrumentação virtual. Um ambiente foi então configurado para uma organização de desenvolvimento de instrumentação virtual, a M&D, originando o ambiente MDM ADS. Para testar o ambiente e o processo de desenvolvimento foi instanciado um ADSOrg para um projeto desta empresa, o ADSOrg LASEEE.

Dentre as contribuições deste trabalho destaca-se:

- Uma proposta de abordagem para o desenvolvimento de instrumentação virtual com suporte computacional em seu escopo, que acredita-se efetiva em seu objetivo de disciplinar tal processo.
- Uma pesquisa que fornece indícios do panorama da aplicação de engenharia de software no desenvolvimento e manutenção de instrumentação virtual
- Um processo de documentação para a Estação TABA, que atende qualquer tipo de software e não apenas instrumentação virtual.
- A implementação da estrutura computacional de apoio ao processo de documentação nos vários níveis de abstração da Estação TABA e uma ferramenta para o planejamento da documentação, a *DocPlan*, ambas as contribuições indiferentes ao tipo de software sendo desenvolvido.
- Um processo de desenvolvimento de instrumentação virtual.

- Um ambiente configurado para uma organização de desenvolvimento de instrumentação virtual que, embora específico para uma organização, possui sua linha base aplicável a qualquer outra organização, o mesmo ocorrendo com o ADSOrg instanciado.

6.2 Perspectivas Futuras

A avaliação desta tese, embora já iniciada com a instanciação para o projeto LASEEE, transcende o tempo de uma tese de mestrado, pois necessita a aplicação em diferentes projetos e organizações. Pretende-se, na continuação, deste trabalho a instanciação de ADSOrgs para outros projetos da M&D, bem como a configuração de ambientes para outras empresas de desenvolvimento baseado em instrumentação virtual.

Pretende-se utilizar as informações provenientes do uso prático do MDM ADS e do ADSOrg LASEEE para aprimoramento deste trabalho, onde se inclui os processos definidos (documentação e desenvolvimento), ambientes (configurado e instanciado) e ferramentas criadas ou existentes.

Outra fonte de alimentação de melhorias para o processo de documentação e suas ferramentas, que são de aplicação para qualquer tipo de software, é a configuração de ambientes de desenvolvimento para empresas associadas a RioSoft, agente Softex sediado ao Rio de Janeiro-RJ, através de convênio firmado entre esta e a COPPE/UFRJ. O retorno sobre o uso dos produtos deste trabalho por estas empresas contribuirá para o aprimoramento do mesmo.

Acredita-se que a definição do domínio de instrumentação virtual, através da modelagem de ontologias, possa trazer significativas contribuições para o MDM ADS, portanto, também sendo considerada uma perspectiva de trabalho futuro.

ABECKER, A., BERNARDI, A., HINKELMANN, K. *et al.*, 1998, "Toward a Technology for Organizational Memories", *IEEE Intelligent Systems*, v. 13, n. 3 (May/June), pp. 40-48.

ABRAN, A., MOORE, J., 2001, *Guide to the Software Engineering Body of Knowledge*, IEEE Computer Society.

ALBRECHT, A. J., 1979, "Measuring Application Development Productivity",In: *Proceedings of IBM Application Development Symposium*, Monterey, CA, outubro, pp-83-92.

ANSI/ANS 10.3-1995, 1995, *American National Standards Insitute and American Nuclear Society.*

ANTONIOL, G., FIUTEM, R., LUTERRI, G., TONELLA, P.,ZANFEI, S., MERLO, E., 1997, "Program understanding and maintenance with CANTO environment,", In: *Proceedings of International Conference on Software Maintenance*, pp. 72-81, Bari, Italia, Outubro.

BARCELLOS, M., P., 2003, *Planejamento de Custos em Ambientes de Desenvolvimento de Software Orientados à Organização,* Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil

BARONE, L., ROUSSEAU, B., 1996, "Software documentation on the Web: The Light project (Life Cycle Global Hypertext)", In: *Proceedings of IX Conference on Software engeneering and its applications*, pp 119-120, Paris, França.

BARRERA, B., LEE., E., A., 1991, "Multirate signal processing in Comdisco's SPW", In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1113-1116, Maio.

BECK, K., 2000, *Extreme Programming Explained*, Addison-Wesley.

BERGER P. M., 2003, "Instanciação de Processos de Software em Ambientes Configurados na Estação Taba", Tese de MSc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, julho 2003.

BOEHM, W. B., ABTS, C., BROWN, A. W., CHULANI, S., CLARK, B. K., HOROWITZ, E., MADACHY, R., REIFER, D., STEECE, B. *Software Cost Estimation with COCOMO II,* Prentice Hall, 2000.

BRAGG, S., DRISKILL, C., 1994 , "Diagrammatic Graphical Programming Languages and DoD-STD-2167A", In: *Proceedings of the 30th IEEE InternationalConference on Automatic Test: AUTOTESTCON*, pp. 216-218.

BRIAND L., C., 2003, "Software Documentation: How Much is Enough?", In: *Proceedings of the Seventh European Conference On Software Maintenance and Reengineering,* pp. 320-322.

BRIGNELL J., TANER, A., 1997 " Virtual instrumentation and intelligent sensors," Sens. Actuators, Junho 1997, vol. 61, no. 1, pp. 427-430.

CHIUEH, T., WU, W., LAM, L., 2000, "VARIORUM, A Multimedia-Based Program Documentation System", In: *Proceedings of I International Conference on Multimedia and Expo,* pp. 155-158,

CHRISSIS, M.. B., KONRAD, M., SHRUM, S., 2003, CMMi – *Guidelines for Process Integration and Product Improvement*, Addison Wesley.

COOK, C., VISCONTI, M., 1996, "New and improved documentation process model", In: *Proceedings of de 14th Pacific Northwest software quality conference,* Portland, Oregon, EUA, pp. 364-380, Outubro.

COOK, C., VISCONTI, M., 2002, "An Overview of industrial Software Documentantion Practice", In: *Proceedings of de XXII International Conference of the Chilean Computer Science Society* (SCCC'02), pp. 130- 138.

CRUZ, C. D., 1998, *Em Direção a um Modelo de Custos de Desenvolvimento de Software Orientado a Objetos*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

CYBULSKY, J. L., REED, K., 1992, "A Hypertext based software-engeneering environment" *IEEE Software*, vol 9, no 2, pp 62-68, Março.

DELANGHE, S., 2000, "Using Learning Styles in Software Documentation", *IEEE Transactions on Professional Communication*, vol. 43, no. 2, pp. 201-205, Junho.

EMAN, K. E., DROUIN, J., MELO, W., 1998, "SPICE – The Theory and Practice of Software Process Improvement and Capability Determination", *IEEE Computer Society Press*.

FALBO, R. A., 1998, *Integração de Conhecimento em um Ambiente de Desenvolvimento*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FALBO, R. A. , MENEZES, C., ROCHA, A. R., 1999, "Assist-Pro: Um Assistente Inteligente para Apoiar a Definição de Processos de *Software*", In: *Anais do XIII Simpósio Brasileiro de Engenharia de Software*, pp. 147-162, Florianópolis, Brasil, Outubro.

FARIAS, L., 2002, *Planejamento de Riscos em Ambientes de Desenvolvimento de Software Orientados a Organização*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FARIAS, L., ROCHA, A.R., TRAVASSOS, G.H., 2001, "Producing Project Risk Plans in Enterprise-Oriented *Software* Development Environments", In: *14th International*

Conference Software & Systems Engineering and their Applications, Paris, December.

FIGUEIREDO, S., M., 2004, *Gerência de Configuração em Ambientes de Desenvolvimento de Software Orientados a Organização,* Projeto Final de Curso, IM/UFRJ, Rio de Janeiro, RJ, Brasil.

FISCHER, G, 1994; "Domain-Oriented Design Environments", *Automated Software Engineering - The International Journal of Automated Reasoning and Artificial Intelligence in Software Engineering*, Vol 1, no. 2, junho, pp 177-203.

FISCHER, G., 1996, "Seeding, Evolutionary and Reseeding: capturing and evolving knowledge in domain-oriented design environments", In: Sutcliffe, A., Benyon, B. van Assche (eds) - IFIP 8. 1/13. *Joint Working Conference – Domain-Knowledge for Interactive System Design*, pp 1-16, Geneva, Switzerland, May.

FOURO, A. M., 2002, *Apoio à Construção de Base de Dados de Pesquisa em Ambientes de Desenvolvimento de Software Orientados a Domínio*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FRENCH, J., C., KNIGHT, J., C., POWELL, A., L., 1997, "Applying Hypertext structures to software documentation", *Information Processing & Management*, vol 33, no. 2. pp. 219-231, Março.

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

GALOTTA, C., 2000, *Netuno: Um Ambiente de Desenvolvimento de Software Orientado ao Domínio de Acústica Submarinha*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

GARG, P., JAZAYERI, M., 1995, "Introduction". In: Garg, P., Jazayeri, M. (eds), *Process-Centered Software Engineering Environments*, chapter 1, IEEE Computer Society Press.

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

GOLDBERG, H., 2000, "What is Virtual Instrumentation?" *IEEE Instrumentation & Measurement Magazine*, dezembro 2000, pp. 10-13.

HELSEL, R., 1997 *Visual Programming for HP-VEE*, Second Edition, Prentice Hall Professional Technical Reference, Fevereiro 1997.

HENNINGER, S., 2001, "Turning Development Standards into Repositories of Experiences", *Software Process Improvement and Practice,* no. 6, pp. 141-155.

HILS, D., D., 1992 "Visual languages and computing survey: Data flow visual Programming languages", *Journal of Visual Languages and Computing*, pp. 69-101.

ISO/IEC TR 15504, 1998, *Parts 1-9: Information Technology - Software Process Assessment.*

ISO/IEC 61131-3, 1993, *Programmable languages PLC software structure, languages and program execution.*

JONES C., 1986, "A Short History of Functions Points and Feature Points", *Software Productivity Research*, Inc., Burlington, MA, Junho.

JONES, C., 1999, "Software Sizing", *IEEE Review*, vol. 45, Issue: 4 , Julho, pp. 165-167.

KODOSKY, J., MACCRISKEN, J., RYMAR, G., 1991, "Visual programming using structured data flow". In : *Proceedings of IEEE Workshop on Visual Languages*, pp. 34-39, Kobe, Japão, Outubro.

KOLB, D., A., 1984, *Experiential Learning: Experience as the Source of Learning and Development*, Englewood Cliffs, NJ: Prentice-Hall.

KONSTANTINIDES, K., RASURE, J., 1994, "The Khoros software development environment for image and signal processing" *IEEE Transactions on Image Processing*, vol. 3, no. 3, pp. 243-252, Maio.

KRUCHTEN, P., 1999, *The Rational Unified Process – An Introduction*, Addison-Wesley.

KYLMÄKOSKI., R, 2003, "Efficient Authoring of Software Documentantion Using RaPID7", In: *Proceedings of the 25th International Conference on Software Engineering*, pp. 231- 240.

LAUWEREINS, R., WAUTERS, P., ADE, M., PEPERSTRAETE J., A., 1994, "Geometric parallelism and cyclo-static data flow in GRAPE-II", In: *5th Intl Workshop on Rapid System Prototyping*, Grenoble, França, Junho.

LEE, E., A., MESSERSCHMITT, D., G., *et al.* 1994, *An overview of the Ptolemy project*, Anonymous ftp from ptolemy.eecs.berkeley.edu, Março.

LEE, J., KIM, Y., YU, S., 2001, "Stage Model for Knowledge Management", In:*Proceedings of the 34th Hawaii International Conference on system Sciences*, Maui, Hawaii.

LETHBRIDGE, T., C., SINGER J, FORWARD, A., 2003 "How Software Engineers Use Documentation: The State of the Practice" *IEEE Software,* novembro/dezembro, pp. 35-38.

MACHADO, L.F.C., 2000, *Modelo para Definição, Especialização e Instanciação de Processos de software na Estação TABA*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MACHADO, L.F., SANTOS, G., OLIVEIRA, K.M., ROCHA, A.R., 2000a, "Def-Pro: uma Ferramenta para Apoiar a Definição de Processo Padrão", *XI Conferencia Internacional de Tecnologia de Software*, pp. 165-179, Curitiba, PR, junho.

MACHADO, L.F., SANTOS, G., OLIVEIRA, K.M., ROCHA, A.R., 2000b, "Def-Pro: Apoio automatizado para Definição de Processos de *Software*", *Caderno de Ferramentas - XIV Simpósio Brasileiro de Engenharia de Software*, pp. 359-362, João Pessoa, PB, outubro.

MAIDANTCHIK, C., 1999, *Modelo de Processo de Gerência de Software para Equipes Geograficamente Dispersas*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MAURER, F., DELLEN, B., HOLZ, H., 1999, Process Support for Virtual Software Organizations, First International Workshop on Learning Software Organizations (LSO 1999), pp.87-98.

MCCABE, T., J., 1976, "A Software Complexity Measure", *IEEE Trans. Software Engineering*, vol. 2 no. 6, dezembro, pp. 308-320.

MCCONNELL, S., 1996, *Rapid Development*, Microsoft Press.

MONTONI, M., 2003, "Aquisição de Conhecimento: Uma Aplicação no Processo de Desenvolvimento de Software.", Tese de MSc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, agosto 2003

MOURA, L. M. V., ROCHA, A. R. C., 1992, *Ambientes de Desenvolvimento de Software*, Publicações Técnicas COPPE/UFRJ, ES-271/92, Rio de Janeiro, Brasil.

NAJORK, M., A., GOLIN, E., 1990, "Enhancing Show-and-Tell with a polymorphic type system and higher-order functions" In: *Proceedings IEEE Workshop on Visual Languages*, pp. 215-220, Skokie, Illinois, EUA, Outubro.

NATIONAL INSTRUMENTS, 2003, *Graphical Object-Oriented Programming*, Online: http://www.ni.com/goop e http://www.endevo.se (verificado em março/2004)

NATIONAL INSTRUMENTS CORP., 2000, *Instrument Reference and Catalog.*

NBR ISO/IEC 12207:1997. Tecnologia da informação – processos de ciclo de vida de software.

ODDO, M., ROCHA, A. R., 2003, "Práticas Relevantes em Engenharia de Software: uma avaliação de especialistas", *II Simpósio Brasileiro de Qualidade de Software,* Ceará, Brasil, setembro 2003.

O'LEARY, D. E., 1998, "Using AI in Knowledge Management: Knowledge Bases and Ontologies", *IEEE Intelligent Systems*, v. 13, n. 3 (May/Jun), pp. 34-39.

OLIVEIRA, K. M., 1999, *Modelo para Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

OLIVEIRA, K. M., ROCHA, A. R., TRAVASSOS, G. H., 1999a, "A Domain-Oriented *Software* Development Environment for Cardiology", In: *Proceedings of America Medical Informatics Association conference – AMIA*, Washington, D.C., Nov. (In press)

OLIVEIRA, K.M, XIMENES, A., MATWIN, S., TRAVASSOS, G., ROCHA, A., 2000, "A Generic Architecture for Knowledge Acquisition Tools in Cardiology"; *5th Intelligent Data Analysis in Medicine and Pharmacology - Workshop at the 14th European Conference on Artificial Intelligence*, pp. 43-45, Berlin, Alemanha, Agosto.

PAULK, M. C., WEBER, C. V., CURTIS, B., CHRISSIS, M. B. (eds), 1997, *The Capability Maturity Model: Guidelines for Improving the Software Process*. Camegie Mellon University, Software Engineering Institute, Addison-Wesley Longman Inc.

PENCE, J., HON, S., 1993, "Building software quality into telecommunications network systems" *Quality Progres*. Outubro, pp. 95-97.

PFLEEGER, S. L., 2001, *Software Engineering: Theory and Practice*, 2nd ed. Upper Saddle River, NJ: Prentice Hall.

PHOHA, V., 1997, "A standard fo software documentation", *IEEE Computer*, outubro 1997.

PITTMAN, D., MILLER, J., 1997, "Software Metrics for Non-Textual Programming Languages" In: *Proceedings of the 33th IEEE InternationalConference on Automatic Test: AUTOTESTCON*, pp. 198-203.

RABE, D., MILLER, J., 1997, "Applyng Software Process to virtual Instrument based Test Program Set Development" In: *Proceedings of the 33th IEEE InternationalConference on Automatic Test: AUTOTESTCON*, pp. 94-97.

RAMAN, S., 2000, "It Is Software Process, Stupid: Next Millennium Software Quality Key", *IEEE AES Software Magazine*, Junho, pp. 33-37.

RASURE, J., YOUNG, M., 1992, "Dataflow visual languages", *IEEE Potentials*, vol.11, no. 2, pp. 30 - 33, Abril

RATNER, D., KERROW, P., 2000, "Using Labview to prototype na industrial-quality real-time solution for the Titan outdoor 4WD mobile robot controller" *Proceedings of the International IEEE/ RSJ Conference on Intelligent Robots and Systems*, pp. 1438-1432.

ROCHA, A. R. C., AGUIAR, T. C., SOUZA, J. M., 1990, "TABA: A Heuristic Workstation for *Software* development", In: *Proceedings of COMPEURO 90*, Tel Aviv, Israel, Maio.

ROCHA, A.R. C., MALDONADO, J.C., WEBER, K.C., 2001. Qualidade de Software. São Paulo. Prentice Hall.

RUS, I, LINDVALL, M., 2002, "Knowledge Management in Software Engineering", *IEEE Software*, (May/Jun), pp. 26-38.

SANTOS, G.S., *Representação da Distribuição do Conhecimento, Hablidades e Experiências Através da Estrutura Organizacional*, Tese de MSc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, março 2003.

SANTOS, G., ZLOT, F., 1999, *Definição e Instanciação e Ambientes na Estação TABA*, Projeto Final de Curso, UFRJ, Rio de Janeiro, RJ, Brasil.

SCHNAIDER, L., 2003, *Planejamento da Alocação de Recursos Humanos em Ambientes de Desenvolvimento de Software Orientados à Organização*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

SEPIN, 2001, *Qualidade e produtividade no Setor de Software Brasileiro*, Secretaria de Política de Informática e Automação SEPIN, Ministério da Ciência e Tecnologia MCT.

SHU, N., C., 1988, *Visual Programming*, New York, Van Nordstrand Reinhold.

SMITH, J., 2003, "A Comparison of the IBM Rational Unified Process and eXtreme Programming" *IBM Rational whitepapers*, Online: http://www-306.ibm.com/software/rational/info/literature/whitepapers.jsp

SPOELDER, H., J., W., 1999, "Virtual Instrumentation and Virtual Environments" *IEEE Instrumentation & Measurement Magazine*, setembro 1999, pp. 14-19.

TOLEMAN M., CARRINGTON D., COOK P., COYLE A., MACDONALD A., WELSH J., JONES T., 2001, "Generic description of a software document environment", In: *Proceedings of the 34th Hawaii International Conference on System Sciences*, pp 1-10.

TRAVASSOS, G. H., 1994, *O Modelo de Integração de Ferramentas da Estação TABA*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

TRINDADE, A. L. P., PESSOA,M. S. P., SPINOLA, M. M. , 1999, "COCOMO II – Uma Compilação de Informações sobre a Nova Métrica", In: *Anais do V Congresso Internacional de Engenharia Informática da Universidade de Buenos Aires* – Argentina, pp. 1-17.

VILLELA, K., 2004, *Definição e Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio* , Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

VILLELA, K., ROCHA, A. R., TRAVASSOS, G.H., 2000, *Ambientes de Desenvolvimento de Software Orientados à Organização*, Publicação Técnica COPPE/UFRJ - ES530/00 Rio de Janeiro, RJ, Abril.

VILLELA, K., SANTOS, G., GALOTTA., C., 2001a, "Estendendo a Estação TABA para a criação de Ambientes de Desenvolvimento de *Software* Orientados a Organização", In: *Caderno de Ferramentas - XV Simpósio Brasileiro de Engenharia de Software*, pp. 359-362, Rio de Janeiro, RJ, outubro.

VILLELA, K., SANTOS, G., GALOTTA., C., 2001b, "Cordis-FBC: um Ambiente de Desenvolvimento de *Software* para Cardiologia", In: *I Workshop de Informática Médica*, Rio de Janeiro, RJ, outubro.

VILLELA, K., SANTOS, G., BONFIM, C., *et al.*, 2001c, "Knowledge Management in *Software* Development Environments", In: *Proceedings of 14th International Conference Software & Systems Engineering and their Applications*, Paris, dezembro.

VILLELA, K., SANTOS, G., TRAVASSOS, G. H., ROCHA, A. R., 2002, "Gestão de Conhecimento em Ambientes de Desenvolvimento de *Software*", *2ª Jornada Ibero-Americana de Engenharia de Software e Engenharia de Conhecimento*, Salvador, Brasil, Outubro

VILLELA, K., OLIVEIRA, K., SANTOS, G., ROCHA, A. R., TRAVASSOS, G., 2003, "Cordis-FBC: an Enterprise-Oriented Software Development Environment", In: *Proceedings of the Learning Software Organizations*.

VXIPLUG&PLAY SYSTEM ALLIANCE, 1994 *VPP-4.1: VISA-1 Virtual Instrument Software Architecture Main Specification*, maio 1994.

WANG, C., GAO, R., X., 2000, "A virtual Instrumentation System for Integrated Bearing Condition Monitoring" *IEE Transactions on Instrumentation And Measurement*, abril 2000, vol. 49, no. 2, pp 325-331.

WANG, O., L., 2001, "Developing Visual Component Library For a Graphical Programming Platform Using Object-Orientation", In: *Proceedings of the 37th IEEE InternationalConference on Automatic Test: AUTOTESTCON,* pp. 459-467.

WHITMIRE, S. A., 1995, "A Introduction to 3D Function Points", Software Development, Abril, pp. 43-53

WINCH, G., 1999, "Knowledge Management", *Manufacturing Engineer*, August, pp.178 – 180.

YOURDON, E., 1989 *Modern Structured Analysis,* Prentice-Hall.

YOURDON, E., COAD, P., 1991, *Object-Oriented Analysis*, Prentice-Hall.

ZLOT, F., 2002, *Conhecimento de Tarefa em Ambientes de Desenvolvimento de Software Orientados a Domínio*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

ZLOT, F. OLIVEIRA,K., ROCHA, A.R., 2002. "Modelling Task Knowledge to Support *Software* Development", *Software Engineering and Knowledge Engineering - SEKE'2002*, Ischia, Itália, Julho.

ANEXO I

Esta pesquisa costitui-se em 5 perguntas e visa levantar as práticas de Engenharia de Software utilizadas pelas empresas que utilizam linguagens de Instrumentação Virtual (ex: Labview e HP VEE). Ela faz parte de um trabalho maior que objetiva aplicar de forma facilitada as boas práticas de Engenharia de Software a área de Instrumentação Virtual, ajudando, por exemplo, empresas desta área a certificações de qualidades como a ISO 9001.

Este trabalho está sendo desenvolvido como uma Tese de Mestrado do Programa de Engenharia de Sistemas da COPPE/UFRJ Os dados obtidos serão utilizados de forma agregada numa base de informações estatísticas e os dados específicos de empresas não serão publicados ou divulgados sob qualquer forma. Os participantes da pesquisa receberam o trabalho para o qual a pesquisa contribuirá como forma de compensação pelo tempo dedicado.

Nome (Opcional):
Empresa (Opcional):
E-mail (Opcional):

Por favor, assinale o quadrado com a opção apropriada caso a empresa conheça alguma das Normas ou Modelo de Maturidade abaixo:

Norma/Modelo	Não conhece.	Conhece, mas não usa	Conhece e está começando a usar.	Conhece e usa.
ISO/IEC 12207 – Information Technology- Software Life Cycle Processes	☐	☐	☐	☐
CMM (Capability Maturity Model)	☐	☐	☐	☐
SPICE – Software Process Improvement and Capability Determination (ISO/IEC TR 15504)	☐	☐	☐	☐
NBR 13596 (ISO/IEC 9126)	☐	☐	☐	☐
ISO/IEC 12119	☐	☐	☐	☐

A empresa Possui certificação para o sistema da qualidade com desenvolvimento de Software no escopo?
01.☐ Sim
02.☐ Não, mas o sistema encontra-se em processo de certificação
03.☐ Não há certificação

Em caso afirmativo ou em processo de certificação qual o tipo:
01.☐ ISO 9001
02.☐ ISO 9002
03.☐ Certificado por Cliente
04.☐ CMM. Em que nível?_____
04.☐ Outras Certificações. Especifique: _____

Práticas de Engenharia de Software adotadas no desenvolvimento e manutenção de softwares:

01. ☐ auditoria
02. ☐ Análise crítica conjunta
03. ☐ Controles de versão de produto
04. ☐ Engenharia da informação
05. ☐ Gerência de configuração
06. ☐ Gerência de requisitos
07. ☐ Gerência de Projetos
08. ☐ Inspeção Formal
09. ☐ Medições de Qualidade (métricas)
10. ☐ Processo de Software definido e documentado
11. ☐ Reuso
12. ☐ Verificação/Revisões
13. ☐ Gestão de mudança
14. ☐ Joint Application Design – JAD
15. ☐ Métodos estruturados
16. ☐ Métodos orientados a objetos
17. ☐ Modelagem de dados
18. ☐ Normas e padrões da organização
19. ☐ Planejamento formal de testes
20. ☐ Projeto da interface com o usuário
21. ☐ Prototipação
22. ☐ Walkthrough Estruturado
23. ☐ Outras. Especifique: _____

Ferramentas:

01. ☒ Analisador de código
02. ☒ CASE Lower
03. ☐ CASE Upper
04. ☐ Depurador interativo
05. ☐ Distribuição de software
06. ☐ Documentador
07. ☐ Driver de teste
08. ☒ Gerador de código-fonte
09. ☐ Gerador de dados de teste
10. ☐ Gerador de entrada de dados
11. ☐ Gerador de gráficos
12. ☐ Gerador de GUI
13. ☐ Gerador de relatórios
14. ☐ Gerador de telas
15. ☐ Gerenciador de bibliotecas de módulos
16. ☐ Gerenciador de configuração
17. ☐ Gerenciador de conteúdo
18. ☐ Gerenciador de documentos
19. ☐ Gerenciador de projetos
20. ☐ Otimizador
21. ☐ Prototipador
22. ☐ Outras. Especifique: _____

Documentação:

01. ☐ Contratos e acordos
02. ☐ Documentação de marketing
03. ☐ Documentação de programas
04. ☐ Documentação do processo de software
05. ☐ Documentação no código
06. ☐ Descrição do produto p/ comercialização
07. ☐ Especificação do sistema
08. ☐ Guia de instalação
09. ☐ Help on-line
10. ☐ Manual de treinamento
11. ☐ Manual do usuário
12. ☐ Plano de controle da qualidade
13. ☐ Plano de testes
14. ☐ Projeto do sistema
15. ☐ Registro formal de revisões e testes
16. ☐ Outras. Especifique: _____

Muito obrigado por sua colaboração com nossa pesquisa

Contato:
Fábio Renato S. Martins / Ana Regina Rocha
COPPE/UFRJ – Sistemas
Caixa Potal 68511

GLOSSÁRIO

(Fonte: Qualidade e Produtividade no Setor de Software Brasileiro.MCT/SEPIN, 2000.)

Analisador de Código

Software que percorre um trecho de código, uma rotina ou um programa, com a finalidade de coletar métricas de complexidade ou de elaborar um grafo ou outra descrição da lógica do código percorrido.

Análise Crítica (Review)

Avaliação profunda e global de um projeto, produto, serviço, processo ou informação com relação a requisitos, objetivando a identificação de problemas e a proposição de soluções. *[Critérios de Excelência da Fundação para o Prêmio Nacional da Qualidade – FPNQ]*

Auditoria

Exame sistemático e independente, para determinar se as atividades da qualidade e seus resultados estão de acordo com as disposições planejadas, se estas foram implementadas com eficácia e se são adequadas à consecução dos objetivos. *[NBR ISO 8402]*

CASE - Computer Aided Software Engineering

Ferramenta de apoio ao desenvolvimento de software. Em linhas gerais, apóia a execução de atividades do desenvolvimento do software de forma automatizada. Em alguns casos, implementa um ambiente relativamente refinado no qual várias atividades de especificação ou codificação são apoiadas por recursos computacionais. Dependendo do tipo de atividade suportada podem ser classificados em Lower CASE, provendo suporte à codificação, teste, depuração e manutenção do código ou Upper CASE, suportando diversas tarefas de análise e projeto de sistemas. Eventualmente, ferramentas CASE podem ser integradas em ambientes de desenvolvimento de software. Neste caso, apoiando parte das atividades previstas em um processo de desenvolvimento de software.

Configuração

Relação entre versões de um objeto composto, ou seja, configuração é uma instância do sistema composta da união de uma versão específica de cada objeto componente. Arranjo de um sistema computacional ou de seus componentes como definidos pelo seu número, natureza e interconexão de suas partes constituintes. *[IEEE Std 610.12]*

Controle de Versão

Procedimento de gestão do ciclo de vida de um produto. Consiste na identificação formal de modificações solicitadas ou efetuadas e no seu agrupamento, de modo a que fiquem incorporadas, todas elas, em uma determinada configuração do produto, num certo momento. Essa configuração recebe o nome de versão.

Depurador Interativo

Software para apoio a testes, cuja função é permitir a visualização passo a passo da execução de uma rotina ou programa e do comportamento de seus elementos antes, durante e após a execução.

Driver de Teste

Software que permite a ativação de determinadas partes do software (módulos) com o intuito de testá-las. Normalmente, utilizam-se massas de teste previamente definidas e produzem resultados do teste, que podem ser verificados através da documentação de teste construída pelo analista ou engenheiro de software responsável pelos testes.

Engenharia da Informação

Popularizada por James Martin, é um caminho direcionado a dados para desenvolvimento de sistemas de informação, oposto à visão de direcionamento a processo de análise estruturada. *[Marciniak J.J., Encyclopedia of Software Engineering]*

Gerador de GUI

O processo de projeto de interfaces com o usuário é iterativo. Ou seja, um modelo de projeto é criado, implementado como protótipo, examinado pelos usuários e modificado, baseado em seus comentários. O jogo de ferramentas (toolkit) de interfaces com usuário ou sistema de desenvolvimento de interfaces com o usuário (User-Interface Development Systems – UIDS), essas ferramentas oferecem módulos ou objetos que facilitam a criação de janelas, menus, interação de dispositivos, mensagens de erro, comandos e muitos outros elementos de um ambiente interativo. Os sistemas de desenvolvimento de interfaces com o usuário (User Interface Development Systems – UIDS) combinam ferramentas CASE individuais para interação humano computador com uma biblioteca de componentes de programa que possibilita que o desenvolvedor construa uma interface humano computador rapidamente. O UIDS oferece componentes de programa que gerenciam dispositivos de entrada, validam entradas do usuário, manipulam condições de erro, processam "*undos*"e *aborts*, oferecem *feedback* visual, *prompts* e socorro, atualizam o *display*, gerenciam dados de aplicação, manipulam *scrolling* e *editing*, isolam a aplicação das funções de gerenciamento da tela e suportam características de customização para o usuário final. *[Pressman R. S., Engenharia de Software, 1995]*

Gerência de Requisitos

Estabelecimento e manutenção de um entendimento/acordo com o cliente sobre os requisitos para o projeto de software. Este acordo refere-se aos requisitos do sistema alocados para o software. O cliente pode ser interpretado como o grupo de engenharia do sistema, o grupo de marketing, outra organização interna, ou um cliente externo. O acordo compreende requisitos técnicos e não técnicos. O acordo forma a base para a estimativa, planejamento, execução e acompanhamento das atividades do projeto de software através do ciclo de vida do software. *[Key Practices of the Capability Maturity Model, versão 1.1, Feb. 1993]*

Gerenciamento de Configuração (Software Configuration Management – SCM)

Atividade abrangente que é aplicada em todo o processo de engenharia de software, podendo ser vista como uma atividade de garantia da qualidade de software. Uma vez que uma mudança pode ocorrer a qualquer tempo, as atividades de SCM são desenvolvidas para identificar a mudança; controlar a mudança; garantir que a mudança esteja sendo adequadamente implementada; e relatar a mudança a outras pessoas que possam ter interesse nela. O gerenciamento de configuração de software é um conjunto de atividades que foi desenvolvido para administrar as mudanças em todo o ciclo de vida do software.

Inspeção Formal

Técnica de revisão sistemática do software ou de alguns de seus componentes, executada, sistematicamente, ao final de cada fase do projeto, com o objetivo único de encontrar erros. A inspeção formal é executada por uma equipe na qual cada membro tem papel preestabelecido. O projetista participa mas não coordena a reunião. Todo o material gerado é lido, os erros anotados e uma estatística dos erros encontrados é mantida, para fins de posterior estudo da eficácia do procedimento.

JAD - Joint Application Design

Conjunto de sessões intensivas e mediadas entre usuários e analistas de um sistema, com o objetivo de explicitar os seus requisitos. A técnica, desenvolvida nos anos setenta pela IBM do Canadá, voltou a ficar em voga com o uso do RAD - Rapid Application Development, metodologia que combina o JAD (para definir rapidamente a especificação do sistema) com o uso de ferramentas CASE e de metodologias de prototipação, para chegar a um produto final em menor tempo.

Otimizador

Software, usualmente embutido no compilador que otimiza o código gerado a partir do exame do programa a ser compilado, eliminando redundâncias, código inacessível, etc.

Processo

Conjunto de recursos e atividades inter-relacionadas que transformam insumos (entradas) em produtos (saídas). *[NBR ISO 8402].* Agrupamento em seqüência de todas as tarefas destinadas a obter um determinado resultado. É a combinação de equipamentos, instalações, mão-de-obra, métodos, técnicas, ferramentas, procedimentos e outros fatores, com a finalidade de elaborar um produto ou alcançar um resultado preestabelecido.

Processo de Software

Conjunto de atividades, métodos, práticas e transformações que as pessoas empregam para desenvolver e manter software e os produtos associados (por exemplo, planos de projeto, documentos de projeto/design, código, casos de teste, manual do usuário).

Programação Orientada a Objetos

Técnica de programação que enfatiza a descrição dos conceitos envolvidos com o domínio do problema (objetos) através de seus dados e operações, encapsulados e representados através de classes. Cada objeto é criado como pertencendo a uma classe. A utilização de um objeto, e sua eventual mudança de estado, se dá a partir de mensagens enviadas a ele, representadas pelas

operações encapsuladas na classe. Novas classes podem ser criadas a partir de classes existentes e organizadas através de um processo de classificação e hierarquização, explorando o conceito de herança. Os programas são construídos como organizadores da ativação de mensagens para os objetos, desta forma fazendo com que as funcionalidades de um sistema sejam obtidas através da cooperação dos objetos.

Projeto da Interface com o Usuário

O processo global para projetar uma interface com o usuário inicia-se com a criação de diferentes modelos de função do sistema. Quatro diferentes modelos entram em cena quando uma HCI vai ser projetada. O engenheiro de software cria um modelo de projeto; um engenheiro humano estabelece um modelo de usuário, o usuário final desenvolve uma imagem mental que muitas vezes é chamada modelo do usuário ou de percepção do sistema e os implementadores do sistema criam uma imagem do sistema. *[Pressman R. S., Engenharia de Software, 1995]*

Prototipação

Método de desenvolvimento que prevê a execução de vários ciclos de análise, especificação e codificação de um sistema. No primeiro ciclo, gera-se um produto simplificado em pouco tempo, de modo que o usuário possa examiná-lo e refinar as suas demandas. Nos ciclos seguintes, o produto é aperfeiçoado e novas funções são sucessivamente implementadas, até se chegar ao produto final.

Walkthrough Estruturado

Técnica de análise estática na qual um projetista ou programador apresentam aos membros do grupo de desenvolvimento e outros profissionais interessados uma parte de documentação ou código, e os participantes fazem perguntas e comentários sobre possíveis erros, violação de padrões de desenvolvimento ou sobre outros problemas.

Processo de Configuração do Meta-ambiente TABA

As atividades que compõem o processo de configuração do Meta-ambiente TABA para uma organização são descritas a seguir, juntamente com as orientações para sua execução e os respectivos responsáveis.

➢ Atividade 1: Contextualizar Configuração

Esta atividade tem como objetivo contextualizar a configuração, indicando as características gerais da organização, a sua cultura na área de software e quais são os objetivos almejados com a configuração, de forma a fornecer os subsídios necessários à elaboração da proposta de configuração.

Responsáveis: Configurador de Ambientes e representantes da organização.

Sub-atividades:

Sub-atividade 1.1: Realizar Entrevistas

O objetivo é realizar as entrevistas necessárias para coletar as informações para caracterização da organização, identificação da cultura organizacional na área de software e identificação dos objetivos a serem atingidos a partir do Ambiente Configurado.

Sub-atividade 1.2: Caracterizar a Organização

O objetivo é registrar informações básicas sobre a organização que indiquem o seu tamanho e área de atuação, o que implica em fornecer respostas para as seguintes perguntas:

- Qual o tamanho da organização? (Micro – 1 a 9 pessoas, Pequena – 10 a 49 pessoas, Média – 50 a 99 pessoas, Grande – mais de 100 pessoas[1])
- A organização pertence a que tipo de indústria[2]? (Ex: Saúde, Telecomunicação, Transporte Aéreo e Software/Hardware)
- Quais são os domínios de conhecimento relacionados ao negócio da organização? (Ex: Cardiologia, Telefonia Celular, Aviação Comercial, Engenharia de Software e Redes de Computadores)

[1] Esta classificação, quanto ao tamanho, das organizações que desenvolvem e mantêm software foi proposta em (SEPIN/MCT, 2002), sendo também utilizada no capítulo 7.
[2] JONES (2000a) propõe, em estudos de software, o uso de códigos elaborados para a classificação de indústrias, listando os códigos das indústrias com maior população de desenvolvedores de software.

- Quais são os domínios de conhecimento envolvidos no apoio ao negócio da organização e para os quais sistemas de software têm sido ou pretendem ser desenvolvidos? (Ex: Administração Hospitalar, Contabilidade e Engenharia de Software)

Sub-atividade 1.3: Identificar Cultura Organizacional na Área de Software

O objetivo é identificar os elementos que compõem a cultura da organização na área de software, o que envolve tipos de software produzidos, tamanho e complexidade dos projetos, tecnologias utilizadas, certificações e/ou avaliações obtidas e problemas enfrentados. Sendo assim, as respostas para as seguintes perguntas precisam ser registradas:

- Quantos profissionais da organização, aproximadamente, atuam na área de software?
- Quais são os tipos de software, considerando a finalidade, que são desenvolvidos pela organização? (Software para uso próprio, Pacote de software para comercialização, Software para uso por terceiros desenvolvido sob encomenda[3])
- Quais são os tipos de software, considerando tecnologia envolvida, que são desenvolvidos pela organização? (Ex: Sistemas Web, Sistemas Cliente-Servidor e Sistemas Especialistas)
- Qual o tamanho e a complexidade dos projetos que têm sido conduzidos pela organização? (Quanto ao tamanho: Pequeno, Médio e Grande[4], e quanto à complexidade: Baixa, Média, Alta e Software Crítico)
- Quais são os paradigmas de desenvolvimento utilizados na organização? (Estruturado, Orientado a Objeto, Baseado em Conhecimento)
- Quais são os métodos, técnicas e ferramentas utilizados na organização?
- A organização já possui processos de desenvolvimento e manutenção definidos? Há interesse em redefini-los?
- A organização possui certificação ISO relativa à área de software? Qual é o escopo?
- A organização possui avaliação segundo o CMM ou CMMI? Em que nível? Formal ou informal?
- Caso a organização não tenha processos definidos, quais são os documentos produzidos na organização em seus projetos de software? Existem procedimentos já definidos? Há a necessidade de executar atividades que não são geralmente

[3] As opções fornecidas são uma simplificação da classificação utilizada em (SEPIN/MCT, 2002).

[4] Esta classificação dos projetos quanto ao tamanho obedece ao critério definido na tabela 7.6 apresentada no capítulo 7.

executadas em outras organizações ou que são executadas de forma particular na organização?

- Quais os principais problemas enfrentados pela organização em seus projetos de software? (Ex: Falta de conhecimento sobre o domínio, falta de experiência dos desenvolvedores, alta rotatividade, estimativas não realistas e muito re-trabalho)

Sub-atividade 1.4: Identificar Objetivos para a Configuração do Ambiente

O objetivo é identificar os objetivos da organização na área de software que estão motivando a configuração do ambiente. Desta forma, respostas para as seguintes perguntas precisam ser fornecidas:

- Quais são os objetivos da organização na área de software? (Ex: Obter certificação ISO, atingir um determinado nível de maturidade segundo o CMM ou CMMI, adotar uma nova tecnologia, atuar em novo domínio de conhecimento e tratar um dos problemas freqüentes nos projetos da organização)

  Quais desses objetivos devem ser apoiados pelo ambiente a ser configurado?

➢ Atividade 2: Elaborar Proposta para Configuração do Ambiente

Esta atividade tem como objetivo elaborar a proposta de configuração de um ambiente para a organização, de acordo com as informações previamente obtidas, o que implica em especificar o conteúdo e as ferramentas que serão disponibilizadas no Ambiente Configurado e em elaborar planos de organização e custos, além de cronograma para a configuração.

Responsável: Configurador de Ambientes.

Sub-atividades:

Sub-atividade 2.1: Identificar Teorias de Domínio

O objetivo é identificar, considerando os domínios de conhecimento mencionados nas atividades anteriores, as Teorias do Domínio que serão disponibilizadas no Ambiente Configurado, definindo os seus respectivos propósitos, o que implica em definir qual será a sua utilização e delimitar o seu escopo. Esta atividade envolve, portanto, uma descrição geral do propósito de cada Teoria do Domínio, a elaboração de cenários de motivação que demonstrem a sua utilização e a formulação de questões gerais de competência para delimitação do seu escopo, que são questões abrangentes que indicam aspectos a serem tratados pela mesma. Em caso de uma Teoria do Domínio já ter sido definida anteriormente, identifica-se a necessidade ou não de evoluí-la.

Sub-atividade 2.2: Identificar Processos

O objetivo é identificar os novos processos a serem definidos e os processos já existentes que precisam ser revistos, de forma a fornecer para a organização um processo padrão contemplando os processos de ciclo de vida de desenvolvimento e manutenção, além dos processos especializados para cada contexto de desenvolvimento de software existente na organização. O processo padrão é a base para a definição dos processos especializados, que o adaptam de acordo com os paradigmas de desenvolvimento, métodos e/ou tipos de software que compõem os contextos específicos de desenvolvimento de software na organização.

Sub-atividade 2.3: Identificar Ferramentas

O objetivo é identificar as ferramentas que serão disponibilizadas para uso no Ambiente Configurado e/ou nos Ambientes Instanciados a partir dele, o que inclui tanto ferramentas internas quanto ferramentas externas. Nesta atividade, deve ser verificada a necessidade de construção e/ou aquisição de novas ferramentas. As atividades necessárias à aquisição de uma ferramenta constituem um processo independente do processo de configuração, devendo ser executadas de acordo com a norma internacional ISO/IEC 14598-5 Parte 4 (ISO/IEC, 1998) e o processo para aquisição de produtos de software da organização. Já as atividades necessárias à construção de uma ferramenta devem ser apoiadas por um ambiente instanciado com esta finalidade, adotando-se o processo de construção de ferramentas definido para o projeto TABA.

Sub-atividade 2.4: Definir Equipes e Responsabilidades

O objetivo é estabelecer quem será o coordenador do projeto de configuração do Meta-ambiente TABA para a organização e definir equipes responsáveis pelas macro-atividades que fazem parte da proposta de configuração. Exemplos de macro-atividades são: Definição dos Processos, Definição da Teoria do Domínio, Construção de Ferramentas, Treinamento e Acompanhamento de Projeto Piloto.

Sub-atividade 2.5: Definir Cronograma

O objetivo é estabelecer uma cronograma para a execução das macro-atividades que compõem a proposta de configuração.

Sub-atividade 2.6: Definir Custos

O objetivo é elaborar uma planilha de custo por macro-atividades, estabelecendo também o custo total da configuração do Meta-ambiente TABA para a organização.

Sub-atividade 2.7: Enviar Proposta

O objetivo é enviar a proposta elaborada ao longo desta atividade para que os Representantes da Organização a analisem, identifiquem possíveis dúvidas e problemas, podendo, assim; fornecer um parecer sobre a mesma.

➢ Atividade 3: Registrar Parecer sobre a Proposta

Esta atividade tem como objetivo registrar o parecer dos Representantes da Organização sobre a proposta de configuração, indicando a sua aprovação, a necessidade de modificações e ajustes, ou a não aprovação da proposta. No caso da necessidade de modificações e ajustes, uma data para envio da nova proposta é estabelecida.

Responsável: Configurador de Ambientes.

➢ Atividade 4: Definir Processo Padrão

Esta atividade tem como objetivo definir o processo padrão da organização, contemplando os processos de desenvolvimento e de manutenção de software.

Responsável: Configurador de Ambientes.

Sub-atividades:

Sub-atividade 4.1: Caracterizar Processo Padrão

O objetivo é caracterizar o processo padrão a ser definido, estabelecendo quais são os processos de ciclo de vida que devem compor o processo padrão e se as definições dos processos de ciclo de vida devem ser baseadas ou não em processos já definidos anteriormente. Caso não devam ser baseados em processos já definidos, a norma ISO/IEC 12207 (ISO/IEC, 1995) é utilizada com padrão de referência.

Sub-atividade 4.2: Incluir Atividades do Processo Padrão Base (quando pertinente)

O objetivo é incluir, automaticamente, no novo processo padrão da organização, as atividades do processo padrão selecionado como base para a definição do processo. Se necessário, atividades podem ser alteradas ou excluídas.

Sub-atividade 4.3: Incluir Atividades ISO/IEC 12207

O objetivo é selecionar as atividades da norma ISO/IEC 12207 (ISO/IEC, 1995) a serem incluídas no processo padrão da organização, o que é de fundamental importância quando um requisito do processo padrão é estar de acordo com a norma.

Sub-atividade 4.4: Incluir Atividades do Tipo de Software

O objetivo é selecionar, para inclusão no processo padrão da organização, as atividades que são próprias do desenvolvimento ou da manutenção de determinados tipos de software. Atividades desta natureza só devem ser incluídas no processo padrão de uma organização quando são aplicáveis a qualquer projeto de software conduzido na organização. Um exemplo é quando a organização só desenvolve e mantém software crítico de tempo real. Nesta situação, tanto atividades próprias de software crítico quanto atividades próprias de software de tempo real podem e devem ser incluídas no processo padrão.

Sub-atividade 4.5: Incluir Atividades Próprias da Organização

O objetivo é definir e incluir as atividades que devem fazer parte do processo padrão e que são específicas da organização para a qual o ambiente está sendo configurado. Uma exemplo de atividade própria da organização pode ser a que estabelece a elaboração de uma proposta comercial.

Sub-atividade 4.6: Incluir Atividades Orientadas a Domínio

O objetivo é incluir as atividades orientadas a domínio no processo padrão da organização. No Meta-ambiente TABA, tem-se cadastrado a atividade de Investigação do Domínio, a ser incluída como sub-atividade de outras atividades, estabelecendo a pesquisa e o uso do conhecimento sobre o domínio que sejam necessários para a execução da atividade correspondente (OLIVEIRA, 1999). No entanto, atividades orientadas a domínio mais específicas podem ser cadastradas para fornecer aos desenvolvedores descrições mais voltadas para as atividades a serem apoiadas, se assim for desejado. Atividades orientadas a domínio só devem ser incluídas no processo padrão se estiver prevista, na proposta de configuração, a inclusão de pelo menos uma Teoria de Domínio referente a um domínio de negócio.

Sub-atividade 4.7: Determinar Atividades Obrigatórias

O objetivo é determinar quais atividades do processo padrão são obrigatórias e, conseqüentemente, não podem ser retiradas dos processos que serão posteriormente definidos a partir do mesmo. As atividades obrigatórias compõem o conjunto mínimo de atividades a serem executadas independente das características específicas de cada projeto.

Sub-atividade 4.8: Adaptar Roteiros de Documentos para a Organização

O objetivo é adaptar roteiros de documentos disponíveis na Meta-ambiente TABA para a organização, de forma a disponibilizá-los para uso no Ambiente Configurado.

➢ Atividade 5: Definir Processo Especializado

Esta atividade tem como objetivo definir, a partir do processo padrão, processos especializados para os paradigmas de desenvolvimento utilizados na organização e, opcionalmente, para diferentes tipos de software.

Responsável: Configurador de Ambientes.

Sub-atividades:

Sub-atividade 5.1: Especializar Atividades do Processo Padrão

O objetivo é alterar a descrição das atividades previstas no processo sendo especializado, de forma a torná-las mais adequadas ao paradigma e respectivos métodos de desenvolvimento. Isto inclui a possível inclusão ou especialização de sub-atividades, definição de novos produtos e utilização de novas ferramentas.

Sub-atividade 5.2: Incluir Atividades do Tipo de Software

O objetivo é incluir, em um processo especializado, as atividades que são próprias do desenvolvimento ou da manutenção de determinados tipos de software, dando origem a um processo especializado tanto para o paradigma quanto para um ou mais tipos de software. No entanto, nem sempre a execução desta atividade é necessária, pois as atividades dessa natureza podem já ter sido incluídas no processo padrão (sub-atividade 4.4: *Incluir Atividades do Tipo de Software*), podem ter a inclusão postergada até a instanciação do processo para um projeto específico, ou os tipos de software desenvolvidos na organização podem não requerer a execução de atividades específicas.

Sub-atividade 5.3: Determinar Atividades Obrigatórias

O objetivo é determinar quais atividades do processo especializado são obrigatórias e, conseqüentemente, não podem ser retiradas durante a definição dos processos para os projetos específicos. As atividades definidas como obrigatórias no processo padrão permanecem como obrigatórias.

Sub-atividade 5.4: Especializar Roteiros de Documentos

O objetivo é especializar, quando necessário e de acordo com as características de cada processo especializado, os roteiros de documentos definidos para a organização, de forma a poder disponibilizar estes roteiros para uso no Ambiente Configurado.

➢ Atividade 6: Definir Teoria do Domínio

Esta atividade tem como objetivo definir uma Teoria do Domínio para um determinado domínio de conhecimento.

Responsáveis: Engenheiro de Conhecimento e especialistas no domínio.

Sub-atividades:

Sub-atividade 6.1: Revisar Propósito

O objetivo é revisar o propósito estabelecido para a Teoria do Domínio na sub-atividade 2.1 (*Identificar Teorias de Domínio*), efetuando os ajustes necessários na descrição geral do propósito, nos cenários de motivação e nas questões gerais de competência.

Sub-atividade 6.2: Especificar Requisitos

O objetivo é especificar os requisitos da Teoria do Domínio através da formulação de questões de competência, que são as questões que a Teoria do Domínio deve possibilitar que sejam respondidas.

Sub-atividade 6.3: Capturar Teoria do Domínio

O objetivo é identificar conceitos, relações, propriedades e restrições da ontologia do domínio que compõe a Teoria do Domínio. Parte-se de um conjunto preliminar de termos e de frases potencialmente relevantes, que são agrupados de acordo com a relação semântica existente, dando origem a grupos de trabalho. Para cada grupo de trabalho, são geradas as definições preliminares dos conceitos representados pelos termos, substituídos os termos ambíguos, eliminados os conceitos redundantes e solucionadas as inconsistências. A partir das frases potencialmente relevantes, são identificadas as relações entre os conceitos. Em seguida, são identificadas as propriedades necessárias nos conceitos e relações, de forma que as questões de competência sejam respondidas. Os grupos de trabalho dão, naturalmente, origem às Subteorias do Domínio. A identificação das Subteorias de um Domínio permite verificar a possibilidade de integração de Subteorias já existentes ou de parte de seus conceitos.

Sub-atividade 6.4 Conceituar Teoria do Domínio

O objetivo é gerar definições precisas e não ambíguas para a ontologia capturada. Nesta sub-atividade devem ser elaborados dicionários de conceitos, de propriedades e de relações entre conceitos, árvores de classificação de conceitos (quando pertinentes) e, também quando pertinente, uma tabela de fórmulas utilizadas para inferir valores numéricos que relacionam propriedades. Para concluir a atividade, restrições axiomáticas (axiomas que estabelecem restrições para as relações entre conceitos ou entre propriedades de um conceito) devem ser descritas em linguagem natural. Estas restrições axiomáticas são definidas com base nas questões de competência.

Sub-atividade 6.5: Formalizar Teoria do Domínio

O objetivo é descrever a ontologia do domínio utilizando uma linguagem de representação que elimine as possíveis ambigüidades provenientes do uso da linguagem natural. A depender da linguagem utilizada, todos os axiomas tem que ser explicitamente definidos, inclusive os axiomas que representam especialização e composição e os que especificam domínio de valores para os atributos.

Sub-atividade 6.6: Identificar Tarefas

O objetivo é identificar quais são as tarefas que são executadas no contexto definido por cada Subteoria do Domínio. Quando uma tarefa identificada não tiver sido ainda descrita no Meta-ambiente TABA, a atividade 7 (*Descrever Tarefa*) deve ser executada.

Sub-atividade 6.7: Avaliar Teoria do Domínio

O objetivo é avaliar a Teoria de Domínio definida, de forma a assegurar que ela cumpre o seu propósito, satisfazendo os requisitos estabelecidos pelas questões de competência e realmente sendo capaz de descrever o mundo real para o qual foi projetada. Além disso, também é objetivo desta atividade avaliar a Teoria do Domínio segundo os critérios de qualidade apresentados em (OLIVEIRA, 1999). Desta forma, esta atividade compreende tanto a verificação quanto validação da Teoria do Domínio.

Sub-atividade 6.8: Implementar Teoria do Domínio

O objetivo é implementar a formalização da Teoria do Domínio no Meta-ambiente. Com isso, a Teoria do Domínio passa a estar disponível como módulos de conhecimento para que possa ser instanciada e acessada. A integração das Subteorias do Domínio já disponíveis no meta-ambiente é fundamental para manter a completude e a consistência da Teoria do Domínio.

As sub-atividades 6.2, 6.3 e 6.4 (*Especificar Requisitos*, *Capturar Teoria do Domínio* e *Conceituar Teoria do Domínio*) são executadas de forma iterativa. A primeira iteração tem como objetivo formular as questões de competência, obter um conjunto preliminar de termos e dividi-los em grupos de trabalho. Tantas iterações quanto necessárias devem ser realizadas a fim de obter o modelo parcial da Teoria do Domínio, o que não inclui a descrição das restrições axiomáticas em linguagem natural. Um dos possíveis critérios para definição do número de iterações é o número de grupos de trabalho. A última iteração tem como objetivo completar o modelo da Teoria do Domínio com as restrições axiomáticas.

➢ Atividade 7: Descrever Tarefa

Esta atividade tem como objetivo descrever uma tarefa genérica que tenha sido identificada como pertinente ao contexto de uma Subteoria do Domínio. Apesar da necessidade de descrição da tarefa estar associada à definição de uma Subteoria, a descrição da tarefa em si é independente do domínio de conhecimento específico, permitindo a formação de uma biblioteca de tarefas que pode ser utilizada para descrever o conhecimento de diversos domínios. Novas tarefas são descritas e acrescentadas a biblioteca na medida em que são necessárias.

Responsáveis: Engenheiro de Conhecimento e especialistas na execução da tarefa.

Sub-atividades:

Sub-atividade 7.1: Descrever Tarefa em Linguagem Natural

O objetivo é descrever a tarefa em linguagem natural, o que implica em descrever o objetivo da tarefa e os elementos (objetos e ações) que são necessários para a solução do problema proposto pela tarefa.

Sub-atividade 7.2: Definir Estratégia para Solução da Tarefa

O objetivo é escolher um método para a solução do problema proposto pela tarefa e, a depender da estratégia adotada pelo método escolhido, descrever a solução do problema em linguagem natural. A estratégia adotada pelo método determina se a tarefa deve ser considerada elementar ou composta, ou seja, se o problema proposto pela tarefa deve ser resolvido diretamente através de inferências ou se é necessário decompor a tarefa original em tarefas mais simples (sub-tarefas). Quando a tarefa for considerada composta, a solução do problema deve ser descrita em linguagem natural, especificando as suas sub-tarefas e o fluxo de controle necessário.

Sub-atividade 7.3: Capturar Conceitos e Relações da Tarefa

O objetivo é identificar, a partir dos objetos mencionados na descrição em linguagem natural da tarefa, a lista de conceitos envolvidos na solução do problema e as relações entre estes conceitos. Os conceitos e relações compõem a ontologia da tarefa, que permite simular o processo de solução do problema, mas não demonstra com o problema pode ser resolvido. Os conceitos da tarefa são, na realidade, os papéis de conhecimento que serão preenchidos por conceitos do domínio quando a tarefa estiver sendo executada no contexto de um domínio específico.

Sub-atividade 7.4: Descrever Fluxo de Controle da Tarefa

O objetivo é descrever, em linguagem estruturada, um procedimento que represente o fluxo de controle necessário para a solução do problema quando a tarefa tiver sido considerada composta. Isto implica que o procedimento deve especificar a ordem de execução das sub-tarefas que compõem a tarefa.

Sub-atividade 7.5: Formalizar Descrição da Tarefa

O objetivo é obter uma representação formal da descrição da tarefa. Tanto os conceitos, relações e restrições da ontologia da tarefa quanto o fluxo de controle anteriormente definidos precisam ser descritos em uma linguagem formal, eliminando ambigüidades. No entanto, se a tarefa tiver sido considerada elementar, ao invés da formalização do fluxo de controle, são definidas regras de inferência, também em linguagem formal, que determinam como solucionar a tarefa a partir da interação entre os seus conceitos.

Sub-atividade 7.6: Avaliar Descrição da Tarefa

O objetivo é avaliar a Descrição da Tarefa, de forma a assegurar que ela realmente descreve o problema proposto pela tarefa e apresenta uma solução para o problema que seja válida no mundo real. Se a tarefa for composta, suas sub-tarefas devem ter sido avaliadas previamente. Outro objetivo desta atividade é avaliar a Descrição da Tarefa segundo um subconjunto de critérios de qualidade apresentados em (OLIVEIRA, 1999) que também se aplicam à avaliação de Descrições de Tarefa. Desta forma, esta atividade compreende tanto a verificação quanto validação da Descrição da Tarefa.

Sub-atividade 7.7: Implementar Descrição da Tarefa

O objetivo é implementar a Descrição da Tarefa no meta-ambiente, codificando a sua formalização e, quando a tarefa for composta, integrando a sua descrição com as descrições de suas sub-tarefas. Como isso, a Descrição da Tarefa torna-se disponível para uso e assegura-se a sua completude e consistência.

No caso da estratégia adotada pelo método (sub-atividade 7.2: *Definir Estratégia para Solução da Tarefa*) considerar a tarefa como composta, todas as sub-atividades acima são executadas de forma recursiva para cada sub-tarefa na qual a tarefa tenha sido decomposta, a menos que a sub-tarefa já conste na biblioteca de tarefas. Conforme estabelecido por ZLOT (2002), a descrição das sub-tarefas acrescenta novos conceitos aos já identificados para a tarefa, de forma a permitir a comunicação entre duas ou mais sub-tarefas.

➢ Atividade 8: Incluir Novas Ferramentas nas Descrições dos Processos

Ferramentas que tenham sido construídas após a definição dos processos padrão e especializados precisam ser incluídas na descrição das atividades que as utilizam, para que, no momento da configuração, possam ser integradas ao Ambiente Configurado. O objetivo desta atividade é, então, introduzir a referência a estas ferramentas na descrição dos processos padrão e/ou especializados.

Responsável: Configurador de Ambientes.

➢ Atividade 9: Criar Ambiente Configurado

O objetivo desta atividade é criar o Ambiente Configurado para a organização. Esta atividade conclui o processo de configuração.

Responsável: Configurador de Ambientes.

Sub-atividades:

Sub-atividade 9.1: Definir Ambiente Configurado

O objetivo é definir o Ambiente Configurado, selecionando o processo padrão e as Teorias de Domínio a serem fornecidas como conteúdo inicial do ambiente.

Sub-atividade 9.2: Gerar Ambiente Configurado

O objetivo é gerar o código executável do Ambiente Configurado, opcionalmente disponibilizando o código fonte, a partir de informações como logomarca da organização, ícone para acesso e diretórios de armazenamento.

Sub-atividade 9.3: Testar Ambiente Configurado

O objetivo é testar o Ambiente Configurado para assegurar que ele foi gerado corretamente.

I.2 Atividades de Preparação do Ambiente Configurado para Uso na Organização

Uma vez que o Ambiente Configurado é entregue à organização, o seu Gerente de Conhecimento prepara o ambiente para utilização, introduzindo conhecimento sobre a organização, além de itens de conhecimento sobre desenvolvimento e manutenção de software que tenham sido anteriormente adquiridos e estejam prontos para serem disponibilizados no ambiente. Não existe um processo que oriente a execução das atividades a seguir, porque elas podem ser executadas de forma aleatória e a qualquer momento.

➢ Atividade: Descrever Organização

Esta atividade tem como objetivo obter e registrar informações sobre a organização, de forma que estas informações possam ser disponibilizadas para apoiar os seus vários projetos de software. A descrição da organização deve ser feita de forma evolutiva, começando pelas informações com uso previsto a curto prazo ou que foram consideradas mais importantes pelos desenvolvedores de software da organização, sempre considerando a relação custo-benefício.

Responsáveis: Gerente de Conhecimento e especialistas das áreas envolvidas.

Sub-atividades:

Sub-atividade: Descrever Aspectos Gerais

O objetivo é obter informações que definem como a organização interage com o ambiente externo, registrando-as no Ambiente Configurado. Desta forma, pode ser relevante obter respostas para as seguintes perguntas:

- Qual é a missão da organização?
- Quais são os seus objetivos? Existe data prevista para que estes objetivos sejam atingidos?
- Quais são os serviços e/ou artefatos oferecidos? Quais são as empresas clientes destes serviços e/ou artefatos?
- Que projetos existem e que organizações estão envolvidas?

Sub-atividade: Descrever Estrutura Organizacional

O objetivo é obter informações sobre como a organização se encontra estruturada, registrando estas informações no Ambiente Configurado. Desta forma, pode ser relevante obter respostas para as seguintes perguntas:

- Quais são as unidades organizacionais existentes e como elas se relacionam?
- Quais são as posições existentes nas unidades organizacionais e como elas se relacionam? Que competências são requeridas? Quais são as pessoas que estão alocadas e que competências possuem?
- Existem comissões na organização? Com que propósito? Quem são as pessoas que fazem parte destas comissões?

Sub-atividade: Descrever Processos Organizacionais

O objetivo é obter informações sobre os processos executados na organização, representando-os graficamente no Ambiente Configurado. Desta forma, pode ser relevante obter respostas para as seguintes perguntas:

- Estes processos da organização obedecem a alguma norma? Qual?

- Quais são as atividades que compõem os processos organizacionais e como elas se relacionam?
- Quais são os insumos e produtos de cada atividade e que competências e recursos materiais e humanos são requeridos para a sua execução?
- Existe algum procedimento que indique como uma determinada atividade deve ser executada?

➢ Atividade: Descrever Organização Cliente

Esta atividade só deve ser executada se a organização tem como atividade de negócio o desenvolvimento e manutenção de software. O objetivo é obter e registrar informações sobre uma organização cliente, de forma que estas informações possam ser disponibilizadas para apoiar os vários projetos de software que envolvem esta organização. A descrição da organização cliente deve ser feita de forma evolutiva, considerando sempre a relação custo-benefício. O foco da atividade são as informações relevantes para os projetos de software da organização e não a obtenção de uma descrição completa da organização cliente. Além disso, as informações registradas no Ambiente Configurado sobre uma organização cliente evoluem a cada novo projeto envolvendo esta organização.

Responsáveis: Gerente de Conhecimento e especialistas em projetos com o cliente.

Sub-atividades:

Sub-atividade: Descrever Aspectos Gerais

O objetivo é obter informações que definem como a organização cliente interage com o ambiente externo, registrando-as no Ambiente Configurado.

Sub-atividade: Descrever Estrutura Organizacional

O objetivo é obter informações sobre como a organização cliente se encontra estruturada, registrando estas informações no Ambiente Configurado.

Sub-atividade: Descrever Processos Organizacionais

O objetivo é obter informações sobre os processos executados na organização cliente, representando-os graficamente no Ambiente Configurado.

As perguntas a serem respondidas em cada uma das sub-atividades acima podem ser extraídas das perguntas mencionadas nas sub-atividades de mesmo nome executadas para a própria organização.

➢ Adquirir Conhecimento Inicial

Esta atividade tem como objetivo disponibilizar, no Ambiente Configurado, itens de conhecimento sobre desenvolvimento e manutenção de software que já estavam disponíveis para a organização antes da configuração em algum formato, local e meio de armazenamento. Esta atividade deve ser executada segundo o processo de aquisição de conhecimento proposto por MONTONI (2003), de forma a possibilitar a filtragem de itens de conhecimento realmente relevantes, assegurar que a representação está adequada para a sua reutilização e que o conteúdo está claro e pode ser facilmente compreendido.

Responsáveis: Gerente de Conhecimento, Comitê de Avaliação e antigos detentores dos conhecimentos.