MARIA EDITH VILLELA PEDRAS

UMA FERRAMENTA DE APOIO AO GERENCIAMENTO DE DESENVOLVIMENTO DE *SOFTWARE* DISTRIBUÍDO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática pelo Curso de Pós-Graduação em Informática, do Setor de Ciências Exatas da Universidade Federal do Paraná, em convênio com o Departamento de Informática da Universidade Estadual de Maringá.

Orientadora: Dr.ª Elisa Hatsue Moriya Huzita

Co-orientadora: Dr.ª Tania Fatima Calvi Tait

CURITIBA

2003

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, da aluna *Maria Edith Villela Pedras* , avaliamos o trabalho intitulado, *"Uma Ferramenta de Apoio ao Gerenciamento de Desenvolvimento de Software Distribuído"*, cuja defesa foi realizada no dia 27 de agosto de 2003, às dez horas, no Anfiteatro B do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação da candidata. (Convênio número 279-00/UFPR de Pós-Graduação entre a UFPR e a UEM - ref. UEM número 1331/2000-UEM).

Curitiba, 27 de agosto de 2003.

Profª. Dra. Elisa Hatsue Moriya Huzita
DIN/UEM – Orientadora

Profª. Dra. Tania Fatima Calvi Tait
DIN/UEM - Orientadora

Profª. Dra. Rosângela Aparecida Dellosso Penteado
DC/UFSCAR – Membro Externo

Profª. Dra. Laura Sánchez Garcia
DINF/UFPR – Membro Interno

Dedico este trabalho
às minhas queridas filhas Cecília e Flávia
e ao meu amado Geziel.

AGRADECIMENTOS

Durante o decorrer destes anos de dedicação ao estudo, passei por grandes provações... Foram momentos difíceis, de desespero, de tristezas e de muitos medos... Em todos estes momentos a vontade foi "vou desistir"...

Agradeço aos meus pais que acreditam em mim,

Aos meus irmãos que me incentivaram a continuar,

À minha orientadora profa. Elisa que me estimulou,

À minha co-orientadora e amiga Tania pelos momentos de desabafo e pelas palavras de estímulo. Por me mostrar que as coisas não são tão complicadas,

À minha amiga Sueleni, por sua presença, estímulo e ajuda.

Ao Gabriel pela sua dedicação, empenho e grande ajuda,

À Vânia, Hamilton, Miriam e Márcia que, com sua experiência profissional, avaliaram a ferramenta desenvolvida,

À querida D. Chiquita, que sempre esteve presente e me ajudou em muitos momentos,

À todos os meus amigos que me ajudaram e em momentos de desânimo me disseram palavras de estímulo e consideração,

E a Deus, pois sem Ele eu não existiria.

Muito obrigada!

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

RESUMO

O desenvolvimento de *software* cada vez mais complexo, envolvendo a empresa e a Internet, exige um controle rápido e seguro do desenvolvimento de projetos. Com sistemas cada vez mais sofisticados e integrados, envolvendo áreas que requerem distribuição, adquirir ferramentas de apoio ao gerenciamento e desenvolvimento de *software* é uma necessidade. Ao tratar de *software* distribuído, a situação se torna mais complexa pela inexistência de ferramentas específicas de apoio ao desenvolvimento de *software* distribuído, principalmente com a abordagem de tratamento dos aspectos gerenciais.

Planejar e organizar as atividades, controlar e verificar se as atividades estão sendo executadas de acordo com o planejado, dentre outros, são elementos importantes no gerenciamento de projeto e desmerece-los pode contribuir para o fracasso dos projetos.

No *software* distribuído, também deve haver controle em termos de compartilhamento de recursos, abertura, concorrência, independência de escala, tolerância a falhas, transparência e estado compartilhado.

Dentro deste cenário, surge a necessidade de tratar os aspectos gerenciais no desenvolvimento de *software* distribuído, que culmina com a elaboração da ferramenta de apoio ao gerenciamento de desenvolvimento de *software* distribuído, DIMANAGER.

A ferramenta DIMANAGER teve como suporte para sua construção: a análise de ferramentas de apoio ao desenvolvimento de *software* disponíveis no mercado, as características do *software* distribuído e a abordagem de aspectos gerenciais.

É importante salientar que a ferramenta DIMANAGER é dinâmica, e mostra ao Gerente de Projeto a situação atual do projeto, com o objetivo de auxiliá-lo no acompanhamento e na tomada de decisão através da comparação entre o planejado e o executado, verificando a situação de cada atividade assim como das equipes envolvidas, dispersas geograficamente.

Palavras-chave: ferramentas *CASE*, desenvolvimento de *software*, gerenciamento e *software*.

ABSTRACT

The development of more and more complex softwares involving enterprises and Internet demands a rapid and safe development of projects. As the systems involving areas which require distribution are more and more sophisticated and integrated, it is necessary to acquire backing tools to manage and develop software.
Planning and organizing activities, controlling and checking if the activities are being executed according to what was planned, are some of important elements to manage a project and disregarding them may contribute to its failure.
At the distributed software there must also be control over shared resources, openning, concurrence, scale independence, fault tolerance, transparence and shared state.
In this scenery it is necessary to deal with the managing aspects in the development of the distributed software, which culminates with the elaboration of the backing tool DIMANAGER to manage and develop distributed software.
DIMANAGER tool had as supports for its construction: backing tools analysis for the development of softwares available in the market, distributed software characteristics and the approach with managing aspects.
It is important to emphasize that DIMANAGER tool is dynamic and shows to the Project Manager the present situation of the project, aiming to help him in accompanying and in taking decisions by comparing what was planned with what was executed, verifying the situation of each activity as well as the situation of the involving geographically dispersed teams.

Key-words: tools *CASE*, development of software, management and software.

Capítulo 1 Introdução

1.1 Considerações Iniciais

O desenvolvimento de *software* cada vez mais complexos, envolvendo a empresa e a Internet, exige um controle rápido e seguro do desenvolvimento de projetos.

Com sistemas cada vez mais sofisticados e integrados, envolvendo áreas que requerem distribuição, tais como: aplicações de trabalho cooperativo (CSCW – *Computer Supported Cooperative Work*), telemedicina; telecomunicação; robótica entre outras, adquirir ferramentas de apoio ao gerenciamento e desenvolvimento de *software* é uma necessidade. Lima Jr (2001) acrescenta que uma única pessoa não consegue entender tais sistemas completamente, surgindo então, a necessidade de formação de equipes para o gerenciamento da comunicação e da informação dentro delas. A equipe deve proporcionar ao usuário uma visão simplificada e uma interface adequada, minimizando a complexidade de sua estrutura. No entanto, o mercado oferece ferramentas integradas desenvolvidas por um único fornecedor. O grande desafio é que as ferramentas desenvolvidas por fornecedores distintos possam prover um mecanismo completo, eficiente e eficaz de gerenciamento de projeto de *software*, obedecendo a padrões de integração.

Lima Jr (2001) esclarece que, sem padrões, não se pode ter componentes de *software*, pois para que um componente possa ser acessível ou acessado, é necessário estabelecer regras básicas de comunicação. Dentre os padrões de arquitetura para processamento distribuído alguns podem ser citados, como por exemplo: CORBA (*Common Object Request Broker Architeture*) do OMG (*Object Management Group*) (OMG, 2000); Java/RMI e *JavaBeans* da *Sun Microsystems* (SUN, 1999) e DCOM (*Distributed Component Object Model*) da *Microsoft Co* (MSDCOM,2003). Além destas arquiteturas, algumas normas internacionais da ISO (*International Standards Organization*) e ITU-T (*International Telecommunication Union*) definem os quesitos esperados de um sistema aberto (Lima Jr, 2001).

As ferramentas de apoio ao gerenciamento de desenvolvimento de *Software* Distribuído devem proporcionar, segundo Pressman (2002), estimativas de custo, esforço e duração do projeto de *software*, assim como a definição de uma estrutura de divisão de trabalho, o planejamento de uma programação viável de projeto e acompanhamento em base contínua dos mesmos. A indicação da produtividade no desenvolvimento de *software* e da qualidade do produto são pontos de fundamental importância. As ferramentas de apoio ao gerenciamento do produto devem ainda, ser capazes de identificar os requisitos iniciais da proposta do cliente até o trabalho de desenvolvimento do *software* que implementará esses requisitos em um sistema a ser entregue.

De acordo com Informativo Técnico (2001), o gerente de projeto deve assegurar que o conjunto dinâmico de pessoas com diferentes interesses, filosofias, valores, abordagens e prioridades, envolvidas em um projeto, se torne coerente e permita desenvolver o trabalho de acordo com o planejamento e cronograma pré-estabelecidos.

Ao tratar de computação distribuída, que permite a conexão de computadores em redes internas e externas, um sistema de segurança com controle de senha, acessos e atualização de dados, também, é necessário para garantir que somente usuários autorizados tenham acesso aos recursos e às informações de forma confidencial e segura.

Neste contexto, o presente trabalho apresenta uma ferramenta de apoio ao gerenciamento de desenvolvimento de *software* distribuído, chamada DIMANAGER e está estruturado em 5 capítulos.

No segundo capítulo é apresentada a metodologia de desenvolvimento da ferramenta, colocando a forma como foi elaborada e encaminhada para validação junto a profissionais da área de informática.

O terceiro capítulo contém os elementos de apoio ao desenvolvimento de software, que compreende o estudo de: software distribuído, ferramentas *CASE* e *I-CASE* e gerenciamento de projeto de software, incluindo o estudo de três ferramentas de apoio ao desenvolvimento de software.

A ferramenta de apoio ao gerenciamento de desenvolvimento de *software*

distribuído, DIMANAGER, será apresentada no quarto capítulo, bem como o desenvolvimento do projeto lógico e a visualização da ferramenta propriamente dita.

Considerações sobre a ferramenta proposta quanto aos seguintes aspectos: gerenciamento de projeto de *software* distribuído, aspectos técnicos e organizacionais, as métricas, a validação, trabalhos futuros e as contribuições da pesquisa para o conhecimento serão abordados no quinto capítulo.

1.2 Cenário Atual

A evolução natural dos sistemas de informações nas duas últimas décadas foi significativa. Passamos de um ambiente centralizado onde era mais fácil administrar, controlar acesso, sintonizar o desempenho e prestar suporte técnico para ambientes distribuídos. O custo no ambiente centralizado era muito elevado para solucionar problemas de pequeno e médio porte, de forma que somente as grandes empresas eram capazes de pagar; a qualidade do serviço ficava abaixo das expectativas dos usuários, em virtude das limitações da tecnologia de interface dos aplicativos e, também, a falta de liberdade do usuário para manipular seus dados. Com a evolução tecnológica de redes e o surgimento de softwares robustos para comporem a infra-estrutura dos ambientes distribuídos, tais como sistemas operacionais e sistemas gerenciadores de bancos de dados, muitos problemas presentes no ambiente anterior foram solucionados.

O uso de modernas tecnologias de informação, como redes internas e externas, assim como as telecomunicações, tornam as distâncias cada vez menores. Além disso a crescente globalização envolve organizações estabelecidas em diferentes pontos do mundo e com políticas particulares de tomada de decisão. Estes fatores fazem com que as empresas procurem agilizar suas atividades através da distribuição de tarefas em locais dispersos geograficamente. O aumento significativo de reuniões por teleconferência, correio eletrônico, *e-commerce*, internet e intranet, entre outros, são prova desta nova realidade. Especificamente na área de engenharia de *software* esta busca de distribuição de tarefas para agilizar os resultados, otimizando recursos e

pessoas torna-se uma necessidade crescente e pesquisas na área demonstram a forma de realização deste compartilhamento e seus benefícios Huzita (1995). As recentes tendências do mercado têm mostrado que a complexidade do *software* continuará a crescer dramaticamente nas próximas décadas, implicando a necessidade de estabelecer/utilizar metodologias apropriadas ou definir um processo adequado.

Com a explosão da Internet, possibilitando a integração de milhões de pessoas, múltiplos servidores heterogêneos interagindo com freqüência e distribuídos geograficamente, o desenvolvimento de *software* distribuído tornou-se uma necessidade.

Primeiramente, as pesquisas voltaram-se para aspectos estritamente técnicos para tratar o desenvolvimento de *software* distribuído, como a forma de comunicação e o hardware necessário para essa comunicação. Estudos referentes a sistemas distribuídos têm evoluído devido à busca de inovações, dos avanços tecnológicos e da utilização de aplicações cada vez mais complexas. Neste contexto, o controle rápido e seguro do andamento dos projetos tornou-se necessário, pois, além do envolvimento da empresa, a Internet também se faz presente.

Os projetos têm sido cada vez mais responsáveis pela implementação das estratégias de negócio, pela lucratividade das empresas e pela obtenção e manutenção de vantagens competitivas. No caso específico dos projetos de sistemas de informação, eles têm desempenhado um papel fundamental em toda e qualquer mudança organizacional. Em decorrência dessa crescente importância para as organizações, a Gerência de Projetos de Software tem evoluído muito em seus processos, práticas, experiências, métodos e ferramentas, sendo que para o profissional de informática, adquirir esses conhecimentos modernos tornou-se um fator crítico de sucesso para a sua carreira.

Atualmente, busca-se o desenvolvimento de um ambiente completo que possibilite o tratamento do *software* distribuído, considerando tanto os aspectos técnicos como os gerenciais, tratados apropriadamente na área de sistemas de informação (Tait, 2000), cuja vinculação com o processo de desenvolvimento de *software* se torna preponderante em uma visão organizacional e integradora de

processos.

1.3 Objetivo Geral

Apresentar uma ferramenta que possibilite controlar e acompanhar o desenvolvimento de projetos de *software*, viabilizando ao gerente de projeto coordenar e controlar as atividades e as pessoas envolvidas no desenvolvimento de um *software* distribuído.

1.4 Objetivos Específicos

- Avaliar algumas ferramentas de apoio ao gerenciamento de desenvolvimento de *software*;
- Integrar os aspectos de controle de projetos ao *software* distribuído;
- Identificar os aspectos gerenciais vinculados ao desenvolvimento de um *software* distribuído;
- Estudar e selecionar os métodos de pesquisa de campo para validação da ferramenta;
- Elaborar questionário para aplicação na validação da ferramenta;
- Validar a ferramenta para o gerenciamento de *software* distribuído.

1.5 Motivação

A Universidade Estadual de Maringá desenvolve um Projeto de Pesquisa voltado à área de objetos distribuídos inteligentes, fundamentado no trabalho "Metodologia para desenvolvimento baseada em objetos distribuídos inteligentes" Huzita (1999). Este projeto tem dois objetivos principais: 1) definir a construção de uma metodologia para desenvolvimento de *software* baseado em objetos distribuídos; 2) construir um ambiente integrado para o desenvolvimento de *software* distribuído considerado por Pascutti (2002).

A construção de uma metodologia para desenvolvimento de *software* baseado em objetos distribuídos, denominada MDSODI, foi apresentada por Gravena (2000).

A arquitetura do ambiente proposta por Pascutti (2002) oferece o suporte necessário através de ferramentas e tecnologias adequadas para o desenvolvimento de *software* distribuído. Em paralelo, vários trabalhos estão sendo desenvolvidos para integrar-se a este ambiente, podendo ser citados: uma ferramenta de *groupware* para apoio à definição de requisitos para a MDSODI (Batista, 2003); uma ferramenta de suporte à persistência de artefatos para o DiSEN (Moro, 2003) e uma ferramenta de apoio ao gerenciamento de desenvolvimento de *software* distribuído, apresentada neste trabalho.

Para Tait (2000), planejar constitui-se em uma atividade de extrema importância em qualquer área onde se pretenda alcançar determinados objetivos, desde planejamentos simples, como o domiciliar, até planejamentos complexos para garantir a competitividade das empresas, principalmente com a utilização da tecnologia para garantir esta competitividade.

O controle e acompanhamento do projeto de desenvolvimento de *software* distribuído são pontos de fundamental importância que visam a produtividade e a qualidade do produto final.

Em concordância, Lima Jr (2001) coloca que sistemas cada vez mais complexos e integrados exigem ferramentas de apoio ao gerenciamento e desenvolvimento de *software*, no entanto, o mercado oferece ferramentas integradas desenvolvidas por um único fornecedor, sendo o grande desafio desenvolver ferramentas que possam prover um mecanismo completo, eficiente e eficaz de gerenciamento de projeto de *software*, obedecendo aos padrões de integração.

O tratamento dos aspectos gerenciais no desenvolvimento de *software* distribuído, muitas vezes negligenciados, interferem diretamente na qualidade do produto final. Os aspectos gerenciais no *software* distribuído a serem observados são controles que possibilitam conhecer a situação do projeto em desenvolvimento, tais como: sua compatibilidade com os recursos orçados; a dimensão de pessoas/horas trabalhadas; a distribuição de tarefas; a execução das etapas propostas dentro do

projeto, o planejamento de uma programação viável de projeto e acompanhamento em base contínua dos mesmos, enfim, as tarefas que um gerenciamento e controle de projetos devem tratar para gerenciar o desenvolvimento de *software* distribuído.

Neste contexto, apresentar uma ferramenta que possibilite controlar e acompanhar o desenvolvimento de projetos de *software*, viabilizando ao gerente de projeto coordenar e controlar as atividades e as pessoas envolvidas no desenvolvimento de um *software* distribuído contribuirá significativamente para o desenvolvimento de projeto de *software* distribuído.

1.6 Limitação do Trabalho

O trabalho proposto aborda temas como: aspectos gerenciais para desenvolvimento de *software* distribuído, processo de gerenciamento e planejamento de projetos e ferramentas de gerenciamento de desenvolvimento de *software*, constituindo o alicerce para a elaboração de uma ferramenta dentro da filosofia de *software* distribuído.

Dois elementos adicionais, considerados em gerenciamento de projetos, não foram tratados pela ferramenta, trata-se de custos, no qual a ferramenta não realiza controle algum, e das métricas.

Neste trabalho, apenas alguns aspectos sobre a identificação das métricas são mencionadas tendo sido tratados na relação horas/homem. O usuário não terá acesso à definição das técnicas de estimativas disponibilizadas no mercado. Para a execução de um projeto estas variáveis são importantes e os custos financeiros envolvem tanto os recursos humanos quanto os materiais. Estes recursos podem fazer parte do planejamento do gerente, mas não estão sendo controlados pela DIMANAGER. Ambos merecem um estudo mais aprofundado.

Capítulo 2 Metodologia de Desenvolvimento do Trabalho

2.1 Introdução

A metodologia proposta envolveu as seguintes etapas: elementos de apoio ao desenvolvimento de *software*, elaboração do projeto da ferramenta, desenvolvimento da ferramenta e validação da ferramenta e pode ser visualizada na Figura 2.1.

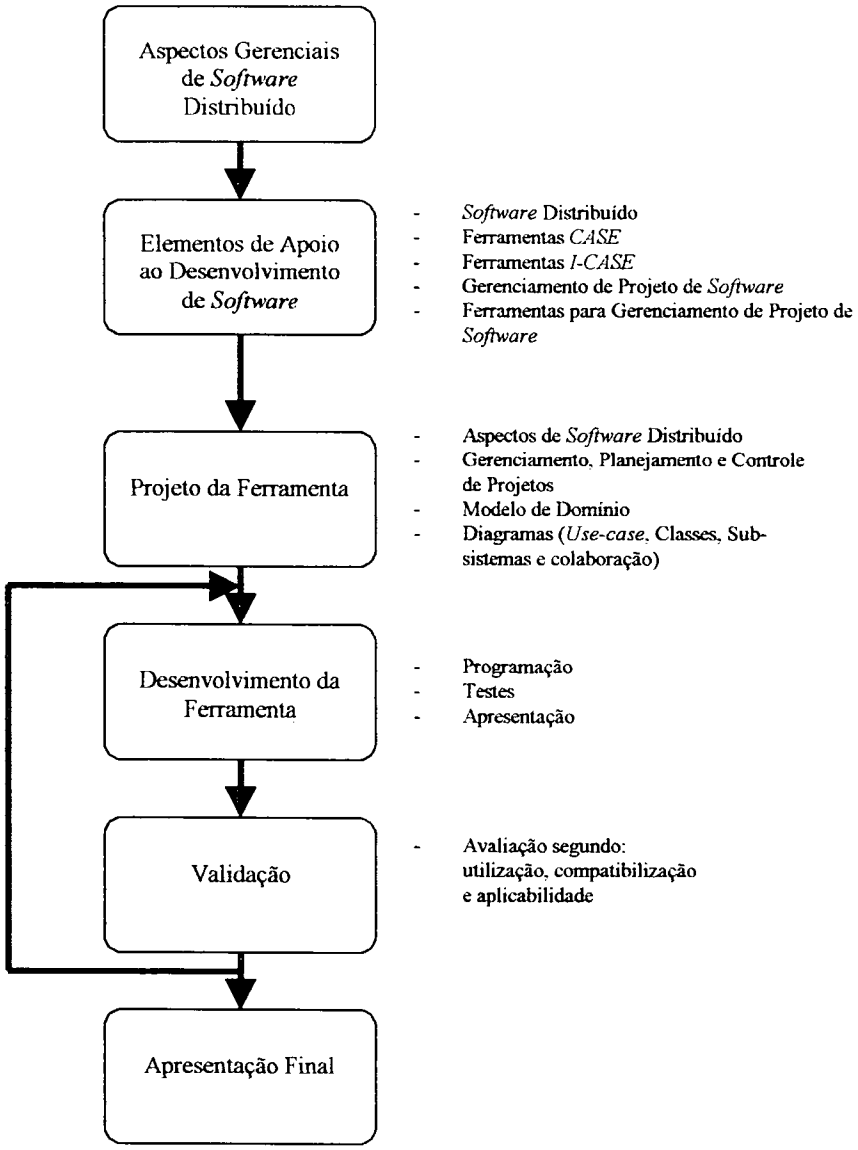

FIGURA 2.1 – ESTRUTURA DA PESQUISA

2.1.1 Fundamentação Teórica

A fundamentação teórica compreendeu o estudo dos elementos de apoio ao desenvolvimento de software nos seguintes temas:

- processo de gerenciamento, planejamento e controle de projetos de forma geral;
- aspectos gerenciais no desenvolvimento de *software*;
- *software* distribuído;
- ferramentas para gerenciamento de projeto de *software*.

Ao final do estudo foram feitas breves considerações sobre as ferramentas estudadas, o que possibilitou realizar o levantamento de pontos a serem considerados em uma ferramenta para gerenciamento de *software* distribuído.

2.1.2 Elaboração do Projeto da Ferramenta

Nesta fase, a partir dos aspectos de *software* distribuído e de gerenciamento e controle de projetos abordados nos elementos de apoio fundamentação teórica se desenvolveu o projeto lógico da ferramenta.

Projetou-se a interface da ferramenta; contendo a estrutura dos dados a serem utilizados pela mesma e escolhida a linguagem de programação.

A linguagem de programação escolhida foi *Java* (*Sun Microsystems*), versão 1.4.2. Esta linguagem tem como características principais: orientada a objetos, portabilidade total (JVM – *Java Virtual Machine*), possibilita a implementação de aplicações distribuídas, é independente de plataforma e multi-thread (concorrência), gera programas para serem executados na Internet ou em uma máquina individual, entre outras.

2.1.3. Desenvolvimento da Ferramenta

A fase de desenvolvimento da ferramenta envolveu a programação do projeto da ferramenta propriamente dita. O projeto da ferramenta considerou as interfaces com

as demais etapas e ferramentas do projeto de pesquisa "Metodologia para desenvolvimento baseada em objetos distribuídos inteligentes" Huzita (1999), ao qual este projeto está vinculado, com a finalidade de obter um ambiente de *software* distribuído.

Para a elaboração dos modelos desenvolvidos na DIMANAGER foi utilizada uma ferramenta de modelagem visual, a Rational Rose (*Rational Software Corporation*), com base na notação da UML (*Unified Modeling Language*), (Furlan, 1998), que permite ao desenvolvedor definir uma arquitetura de *software*.

2.1.4 Validação da Ferramenta

A ferramenta DIMANAGER foi apresentada aos profissionais da área de informática de quatro empresas da cidade de Maringá, sendo duas do setor público, uma do setor privado e outra do ramo de cooperativa, para avaliação dos seguintes aspectos: utilização, compatibilização e aplicabilidade.

Para esta avaliação utilizou-se uma entrevista com o preenchimento de um questionário (Anexo IV) e a apresentação da ferramenta DIMANAGER como métodos de validação.

Capítulo 3 Elementos de Apoio ao Desenvolvimento de *Software*

Este capítulo aborda assuntos que devem ser considerados para a proposta de desenvolvimento de uma ferramenta de apoio ao gerenciamento de desenvolvimento de *software* distribuído. Através de conceitos e definições a proposta da ferramenta foi estruturada visando o gerenciamento de projetos. Desta maneira, são abordadas as ferramentas *CASE* e *I-CASE*. Inicialmente, um aspecto importante a ser tratado é o *software* distribuído, devido a sua vinculação dada por esta pesquisa com os aspectos de gerenciamento.

3.1 *Software* Distribuído

Segundo Barreto (2003), um sistema distribuído consiste de uma coleção de computadores autônomos ligados por uma rede de computadores e equipados com *software* de sistema distribuído. Por outro lado, o *software* distribuído coordena atividades e compartilha os recursos disponíveis (*hardware*, *software* e dados).

Pressman (2002) coloca que, atualmente, o *software* desempenha um duplo papel: o produto e, ao mesmo tempo, o veículo para entrega do produto. Como produto, ele está presente no computador e, mais amplamente, numa rede de computadores acessível pelo *hardware* local, podendo estar residente desde o telefone celular até em um computador de grande porte, sendo, desta forma, responsável pela transformação da informação. E, como veículo para entrega do produto: o *software* age como uma base para controle do computador, presente nos sistemas operacionais; para a comunicação da informação, presente nas redes; e para a criação e o controle de outros programas, presente nas ferramentas e ambientes de *software*.

Neste contexto, vários fatores têm contribuído para o aumento significativo de sistemas distribuídos e de acordo com Borghoff (2000) podemos destacar: o custo dos processadores, das unidades de armazenamento e da tecnologia de rede com altas taxas de transmissão, tem baixado drasticamente nos últimos anos; aplicações das mais variadas áreas, como por exemplo, gerenciamento de fluxo de trabalho, informações

corporativas, telecooperação, e outros, estão crescendo. Estas aplicações, em contraste com o tradicional sistema centralizado, providenciam o uso de interfaces e o suporte cooperativista de forma bastante satisfatória entre os usuários geograficamente dispersos.

Lima Jr (2001) acrescenta que os sistemas distribuídos têm adquirido uma importância crescente na indústria da computação principalmente com a ampla difusão da Internet. As arquiteturas para o desenvolvimento de aplicações evoluíram para arquiteturas a objetos, onde aplicações são vistas não mais como blocos monolíticos de *software*, mas sim como coleções de componentes de *software* que cooperam para implementar a funcionalidade requerida da aplicação. Deste modo, programadores podem lidar de uma forma mais eficaz com a complexidade inerente de aplicações do mundo real. Os componentes de um sistema assim construído não precisam estar localizados necessariamente na mesma máquina, não precisam ser escritos na mesma linguagem de programação e podem estar sendo executados sobre sistemas operacionais diferentes em máquinas diferentes (PCs, estações de trabalho, Macintoshes, etc). Neste contexto, *software* geograficamente distribuído faz com que os elementos do *software* trabalhem de forma transparente, ou seja, como se estivessem em uma única máquina.

Uma série de propriedades interessantes dos sistemas distribuídos, são apresentados por Borghoff (2000), como a existência de unidades funcionais múltiplas, que permitem, por exemplo, a transmissão dinâmica entre diferentes recursos, independente do tipo de execução. Estas unidades funcionais são distribuídas e ao mesmo tempo interconectadas através de uma rede. Os controles do sistema operacional integram e homogenizam os componentes nestas unidades. A comunicação entre as unidades funcionais proporciona sincronização e garantia de consistência entre as partes. Outras características de sistemas distribuídos são: a autonomia cooperativa durante a integração entre as unidades funcionais físicas e lógicas, a independência durante a falha de alguma unidade e um alto grau de transparência. Outro requisito a ser considerado é o econômico, devido à utilização de recursos, como dispositivos de *swap*, que barateiam o produto mas aumentam a

complexidade do mesmo.

Por outro lado Bessani (2000) enumera algumas razões que justificam a construção de sistemas distribuídos como sendo: pessoas e informações são distribuídas, pois os sistemas distribuídos são compostos por estações de trabalho. Os usuários que trabalham nestas estações compartilham informações e recursos (impressoras); performance/custo; modularidade: como a interface entre os componentes deve ser melhor definida do que em sistemas centralizados a modularidade nos sistemas distribuídos facilita o gerenciamento e a manutenção destes sistemas; expansibilidade: é mais fácil de ser incrementado com novos módulos devido à sua grande modularidade; disponibilidade: a redundância de recursos e serviços pode ser implementada de forma mais fácil, pois os sistemas distribuídos são baseados em diversos computadores; e, confiabilidade: um sistema distribuído é baseado em diversas máquinas e, a queda de uma delas não deve afetar o sistema. É possível construir um sistema tolerante a falhas com recuperação automática das informações.

Assim como existem razões que justificam a construção de sistemas distribuídos descritos acima, Bessani (2000) também apresenta as características desta classe de sistemas que definem aspectos importantes e fundamentam as principais razões para a grande complexidade dos mesmos. São elas:

- Compartilhamento de recursos: o compartilhamento de recursos, como impressoras, discos de armazenamento e dados é uma das principais vantagens dos sistemas distribuídos;
- Abertura: oferece maiores possibilidades para extensão, tanto em relação ao hardware, como periféricos, memória ou interfaces de comunicação, quanto em relação ao *software*, como módulos, protocolos e serviços. Como exemplo pode-se citar o padrão CORBA;
- Concorrência: uma característica natural deste sistema é a concorrência de processos, pois podemos ter mais de um processo sendo executado ao mesmo tempo. A complexidade aumenta na proporção em que mais elementos são incluídos no sistema;

- Independência de escala: teoricamente um sistema distribuído com 10 máquinas deve funcionar tão bem quanto um sistema com 100 máquinas, pois os elementos são distribuídos na rede.
- Tolerância à falhas: o sistema deve tolerar falhas e continuar operando sem produzir resultados incorretos. Duas abordagens devem ser consideradas: a redundância de hardware , com o uso de componentes duplicados e a recuperação de *software*, através de programas que recuperam os dados;
- Transparência: o usuário não percebe que o sistema está distribuído em uma rede de computadores, para ele, o processo ocorre em uma única grande máquina;
- Estado compartilhado: todos os componentes do sistema devem ser compartilhados, a fim de que a falha em um deles não cause a perda de informações sobre o estado do sistema.

A construção de *software* distribuído não é fácil, sendo em geral, bastante complexa e de difícil desenvolvimento, pois a interconexão de uma série de componentes pode causar comportamentos estranhos no sistema. Segundo Bessani (2000), os problemas tendem a se agravar de acordo com alguns fatores, tais como: o número de componentes interligados, a interferência de componentes externos e a falha dos mesmos com a propagação dos efeitos, que podem causar graves conseqüências no sistema.

3.2 Ferramentas *CASE*

Para Pressman (2002) uma boa oficina de engenharia de *software* deve ter três características essenciais: 1. uma coleção de ferramentas úteis que ajudem em cada passo da construção de um produto; 2. um *layout* organizado que possibilite que as ferramentas sejam encontradas rapidamente e usadas eficientemente; 3. um artesão habilitado que entenda como usar as ferramentas efetivamente. Os engenheiros de *software* reconhecem que necessitam de um maior e mais variado número de ferramentas, denominado *CASE*, e uma oficina eficiente e organizada para armazenar estas ferramentas, que o autor denomina de *ambiente integrado de suporte a projetos*.

Mason, Buda e Gazeta (2002) definem *CASE* (*Computer-Aided Software Engineering* - Engenharia de *software* auxiliada por computador) como uma ferramenta do ambiente de suporte do engenheiro de *software*, permitindo-se automatizar atividades manuais e melhorar a qualidade antes mesmo de o produto ser produzido. Em linhas gerais, as ferramentas *CASE* implementam um ambiente relativamente refinado no qual diversas atividades de especificação ou codificação são apoiados por recursos computacionais.

Pressman (2002) acrescenta que as tecnologias *CASE* compreendem uma ampla variedade de tópicos que abrangem métodos de engenharia de *software* e procedimentos de gerenciamento de projetos.

Mason, Buda e Gazeta (2002) dividem as ferramentas *CASE* em ambientes simples (*Lower CASE*) que dão suporte à codificação, testes e depuração; e, em ambientes complexos (*Upper CASE*), que automatizam diversas tarefas de análise e projeto de sistemas. E Pressman (2002) classifica estas ferramentas por função, pois desempenha o papel de instrumento para gerentes e pessoal técnico, pelo uso através de sua utilização nas várias etapas do processo de engenharia de *software* e pela arquitetura de ambiente (*hardware* e *software*) que as suporta ou até mesmo pela origem ou custo delas.

Tanto Mason, Buda e Gazeta (2002) quanto Pressman (2002) subdividem as ferramentas *CASE* em: ferramentas de planejamento de sistemas comerciais; ferramentas de gerenciamento de projetos (ferramentas de planejamento de projetos, rastreamento de requisitos e de métricas e gerenciamento); ferramentas de apoio (ferramentas de documentação, de *software* básico, de garantia de qualidade e de banco de dados e SCM-*Software Configuration Management*); ferramentas de análise e projeto; ferramentas de programação; ferramentas de integração de testes; ferramentas de prototipação; ferramentas de manutenção e ferramentas de estrutura.

O Artigo Técnico Dr. *CASE* 3.0 (2001) acrescenta que as técnicas de análise e projeto de sistemas são complexas e demoradas e que o uso de ferramentas *CASE* supera a baixa produtividade. Este artigo apresenta grandes vantagens no uso destas ferramentas, destacando-se:

- Maior praticidade no uso das técnicas de modelagem de dados, já que modelar sistemas complexos sem o apoio de ferramentas gráficas é uma tarefa das mais ingratas. Este fato explica a grande resistência ao uso de metodologias no passado;
- Difusão do uso das técnicas de projetos de sistemas, pois em um ambiente *CASE*, a facilidade de uso das técnicas de projeto favorece a adoção destas técnicas pelos desenvolvedores;
- Simplificação do trabalho de projeto de sistemas. As ferramentas *CASE* oferecem vários recursos de apoio às tarefas de análise;
- Diminuição do número de erros no projeto. A ocorrência de erros diminui pois existem recursos de consistência e, principalmente, de automação de várias etapas do projeto;
- Existe um ganho real em qualidade e produtividade através da automação. Tarefas com um elevado grau de dificuldade e caráter repetitivo são resolvidas em segundos pelas ferramentas *CASE*. Se este tipo de tarefa for executada por humanos, a probabilidade de erros e a demanda de tempo aumentam;
- Diminuição do número de profissionais necessários ao projeto já que o desenvolvimento de *software* através da utilização de ferramentas *CASE* fica mais simples e a produtividade aumenta;
- Liberação dos desenvolvedores para se concentrarem nas partes mais criativas do desenvolvimento. Como muitas tarefas são automatizadas com o uso de ferramentas *CASE*, o projetista deve se concentrar mais na análise de requisitos e na validação do projeto junto ao cliente.

Mason, Buda e Gazeta (2002) acrescentam ainda que um dos principais problemas apresentados nas empresas é a manutenção nos sistemas. O uso de ferramentas *CASE*, juntamente com as Metodologias de Desenvolvimento, proporcionam aumento de produtividade facilitando a sua manutenção.

Em contrapartida, os autores acima citados expõem que em uma pesquisa realizada em países europeus descobriu-se que a grande maioria das organizações não utiliza ferramenta *CASE* e apenas algumas as utilizaram em um ou dois projetos, pois a grande maioria dos analistas desconhece a existência das mesmas. Aqui no Brasil,

Sobrino (2002) ressalta que as ferramentas *CASE* também são muito pouco conhecidas e que há um considerável número de empresas de tecnologia que sequer adotam qualquer tipo de metodologia para o desenvolvimento e conseqüente documentação de seus sistemas – e, portanto, não seria lógico esperar que adotassem uma ferramenta *CASE* para auxiliar, de alguma forma, na elaboração de seus projetos. O uso de ferramentas *CASE* não funciona caso a empresa não faça uso de uma metodologia para o desenvolvimento de seus sistemas.

Apesar de apresentarem um significativo aumento de produtividade da qualidade dos sistemas, da documentação, da padronização e redução no tempo de entrega do produto, as ferramentas *CASE* ainda continuam não sendo utilizadas, devido a fatores como: custo elevado e treinamento das mesmas, falta de conhecimento e de orientação técnica, grande resistência dos grupos de trabalho, falta de metodologias para desenvolvimento devido à facilidades das ferramentas de quarta geração.

3.3 Ferramentas *I-CASE*

O verdadeiro poder do *CASE* só pode ser medido mediante *integração*. E Pressman (2002) coloca quatro necessidades básicas do *CASE integrado* (*I-CASE*): transferir harmoniosamente as informações (modelos, programas, documentos, dados) de uma ferramenta para outra e de uma etapa da engenharia de *software* para a seguinte; reduzir o esforço exigido para realizar atividades de gerenciamento de configuração de *software*, garantir a qualidade e produção de documentação; aumentar o controle do projeto, que é obtido por meio de um melhor planejamento, monitoração e comunicação; coordenar melhor os membros de uma equipe que esteja trabalhando num grande projeto de *software*.

Por outro lado, o *I-CASE* apresenta grandes desafios, pois a integração exige uma representação consistente das informações de engenharia de *software* onde as interfaces devem ser padronizadas entre o engenheiro de *software* e cada uma das ferramentas, assim como uma abordagem efetiva que possibilite o *I-CASE* se

movimentar entre as diversas plataformas de hardware e sistemas operacionais.

Zarrella (2003) em seu artigo "*CASE* Tool Integration and Standardization", completa que estes fatores que complicam a integração de ferramentas no ambiente de desenvolvimento precisam de ações para resolvê-los e, neste contexto, esforços estão sendo feitos para solucionar técnica e metodologicamente os problemas, criando padrões para as ferramentas de integração, sendo esta a base da evolução das ferramentas *CASE*.

De acordo com Pressman (2002) a integração das ferramentas *CASE* deve permitir:

- Intercâmbio de dados - a maioria das ferramentas deve permitir exportar as informações coletadas para outra ferramenta, evitando, desta forma, que erros tipográficos sejam introduzidos desnecessariamente;
- Acesso comum a ferramentas - este nível de integração deve possibilitar ao usuário a utilização de diferentes ferramentas de maneira semelhante, por exemplo, através de um menu *pull-down* no gerenciador de janelas do sistema operacional;
- Gerenciamento de dados comum - um banco de dados único deve ser mantido para conter os dados de várias ferramentas. Este banco de dados pode estar fisicamente centralizado ou distribuído simplificando o intercâmbio de informações e melhorando a integridade dos dados compartilhados, visto que cada ferramenta tem acesso imediato às últimas informações. As facilidades de gerenciamento de versão devem estar disponíveis e os direitos de acesso devem ser controlados;
- Compartilhamento de dados - as estruturas de dados e semântica devem ser compatíveis para que haja a troca de informações sem a necessidade de tradução;
- Interoperabilidade - as ferramentas que possuem a característica de compartilhar dados assim como ter acesso comum às ferramentas devem ser *interoperáveis*;
- Integração plena - dois aspectos devem ser considerados: gerenciamento de metadados e uma facilidade de controle. Os dados produzidos por ferramentas *CASE* individuais geram informações denominadas metadados, que são: definições de objetos; relações e dependências entre objetos de granularidade arbitrária; normas de projeto de *software*; procedimentos e eventos. As facilidades de controle

devem permitir que as ferramentas individuais informem o restante do ambiente sobre eventos significativos e permitam o envio de pedido de ações a outras ferramentas e serviços por meio de *triggers*.

Ainda, segundo Pressman (2002) a arquitetura de integração é um conjunto de informações de engenharia de *software* criado para facilitar a transferência de informações, conforme Figura 3.1.

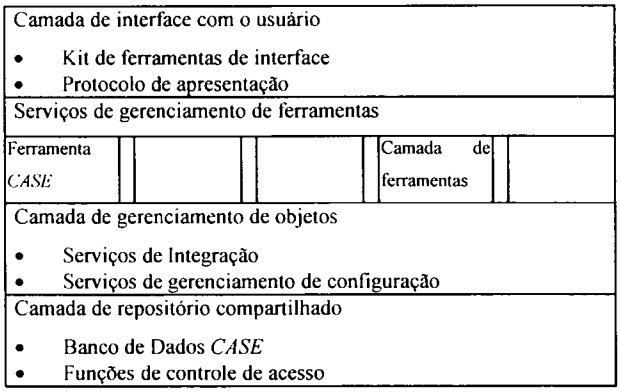

FIGURA 3.1 – MODELO DE ARQUITETURA DA ESTRUTURA DE INTEGRAÇÃO (PRESSMAN, 2002)

A camada de interface com o usuário deve oferecer os caminhos entre as ações praticadas por ele e as ferramentas contidas no ambiente, fornecendo um mecanismo consistente para a comunicação entre a interface e as ferramentas *CASE* individuais. As padronizações de *layout* de tela, nomes e organizações de menus, ícones, nomes de objetos, uso do teclado e do mouse e o mecanismo para acesso às ferramentas são definidos como parte do protocolo de apresentação. A camada de ferramentas é um mecanismo de gerenciamento de ferramentas para controlar o comportamento das ferramentas dentro do ambiente. No uso de multitarefas para a execução de uma ou mais ferramentas os *serviços de gerenciamento de ferramentas* (Tools Management Services - TMS) se responsabilizam pela sincronização e comunicação das multitarefas, coordenando o fluxo de informações do repositório e do sistema de gerenciamento de objetos para as ferramentas, realizam funções de segurança e auditoria e compilam métricas sobre o uso de ferramentas. A camada de

gerenciamento de objetos (Object Management Layer - OML) é responsável pela integração das ferramentas com o repositório *CASE*, além de oferecer serviços de configuração identificando todos os objetos de configuração, controlar versões e oferecer suporte para o controle de mudanças, auditorias e relatos de status. A camada de repositório compartilhado é um banco de dados para armazenar as informações e o controle para interação entre os dados e a camada de gerenciamento de objetos.

Mason, Buda e Gazeta (2002) apresentam a integração das ferramentas *CASE* , como pode ser visto na Figura 3.2.

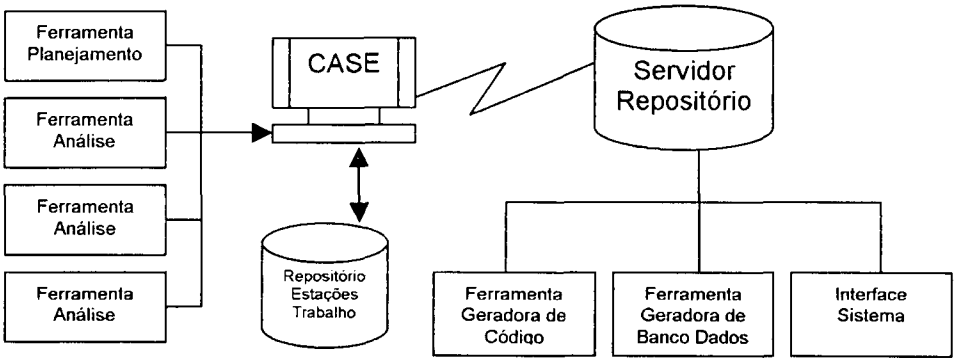

FIGURA 3.2 – INTEGRAÇÃO DAS FERRAMENTAS *CASE* (MASON, BUDA E GAZETA, 2002)

A implementação da integração das ferramentas *CASE* depende da arquitetura, da plataforma e da filosofia do projetista do ambiente, mas todos os ambientes *CASE* implementam mecanismos de execução e mecanismos de comunicação. Os mecanismos de execução iniciam um processo a partir de seu contexto estático e oferecem características para suspender, retomar e terminar um processo. Os mecanismos de comunicação são responsáveis pela comunicação entre os processos e estabelecem filas de mensagens possibilitando a comunicação entre as ferramentas. Em ambientes distribuídos, um mecanismo de distribuição deve possibilitar que os mecanismos de execução e de comunicação sejam distribuídos ao longo da rede e oferecer as seguintes capacidades: administrar e supervisionar todas as estações de trabalho ligadas à rede; gerenciar cada nó da rede; e, distribuir de forma transparente as funções de execução e comunicação.

3.4 Gerenciamento de Projeto de *Software*

O gerenciamento de projeto tem como objetivo assegurar que processos estabelecidos sejam seguidos, coordenando e monitorando as atividades da engenharia de *software*. O gerenciamento de projetos é pouco abordado e praticado e as equipes de desenvolvimento não conseguem medir o desempenho dos projetos, verificar pontos falhos, estipular melhorias e outros (Oliveira, Rouiller e Vasconcelos, 2001).

Por outro lado, muitas dificuldades são inerentes ao gerenciamento de projeto de *software*, tais como: dinamicidade do processo: alterações constantes nos planos de projetos, redistribuição de atividades, inclusão/exclusão de atividades, adaptação de cronogramas, realocação de recursos, novos acordos com os clientes, entregas intermediárias não previstas, entre outros; adaptação ao ambiente, dependendo da disponibilidade de recursos, ferramentas e habilidade do pessoal ou equipe; por ser uma atividade criativa e intelectual o desenvolvimento de *software* é difícil de ser medido e controlado; grande quantidade de variáveis envolvidas como metodologias, modelos de ciclo de vida, técnicas, ferramentas, tecnologia, recursos, atividades e outras; cada processo de *software* é único, devendo-se, portanto, manter as características de cada cliente. As dificuldades no gerenciamento de projeto de *software* aumentam quando se trata de *software* distribuído, pois as características de um ambiente assim constituído devem ser administradas supervisionando todas as estações de trabalho ligadas à rede, gerenciando cada nó e, distribuindo de forma transparente as funções de execução e comunicação.

Teixeira (2001) coloca que o gerenciamento do projeto de *software* abrange desde o planejamento até a fase de teste, documentação e implantação do sistema. O gerenciamento efetivo de um projeto de *software* depende totalmente do seu planejamento e, nesta fase deve-se definir antecipadamente o quê e como fazer, quando e quem vai executar as tarefas. Sem objetivos claros e definidos, orçamento e cronograma realistas, a base para o comprometimento do gerente com o projeto não existe e, portanto, há pouca esperança de se chegar a algum lugar.

Neste contexto, desenvolver ferramentas para gerenciamento de projeto de

software é complexo pois interfere na produtividade e qualidade do produto final.

3.4.1 Ferramentas para Gerenciamento de Projeto de *Software*

Vários estudos estão sendo realizados por engenheiros de *software* para desenvolver ferramentas que agilizem o processo de criação, garantam técnicas de segurança para o trabalho em grupo e a integração dos dados, estabeleçam políticas de manutenção e reuso de componentes (Pfaffenseller, Pfaffenseller e Kroth, 2001). As ferramentas se preocupam com o desenvolvimento do *software*, envolvendo qualidade, segurança de dados e uma boa documentação e verifica-se o empenho dos pesquisadores em desenvolver ferramentas que se preocupam também com a evolução do projeto obedecendo o cronograma estabelecido (Pfaffenseller, Pfaffenseller e Kroth, 2001), (Lima, Reis e Nunes, 1998), (Oliveira, Rouiller e Vasconcelos, 2001).

No presente trabalho, inicialmente, três ferramentas de apoio ao desenvolvimento de *software* serão apresentadas: a TOOLManager, centrada em processo; uma Ferramenta de Apoio ao Gerenciamento de Componentes, voltada a componentes; e a ferramenta para Gerenciamento do Processo de Desenvolvimento Cooperativo de *Software* no Ambiente PROSOFT, também centrada em processo.

Estas ferramentas foram escolhidas por abranger aspectos gerenciais no desenvolvimento de projetos e também serem o resultado de pesquisas relevantes no mercado brasileiro.

3.4.1.1 ToolManager: Uma Camada de Gerenciamento de Ferramentas *CASE* a um Ambiente Centrado no Processo

O processo de desenvolvimento de *software* envolve atividades, pessoas, recursos e ferramentas necessárias para produzir *software* e precisa ser modelado através de linguagens de processo, para que os mesmos possam ser avaliados e melhorados. Neste contexto, viu-se a necessidade do desenvolvimento de uma camada gerenciadora de ferramentas a um ambiente centrado no processo, permitindo o

controle da ativação de ferramentas a serem utilizadas nesse ambiente segundo variantes da sua utilização. (Oliveira, Rouiller e Vasconcelos, 2001).

A camada gerenciadora de ferramentas apresentada é a ToolManager, que junto ao Ambiente de Desenvolvimento de *Software* (ADS) trabalha para melhor sustentar o processo de desenvolvimento de *software*. O ambiente de Gerenciamento de Projeto de *Software* Centrado no Processo escolhido para desenvolver esta camada gerenciadora de ferramentas é o ProjectSpace, que tem como objetivo a criação de um ambiente para monitoração e gerenciamento de projetos de *software*. Para que o projeto possa produzir um produto e/ou serviço de acordo com seus requisitos, o ProjectSpace deve ser capaz de identificar, estabelecer, coordenar e monitorar todas as atividades, tarefas e recursos necessários.

A ToolManager contém um conjunto de Ferramentas *CASE* que trabalham juntas de acordo com um conjunto de dados e protocolos. Foi idealizada para integrar-se ao Ambiente ProjectSpace e ser capaz de gerenciar as questões referentes ao uso, registro e controle dessas Ferramentas durante a execução das Atividades definidas em um Plano de Projeto. A Figura 3.3 mostra a relação entre o ProjectSpace e a ToolManager.

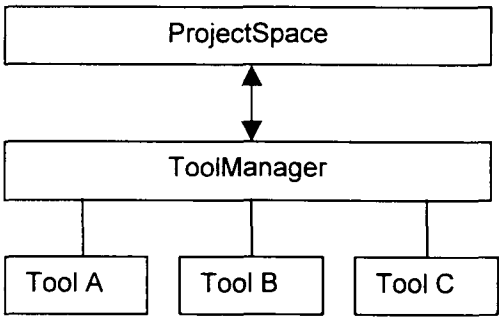

FIGURA 3.3 – RELAÇÃO ENTRE O PROJECTSPACE E A TOOLMANAGER (OLIVEIRA, ROUILLER E VASCONVELOS, 2001)

Quatro níveis de gerenciamento e alguns componentes são requeridos para que a ToolManager controle e ative todas as ferramentas necessárias às diversas formas de administração dos serviços solicitados no ProjectSpace, sendo assim distribuídos: quatro níveis de gerenciamento (apresentação, controle, atividade e dados), responsáveis para controlar os serviços que a camada se dispõe a gerenciar; um

conjunto de ferramentas responsável pelas ferramentas *CASE* quanto à sua utilização pelo ProjectSpace; um fluxo de mensagem que controla a passagem de informações entre os diferentes níveis, facilitando assim a comunicação entre eles.

O ProjectSpace e a ToolManager se relacionam para controlar o registro das Ferramentas *CASE* definidas para uso durante a execução de seus planos de projeto; para permitir que tais ferramentas possam ser ativadas de acordo com condições próprias entre elas; para gerar e acompanhar o progresso de execução das Atividades através do histórico de uso das ferramentas.

A Figura 3.4 mostra o limite entre o ProjectSpace e a ToolManager. Os eventos de integração entre eles são: receber, na ToolManager, as informações sobre a Atividade a ser executada que caracteriza a ativação de uma Ferramenta *CASE* específica; enviar, através da ToolManager, ao ambiente a informação do estado do desenvolvimento da Atividade, em função do histórico de uso das ferramentas; armazenar e manipular documentos gerados pelas ferramentas *CASE* em forma de Arquivo e LOG's (históricos) quanto a sua utilização/manipulação gerados pela ToolManager sobre o que já foi executado ou não; a partir do ProjectSpace, registrar as informações das Ferramentas *CASE* a serem utilizadas e suas Associações com as Atividades para as quais se destinam à execução; viabilizar o conjunto de Atividades definidas e registradas previamente no ambiente a fim de permitir a construção de uma estrutura contendo as condições de execução entre elas; permitir o acesso a uma Base de Dados comum.

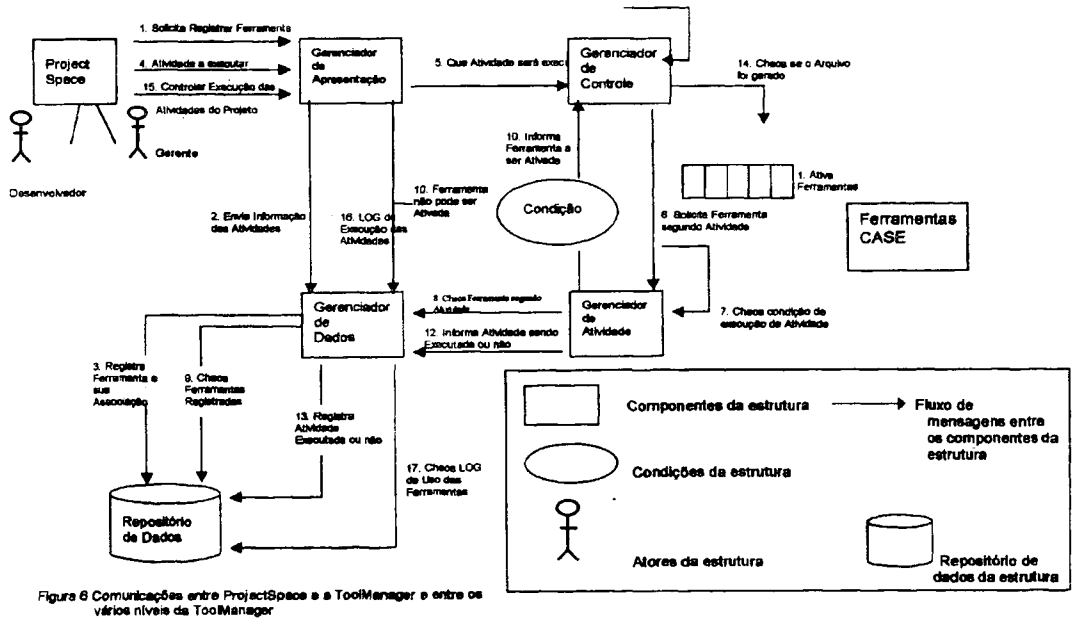

Figura 6 Comunicações entre ProjectSpace e a ToolManager e entre os vários níveis da ToolManager

FIGURA 3.4 – COMUNICAÇÕES ENTRE O PROJECTSPACE E A TOOLMANAGER E ENTRE OS NÍVEIS DA TOOLMANAGER (OLIVEIRA, ROUILLER E VASCONVELOS, 2001)

Os atores que interagem na arquitetura proposta são o Gerente e a própria ProjectSpace. O Gerente deve acompanhar o andamento da execução das Atividades definidas no Plano do Projeto à partir do LOG (histórico) gerado pela ToolManager; registrar as Ferramentas *CASE* que serão utilizadas pelo Projeto e suas Associações com as Atividades para determinar a que se destinam, e a ProjectSpace deve centralizar todas as informações referentes às Atividades definidas no Plano do Projeto e, conforme a necessidade do Desenvolvedor, solicitar que uma Atividade seja desenvolvida ativando assim determinadas Ferramentas *CASE*, de acordo com as condições de execução.

A ToolManager deve registrar, remover e alterar as ferramentas; registrar as ferramentas com as atividades a serem executadas; disponibilizar a visibilidade das ferramentas registradas; permitir a execução da atividade; ativar ferramentas; gerar LOG (histórico) de Uso das Ferramentas; e, acompanhar a execução das atividades do Plano do Projeto.

3.4.1.2 Uma Ferramenta de Apoio ao Desenvolvimento de *Software* Baseado em Componentes

O processo de desenvolvimento de *software* é complexo e consome grandes recursos pessoais e financeiros. Preocupado com o gerenciamento do processo de desenvolvimento de *software*, Pfaffenseller, Pfaffenseller e Kroth (2001) apresentam uma ferramenta de apoio ao gerenciamento de componentes que agiliza o processo de criação do *software*, propicia técnicas de segurança para o trabalho em grupo, com políticas de manutenção e ferramentas de auxílio ao reuso do *software* além de facilitar a visualização de hierarquias entre classes de componentes, com suas funções, características e fundamentação de cada item.

As principais características desta ferramenta são: implementar técnicas de armazenamento e busca de componentes e aprimorar o reuso das soluções já desenvolvidas; apresentar a cadeia de dependências entre os componentes, permitir visualizar suas ligações e verificar se alguma alteração pode afetar outros componentes; armazenar informações sobre cada componente gerenciado; gerenciar as alterações ocorridas nos componentes; gerenciar as versões dos componentes; definir políticas de manutenção e controlar os acessos aos componentes com níveis de permissões aos desenvolvedores.

O repositório de componentes armazena: informações técnicas que são úteis para manutenções e análise dos componentes (nome do componente, nome e local do arquivo, classificação, criador e data de inclusão); informações adicionais que são as informações estruturais dos componentes como os eventos, métodos, propriedades (atributos) e dependências efetuadas através de análise do componente durante sua inclusão no repositório ou sua atualização; palavras-chave: utilizadas para busca e recuperação de componentes; exemplo de utilização: o autor do componente descreve um exemplo da utilidade do mesmo para facilitar a compreensão das funcionalidades por parte dos desenvolvedores; código fonte: para cada versão do componente, antiga ou atual, criada no repositório armazenam-se os arquivos do mesmo para que seja possível a atualização nas máquinas dos desenvolvedores assim como a restauração de

versões antigas; e, classificação de componentes: para organizar a base de pesquisa os componentes são separados de acordo com o tipo de função para o qual ele foi criado. Alguns padrões foram estabelecidos, tais como: componentes de estrutura, de banco de dados, de interface, para desenvolvimento distribuído e de domínio do problema.

Os desenvolvedores têm acesso a este repositório para obter informações referentes aos componentes, permitindo dessa forma a sua utilização e objetivando o seu reuso, através de métodos de pesquisa de componentes existentes.

A representação dos componentes existentes é outra funcionalidade oferecida pelo repositório. Compreende a identificação, a exibição da cadeia hierárquica, os atributos e as funções de cada componente. A diagramação formal é a ferramenta adequada para que haja troca de informações entre os vários desenvolvedores que trabalham em conjunto no desenvolvimento de um sistema.

É importante citar que a relação de dependência entre os componentes deve ser mostrada ao desenvolvedor para que ele saiba quais os componentes que podem afetar o seu funcionamento assim como os componentes que podem ser afetados por utilizarem-no, para que eventuais alterações não afetem o funcionamento do sistema.

Esta ferramenta possui o modelo serial para controlar as versões, pois proporciona maior segurança no gerenciamento das atualizações dos componentes entre o repositório e a cópia local do desenvolvedor.

O modelo controla a atualização das versões na base de componentes local do desenvolvedor, providencia o bloqueio do mesmo quando este está sendo modificado por algum elemento do grupo e remove o bloqueio do arquivo liberando-o para outros desenvolvedores ao término da modificação, atualizando o componente no repositório.

Para evitar que alterações freqüentes com o mesmo objetivo gerem muitas versões foi criado o conceito de *Etiquetas*, onde várias alterações são gravadas, mantendo sempre as informações da última atualização com a versão atual. O desenvolvedor poderá liberar o componente com uma nova versão quando o objetivo desta *Etiqueta* for atingido.

Dois usuários com papéis distintos são definidos na política de manutenção: o administrador e o desenvolvedor. O administrador é responsável por gerenciar o

trabalho dos desenvolvedores e definir regras, direitos e permissões que permitam controlar as atividades pré-definidas, remover componentes, voltar qualquer versão, atribuir *Etiquetas* globais, além de poder executar todas as tarefas do desenvolvedor. O desenvolvedor é responsável para executar as tarefas de bloqueio, remover bloqueio, adicionar componente, voltar versão anterior e atribuir *Etiqueta* privada.

3.4.1.3 Gerenciamento do Processo de Desenvolvimento Cooperativo de *Software* no Ambiente PROSOFT

O principal objetivo do PROSOFT é apoiar o desenvolvimento formal de *software*, providenciando a integração dos dados, controle e de apresentação entre suas ferramentas. (Lima, Reis e Nunes, 1998) Os componentes do PROSOFT são denominados pela sigla ATOs – Ambientes de Tratamento de Objetos. Figura 3.5

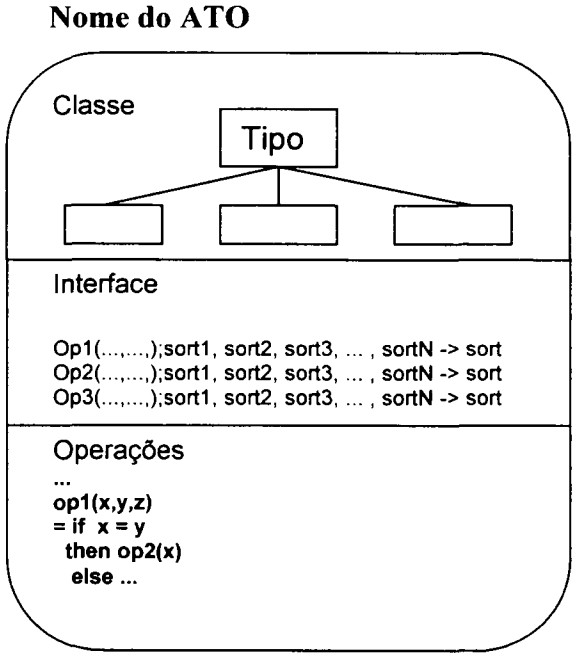

FIGURA 3.5 – ESTRUTURA DE UM ATO (LIMA, REIS E NUNES, 1998)

O ATO representa um tipo abstrato de dados. Compõe-se de um nome e na sua definição são especificados a sua classe (instanciação), a interface das operações e a especificação (semântica) das operações que atuam sobre os objetos dessa classe.

Os ATOs se comunicam através de troca de mensagens feita pela ICS – Interface de Comunicação de Sistema, que é responsável pela comunicação entre o PROSOFT e os ATOs. O desenvolvimento de um sistema envolve um ou mais ATOs, e cada um deles se preocupa apenas com os termos do sort (tipo) por ele definido.

A Figura 3.6 mostra um modelo esquemático da comunicação dos ATOs através da ICS. Quando ocorre uma chamada ICS, o objeto do ATO chamado é transformado pela operação definida na sua interface.

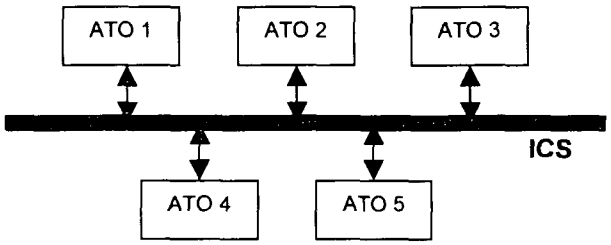

FIGURA 3.6 – ESTRUTURA DO PROSOFT (LIMA, REIS E NUNES, 1998)

O PROSOFT é um ambiente homogêneo (fechado) e reconhece somente os componentes (ATOs) desenvolvidos sob o seu modelo.

Cada objeto do PROSOFT armazena o registro da última alteração, o nome do objeto, o estado que o objeto se encontra, o nome do objeto ancestral (se houver), a lista de versões derivadas, o histórico recente das últimas modificações realizadas no objeto (para permitir "desfazer" as modificações), as especificações do ATO a que pertence e a descrição textual do objeto.

O processo de produção de *software* envolve atividades complexas desempenhadas por pessoas. A modelagem e a execução de processos de *software* são de fundamental importância para o aumento da qualidade do produto de *software* e têm sido tema de estudos pela comunidade de Engenharia de *Software* com a finalidade de construir ferramentas que as suportem.

Várias informações são necessárias para integrar um modelo de processo de *software* sendo, portanto, necessário uma linguagem de modelagem do processo do *software* que ofereça recursos para descrever e manipular os passos do processo.

Questões como a coordenação de múltiplos usuários, assim como a integração entre as ferramentas automatizadas e os desenvolvedores devem ser consideradas na fase de execução de modelos do processo de *software*.

O gerenciador de processos deve ser capaz de coordenar as atividades realizadas por pessoas e as ferramentas automatizadas e se preocupar com as seguintes questões:

- Automação de Processo: ativar automaticamente atividades sem intervenção humana, através de uma integração com as ferramentas do ambiente;
- Trabalho Cooperativo: suportar a coordenação e cooperação de pessoas trabalhando em um projeto de *software*.
- Monitoração: providenciar visões diferentes da execução do processo, permitindo que o gerente de projetos obtenha informações sobre o andamento correto das atividades;
- Registro da história do processo: coletar dados da evolução do processo para permitir que o processo melhore onde houver necessidade, e seja corrigido para atender novos requisitos.

Alguns requisitos de execução devem ser satisfeitos pelo mecanismo de execução e pelo ambiente de desenvolvimento que suportam o processo para que o gerenciador de processo possa realizar seu objetivo, como:

- Fluxo de Controle: a seqüência de atividades no processo deve ser representada e obedecida;
- Iteração: algumas atividades podem necessitar de repetição e a ativação automática dessas atividades é tarefa do mecanismo de execução;
- Interação humana: a interface com os desenvolvedores deve ser adequada e permitir que o *feedback* necessário seja assegurado;
- Gerência da Informação: o armazenamento persistente de todos os dados envolvidos no processo, com controle de acesso e gerência de versões;
- Métricas: a coleção automática de dados sobre o processo e o produto;
- Adaptação dinâmica: a mudança do processo durante a sua execução;

- Execução de ferramentas: suporte à interação com vários tipos de ferramentas do ambiente;
- Restrições e alocação de recursos: quem está trabalhando com o que e que recursos são necessários para quais atividades.

Durante a execução de processos de *software* são manipuladas informações sobre o próprio processo e seu estado. Isto significa que uma das tarefas do mecanismo de execução é manter o estado do processo descrito internamente consistente com o seu estado real. Além disso, os documentos, códigos e quaisquer informações sobre os produtos também devem ser manipulados pelo mecanismo de execução.

O uso de ambientes orientados a processo necessitam de uma definição rigorosa da execução do processo, permitindo uma melhora significativa de comunicação entre as pessoas envolvidas e a consistência do que está sendo feito. Como conseqüência, o treinamento de novos usuários é facilitado, pois o estudo do processo fornece uma visão do funcionamento da organização. O ambiente sempre fornece informações que guiam o desenvolvedor a realizar o seu trabalho com maior eficiência, podendo ser inclusive "liberado" de algumas tarefas repetitivas. As definições do processo são reunidas em uma biblioteca, permitindo o reuso por outros usuários sem muito esforço.

Alguns requisitos são considerados indispensáveis a um ambiente de desenvolvimento de *software*, sendo eles:

- Suporte a múltiplos usuários: mecanismos para atender vários usuários ao mesmo tempo requisitando serviços do ambiente;
- Gerência de objetos: controle do acesso e da evolução de objetos compartilhados. As versões dos documentos e produtos de *software* devem ser gerenciadas para permitir cooperação e consistência;
- Gerência de comunicação entre pessoas: as pessoas que estarão envolvidas no desenvolvimento de *software* devem ter acesso a mecanismos de comunicação, tais como mensagens eletrônicas e conferência eletrônica;
- Gerência de cooperação: a edição cooperativa de documentos e itens de *software* deve ser gerenciada pelo ambiente de forma que os usuários envolvidos obtenham comunicação síncrona sobre os produtos que estão manipulando;

- Gerência de processo: o suporte à modelagem e execução do processo de *software*. Dentro deste requisito, encontra-se a necessidade de um formalismo de modelagem de processo e uma máquina de execução das definições de processo;
- Extensibilidade: permitir extensão do ambiente através da inclusão de novas ferramentas, sejam elas de apoio ao desenvolvimento de *software* ou com outras funções;
- Integração entre todos os módulos: todos os níveis de integração devem estar disponíveis para viabilizar os outros requisitos.

Estes requisitos estão assim distribuídos na arquitetura PROSOFT: de extensibilidade e integração entre os módulos são identificados na arquitetura do PROSOFT Convencional; o de suporte a múltiplos usuários é provido pelo PROSOFT Distribuído; as gerências de objetos, comunicação e cooperação pelo PROSOFT Cooperativo; e a gerência de processos é provida pelo Gerenciador de Processos de *Software* do PROSOFT.

Um processo de desenvolvimento é um conjunto de atividades. As operações sobre um processo são refletidas diretamente nas suas atividades. As atividades de um processo podem ter sub-atividades, mas em qualquer nível de granulosidade possuem as mesmas características básicas. O ATO Atividade armazena informações importantes e necessárias para o gerenciador de processos, sendo seus principais componentes descritos a seguir:

- Agentes: são responsáveis pela execução da atividade que serão notificados pelo GP quando a atividade estiver pronta para começar;
- Produtos envolvidos: um conjunto de produtos de entrada (que serão consumidos pela atividade) e um conjunto de produtos de saída (produzidos pela mesma).
- Condições: a execução de uma atividade pode estar condicionada à existência de um produto ou ao estado de um objeto cooperativo. Assim, uma atividade pode ter uma pré-condição, avaliada antes da execução, e uma pós-condição, avaliada depois da sua execução;
- Estados: cada atividade pode estar em um dos seguintes estados: esperando, pronta, parada e completa. O controle da transição de estados das atividades é

automatizado, não sendo permitida sua manipulação por qualquer usuário do ambiente;

- Recursos: o conjunto de recursos alocados quando a atividade está ativa;
- Cronograma: com datas de início e fim para cada atividade;
- *Script*: o *script* envolve a execução das atividades sendo composta pelos elementos: descrição informal (texto produzido pelo projetista do processo); operação automática (atividade é executada por uma ferramenta do PROSOFT); e processo (decomposição da atividade).

O ambiente gerenciador de processos é composto por 10 ATOs PROSOFT construídos especificamente para ele e outros ATOs disponíveis no ambiente.

3.4.2 Considerações sobre as Ferramentas

Não existe uma ferramenta que seja única para gerenciar o projeto de *software*, mesmo assim, a produtividade na construção destes teve um aumento significativo quando poderosas ferramentas foram incorporadas às habilidades humanas. Oliveira, Rouiller e Vasconcelos (2001) afirmam que o gerenciamento é composto por um conjunto de ferramentas em um Ambiente de Desenvolvimento de *Software* (ADS) e que em cada fase do gerenciamento do projeto de *software* aparecem ferramentas capazes de auxiliar o processo. Estas ferramentas devem ser capazes de compartilhar informações sobre os vários projetos, pois usuários precisam utilizar informações geradas por outra ferramenta. O Quadro 1 demonstra as principais características das três ferramentas apresentadas.

QUADRO 1 – BREVE COMPARAÇÃO ENTRE AS FERRAMENTAS

	ToolManager	**Apoio ao Desenvolvimento de Componentes**	PROSOFT
Foco	Processo	Componentes	Processo
Ambiente de Gerenciamento de *Software*	Project Space	Não especifica	Prosoft
Objetivo	Gerenciar questões referentes ao uso, registro e controle das ferramentas *CASE* durante a execução das atividades definidas em um plano de projeto.	Integrar ferramentas de modelagem, controlar versões, trabalho em grupo e políticas de manutenção abrangendo novos conceitos como componentes e reuso de *software*.	Apoiar o desenvolvimento formal de *software*, providenciando a integração dos dados, do controle e de apresentação entre suas ferramentas.
Composição da Ferramenta	• Conjunto de ferramentas; • Fluxo de Mensagem.	• Repositório de componentes; • Biblioteca de Componentes; • Controle de versões.	Composta por ATO's – Ambientes de Tratamento de Objetos.
Camadas	**Níveis de gerenciamento:** • Gerenciador de Apresentação; • Gerenciador de Controle; • Gerenciador de Atividade; • Gerenciador de Dados;	Não especifica	• PROSOFT Convencional; • PROSOFT Distribuído; • PROSOFT Cooperativo; • GP - Gerenciador de Processo de *Software* do PROSOFT.
Mensagens	Fluxo de mensagem que controla a passagem de informações entre os diferentes níveis de gerenciamento, facilitando a comunicação entre eles.	O fluxo de mensagem ocorre através da diagramação formal, definida como sendo a ferramenta adequada para que haja troca de informações entre os vários desenvolvedores que trabalham em conjunto no desenvolvimento de um sistema.	Os ATOs se comunicam através de troca de mensagens feita pela ICS – Interface de Comunicação de Sistema, responsável pela comunicação entre o PROSOFT e os ATOs.
Atores envolvidos	• Gerente • ProjectSpace	• Administrador do repositório • Desenvolvedor	• Super-usuário • Gerentes • Usuários-comuns

Embora cada uma delas tenha sido desenvolvida em ambientes diferentes e o foco tenha sido no processo e em componentes, nota-se que todas se preocupam em acompanhar o projeto, mantendo um fluxo de mensagem entre as camadas, a integração e o controle das ferramentas envolvidas. A estrutura de trabalho de cada uma é diferente mas todas se preocupam em manter um controle de versões garantindo o acompanhamento do projeto.

Os atores envolvidos têm características diferentes, mas desempenham funções semelhantes. Nos três casos somente um dos atores é o responsável por gerenciar o trabalho dos demais, definir regras, direitos e permissões que possibilitam controlar as atividades pré-definidas. Os outros atores são responsáveis pelo desenvolvimento mas, no caso da ToolManager, o ambiente ProjectSpace centraliza as informações referentes às atividades definidas no Plano do Projeto e ativa ferramentas *CASE* conforme a necessidade do Desenvolvedor.

Cada uma das ferramentas apresentadas foi desenvolvida para facilitar o processo de desenvolvimento de *software*, propiciando técnicas de segurança,

integração entre os grupos de trabalho e o acompanhamento do projeto através do gerenciamento das atividades estabelecidas no Plano de Projeto.

Capítulo 4 Uma Ferramenta de Apoio ao Gerenciamento de Desenvolvimento de *Software* Distribuído

4.1 Introdução

Baseado no gerenciamento de questões referentes ao uso, registro, controle e integração das ferramentas *CASE* durante a execução das atividades definidas em um plano de projeto, assim como o controle de versões, o trabalho em grupo, as políticas de manutenção e o apoio ao desenvolvimento formal de *software* (Pfaffenseller, Pfaffenseller e Kroth, 2001); (Lima, Reis e Nunes, 1998) e (Oliveira, Rouiller e Vasconcelos, 2001) foi desenvolvida Uma Ferramenta de Apoio ao Gerenciamento de Desenvolvimento de *Software* Distribuído com um enfoque diferenciado ao tratar *software* em sistemas distribuídos.

Portanto, a ferramenta se baseia em alguns aspectos relevantes, tais como:

- Planejar, fixando objetivos e as alternativas para atingi-los, inclusive o provimento dos recursos necessários, humanos e materiais;
- Organizar as atividades necessárias para atingir os objetivos fixados;
- Controlar, verificando se as atividades realizadas estão sendo executadas de acordo com o planejamento estabelecido. Esta atividade está diretamente relacionada ao planejamento e a essência do controle é a simplicidade;
- Alocar pessoal: distribuindo as equipes de trabalho para cada atividade estabelecida;
- Coordenar, relacionando convenientemente as atividades às equipes de trabalho envolvidas em cada processo, assegurando a cooperação de todas as equipes envolvidas em uma atividade, não podendo ser separado das outras funções de gerenciamento.

Algumas características importantes foram levadas em consideração na elaboração da ferramenta, como: o *software* distribuído, o uso das informações gerenciais e seu uso através da Internet.

Neste contexto, Uma Ferramenta de Apoio ao Gerenciamento de

Desenvolvimento de *Software* Distribuído, além de estar preparada para atender os requisitos apresentados, é dinâmica, e mostra ao Gerente de Projeto a situação atual do projeto, tendo como objetivos: acompanhar o desenvolvimento do projeto através da comparação entre o planejado e o executado; verificar a situação de cada atividade assim como das equipes envolvidas; controlar as versões de cada atividade e os aspectos inerentes ao desenvolvimento de sistemas distribuídos apresentados no capítulo 3.

4.2 Uma Ferramenta de Apoio ao Gerenciamento de Desenvolvimento de *Software* Distribuído

A Ferramenta de Apoio para Gerenciamento de Desenvolvimento de *Software* Distribuído apresentada neste trabalho oferece apoio à Metodologia para Desenvolvimento de *Software* Distribuído (MDSODI) apresentado por Gravena (2000) e foi denominada DIMANAGER – *Distributed Software Manager*.

A metodologia MDSODI leva em consideração algumas características apresentadas em sistemas distribuídos, tais como: paralelismo/concorrência, comunicação, sincronização e distribuição, durante todo o projeto, desde a fase inicial até a fase final.

Em se tratando de sistemas distribuídos, os produtos de *software* desenvolvidos são cada vez mais complexos. Esta metodologia surgiu para facilitar o desenvolvimento do projeto por etapas que buscou aliar este conceito a um processo de desenvolvimento de *software* que propiciasse a utilização de objetos distribuídos, assim como, o desenvolvimento de componentes de *software* reusáveis.

Neste contexto, um processo e uma metodologia de desenvolvimento de *software* tornaram-se um estudo importante que contribuíram para o desenvolvimento da MDSODI: *Unified Process* (Furlan, 1998) e a *MOOPP*, apresentada por Huzita (1995).

O *Unified Process*, é um processo de desenvolvimento de *software* que, através de um conjunto de atividades transforma os requisitos do usuário em um produto de

software Furlan (1998). E a *MOOPP* é uma metodologia orientada a objetos para desenvolvimento de *software* para processamento paralelo, tratando os aspectos estáticos e dinâmicos do sistema, e o paralelismo já nas fases iniciais do desenvolvimento de *software*.

A MDSODI mantém as principais características da *Unified Process*: dirigida a *use-case*; centrada na arquitetura; desenvolvimento iterativo e incremental; e baseada em agentes/componentes. E, também, utiliza algumas características propostas na *MOOPP*, como por exemplo, a representação gráfica para os possíveis tipos de objetos e classes, tendo como base o paralelismo.

A Figura 4.1 mostra o modelo de ciclo de vida da MDSODI e a Figura 4.2 as fases de cada incremento do ciclo de vida.

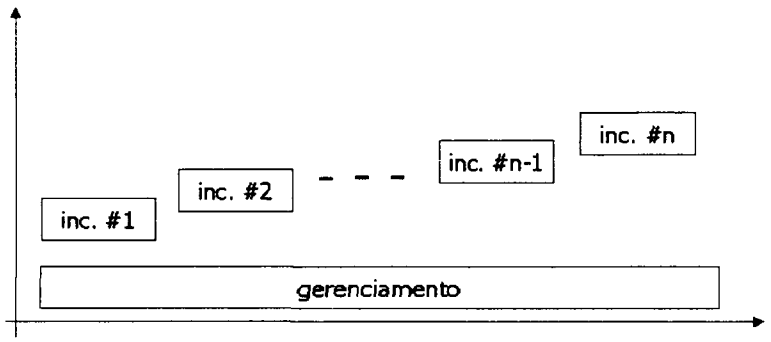

FIGURA 4.1 – CICLO DE VIDA DA MDSODI

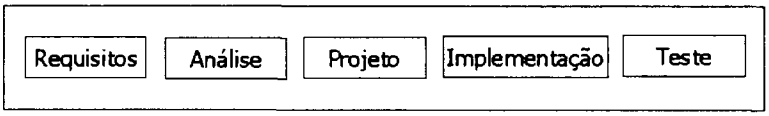

FIGURA 4.2 – FASES DE CADA INCREMENTO DO CICLO DE VIDA DA MDSODI

As fases do processo de desenvolvimento de *software* da MDSODI estabelecidos por Gravena (2000) são:

- Requisitos: esta fase tem como objetivo principal, a partir das necessidades do usuário, identificar as funcionalidades necessárias para o desenvolvimento do sistema de forma adequada e eficiente. Deve-se entender os requisitos funcionais e não funcionais do sistema através de entrevistas com o solicitante do produto para que através das informações recebidas o sistema a ser desenvolvido se aproxime de suas necessidades. Os artefatos produzidos são: o modelo de domínio para ajudar o

entendimento sobre o contexto do sistema; primeira versão do diagrama de *use-cases*, sem considerar implementação; diagrama de colaboração, a partir do modelo de *use-case* proposto após a avaliação do mesmo.

- Análise: esta fase consiste em analisar os requisitos descritos na fase anterior, refinar a estrutura verificando quais são realmente importantes e quais precisam ser acrescentados. Com base nos requisitos identificados e no diagrama de *use-case* elaborado, definem-se as classes e objetos do sistema, fornecendo uma visão mais definida do sistema. Os requisitos devem ser analisados para se verificar a existência de redundâncias e inconsistências. Deve-se: definir os atributos e operações das classes identificadas; analisar os *use-cases* definidos em termos das classes identificadas, realizando-se as alterações necessárias para uma maior consistência; elaborar o diagrama de classes; identificar aspectos de concorrência/paralelismo, distribuição e comunicação no diagrama de classes; reavaliar o diagrama de *use-case* de acordo com as novas definições de classes, realizando-se as alterações necessárias; analisar classes e objetos identificados para se certificar da consistência, e, sendo necessário realizar modificações; dividir o diagrama de *use-cases* em pacotes, de forma a facilitar o gerenciamento do sistema; identificar aspectos do paralelismo, distribuição e sincronização entre pacotes através de descrição textual; e, elaborar o diagrama de pacotes considerando os aspectos de sistemas distribuídos.
- Projeto: nesta fase molda-se o sistema incluindo também os requisitos não funcionais e outras considerações não identificadas nas fases anteriores, como por exemplo: linguagem de programação, interface com o usuário, reuso de componentes, entre outras. Neste momento a análise do diagrama de *use-cases* deve se preocupar com a localização dos *use-cases* e atores em subsistemas diferentes e questões de concorrência/paralelismo, e necessidades de implementação. A análise do diagrama de classes deve levar em consideração a concorrência e a localização de métodos, classes e objetos. O diagrama de seqüência deve se preocupar com os aspectos de comunicação entre os diferentes objetos do sistema, paralelismo e sincronização. É importante listar os requisitos de

implementação identificados anteriormente através de descrição textual, como: linguagem e ambiente de programação escolhido, localização física, e questões de concorrência entre métodos. O sistema deve ser dividido em camadas, para um melhor gerenciamento dos aspectos funcionais e não funcionais, podendo ser feito da seguinte forma: camada de aplicação específica, relacionando os aspectos funcionais específicos para cada um dos aspectos genéricos identificados; camada de aplicação genérica: relacionando aspectos mais gerais e funcionais do *software*; camada *middleware*: relacionando aspectos não funcionais, como exemplo podemos citar: os tipos de serviços oferecidos, os aspectos de concorrência/paralelismo, entre outros; camada de sistema de *software*: identificando configurações de redes necessárias para a comunicação entre as demais camadas. Os algoritmos devem ser detalhados na implementação, com base nos métodos identificados no diagrama de classes.

- Implementação: esta fase tem como objetivo principal a construção/implementação do sistema, tendo como base aspectos identificados nas fases de requisitos, análise e projeto. Deve-se verificar os aspectos identificados no projeto no que se refere a implementação (linguagem de programação utilizada e geração de código). As interfaces entre os subsistemas identificados na fase de projeto também devem ser definidas. Os aspectos da arquitetura devem ser considerados, acrescentando-se a divisão dos subsistemas de projeto, suas interfaces e dependências entre os mesmos. Para isso utiliza-se como entrada a visão arquitetural da fase de projeto, bem como os subsistemas e camadas definidos neste modelo. Os métodos das classes identificadas nas fases anteriores devem ser detalhadas e implementadas. E, deve-se identificar mecanismos de sincronização (exclusão mútua, monitores) bem como os mecanismos a serem utilizados para tratar do balanceamento de carga entre os nós de processamento, considerando os requisitos de distribuição, paralelismo, sincronização e comunicação identificados no projeto.
- Testes: Gravena (2000) destaca a importância dessa fase em processo de desenvolvimento de *software* a fim de se atingir um produto de qualidade.

A partir da MDSODI foi desenvolvido um ambiente de desenvolvimento de

software – ADS, chamado *DiSEN* (*Distribuited Software Engineering Environment*). Pascutti (2002)

O estilo adotado no *DiSEN* é em camadas proporcionando uma maior flexibilidade de desenvolvimento. Figura 4.3

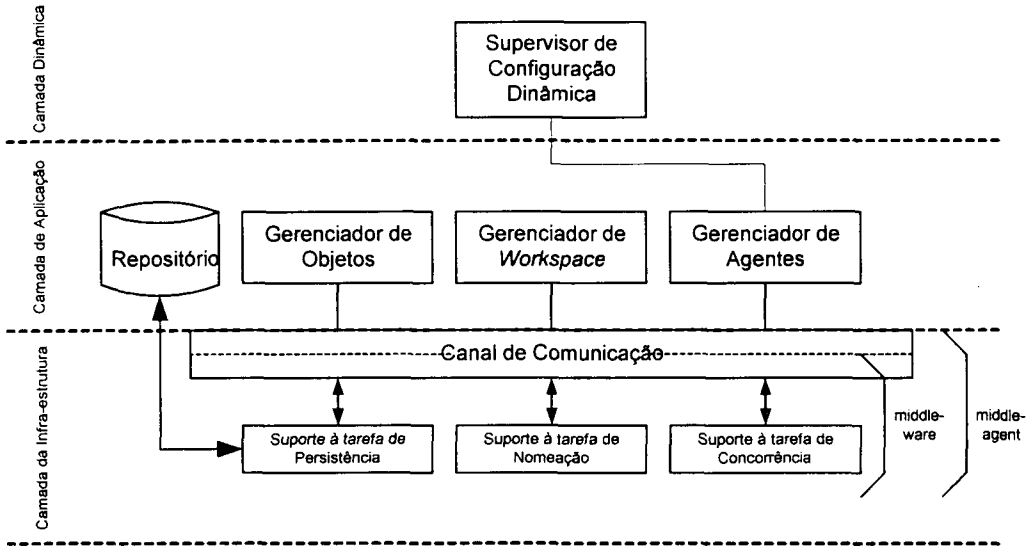

FIGURA 4.3 – ARQUITETURA DE UM AMBIENTE DE DESENVOLVIMENTO DE *SOFTWARE* DISTRIBUÍDO BASEADA EM AGENTES (PASCUTTI, 2002)

A arquitetura do *DiSEN* possui:

- uma camada dinâmica: que irá permitir, por exemplo, que sejam adicionados, excluídos, configurados ou modificados componentes de *software* e serviços de forma dinâmica, isto é, em tempo de execução;
- uma camada de aplicação, que irá suportar a MDSODI, gerenciamento de *workspace*, gerenciamento de agentes e irá conter o(s) banco(s) de dados necessário(s) para armazenamento dos dados sobre o ambiente, ou seja, todas as informações geradas durante o desenvolvimento de *software* e as informações da aplicação, assim como a base de conhecimento. É nesta camada que será tratada a ferramenta DIMANAGER pelo Gerenciador de Objetos com os demais elementos do ambiente;
- uma camada da infra-estrutura, que define o alicerce da arquitetura e proverá suporte às tarefas de persistência, nomeação e concorrência.

A DIMANAGER deve fornecer e receber informações das ferramentas relacionadas ao ambiente *DiSEN*.

Com base nas arquiteturas *DiSEN* (Pascutti, 2002) e no Suporte à Persistência de artefatos para o *DiSEN* (Moro, 2003), a Figura 4.4 mostra a representação da DIMANAGER neste ambiente, o qual está inserido dentro do *Workspace*. O *Workspace* na camada de aplicação é o ambiente de trabalho na estação do usuário que estará utilizando a ferramenta para o desenvolvimento do projeto.

O Gerenciador de Workspace é o responsável pela comunicação entre a DIMANAGER e o Repositório proposto por (Moro, 2003). As ferramentas relacionadas ao desenvolvimento do projeto devem fornecer as informações solicitadas para que a DIMANAGER gerencie o desenvolvimento das atividades.

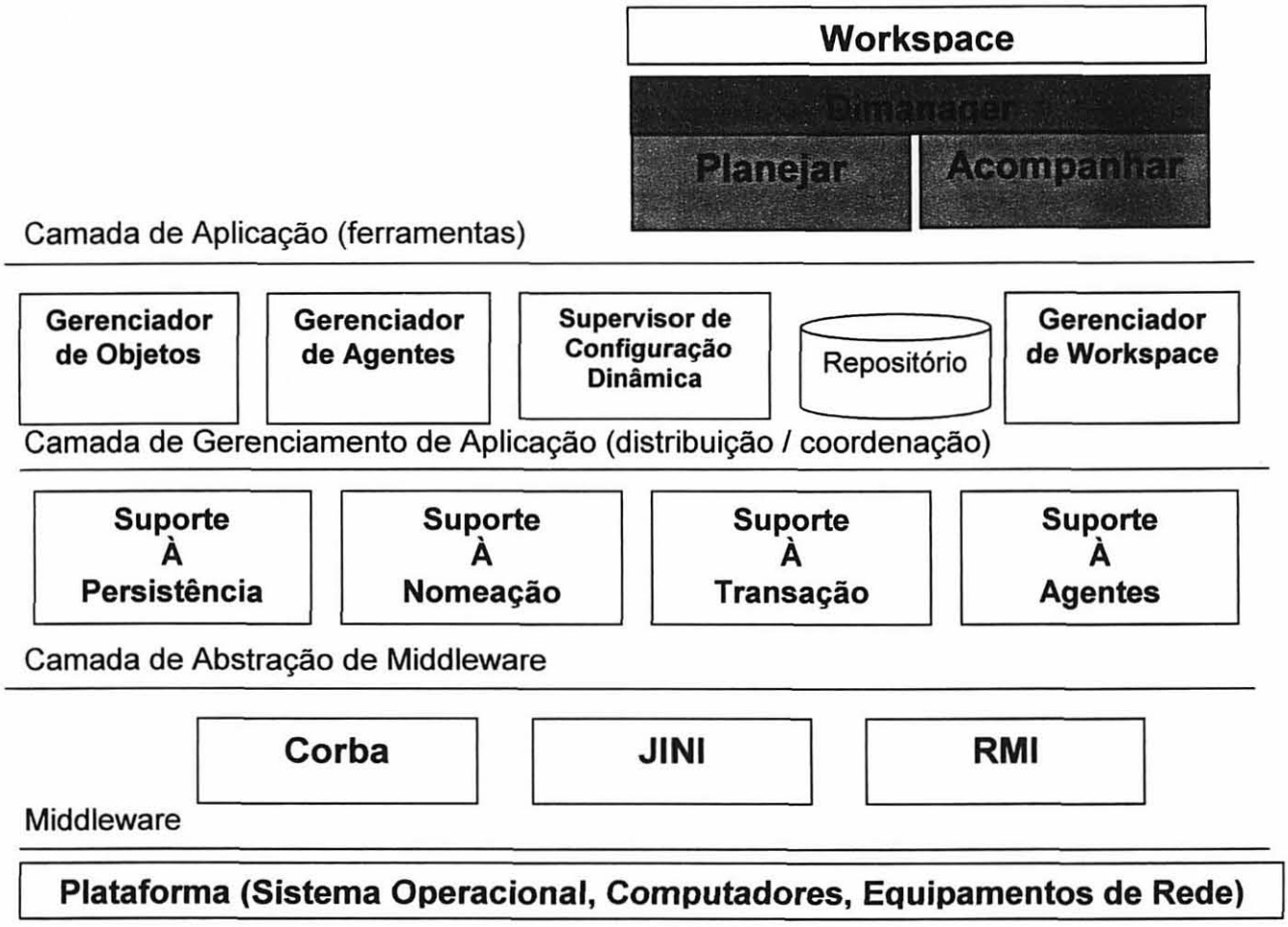

FIGURA 4.4 – REPRESENTAÇÃO DA DIMANAGER NA ARQUITETURA *DiSEN*

A representação dos modelos desenvolvidos na DIMANAGER está baseada na UML (*Unified Modeling Language*) (Furlan, 1998) por ser uma linguagem de modelagem de objetos unificada que aborda os conceitos fundamentais da orientação a objeto.

Neste contexto, os cenários têm sido utilizados para compreender melhor as exigências do usuário de *software*, tanto na orientação a objeto quanto nas abordagens estruturadas Furlan (1998). Os diagramas de *use CASE* descrevem a visão externa do sistema e suas interações com o mundo exterior, e representam uma visão de alto nível de funcionalidade intencional mediante o recebimento de um tipo de requisição do usuário. Desta forma, a visão do sistema deve ser comparada a uma caixa-preta fornecendo situações de aplicação. Seu funcionamento interno não é importante, mas deve ser utilizada para visualizar as necessidades de um novo sistema bem como para desenvolver novas versões.

Para completar o entendimento e o desenvolvimento da DIMANAGER foi definido, em conjunto com Batista (2003), o Modelo de Domínio da MDSODI, baseado no modelo de ciclo de vida e nas fases de cada incremento do ciclo de vida da MDSODI, como mostra a Figura 4.5.

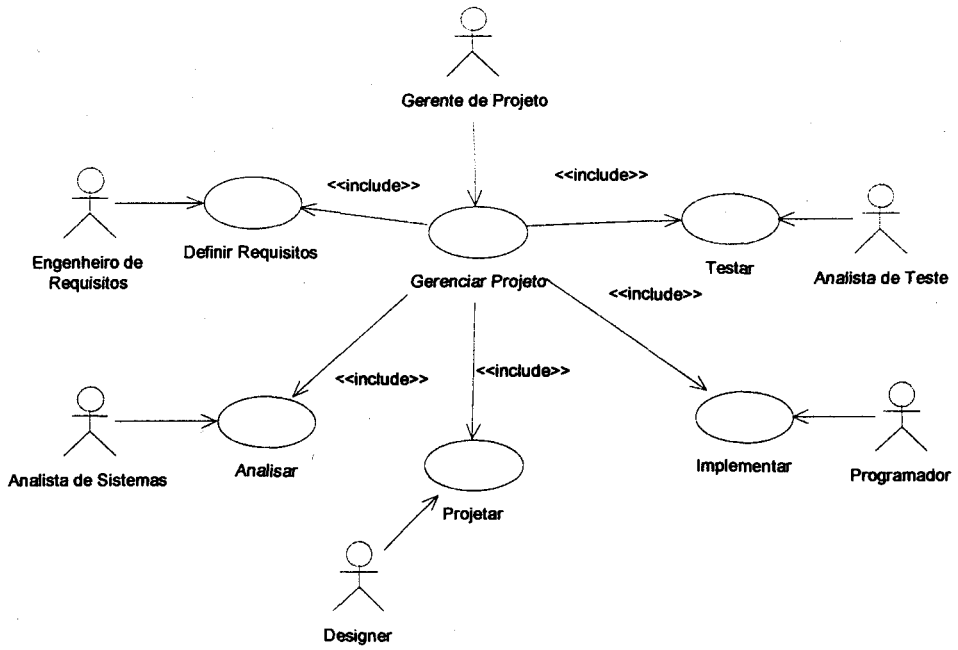

FIGURA 4.5 – MODELO DE DOMÍNIO DA MDSODI

4.3 Arquitetura da Ferramenta DIMANAGER

Os requisitos funcionais da DIMANAGER são: planejar e acompanhar o projeto. Com base nestes requisitos o primeiro diagrama de *use-case* da DIMANAGER, denominado modelo de domínio, está representado na Figura 4.6. O modelo de domínio representa uma primeira ordem de divisão do domínio do problema em seus comportamentos fundamentais, representando um conjunto de cenários de controle que é encapsulado dentro de um único objeto.

A seguir estão representados os modelos e diagramas elaborados para a especificação da DIMANAGER.

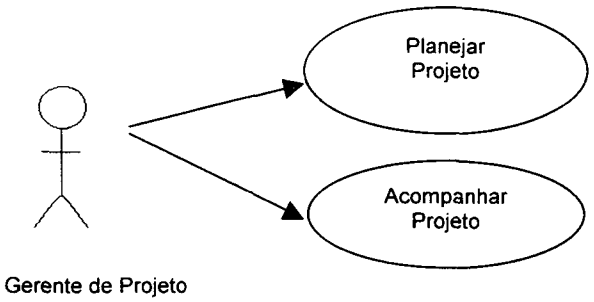

FIGURA 4.6 – MODELO DE DOMÍNIO DA DIMANAGER

Algumas funcionalidades, apresentadas no trabalho de Gomes et al. (2001), tais como: um calendário de atividades e o cadastro de usuários, capaz de apoiar e orientar o levantamento e a análise de dados incorporados ao ambiente de *software* instanciados pela Estação Taba contribuíram para identificar e elaborar o cadastro de usuários, participantes e cronograma à ferramenta DIMANAGER, pois fazem parte do gerenciamento das atividades desenvolvidas.

Além destas funcionalidades outras informações foram consideradas para a definição do planejamento do projeto, tais como:

- identificação das atividades dentro do projeto: identificar as atividades, obedecendo as fases estabelecidas no processo de desenvolvimento de *software* de acordo com a metodologia MDSODI que são: requisitos, análise, projeto, implementação e testes.
- identificação das métricas: estabelecer métricas que deverão ser analisadas para o

acompanhamento do projeto devem ser escolhidas com base na necessidade de cada projeto. Os estudos sobre métricas são complexos e dentre as técnicas de estimativas disponibilizadas para o desenvolvimento de *software* podem ser citadas, segundo Pressman (2002): estimativas de linhas de código (LOC), estimativas de pontos-por-função (FP) ou estimativas do esforço. Neste trabalho adotaremos a estimativa do esforço definida através do número de homens por hora de atividade (humano-hora);

- definição dos participantes: a identificação dos participantes com atribuição de funções e habilidades é de extrema importância, pois dela depende o bom andamento do projeto. Cada participante, embora tenha uma determinada função pode participar do projeto em várias atividades de acordo com seus conhecimentos e habilidades sendo também um elo de ligação entre os setores envolvidos no projeto. Todo participante deve ser cadastrado no sistema como usuário, desta forma ele terá acesso ao projeto como participante do mesmo.
- elaboração do cronograma de atividades: o cronograma contém o planejamento das atividades no projeto e deve: estabelecer a atividade, o esforço (que informa o número de homens por hora de atividade = homem/hora), data inicial e data final prevista para cada atividade, e o estado em que a atividade se encontra que pode ser: em andamento, parado, suspenso, em planejamento ou encerrado. O estado da atividade será atualizado pelo participante, coordenador de área ou gerente do projeto sempre ocorrer algum imprevisto (parado, suspenso e em planejamento), caso contrário, a atualização do estado será automática (em andamento ou encerrado). Os membros envolvidos na elaboração do projeto devem ser selecionados de acordo com o conhecimento de cada um. Para a distribuição de atividades em cada fase do projeto, a função, o conhecimento e a habilidade de cada participante deve ser levado em consideração para que o mesmo desempenhe a atividade estabelecida de forma eficaz e eficiente.

Cada participante deve documentar as atividades executadas durante o decorrer dos trabalhos gerando o acompanhamento do projeto. O registro da execução da atividade deve conter a identificação do participante, a fase do projeto, a atividade em

desenvolvimento, a data da execução, a hora inicial e final da mesma assim como o estado em que se encontra a atividade.

A seguir estão representados os diagramas de *use-case* das funcionalidades apresentadas no Modelo de Domínio, que são: Planejar Projeto e Acompanhar Projeto, através das Figuras 4.7 e 4.8.

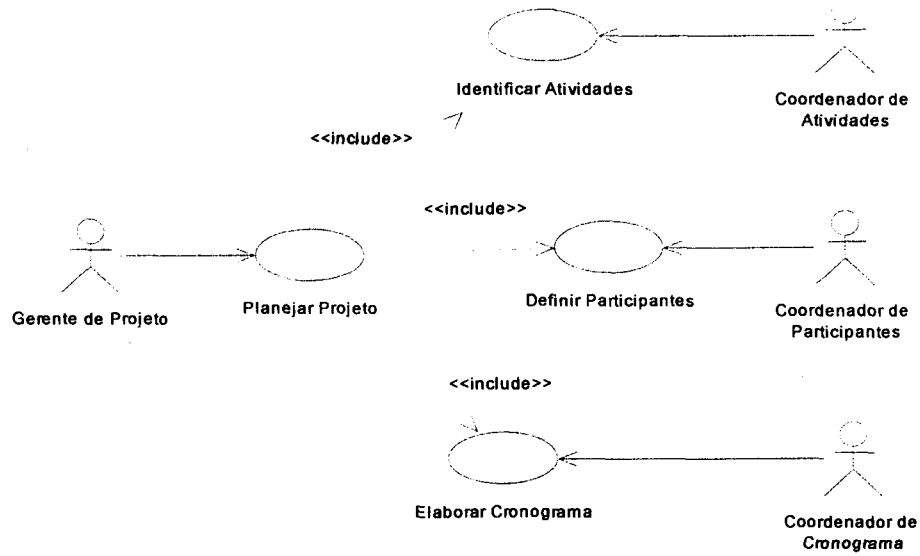

FIGURA 4.7 – DIAGRAMA DE *USE-CASE* – PLANEJAR PROJETO DA DIMANAGER

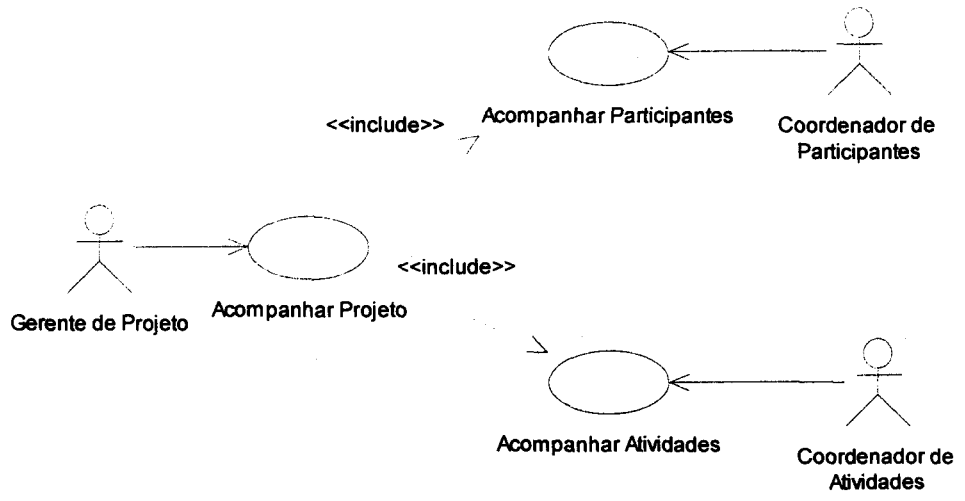

FIGURA 4.8 – DIAGRAMA DE *USE-CASE* – ACOMPANHAR PROJETO DA DIMANAGER

Analisando as funcionalidades de cada use-case, as classes identificadas nos modelos anteriores foram representadas no Diagrama de Classes da DIMANAGER

apresentado na Figura 4.9, onde o relacionamento entre as classes também são estabelecidos.

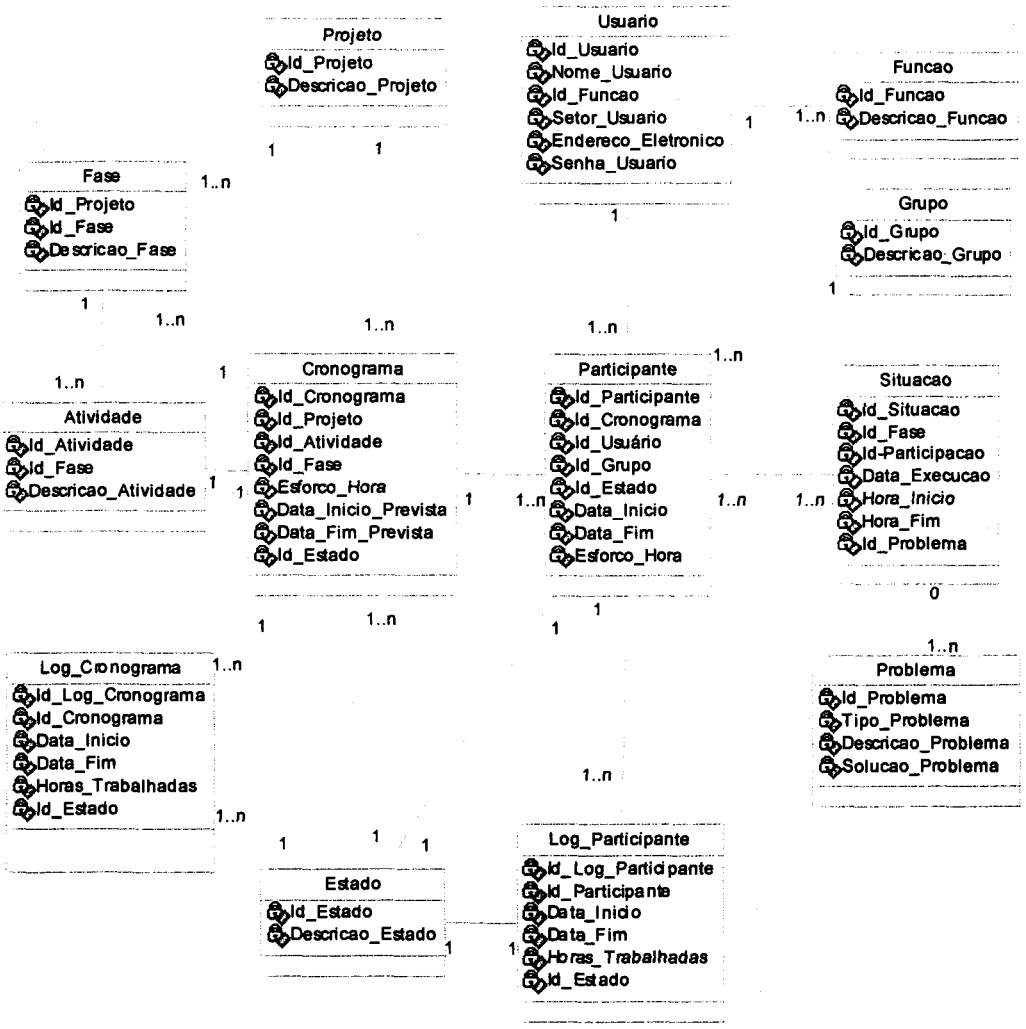

FIGURA 4.9 – DIAGRAMA DE CLASSES DA DIMANAGER

A ferramenta DIMANAGER gerou dois pacotes: Planejar Projeto e Acompanhar Projeto. Estes pacotes compõem a camada específica da aplicação, e a camada genérica deve ser composta por sub-sistemas que possam dar origem a um componente de *software* reusável Furlan (1998).

No Diagrama de Classes (Figura 4.9) foram identificadas as classes de cada pacote, sendo elas:

- no pacote Planejar Projeto: Projeto, Fase, Atividade, Usuário, Funcao, Problema, Situacao, Cronograma, Participante e Estado.
- no pacote Acompanhar Projeto: Log_Participante e Log_Cronograma.

A Figura 4.10 apresenta o Diagrama de Sub-sistemas de acordo com a divisão em camadas apresentadas em Gravena (2000) na fase de projeto e identifica os subsistemas que compõem a camada específica da aplicação.

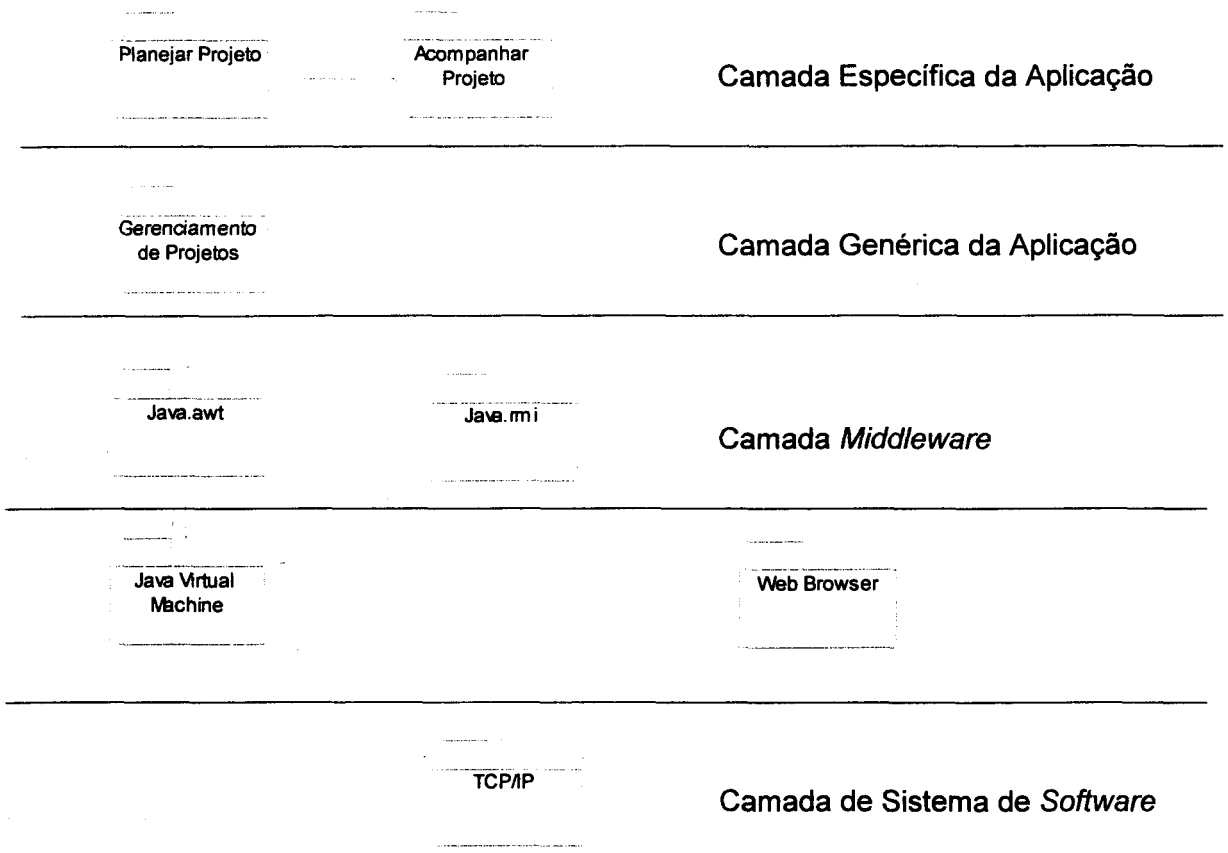

FIGURA 4.10 – DIAGRAMA DE SUB-SISTEMAS DA DIMANAGER

Segundo Gravena (2000) o Diagrama de Sub-sistemas deve ser desenvolvido baseado na fase de projeto moldando-se os requisitos não funcionais e outras considerações não identificadas nas fases anteriores, como por exemplo: linguagem de programação, interface com o usuário, reuso de componentes e outras.

Os Diagramas de Colaboração foram elaborados com base nas classes identificadas no Diagrama de Classes e nos *use-cases*, sendo eles: Identificar Atividade, Definir Participantes, Elaborar Cronograma e Acompanhar Projeto. A fase Acompanhar Projeto é totalmente dependente do planejamento do projeto e foram definidos dois diagramas de colaboração: Acompanhar Participantes e Acompanhar Atividades, pois estas fazem parte do Acompanhamento do Projeto.

No Diagrama de Colaboração - Identificar Atividade (Figura 4.11), o Gerente do Projeto em conjunto com o Coordenador de Atividades devem identificar as atividades envolvidas em cada fase do Plano de Projeto. De acordo com Gomes et al. (2001) as métricas estabelecidas para cada atividade devem ter o objetivo de prover as informações necessárias para apoiar decisões gerenciais (distribuição de tarefas, controle do cronograma, distribuição de recursos, etc) durante o ciclo de vida do *software* em desenvolvimento.

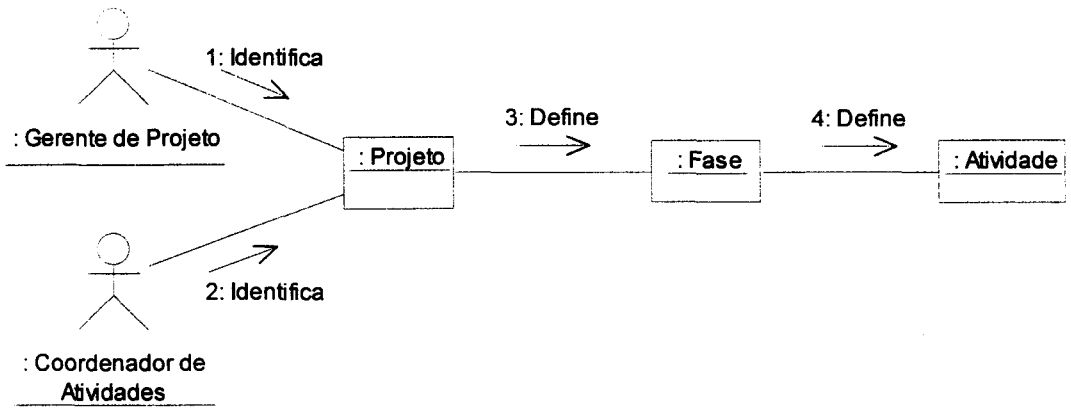

FIGURA 4.11 – DIAGRAMA DE COLABORAÇÃO – IDENTIFICAR ATIVIDADE

Na definição dos participantes o Gerente de Projeto solicita o envolvimento do Coordenador de Participantes (Figura 4.12) para escolher aqueles cuja função e habilidades são necessárias para o bom desempenho de cada atividade.

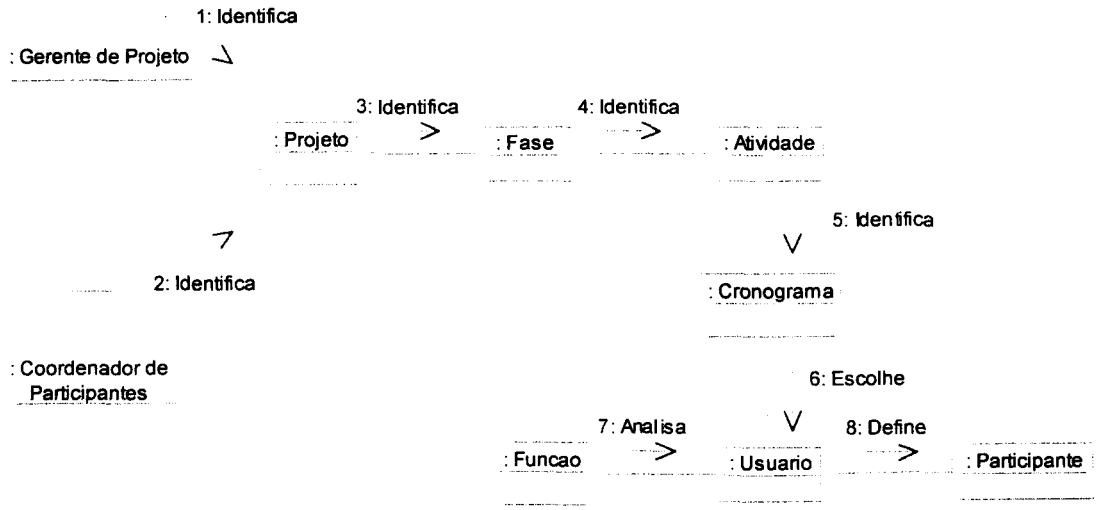

FIGURA 4.12 – DIAGRAMA DE COLABORAÇÃO – DEFINIR PARTICIPANTES

De acordo com Pressman (2002) a programação do projeto de *software* não é, de fato, diferente de qualquer projeto de engenharia. Um conjunto de tarefas de projeto é identificado. Interdependências entre as tarefas são estabelecidas e o esforço associado a cada tarefa é estimado. Pessoas assim como outros recursos são atribuídos e uma "rede de tarefas" é criada.

Desta forma, para que a atividade estabelecida no cronograma seja bem definida o Gerente de Projeto solicita a participação do Coordenador de Cronograma, para que juntos possam estabelecer o cronograma de atividades, conforme mostrado na Figura 4.13.

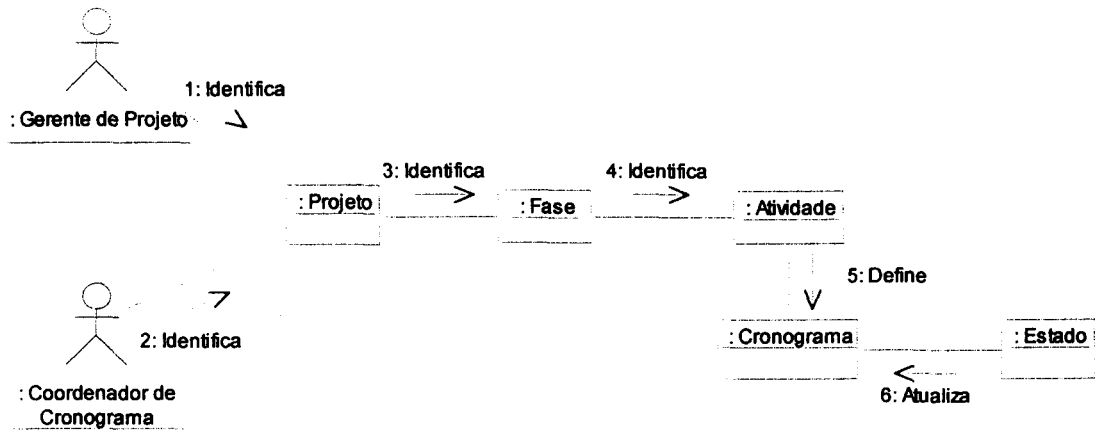

FIGURA 4.13 – DIAGRAMA DE COLABORAÇÃO – ELABORAR CRONOGRAMA

O Acompanhamento do Projeto é a parte mais importante da ferramenta DIMANAGER. O Gerente de Projeto pode solicitar a participação dos Coordenadores de Participantes e de Atividades para analisar os resultados obtidos. Baseado nos participantes, atividades e métricas já definidas (humano-hora), o Gerente de Projeto visualiza resultados referentes às atividades desenvolvidas por cada participante, analisa o desempenho de cada um, e verifica se o projeto está evoluindo dentro do prazo estabelecido.

Tendo como base as informações do Acompanhamento o projeto recupera todas as informações referentes ao Projeto tais como: Projeto, Fase, Atividade, Cronograma, Participante, Função, Participação, Situação e Lista de Problemas. Através destas informações são apresentadas *views* para serem analisadas. Os Diagramas de Colaboração para Acompanhar Projeto – Participantes e Atividades são demonstrados nas Figuras 4.14 e 4.15.

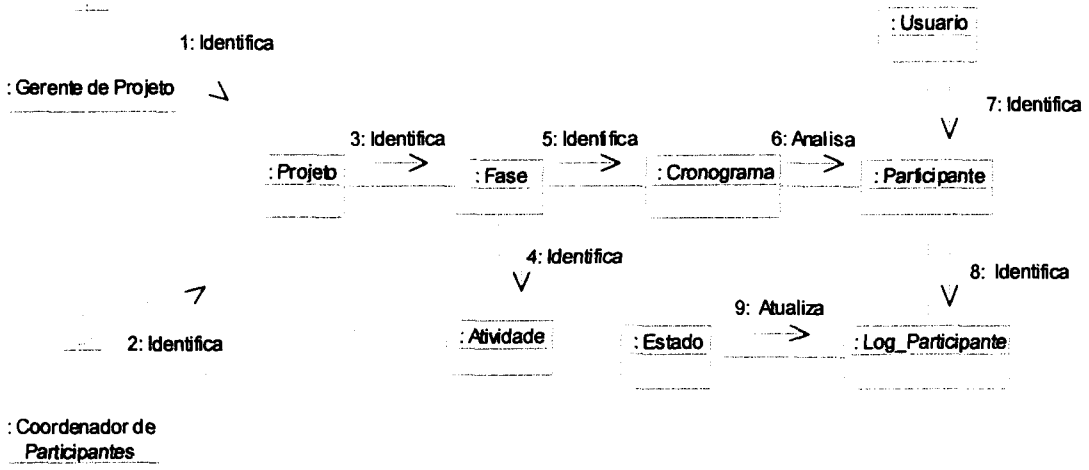

FIGURA 4.14 – DIAGRAMA DE COLABORAÇÃO – ACOMPANHAR PROJETO – PARTICIPANTES

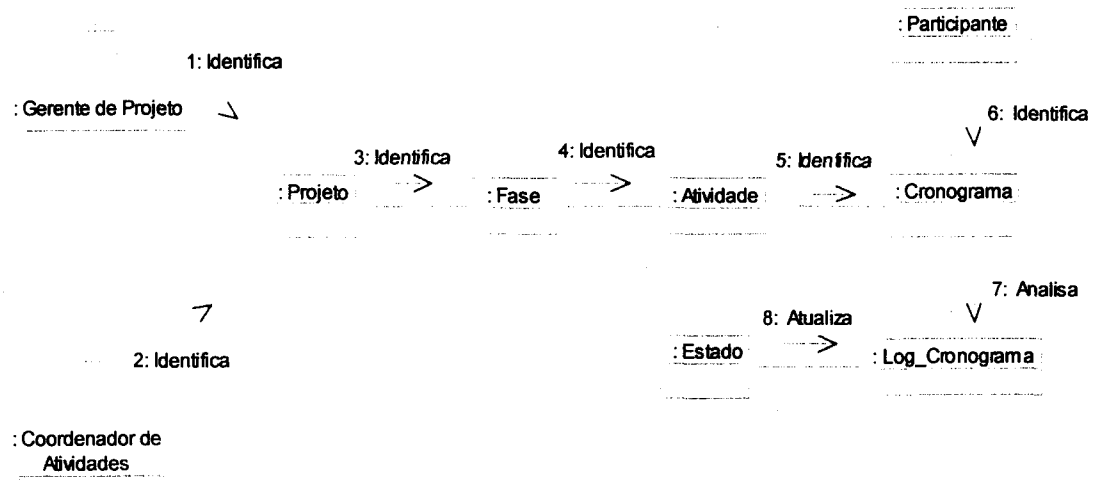

FIGURA 4.15 – DIAGRAMA DE COLABORAÇÃO – ACOMPANHAR PROJETO – ATIVIDADES

4.4 Apresentação da Ferramenta DIMANAGER

É importante salientar que os projetos são o meio principal pelo qual a organização lida com as mudanças e que cada projeto tem objetivos específicos relacionados aos custos, à programação e à capacidade de desempenho técnico.

O gerenciamento de um projeto é o processo de tomar decisões que envolvem o uso de recursos, para realizar atividades temporárias, com o objetivo de fornecer um resultado (Maximiano, 2002). A essência do gerencimento de um projeto é o planejamento e a execução das atividades de seu ciclo de vida, para que o produto seja fornecido ao final.

A gerência de projetos é executada mediante um processo de administração que engloba as fases de Cleland e Ireland (2002):

- Planejamento: desenvolver os objetivos, metas e estratégias que proporcionem o compromisso de recursos para apoiar o projeto;
- Organização: identificar os recursos humanos e materiais necessários, fornecer uma distribuição adequada dos mesmos e estabelecer os papéis individuais e coletivos dos membros das equipes de projetos;

- Motivação: estabelecer um sistema cultural que faça vir à tona o melhor que as pessoas podem fazer em seu projeto de trabalho;
- Direção: proporcionar a competência necessária de liderança para garantir a tomada e a execução de decisões que envolvem o projeto;
- Controle: monitorar, avaliar e controlar o emprego dos recursos no projeto que sejam coerentes com ele e com os planos organizacionais.

Alguns fatores influenciam o desenvolvimento e desempenho dos projetos, são eles: mudanças de produtos, serviços e pessoas; comunidades de *stakeholders*[1]; condições da empresa; ambiente cultural; e, contexto estratégico e operacional. O gerente do projeto deve se preocupar com estes fatores, estando sempre atualizado em relação às mudanças que estão ocorrendo quanto aos produtos, clientes, colaboradores e planejamento estratégico e operacional da empresa, durante o planejamento do projeto assim como durante o acompanhamento do mesmo para que o resultado obtido esteja o mais próximo possível do idealizado.

Durante o planejamento e a organização do projeto o gerente deve ser cuidadoso com as informações que serão colocadas no projeto.

A primeira janela disponível na DIMANAGER, Figura 4.16, disponibiliza ao gerente três áreas de atuação: Conheça a DIMANAGER; Serviços; e Usuários.

[1] *Stakeholders:* todos os agentes que contribuem para o desempenho da organização, como por exemplo, funcionários de todos os níveis, acionistas, clientes, comunidade, governos federal, estadual e municipal. Cleland e Ireland (2002)

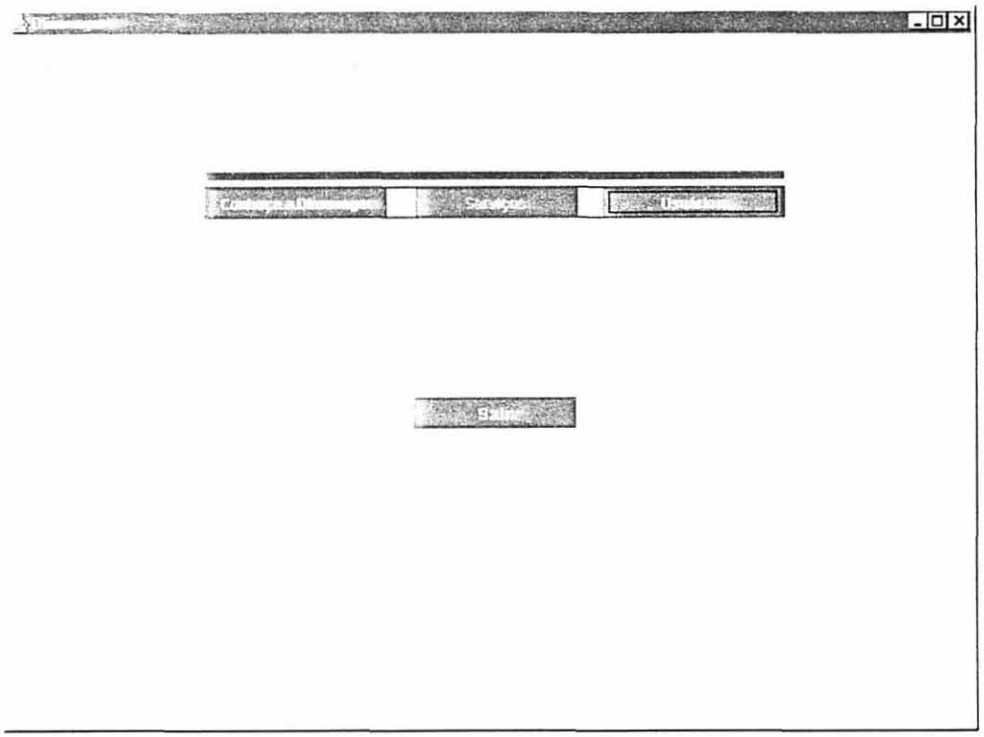

FIGURA 4.16 – JANELA DE ABERTURA DA DIMANAGER

A primeira opção "Conheça a DIMANAGER", é uma janela contendo uma descrição da ferramenta. Apresenta os conceitos, sua estrutura e finalidade e permite ao usuário ter uma noção do que a DIMANAGER pode oferecer.

Na opção "Serviços", o gerente deve cadastrar as informações que comporão o projeto, como por exemplo, as atividades estabelecidas para cada fase de desenvolvimento, de acordo com a metodologia MDSODI, os grupos de trabalho e suas respectivas atividades, a identificação de possíveis problemas e suas soluções, o cronograma de atividades e as metas a serem cumpridas em cada atividade.

A terceira opção, "Usuários", disponibiliza ao gerente do projeto o cadastro dos usuários que participarão no desenvolvimento do projeto.

Qualquer usuário poderá conhecer a DIMANAGER através da primeira opção. Para mais detalhes verificar Anexo I. A janela apresentada (Anexo I), descreve a DIMANAGER para que o usuário possa entender o funcionamento da ferramenta.

Na descrição da DIMANAGER encontra-se um resumo de suas características e de sua essência. Uma descrição completa do conteúdo desta janela encontra-se no Anexo II.

Ao iniciar um novo projeto, o gerente de projeto deve primeiramente, definir o projeto, escolhendo a opção "Serviços" e em seguida, "Planeje seu Projeto", conforme Figura 4.17.

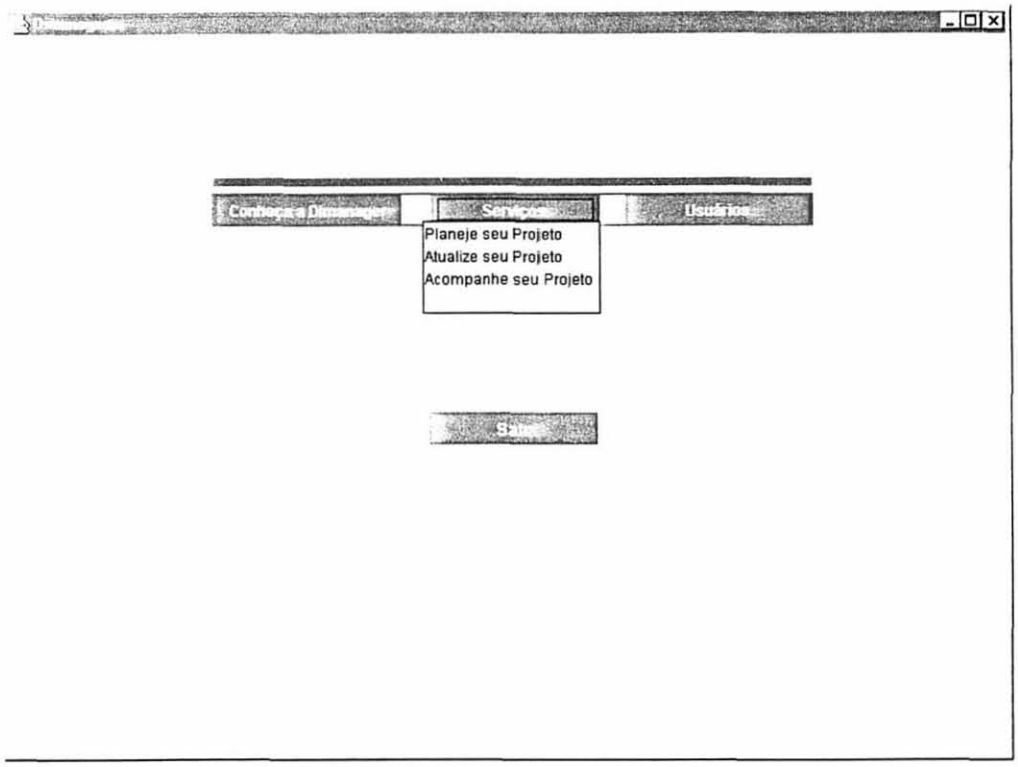

FIGURA 4.17 – PLANEJE SEU PROJETO

O gerente de projeto deve especificar detalhadamente as informações do projeto a ser cadastrado, com uma linguagem clara e de fácil compreensão, facilitando o conhecimento do mesmo (Figura 4.18). Neste momento, deve ser informado o responsável pelo projeto em andamento. O responsável pelo projeto é o primeiro usuário a ser cadastrado, e, já deve estar registrado no sistema. (Anexo I)

Na fase do planejamento de um novo projeto, informações como: estado, lista de problemas, grupo, atividades, cronograma, participação e situação devem ser informadas.

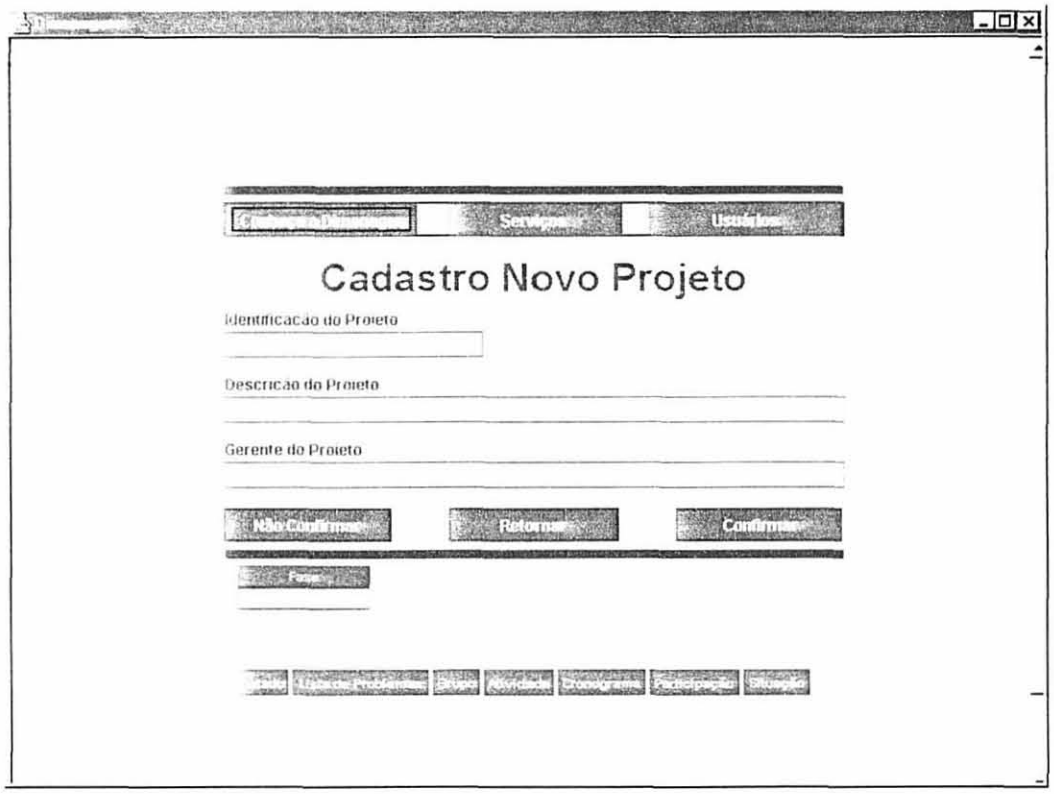

FIGURA 4.18 – CADASTRO NOVO PROJETO

O participante selecionado para desenvolver uma atividade no projeto em planejamento, deve ser cadastrado na DIMANAGER como um usuário.

Os usuários da ferramenta serão cadastrados logo após a definição dos grupos de trabalho, sendo atribuídas as permissões de acesso de acordo com as atividades de cada um, mas os coordenadores do projeto podem ser cadastrados no primeiro momento, devendo planejar e acompanhar o projeto em conjunto com o gerente .

O gerente do projeto deve cadastrar os usuários com as seguintes informações: nome completo, número de identificação, o setor em que o usuário trabalha, sua função, o endereço de correio para correspondência eletrônica, conforme Figura 4.19. A senha e a confirmação de senha devem ser preenchidas em conjunto com o usuário. As senhas devem ser mantidas em rigoroso sigilo e o usuário é o único responsável por elas.

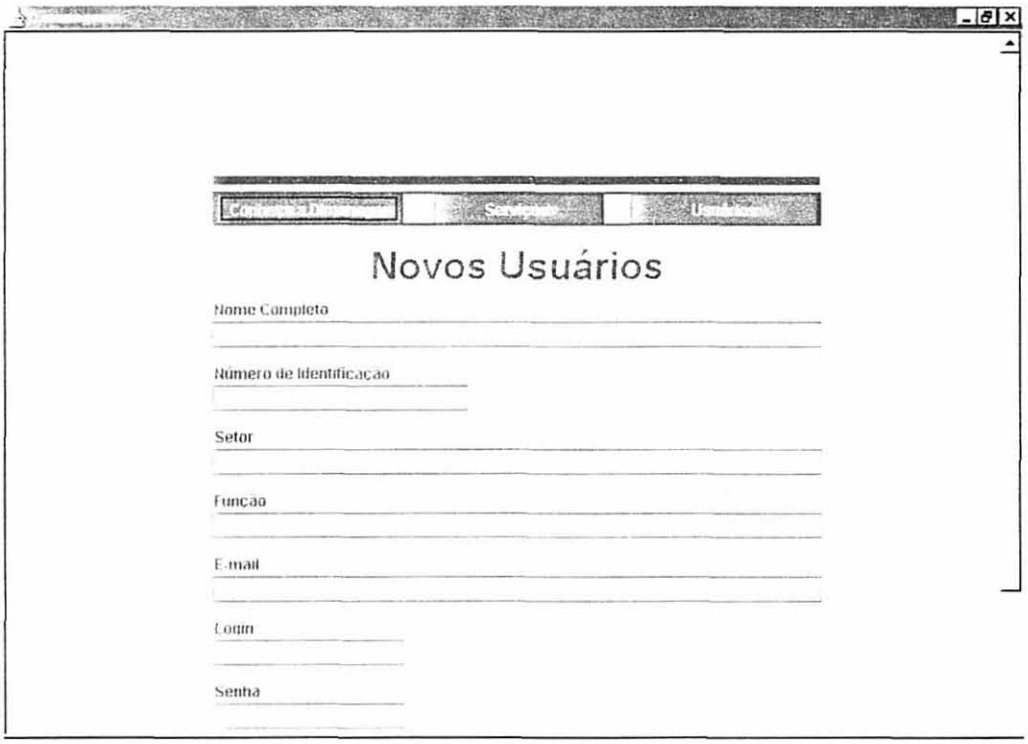

FIGURA 4.19 – USUÁRIOS

Cada usuário tem um nível de acesso, podendo ser classificado como: gerente ou participante. O gerente ou coordenador de área tem um nível mais amplo de permissões de acesso, e o usuário classificado como participante poderá trabalhar somente na atividade definida para ele.

Caso o usuário não se lembre de sua senha, tanto o gerente do projeto quanto os coordenadores de áreas podem alterar a senha do mesmo, não havendo necessidade de saber a senha anterior.

O usuário pode modificar sua senha em qualquer tempo, assim que tiver necessidade de alterá-la, mas para executar esta operação, o usuário deve saber a senha que deverá ser modificada.

A janela de alteração de senha (Anexo I) permite a visualização ou não da senha a ser alterada. O programa solicitará a digitação da nova senha e uma confirmação da mesma que será comparada com a anterior, para depois atualizá-la. Caso a senha não seja a mesma o programa deverá informar ao usuário para executar o procedimento corretamente.

O passo seguinte deve ser a definição e o cadastramento das atividades. As atividades devem ser especificadas de acordo com cada fase do projeto. A DIMANAGER fixou estas fases com base na metodologia MDSODI: definir requisitos, analisar, projetar, implementar e testar. A Figura 4.20 ilustra o cadastramento da atividade.

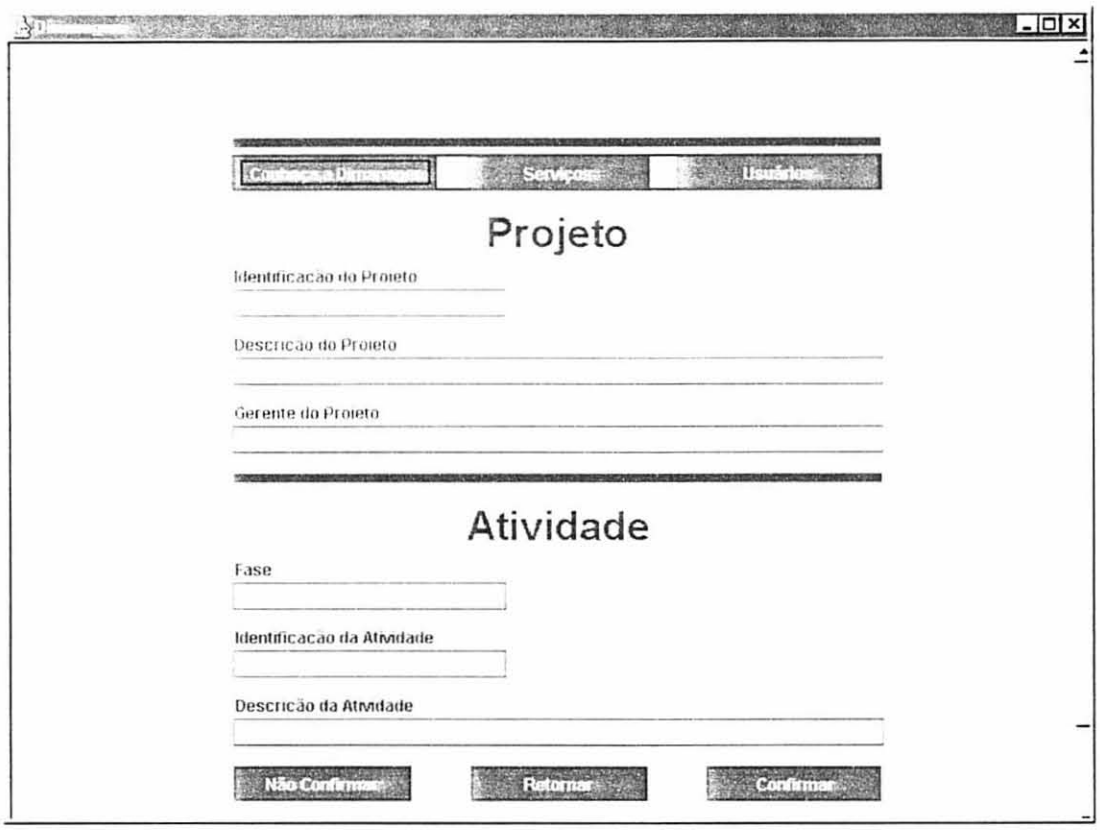

FIGURA 4.20 - ATIVIDADE

O gerente de projeto em conjunto com o coordenador de atividades deve estabelecer as atividades após escolher a fase a ser detalhada. As atividades de cada fase devem ser descritas de forma clara e precisa para que os participantes do projeto possam entender e desenvolver o projeto dentro do objetivo especificado. Um código de identificação da atividade assim como sua descrição são necessárias para poder relacionar cada atividade com o cronograma e os participantes.

De acordo com Maximiano (2002), a preparação de uma lista de atividades pode ser considerada como sendo a primeira parte do planejamento. A definição das atividades necessárias para realizar o projeto começa com o planejamento do produto,

que é o resultado do projeto. O desenho da estrutura analítica, também chamada estrutura do projeto, é uma técnica que consiste em dividir o produto principal em partes administráveis, que retrata a divisão do produto em partes. Há diversos critérios e formatos para representar uma estrutura de projeto: organogramas, listas, chaves sinópticas, esquemas e outros, conforme Figura 4.21.

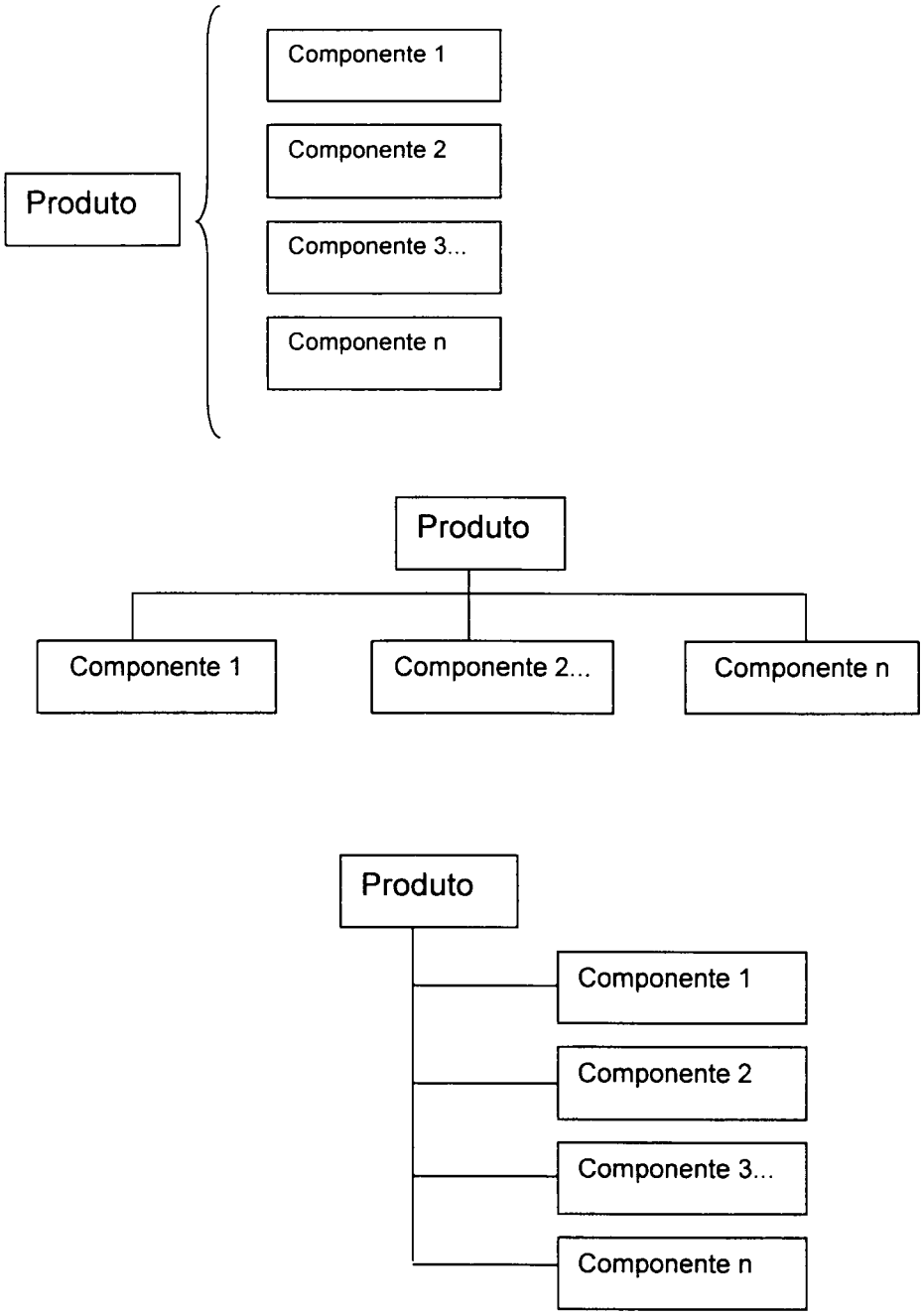

FIGURA 4.21 – EXEMPLO DE ESTRUTURAS DE PROJETOS

Depois de preparar a lista, deve-se identificar a seqüência em que as atividades serão realizadas, permitindo desta forma, estabelecer as prioridades e o encadeamento das mesmas.

A etapa seguinte é a definição dos estados em que cada atividade deve ser classificada. O cadastramento dos estados deve ser informado através da janela que exige a identificação do estado e a descrição do mesmo. O gerente de projeto deve escolher os estados, os quais podem ser: em andamento, parado, suspenso, em planejamento e encerrado.

O gerente de projeto deve ser capaz de identificar problemas que podem ocorrer no desenvolvimento do projeto. Todo projeto envolve riscos e a gerência deve identificá-los durante a iniciação do projeto e, em seguida, no seu planejamento (Cleland e Ireland, 2002). Os problemas, ou riscos, podem ser classificados em duas áreas: problemas técnicos e problemas organizacionais.

Tait (2000) informa que, muitas vezes, os problemas considerados técnicos, fogem ao controle das organizações e representam um desafio a ser encarado com bom senso, acompanhando a evolução crescente da TI (Tecnologia de Informação), sem no entanto desprezar sua experiência e potencialidade.

Neste aspecto, deve-se questionar a tecnologia existente e, a seguir, registrar quaisquer ocorrências que possam falhar e a probabilidade de fracasso. Estes problemas devem ser registrados, para que o gerente possa resolver o problema de maneira que o usuário não perceba a falha do sistema. Estes problemas podem ser cadastrados no início do projeto, já que algumas falhas de comunicação podem ocorrer e são conhecidas. Os problemas técnicos podem ser classificados também como problemas internos e dizem respeito à entrega dos resultados desejados (Cleland e Ireland, 2002).

Alguns problemas técnicos já foram identificados e estão relacionados no Anexo III.

Quanto aos problemas organizacionais, podem ser classificados como problemas internos ou ainda externos. Englobam recursos humanos, negócios e metas,

culminando em uma postura administrativa que considere todos os elementos que, ignorados, podem levar ao insucesso (Tait, 2000).

Normalmente, não se tem muito controle sobre os problemas externos, que podem receber a influência de acordos e contratos com terceiros. Os problemas externos podem ser resolvidos de acordo com a influência do gerente de projeto junto a outros gerentes de projetos, gerentes funcionais, distribuidores e entidades contratantes. Quanto aos problemas internos é necessário identificar as falhas no cronograma na medida em que se executa o planejamento Cleland e Ireland (2002). relacionam uma lista para a identificação de problemas que podem ser previstos e cadastrados para se evitar discussões e perda de tempo se os mesmos ocorrerem, conforme segue:

- *Checklist* para áreas de risco do projeto;
- Lições aprendidas com projetos anteriores;
- Listas de recursos disponíveis;
- Registros de treinamento de recursos para habilidades aplicáveis;
- Revisão dos planos do projeto por especialistas;
- Revisões dos planos pela alta administração;
- Auditoria de capacidade para gerência de projetos organizacionais.

É claro que nem sempre os mesmos problemas ocorrem em projetos distintos, pois algumas medidas podem ser tomadas para que se evitem transtornos. E também nem sempre se consegue prever todos os problemas que podem acontecer, por isso é interessante manter o Repositório de Dados (Moro, 2003) sempre atualizado, com os problemas e suas soluções, para que possa ser consultado sempre que necessário.

Se os problemas são cadastrados, fica fácil resolvê-los quando ocorrerem novamente e deve-se considerar que estas informações não serão perdidas em futuros projetos.

O cronograma do projeto só poderá ser definido após a especificação de todas as etapas anteriores. O cronograma de atividades consiste em decidir quando as atividades acontecem e quais são os participantes que devem executar a atividade descrita. A Figura 4.22 mostra como deve ser cadastrado o cronograma de atividades.

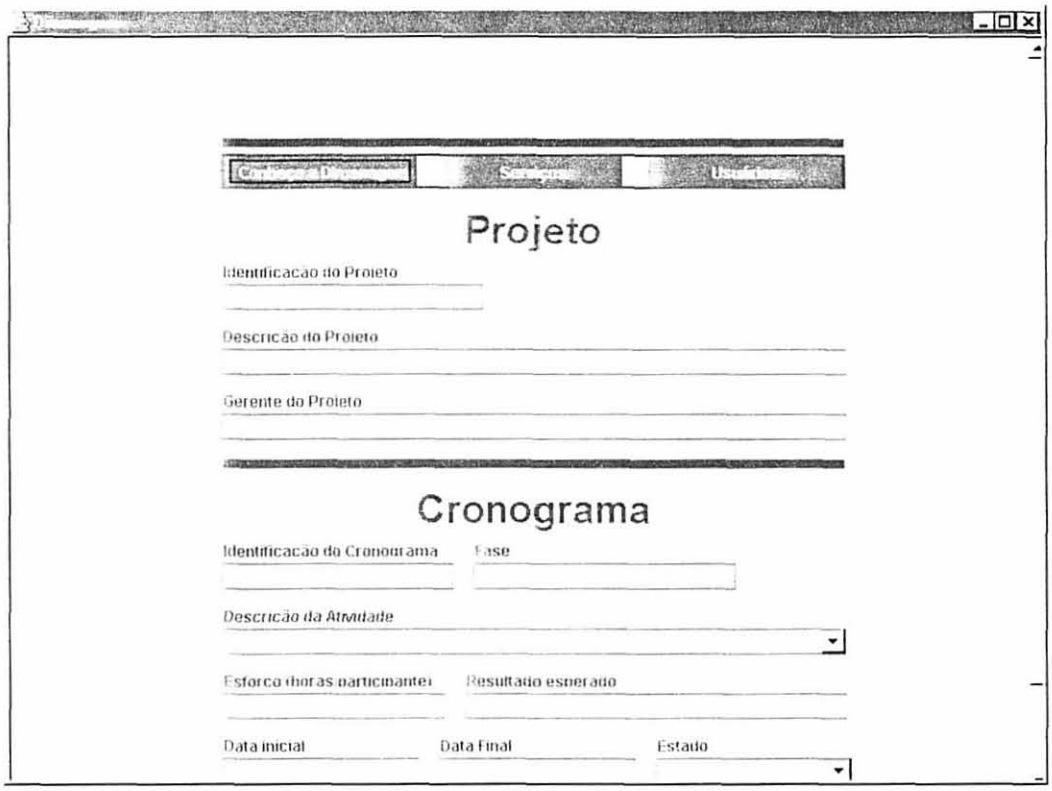

FIGURA 4.22 - CRONOGRAMA

Maximiano (2002) coloca que o retrato do cronograma se baseia em decisões do planejamento e a preparação do mesmo na duração das atividades. A duração das atividades é determinada pela quantidade de recursos previstos assim como a participação de serviços e produtos fornecidos por terceiros. Em teoria, quanto maior a quantidade de pessoas e recursos alocados ao projeto, maior velocidade no ciclo de vida do mesmo.

Na descrição do cronograma, cada atividade identificada será programada de acordo com o esforço necessário para sua execução (horas/participante); o resultado que se espera alcançar, a data inicial e final e o estado em que se encontra a atividade.

Por se tratar de uma ferramenta de gerenciamento de desenvolvimento de *software* distribuído, optou-se pela definição de grupos de trabalho, visando maior rapidez no desenvolvimento do *software*. Os grupos de trabalho podem ser definidos baseando-se na atividade desenvolvida ou na localização geográfica dos participantes em que ocorre o desenvolvimento.

O acompanhamento através dos grupos de trabalho deve ser analisado pelo gerente de projeto de maneira a tornar mais prático e ágil o desenvolvimento do projeto.

O usuário tornar-se-á um participante quando o gerente de projeto ou coordenador de participantes cadastrá-lo no projeto, o que deve ocorrer após a definição do cronograma e dos grupos de trabalho. O ideal é que as duas etapas – cronograma e participantes – sejam estabelecidas em conjunto, de maneira a determinar a duração das atividades de acordo com os recursos humanos alocados para cada uma, conforme Figura 4.23.

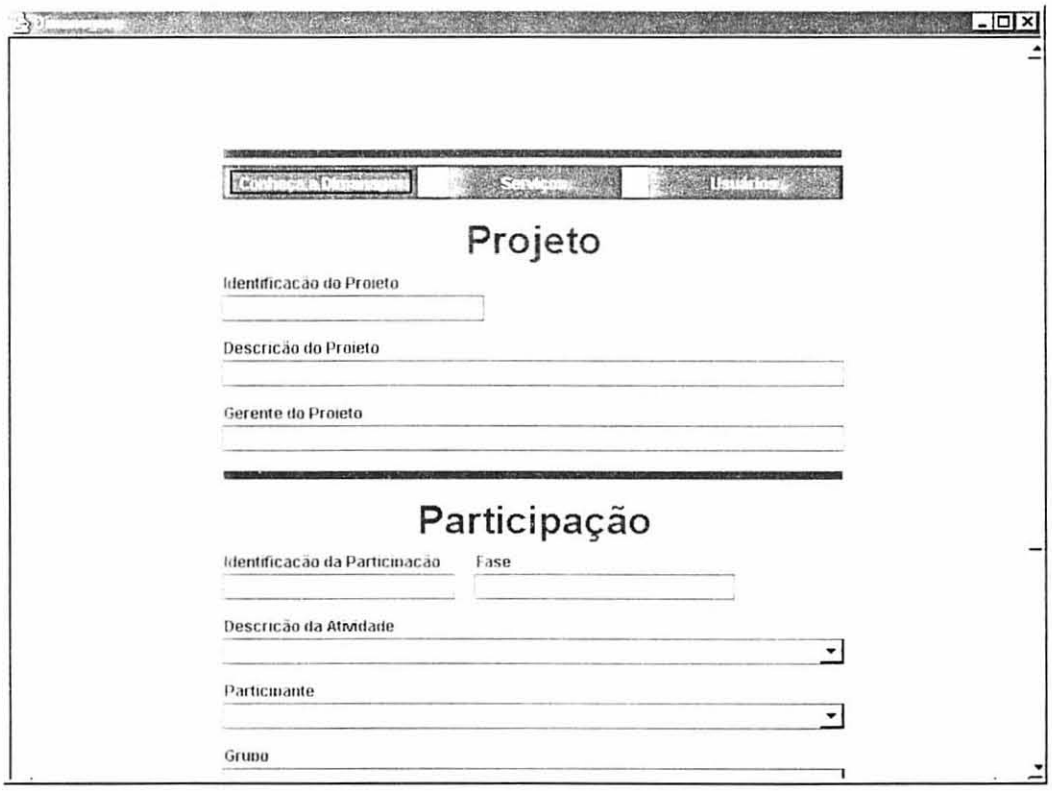

FIGURA 4.23 - PARTICIPANTES

A escolha dos participantes é importante e requer um certo cuidado pois o desenvolvimento das atividades terá um melhor ou pior desempenho de acordo com o conhecimento e habilidade de cada um.

Os participantes devem pertencer a um grupo de trabalho que será definido pela atividade ou pela localização geográfica dos mesmos.

O cadastro dos participantes requer a definição da fase; da atividade; do participante (que já deve ter sido cadastrado como um usuário); o grupo a que pertence; a data inicial e final, o esforço necessário para o desempenho daquela atividade (horas/participante); e o estado que se encontra o desenvolvimento da atividade do participante envolvido.

A situação de cada atividade deve ser mantida atualizada para um melhor acompanhamento delas. A situação deve ser cadastrada, conforme Figura 4.24.

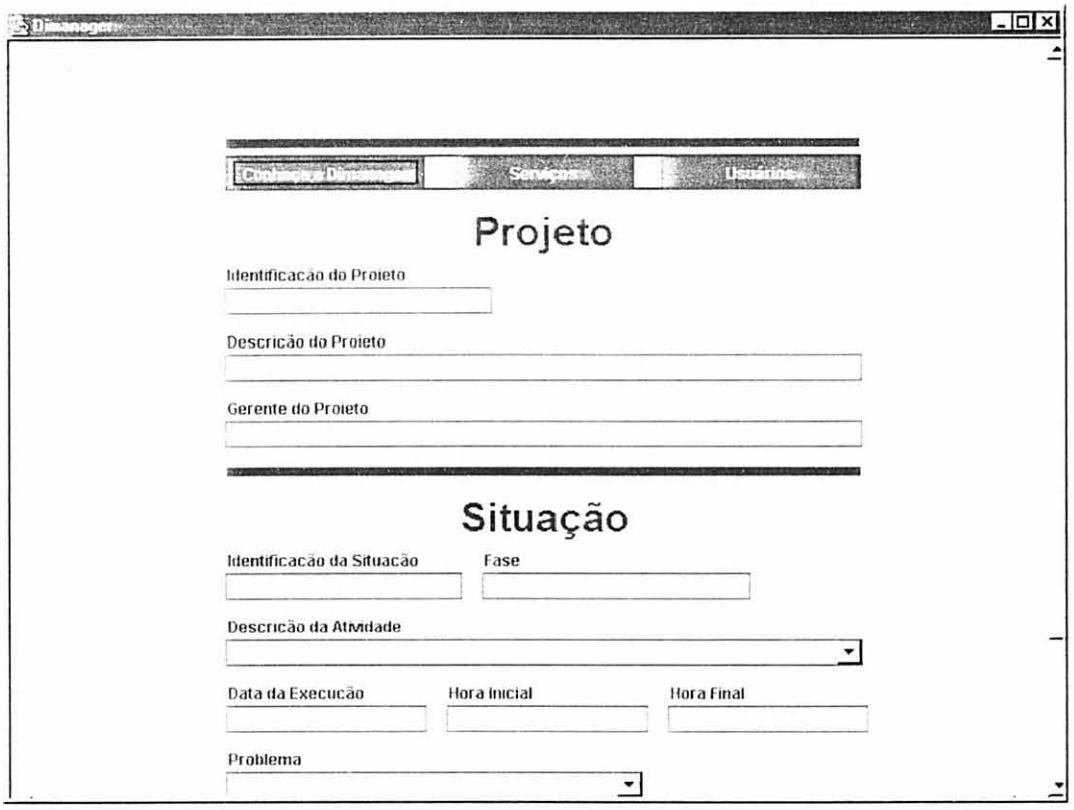

FIGURA 4.24 – SITUAÇÃO

O acompanhamento da situação é importante para que o gerente de projeto verifique se está ocorrendo algum problema durante o processo de desenvolvimento. É importante também verificar quais problemas (queda de conexão, afastamento de participantes, recursos financeiros escassos, entre outros) estão ocorrendo e as soluções encontradas para a resolução dos mesmos. Os coordenadores do projeto são elementos de fundamental importância, pois devem ajudar o gerente a detectar as falhas corrigindo-as ou levando para o gerente resolver as questões mais importantes

de forma que o usuário que está desenvolvendo o projeto não perceba o problema e continue a trabalhar de forma normal.

A execução das atividades se baseia no processo de planejamento. Os planos evoluem à medida que a execução avança, sendo detalhados e modificados, para incorporar novas decisões e para implementar ações corretivas (Maximiano, 2002).

O processo de acompanhamento e controle é a contrapartida do processo de planejamento. Acompanhar a execução de alguma atividade e compará-la com a ação planejada é o objetivo da DIMANAGER. A janela ilustrada na Figura 4.25, mostra os pontos a serem acompanhados pela ferramenta: cronograma, grupos, participantes e problemas.

FIGURA 4.25 – ACOMPANHAR PROJETO

Maximiano (2002) enfatiza que acompanhar tem como funções:

- assegurar que os objetivos sejam alcançados ou preservar um padrão de desempenho;

- mostrar a eventual necessidade de modificar a atividade ou até o resultado esperado;
- verificar se a atividade está sendo realizada.

O objetivo do acompanhamento é disponibilizar ao gerente informações sobre o escopo e o prazo do projeto. Quanto ao escopo, questionamentos devem ser feitos, quanto à realização ou não do projeto, se o projeto fornecerá o resultado esperado, e se há modificações solicitadas pelos clientes ou outros participantes do projeto. O acompanhamento do prazo tem como foco a duração prevista do projeto, as datas previstas para o início e fim de cada fase, e as datas previstas para início e fim das atividades e, deve levantar questões como: as datas previstas estão sendo respeitadas? O projeto está caminhando conforme o cronograma, atrasado ou adiantado? É necessário refazer a programação? Quais as conseqüências caso seja necessário reprogramar o cronograma?

O acompanhamento do projeto envolve atitudes administrativas que vão influenciar no desempenho do projeto. Estas atitudes devem ser executadas sempre em tempo hábil para não prejudicar o projeto.

O acompanhamento do cronograma mostra ao gerente até três níveis de controle: geral (Figura 4.26), detalhado e uma fase específica. (Anexo I)

Maximiano (2002) é bastante realista quando expõe que os cronogramas vão sendo redesenhados conforme as atividades são realizadas. A antecipação ou atraso de datas deve ser registrado no cronograma, para que a equipe possa acompanhar o progresso do projeto e tomar as decisões necessárias.

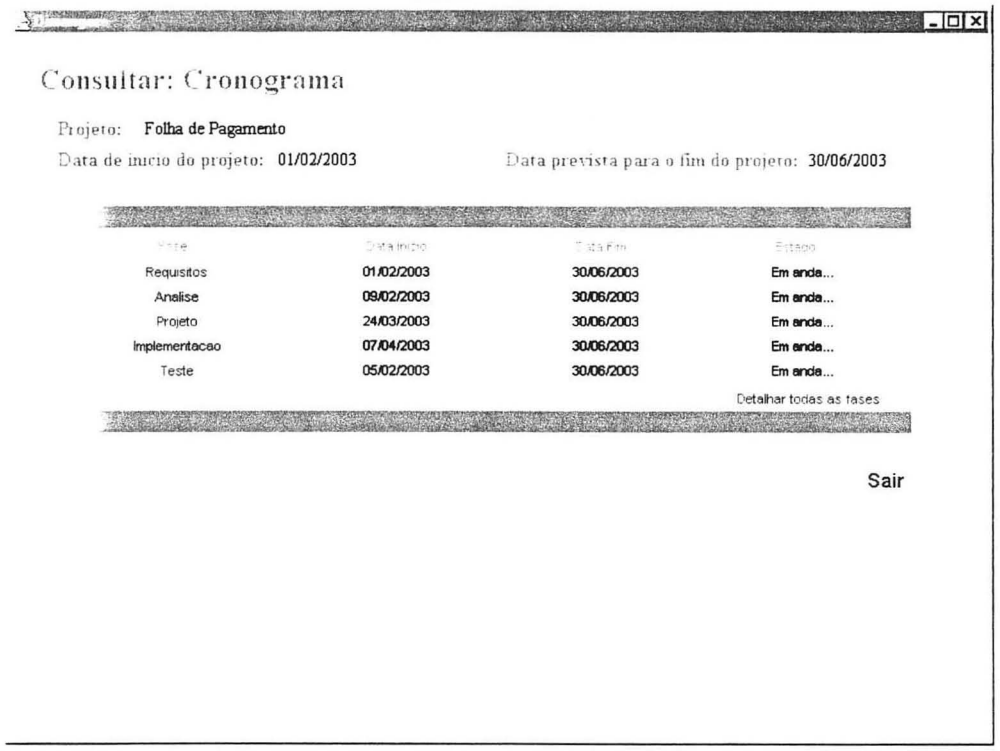

FIGURA 4.26 – ACOMPANHAR CRONOGRAMA - GERAL

A atualização das datas de execução de cada atividade é atualizada no cronograma e as horas trabalhadas são acumuladas, possibilitando a visualização de sua realização.

É importante saber que ao planejar um projeto os problemas internos e externos, já mencionados, irão influenciar no desempenho do projeto. Por isto é que a fase de planejamento é considerada a mais importante e a mais crítica. Após o início dos trabalhos de desenvolvimento do projeto, o gerente deve se preocupar em acompanhar a evolução do mesmo, garantindo a menor variação possível.

Ao atualizar as datas de execução de cada atividade é possível visualizar o desempenho do grupo de trabalho em três níveis de acompanhamento: geral (Figura 4.27), detalhado e um grupo específico (Anexo I).

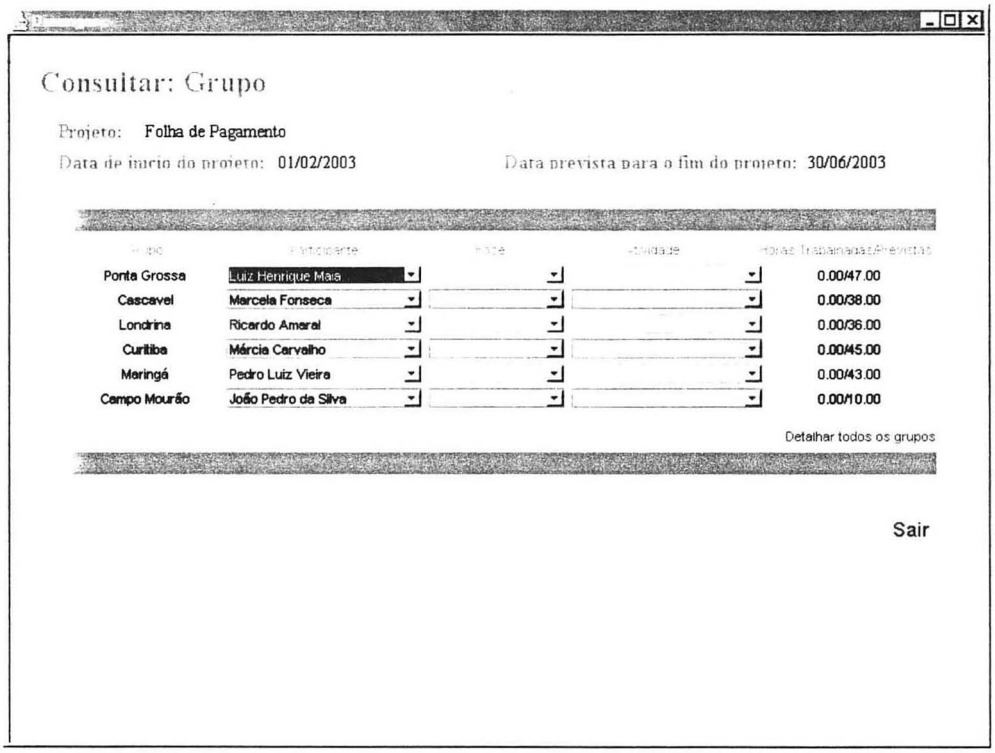

FIGURA 4.27– ACOMPANHAR GRUPO - GERAL

O acompanhamento das atividades executadas de cada participante está disponível ao gerente nos níveis: geral, por atividade e por participante, conforme Figura 4.28.

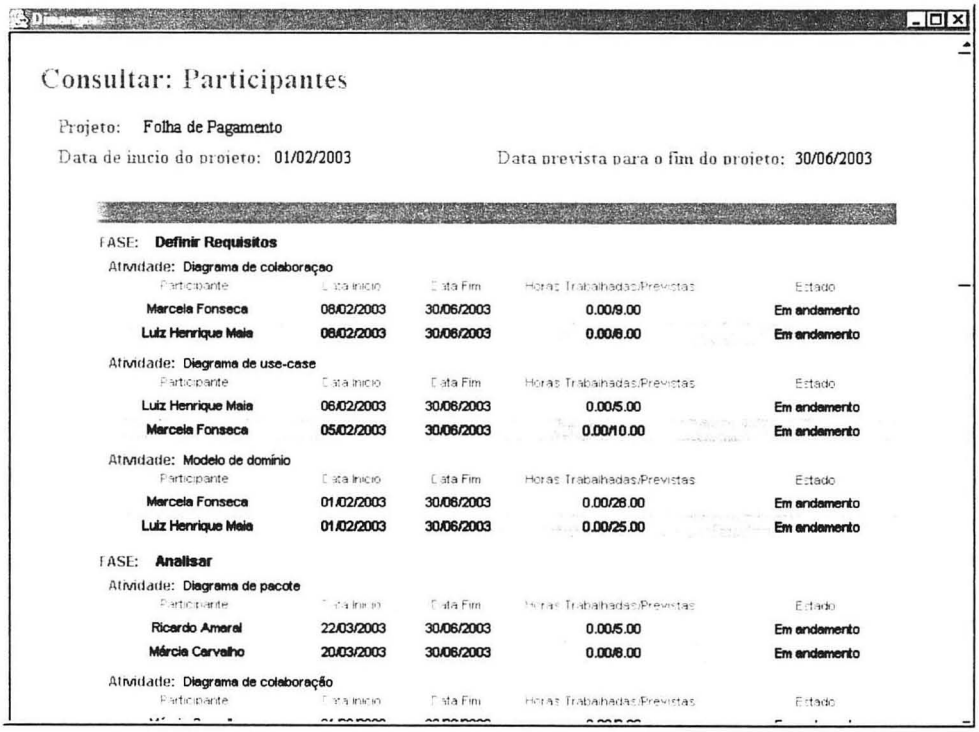

FIGURA 4.28 – ACOMPANHAR PARTICIPANTE – POR ATIVIDADE

Para acompanhar o participante o gerente deve informar o nome do usuário participante, conforme Figura 4.29.

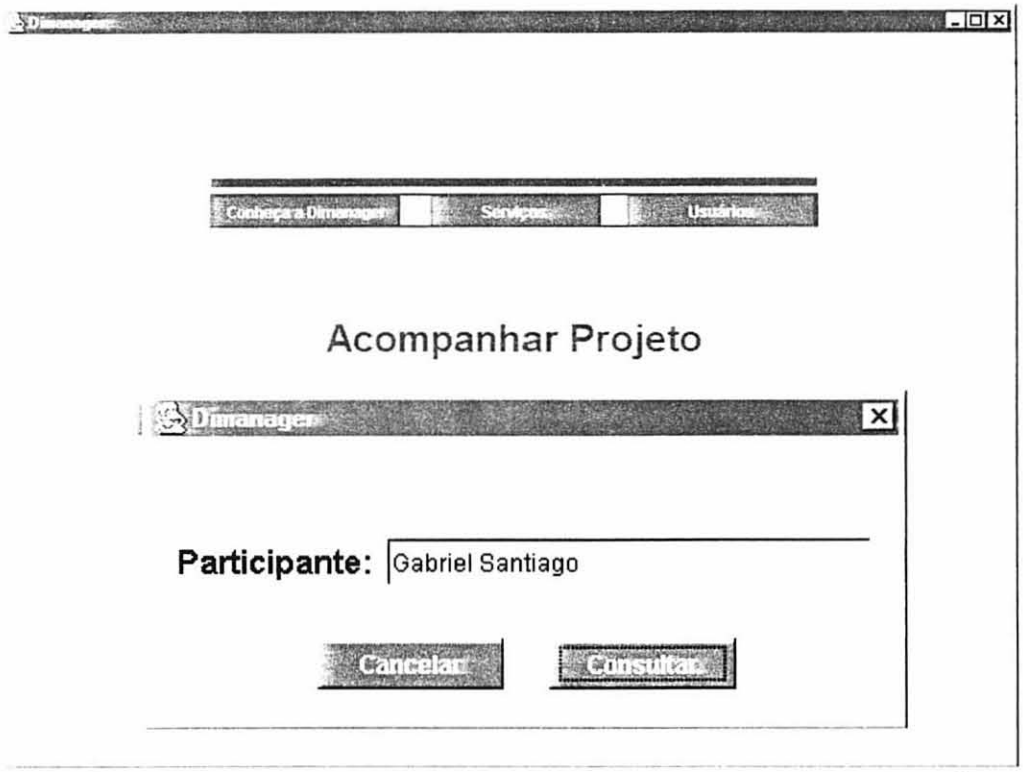

FIGURA 4.29 – ACOMPANHAR PARTICIPANTE – PARTICIPANTE ESPECÍFICO

A atualização das datas de execução de cada atividade, para cada participante, é atualizada no cronograma de participações e as horas trabalhadas são acumuladas, possibilitando a visualização de sua realização, assim como o estado em que se encontra cada atividade.

Desta forma, Cleland e Ireland (2002) colocam que com o objetivo de avaliar-se corretamente o progresso do projeto, algumas condições e entendimentos são necessários, tais como:

- os componentes das equipes devem compreender e estar comprometidos com a importância do processo de monitoração, avaliação e controle do projeto;
- Informações obtidas através da opção "Acompanhar Projeto" são necessárias para a medição do progresso do projeto;
- As atividades de cada fase são as unidades básicas do projeto em torno do qual o progresso do projeto pode ser medido e avaliado;

- Informações usadas para fins de controle do projeto devem ser relevantes, precisas e acessíveis à demarcação de tendências no uso de recursos do projeto;
- A medição dos resultados do projeto deve iniciar com uma avaliação do estado de todas as atividades existentes no projeto;
- Informações coletadas e compiladas a respeito do estado do projeto devem ser ajustadas através do julgamento feito pelos componentes da equipe: gerente, coordenadores e executivos envolvidos no projeto.

Neste contexto, a situação do projeto, envolvendo os problemas encontrados e solucionados torna-se bastante útil. A situação do projeto pode ser obtida através de consulta própria, onde será visualizada a posição geral, detalhada e específica por atividade.

Alguns problemas que ocorrem no desenvolvimento do projeto podem ser previstos e informados ao sistema, possibilitando ao gerente do projeto reduzir suas conseqüências mediante ações diretas, como o desenvolvimento de planos de contingências. Alguns problemas externos podem estar fora do alcance do gerente, quando depender de ações de terceiros.

Os problemas ocorridos no desenvolvimento do projeto podem ser acompanhados através de: uma visão geral (problemas técnicos e organizacionais, problemas técnicos, problemas organizacionais e problemas especificados por fase) (Anexo I).

Cleland e Ireland (2002) colocam que ocorrências de risco constituem os efeitos potenciais adversos ao projeto, sendo ideal a sua identificação durante o início do projeto e, em seguida, no seu planejamento.

O gerente do projeto deve ter alguns atributos básicos para poder gerenciar um projeto com sucesso, sendo elas: ter um certo conhecimento através de aprendizado e experiência; ter e demonstrar habilidades, ou seja, capacidade de usar o conhecimento de forma eficaz e eficiente na execução ou desempenho como gerente de projeto; e, ter atitudes apropriadas, com sentimentos positivos e mente aberta em relação a um fato ou situação.

Cleland e Ireland (2002) acrescentam ainda que uma única falha nestes atributos, pode limitar a capacidade do gerente na administração do projeto.

Embora a DIMANAGER não exiba as informações obtidas no acompanhamento de modo gráfico, as informações poderão ser exportadas para planilhas eletrônicas, possibilitando ao gerente uma visão diversa do desenvolvimento do projeto.

A DIMANAGER possui um padrão de inclusão de informações para que o usuário se familiarize com todas as janelas e não encontre dificuldades em manusear o sistema.

4.5 A Ferramenta DIMANAGER no Ambiente *DiSEN*

Ao planejar, organizar e controlar um projeto, o gerente de projeto deve levar em consideração: quais informações são necessárias para manter os principais gerentes da organização informados sobre o *status* do projeto; quais as informações exigidas que capacitam a alta administração a avaliar a adequação operacional e estratégica do projeto à organização; e se as informações disponíveis são suficientes para a tomada e implementação de decisão (Cleland e Ireland, 2002). Estas informações devem estar disponíveis para o gerente de projeto e usuários autorizados e são armazenadas no Repositório (Moro, 2003) do ambiente *DiSEN*.

A DIMANAGER disponibiliza ao gerente de projeto várias informações relevantes ao acompanhamento do projeto, tais como: o estado das atividades por fase, grupo e participante; o cronograma; o desempenho técnico dos participantes; e a situação das atividades em relação aos problemas que possam ocorrer durante o desenvolvimento das mesmas.

A troca de informações entre a DIMANAGER e as ferramentas disponíveis no ambiente *DiSEN* ocorre através do Repositório. A Figura 4.30 mostra uma visão macro desta troca de informações.

As informações contidas no planejamento do projeto ficam armazenadas no Repositório e não podem ser alteradas pelos usuários de desenvolvimento. O gerente

de projeto é o único responsável por qualquer alteração necessária, caso se verifique alguma distorção no acompanhamento do resultado esperado.

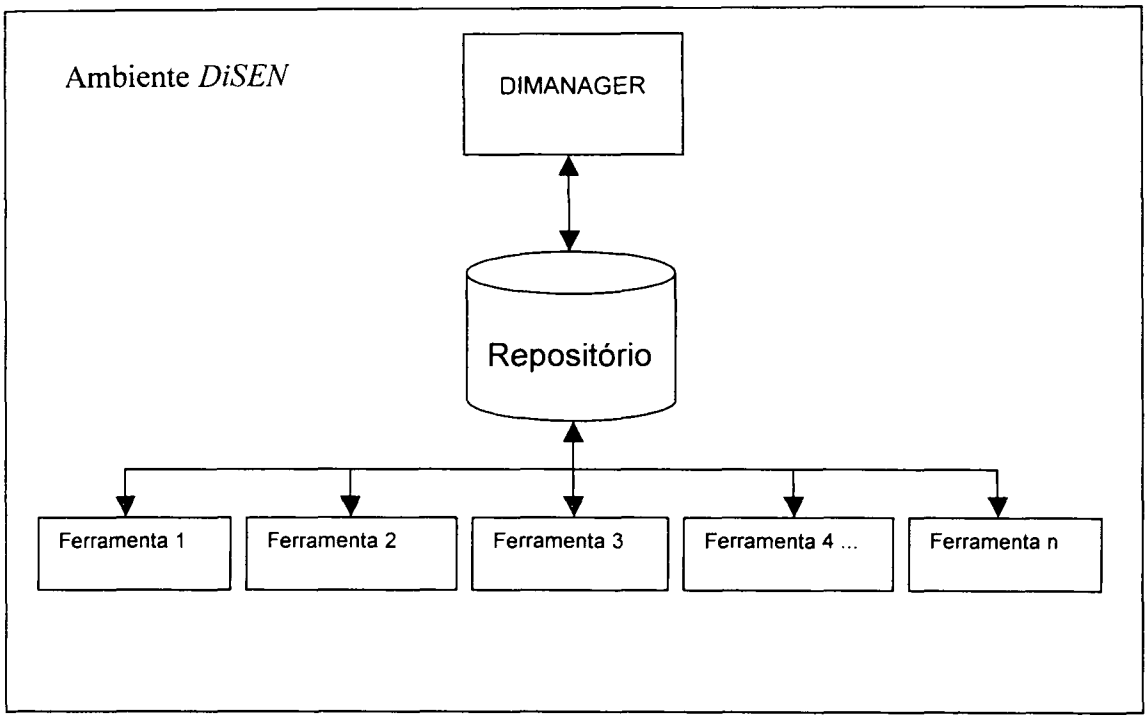

FIGURA 4.30 – A FERRAMENTA DIMANAGER NO AMBIENTE *DiSEN*

Os usuários, ao fazerem uso das ferramentas disponíveis no ambiente *DiSEN* para dar continuidade ao desenvolvimento do projeto, devem acessar o Repositório e verificar a validade de seus acessos. No repositório são mantidos LOG's de participantes e cronograma. Estes LOG's serão atualizados sempre que um usuário estiver desenvolvendo qualquer atividade planejada.

É através dos LOG's de participantes e cronograma que as informações sobre o acompanhamento do projeto serão disponibilizadas ao gerente de projeto. Estas informações devem ser armazenadas e recuperadas através do Repositório.

A DIMANAGER foi desenvolvida para auxiliar o gerente de projeto no desempenho e acompanhamento das atividades, mas algumas funções são exclusivas do gerente, tais como: manter a equipe bem informada sobre o progresso do projeto; facilitar a comunicação entre os *stakeholders*; prever os possíveis resultados do uso dos recursos destinados ao projeto; testar estratégias no uso de recursos no projeto; compreender a necessidade de mudanças e os possíveis resultados no projeto; enfim,

analisar as condições passada, presente e futura do projeto em andamento. Um planejamento bem elaborado em conjunto com as atribuições gerenciais tem menos chances de um resultado inadequado e até mesmo indesejável.

4.6 Considerações Finais

Neste capítulo foi apresentada a ferramenta de apoio ao gerenciamento de desenvolvimento de *software* distribuído, DINAMAGER. As características e diferenças encontradas nas três ferramentas (Pfaffenseller, Pfaffenseller e Kroth, 2001), (Lima, Reis e Nunes, 1998) e (Oliveira, Rouiller e Vasconcelos, 2001) abordadas, juntamente com os conceitos de *software* distribuído, gerenciamento de projeto de *software* e ferramentas *CASE* e *I-CASE*, possibilitaram o levantamento de elementos para o desenvolvimento desta ferramenta, visando o gerenciamento em *software* distribuído.

A DIMANAGER foi desenvolvida para a administração do projeto em si. Após o esclarecimento das necessidades a serem atendidas, um escopo coerente deve ser definido e, em seguida, o prazo e futuramente a inclusão do custo, possam ser planejados. A DIMANAGER deve gerenciar esta área operacional do projeto, com a finalidade de abranger o desenvolvimento de *software* distribuído, agilizando a produção, mantendo grupos de trabalho dispersos geograficamente, e gerenciando o desenvolvimento do *software*, com o planejamento dos objetivos, a organização e controle das atividades e, a coordenação das equipes de trabalho.

Os elementos aqui relacionados, como: as razões que justificam assim como os elementos que fundamentam a grande complexidade na construção de sistemas distribuídos Bessani (2000); as vantagens do uso das ferramentas *CASE* (Artigo Técnico Dr. *CASE* 3.0); os requisitos básicos de um ambiente *CASE* integrado Pressman (2002); as dificuldades inerentes ao gerenciamento de projeto de *software* (Oliveira, Rouiller e Vasconcelos, 2001); as fases de planejamento de projeto (Teixeira, 2001); as características das três ferramentas estudadas (Pfaffenseller, Pfaffenseller e Kroth, 2001), (Lima, Reis e Nunes, 1998) e (Oliveira, Rouiller e

Vasconcelos, 2001); a MDSODI (Gravena, 2000); o ambiente *DiSEN* (Pascutti, 2002) e o Repositório (Moro, 2003), formaram a base para a formatação da ferramenta de apoio ao gerenciamento de desenvolvimento de *software* distribuído - DIMANAGER.

Capítulo 5 Considerações sobre a DIMANAGER

5.1 A Ferramenta DIMANAGER e o Gerenciamento de Projeto de *Software* Distribuído

O gerenciamento de projeto tem como objetivo assegurar que os processos estabelecidos sejam seguidos, coordenando e monitorando as atividades da engenharia de *software* e uma ferramenta que possibilite ao gerente de projeto controlar e acompanhar o desenvolvimento de *software*, permite otimizar recursos e controlar as atividades e as pessoas envolvidas.

É importante salientar que ao planejar o projeto deve-se elaborar o escopo, prazo e custo do mesmo, que dependerão do tempo investido. Para Maximiano (2002), o acompanhamento das atividades desenvolvidas torna-se mais claro quando o planejamento é elaborado com um grau maior de detalhamento.

Assim, a ferramenta DIMANAGER aqui apresentada está direcionada ao gerenciamento do projeto, no desenvolvimento distribuído de software. Elementos como: cronograma, participantes, atividades, problemas entre outros são tratados pela ferramenta.

A Figura 5.1 representa a DIMANAGER no gerenciamento distribuído de desenvolvimento de *software*. Os *softwares* devem ser identificados, podendo estar ou não distribuídos na rede. É importante planejar as atividades, distribuindo as equipes de trabalho dentro de cada uma delas, estabelecendo o tempo de duração de cada atividade.

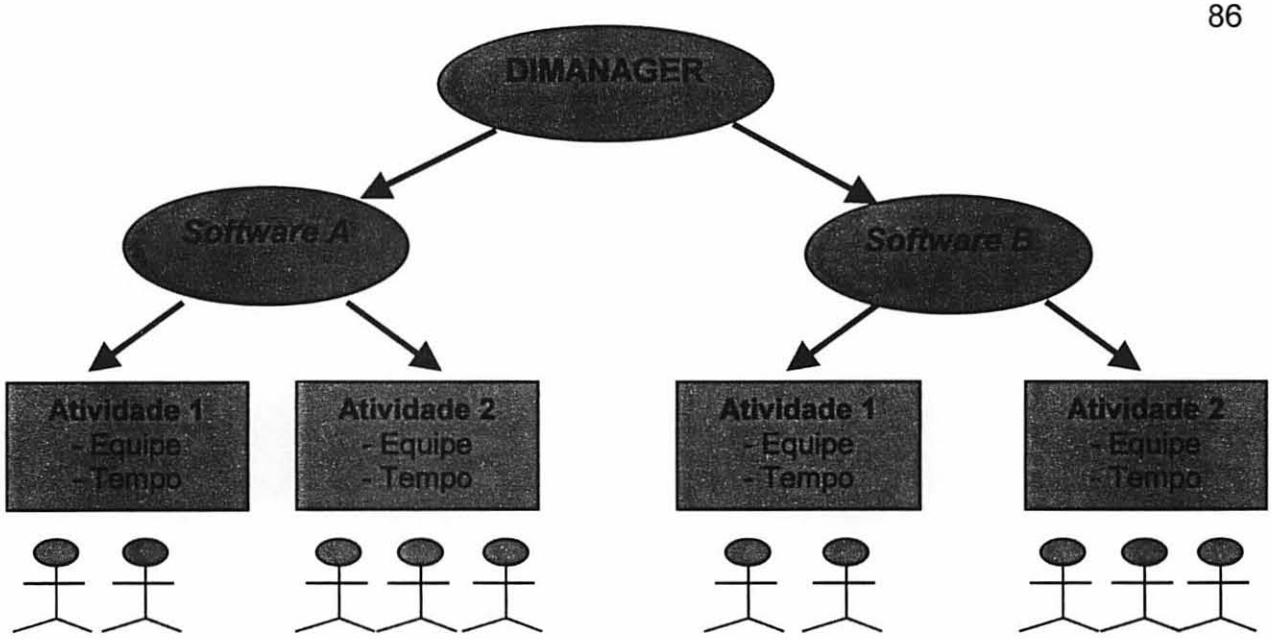

FIGURA 5.1 – A FERRAMENTA DIMANAGER NO GERENCIAMENTO DISTRIBUÍDO DE *SOFTWARE*

Administrar pessoas não é uma tarefa fácil. Administrar pessoas em locais dispersos geograficamente é uma tarefa ainda mais complexa, exigindo do gerente de projeto uma avaliação constante do andamento do mesmo. O gerente de projeto deve se preocupar com as atualizações assim como o resultado do projeto e a dinamicidade dos processos. Para obter um acompanhamento mais preciso do desenvolvimento distribuído de atividades o gerente do projeto deve estar sempre em comunicação com os coordenadores de áreas que, por sua vez, também devem estar distribuídos geograficamente.

Com base nos princípios apresentados por Teixeira (2001) no capítulo três, a ferramenta proposta deve: estabelecer uma divisão de trabalho em fases e atividades independentes; determinar o tamanho do sistema; elaborar um cronograma de atividades; definir os riscos do projeto; revisar e avaliar o plano de projeto; e aprovar o plano de projeto através de um documento contendo todas estas informações. Por outro lado, também deve considerar outros aspectos importantes colocados por Bessani (2000), como: o compartilhamento de recursos; a independência de plataforma de *hardware* e *software*; o controle de concorrência; a independência de escala onde o sistema distribuído deve trabalhar tão bem com 10 máquinas quanto com 100; a tolerância a falhas; a transparência e o estado compartilhado.

Esta ferramenta além de gerenciar as fases estabelecidas na MDSODI: requisitos, análise, projeto, implementação e teste (Gravena, 2000), apresenta dois subsistemas: o do planejamento e o do acompanhamento do projeto. O fluxo de mensagens estabelecido ocorre entre a ferramenta de apoio ao gerenciamento e as ferramentas de desenvolvimento do *software* através do Repositório. Destacamos ainda a participação dos seguintes atores: o gerente de projeto e os coordenadores de atividades, participantes e cronograma. Cada um dos atores tem uma atividade bem definida o que contribui significativamente para o sucesso do projeto.

A definição do Modelo de Domínio da metodologia MDSODI foi importante para a definição das fases e da elaboração da ferramenta de apoio ao gerenciamento de *software* distribuído - DIMANAGER. O modelo desenvolvido para a DIMANAGER transcorreu mais facilmente.

5.2 A Ferramenta DIMANAGER e as Ferramentas Estudadas

As ferramentas estudadas: ToolManager: Uma Camada de Gerenciamento de Ferramentas CASE a um Ambiente Centrado no Processo; uma Ferramenta de Apoio ao Gerenciamento de Componentes e a Ferramenta para Gerenciamento do Processo de Desenvolvimento Cooperativo de *Software* no Ambiente PROSOFT, serviram de apoio e orientação para a elaboração da ferramenta DIMANAGER.

O Quadro 2 apresenta uma comparação entre as ferramentas estudadas e a ferramenta DIMANAGER quanto aos seguintes aspectos: foco, ambiente de gerenciamento de *software*, objetivos, mensagens, tarefas de gerenciamento, atores envolvidos e aspectos específicos de cada uma.

QUADRO 2 – COMPARAÇÃO ENTRE AS FERRAMENTAS ESTUDADAS E A FERRAMENTA DIMANAGER

	ToolManager	**Apoio ao Desenvolvimento de Componentes**	**PROSOFT**	**DIMANAGER**
Foco	Processo	Componentes	Processo	Processo
Ambiente de Gerenciamento de *Software*	ProjectSpace	Não Especifica	Prosoft	*DiSEN*
	Gerenciar questões referentes ao uso, registro e controle das ferramentas *CASE* durante a execução das atividades definidas em um plano de projeto.	Integrar ferramentas de modelagem, controlar versões, trabalho em grupo e políticas de manutenção abrangendo novos conceitos como componentes e reuso de *software*.	Apoiar o desenvolvimento formal de *software*, providenciando a integração dos dados, do controle e de apresentação entre suas ferramentas.	Controlar e acompanhar o desenvolvimento de projetos de *software*, viabilizando ao gerente de projeto coordenar e controlar as atividades e as pessoas envolvidas no desenvolvimento de um *software* distribuído.
Mensagens	Fluxo de mensagem que controla a passagem de informações entre os diferentes níveis de gerenciamento, facilitando a comunicação entre eles.	O fluxo de mensagem ocorre através da diagramação formal, definida como sendo a ferramenta adequada para que haja troca de informações entre os vários desenvolvedores que trabalham em conjunto no desenvolvimento de um sistema.	Os ATOs se comunicam através de troca de mensagens feita pela ICS – Interface de Comunicação de Sistema, responsável pela comunicação entre o PROSOFT e os ATOs.	O Fluxo de mensagens entre a DIMANAGER e as ferramentas do ambiente DiSEN ocorre através do Repositório de Dados que é o elemento responsável em armazenar as informações que atualizarão as bases de acompanhamento do projeto.
Tarefas de Gerenciamento	Oferece apoio ao gerenciamento das ferramentas a serem utilizadas no ambiente Project Space.	Oferece apoio ao gerenciamento de componentes de desenvolvimento de *software*.	Oferece apoio ao desenvolvimento formal de *software*.	Oferece apoio ao gerenciamento do desenvolvimento distribuído do *software*.

QUADRO 2 – COMPARAÇÃO ENTRE AS FERRAMENTAS ESTUDADAS E A FERRAMENTA DIMANAGER (CONTINUAÇÃO)

	ToolManager	**Apoio ao Desenvolvimento de Componentes**	**PROSOFT**	**DIMANAGER**
Atores envolvidos	• Gerente • ProjectSpace	• Administrador do repositório • Desenvolvedor	• Super-usuário • Gerentes • Usuários-comuns	• Gerente do Projeto • Coordenador de Atividades • Coordenador de Participantes • Coordenador de Cronograma
Aspectos específicos	Trabalha em camadas e possui 4 níveis de gerenciamento: • Gerenciador de Apresentação; • Gerenciador de Controle; • Gerenciador de Atividade; • Gerenciador de Dados.	Possui um Repositório de Componentes que permite aos desenvolvedores o acesso às informações referentes aos componentes existentes.	Composta por ATO's – Ambientes de Tratamento de Objetos.	• Faz uso do Repositório de Dados desenvolvido para o ambiente *DiSEN*; • Faz uso das fases da metodologia MDSODI – fases; • Salienta os aspectos técnicos e organizacionais no desenvolvimento do *software* distribuído.

Todas as ferramentas foram desenvolvidas para facilitar o processo de desenvolvimento de *software*, mas apenas a DIMANAGER salienta os aspectos técnicos e organizacionais envolvidos no desenvolvimento do *software* distribuído.

5.3 A Ferramenta DIMANAGER e os Aspectos Técnicos e Organizacionais

Os aspectos técnicos e organizacionais devem ser considerados na elaboração de projetos. Os aspectos técnicos estão relacionados a toda a infraestrutura física necessária para o planejamento e desenvolvimento do projeto e devem ser encarados com bom senso, acompanhando a evolução crescente, sem no entanto, desprezar a experiência e potencialidade do gerente responsável pelo projeto. Os aspectos organizacionais englobam recursos humanos, negócios e metas, e devem ser tratados pela administração que deve considerar todos os elementos que, ignorados, podem levar ao insucesso (Tait, 2000). Em relação a estes aspectos a DIMANAGER procurou disponibilizar ao gerente a documentação de todos os problemas encontrados, tanto técnicos quanto organizacionais, para auxiliar as pessoas envolvidas no projeto. O objetivo também é disponibilizar esta documentação para outros projetos evitando assim atrasos e desgastes desnecessários da equipe de desenvolvimento.

5.4 A Ferramenta DIMANAGER e as Métricas

Um aspecto relevante a ser considerado é a questão das métricas. Segundo Pressman (2002) na maioria dos empreendimentos técnicos, as medições e as métricas ajudam a entender o processo técnico usado para se desenvolver um produto, como também o próprio produto. O processo é medido, num esforço para melhorá-lo, assim como o produto é medido, num esforço para aumentar sua qualidade. A medição capacita a quantificar e também a administrar mais efetivamente. Mas alguns pontos devem ser abordados tais como: Quais são as métricas apropriadas para o processo e para o produto? Como os dados compilados devem ser usados? É justo usar medições para se comparar pessoas, processos e produtos? Essas e outras perguntas sempre aparecem quando é feita a tentativa de se medir a engenharia de *software* (o processo) e o *software* (o produto).

Pressman (2002) coloca ainda que uma das atividades fundamentais do processo de gerenciamento de projetos de *software* é o *planejamento*. Quando o projeto de *software* é planejado, estimativas do esforço humano exigido (normalmente,

pessoas-mês), duração cronológica do projeto (em tempo de calendário) e custo devem ser derivadas. Desta forma, as medidas e métricas a serem estabelecidas para cada atividade devem ser escolhidas de acordo com as técnicas de estimativas disponibilizadas para o desenvolvimento de *software*, podendo ser: estimativas de linhas de código (LOC), estimativas de pontos-por-função (FP) ou estimativas do esforço. Gomes et al. (2001) acrescentam que muitas métricas foram propostas e aplicadas em casos práticos, mas poucos foram os casos onde o seu uso não foi considerado ineficiente e até mesmo desaconselhável.

Diante desta complexa realidade, o presente trabalho estabeleceu que a métrica utilizada na ferramenta DIMANAGER é a do esforço humano (humano-hora). Esta métrica está sendo adotada tendo como base dois aspectos: as pessoas que trabalharão no projeto, e as horas gastas em cada atividade do projeto.

5.5 Validação da Ferramenta DIMANAGER

As empresas participantes da validação foram 4: 3 empresas de grande porte e 1 de médio porte da cidade de Maringá, sendo duas do setor público, uma do setor privado e outra do ramo de cooperativa. As grandes empresas contam em torno de 4000 colaboradores e a de médio porte com 600 colaboradores. A ferramenta DIMANAGER foi avaliada por profissionais da área de informática que atuam nas áreas de análise, programação, gerência e direção. Estes profissionais tem em torno de 10 anos de experiência nesta área.

A validação da ferramenta se deu em dois aspectos: quanto aos indicadores de aspectos gerenciais e a ferramenta propriamente dita.

A validação dos indicadores de aspectos gerenciais foi complementada por Santiago e Tait (2003). Este trabalho avaliou os seguintes aspectos da ferramenta DIMANAGER: quanto ao processo de gerenciamento de projetos; às necessidades do gerente de projetos; os dados para acompanhamento; a organização dos indicadores; a inclusão de restrições e o resultado geral.

Quanto ao primeiro aspecto, o processo de gerenciamento de projetos, concluiu-se que os indicadores abordam aspectos importantes nos processos de

desenvolvimento de projetos, auxiliando no acompanhamento do mesmo. Embora o processo de gerenciamento de projetos é muito complicado, principalmente quando se trata de ambientes distribuídos, conforme já referenciado no capítulo 3, tais indicadores abordam itens básicos necessários ao gerenciamento.

O segundo aspecto, que tratou das necessidades do gerente de projetos, verificou-se que o espaço de abrangência dos indicadores levantados é de grande importância e valia, mas alguns aspectos devem ser considerados, podendo ser citados: a necessidade de um aprimoramento no inter-relacionamento entre as fases e atividades, especificando quando uma atividade pode ter início após o término da anterior, tratando de forma mais detalhada esta dependência; e que os indicadores especificados abrangem aspectos de acompanhamento de controle, não tratando os aspectos gerenciais do projeto, sendo este último necessário para a obtenção de resultados positivos.

O terceiro aspecto, os dados para o acompanhamento do projeto foram considerados satisfatórios, mas algumas informações podem ser acrescentados: pré-requisitos para cada uma das atividades do cronograma, para melhor escolher os participantes daquela atividade; e, exibir o perfil técnico de cada participante, com informações sobre suas habilidades profissionais, a fim de melhor adequá-los às fases e respectivas atividades do projeto.

Quanto à organização e apresentação dos indicadores, foram considerados de forma adequada e satisfatória, possibilitando uma boa análise e interpretação dos dados.

O quinto aspecto, tratou a necessidade da inclusão de outros indicadores no gerenciamento de projetos. Os profissionais sugeriram a inclusão de um indicador para mostrar as restrições impostas ao projeto, naquilo que se refere a custos, tempo, qualidade do produto final ou qualquer outro aspecto importante.

Quanto ao sexto e último aspecto, foi ressaltada a não abrangência dos indicadores quanto ao planejamento do projeto e como sugestão foi colocada que, sendo uma ferramenta de gerenciamento de desenvolvimento de projetos de *software* distribuído, é interessante a implementação de um sistema de monitoramento

automático do andamento do projeto, com envio de mensagens, através de correio eletrônico, para alertar o gerente do projeto sobre os atrasos ocorridos no desenvolvimento do mesmo ou sobre os participantes que não correspondam às expectativas iniciais.

Para a avaliação da ferramenta dois métodos de validação foram utilizados: uma entrevista com o preenchimento de um questionário (Anexo IV) e a apresentação da ferramenta DIMANAGER. Esta avaliação abordou os seguintes itens: visualização, utilidade para o gerenciamento de projeto, aplicabilidade no trabalho de gerenciamento e sua utilidade para projetos desenvolvidos em locais dispersos geograficamente.

Quanto à visualização da ferramenta DIMANAGER, os profissionais foram unânimes em afirmar que suas etapas estão claras assim como a utilização de cada uma delas.

A ferramenta tem utilidade para o gerenciamento de projetos onde controla cada fase, desde o planejamento do cronograma até os testes, e facilita o trabalho do gerenciamento pois serve de apoio e busca das informações dando um maior controle sobre o andamento do projeto.

A estrutura da ferramenta DIMANAGER apresenta utilidade para projetos desenvolvidos em locais dispersos geograficamente, pois sua estrutura deve funcionar em redes internas e externas, como na Internet, de acordo com afirmação dos participantes da pesquisa.

Algumas sugestões foram colocadas para melhorar a apresentação e o manuseio da ferramenta, tais como: melhorar o dimensionamento das telas para uma visualização mais global das informações de cada janela; disponibilizar todos os projetos cadastrados para que o gerente escolha o projeto a ser alterado/acompanhado; formatação dos campos de data para que o usuário não fique desorientado quanto ao preenchimento da informação; relatório com percentual de desempenho por grupo/fase e do projeto como um todo.

Uma sugestão colocada por um dos profissionais foi a inclusão de uma janela inicial, com o "login" do usuário e senha. Esta sugestão não foi aceita pois a

ferramenta está disponível para que qualquer usuário possa conhece-la através da primeira opção, "Conheça a DIMANAGER", mostrada na figura 4.16.

Em termos gerais, o resultado da validação foi satisfatório, possibilitando a implementação das experiências de cada profissional e a confirmação da relevância da ferramenta DIMANAGER, expressa pelos entrevistados afirmativamente na frase "sua estrutura serve tanto para redes internas quanto externas ou Internet", a qual resume a opinião dos mesmos.

Vale frisar também dois aspectos abordados pelos entrevistados: a questão da ferramenta apontar para as falhas do projeto; e, a utilização do cronograma de execução do projeto, pelo acompanhamento de cada fase.

5.6 Trabalhos Futuros

A partir da elaboração da ferramenta DIMANAGER, cinco elementos são sugeridos para implementação em trabalhos futuros: 1) custos; 2) interface para o repositório de dados; 3) controle de versões; 4) relatórios gerenciais na DIMANAGER; e, 5) avaliação da ferramenta DIMANAGER no ambiente *DiSEN*.

Com relação ao elemento custos, sabe-se de sua relevância para o gerenciamento de projetos, no entanto, não fez parte dos elementos componentes da DIMANAGER nesta fase de elaboração, cuja preocupação maior voltou-se para a produtividade no desenvolvimento de projetos.

Para a execução de um projeto as variáveis: métricas e custos, são importantes e os custos financeiros envolvem tanto os recursos humanos quanto os materiais. Estes recursos podem fazer parte do planejamento do gerente mas não estarão sendo controlados pela DIMANAGER.

O ambiente *DiSEN* já conta com um Repositório remoto de metadados (Moro, 2003) que pode ser utilizado em um ambiente distribuído de desenvolvimento de *software* por uma ou mais ferramentas, possibilitando a estas o acesso às funcionalidades do repositório através da rede. A troca das informações de forma automatizada, entre as ferramentas de desenvolvimento e a DIMANAGER, através do Repositório ainda não está disponível.

Em relação ao controle de versões, cada ferramenta desenvolvida para trabalhar sob a supervisão da DIMANAGER deverá ter uma política de manutenção, mantendo atualizações permanentes no LOG (histórico de ocorrências). Este LOG deverá prever as atividades paralelas e concorrentes de cada fase e disponibilizar informações para que a DIMANAGER acompanhe o desenvolvimento de cada atividade.

Optou-se pela inclusão de relatórios gerenciais como trabalho futuro pela importância dada a este tipo de relatório pelos entrevistados, os quais, inclusive, sugeriram a apresentação das seguintes informações: uma visão geral do projeto, com o percentual de produtividade prevista em comparação com o realizado, bem como os problemas ocorridos em cada fase e o estado em que se encontra cada uma delas.

Esta inclusão se justifica pela necessidade de os gerentes de desenvolvimento de *software* controlarem o projeto, realizando avaliações sobre a situação do mesmo.

A avaliação da DIMANAGER no ambiente *DiSEN* será possível quando todas as ferramentas envolvidas no Projeto de Pesquisa (Huzita, 1999) estiverem prontas e integradas, fazendo com que o ambiente funcione como um todo.

5.7 Contribuições da Pesquisa para o Conhecimento

Um aspecto relevante na construção da ferramenta DIMANAGER é a abordagem de aspectos gerenciais no desenvolvimento de software distribuído. Ao tratar duas áreas com preocupações aparentemente distintas, chega-se a construção de uma ferramenta de apoio ao desenvolvimento de *software* distribuído que contempla em seu bojo elementos comumente considerados como preocupações gerenciais.

A preocupação com aspectos meramente técnicos leva, muitas vezes, a desconsiderar elementos da própria estrutura organizacional (Tait, 2000) que possibilitam o desenvolvimento de um projeto mais completo e integrado as questões inerentes a cada organização. Inicialmente, as pesquisas em *software* distribuído voltam-se para aspectos classificados como técnicos, tais como queda de rede, tolerância a falhas etc.

Esta pesquisa, além de tratar os aspectos técnicos devido a sua óbvia importância, acrescenta no cenário do desenvolvimento do *software* distribuído, os

aspectos gerenciais, cuja abordagem volta-se para as pessoas envolvidas, suas atividades, seu cronograma de desenvolvimento etc.

Reafirma-se, desta forma, a relevância da integração de aspectos técnicos e gerenciais, notadamente em uma área complexa e em expansão, como é o desenvolvimento do *software* distribuído.

Além da construção da ferramenta DIMANAGER e de sua integração com o ambiente *DiSEN*, que por si, já tem grande significado como contribuição, pode-se considerar o tratamento dos aspectos gerenciais no *software* distribuído como um elemento que se consolida e que, certamente, por ser o início de uma área de atuação, ainda trará muitos frutos na ligação de aspectos gerenciais e desenvolvimento de *software* distribuído.

REFERÊNCIAS

Artigo Técnico do Dr. *CASE* 3.0. Disponível em: <http://squadra.com.br> Acesso em: 20 set. 2001.

BARRETO, R. **Sistemas Distribuídos.** Disponível em: < http://www.dcc.fua.br/~barreto/redes/introducao_sist_distr.ppt> Acesso em: 27 jun. 2003.

BATISTA, S. M. **Uma Ferramenta de Apoio à Fase de Requisitos da MDSODI no Contexto do Ambiente *DiSEN*.** Curitiba-PR, 2003. 83 f. Dissertação (Mestrado em Informática) – Setor de Ciências Exatas, Universidade Federal do Paraná em convênio com a Universidade Estadual de Maringá.

BESSANI, A. N. **Análise Comparativa entre Arquiteturas de Conectividade de Sistemas Distribuídos: CORBA e Jini.** Maringá-PR, 2000. Trabalho de Graduação - Curso de Ciência da Computação, Departamento de Informática. Universidade Estadual de Maringá.

BORGHOFF, U. M.; SCHLICHTER, J. H. **Computer-Supported Cooperative Work Introdution to Distributed Applications.** Germany: Springer, 2000.

CLELAND, D. I.; IRELAND, L. R. **Gerência de Projetos.** Rio de Janeiro-RJ: Reichmann & Affonso Editores, 2002.

FURLAN, J. D. **Modelagam de Objetos através da UML.** São Paulo: MAKRON Books, 1998.

GOMES, A.; SÔMULO M.; OLIVEIRA, K.; ROCHA, A. R. Avaliação de Processos de *Software* na Estação Taba. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE *SOFTWARE*, XV, 2001, Rio de Janeiro-RJ. **Anais.** Rio de Janeiro-RJ, 2001, p. 344-349.

GRAVENA, J. P. **Aspectos importantes de uma metodologia para desenvolvimento de *software* com objetos distribuídos.** Maringá-PR, 2000, 64 f. Trabalho de Graduação - Curso de Ciência da Computação, Departamento de Informática. Universidade Estadual de Maringá.

HUZITA, E. H. M. **MOOPP: Uma metodologia para auxiliar o desenvolvimento de aplicações para processamento paralelo.** São Paulo-SP, 1995. Tese de doutorado. Universidade de São Paulo.

HUZITA, E. H. M. **Metodologia para desenvolvimento baseada em objetos distribuídos inteligentes.** Maringá-PR, 1999. Projeto de Pesquisa em

Desenvolvimento na área de informática - Departamento de Informática. Universidade Estadual de Maringá.

Informativo Técnico. Disponível em: <http://www.revista.unicamp.br/infotec/informação/inf44.htm> Acesso em: 06 set. 2001.

JACOBSON, L.; BOOCH, G.; RUMBAUGH, J. **The Unified *Software* Development Process.** Assison Wesley, 1999.

LIMA, C. A. G. de; REIS, R. Q.; NUNES, D. J. Gerenciamento do Processo de Desenvolvimento Cooperativo de *Software* no Ambiente PROSOFT. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE *SOFTWARE*, XII, 1998, Maringá-PR. **Anais.** Maringá-PR, 1998, p. 221-236.

LIMA Jr, L. A. de P. Objetos Distribuídos. In: ESCOLA DE INFORMÁTICA DA SBC-SUL – INSTITUTO DE INFORMÁTICA-UFRGS, IX, 2001, Maringá. **Anais.** Maringá-PR, 2001, p. 145-173.

MASON, V.; BUDA, M. A.; GAZETA, W. **Ferramentas *CASE.*** Disponível em: <http://www.cazarini.cpd.eesc.sc.usp.br/sep5745/1999/sep5745-Gr5/*CASE*-f/sld001.htm> Acesso em: 17 jul. 2002.

MAXIMIANO, A. C. A. **Administração de Projetos.** 2ª. Edição. São Paulo: Editora Atlas S.A., 2002.

MORO, C. F. **Suporte à Persistência de Artefatos para o Ambiente Distribuído de Desenvolvimento de *Software DiSEN.*** Curitiba-PR, 2003. 100 f. Dissertação (Mestrado em Informática) – Setor de Ciências Exatas, Universidade Federal do Paraná em convênio com a Universidade Estadual de Maringá.

MSDCOM Microsoft Co. **Distributed Component Object Model - DCOM.** Disponível em: <http://www.microsoft.com/com/tech/dcom.asp> Acesso em: 10 set. 2003.

OLIVEIRA, S. R. B.; ROUILLER, A. C.; VASCONCELOS, A. M. L. de. ToolManager: Uma Camada de Gerenciamento de ferramentas *CASE* a um Ambiente Centrado no Processo. In: CONFERÊNCIA INTERNACIONAL DE TECNOLOGIA DE *SOFTWARE*: QUALIDADE DE *SOFTWARE,* XII, 2001, Curitiba-PR. **Anais**, 2001. p. 84-94.

OMG **CORBA: The Common Object Broker – Architecture and Specification.** Version 2.4.1, Nov. 2000.

ORFALI, H. E. **The essential distribuited objetct survival guide.** John Wiley & Sons, 1996.

PASCUTTI, M. C. D. **Uma Proposta de Arquitetura de um Ambiente de Desenvolvimento de *Software* Distribuído Baseada em Agentes.** Porto Alegre-RS, 2002, 101 f. Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande do Sul.

PFAFFENSELLER, M.; PFAFFENSELLER, M.; KROTH E. Uma ferramenta de apoio ao Desenvolvimento de *Software* Baseado em componentes. In: WORKSHOP DE DESENVOLVIMENTO BASEADO EM COMPONENTES, I, 2001, Maringá-PR. **Anais.** Universidade Estadual de Maringá - Maringá-PR, 2001.

PRESSMAN, R. S. **Engenharia de *Software*.** 5ª. Edição. São Paulo: MCGRAW-Hill Interame, 2002.

SANTIAGO, G. P.; TAIT, T. F. C. **Indicadores de Aspectos Gerenciais para o Desenvolvimento de *Software* Distribuído.** Relatório de Projeto de Iniciação Científica – PIBIC – CNPQ, 2002-2003.

SCHELEBB, H. **Distribuited PROSOFT.** Departament of Computer Science, Technical Report, University of Stuttgart, Germany, 1995.

SOBRINO, A. **Ferramentas *CASE*: Uma Visão Complementar.** Disponível em: <http://www.stcecilia.br/~sobrino/prov-ana-*CASE*.htm> Acesso em: 22 jul. 2002.

SUN Microsystems, Inc. **Java Remote Method Invocation Specification.** Revision 1.7, Java 2 SDK, Standard Edition, v1.3.0, Dec. 1999.

TAIT, T. F. C. **Um modelo de arquitetura de sistemas de informação para o setor público: estudo em empresas estatais prestadoras de serviços de informática.** Florianópolis, 2000. 242 f. Tese de doutorado. Universidade Federal de Santa Catarina.

TEIXEIRA J. W. Proposta de um Modelo de Planejamento de Projeto de Software para Melhoria de Gerenciamento de Projetos. In: CONFERÊNCIA INTERNACIONAL DE TECNOLOGIA DE *SOFTWARE*: QUALIDADE DE *SOFTWARE*, XII, 2001, Curitiba-PR. **Anais.** Curitiba-PR: Editora Universitária Champagnat, 2001, p. 68-83.

ZARRELLA, P. ***CASE* TOOL INTEGRATION AND STANDARDIZATION.** Disponível em: <http://www.sei.cmu.edu/publications/documents/90.reports/90.tr.014.html> Acesso em: 23 jun. 2003.

ANEXO I – FERRAMENTA DIMANAGER

A Figura 1 descreve a ferramenta DIMANAGER disponibilizando ao usuário informações para que o mesmo possa entender o funcionamento da DIMANAGER.

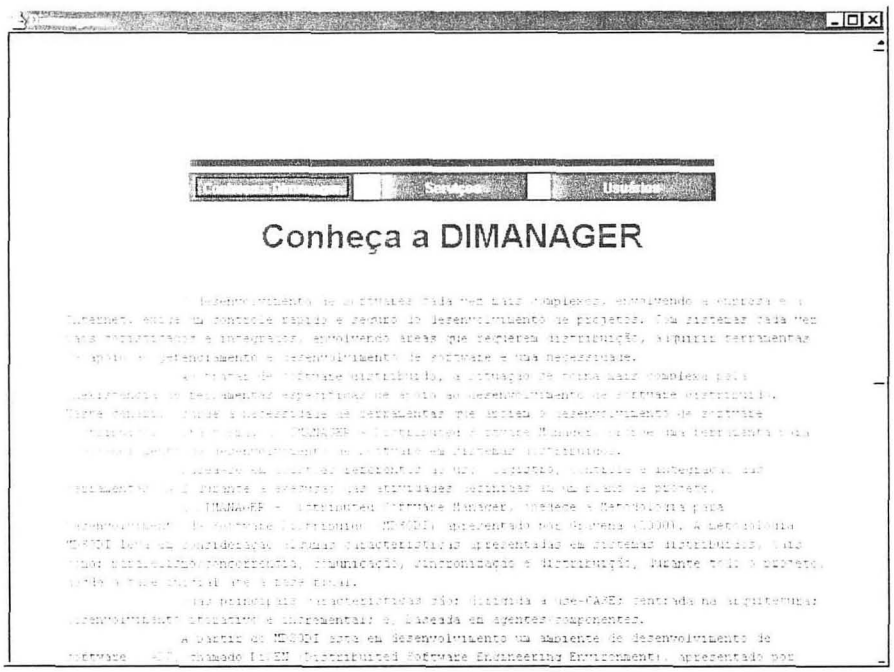

FIGURA 1 – CONHEÇA A DIMANAGER

A Figura 2 disponibiliza o cadastramento de novos usuários bem como a alteração de senhas já existentes.

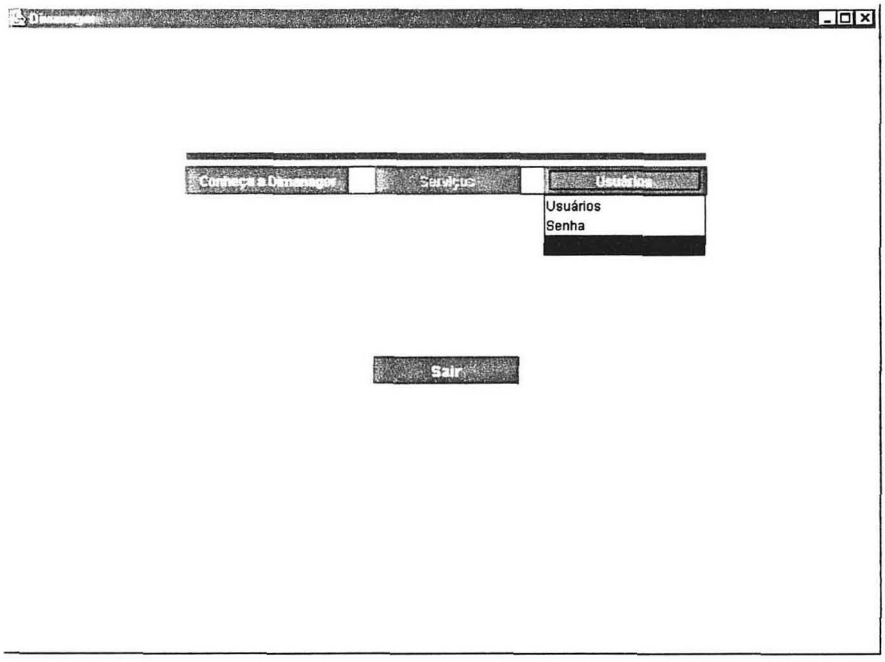

FIGURA 2 – OPÇÃO: USUÁRIOS

A Figura 3 disponibiliza o cadastramento de novos usuários bem como a alteração de senhas já existentes.

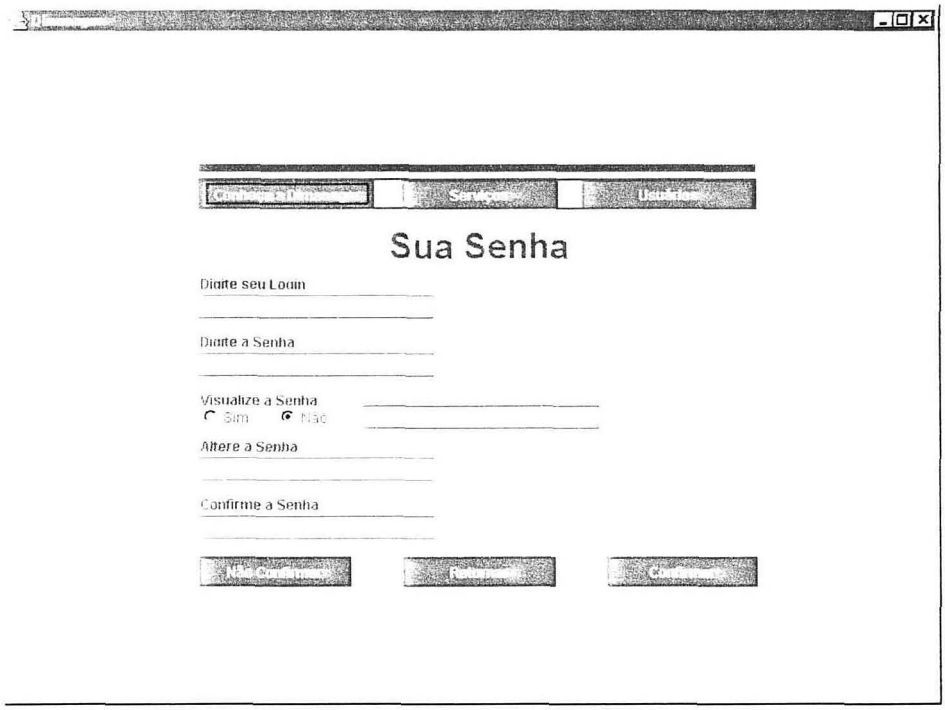

FIGURA 3 – SUA SENHA

A Figura 4 apresenta a identificação dos estados e a descrição dos mesmos, que podem ser: em andamento, parado, suspenso, em planejamento e encerrado.

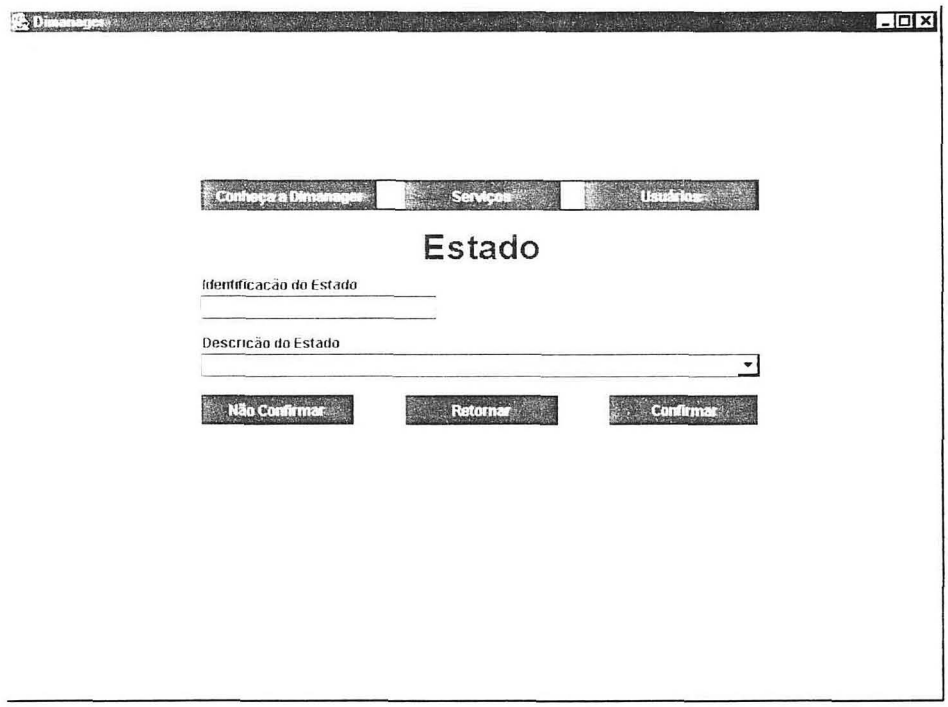

FIGURA 4 - ESTADO

A Figura 5 disponibiliza ao gerente de projeto cadastrar os problemas que podem ocorrer, assim como os ocorridos no desenvolvimento do projeto.

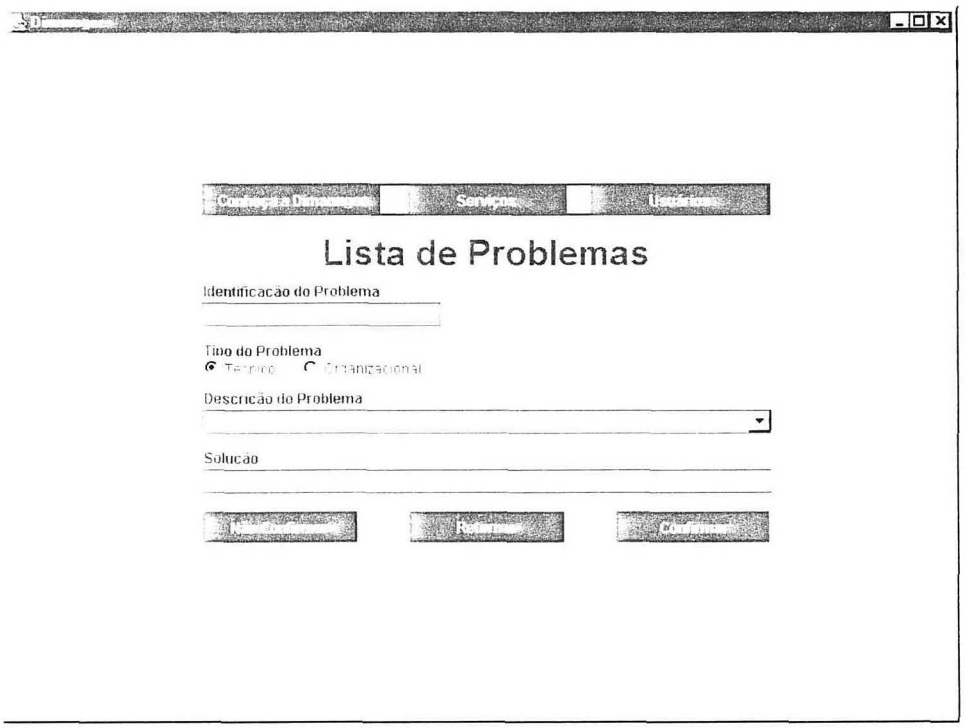

FIGURA 5 – LISTA DE PROBLEMAS

Os grupos de trabalho devem ser cadastrados através da Figura 6.

FIGURA 6 - GRUPO

O cronograma pode ser acompanhado de forma detalhada, mostrando ao gerente de projeto a data de início e fim, assim como o estado e o resultado esperado em cada atividade, conforme Figura 7.

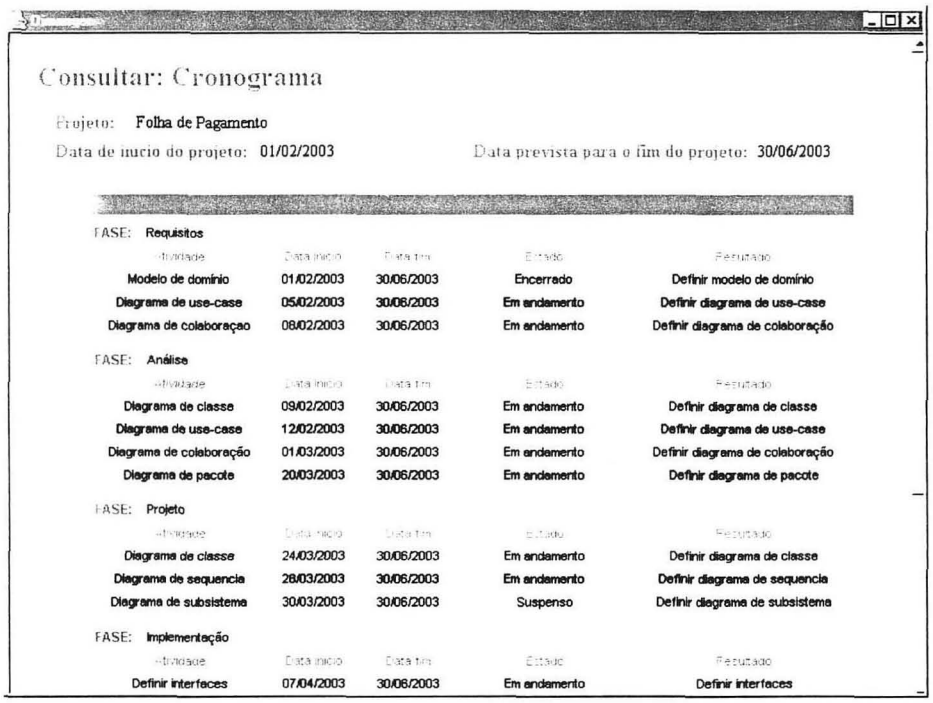

FIGURA 7 – ACOMPANHAR CRONOGRAMA – DETALHADO

O gerente de projeto pode acompanhar o cronograma através de uma fase específica, contendo as seguintes informações: data de início e fim, assim como o estado e o resultado esperado em cada atividade, conforme Figura 8.

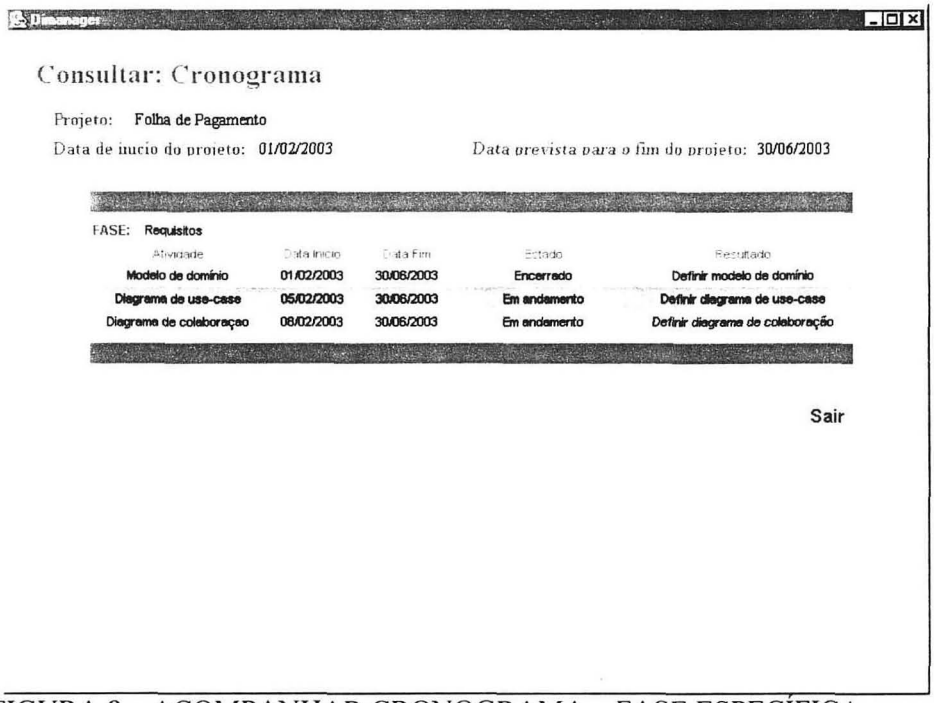

FIGURA 8 – ACOMPANHAR CRONOGRAMA – FASE ESPECÍFICA

Os grupos de trabalho podem ser acompanhados através da Figura 9.

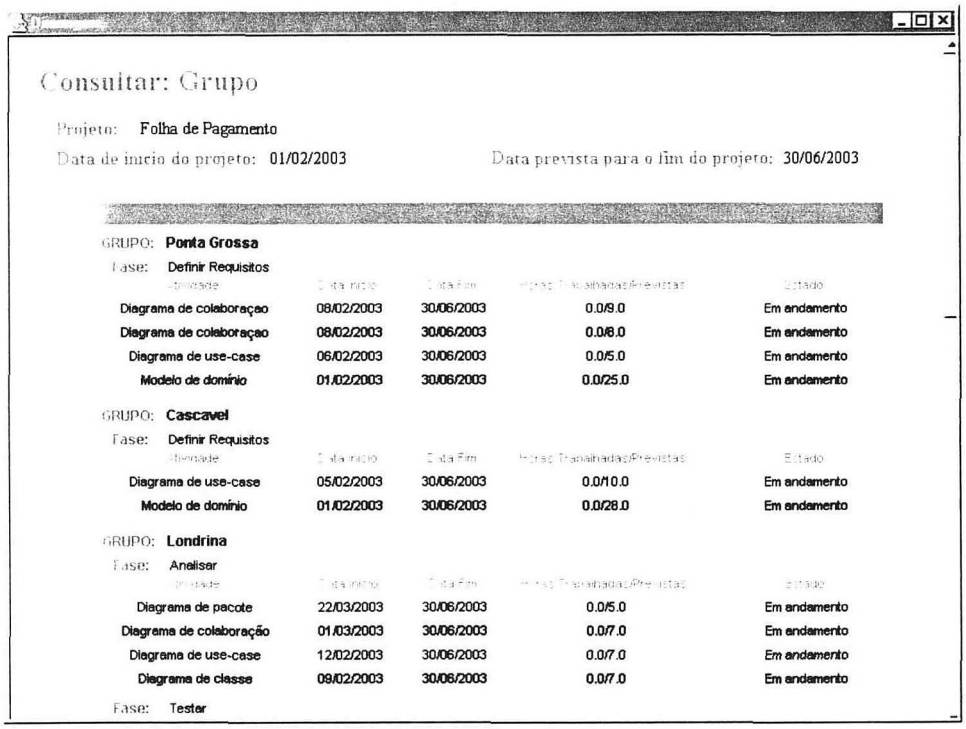

FIGURA 9 – ACOMPANHAR GRUPO – DETALHADO

Para verificar o desempenho dos participantes, o gerente de projeto pode conferir suas atividades através da Figura 10.

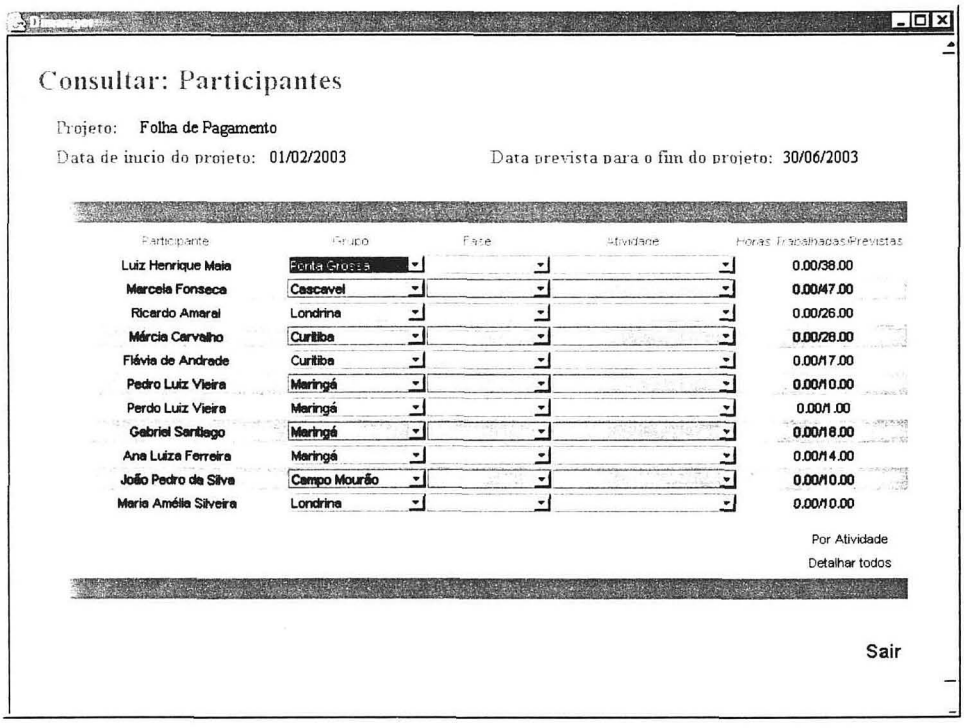

FIGURA 10 – ACOMPANHAR PARTICIPANTE – GERAL

O gerente de projeto pode acompanhar as atividades de cada participante, de forma detalhada, conforme apresenta a Figura 11.

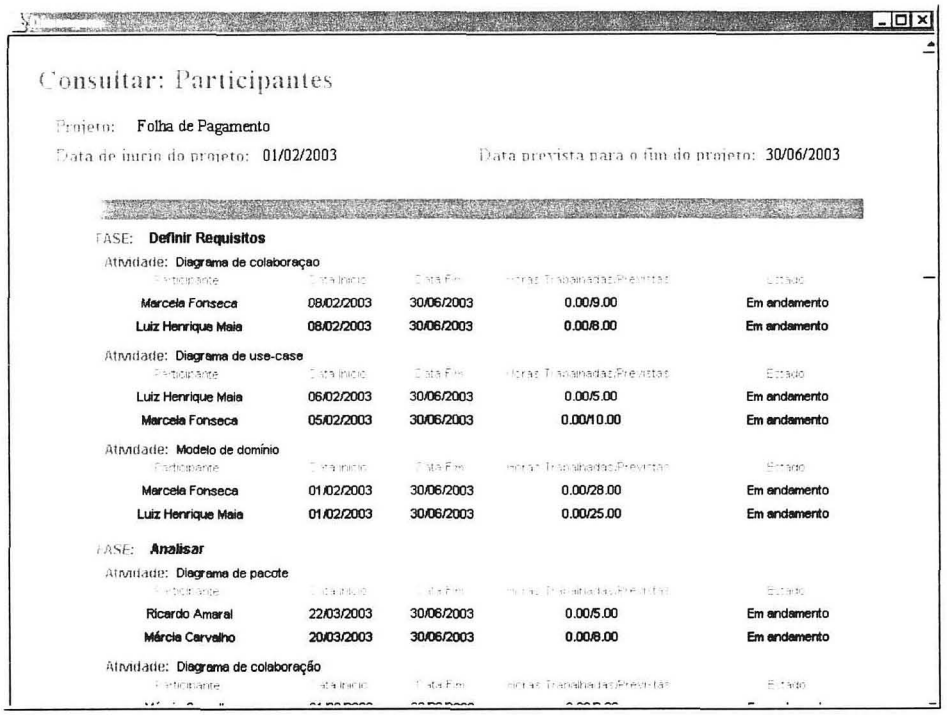

FIGURA 11 – ACOMPANHAR PARTICIPANTE – DETALHADO

Os problemas ocorridos no desenvolvimento do projeto, são apresentados na Figura 12.

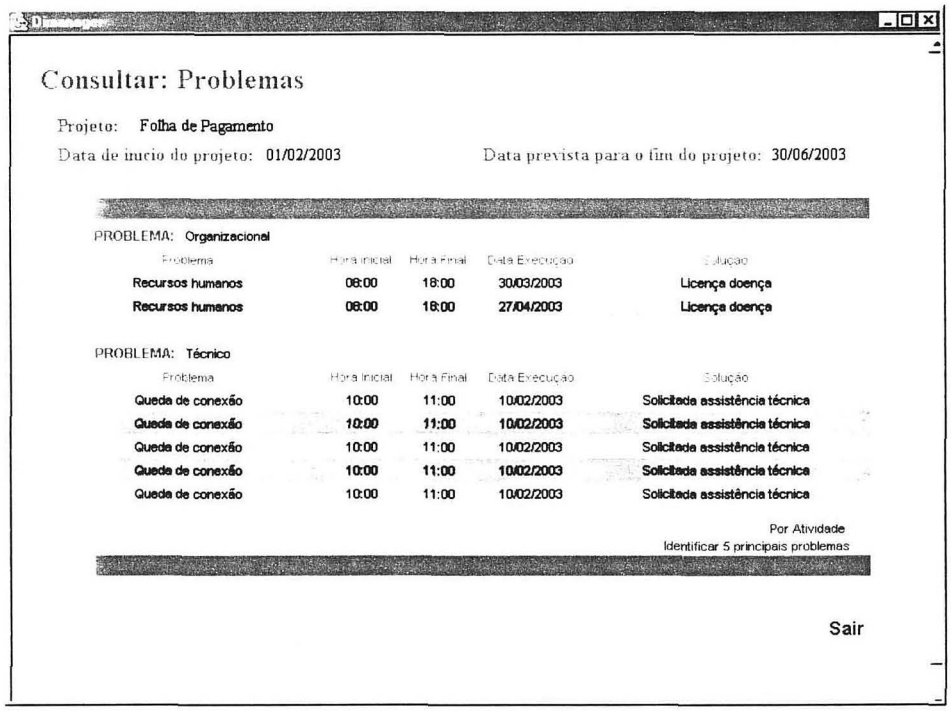

FIGURA 12 – ACOMPANHAR PROBLEMAS – GERAL (PROBLEMAS TÉCNICOS E ORGANIZACIONAIS)

O gerente de projeto pode verificar os problemas técnicos ocorridos através Figura 13.

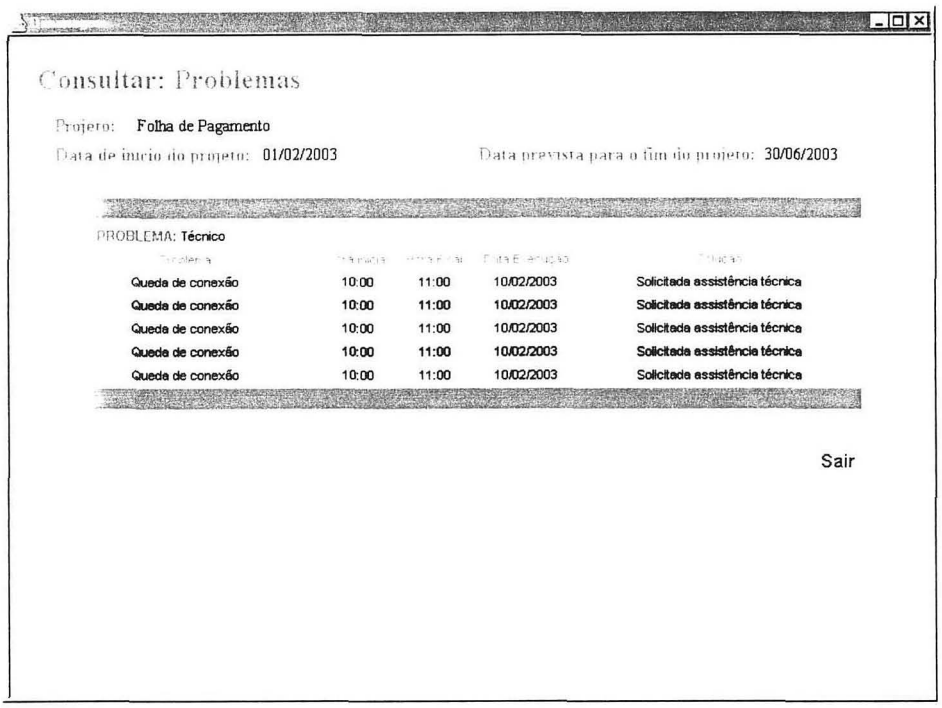

FIGURA 13 – ACOMPANHAR PROBLEMAS – PROBLEMAS TÉCNICOS

O gerente de projeto pode verificar os problemas organizacionais ocorridos no desenvolvimento do projeto através Figura 14.

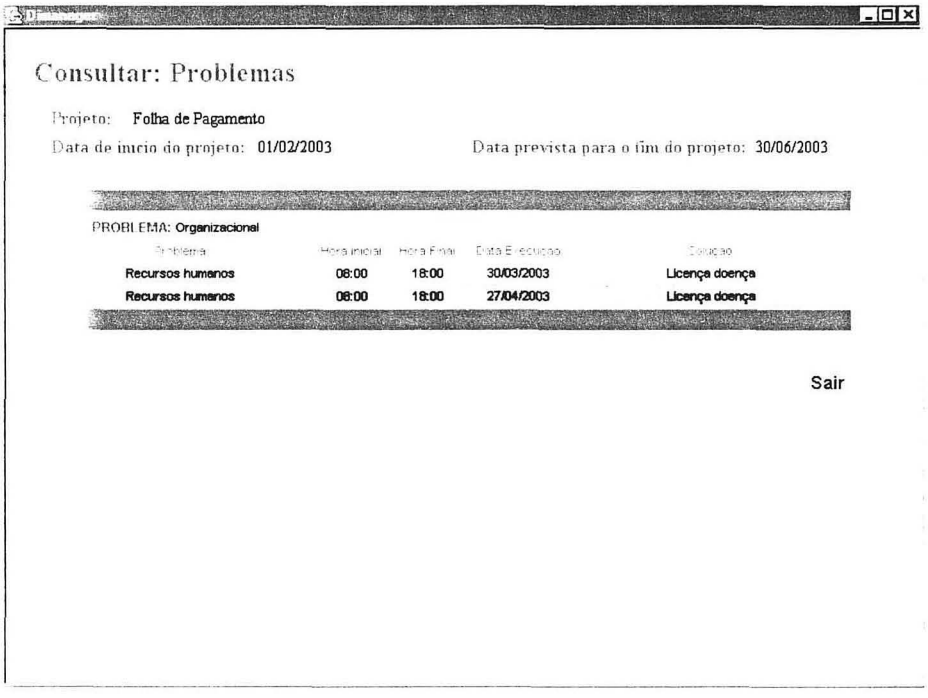

FIGURA 14 – ACOMPANHAR PROBLEMAS – PROBLEMAS ORGANIZACIONAIS

Os problemas ocorridos em cada fase do projeto podem ser acompanhados na Figura 15.

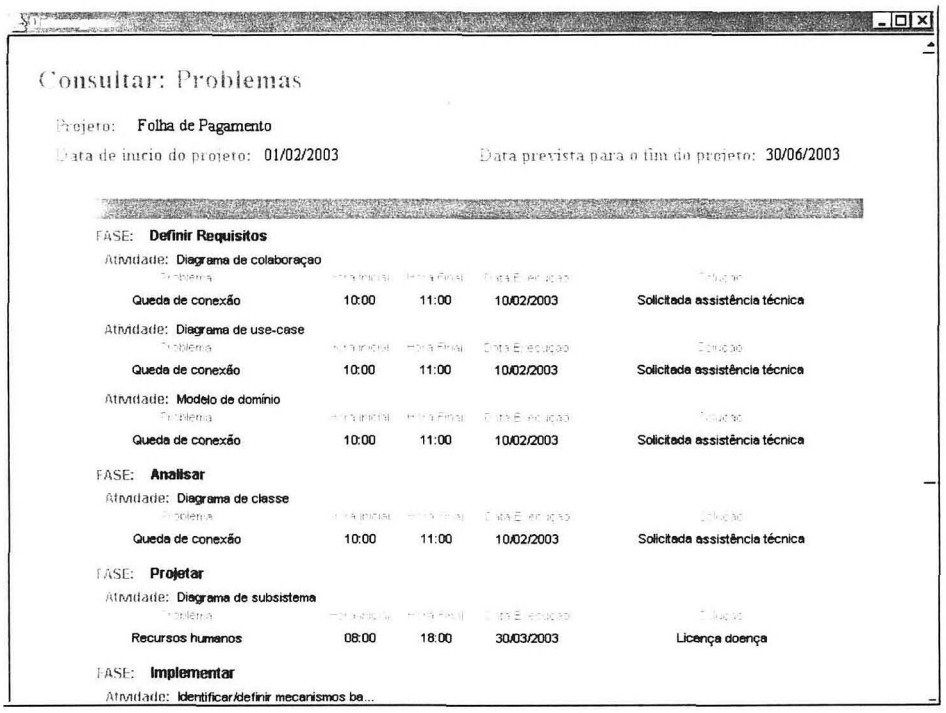

FIGURA 15 – ACOMPANHAR PROBLEMAS – POR FASE

A ferramenta DIMANAGER disponibiliza também os principais problemas ocorridos no desenvolvimento do projeto, conforme Figura 16.

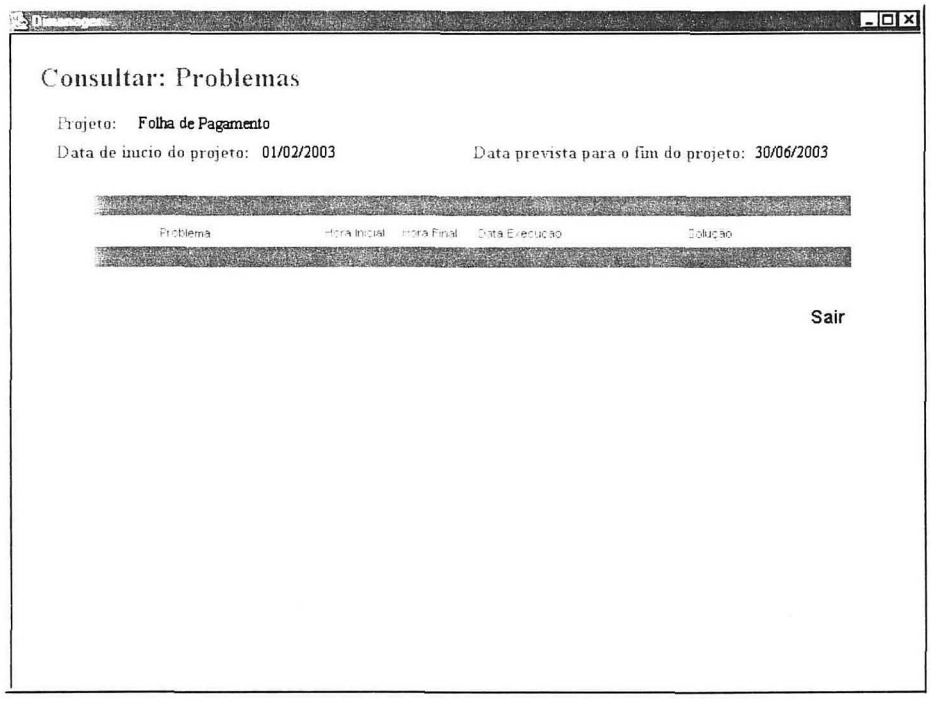

FIGURA 16 – ACOMPANHAR PROBLEMAS – PRINCIPAIS PROBLEMAS

ANEXO II – CONHEÇA A DIMANAGER

O desenvolvimento de *softwares* cada vez mais complexos, envolvendo a empresa e a Internet, exige um controle rápido e seguro do desenvolvimento de projetos. Com sistemas cada vez mais sofisticados e integrados, envolvendo áreas que requerem distribuição, adquirir ferramentas de apoio ao gerenciamento e desenvolvimento de *software* é uma necessidade.

Ao tratar de *software* distribuído, a situação se torna mais complexa pela inexistência de ferramentas específicas de apoio ao desenvolvimento de *software* distribuído. Neste cenário, surge a necessidade de ferramentas que apoiem o desenvolvimento de *software* distribuído. Desta forma, a DIMANAGER - *Distributed Software Manager,* propõe uma ferramenta para o gerenciamento de desenvolvimento de *software* em sistemas distribuídos. Basea-se em questões referentes ao uso, registro, controle e integração das ferramentas *CASE* durante a execução das atividades definidas em um plano de projeto.

A DIMANAGER - *Distributed Software Manager*, oferece apoio à Metodologia para Desenvolvimento de *Software* Distribuído (MDSODI) apresentado por Gravena (2000). A metodologia MDSODI leva em consideração algumas características apresentadas em sistemas distribuídos, tais como: paralelismo/concorrência, comunicação, sincronização e distribuição, durante todo o projeto, desde a fase inicial até a fase final.

Suas principais características são: dirigida a *use-case*; centrada na arquitetura; desenvolvimento iterativo e incremental; e, baseada em agentes/componentes.

A partir do MDSODI está em desenvolvimento um ambiente de desenvolvimento de *software* – ADS, chamado *DiSEN* (*Distribuited Software Engineering Environment*), apresentado por Pascutti (2002)

Os requisitos funcionais da DIMANAGER são: planejar e acompanhar o projeto.

Algumas funcionalidades, apresentadas no trabalho de Gomes et al. (2001), tais como: um calendário de atividades e o cadastro de usuários, capaz de apoiar e orientar

o levantamento e a análise de dados incorporados ao ambiente de *software* instanciados pela Estação Taba foram adaptadas ao DIMANAGER, pois fazem parte de um bom gerenciamento das atividades desenvolvidas.

Além destas funcionalidades outras informações foram consideradas para a definição do planejamento do projeto, tais como: identificação das atividades dentro do projeto; definição dos participantes; elaboração do cronograma de atividades.

Dois elementos considerados importantes não são tratados pela DIMANAGER, pois merecem um estudo à parte e mais profundo. Quanto as métricas, está sendo tratada pela ferramenta na relação horas/homem. Para a execução de um projeto os custos financeiros envolvem tanto os recursos humanos quanto os materiais. Estes dois elementos: métricas e custos, devem fazer parte do planejamento e acompanhamento do gerente, mas não estarão sendo controlados pela DIMANAGER. Este controle fica como sugestão para implementação em trabalhos futuros.

ANEXO III – PROBLEMAS TÉCNICOS

a. Queda de conexão

- Servidor fora do ar
- HUB com problema
- Porta do HUB queimada
- Fio corrompido
- Estação com problema de hardware
- Mau contato

b. Queda de conexão com a Internet

- São 3 (três) problemas que podem estar influenciando essa constante queda de conexão:

1. O protocolo **TCP/IP** é o responsável pela sua conexão, bem como pelo tráfego de todas as informações, seja download, upload, chat, navegação, ftp ou qualquer outro serviço de internet. Se por algum motivo o protocolo estiver com problemas a conexão é interrompida constantemente, ou a conexão não é confirmada, podendo inclusive apresentar constantes erros de senha. Esse protocolo **TCP/IP** pode ficar corrompido ou danificado, se por exemplo, em algum momento durante a sua navegação a linha "caiu" ou ainda se você desligou o micro quando o arquivo estava em utilização. Ou até mesmo quando seu micro "trava" por algum motivo e você precisa desligá-lo de forma a não sair normalmente do Windows, o **TCP/IP** também pode ter ficado danificado. É importante salientar que você não precisa estar necessariamente na internet para que o arquivo fique danificado.A melhor forma de corrigir este problema é a remoção e a re-instalação desse protocolo, para isso basta que você possua os discos ou o CD de

instalação do Windows 95 ou 98, dependendo do seu Sistema Operacional.

2. Driver do **Modem**. Todos os **modems**, independente das marcas possuem um arquivo que habilita o Windows 95 ou 98 a reconhecê-lo. Este arquivo trabalha em conjunto com o **TCP/IP**, afinal de contas é ele que encaminha a solicitação de conexão, bem como o início das transmissões. Se por algum motivo este **driver**, estiver danificado ele também não conseguirá efetuar as conexões de forma correta, implicando também na queda de conexão. A solução também implica em remover este **driver** e instalá-lo novamente.

3. **Linha Telefônica**. Caso você esteja tendo problemas em sua **linha telefônica** sua ligação pode ser interrompida a qualquer instante. **IMPORTANTE:** O problema de linha telefônica, não significa, necessariamente que o uso para ligações normais da sua **linha telefônica** esteja com problemas também. Quando você efetua uma ligação para a estação 814, ela não chega direto, ela passa por estações intermediárias, que podem estar com problemas. Neste caso somente a intervenção da sua **Companhia Telefônica** pode apresentar um parecer técnico.

c. Estação desliga sozinha

- Cooller
- Fonte

d. Mensagens de erros

- "Este programa efetuou uma operação ilegal..."

1. Reinicie o seu computador e caso o problema persista será necessário chamar o técnico de informática
2. Neste caso poderá ser necessário efetuar a reinstalação do programa gerador de mensagem

- O repositório o banco de dados pode estar fora do ar para manutenção

ANEXO IV - QUESTIONÁRIO

Universidade Estadual de Maringá
CENTRO DE TECNOLOGIA
Departamento de Informática

Mestrado em Informática

Apresentação da DIMANAGER

Data:

Nome da Instituição:

Nome do Profissional: _____

Função: _____

Área de atuação: _____

1. A apresentação permitiu visualizar a ferramenta e sua utilidade ?
   ( ) Sim ( ) Não Justifique _____

   _____

2. A ferramenta apresentada tem utilidade para o gerenciamento de projetos :
   ( ) Sim ( ) Não Justifique _____

   _____

3. Você considera que o manuseio da ferramenta facilita o trabalho do gerenciamento ?
   ( ) Sim ( ) Não Justifique _____

   _____

4. A ferramenta apresenta utilidade para projetos desenvolvidos em locais dispersos geograficamente:
   ( ) Sim ( ) Não Justifique _____

   _____

5. Sugestões/opiniões/criticas

_____

_____

_____

_____