

Universidade Federal da Paraíba - UFPB
Centro de Ciências e Tecnologia - CCT
Coordenação de Pós-Graduação em Informática - COPIN

UM MOLDE PARA O PROCESSO DE PRODUÇÃO DE SOFTWARE COMERCIAL.

ILUSTRAÇÕES COM SOFTWARE UNIX DE PRATELEIRA

Luiz Maurício Fraga Martins

Campina Grande - PB

Dezembro - 1993

UM MOLDE PARA O PROCESSO DE PRODUÇÃO DE SOFTWARE COMERCIAL.
ILUSTRAÇÕES COM SOFTWARE UNIX DE PRATELEIRA

Luiz Maurício Fraga Martins

Dissertação apresentada ao curso de
MESTRADO EM INFORMÁTICA da
Universidade Federal da Paraíba, em
cumprimento às exigências para
obtenção do grau de mestre.

Área de Concentração : Ciência da Computação

José Antão Beltrão Moura

Orientador

Campina Grande - PB

Dezembro - 1993



M386m

Martins, Luiz Maurício Fraga.

Um molde para o processo de produção de software comercial : ilustrações com software UNIX de prateleira / Luiz Maurício Fraga Martins. - Campina Grande, 1993. 191 f.

Dissertação (Mestrado em Informática) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1993. "Orientação : Prof. José Antão Beltrão Moura". Referências.

1. Engenharia de Software. 2. Produção de Software Comercial. 3. Software UNIX de Prateleira. 4. Dissertação - Informática. I. Moura, José Antão Beltrão. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 004.41(043)

Um Molde para o Processo de Produção de Software Comercial.
Ilustrações com Software Unix de Prateleira

LUIZ MAURÍCIO FRAGA MARTINS

Dissertação Aprovada em : 22/12/93

Banca Examinadora :


JOSÉ ANTÃO BELTRÃO MOURA, PhD
Presidente


MARIA DE FÁTIMA CAMELO, M. Sc


EDUARDO MOREIRA COSTA, PhD

Campina Grande
Dezembro - 1993

Agradecimentos

O autor gostaria de agradecer aquelas pessoas que juntas forneceram as condições para a realização deste trabalho :

Ao meu orientador, pela atenção a mim dedicada durante estes dois anos de trabalhos, pela humildade e sabedoria com que transmitiu seus conhecimentos e vasta experiência sobre o tema.

Aos meus pais e irmãos, que sempre foram os meus guias e me fazem perceber cada vez mais o quanto são importantes na minha vida e quanto os amo.

A minha esposa Cristiana e ao meu filho Luan, pelo amor que tenho a vocês e pela motivação de vida que me proporcionam.

A Infocon Tecnologia e funcionários, em especial a Celso, Reginaldo e Nivaldo, pelo apoio proporcionado nesta jornada.

A todos os colegas do mestrado, funcionários e professores do DSC.

A CAPES, pelo apoio financeiro prestado.

Aos meus amigos-irmãos Fernando, Vicente e Lauro, pela mensagem de vida que cada um de vocês me transmitiu.

A todos,

meu muito obrigado.

Resumo

A utilização de modelos de processos de produção de software norteia os produtores de software no planejamento e controle das atividades integrantes do ciclo de vida do software. A literatura propõe diversos modelos de processos com este propósito. No entanto, a prática demonstra que os modelos de processos existentes não condizem com a realidade de produção de software para o mercado (software comercial), principalmente pela excessiva concentração em aspectos de desenvolvimento.

Este trabalho propõe um "molde" genérico mais realista e abrangente para a produção e disponibilização de software comercial, a partir do qual modelos específicos possam ser construídos e validados para casos particulares e realistas da indústria. O molde estende a abordagem tradicional dos modelos de processos, identificando fatores críticos para o sucesso de um software comercial, tais como marketing, comercialização e suporte técnico, que interferem diretamente nos prazos e orçamentos de projetos de software.

A partir da definição do molde, busca-se instanciar sua utilização na produção de software de prateleira para o segmento Unix. Este estudo se concentra nas fases de preparação e disponibilização de software para o mercado, onde foram verificados procedimentos técnicos e gerenciais imprescindíveis na composição de um modelo realista para a produção e disponibilização de software comercial.

A construção do molde e sua instanciação resultam na identificação e exemplificação de procedimentos considerados críticos para o sucesso não apenas da produção como também da disponibilização de software para o mercado.

O trabalho pode ter continuidade natural, instanciando-se o molde para software destinado a outros segmentos de mercado (que não apenas Unix).

Abstract

Production Process Models are part of the planning and control phases of a software life cycle. Current literature proposes several processes models for this purpose. Practice, however, shows that the existing process models do not conform with the production scenario of the software industry (commercial software) due to their great focus on development issues.

This work proposes a more realistic and generic template for modelling the production and delivery of commercial software upon which other models can be built and validated for specific and real industry cases. This template extends the traditional approach of process models identifying critical aspects for the success of a commercial software such as marketing, sales, distribution and technical support, which play an important role on software project budgets and schedules.

After defining the template, a case study of its utilization in the software production within the Unix segment is carried out. This study concentrates on the preparation and delivery phases of commercial software, where technical and management procedures were found to be vital for the composition of a realistic model for the production and delivery of commercial software.

The definition of the template and its case study allow the identification and illustration of procedures considered critical for the success of both the production and delivery of commercial software.

The application of the template to other software market segments (besides Unix) is a natural pathway to further pursue this research.

Índice

1. Introdução, 1
 - 1.1 Um Breve Histórico, 1
 - 1.2 O Processo de Produção de Software, 3
 - 1.3 Necessidades de Modelos de Processos, 4
 - 1.4 Objetivos da Dissertação, 6
 - 1.5 Contribuições da Dissertação, 7
 - 1.6 Organização da Dissertação, 8

- Parte I - Modelos de Processos de Produção de Software, 12**

2. A Modelagem do Processo de Produção, 13
 - 2.1 Modelos Clássicos, 13
 - 2.1.1 Modelo "Cascata", 14
 - 2.1.2 Modelos Evolutivos, 17
 - 2.1.3 O Modelo Espiral, 19
 - 2.2 Comentários sobre os Modelos Clássicos, 21
 - 2.3 Novos Modelos, 24
 - 2.3.1 O Modelo CDPM, 24
 - 2.3.2 O Modelo COSMOS, 27
 - 2.4 Um Molde Genérico Mais Realista, 29

3. Um Molde Mais Realista, 34
 - 3.1 Motivação, 34
 - 3.2 Princípios Adotados para a Definição do Molde, 36
 - 3.3 Um Molde mais Realista, 39
 - 3.4 Fases e suas Atividades, 42
 - 3.5 Atividades Polifásicas, 46
 - 3.5.1 Investimentos, 46
 - 3.5.2 Documentação, 48
 - 3.5.3 Marketing, 50
 - 3.5.4 Controle de Qualidade de Software, 52

Parte II - Comentários Sobre Preparação e Disponibilização de Software, 56

- 4. Empacotamento, 57
 - 4.1 Instalação, 58
 - 4.2 Proteção ao Software, 63
 - 4.2.1 Aspectos Legais, 64
 - 4.2.1.1 Segredos de Negócio, 64
 - 4.2.1.2 Direitos Autorais, 67
 - 4.2.1.3 Patentes, 70
 - 4.2.2 Aspectos Técnicos, 72
 - 4.2.2.1 Esquema Infocon, 74
 - 4.2.2.2 Esquema Lotus, 77
 - 4.3 Reprodução, 80
- 5. Testes de Aceitação, 83
 - 5.1 Testes Alfa, 84
 - 5.2 Testes Beta, 88
- 6. Vendas & Distribuição, 94
 - 6.1 Vendas : Necessidades, Dependências e Importância no Ciclo de Vida do Software, 95
 - 6.2 Alternativas de Venda, 97
 - 6.3 Definição de Preço, 98
 - 6.4 Canais de Distribuição, 101
 - 6.5 Homologação de Software, 105
- 7. Suporte Técnico, 111
 - 7.1 Introdução, 112
 - 7.2 Uma Visão de Suporte, 113
 - 7.3 Ambiente Proposto, 115
 - 7.4 Perfil do Profissional de Suporte, 115
 - 7.5 Organização das Atividades de Suporte, 117
 - 7.5.1 Suporte Pré-Venda, 117
 - 7.5.2 Suporte Pós-Venda, 118
 - 7.6 Atendimento ao Usuário, 120
 - 7.7 Metodologia para Automação do Suporte, 121
 - 7.7.1 Descrição do Banco de Dados, 123
 - 7.7.2 Procedimentos, 125
 - 7.7.3 Análise dos Dados, 130
 - 7.8 Alguns Objetivos de Suporte, 135

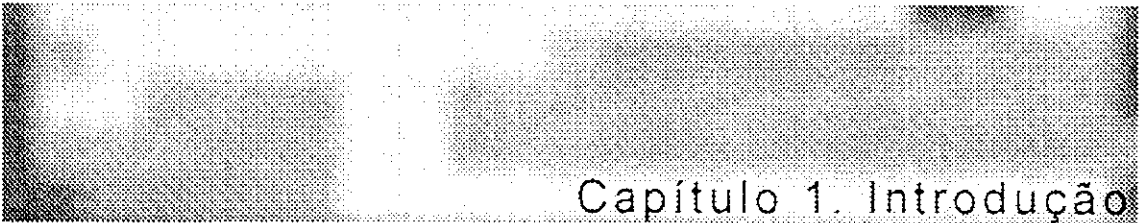
- 8. Manutenção, 137
 - 8.1 Introdução, 138
 - 8.2 Mudanças e suas consequências, 140
 - 8.3 Estruturando a Manutenção em Unix, 142
 - 8.3.1 Recebimento de Solicitações, 143
 - 8.3.2 Priorização de Solicitações, 145
 - 8.3.3 Seleção Funcional, 146
 - 8.3.4 Ativando o Processo de Mudanças, 150
 - 8.3.5 Liberação, 151
 - 8.3.6 Documentação, 151
 - 8.4 Gerência de Configuração de Software (GCS), 153
 - 8.4.1 Funções da GCS, 154
 - 8.4.2 Versionamento, 156
 - 8.4.2.1 Identificando Versões e Releases, 158
 - 8.4.3 Ferramentas de GCS, 159
 - 8.5 Portabilidade, 163
 - 8.5.1 O transporte, 164
- 9. Conclusões e Trabalhos Futuros, 168
 - 9.1 Sumário, 168
 - 9.2 Conclusões, 172
 - 9.3 Trabalhos Futuros, 175
- Apêndice A - Arquivos para Instalação de um Software, 178
- Apêndice B - Formulário Infocon de Licenciamento de Software, 181
- Apêndice C - Formulário Lotus de Licenciamento de Software, 182
- Apêndice D - Exemplo de Portabilidade de Código, 183
- Bibliografia, 186

Lista de Figuras

- 1.1 - Motivação para Modelos de Processos, 4
- 1.2 - Organização da Dissertação, 9
- 2.1 - Modelo Cascata, 15
- 2.2 - Modelo Evolutivo (prototipagem), 19
- 2.3 - O Modelo Espiral para Produção de Software, 20
- 2.4 - Modelo de Produção de Desenvolvimento Concorrente (CDPM), 25
- 2.5 - Ilustração de Segmentação de Mercado, 30
- 3.1 - Um Molde Realista para Produção de Software Comercial, 40
- 3.2 - Visão Simplificada das Fases do Molde, 43
- 4.1 - Visão Funcional do Instalador Ic da Infocon, 61
- 4.2 - Esquema de Proteção e Licenciamento da Infocon, 75
- 4.3 - Esquema de Proteção e Licenciamento da Lotus, 77
- 5.1 - Formulário Infocon para Testes Alfa, 86
- 5.2 - Formulário Infocon para Testes Beta, 91
- 6.1 - Ilustração dos Custos x Margens de Lucro em Função do Canal de Comercialização, 105
- 7.1 - O Suporte como Estratégia de Integração, 114
- 7.2 - Metodologia para Automação do Suporte, 122
- 7.3 - Gráfico de Pareto para Identificação dos Problemas Críticos, 132
- 8.1 - Fatores que Influenciam a Manutenção de Software, 139
- 8.2 - Organização da Estrutura de Manutenção de Software, 144
- 8.3 - Formulário de Votação para Evolução de Software, 148
- 8.4 - Exemplo de Arquivo de Documentação para Nova Release, 153
- 8.5 - Versões do Software e Seus Componentes, 157
- 8.6 - Versionamento Utilizando ScsS, 160
- 8.7 - Estrutura de Diretórios para Utilização do ScsS, 161
- 8.8 - Transporte de Software, 163
- 9.1 - Cooperação Academia-Indústria, 176

Lista de Tabelas

- 7.1 - Serviços Oferecidos pelo Suporte Pós-venda, 119
- 7.2 - Classificação de Problemas para Análise de Pareto, 132
- 8.1 - Tabela de Prioridades de Manutenção, 145



Capítulo 1. Introdução

1.1 Um Breve Histórico

Depois de serem realizados grandes avanços tecnológicos na área de hardware, principalmente em termos de minimização de custos e aumento da capacidade de processamento e armazenamento, os interesses mundiais se voltam para o desenvolvimento do software. A justificativa destes interesses tem origem na disseminação da cultura de microinformática pelas organizações e pelo valor estratégico que os sistemas de informações adquiriram nos últimos tempos.

Como consequência, houve um aumento significativo na demanda por novos produtos de software, emergindo um mercado expressivo que em 1991 movimentou cerca de cento e cinquenta e um bilhões de dólares, segundo a International Data Corporation. Leve-se ainda em consideração que o potencial existente atualmente para desenvolvimento de software não consegue atender toda a demanda fazendo com que usuários americanos,

por exemplo, aguardem de dois a três anos em média, para que suas solicitações sejam atendidas [Boeh88].

A importância do software na sociedade moderna demandou procedimentos para obtenção de software de qualidade, e na busca de mecanismos que melhor direcionassem o processo de produção, surgiu a Engenharia de Software. Esta nova área de conhecimento preconiza o desenvolvimento e aperfeiçoamento sistemático de softwares de alta complexidade como um trabalho de engenharia, onde o processo de produção de software seria suportado por teorias, metodologias, técnicas, ferramentas e gerenciamento [Ghez91].

Apesar dos avanços alcançados pela Engenharia de Software na década passada, principalmente com o surgimento de linguagens de quarta geração e ferramentas CASE¹ voltadas para a modelagem e geração de código intencionando melhorar a produtividade de desenvolvimento do software, não podemos afirmar categoricamente que a produção de software tenha se tornado um processo de Engenharia. A realidade atual mostra, infelizmente, que a produção de software não atingiu um grau de padronização e formalismo necessário para se atender fielmente e de imediato a suas especificações.

As consequências da imaturidade da Engenharia de Software são visíveis, seja nos altos custos de manutenção de software, em torno de sessenta por cento [Long90] dos custos totais do software, seja na dificuldade de se atender a prazos e orçamentos planejados. Estes fatores

¹O termo CASE ou Computer Aided Software Engineering é referenciado para representar o conjunto de ferramentas automatizadas via computador, utilizadas para auxiliar em etapas do desenvolvimento de software. Estas ferramentas podem estar interligadas no sentido de abranger várias etapas do processo de produção e metodologias de desenvolvimento, conhecidas neste caso como I-Case (*Integrated Case*).

combinados originaram a "crise do software", considerada um fator determinante para a criação e evolução da Engenharia de Software.

1.2 O Processo de Produção

O processo de produção de software, segundo Humphrey[Hump89], representa o conjunto de atividades, métodos e práticas utilizados na produção e evolução do software.

As últimas décadas marcaram o crescente aumento da complexidade dos aplicativos de computadores (software), à medida em que os aplicativos passaram a oferecer recursos mais sofisticados, visando a satisfação e produtividade dos seus usuários.

A produção de software passou então a se tornar uma tarefa complexa, sendo identificadas diversas alternativas de execução das tarefas com alto grau de dependência do perfil da organização produtora, do tipo de produto, do seu mercado alvo, das ferramentas e metodologias disponíveis e até mesmo do sistema operacional.

Afim de ordenar as atividades, métodos e práticas envolvidas na produção de software, surgiram os modelos de processos. Estes modelos são utilizados para guiar os trabalhos dos Engenheiros de software e balisar as atividades de gerência de software quanto ao planejamento, organização e controle de todo o processo de produção [Ghez91].

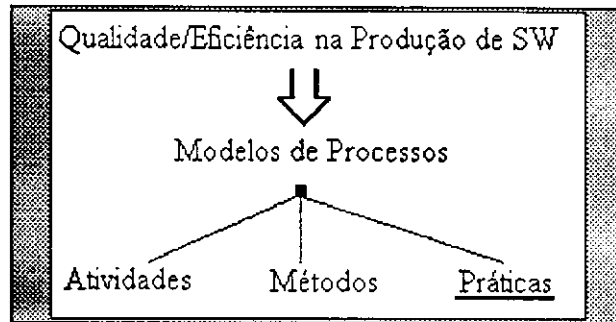


Figura 1.1 - Motivação para Modelos de Processos

Sobre modelos de processos da produção de software, Krasner [Kras92] possui uma visão bastante lúcida e significativa para este trabalho:

"Técnica utilizada para definir e analisar os aspectos significantes de desenvolvimento e evolução de software, criada para facilitar o entendimento humano, para dar suporte ao gerenciamento do processo e continuamente impor melhorias, facilitando a sua automação"

Este trabalho se dedica especificamente ao estudo de modelos de processos, que auxiliem efetivamente na produção de software comercial de alta qualidade, considerando suas particularidades em função das práticas verificadas no mercado, conforme ilustrado na figura 1.1.

1.3 Necessidades de Modelos de Processos

A produção de software exige competência, dinamismo, preço e sobretudo qualidade para alcançar com sucesso o mercado exterior e garantir a auto sustentação no mercado interno, frente a concorrência internacional já instalada.

De fato, a concorrência pelo mercado de software se tornou muito acirrada, fazendo com que produtores se especializassem em determinados segmentos de mercado para poderem fornecer software de melhor qualidade. Esta estratégia permite aos produtores acompanhar melhor as aceleradas mudanças tecnológicas, seguir os movimentos dos seus concorrentes e atender às necessidades específicas do seu segmento.

Esta afirmação é corroborada pela desistência da NEXT em ser fornecedor de soluções completas (hardware e software) para se especializar na oferta de software (o NextStep) e ainda mais pela observação de criação de novas empresas com vastos recursos (p.ex., a Taligent uma associação entre as fortes IBM e Apple para desenvolver um sistema operacional orientado a objetos) e da acirrada concorrência do New Technology (NT) da Microsoft contra o Unix da Novell-X/Open.

Modelos possuem o dom, quando bem sintonizados com às necessidades reais, de fornecer indicadores para investimentos e recursos necessários para a execução de um projeto de software, além de disciplinar as ações com mecanismos de controle e avaliação do processo.

No mercado competitivo de software, onde por vezes existem dezenas de concorrentes para um mesmo produto (p.ex. : editores de texto, bancos de dados), a devida organização do processo de produção pode ser o diferencial, resultando na melhor distribuição de recursos e conseqüentemente os investimentos podem ser redirecionados para controle de qualidade do produto, marketing, ou suporte técnico.

Por outro lado, os modelos de processos são instrumentos utilizados pela academia na formação dos futuros profissionais. A sua utilização no ensino facilita a visualização das atividades envolvidas no processo de

produção de software, familiarizando o alunado com a realidade que os espera no mercado de trabalho. Portanto, a atualização dos modelos de processos disseminados nas salas de aula se faz necessário para aproximar a formação do alunado das práticas de mercado.

1.4. Objetivos da Dissertação

Complacente com o atual estágio de especialização da produção de software em função da segmentação de mercado, de sua complexidade inerente e da diversidade de alternativas de execução, verifica-se que não é plausível tentar elaborar um modelo único de produção de software universal e representativo nos diversos segmentos de mercado. Inclusive, **[Hump89]** e **[Ghez91]** criticam a generalidade dos modelos de produção de software existentes, por representarem fluxos de trabalho genéricos, dificultando o refinamento necessário para guiar o trabalho dos seus produtores.

Assim sendo, este trabalho caracteriza um cenário de produção de software comercial, propondo um "molde" mais adequado à realidade do processo de produção vivenciado na prática. Como será visto no decorrer do trabalho, o molde pode ser instanciado para os diversos segmentos de mercado, afim de possibilitar o desenvolvimento de manuais ou guias práticos que auxiliem efetivamente no processo de produção de software destes segmentos de mercado.

O material resultante, acoplado ao material bibliográfico disponível sobre Engenharia de Software, pretende servir como ponto de referência para amenizar nossas atuais necessidades de desenvolver produtos de software de qualidade para se defrontar com o exigente mercado internacional.

Evidentemente que pela própria natureza e dimensão do trabalho seria ambicioso dizer que serão propostas fórmulas "mágicas" de como se deve proceder para se obter sucesso na produção e disponibilização de software comercial, ou seja, como alcançar a produção de software competitivo, de qualidade, a um preço atraente e em tempo hábil.

Assim sendo, são apresentados e discutidos de maneira organizada, os princípios e a estrutura de um molde mais realista para o processo de produção de software comercial. Verifica-se também a aplicabilidade do molde no processo de preparação e disponibilização de software Unix de prateleira para o mercado. Este processo é ilustrado por conhecimentos e experiências de um produtor local (Infocon), metodologias, técnicas e ferramentas consideradas determinantes para o sucesso de um software no mercado.

Espera-se também indicar passos em falso a serem evitados por produtores nacionais que almejam colocar software no mercado, ou para aqueles que já o fazem, fornecendo subsídios técnico-gerenciais que são utilizados de fato em projetos reais, suportando a tarefa de planejamento e controle do processo de produção de software.

1.5 Contribuição da Dissertação

Este trabalho contribui para a Engenharia de Software, do ponto de vista prático, já que estabelece diretrizes para balizar o investimento de recursos para se atingir um nível aceitável de qualidade com o processo de produção e distribuição de software para o segmento comercial. Mais ainda, ele complementa e estende textos tradicionais sobre Engenharia de Software, identificando aspectos críticos para a produção de software, ainda não contemplados pelos modelos de processos existentes.

A importância dos trabalhos nesta linha é realçada pelo esforço nacional, apoiado pelo CNPQ, de tornar o país exportador expressivo de software até o final do século. Neste sentido, está em curso um programa de desenvolvimento estratégico na área de informática, fomentada pelo CNPQ em conjunto com as Nações Unidas, que pretende beneficiar desde o ensino até o segmento produtivo de software, criando a infra-estrutura necessária, estreitando as relações entre empresas e instituições acadêmicas de pesquisa, e implantando mecanismos de fomento à produção de software para o mercado mundial. O programa, denominado SoftEX 2000, almeja colocar o desenvolvimento brasileiro de software num patamar de 1% a 2% da produção mundial, tornando o software um dos principais produtos de exportação do Brasil até o ano 2000.

Espera-se que o presente trabalho seja uma contribuição da academia para este programa, orientando um pouco os esforços de produção de software, com base em experiências passadas, e alertando para as necessidades de procedimentos para lançamento e evolução de produtos de software no mercado.

1.6 Organização da Dissertação

Para melhor visualização e compreensão do seu conteúdo, o documento está dividido em duas partes, distribuídas da seguinte maneira:

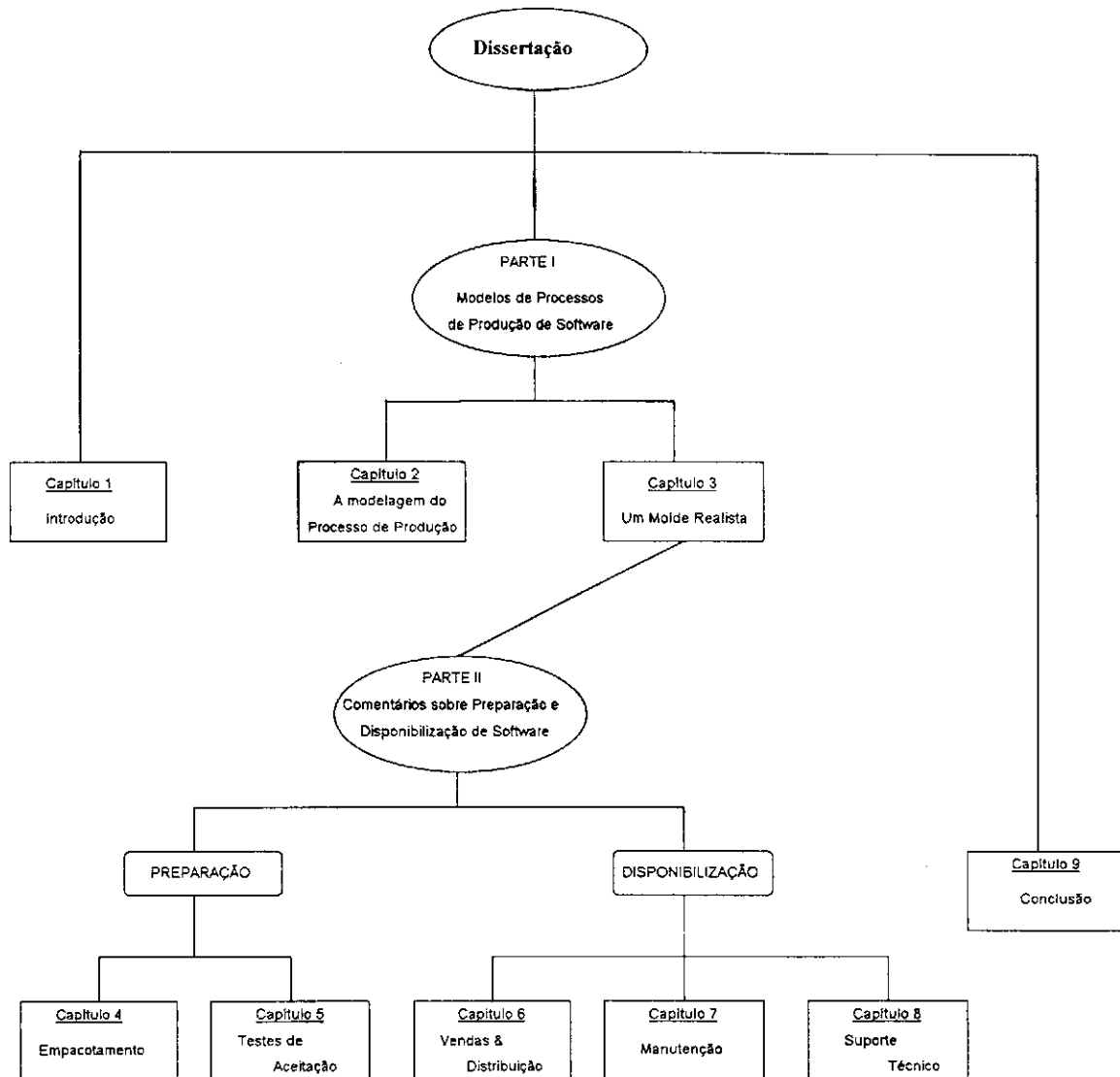


Figura 1.2 - Organização da Dissertação

A **Parte I** se dedica, em linhas gerais, aos Modelos de Processos de Produção de Software, destacando a relevância do trabalho para a Engenharia de Software. São descritos e criticados (nos capítulos 2 e 3) os trabalhos existentes na área, propondo-se um molde para a produção de software comercial frente a atual carência de trabalhos orientados ao mercado.

Mais especificamente, o capítulo 2 analisa os modelos de processos de produção de software, discutindo os principais modelos sugeridos no

passado (modelos clássicos) pela Engenharia de Software, e as novas propostas de modelos realizadas pela indústria e grupos de pesquisa (novos modelos). A análise finaliza identificando questões que desencadeiam na proposta de um molde que suporte melhor a realidade de produção de software comercial.

Uma visão completa do molde envolvendo os princípios adotados, a descrição das atividades previstas e a estrutura necessária para suportar o processo de produção de software se encontra no capítulo 3.

A **Parte II** instancia a utilização do molde genérico, identificando procedimentos componentes de um modelo para a produção de software Unix de prateleira. Esta tem como pré-requisito a leitura da parte I, pois necessita da compreensão do molde proposto, já que realiza comentários sobre a preparação e disponibilização de software para o mercado, as quais são fases integrantes do molde descrito no capítulo 3.

Foram exploradas as fases de preparação e disponibilização de software, por não serem devidamente consideradas nos demais modelos existentes e pela sua importância no processo de produção de software comercial como será verificado. A inclusão destas fases no processo de produção de software comercial colaboram para a definição de um ciclo de vida real para o software.

A parte II pode servir como ponto de referência para a construção de modelos ou guias de procedimentos práticos para a produção de software comercial. Embora sejam ilustrados os procedimentos referentes à produção de software Unix de prateleira, são destacados procedimentos que independem do segmento de mercado considerado.

A discussão da fase de preparação do software para o mercado se inicia no capítulo 4, onde são descritos os procedimentos necessários para que o software possa ser empacotado e eventualmente liberado para uso. São considerados aspectos técnicos e legais que merecem a atenção de produtores por influenciarem diretamente nos prazos e orçamento dos seus projetos.

A fase de preparação é concluída no capítulo 5, com a descrição dos testes de aceitação do produto antes de sua liberação definitiva para comercialização no mercado. São ilustrados os procedimentos necessários para a execução dos testes alfa e beta como mecanismos de avaliação do produto antes de disponibilizá-lo definitivamente para o mercado.

A discussão sobre disponibilização de software se inicia no capítulo 6, com a descrição dos principais canais de distribuição para software de prateleira e as práticas de vendas mais conhecidas, considerando suas margens de lucro, custos envolvidos e definição de preços. O suporte técnico com sugestões de procedimentos e métricas gerenciais é abordado no capítulo 7.

O capítulo 8 analisa a estrutura de manutenção de produtos no mercado, associada a pontos relevantes como gerência de configuração e portabilidade de software.

Os resultados obtidos com o desenvolvimento deste trabalho são discutidos conclusivamente no capítulo 9. São registradas também, as principais contribuições obtidas e algumas perspectivas para continuação e implementação deste trabalho.

Parte I

MODELOS DE PROCESSOS DE PRODUÇÃO DE SOFTWARE

Um Molde Genérico mais Realista para Produção e Disponibilização de Software Comercial

Capítulo 2.

A Modelagem do Processo de Produção

Neste capítulo são apresentados e analisados os principais modelos do processo de produção de software, tanto os tidos como clássicos como os modelos emergentes. Tem destaque também neste capítulo as questões que fomentaram a concepção de um molde representativo para a produção de software comercial.

2.1 Modelos Clássicos

Quando se faz referência ao processo de desenvolvimento de software, normalmente se associa um modelo ao processo que se deve seguir para construir, liberar e evoluir o produto desde a concepção de uma idéia até a retirada do sistema de operação. Tal modelagem de processo é frequentemente citada na literatura como ciclo de vida do software **[Ghez91]**.

Com a necessidade de se planejar e organizar melhor a construção de produtos de software, foram concebidos diversos modelos que tentaram disciplinar o ciclo de vida do software, definindo passos e atividades a serem fielmente seguidos. Cada um destes modelos possui ponto de vista próprio sobre o processo de produção do software. Dentre os modelos mais conhecidos e utilizados podemos citar o modelo em cascata, o modelo evolutivo, o modelo em espiral e o modelo transformacional, que formam basicamente os modelos tidos como clássicos.

2.1.1 Modelo Cascata

O **modelo cascata ou tradicional** foi o precursor na modelagem do processo de produção de software, especificado originalmente por Royce [Royc70], tendo sido bastante influenciado pelos projetos americanos de defesa aérea. O modelo detectou a falta de planejamento e os problemas decorrentes, reconhecendo a importância de se disciplinar a produção de software.

A principal proposta do modelo em cascata é obter um desenvolvimento sistemático de software, marcado por uma sequência de atividades genéricas (fases), geradoras de produtos intermediários que se agregam até chegar ao produto final, o software. Os produtos intermediários gerados pelas fases, geralmente são documentos que formalizam a inicialização da fase seguinte, como ilustra a figura 2.1.

Apesar do modelo original ter sido bastante modificado, na tentativa de se adequar ao dinamismo imposto pela evolução da tecnologia e dos ambientes de desenvolvimento, com a utilização de linguagens de quarta geração e ferramentas de desenvolvimento, este se caracteriza basicamente pelas seguintes atividades :

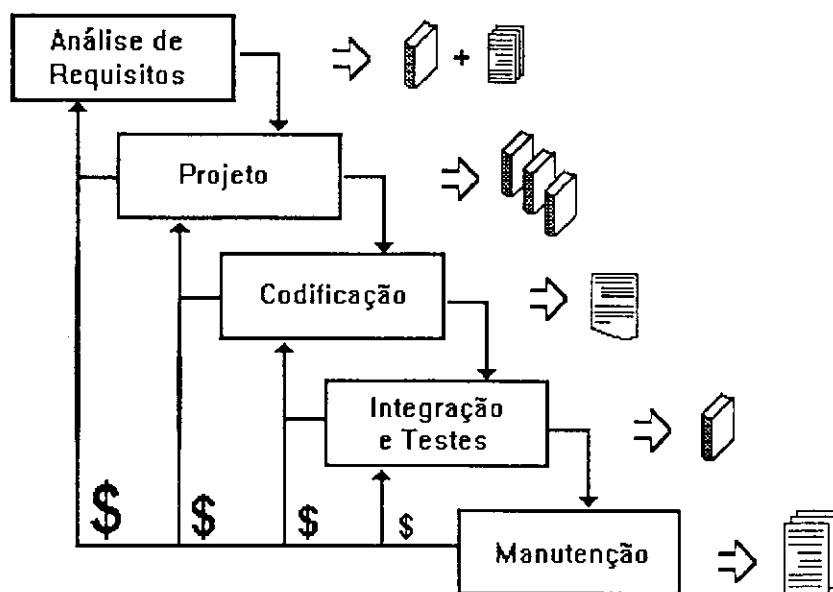


Figura 2.1 - Modelo Cascata

- Análise de Requisitos - Identificação e documentação dos requisitos do usuário do software. Preocupação em estabelecer o *que* será executado. Como produto desta fase tem-se uma especificação descritiva de requisitos e metas a serem atingidas, e idealmente um manual do usuário para formalização junto ao usuário.
- Projeto - Especificação de *como* o software atingirá os requisitos especificados anteriormente. Fase multidisciplinar que envolve a definição da arquitetura, estruturas de dados e procedimentos detalhados prontos para serem codificados.
- Codificação - Tradução da especificação detalhada de projeto para a linguagem de programação. São realizados testes para certificação de conformidade dos programas individualmente.

- **Teste e Integração** - Verifica se o conjunto de programas, ou sistema, alcançou os requisitos e metas (desempenho, portabilidade, robustez, etc...) estabelecidos. Todo o processo de aferimento envolve estratégias do tipo inspeção de código, *walkthrough*, testes da caixa-preta e caixa-branca [Perr88]. Esta fase libera o software para o usuário.
- **Manutenção** - Extensão do modelo original de Royce, que gerencia todo o processo de evolução de funcionalidade, adaptação ao ambiente (legislação, hardware, nova plataforma) e correção de erros. Esta fase gira em torno da adequação do software às necessidades do usuário.

Considerado como o ciclo de vida tradicional, o modelo em cascata foi o paradigma mais utilizado como base para o desenvolvimento da engenharia de software. Contudo, recebeu muitas críticas pela sua passividade e formalismo excessivo, que [Pres87] resume muito bem nos seguintes itens :

1. O fluxo sequencial do modelo raramente se verifica em projetos reais.
2. O modelo requer um entendimento completo dos requisitos do usuário no início do projeto, antecipando por completo suas necessidades, que geralmente são melhor compreendidas no seu decorrer, como resultado de um amadurecimento natural dos requisitos por parte da equipe de desenvolvimento e dos próprios usuários.
3. O usuário só tem acesso ao software no final do ciclo, podendo ter resultados desastrosos, que podem custar muito caro

para serem reparados. Este problema pode ser agravado com as possíveis mudanças de necessidades do usuário, que não são acomodadas dinamicamente neste modelo .

Além destas peculiaridades, o modelo cascata se mostra extremamente burocrático, pela sua excessiva preocupação na redação e aprovação de documentos para continuidade do ciclo de vida. A documentação em si é necessária para a continuidade de qualquer projeto, mas deve ser flexível para impor maior ou menor grau de formalismo ao processo de acordo com as necessidades de registro. Por exemplo, o desenvolvimento de protótipos requer baixo nível de documentação em suas versões iniciais, exatamente para dinamizar a interação constante com os usuários.

Estas deficiências no modelo tradicional fomentaram o surgimento de outros modelos de produção de software, descritos nas seções seguintes.

2.1.2 Modelos Evolutivos

Vários modelos de produção de software surgiram com o intuito de complementar o modelo cascata. Na tentativa de se obter um modelo que não fosse monolítico nem apresentasse a rigidez do modelo tradicional, os **modelos evolutivos** promovem o processo de produção incrementalmente com participação intensa dos usuários. Os subprodutos ou incrementos são liberados na medida em que estiverem prontos para avaliação dos usuários, que indicam os ajustes necessários para sua evolução, e assim sucessivamente.

Estes modelos se baseiam no princípio advocatedo por [Broo75], que defende que a primeira versão de um sistema deve ser encarada como um protótipo descartável, cuja proposta é consolidar os requisitos dos usuários para se montar uma base sólida para o desenvolvimento real do software.

Existem na literatura vários modelos de processo que utilizam esta abordagem evolutiva. No modelo de implementação incremental, a análise de requisitos e o projeto são executados de forma a identificar componentes do sistema que possam ser desenvolvidos e liberados em tempos diferentes, com ciclo de vida próprio, podendo assim ser incrementados a medida em que se conhece melhor o sistema.

Uma variação do modelo de implementação incremental é encontrado no modelo de desenvolvimento incremental, que procura acomodar mudanças explicitamente durante o desenvolvimento. Segundo o modelo, os incrementos são definidos na fase de análise e passam separadamente pelo restante do ciclo de vida. A diferença é que o usuário avalia cada incremento liberado, aumentando o conhecimento de todos sobre o sistema, o que possivelmente permite acomodar mudanças de requisitos em tempo hábil [Ghez91].

A prototipagem é o modelo evolutivo de produção de software mais referenciado na literatura. É definido por [Agre86] como "o processo de construção de um modelo de funcionamento de um sistema ou parte dele ... tendo como objetivo esclarecer as características e operações do produto ou sistema pela construção de uma versão que pode ser trabalhada".

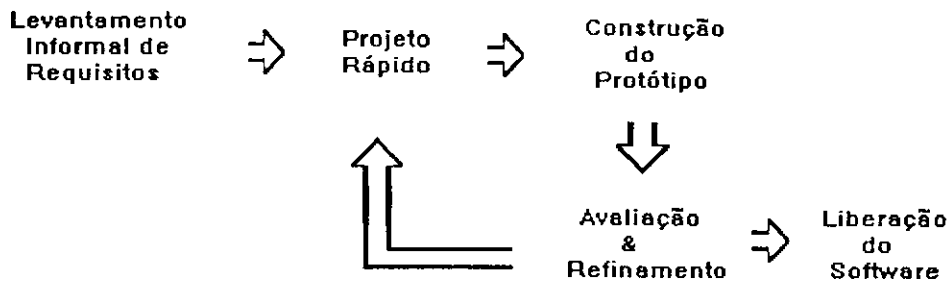


Figura 2.2 - Modelo Evolutivo (prototipagem)

A técnica de prototipagem é utilizada fundamentalmente para transformar progressivamente um modelo inicial do sistema em um software, conforme ilustrado na figura 2.2. Situações nas quais os usuários e desenvolvedores não estão bem seguros quanto aos requisitos a serem considerados pelo sistema são candidatas a utilizar prototipagem. Assim sendo, o objetivo é liberar rapidamente um protótipo inicial a partir das informações levantadas preliminarmente, facilitando a comunicação com o usuário. Este protótipo é então refinado com o usuário, até que se desenvolva um protótipo considerado operacional para os requisitos do usuário.

2.1.3 O Modelo Espiral

O modelo espiral com uma filosofia voltada para análise de risco e controle de custos[Boeh88], além de fornecer estrutura para aspectos técnicos, considera a análise de custos agregados durante o processo de produção como um fator de risco na gerência. Este metamodelo (pois acomoda outros modelos de produção) chama atenção para a continuidade cíclica da produção do software em contraposição à linearidade do modelo cascata.

O modelo espiral, conforme ilustrado na figura 2.3, se apoia na execução cíclica de atividades em forma de espiral, onde cada ciclo da

espiral passa necessariamente por quatro estágios distintos no processo de produção. Cada estágio é representado por um quadrante no eixo cartesiano e o raio da espiral representa os custos associados, que formam a base para a análise contínua dos riscos.

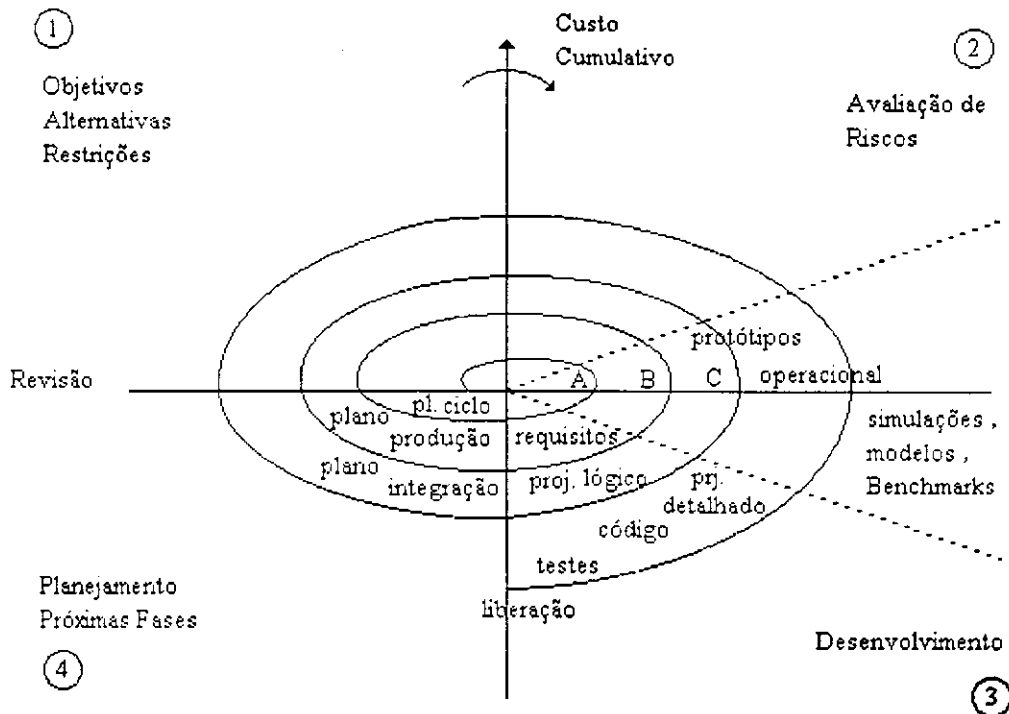


Figura 2.3 - O Modelo Espiral para Produção de Software

Portanto, o ciclo se compõe da determinação dos objetivos no seu primeiro estágio, seguido de um avaliação dos riscos envolvidos na seleção das alternativas para a execução dos objetivos no segundo quadrante.

No terceiro estágio se concretiza o desenvolvimento das atividades previstas, verificando sempre o nível de dificuldade econômica para a continuidade da elaboração do produto, o que pode significar a passagem de um projeto lógico para um projeto detalhado ou a evolução para uma nova versão. Caso seja acordada a continuidade no ciclo de vida, o

planejamento das próximas fases é realizado no estágio quatro, onde são revisados os avanços alcançados e as próximas metas.

Além destes modelos podemos citar também o modelo transformacional, que se baseia na utilização de especificações formais para gradualmente ser transformado em código. Infelizmente, a utilização de mecanismos formais de especificação ainda está restrita a algumas áreas de aplicação como especificação de protocolos de comunicação e não se constitui numa abordagem utilizada na produção de software comercial. Uma descrição mais aprofundada desta área pode ser encontrada em [Agre86].

2.2 Comentários sobre os Modelos Clássicos

O surgimento de modelos de representação do processo de produção de software foi motivada originalmente pela necessidade de ordenação das atividades de desenvolvimento frente à complexidade verificada nos sistemas de informação manipulados na época. A partir deste momento, verificou-se a tentativa de assimilação de novas características que tornassem os modelos mais representativos frente à diversidade de alternativas de produção de software, valorizando-se alguns aspectos específicos.

O modelo Cascata apesar de ter recebido várias modificações na sua versão original, foi sempre um modelo dirigido por *documentos*. De forma que se baseia na elaboração de documentos para que o processo possa evoluir entre suas fases estanques. É considerado um modelo extremamente rígido e monolítico, com dificuldade de alterar requisitos e seu fluxo previsto de trabalho. Os demais modelos citados também possuem suas particularidades. O modelo evolutivo trata basicamente de *incrementos funcionais*, que levam ao escalonamento das atividades até a

conclusão do projeto. O Modelo transformacional é dirigido por sucessivos refinamentos de *especificações* até a sua implementação, e o modelo espiral valoriza nitidamente a questão de análise de *custos*.

Ao se examinar a utilidade destes modelos na produção de software comercial, nosso objeto de estudo, concluí-se que: eles servem principalmente como instrumentos didáticos para explicar o processo de desenvolvimento ; adotam fluxos de trabalhos genéricos [Hump89] ; e, apresentam as seguintes restrições para serem aplicados à realidade de software no mercado :

- i) Não cobrem todo o ciclo de produção do software comercial (excluem por exemplo, aspectos críticos para o sucesso como empacotamento para distribuição, marketing e suporte técnico, que necessitam planejamento e procedimentos técnicos para realização) ;
- ii) Reforçam aspectos de desenvolvimento, desfocando visão de atividades com maior fatia de custos (tais como suporte e comercialização) ;
- iii) Dificilmente assimilam mudanças estratégicas inesperadas no processo ;
- iv) Não condizem com à realidade vivenciada no mercado, seja pelo excessivo formalismo, pela inadequação do modelo aos ambientes ou pela sua generalidade;
- v) São, na grande maioria, modelos orientados à tarefas, refletindo a familiaridade com outras engenharias que trabalham essencialmente com produtos manufaturados (software é uma atividade intelectual e multidisciplinar e não se constrói da mesma forma que um prédio como o modelo cascata prevê)

Em se fazendo estas afirmações, não se quer dizer que os modelos clássicos não tenham utilidade atualmente, nem sugerir que sejam totalmente descartados para balizar a produção de software. Pelo contrário, estes modelos possuem características (incrementos, custos, documentos, especificações) que os credenciam para modelar o desenvolvimento de determinados tipos de software. Por exemplo: os modelos evolutivos endereçam muito bem a questão de desenvolvimento interativo de interface junto a usuários que não conhecem bem seus requisitos ; o modelo cascata pode ser utilizado para a construção de software onde se tenha certeza de que seus requisitos não serão modificados durante sua construção (como na produção de software embutido para automóveis e robôs pré-programados).

Entretanto, a questão crucial desta análise diz respeito a situação atual da indústria de software como um todo, que é mais preocupante do que transparece pelas restrições acima na aplicação de modelos de processos. A indústria parece operar sem modelo documentado ou compartilhado por membros da equipe !

Levantamentos preliminares realizados junto a casas-de-software nacionais na tentativa de modelar o processo de produção de software comercial, evidenciaram que as atividades de produção e gerência de software deste segmento são realizadas empiricamente, sem o apoio efetivo de modelos de processos. No acompanhamento de empresas emergentes que desejam produzir software para exportação vinculados ao programa SoftEX 2000, verificou-se a mesma carência.

A realidade no exterior não se distancia da vivenciada localmente. Dados coletados em estudo de campo realizado pelo Instituto de Engenharia de Software (SEI) da Universidade Carnegie Melon nos

Estados Unidos [Kras92], mostram que a vasta maioria das organizações desenvolvedoras de software daquele país não possuem ou não usam modelos de processos.

O desinteresse da indústria pelo uso de modelos de processos, pode ser atribuído parcialmente à dissociação de tais modelos da realidade de mercado.

Esta realidade tem fomentado o aparecimento de novos modelos que sejam mais efetivos no auxílio ao processo de produção. A próxima seção explicita alguns destes modelos e seus respectivos propósitos.

2.3 Novos Modelos

Recentemente surgiram alguns trabalhos, ainda em fase de ajustes e validação, sugerindo mudanças na modelagem de processos, buscando elaborar estruturas de apoio aos produtores de software diante do dinamismo e complexidade do processo de produção.

Serão discutidos dois destes modelos por serem trabalhos mais representativos, um desenvolvido e utilizado internamente numa indústria japonesa e o segundo financiado pelo departamento de defesa norte-americano.

2.3.1 O Modelo CDPM

Com o objetivo de agilizar o seu processo de produção de software, a Fujitsu japonesa tem trabalhado desde 1987 na construção e calibragem do Modelo de Processo de Desenvolvimento Concorrente (CDPM). Essencialmente, o modelo visa reduzir o ciclo de desenvolvimento saindo de uma estrutura de desenvolvimento sequencial e linear para o

desenvolvimento concorrente de múltiplas atividades, desde a fase de especificação até o sistema de testes [Aoya93].

O CDPM segue uma tendência na utilização mista de modelos existentes, onde são estipulados incrementos ou melhorias funcionais, como nos modelos *evolutivos*, sob a forma de módulos e funções que são implementados simultâneamente seguindo a maioria das fases previstas no modelo *casca* : análise de requisitos, projeto, implementação, teste de unidade, teste de pré-integração, integração do sistema, teste da integração (vide figura 2.4).

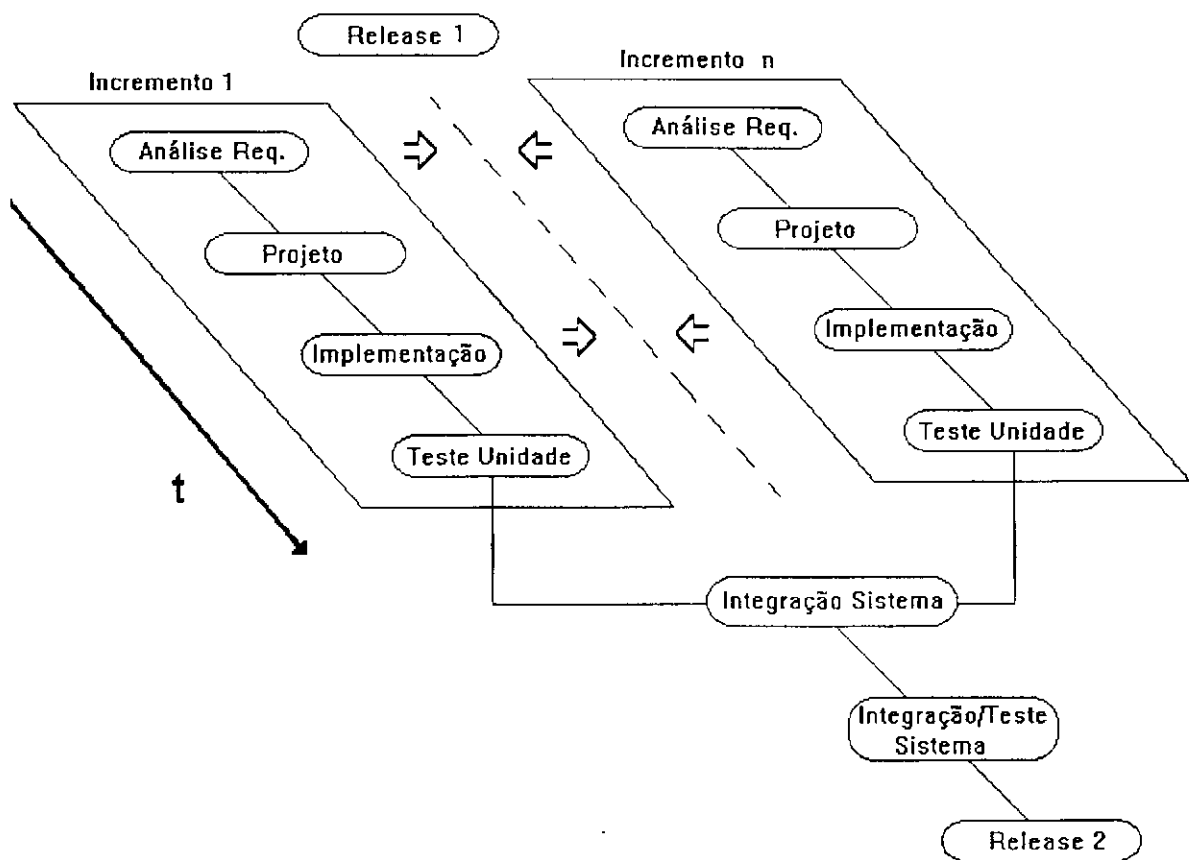


Figura 2.4 - Modelo de Produção de Desenvolvimento Concorrente (CDPM)

A grande vantagem deste modelo está na velocidade de liberação de múltiplas funções executadas paralelamente, que são integradas e testadas antes do tempo normal esperado para o desenvolvimento utilizando um modelo convencional, reduzindo assim o tempo de liberação das versões do software.

Evidentemente que este benefício exige uma perfeita harmonia entre técnicas de gerenciamento de processo, produtos, projeto e da própria organização. Os principais desafios são inerentes à consistência de produtos ou componentes compartilhados pelos diversos times de desenvolvimento e o sincronismo necessário para liberação simultânea para testes. Técnicas de modularização, reutilização e gerência de configuração são essenciais para este ambiente.

Técnicas de gerenciamento de projeto envolvendo fixação da quantidade de pessoas nos times e respectiva carga de trabalho, redistribuição permanente de trabalho e sincronismo de grupos estão sendo utilizadas e refinadas durante a calibragem do modelo. Os resultados são altamente positivos até o momento para os sistemas produzidos internamente na Fujitsu.

A utilização deste modelo em projetos com alto nível de comunicação e com mais de um milhão de linhas de código fonte, acusou uma redução de 75% do tempo de desenvolvimento em relação a sistemas similares produzidos segundo modelos convencionais.

Algumas das restrições dos modelos clássicos para a produção de software comercial, identificadas na análise realizada anteriormente (seção 2.2), continuam no modelo CDPM. Isto se verifica principalmente porque, apesar do dito modelo utilizar frentes de trabalho simultâneas para reduzir o ciclo de produção, ele utiliza o modelo cascata como base para

produção, focando apenas aspectos de desenvolvimento. Mesmo realizando uma reengenharia para adaptação do modelo cascata à realidade de produção de software concorrentemente, o CDPM continua não abordando o ciclo completo de produção, desconsiderando aspectos de mercado (p.ex. marketing, suporte técnico e empacotamento) que permeiam a produção de software comercial.

2.3.2 O Modelo Cosmos

O modelo COSMOS [Raym91] rompe com a arquitetura tradicional dos outros modelos, explicitando a necessidade de considerar aspectos gerenciais no processo de produção de software. O surgimento do COSMOS destaca a necessidade de se visualizar a produção de software como um universo, composto por diversas variáveis que o influencia diretamente, em oposição a visão simplista de solucionar apenas as etapas de produção como na maioria dos modelos clássicos.

Motivado pela dificuldade que os modelos de processos existentes tem em ordenar o fluxo de informações existente nos projetos de software e principalmente orientar eficientemente a gerência como proceder neste sentido, este modelo combina algumas características de modelos existentes, incorporando uma estrutura tridimensional, ACI, de gerência coevolutiva, composta pelos elementos *Atividades, Comunicação e Infraestrutura*.

A estrutura de *atividades* se preocupa em como o processo será executado em termos de definição de tarefas, ordenação e dependências entre tarefas, e escalonamento. Enfim, define a abordagem de produção em função do contexto, podendo ser, por exemplo, uma abordagem espiral se os custos forem uma questão prioritária, ou prototipação caso os requisitos não sejam bem conhecidos. Por ser um modelo voltado para a

gerência, são inseridas também questões não técnicas a nível de controle para avaliar o andamento e fornecer possíveis alternativas ao projeto. Esta estrutura defende fortemente a prototiteração (compreensão de um problema por prototipagem) com os colaboradores do projeto e a coevolução com os elementos de comunicação e infraestrutura.

A estrutura de *comunicação* fornece à estrutura de atividade a base para execução das suas tarefas, modelando através de um mapa de funções as responsabilidades e dependências entre as pessoas, os canais de comunicação e as respectivas funções. Com a utilização do mapa de funções são identificadas a estrutura real de operação do ambiente, as necessidades, os gargalos, as desconexões e redundâncias de uma maneira geral, além de explicitar a contribuição de cada um para o processo.

Visando o cumprimento dos objetivos imediatos da empresa e principalmente o estabelecimento dos objetivos a longo prazo, foi concebido o elemento *infraestrutura* para estipular as linhas de ação da empresa em termos de qualidade dos produtos e produtividade da empresa. A infraestrutura apoia a execução das atividades e da estrutura de comunicação, sendo necessário fundamentalmente para a evolução do produto, um ponto crítico no ciclo de vida do software.

O modelo como um todo ainda se mostra muito incipiente, deixando em aberto a sua utilização como ferramenta de apoio para diversos tipos de ambientes e necessidades. Apesar de apresentar uma visão mais abrangente sobre a produção de software, não se pode afirmar que este venha a amenizar eficientemente a distância entre os modelos de processos existentes e a realidade de produção de software comercial. De uma maneira geral, são valorizadas as questões de gerência de

comunicação de projeto e a infraestrutura necessária para a organização atingir seus objetivos de curto e longo prazo, ficando em aberto como o ciclo de produção deve ser conduzido.

2.4 Um Molde Genérico para Modelos mais Realistas

Ao se analisar a evolução dos modelos de processos e dos próprios processos de produção de software em termos de qualidade e sofisticação requeridas, verifica-se que tanto o processo de produção, quanto a construção de modelos eficazes no apoio do processo de produção, são complexos. Isto porque o processo e o modelo devem se ajustar dinamicamente a novas necessidades durante sua execução, e contemplar a multidisciplinaridade do processo de produção, envolvendo atividades de gerência de processos e produtos.

Como foi observado, os modelos clássicos tentaram atender uma necessidade da época em fornecer uma visão abrangente do processo de produção (enfocando mais os aspectos de desenvolvimento) para auxiliar no planejamento e controle da produção de software. Atualmente, verificamos que estes modelos adquiriram caráter didático, não sendo utilizados de fato como instrumentos de controle da produção de software.

A tentativa de construir um modelo representativo para a produção de software comercial, nosso objetivo, esbarrou no delineamento do universo de trabalho, tendo em vista a diversidade de ambientes computacionais existentes com alternativas próprias de atuação em função dos seus recursos e necessidades.

De fato, a produção de software se especializou em função da diversificação da demanda, a ponto de segmentar naturalmente o mercado para poder atender eficientemente os seus usuários e superar a forte

concorrência estabelecida, ou então identificar novos nichos de mercado ainda não explorados. Como exemplo desta realidade, encontramos software em ambientes militares, em estabelecimentos comerciais, no governo, na escola e até nas nossas casas e carros, o que caracteriza ambientes de produção distintos com forma de atuação específica de acordo com a demanda do seu segmento de mercado.

Assim sendo, a construção de modelos efetivos para o atual estágio em que se encontra a produção de software para o mercado, deve considerar os procedimentos que caracterizam cada segmento de mercado (vide figura 2.5).

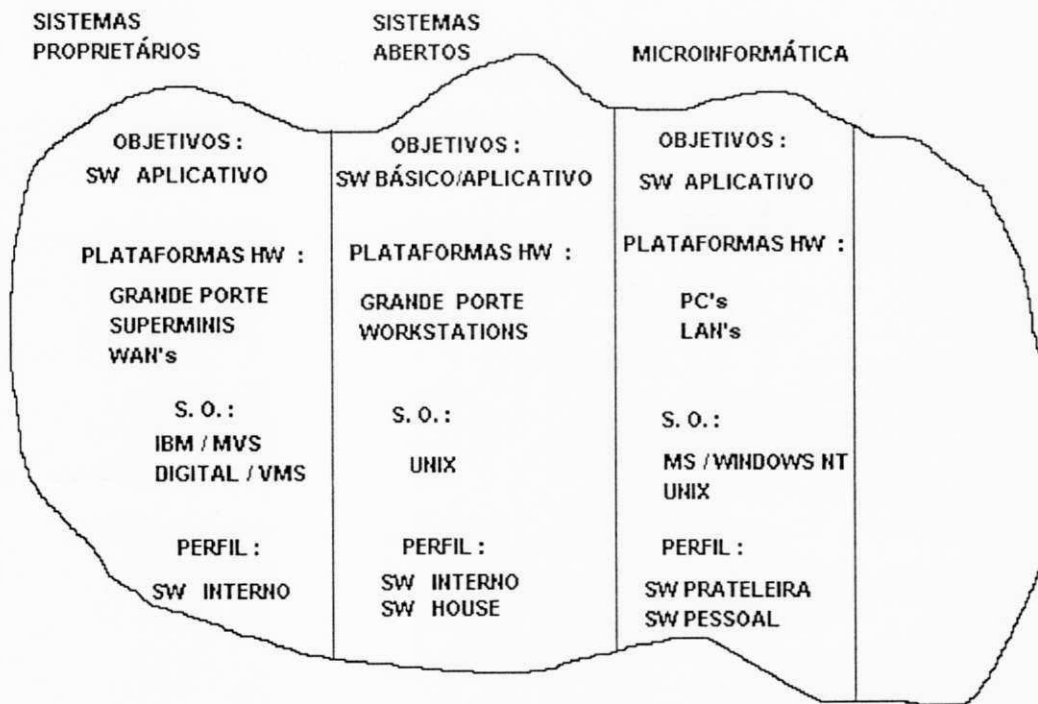


Figura 2.5 - Ilustração da Segmentação de Mercado

Os segmentos de mercado podem ser diferenciados em função dos seguintes fatores :

- Objetivo(s) em termos do tipo de software a ser desenvolvido, como por exemplo, software aplicativo, básico, científico ou embutido¹.
- Plataformas de Hardware que consistem da combinação de equipamentos eletrônicos sobre os quais funcionam o software, os sistemas operacionais, e as ferramentas disponíveis para a produção de software (CASE, compiladores, depuradores, pacotes geradores de códigos, dentre outras)
- Sistemas Operacionais - Conjunto de programas que gerenciam as atividades e recursos de um computador ou conjunto de computadores.
- Perfil Mercadológico o qual espelha a postura do produtor de software perante o mercado de informática, podendo possivelmente ser incluído dentre uma das seguintes categorias :
 - Desenvolvimento de software pessoal
 - Desenvolvimento interno (Centro de Processamentos de Dados ou Centro de Informações)
 - Produção de software comercial para o mercado ("Software-House")
 - Desenvolvimento sob encomenda
 - Terceirização
 - Agregação de valor a partir de produtos de terceiros (VAR - Value Added Reseller)

¹Software embutido ou "embedded" na língua inglesa, é um software residente em memória do tipo read-only, utilizado para controlar produtos da indústria e de mercado de consumo [Pres87]. O seu desenvolvimento é realizado sobre especificações rígidas, com a realização de testes exaustivos pelos seus produtores. Geralmente, seus usuários não tem consciência de que o benefício fornecido é resultado de uma execução de software. Controle digital nos painéis de carro é um exemplo comum de software embutido.

Para se identificar um segmento de mercado devem ser considerados todos os fatores, pois existem subdivisões dentro de cada segmento. Por exemplo, dentro do segmento de sistemas abertos, existem algumas empresas que concentram seus interesses em produzir software aplicativo (objetivo) para Unix (sistema operacional), a ser comercializado no mercado (perfil mercadológico) de workstations. Um outro subsegmento pode ser atribuído a um produtor de software aplicativo para Unix que atenda ao mercado de redes de microcomputadores.

A diferenciação de um segmento de mercado para outro, parece ser sutil, entretanto altera substancialmente a forma de produção de software. Ambientes de produção de software para uso interno, diferenciam consideravelmente daqueles que desejam atender ao mercado de uma maneira geral. Mesmo possuindo as mesmas características de hardware, sistema operacional e objetivos (p.ex. software aplicativo), o mercado é o diferencial, exigindo padrões de qualidade e atendimento a clientes que seriam muito mais rígidos do que os normalmente exigidos para o software interno.

Ciente desta realidade, verificamos que não é realista recomendar um único modelo para qualquer tipo de software, empresa, equipe e segmento de mercado, pela complexidade do processo e especialização do mercado. Portanto, parece plausível propor um "molde genérico" para a produção de software comercial, a partir do qual modelos específicos possam ser construídos e validados para casos particulares e realistas da indústria. Dizemos que a construção de um modelo específico resulta da "instanciação" do molde a um segmento particular de mercado.

A definição de um molde genérico para a produção de software comercial, facilita a compreensão humano dos procedimentos

envolvidos na produção de software para o mercado, identificando os aspectos críticos para o sucesso e amenizando as dificuldades para a concepção de modelos específicos que atendam os diversos segmentos de mercado.

Na definição do molde genérico, este trabalho lançou mão da experiência de empresas produtoras de software comercial, afim de tornar o molde o mais realista possível, mesmo sendo um trabalho incipiente que necessita do retorno da indústria para seu aperfeiçoamento.

O capítulo seguinte se dedica à descrição mais abrangente do molde genérico para construção de modelos de produção de software comercial. Serão relatados seus princípios e características que tentam resgatar as necessidades de um molde destinado a atender mais realisticamente as atividades, métodos e práticas de mercado, encerrando assim a parte I desta dissertação. Os demais capítulos, referentes a parte II, ilustram a instanciação do molde quando utilizado para identificar alguns procedimentos componentes de um modelo para a produção de software Unix de prateleira.

Capítulo 3. Um Molde Mais Realista

3.1 Motivação

Como se pode observar, o mercado de software atingiu um elevado grau de especialização de atividades em consequência da diversificação de demandas, requerendo capacitações específicas dos produtores para aumentar sua competitividade no acirrado mercado. Os mercados segmentados desejam produtos de qualidade, fornecidos em tempo hábil, com preços atraentes e principalmente com uma estrutura adequada de comercialização, suporte técnico e evolução destes produtos.

Esta realidade marca uma nova etapa na produção de software comercial, onde o sucesso não pode ser alcançado sem o pleno exercício de atividades de gerência que orientem os esforços de produção. A solução fornecida pela engenharia de software para orientar o atendimento destas necessidades foi a elaboração de modelos de processos de produção de software, como visto em capítulos anteriores.

Entretanto, ao se analisarem os modelos de processos, verifica-se uma dissociação destes com a realidade de produção de software para o mercado. A produção de software para os diversos segmentos de mercado necessita de modelos dinâmicos que mapeiem a realidade vivenciada para apoiar efetivamente a condução do processo de produção.

Definitivamente, a proposta dos modelos conhecidos até o momento possui caráter eminentemente didático em se tratando de software comercial. A grande maioria destes modelos desconsidera o ciclo real de vida do software, ficando restritos aos aspectos técnicos de *como se desenvolver* produtos. O enfoque na disponibilização de produtos é mínimo, se não inexistente.

A concentração dos modelos existentes em desenvolvimento, faz com que estes se distanciem de questões relevantes para o sucesso do projeto, responsáveis por grande fatia dos custos totais do software. [Kras92] compartilha deste sentimento, alertando para a desatenção a questões de concepção de produtos, comercialização, suporte pré e pós-venda, dentre outras.

Reiterando as restrições colocadas no capítulo anterior, quando da análise dos modelos de processos existentes (seção 2.2), a produção de software não pode ser considerada uma atividade linear, bem comportada e de acordo com algumas fases estáticas que visam simplesmente construir um produto. O desenvolvimento e disponibilização de software comercial devem ser encarados como multidisciplinares, que necessitam de sinergia entre suas diversas atividades integrantes. Aqui, enxerga-se a produção de software comercial composta basicamente pelas atividades de administração, desenvolvimento, suporte técnico, manutenção, marketing, vendas e controle de qualidade.

De fato, tanto o processo de produção, quanto a construção de modelos efetivos no apoio do processo de produção de software são complexos. Isto porque tanto o processo quanto o modelo devem se ajustar dinamicamente a novas necessidades, e contemplar a multidisciplinaridade do processo de produção, envolvendo atividades de gerência de processos e produtos.

Para possibilitar o aumento progressivo do conhecimento sobre modelagem de processos de produção de software, e com base em observações sobre o comportamento do mercado, foram desenvolvidos protótipos de modelos, denominados de "moldes".

A necessidade de um molde genérico surgiu à medida em que seria inadequado recomendar um único modelo representativo para qualquer tipo de software, empresa, equipe e segmento de mercado. Propõe-se então, um molde representando genericamente a produção do software comercial, a partir do qual modelos específicos possam ser construídos e calibrados para casos particulares e realistas do mercado.

Faz-se em seguida, uma descrição um pouco mais abrangente do molde desenvolvido para a produção de software comercial que resgata questões desprivilegiadas pela literatura, e que no nosso entendimento são essenciais para gerenciar projetos de software que visem atender ao mercado de fato, com uma visão mais abrangente e realista do ciclo de vida.

3.2 Princípios Adotados para a Definição do Molde

A existência de modelos de processos, clássicos ou novos, se baseia numa gama de princípios que cada modelo em particular deseja destacar. Assim como o modelo espiral se preocupa com custos, o modelo cascata com a execução linear de tarefas geridas por documentos, a

modelagem do processo de produção de software comercial obrigatoriamente deve resgatar a realidade de mercado.

Baseado nas restrições dos modelos existentes para o tipo de modelagem desejada e nas necessidades verificadas em contacto com empresas nacionais (Infocon, Light Software e empresas emergentes vinculadas ao programa SoftEX 2000) foram isolados alguns princípios que demonstram as particularidades do software comercial em função das variantes de mercado. Os princípios descritos abaixo são a base para a construção do molde desejado :

- *Abrangência*. Representação da realidade de produção de software comercial em todo o seu escopo, destacando todos os aspectos que devem ser considerados no ciclo de vida real do software. Sugere uma extensão dos modelos existentes para identificar aspectos extra-desenvolvimento, responsáveis por grande parcela dos custos do ciclo de vida. As questões de comercialização e sustentação são críticas para produtores que não tem uma estrutura mais sólida no mercado, podendo fazer com que seus produtos não prevaleçam no mercado.

- *Separação de Domínios*. Decomposição do problema para facilitar a divisão de tarefas, permitindo a execução de estratégias de produção e de disponibilização, compatibilizando a complexidade do problema com os recursos disponíveis. Além de possibilitar o escalonamento de tarefas em função da estrutura de produção disponível, a modularização das atividades permite acelerar ou retardar a colocação de produtos no mercado de acordo com a demanda (real ou potencial) de mercado. Os produtores monitoram os seus concorrentes, tentando sempre acompanhar as novas características que os produtos similares já possuem, ou se antecipar aos seu concorrentes para ter vantagens no mercado.

• *Paralelismo*. Possibilidade de executar múltiplas funções de gerência de processos e de produtos simultâneamente. O mercado não aceita esperar muito tempo por produtos que lhe atendam. Caso um produtor seja moroso, provavelmente outro que possua um melhor tempo de resposta ("time-to-market") ganhará a concorrência. Estávamos acostumados a ver um paralelismo apenas na implementação de rotinas (código, módulos) pelos diversos membros do time de desenvolvimento. Entretanto, esta necessidade se expandiu para todo o ciclo de produção, executando paralelismo em atividades de uma versão específica (concepção, marketing, testes) e entre versões (ciclos independentes). Baseado na separação de domínios, versões de produtos hoje são desenvolvidos simultâneamente, reduzindo consideravelmente o seu tempo de desenvolvimento em relação a desenvolvimento sequencial, apoiado por técnicas e modelos específicos [Aoya93]. Isto pode ser notado no mercado, onde se verifica lançamentos quase que simultâneos de produtos entre concorrentes, que evidentemente estavam aguardando a melhor hora para lançá-los no mercado, trabalhando sempre na evolução concorrente de produtos com diversos grupos de trabalho.

• *Feedback*. Capacidade que deve ter o modelo para retomar o processo de produção a fases ou atividades anteriores a qualquer momento, em virtude de problemas técnicos ou decisões estratégicas da empresa. O redirecionamento de produtos deve ser acomodado pelo modelo. Atividades como marketing, controle de qualidade e testes de aceitação com usuários podem exigir o redesenvolvimento de partes do produto ou até mesmo o abandono do produto, se for o caso.

• *Integração*. Assimila as necessidades de cooperação entre as atividades multidisciplinares componentes da produção de software comercial. A atuação de grupos isoladamente sem a devida sintonia não exterioriza aos usuários a filosofia do produto ou até mesmo da empresa. Os exemplos de integração são

fatos. Não se pode conceber por exemplo, que os grupos de qualidade e desenvolvimento não tenham uma perfeita sintonia para o estabelecimento e uso de métricas da organização, assim como o suporte técnico, vendas e marketing devem estar sincronizados para representar a empresa junto a seus usuários.

- *Evolução.* Propriedade que garante a continuidade cíclica do produto no mercado. A utilização deste princípio prevê o incremento de melhorias ou ajustes na última versão colocada no mercado. O mercado está continuamente exigindo mudanças, ajustes e melhorias nos produtos existentes. A evolução garante a continuidade do produto no mercado e conseqüentemente a vitalidade da própria empresa.

- *Contexto mercadológico.* O modelo deve estar em completa sinergia com o mercado, destacando seus conceitos, valores e práticas, de forma a colocar a Engenharia de Software a serviço das necessidades mercadológicas.

- *Flexibilidade.* Conseguída com a agregação dos demais princípios, dinamiza o processo de produção. A flexibilidade pode significar a assimilação de mudanças de requisitos, prazos, metas e orçamentos, pessoal, de ferramentas e plataformas de desenvolvimento.

3.3 Um Molde Mais Realista

A construção de um molde representativo para o processo de produção de software comercial, fundamentado nos princípios descritos, se iniciou com a caracterização dos elementos componentes do ciclo de vida do software comercial e seu comportamento em relação ao tempo.

Segundo o molde (ilustrado na figura 3.1), o processo de produção de software comercial pode ser organizado basicamente em torno de três

componentes integrados: fases, atividades monofásicas e atividades polifásicas. Assim sendo, num nível mais alto de abstração, encontram-se as **fases**, que identificam os principais estágios distintos no processo de produção de software, independentemente do segmento de mercado. Num nível imediatamente abaixo, estão as **atividades monofásicas** que operacionalizam as fases de acordo com a realidade de cada segmento de mercado. Por fim, as **atividades polifásicas** fornecem a sustentação necessária para o sucesso do produto durante todo o ciclo de vida real do software.

O molde identifica então, quatro fases que caracterizam genericamente o ciclo de vida de um software comercial : *Concepção, Desenvolvimento, Preparação e Disponibilização*. Estas fases condizem com as atitudes necessárias para o ciclo de produção, onde se começa a idealizar o produto, traçando estratégias para sua execução (concepção), seguido da transformação destas idéias num software (desenvolvimento), passando pela necessidade de transformar o software num produto comercializável com rigoroso controle de qualidade (preparação) e findando na sua comercialização e sustentação no mercado (disponibilização).

Com relação as atividades monofásicas, verifica-se que estas possuem propósitos específicos, que são atingidos com a realização de procedimentos visando a qualidade do produto. Com relação ao modelo cascata, já que foi a base para os modelos clássicos e alguns dos modelos recentes, suas fases estanques (especificação de requisitos, projeto, implementação, etc.) são consideradas atividades monofásicas de algumas fases do molde, representando uma forma de operacionalização das fases genéricas.

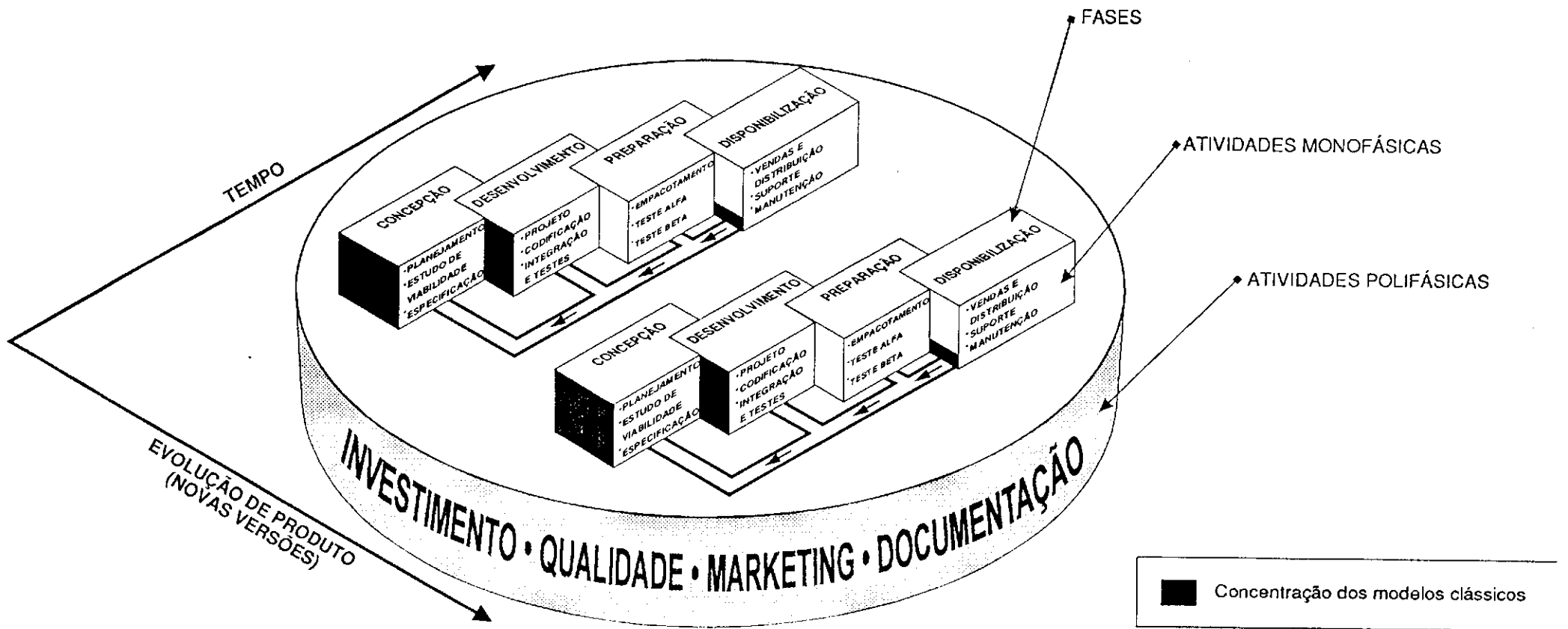


Figura 3.1 - Um Molde Realista Para a Produção de Software Comercial

De acordo com o molde proposto, são detectadas atividades que ocorrem em mais de uma fase ou durante todo o ciclo de vida do software, demonstrando serem de extrema importância para a gestão, controle e acompanhamento do ciclo de vida do software. Estas atividades, denominadas atividades polifásicas ou de acompanhamento, possuem normalmente estrutura de funcionamento e procedimentos próprios. As principais atividades polifásicas identificadas pelo molde são: *Investimento, Qualidade, Documentação e Marketing*.

As atividades polifásicas são extremamente importantes para a representatividade do molde, uma vez que podem alterar o fluxo convencional do ciclo de vida em função da realidade de mercado. Ao se considerar, por exemplo, o movimento de concorrentes do mercado através do marketing e a disponibilidade de investimentos da empresa, um produtor pode decidir retornar à fases anteriores para redesenho do projeto ou inclusão de novas características.

Vale ressaltar que o molde se preocupa em adotar realmente os princípios apregoados para a produção de software comercial. A inclusão da noção de tempo, impõe dinamismo ao molde, possibilitando a execução de fases concorrentemente e a evolução de produtos para novas versões.

Na sequência do capítulo serão apresentados mais detalhadamente as fases, atividades monofásicas e atividades polifásicas que compõem o molde.

3.4 Fases e suas atividades

Para efeito desta seção, o processo de produção de software é simplificado pela ilustração na figura 3.2, onde se destacam interações entre as quatro fases especificadas, cada uma com suas atividades

monofásicas e procedimentos correlatos, possuindo ainda atividades polifásicas que completam a sustentação ao processo.

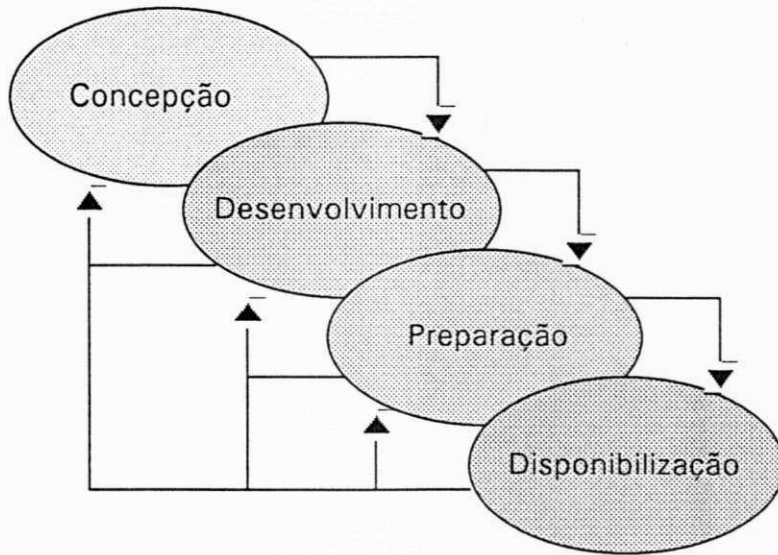


Figura 3.2 - Visão Simplificada das Fases do Molde

A fase de **concepção** se concentra no processo de identificação da informação a ser processada e é disparada seja pela descoberta de oportunidades no mercado, solicitações de clientes, contratos com parceiros ou evolução planejada de um software já disponível.

Nesta fase se discute amplamente *o quê* realmente se quer conceber em termos de funcionalidade, interfaces, performances, critérios de validação e restrições de projeto, para ficarem estabelecidos os parâmetros de um projeto bem sucedido para a gerência, desenvolvedores e futuros usuários.

Reconhecidamente a fase estratégica do projeto, a fase de concepção define as linhas gerais do projeto. São realizados estudos para viabilização e planejamento do projeto, para que de uma maneira geral sejam estimados os prazos e custos, e captados os recursos necessários

(pessoal, equipamento, capital, etc...). A atividade monofásica de especificação conclue esta etapa, apresentando a especificação completa do software juntamente com o manual do usuário.

A fase de **desenvolvimento** tem como objetivo transformar a especificação em um sistema operacionalmente funcionando (programa). Neste momento do ciclo de vida, a equipe encarregada tem a missão de definir mais especificamente *como* o software será projetado, implementado e com seus componentes testados [Pres87].

Posteriormente ao desenvolvimento, quando o software é considerado pronto pelos seus desenvolvedores, surge a necessidade de transformar o software num produto destinado ao mercado. Nesta fase, denominada **preparação**, a atividade monofásica de empacotamento ativa procedimentos que controlam a geração do software para utilização no mercado (versões, mídias, documentação, tipo de licença), garante a instalação adequada na máquina do usuário e protege o software contra utilização indevida. O empacotamento, apesar de parecer ser restrito à tarefas administrativas, requer implementação de esquemas e rotinas, utilização de ferramentas e controle de qualidade. O capítulo 4 detalha estas questões, demonstrando a sua importância no processo de produção de software comercial.

Antes porém do produto ser comercializado, os produtores de software costumam submeter o software a apreciação dentro da própria empresa e no mercado consumidor em potencial, com o produto acabado. Estas atividades monofásicas, denominadas respectivamente de testes alfa e beta, são consideradas como pontos de decisão (*milestones*) no ciclo de vida do software comercial. A partir da aceitação interna e do mercado, o produtor decide por comercializar o software ou retornar a qualquer uma

das atividades monofásicas anteriores (*feedback*) para atender determinados requisitos que melhoram a aceitação do produto no mercado.

A fase de **disponibilização**, também referenciada na literatura como **manutenção**, se focaliza em suportar tecnicamente os produtos vendidos e gerenciar o processo de mudanças, fruto de erros detectados e alterações de requisitos de usuários ou do próprio ambiente. A disponibilização responde pela continuidade do ciclo para um produto de software existente, reiniciando o ciclo de produção com novos requisitos.

A disponibilização de um software no mercado, representa a maior parcela em custos e tempo durante sua existência, justificando um esforço da gerência em planejamento e controle. As necessidades detectadas pela comunidade usuária juntamente com decisões estratégicas de evolução do produto (mudança de plataforma, novas funcionalidades, novas tecnologias, etc.), devem impulsionar a sua evolução, visto que, o software para representar uma solução precisa atender eficientemente seus usuários e se antecipar a futuras necessidades (projeto defensivo).

Nos capítulos seguintes serão enfatizadas as fases de Preparação e Disponibilização, pelo fato de terem sido ainda pouco exploradas na literatura ao contrário das demais fases (concepção e desenvolvimento), apesar da sua reconhecida importância para a produção competitiva e qualitativa de software para o mercado. A discussão será enriquecida com exemplos ilustrativos da aplicação do molde à produção de software Unix de prateleira, servindo como base para o desenvolvimento de guias práticos para a produção de software deste segmento.

3.5 Atividades Polifásicas

Esta seção descreve as quatro atividades polifásicas, ou de acompanhamento, consideradas pelo molde proposto: Investimento, Documentação, Marketing e Qualidade. São discutidos os aspectos que fazem com que estas atividades sejam necessárias ao longo do processo de produção de software comercial.

Além da discussão que se segue, as atividades polifásicas serão referenciadas no restante da dissertação sempre que necessário, pois estão presentes em todo o processo de produção de software.

3.5.1 Investimento

A customização dos investimentos tem sido primordial para o mercado de produção de software, cada vez mais especializado e disputado. A atividade de investimento se mostra presente em todo o processo de produção, sendo um dos sustentáculos do processo desde a viabilização do projeto até sua comercialização e sustentação no mercado.

Um dos grandes problemas na gerência de software se caracteriza pela falta de controle dos investimentos, e vem se tornando cada vez mais crítico a medida em que passa a ser um dos fatores determinantes para o sucesso do projeto. Este fato fomentou inclusive a criação de modelos de processos que priorizam custos [Boeh88].

As empresas vivenciam a crise econômica instalada no país, tendo que se superar para aumentar a sua produtividade e a qualidade dos seus produtos. A escassez de recursos (dinheiro, pessoal, ferramentas, etc..) existe, mas o mercado de informática é dinâmico e exige soluções que atendam eficientemente aos requisitos de seus usuários. A saída para as empresas de software é procurar otimizar os recursos disponíveis.

A eficácia no planejamento dos custos de software reside basicamente na precisão das estimativas realizadas para o desenvolvimento e suporte do produto. A área de informática tradicionalmente não apresenta boas estimativas quando se trata de tempo e recursos humanos necessários para a geração de um produto, criando-se uma descrença nos profissionais da área e até inviabilizando produtos.

Vários estudos foram realizados no sentido de amenizar este quadro. Recorrendo-se à observação do comportamento de projetos e com o auxílio de métodos estatísticos foram montados alguns modelos preditivos de custos: Modelo Jensen [**Jens83**], Modelo COCOMO [**Boeh81**] e Modelo BOEING [**Blac77**].

Os modelos citados são resultados de um processo de análise contínua do próprio ambiente de desenvolvimento, onde se define um ciclo partindo-se de uma estimativa inicial, verificando-se a precisão da estimativa através da comparação de valores reais e previstos e refinando-se os parâmetros do ambiente para dar suporte a projeção de novos valores. Note que a estimativa é realizada apenas no primeiro ciclo, surgindo para os próximos ciclos o conceito de projeção empírica de custos como resultado das relações estatísticas verificadas nos dados de projetos anteriores [**Dema89**].

A grande maioria dos modelos desenvolvidos para predição de custos são dependentes de uma estimativa inicial do tamanho de código (KLOC : milhares de linhas de código fonte) para derivação do esforço (PM: homens-mês) necessário durante o desenvolvimento.

Os resultados obtidos com a utilização destes modelos devidamente calibrados para o ambiente, depois de aplicados em vários projetos, indicam uma correlação em torno de oitenta per cento com a realidade de

produção [Boeh84]. Entretanto, estes modelos consideram apenas o ciclo tradicional de desenvolvimento, desprivilegiando questões importantes como comercialização, suporte técnico e empacotamento de produtos, que representam grande fatia dos custos totais no ciclo de vida do software.

A preocupação com custos deve ser mantida não só pela otimização dos escassos recursos, mas também pela íntima relação com a qualidade do produto de software gerado com estes recursos. Pelo próprio exemplo da indústria vemos que os produtos mais bem manufaturados são decorrentes de uma política ofensiva de investimentos no produto e no controle de qualidade deste além da filosofia de qualidade implantado por toda a empresa [Falc90].

A compreensão e controle dos custos de software inevitavelmente nos leva a um bom nível de compreensão e controle dos aspectos de qualidade do software [Boeh88]. Provavelmente, aqueles que souberem melhor dosar o nível de investimentos, considerando o ciclo de vida real, com a qualidade do produto terá melhores chances de sucesso no mercado de software comercial.

3.5.2 Documentação

Em qualquer ambiente de produção de software, que trabalhe com produtos evolutivos, se faz necessário o registro dos procedimentos adotados durante o seu desenvolvimento e manutenção, para assegurar a sua longevidade e disseminação das suas propriedades entre usuários finais e profissionais envolvidos com a manutenção. A atividade polifásica de documentação dos produtos vem em direção das necessidades da Engenharia de Software de produção de softwares multi-pessoais e multi-

versões [Ghez91], contribuindo para a continuidade de um produto, independentemente do seu desenvolvedor e da atual equipe envolvida.

Uma documentação voltada para o usuário final deve tentar apresentar os objetivos do software e principalmente os procedimentos necessários para melhor utilização das suas características. Os problemas mais comuns são de confecção e interpretação dos manuais de usuários, basicamente porque quem os escreve não domina o produto ou simplesmente não atribui a clareza necessária. Por outro lado, os usuários não possuem o hábito de acessar esta documentação, que por vezes não soluciona objetivamente seus questionamentos.

A documentação do projeto possui importância a medida em que se queira obter formalização das atividades de desenvolvimento, o que facilita o processo de reutilização dos componentes do produto de software (Especificação, Projetos, Código, Massa de Teste, Bibliotecas, Dicionário de Dados, Formulários e Resultados dos Testes Alfa e Beta, dentre outros).

De uma forma geral, a documentação visa:

- registrar as atividades técnicas de concepção e desenvolvimento do software
- orientar o usuário (p.ex., manual de instalação, manual do usuário)
- controlar a operação da empresa (p.ex., contratos de negócios, resultados de investimentos, registro do software, direitos autorais)
- controlar a qualidade do software (p.ex., inspeções, resultados de testes)

- apoiar a área comercial (p.ex., descrição de produtos, perguntas & respostas de vendas, roteiro de apresentação do produto, resumo executivo, comparativo com concorrentes)
- amparar o suporte no fornecimento de notas técnicas para usuários, *workarounds*, dentre outros.

A obtenção de registros das atividades do processo de produção nos seus diversos estágios e disciplinas, forma a base para a implantação de melhorias e auxilia a gerência na estimativa de novos projetos, a partir do estabelecimento histórico do ambiente.

3.5.3 Marketing

Decididamente, os mecanismos de marketing tem assumido papel importante no cenário da informática mundial. Com a "coqueluche" dos microcomputadores e a massificação dos produtos de software nas organizações e para o público em geral, abriu-se um espaço para o sucesso do marketing direcionado para venda de tecnologia.

Diferentemente de se vender qualquer produto manufaturado como um sapato ou um carro, a venda de software requer uma divulgação apropriada para o mercado que estará adquirindo serviços automatizados que customizem suas atividades de trabalho. Então, o marketing profissional tem sido aplicado à informática para alcançar o seu mercado alvo, destacando-se os diferenciais técnicos ou promocionais do software.

A atividade polifásica de marketing tem sido levado tão a sério pelos grandes produtores, que o seu planejamento tem direcionado as ações de produção e comercialização dos produtos. Hoje em dia, não é plausível pensar em disponibilizar um produto para o mercado sem o apoio do

marketing profissional, cujos custos, dependendo do produto e mercado alvo, podem suplantar os custos de produção.

Assim sendo, o marketing vem utilizando basicamente duas estratégias para divulgação de software no mercado, que apesar de não serem as únicas, resumem a grande maioria das estratégias empregadas:

- "*Market Push*". O produtor vai fisicamente ao mercado demonstrar o potencial do produto, gerando a demanda. Estratégia muito utilizada para venda de produtos Unix no Brasil, pela limitação de mercado e concentração de grandes usuários corporativos.

- "*Market Pull*". O produtor atrai a demanda através de divulgação do seu produto, destacando suas potencialidades e diferenciais, para realizar a compra nos canais de distribuição estabelecidos para o produto. Podem ser utilizados para esta estratégia recursos de mala-direta, telemarketing, propaganda (jornais, revistas especializadas, televisão, etc...)

Entretanto, o marketing não se restringe às atividades de divulgação do produto para o seu público alvo. Seu domínio vai mais além, mostrando que esta atividade polifásica acompanha o processo de produção desde os seus primeiros passos.

O trabalho de marketing tem se mostrado útil na detecção de oportunidades para o desenvolvimento e/ou evolução de produtos e levantamento da viabilidade de mercado. Talvez mais importante do que saber como fazer o produto é saber se alguém quer consumi-lo e se esta demanda justifica os investimentos a serem realizados para a sua produção.

Após viabilizar a produção e estabelecer metas para o produto que será desenvolvido, o marketing deve preparar o mercado, gerando

expectativa de consumo para o produto. Uma vez que o produto esteja no mercado, o marketing deve monitorar a aceitação do produto e seus possíveis concorrentes, determinando as estratégias para sua sustentação e evolução.

Esta rápida descrição das responsabilidades do marketing voltado para o software nos dá uma idéia da sua necessidade e dos custos associados. São atividades contínuas que acompanham o produto antes e durante a sua existência, que efetivamente devem ser considerados nas planilhas de custo do produtor pois os mecanismos de marketing podem custar caro em função do tamanho do seu mercado alvo.

3.5.4 Controle da Qualidade de Software

O grande desafio para o segmento de software nesta década será conseguir produzir softwares de qualidade aceitável, conseguir mecanismos práticos para quantificar "qualidade" e orientar os fatores desejáveis de qualidade para o processo de desenvolvimento [Basi91].

Esta preocupação surge na medida em que os usuários cada vez mais procuram obter produtos confiáveis que melhor atendam as suas necessidades. A realidade dos softwares existentes no mercado é marcada por custos elevados de manutenção e correção, representando até oitenta por cento do custo total do sistema durante o seu ciclo de vida, derivados diretamente da baixa qualidade do produto liberado [Dema89].

Infelizmente não existe uma definição exata de qualidade de software, por ser um conceito multidimensional que envolve uma série de características definidas particularmente para cada ambiente. Além do mais, existem várias correntes de pensamento sobre qualidade, dentre as quais podemos destacar as seguintes definições :

"aptidão de um produto ou serviço a satisfazer as necessidades (expressas ou potenciais) dos usuários" **[Iso9000]**.

"atender fielmente às especificações" (Phillip Crosby) **[Cros79]**.

"implementar a tempo e dentro dos orçamentos previstos" (Juran) **[Jura79]**.

Evidentemente que nenhum destes conceitos é absoluto e exclusivo; são pontos de vista pessoais que refletem as preocupações específicas com qualidade. E o conceito de qualidade difere para cada indivíduo e ambiente, de acordo com valores pessoais, necessidades e disponibilidade de recursos, que não o torna universal.

Entretanto, o conceito de qualidade deve ser abrangente, de acordo com o seu domínio, devendo ser abordado basicamente sob dois pontos de vista : a qualidade do produto e a qualidade do processo de produção.

Quando se fala em qualidade do produto ou até mesmo em qualidade de software, a intenção é de prover os usuários com um produto que atenda as suas expectativas em termos de confiabilidade, usabilidade, manutenibilidade, eficiência, funcionalidade e portabilidade **[Iso9126]**, preço e data de disponibilidade do produto **[Basi91]**.

No conceito de qualidade do processo de produção, o software deve ser visto como uma sequência de refinamentos de forma que cada etapa do desenvolvimento forneça um produto intermediário com critérios específicos de qualidade a serem verificados. Segundo **[Falc90]**, a qualidade deve partir do indivíduo para o processo numa abordagem *bottom-up*, onde com o estabelecimento das rotinas, metas e parâmetros de controle, o indivíduo forneça um serviço de qualidade para o processo.

Esta corrente de serviços de qualidade, constrói etapas de produção qualitativas que fornecem um produto de qualidade.

A filosofia de controle de qualidade já se incorporou definitivamente na estrutura das indústrias, que devido a forte concorrência existente e ao desejo de produtos mais confiáveis, mantém grupos dedicados utilizando métodos rigorosos e reconhecidamente aceitos.

A questão do controle de qualidade do software, tanto do produto como do processo de produção, apresenta dificuldades bastante peculiares, pois o software não possui técnicas e métodos de medição (métricas) universalmente aceitas que permitam verificar o cumprimento das características desejáveis de qualidade de forma a atender as necessidades específicas de seus usuários, ficando, até então, a cargo de especialistas da área o julgamento da qualidade do software.

A utilização de métricas de qualidade tem se concentrado excessivamente no processo de codificação, considerando principalmente a complexidade do código produzido como indicador da dificuldade de manutenção, através das métricas como o número ciclômático de McCabe e a ciência de software de Halstead, avaliadas em [Pere91]. Outros trabalhos consideram a avaliação subjetiva da qualidade durante as diversas etapas do processo de produção [Roch92], partindo da utilização de modelos de processos clássicos.

Diante deste cenário, o molde proposto pretende trazer à tona, a busca contínua por qualidade, baseada em métricas referentes ao produto durante toda a existência do software. A definição de métricas efetivas para o produto e para o processo, e a utilização de conceitos de gestão de qualidade por todos os componentes da empresa serão decisivos para a aceitação de produtos no mercado de software.

Conclue-se assim, a descrição do molde proposto para a produção de software comercial.

O restante da dissertação constitui a parte II, que ilustra parcialmente a aplicação do molde proposto para a produção de software Unix de prateleira. São ilustradas as fases de preparação (capítulos 4 e 5) e disponibilização de software comercial (capítulos 6, 7 e 8).

Parte I

MODELOS DE PROCESSOS DE PRODUÇÃO DE SOFTWARE

Um Molde Genérico mais Realista para Produção e Disponibilização de Software Comercial

Capítulo 4. Empacotamento

O termo empacotamento, muito utilizado no mercado de software, diz respeito às atividades necessárias para transformar o software liberado pelo desenvolvimento em produto comercializável. Esta atividade monofásica, ligada à fase de preparação do software, envolve atividades técnicas e gerenciais que serão expostas neste capítulo.

O capítulo destaca inicialmente as precauções que o produtor deve tomar com a instalação do software na máquina do usuário. A descrição destes procedimentos visa alertar os produtores para uma série de atividades técnicas que devem ser preparadas para garantir a inserção do software adquirido de acordo com a configuração da máquina do usuário.

Logo em seguida são explorados os aspectos legais de proteção da propriedade intelectual da indústria de informática: o software. São discutidas as estratégias de segredo de negócio, direitos autorais e patentes. A discussão da proteção do software é enriquecida com a

ilustração de alguns esquemas técnicos de proteção utilizados por produtores de software Unix de prateleira.

O capítulo fecha com uma rápida abordagem dos procedimentos necessários para a reprodução correta do software para a comercialização no mercado. São enfatizadas algumas preocupações com o controle de qualidade da reprodução do produto.

Como será visto, qualquer produtor que deseja colocar seu software de prateleira à disposição do mercado deve considerar os aspectos de empacotamento no planejamento de recursos para a produção do software.

4.1 Instalação

Tudo começa na instalação. O produtor deve assegurar que o software seja devidamente introduzido no novo ambiente de trabalho (a máquina UNIX do usuário).

A instalação de um software, em linhas gerais, objetiva inserir fisicamente o produto na máquina do usuário, configurando-o para ser executado eficientemente e, quando necessário, alterando a configuração do próprio ambiente para possibilitar seu uso. Nesta seção serão discutidas estas características de instalação de software.

Os fabricantes de software para UNIX vem acumulando problemas de instalação de software, devido principalmente a diversidade de plataformas de hardware e periféricos envolvidos, gerando as mais diversas configurações possíveis.

Uma das saídas utilizada tem sido a adoção de instaladores genéricos para adequar o produto às diversas configurações dos seus usuários. Dentre os instaladores genéricos, podemos citar o **customs** do

XENIX (R) e o Ic da Infocon (R), que servem como parâmetro de instalação no mundo UNIX. Por questões de facilidade de acesso, tomamos como base o instalador Ic da Infocon para ilustrar a discussão, o qual não difere substancialmente do instalador Customs.

O Ic supervisiona toda a interação com o usuário para garantir que sejam seguidos exatamente os procedimentos necessários de instalação. O usuário interage com a rotina de instalação a partir dos seguintes passos:

1. Verifica o espaço em disco disponível na máquina do usuário. Exibe informações sobre alocação de espaço necessário para execução do software completo e de cada módulo funcional individualmente ;
2. Solicita que o usuário escolha os recursos ou módulos do software que irão estar disponíveis para uso ;
3. Solicita o diretório para residência do software ;
4. Checa a existência do diretório e de alguma versão do software na máquina do usuário e, se for o caso, oferece ao usuário a possibilidade de instalação preservando a versão anterior ;
5. Gerencia a sequência dos disquetes de instalação inseridos pelo usuário;
6. Informa constantemente ao usuário sobre o que está sendo realizado, eventualmente solicitando alguma intervenção ;
7. Identifica a legitimidade do software a ser instalado com a checagem do seu número de série e chave de ativação (mais detalhado na seção 4.2.2).

Os passos para instalação do produto na máquina-alvo são devidamente ilustrados na documentação de instalação que geralmente o usuário recebe junto com o pacote de software, ou vem incluído no próprio manual do produto. O Manual de Instalação, assim conhecido, deve ser um guia que conduza o usuário à perfeita instalação do produto.

Contudo, a transparência da instalação pode custar caro para algum produtor desavisado, que porventura esteja iniciando sua produção de software para o segmento UNIX sem considerar estes aspectos no seu planejamento. A complexidade do instalador está embutida na possibilidade de executar funções do tipo: reinstalação completa, desinstalação, verificação da integridade da instalação, mudança de versão (upgrade) e reativação do produto através de chaves de ativação.

Observando o funcionamento do `lc` mais cuidadosamente, verifica-se que ele é um roteiro (script) escrito numa sequência de comandos básicos do Shell do UNIX. Basicamente, este roteiro é colocado sempre no primeiro disco de instalação e se encarrega de executar em memória os sete procedimentos padrões de instalação mencionados nesta seção.

Como ilustra a figura 4.1, o `lc` necessita basicamente de dois arquivos texto contendo respectivamente a especificação do software a ser instalado e as suas particularidades de instalação.

O instalador interage com o usuário para executar os procedimentos iniciais (procedimentos 1 a 4 descritos nesta seção) para então acessar o arquivo de especificação do produto para efetivar a instalação do software.

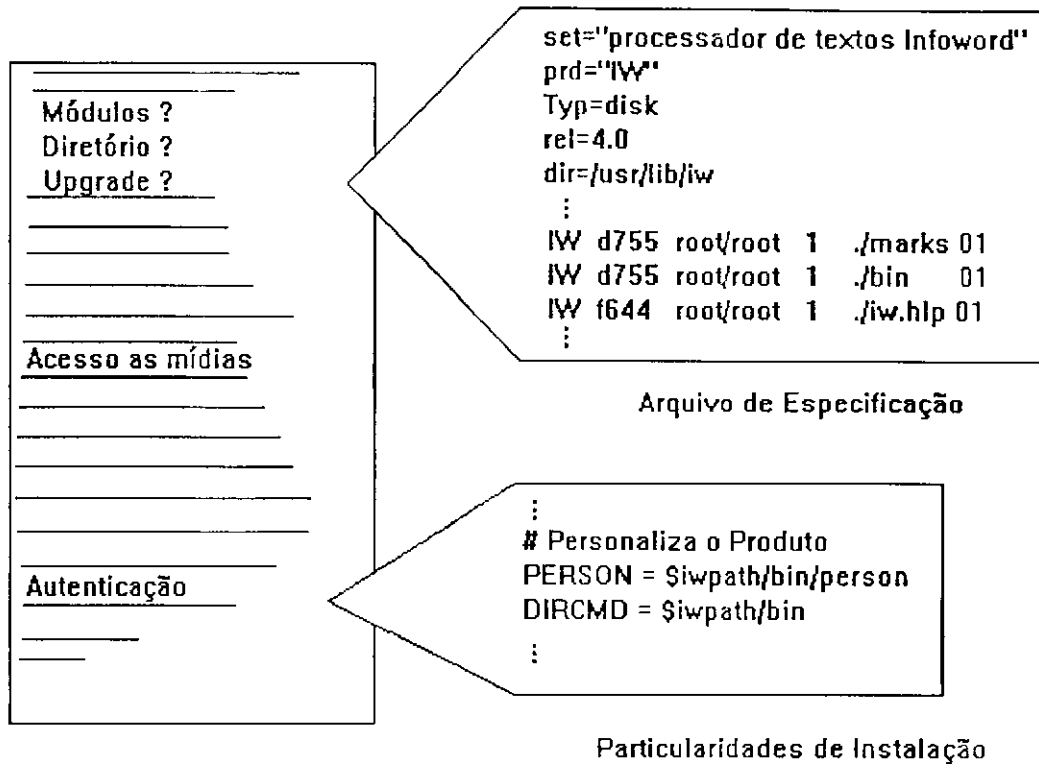


Figura 4.1 - Visão Funcional do Instalador Ic da Infocon

Um exemplo completo do arquivo de especificação se encontra no apêndice A, o qual contém basicamente:

- Permissões de arquivos ;
- Organização de diretórios para funcionamento do sistema ;
- Identificação de códigos executáveis e não-executáveis ;
- Identificação das bibliotecas do produto.

A instalação é concluída quando o Ic analisa o segundo arquivo texto, o qual contém as particularidades necessárias para o término da instalação do software. Como pode ser observado também no apêndice A, este arquivo especial contém procedimentos para checar a chave de ativação, cadastrar usuários no aplicativo, inicializar variáveis de ambiente,

executar a instalação manualmente sem o auxílio do Ic, dentre outras funções.

A instalação apesar de ser um processo único na vida do software na empresa (pelo menos é o que se espera), nem sempre agrada o usuário final principalmente pela dificuldade de interação, forçando-o a tomar decisões para as quais ele nem sempre está preparado. Para minimizar atropelos, o produtor deve atentar para as características desejáveis de uma instalação de qualidade :

- Auto-explicação: garantia de que o usuário acompanhe corretamente o processo de instalação sem, eventualmente, consultar o seu manual de instalação ;
- Transparência: os procedimentos de instalação não devem deixar dúvidas quanto a sua execução ;
- Preservação da configuração do usuário ;
- Conservação dos dados dos usuários. Principalmente quando se estiver migrando para uma nova versão do software ;
- Acompanhamento contextual : posiciona o leitor sobre o que a rotina de instalação está executando, ou que atitude (resposta, seleção funcional do software, etc.) o usuário deve tomar.

No planejamento de um projeto para UNIX, os aspectos envolvidos com instalação de software são geralmente esquecidos. As discussões se concentram normalmente, em prazos e custos envolvidos na especificação, projeto e desenvolvimento, e talvez nesta última atividade, surja alguma preocupação com testes. Deve ser lembrado que para uma instalação adequada, o produtor deve cuidar do desenvolvimento de rotinas

apropriadas, testes específicos, preparação de arquivos de especificação da instalação do produto, descrições de terminais e outras atividades inerentes.

Para se ter uma idéia da dimensão do trabalho de instalação, o Ic da Infocon possui atualmente 1218 linhas de comandos do Shell padrão do Unix, desenvolvidas em três meses de trabalho ininterruptos por dois profissionais da área, e os arquivos de especificação da instalação de cada produto e de suas particularidades contém em média 150 linhas de arquivo texto (como pode ser visto no apêndice A).

4.2 Proteção ao Software

Quando o assunto proteção vem à tona, seja lá qual for a sua aplicação, imagina-se logo que alguém está se preocupando em criar procedimentos que garantam a não violação dos seus direitos. Por exemplo, a Consituição Nacional foi reformulada no desejo de proteger os Brasileiros de uma injustiça social, assegurando-lhes os seus direitos e instituindo seus deveres. Como ela realmente funciona é uma questão de interpretação jurídica e ajuste das suas leis às necessidades da população.

Considerando nosso objeto de estudo, o aspecto proteção assume uma caráter fundamental para assegurar direitos aos produtores de software. A facilidade de reprodução de software por qualquer mídia magnética, e sua divulgação no mercado de massa, facilita e, por vezes, estimula a reprodução não autorizada.

A atividade de produção de software para ambiente UNIX requer meses de trabalho a fio e um esforço empresarial considerável para disponibilizá-lo. Isto se deve, primordialmente, à diversidade de

plataformas e periféricos existentes (portabilidade), uma das principais características deste sistema operacional.

Investimentos consideráveis são realizados até a disponibilização do software no mercado, com a devida estrutura de suporte e evolução do produto. Quando uma empresa chega a este estágio, o mínimo que se deseja é a garantia dos direitos sobre o trabalho realizado, para que funcionários, empresários e acionistas da empresa obtenham o retorno adequado dos investimentos realizados.

4.2.1 Aspectos Legais

Por se tratar de uma área relativamente nova e bastante dinâmica, a proteção jurídica do software ainda está em processo de amadurecimento. Apesar de se basear em leis antigas e vulneráveis, que estão sendo constantemente avaliadas e reformuladas, o produtor pode garantir legalmente a proteção da propriedade do software.

Não se pode afirmar categoricamente que existe uma estratégia única, mais adequada para determinado tipo de software. O que se verifica na prática, são um conjunto de doutrinas e mecanismos legais, que combinados fornecem maior grau de segurança em função do tipo de programa e da abrangência do código produzido.

Na sequência desta seção, serão apresentadas estratégias legais de proteção, buscando sugerir aonde devem ser aplicadas e sobre que condições. Sempre que necessário são indicadas referências bibliográficas que tratam da questão com maior profundidade, principalmente no que diz respeito a questões puramente jurídicas, o que não é alvo de nossa apreciação.

4.2.1.1 Segredos de Negócio

No mundo empresarial, o dizer popular " o segredo é a alma do negócio" ainda prevalece. Esta frase caracteriza bem a necessidade existente nas empresas em proteger suas informações consideradas cruciais para garantir alguma vantagem competitiva. Informações estas do tipo: receitas, fórmulas, metodologias, técnicas e lista de clientes, consideradas de extremo sigilo pelas empresas, são conhecidas como segredo de negócio.

Se uma empresa é proprietária de algum segredo de negócio, este é protegido pela lei do segredo de negócio. Lei esta responsável pela regulamentação de procedimentos que identificam, julgam e asseguram os direitos às empresas ditas proprietárias de algum segredo de negócio.

Cada vez mais, empresas de base tecnológica se preocupam em aplicar a lei do segredo de negócio, principalmente por serem elementos ainda desconhecidos por uma indústria dinâmica. Produtores de software atentos para esta possibilidade, utilizam este recurso principalmente para proteção de :

- Código fonte e objeto, listagem de programas ;
- Técnicas de programação ;
- Ferramentas de desenvolvimento ;
- Ferramentas de gerência ;
- Técnicas para solução de problemas ;
- Bibliotecas padrão ;
- Rotinas de suporte ;

- Lista de Clientes da empresa ;
- Informações estratégicas da empresa ;

Um segredo de negócio se configura quando determinada informação, processo ou idéia não é conhecida pela indústria de software de uma maneira geral, e a sua confidencialidade lhe assegura competitividade na área de atuação. Um outro detalhe pertinente é que a informação, processo ou idéia não pode ser de domínio público, tais como leis, padrões industriais e algoritmos publicados.

Evidentemente que para um segredo de negócio manter sua natureza secreta, devem ser estabelecidas uma série de medidas de precaução. Medidas estas que continuamente protejam o objeto do segredo de negócio de qualquer possibilidade de violação por parte de qualquer um que a ele tenha acesso. Isto é definitivo para a confidencialidade, pois qualquer divulgação não controlada do segredo de negócio, elimina os seus direitos sobre ele.

Dentre as medidas preventivas, uma das mais importantes é a elaboração de um acordo de não-divulgação por aqueles que venham a ter acesso ao segredo de negócio. Muito utilizado para funcionários, tanto na admissão quanto na saída da empresa, este acordo pode ser estendido para fornecedores, editores, pessoal de marketing, usuários em teste beta e os próprios executivos da empresa. Em [Reme87], encontra-se um guia completo sobre como definir e aplicar estes acordos, além de medidas cautelares para manutenção do segredo de negócio.

Apesar de não ter limite de duração e ser bastante efetivo, quando se tomam as devidas precauções, o segredo de negócio apresenta as seguintes contrapartidas :

- O segredo de negócio pode ser legalmente adquirido por engenharia reversa, eliminando as possibilidades de reclamação por parte do proprietário. A decompilação, remapeamento do código objeto para o código fonte, e a execução de um projeto a partir de um software pronto, são permitidos.
- A descoberta independente de um segredo de negócio existente, devidamente comprovada, automaticamente o anula, tornando-o público.
- A administração das medidas de precaução é trabalhosa e cara. Por isso deve-se analisar muito bem o que é realmente segredo de negócio.

4.2.1.2 Direitos Autorais

Os chamados direitos autorais, ou direitos autorais de cópia, representam um instrumento legal de proteção de software, amplamente difundido e utilizado pela sua indústria. Este instrumento, na verdade, protege a forma pela qual uma idéia foi expressa pelo seu autor (empresa ou indivíduo), não sendo aplicável para proteção da idéia propriamente dita.

Analisando objetivamente, a maneira pela qual o código foi escrito pode ter seus direitos autorais protegidos¹, em oposição a abordagem utilizada para construí-lo. De fato, os direitos autorais não cobrem qualquer metodologia, princípio, técnica, conceito, procedimento ou descoberta [Alte93].

¹Neste trabalho são referenciadas as leis de direitos autorais americanas, por terem sido, até certo ponto, reconhecidas pela Lei do Software, promulgada em 1988.

Considerando-se, por exemplo, que uma empresa desenvolveu o primeiro editor de texto para UNIX. Esta empresa pode solicitar seus direitos autorais sobre o software em si, fato este que não impede que um concorrente desenvolva outro editor de texto também para UNIX. Os direitos autorais preservam a forma pela qual a empresa desenvolveu o editor, ou seja, suas rotinas e estrutura geral do projeto, não protegendo a idéia de concepção de um editor de texto (ver patente na próxima seção).

A seguir são listados os tipos de software que tem devida proteção pela lei dos direitos autorais :

- Código Fonte e Código Objeto ;
- Sistemas Operacionais ;
- Telas de jogos de computadores ;
- Artes geradas por computador ;
- Bases de dados eletrônicas ;
- ROMS.

Não possuem ainda uma posição definitiva quanto a sua instituição, tendo decisões isoladas tomadas em diversos tribunais :

- Microcódigo ;
- Formato de telas de computador do tipo "look-and-feel" ².

Para que um software possa ter seus direitos autorais protegidos, antes de tudo deve ser comprovadamente original. Pode-se inferir que pela

²A Lotus Development Corporation, recentemente, ganhou uma causa de direitos autorais contra a Paperback Software Int'l, autora da planilha VP Planner, considerada uma imitação da linguagem de menu utilizada na planilha I-2-3. Segundo o tribunal, a linguagem de menu criada pela Lotus se constitui na essência do valor comercial e intelectual do produto.

sua própria natureza, quase todo software, como atividade de criação humana, possui elementos de criatividade, sendo passível de proteção por direitos autorais. Apesar da subjetividade do julgamento, nem todo software possui características originais. Aqueles que se valem de recursos largamente utilizados pela comunidade, e demasiadamente simples, não são aprovados.

Ao serem atribuídos, os direitos autorais fornecem ao seu proprietário, o direito exclusivo de cópia, distribuição, alteração a partir do modelo original e execução da forma de expressão.

O processo de comprovação da violação dos direitos autorais pode ser uma tarefa não muito elementar. A violação ou plágio, por assim dizer, fica caracterizada quando se constata que terceiros tiveram acesso a expressões protegidas, na maioria dos casos ao seu código fonte. Expressões desprotegidas consideradas nos tribunais competentes, são aquelas que mencionam idéias abstratas, expressões de domínio público e padrões industriais [Alte93].

As maneiras mais usuais de se constatar violação de direitos autorais têm sido através da constatação de erros similares, verificação de respostas do software a determinadas ações e análise detalhada da estrutura global do software (gerência de memória, estrutura de arquivos, sistemas de menus, modularização de programas, código fonte, etc...)

A lei dos direitos autorais foi criada no sentido de garantir o direito da propriedade intelectual dos seus criadores, e para tal prevê as seguintes penalizações para aqueles que confrontem com suas alíneas :

- Suspensão imediata do ato ilegal ;
- Penas criminais quando a violação for considerada intencional ;
- Ressarcimento de danos e lucros reais (uma faixa de quinhentos a vinte mil dólares por direito infringido) ;
- Pagamento de todos os gastos judiciais .

4.2.1.3 Patentes

A demarcação de uma patente, geralmente se associa a algum novo invento, cujo autor tem intenção de exercer monopólio sobre a idéia que acredita perdurar por muito tempo, e conseqüentemente lhe trará vantagens durante sua existência.

Diferenciadamente dos segredos de negócio e dos direitos autorais, a lei da patente protege a idéia fundamental do novo invento ou concepção, permitindo que esta seja expressa e conhecida publicamente. Aqueles que de alguma forma manifestarem o conhecimento e utilização da patente, não serão considerados infratores, mas terão que pagar os direitos ao dono da patente **[Alte93]**.

A patente ainda não é amplamente utilizada para software, por ter um processo de registro moroso (dura aproximadamente dois anos), e dispendioso (com um ônus que varia de setenta a duzentos e cinquenta mil dólares) . Entretanto, o principal problema está no processo de verificação das exigências da lei da patente e da análise dos prováveis benefícios.

Uma análise inicial deve ser promovida para verificar se realmente o software se configura como um objeto factível de ser patenteado. Este análise tem sido foco de discussão em diversos tribunais americanos, que

se veem com diversas solicitações de patente, já sendo constatados vários casos de aceitação³. As solicitações para patente de software, são amplamente discutidos em função dos seguintes requisitos exigidos na lei das patentes :

- Novidade : Análise subjetiva que prima pela identificação de elementos que diferenciam o software a ser patenteado, dos programas já conhecidos. A patente não pode exigir monopólio de um algoritmo ;
- Não pode ser óbvio: Questionamento da complexidade e habilidade necessária envolvidos na descoberta do software ;
- Não se utiliza de algoritmos matemáticos.

Quanto aos direitos resultantes de uma patente de software, não se pode questioná-los, são bastante rígidos. Estes asseguram tranquilidade aos seus proprietários que podem se beneficiar com :

- Suspensão imediata do uso indevido por terceiros ;
- Recompensa financeira por danos eventuais. Que podem ser caracterizados como perda de lucros ou pagamento de direitos de patentes (royalties) ;
- Reembolso numa ordem de trezentos por cento dos danos reais, quando comprovada a intencionalidade do uso sem autorização.

A crucialidade da questão da patente para software reside no impacto que o objeto da patente terá na indústria de informática nos

³A Microsoft Corporation possui atualmente dezesseis software patenteados com mais alguns pedidos de patente em julgamento.

próximos dezessete anos, prazo de sua validade. Vale ressaltar que, neste prazo, o proprietário da patente tem todos os direitos de uso e comercialização, não podendo ser desapropriado, mesmo que seja por engenharia reversa ou descoberta por terceiros.

A concessão de patentes pelos órgãos responsáveis, tem levado em conta o relacionamento do software com o hardware. Basicamente, tem-se interesse em observar os impactos dos comandos e funções do software nos dispositivos físicos do tipo processador, barramento de dados, memória, placa de comunicação e periféricos em geral.

Levando-se em conta os aspectos incutidos na lei da patente, pode-se chegar a conclusão que sua adoção pode ser de extrema utilidade, dependendo da natureza do programa e de suas características (novidade e não-obviedade). Apesar da dificuldade existente para obtenção da patente, este mecanismo de proteção se adequa a sistemas não-comerciais que interagem fortemente com o hardware e, principalmente, com perspectiva de utilização em larga escala num médio prazo.

Alguns candidatos imediatos podem ser os sistemas operacionais, linguagens de programação, compiladores, sistemas de controle de produção, sistemas de gerenciamento de redes, sistemas operacionais de redes, dentre outros.

4.2.2 Aspectos Técnicos

A utilização ilegal de software é provavelmente um dos problemas mais sérios existente no mercado, e vem sendo combatido veementemente no mundo inteiro. O tratamento deste problema a nível jurídico vem evoluindo e ganhando espaço, como discutido anteriormente neste capítulo

(seção 4.2.1). Esta seção expõe alguns esquemas técnicos utilizados para proteger o software.

O mundo UNIX por ser caracterizado majoritariamente por máquinas multi-usuárias instaladas em ambientes corporativos, tem uma menor incidência de problemas de violação de software do que as evidências dos softwares para o mercado de massa. Depois de anos de experiência, os usuários corporativos ganham a consciência de que os produtos de software devem ser produtos adquiridos legalmente, uma vez que necessitam do apoio técnico dos seus produtores e obviamente não desejam problemas judiciais .

Do outro lado, o produtor de software só concede suporte técnico, notificações sobre evolução do produto e todas as vantagens associadas, aos usuários registrados oficialmente. O registro não é burocrático, e acontece instantaneamente quando o usuário envia o cartão de licenciamento para o produtor de software, contendo seus dados pessoais e identificação única do software. Uma ilustração de um cartão de licenciamento se encontra no apêndice B.

Apesar da consciência formada sobre as perdas com a violação de software, o produtor deve garantir que o seu software seja realmente instalado e utilizado conforme o acordo de licença de uso vendida para o usuário. Para tal, esta seção ilustra alguns esquemas técnicos e gerenciais utilizados por dois produtores de software Unix de prateleira (Lotus e Infocon) para disciplinar o uso do software (licenciamento) e assegurar sua instalação na máquina do usuário licenciado (proteção).

4.2.2.1 Esquema INFOCON (R)

O software produzido pela Infocon, ao estar liberado para ser comercializado, passa por um processo de geração de volumes, onde estão embutidos os procedimentos de segurança contra reprodução não autorizada.

A empresa se vale de um esquema simples e eficiente (ilustrado na figura 4.2), onde basicamente se garante que o software só possa ser instalado pelo seu proprietário, já que este possui a chave de ativação do software e seu número de série, informações que acompanham o produto. Isto é o bastante para colocar o software em produção.

A chave de ativação é gerada por uma rotina interna da empresa que calcula o seu valor pela função F_{chave} :

$$F_{\text{chave}} = F(\text{Número de série, número de usuários e tipo de uso})$$

onde "tipo de uso" pode significar licença de uso, cópia para homologação, teste alfa e beta ou ainda cópia demonstrativa. Algumas destas licenças podem restringir o acesso ao software até uma determinada data.

O valor da chave de ativação é criptografado e armazenado pelo personalizador num banco de licenças para ser checado com a chave de ativação informada pelo usuário quando estiver instalando ou acessando o produto.

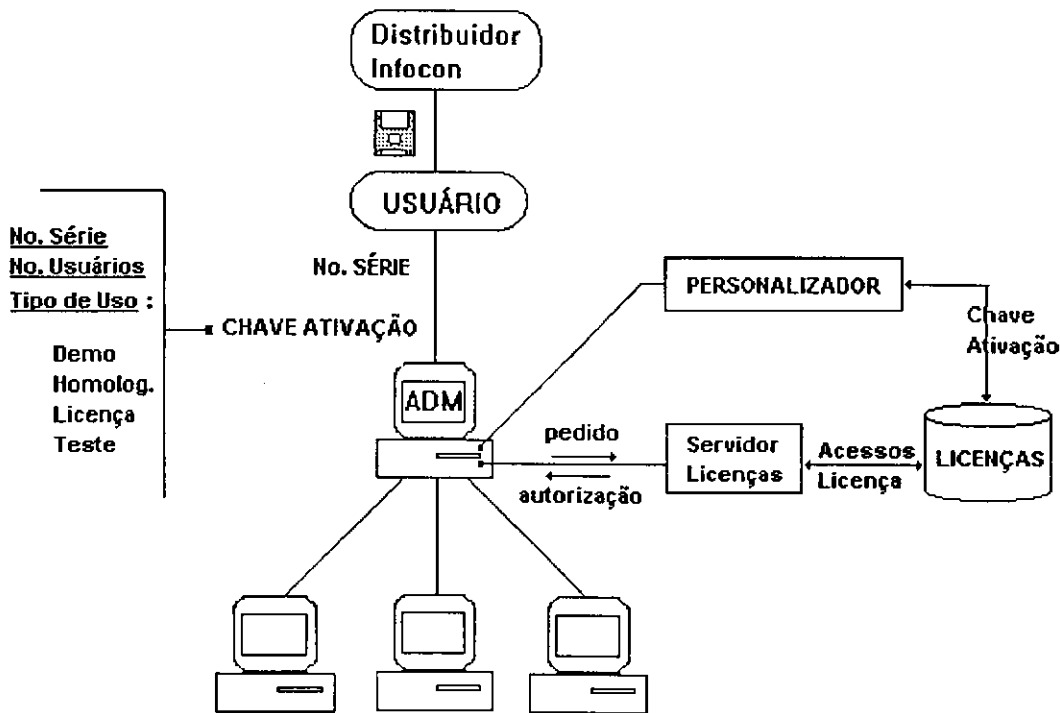


Figura 4.2 - Esquema de Proteção e Licenciamento da Infocon

Esquemas de proteções existem em função da filosofia sobre a qual eles são implementados. A empresa Infocon optou por colocar a responsabilidade de propriedade do software sobre o usuário. Conscientemente, a empresa se preveniu contra cópias realizadas por terceiros que não tenham acesso ao número de série, muito menos a chave de ativação. A idéia é de que quem adquiriu um software justamente para não ficar ilegal, não incentivaria este tipo de atitude ilícita, visto que poderia ser penalizado por isso.

Além dos problemas de proteção na instalação do software, o produtor deve assegurar o acesso ao software conforme as regras de utilização previstas no acordo de licença de uso vendido. Para isto, o produtor deve implementar ou adquirir o serviço de licenciamento de software. O servidor de licenças ("license servers") que é próprio para ambientes multiusuários, normalmente se baseia na arquitetura cliente-

servidor e oferece uma interface para a aplicação solicitar o controle de acessos ao software.

A Infocon implementou um servidor de licenças próprio para os seus produtos. Basicamente, o servidor de licenças da Infocon se preocupa em controlar o número de usuários que acessam o produto concorrentemente e a data de validade de um produto em homologação ou testes.

O funcionamento do servidor de licença da Infocon pode ser simplificado demonstrado nas seguintes operações, conforme ilustrado na figura 4.2 :

1. A aplicação solicita ao servidor de licenças uma licença para utilizar a aplicação.
2. O servidor de licenças verifica no banco de licenças a permissão de uso (número de usuários ou data de expiração) de acordo com a sua chave de ativação e a disponibilidade de licenças de uso no momento.
3. Caso o tipo de licença de uso permita o acesso ao software e o número de usuários acessando a aplicação no momento for menor do que o número máximo de acessos concorrentes permitidos, então o servidor autoriza o acesso e incrementa um contador de usuários.
4. A aplicação dá continuidade à sua execução após receber a autorização. Caso esta seja negativa, a aplicação bloqueia o acesso ao usuário temporariamente.
5. Ao finalizar a aplicação do usuário, o servidor recebe uma sinalização para decrementar o contador de usuários.

4.2.2.2 Esquema LOTUS (R)

A estratégia de proteção contra cópias não autorizadas, realizada pela Lotus Development Corporation para seus produtos UNIX, consiste basicamente na averiguação da legitimidade da cópia a ser instalada na máquina do usuário.

A Lotus montou uma estrutura que, para se liberar o software em sua plenitude, o usuário deve se comunicar com a empresa, ou com seu distribuidor, para obter uma chave que libera o sistema para ser usado por seus clientes (terminais). Esta chave, denominada chave de ativação, é fornecida pela Lotus, em função dos parâmetros de instalação, os quais dependem da identificação da máquina Unix do usuário.

Todo o processo de segurança, ilustrado na figura 4.3, se inicia com o recebimento do pacote de software, composto pela mídia (disquetes), documentação completa de instalação e operação, e pelo número de série do software adquirido. Cada usuário com direito de acesso ao produto, recebe em anexo um conjunto simplificado de manuais contendo um número de série que os identifica.

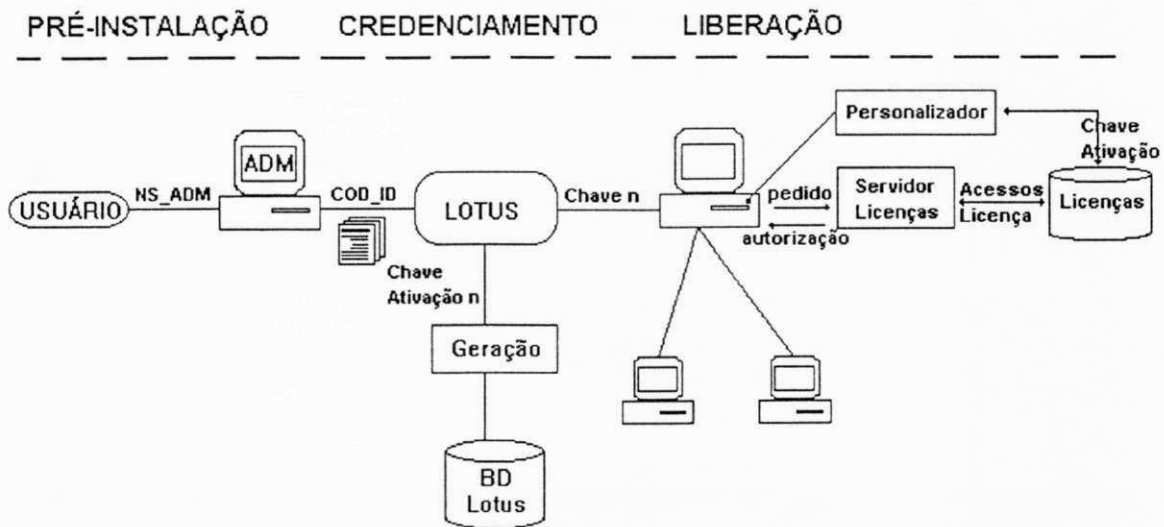


Figura 4.3. - Esquema de Proteção e Licenciamento da Lotus

A instalação do software na máquina do usuário Lotus, se realiza em três etapas distintas. Num primeiro momento, o administrador do sistema, possivelmente localizado no servidor, caso esteja ligado em rede, executa uma rotina de pré-instalação do software. Esta rotina libera o software para funcionar em apenas um terminal, e fornece um código de identificação do software para a máquina servidora UNIX.

De posse do código de identificação do software, dos números de série tanto do volume principal quanto dos volumes destinados aos usuários terminais, o usuário ou o distribuidor os enviam para a Lotus juntamente com seus dados pessoais. Um exemplo deste formulário contendo os dados necessários se encontram no apêndice C.

Para concluir esta segunda etapa, a Lotus executa uma rotina que a partir destas informações gera uma chave de ativação para o software. A chave de ativação libera o acesso simultâneo ao software pelos usuários com permissão, e, que num mais alto nível, pode ser expressa como :

$\text{Chave} = F(\text{Cod_ID}, \text{NS_Adm}, \text{NS_Usr1}, \text{NS_Usr2}, \dots, \text{NS_Usrn})$
--

Chave - Chave de Ativação ;

Cod_Id - Código de Identificação do Software na Máquina instalada

NS_Adm - Número de Série do Administrador do Software ;

NS_Usr1 - Número de Série do Usuário 1 ;

NS_Usr2 - Número de Série do Usuário 2 ;

NS_Usrn - Número de Série do Último Usuário Habilitado.

Fechando o ciclo de proteção do software Lotus, o usuário aplica a chave de ativação ao seu software pré-instalado, liberando-o definitivamente para produção.

Processo considerado nem sempre muito confortável para o usuário, que se vê obrigado a interagir com o produtor de software para solicitar acesso a algo que já foi adquirido legalmente⁴. Em contrapartida, o usuário toma conhecimento de que existe uma infra-estrutura bem montada na qual ele pode se apoiar quando for preciso. Durante o processo de solicitação da chave de ativação, o usuário fica cadastrado como usuário Lotus, o que o credencia para obter todos os benefícios de garantia, suporte técnico e notificações sobre evoluções de produtos, além de ser consultado para possíveis melhorias no produto.

Esta estratégia de interação com o usuário traz benefícios diretos para os produtores de software. Além de se prevenirem contra a cópia ilegal de seus produtos, os produtores obtêm informações precisas sobre a massa crítica da sua comunidade usuária, podendo atuar mais objetivamente sobre esta.

Com relação aos problemas de controle de acessos ao software conforme o acordo de licença de uso (licenciamento), a Lotus utiliza um servidor de licenças adquirido no mercado e bastante poderoso, o qual gerencia licenças em redes heterogêneas, baseado no modelo cliente-servidor.

Para se ter uma idéia do impacto desta atividade no planejamento da produção de um software para o mercado Unix, o servidor de licenças que a Lotus utiliza custa aproximadamente US\$ 14,000 para a primeira

⁴ Situação mais agravante para o usuário brasileiro que tem que solicitar a chave de ativação para o escritório da Lotus na Irlanda (48 horas via FAX)

plataforma, sendo reduzido a cada plataforma adicional. No caso da Infocon que optou por desenvolver um servidor de licenças simples para ambientes multiusuários, foram gastos em torno de 30 dias na concepção, desenvolvimento, testes e liberação para o mercado de 600 linhas de código fonte com a dedicação exclusiva de um profissional.

4.3 Reprodução

A reprodução do software para o usuário deve ser executada em momentos distintos do ciclo de produção, devendo ser exercido um rigoroso controle de qualidade para assegurar o fornecimento correto do produto a ser enviado.

O fabricante deve construir mecanismos que controlem a reprodução do software em função do seu momento no ciclo de produção e da necessidade de seus usuários. Mais especificamente, o fabricante exerce a atividade de reprodução em função das estratégias de vendas e de divulgação do produto no mercado. Por isto, a reprodução recebe tratamento diferenciado para cada uma das situações abaixo :

- . Cópia para Homologação
- . Licença de Uso em função do número de usuários
- . Cópia Demonstrativa
- . "Site license"
- . "Upgrade"

Em cada situação das citadas acima, existem características de reprodução diferentes em termos do conteúdo do material a ser enviado ao usuário final, que serão observadas isoladamente em seguida.

Entretanto, existem algumas considerações que podem ser observadas para melhorar a qualidade da reprodução de software, independentemente do seu objetivo. São listados em seguida, os fatores que devem ser considerados na reprodução de software, segundo as normas ISO 9001 para software :

1. Número de cópias para cada software a ser liberado ;
2. Tipo de mídia para cada item de software, incluindo o seu formato, versão e forma de leitura humana ;
3. Definição da documentação necessária (Manuais de usuário, manuais de treinamento, manuais de instalação,...) ;
4. Explicitação dos domínios da licença do software e dos direitos autorais devidamente endereçado e acordado ;
5. Custódia do disco mestre e possíveis backups quando aplicável, incluindo planos de recuperação de desastres ;
6. Período de garantia de fornecimento de cópias ;

Quando o produto atinge o estágio de reprodução para envio ao mercado, o fabricante necessita controlar a qualidade do processo para garantir o processo de geração de volumes em função dos pedidos de aquisição, seja do usuário, da revenda ou do distribuidor do software.

O controle de qualidade consiste na eliminação de eventuais problemas da reprodução de software, garantindo que o pacote de software seja repassado ao usuário com todos os seus componentes consistentes. Isto envolve basicamente a verificação da mídia gravada, e a documentação enviada em função do pedido solicitado.

Normalmente se espera que o controle de qualidade efetuado para liberação do produto para reprodução tenha sido efetivado a contento para evitar inconvenientes para o usuário. Toda versão do produto deve ser exaustivamente testada, inclusive os procedimentos de reprodução para as diferentes mídias e a instalação do produto na máquina do usuário, para gerar uma versão mestre que servirá de base para reprodução.

Capítulo 5. Testes de Aceitação

A prática de mercado tem incluído no caminho do software, alguns conceitos que são particulares ao desenvolvimento de aplicativos para o mercado. Dentre estes conceitos, encontram-se os testes de aceitação, que podem ser considerados como atividades monofásicas para adequação do produto ao uso e melhoria no seu nível de qualidade antes do lançamento no mercado.

Nem sempre largamente utilizados pelos produtores de aplicativos, por questões de prazo para lançamento do produto, pela falta de hábito ou por puro desconhecimento, os testes de aceitação geralmente são realizados em dois níveis: o teste de aceitação interna, realizado no próprio ambiente de desenvolvimento, chamado de **teste alfa**, e o teste de aceitação externa, realizado por usuários reais, conhecido como **teste beta**.

A utilização dos testes de aceitação tem como prerrogativa a liberação completa do software pela equipe de desenvolvimento, devidamente documentado, com seus programas e módulos testados e pronto para sair para o mercado. Enfim, todos os aspectos de desenvolvimento e empacotamento de produto devem estar concluídos, ficando a cargo da gerência do produto determinar exatamente o momento para comercialização do produto.

Esse momento pode ser determinado a partir das informações das seções seguintes, onde são descritos os procedimentos para aceitação do produto tanto interna quanto externamente, com os testes alfa e beta respectivamente.

5.1 Testes Alfa

Ao se ter o produto devidamente empacotado, não se pode considerar o software totalmente pronto para comercialização no mercado. Além dos testes realizados na fase de desenvolvimento, do tipo teste individual, teste de integração, inspeção formal, revisões e *walkthroughs*, são necessários testes que evidenciem a qualidade do produto, levando-se em conta a sua adequação para uso .

Num primeiro momento, os produtores costumam submeter o software para teste na própria organização. Estes testes confinados, comumente chamados de testes alfa, objetivam obter um primeiro retorno da utilização do software na prática.

A aplicação dos testes alfa a um software não se restringe à verificação dos atributos do software (desempenho, funcionalidade, confiabilidade, tempo de resposta, etc..) em si, mas também à toda a estrutura envolvida para interação com os usuários. São avaliados os procedimentos de recebimento, embalagem, distribuição e suporte técnico.

Com isso, os testes alfa pretendem ser uma simulação completa do ambiente do usuário, para observar as falhas do software e da estrutura, a fim de aumentar a qualidade do serviço fornecido.

A disseminação do software pela organização pretende descentralizar os testes de aceitação final do software, para que este possa ser avaliado por pessoas com outra visão do assunto de interesse do software. As equipes de desenvolvimento e testes tendem a se concentrar em determinados aspectos e incorrem, a partir de um determinado estágio, nos mesmos erros e acertos.

A execução dos testes alfa deve ser planejada para que sejam cumpridos os objetivos, prazos e regras desta atividade. [Nise84] estabelece três requisitos básicos para o melhor aproveitamento dos testes alfa :

- ◆ Expandir o uso do teste para toda a organização, não se limitando ao corpo técnico. Quanto maior o domínio, maior a possibilidade de detecção de problemas e sugestões para o produto ;
- ◆ Registrar todos os problemas e sugestões formalmente ;
- ◆ Dedicar um grupo para acompanhamento dos testes e priorização das modificações.

Ao entrar na atividade de teste alfa, os usuários devem ser orientados para registrar seu contacto com o produto em avaliação. A seguir, temos a descrição do formulário destinado aos usuários alfa (aqueles que realizam os testes alfa), utilizado pela Infocon Tecnologia e que foi baseado no trabalho de [Reme87].

A **Infocon Tecnologia** aprecia seu esforço no preenchimento deste questionário. Suas sugestões e comentários são de grande importância para o aperfeiçoamento e sucesso deste produto.

Nome : _____ Função : _____

Período de Testes : _____

Produto : _____ Versão : _____

Durante o período de teste, você utilizou o produto

- A. [] Mais de 4 horas por dia
- B. [] De 1 a 4 horas por dia
- C. [] Menos de 1 hora por dia

Estimo que utilizei o produto aproximadamente ____ horas durante o teste

Caso tenha conhecimento ou experiência com algum produto similar favor enumerar as máquinas e software utilizados

As questões seguintes requerem respostas mais extensas. Favor utilizar espaço ou papel adicional, se necessário.

No seu entender, quais as melhores características do produto ?

E quais são as piores características do produto ?

Caso tenha detectado algum problema, situação indesejada para o cliente ou dificuldades de manipulação do produto, favor citar :

Continua

Como você avalia o desempenho do produto :

Que características você acredita que poderiam ser adicionadas ao produto para torná-lo mais atraente para o usuário :

Usando uma escala de 1 a 10 (1=ruim, 5=médio, 10=excelente), favor indicar :

Embalagem	_____
Recebimento	_____
Documentação	_____
Interface	_____
Facilidade de Uso	_____
Aplicabilidade no seu trabalho	_____
Segurança	_____
Desempenho	_____
Robustez	_____
Satisfação geral com o produto	_____

Voce acredita que este produto esteja pronto para ser disponibilizado ao mercado ? Caso deseje, faça seus comentários sem restrições sobre o produto

Figura 5.1 - Formulário Infocon para Testes Alfa

Ao serem devidamente respondidos pelos usuários e recolhidos pelo grupo de acompanhamento dos testes, os formulários são agrupados para constituir uma base para avaliação do produto. A partir das informações levantadas, o produtor e sua equipe devem decidir pela submissão aos testes betas, reativação do ciclo de produção para alteração do produto ou comercialização do produto.

A decisão sobre os resultados alcançados com os testes alfa, deve levar em consideração alguns pontos importantes para o destino do produto no seu ciclo de vida :

- Criticidade dos erros para o funcionamento do software ;
- Relação custo-benefício para reparar os erros detectados ;
- Avaliação das sugestões de melhoria em função dos propósitos do software para a versão que será lançada ;
- Maturidade do software para submissão aos testadores beta ;
- Possibilidade de comercialização em seguida

5.2 Testes Beta

Mesmo com a realização dos testes alfa, o produtor nem sempre pode assegurar que o seu software esteja definitivamente a prova de erros e atendendo totalmente as necessidades da maioria dos usuários no mercado. Com isso, tem-se adotado no mercado a realização dos chamados testes beta¹, para a avaliação do software numa massa crítica de usuários. Os testes betas basicamente consultam o mercado para avaliar sua aceitação e refinar o produto de acordo com as considerações dos usuários testadores.

Assim como os testes alfa, os teste beta devem ser altamente controlados, com o auxílio do grupo de marketing e suporte do produto. A princípio são identificados quais serão os testadores mais significativos para o produto em questão. Os testadores beta geralmente são escolhidos entre fornecedores e grandes usuários da empresa, por ser uma atividade

¹Estes testes são chamados de testes beta pelo simples fato de serem subsequentes aos testes alfa.

trabalhosa que depende da experiência de mercado e interesse dos testadores. Este trabalho pode ser remunerado em certos casos, mas normalmente é realizado em troca de uma cópia grátis do produto final ou descontos substanciais quando da compra da versão comercial definitiva.

No ambiente de produção de software Unix de prateleira estudado, no caso a Infocon, verificou-se a dificuldade de execução de testes betas no Brasil, apesar da empresa saber da sua importância, realizando-os em algumas ocasiões. A principal contribuição para tal fato é a falta de massa crítica de usuários com potencial de testes, sem esgotar o mercado alvo do produto. Essa dificuldade não acontece em países com maior potencial de mercado, como os Estados Unidos, porque os testes betas são quase que uma praxe de mercado, sendo realizados até para mais de dois mil usuários (no caso do windows NT este número ultrapassou dez mil).

Para se ter uma idéia da dimensão dos testes beta de um produto, os resultados obtidos e a própria divulgação do produto dentre os principais usuários são utilizados como estratégia de marketing para valorização do produto. Os testadores betas que porventura tenham adquirido o software como forma de pagamento do trabalho realizado, são citados na lista de usuários que aprovaram o novo produto, atraindo a confiança do mercado alvo.

No entanto, não existem relatos na literatura consultada, procedimentos uniformes de como executar os testes betas, ficando a critério dos produtores a sua forma de execução. Apesar desta carência, são relatados em [Nise84] algumas questões que devem ser consideradas para o sucesso desta atividade monofásica:

- ◆ A atividade deve ser altamente controlada. Devem ser estabelecidos prazos, contratos com os testadores beta, formulários de teste e grupos de acompanhamento (se possível, com elementos de marketing, qualidade, suporte e desenvolvimento) com disponibilidade de atendimento aos testadores;
- ◆ O software não deve ser utilizado em ambiente de produção. Esta medida garante que a ocorrência de eventuais problemas no software, não venham a danificar os dados reais do ambiente do usuário ;
- ◆ Os testadores beta devem relatar todos os problemas, sentimentos e sugestões sobre o produto ;
- ◆ A consistência da documentação com os recursos oferecidos pelo software deve ser verificada.

Estas considerações merecem atenção, para que todo o esforço tenha os resultados desejados pelo produtor, sejam eles negativos ou positivos. O produtor está submetendo o seu produto ao mercado e deve ter garantias de que este não será disseminado a terceiros antes de ser disponibilizado ao mercado para comercialização. Por isto, deve haver um acordo legal entre testadores beta e produtores para a manutenção do sigilo e garantia de que o produto será devidamente testado, com o formulário do teste beta preenchido corretamente. O aprofundamento das questões jurídicas que envolvem o teste beta, junto com exemplos de acordos entre as partes pode ser obtido em [Reme87].

Logo abaixo, temos o exemplo de um formulário de teste beta, baseado no utilizado pela Infocon para assimilar os resultados dos testes realizados em seus produtos.

A **Infocon Tecnologia** aprecia seu esforço no preenchimento deste questionário. Suas sugestões e comentários são de grande importância, tanto na avaliação quanto no aperfeiçoamento de nossos produtos.

Distribuidor / Empresa : _____
 Nome : _____ Data : _____
 Início dos Testes : _____ Fim dos Testes : _____
 Produto : _____ Versão : _____

Durante o período de teste, você utilizou o produto:

- A. [] Mais de 4 horas por dia
- B. [] De 1 a 4 horas por dia
- C. [] Menos de 1 hora por dia

Estimo que utilizei o produto aproximadamente ____ horas durante o teste

Você tem experiência com algum produto similar em _____
 ou em outros computadores. Caso positivo, favor enumerar as máquinas e softwares utilizados.

Há quanto tempo você vem utilizando computadores na sua atividade de trabalho ? Se desejar, forneça uma breve descrição da sua experiência.

Usando uma escala de 1 a 10 (1=ruim, 5=médio, 10=excelente), favor indicar :

- Embalagem _____
- Recebimento _____
- Documentação _____
- Interface _____
- Facilidade de Uso _____
- Aplicabilidade no seu trabalho _____
- Segurança _____
- Desempenho _____
- Robustez _____
- Suporte Técnico _____

- Satisfação geral com o produto _____

Continua

As questões seguintes requerem respostas mais extensas. Favor utilizar espaço ou papel adicional, se necessário.

Favor citar as melhores características do produto na sua opinião.

Quais são as piores características do produto ?

Caso voce tenha percebido alguma característica estranha ou situação indesejada no produto, favor listar

Caso tenha detectado algum problema no produto, favor citar :

Citar abaixo as características de desempenho do produto :

Favor listar algumas características que poderiam ser adicionadas ao produto :

Você recomendaria este produto para algum amigo ou empresa conhecida ?

Este espaço é reservado para registrar a sua avaliação final sobre o produto. Gostaríamos que comentasse as perspectivas deste produto no mercado.

Figura 5.2 - Formulário Infocon para Testes Beta

Tendo realizado os testes no mercado, a empresa tem que assumir uma postura séria perante a comunidade usuária e distribuidores que lhe deram crédito e aguardam um posicionamento da empresa com relação ao

destino do produto. Diferentemente dos teste alfa, a partir dos testes betas devem ser tomadas atitudes mais definitivas com relação ao produto, comunicando aos usuários o resultado dos testes e o tipo de encaminhamento que será dado. Não sendo necessário justificar as decisões da empresa com relação ao produto, é interessante mostrar ao usuário que quanto as suas sugestões, a empresa decidiu:

- Consertar o problema ;
- Fornecer mecanismos para contornar o problema (*Work-Around*), comunicando ao usuário a solução adotada para o problema, mudanças na documentação, mudanças no material de treinamento e dicas em geral sobre como operar melhor o sistema para atender suas necessidades ;
- Informar-lhe que o problema será considerado num momento posterior frente os objetivos atuais do produto ;
- Informar-lhe que o problema não será considerado ;
- Assimilar as sugestões para serem incorporadas em versão futura do software.

Capítulo 6. Vendas & Distribuição

Neste capítulo são discutidas brevemente as principais estratégias de vendas de software Unix de prateleira. As informações apresentadas resultam da observação do comportamento de um fornecedor (Infocon) no mercado e devem ser ajustadas (principalmente no que se refere a valores exibidos) de acordo com a estratégia de vendas a ser seguida por cada produtor. Acredita-se porém que elas espelhem genericamente as atuais práticas de mercado Unix.

O conteúdo almeja, então, identificar pontos básicos que devem ser ponderados por produtores de software que desejam comercializar seus produtos de software no mercado de acordo com as características destes produtos e dos diversos canais de distribuição.

6.1 Vendas : Necessidades, Dependências e Importância no Ciclo de Vida do Software

A área de vendas no mercado de software tem sido direcionada, assim como em outros mercados, pelo trabalho de marketing. Em software esta afirmação atinge dimensões tão importantes que a venda de um software é considerada uma consequência direta do marketing. Isto nem sempre fica aparente, mas o marketing possui o papel de geração de demanda ou atração do mercado existente com a divulgação do diferencial do produto, utilizando suas estratégias básicas, conforme descrito no capítulo 3.

A equipe de vendas tem o papel de concretizar a compra de um software pelo usuário, que foi atraído pelo marketing. Consequentemente, a ligação entre marketing e vendas precisa ser bem sintonizada, pois a expectativa gerada pelo marketing deve ser correspondida à contento. Isto requer um pessoal de vendas preparado tecnicamente para poder colocar aos futuros usuários os reais benefícios e diferenciais do produto, esclarecendo questões mais específicas do interesse do usuário.

Vendedores de software são identificados frequentemente como "representantes técnicos" (RT) do produto, que além de possuir o perfil de vendas com habilidades para tal e o tradicional "papo de vendedor", devem ter conhecimentos técnicos para uma discussão mais motivante para o usuário.

A realidade mostra que vender software significa vender tecnologia para usuários que procuram soluções para otimizar o seu ambiente de trabalho. É uma tarefa complexa, que requer treinamento e apoio contínuo ao pessoal comercial. O treinamento deve preparar o RT para vender os produtos da empresa, e deve se preocupar em elaborar documentos que os

auxiliem nestas atividades ressaltando os principais atrativos do produto. Alguns destes documentos são : visão geral do produto, perguntas e respostas, prospectos, tabela de vantagens e benefícios e roteiros de diálogo.

Os produtores de software diversificam constantemente a forma de comercializar seus produtos, sendo uma tarefa de difícil modelagem a ponto de se delinear em fórmulas exatas de como se vender com sucesso. Existem alguns indicadores e referências mais extensas sobre o assunto [Nise84], mas que podem ter alguns pontos desatualizados devido à evolução do mercado. O que podemos afirmar com certeza é que as atitudes de vendas devem ser guiadas de acordo com as necessidades do mercado, tomando como condicionantes fatores como:

- Tipo de produto ;
- Segmento de mercado ;
- Investimentos realizados ;
- Distribuição geográfica do mercado ;
- Preço ;
- Pré-venda ;
- Custos.

A preocupação com vendas e sua consideração no ciclo de vida, se justifica pela necessidade deste esforço que representa a sobrevivência do produto no mercado e a sua aceitação, com impacto expressivo na evolução do produto. Além do mais, a execução das estratégias de vendas sincronizadas com os diversos canais de distribuição necessitam de

planejamento e monitoração que influenciam diretamente a alocação de recursos (treinamento, pessoal, custos e tempo).

As seções seguintes destacam as alternativas de comercialização, os fatores que influenciam no preço de um software e as particularidades dos canais de distribuição em termos de margens de lucros, investimentos e suporte necessário.

6.2 Alternativas de Venda

A comercialização de software Unix de prateleira, onde são repassados ao usuário final basicamente o pacote de software com o código executável do produto e a documentação necessária, normalmente acontece de uma das seguintes maneiras:

a) Por assento : o software é licenciado para uso de acordo com o número de usuários (assentos) que poderão acessar simultaneamente o software em ambientes tipicamente multusuários ou que utilizem a arquitetura cliente / servidor. Este tipo de licença de uso requer a utilização de servidores de licença (descrito na seção 4.2.2) para controlar a utilização do software em tempo de execução, .

b) "Site License" : negociação para utilização do software em larga escala numa única corporação. Consiste de um contrato para limitar o número de cópias ao longo da estrutura do cliente, fornecendo descontos significativos devido ao volume do negócio.

c) "Upgrade" : migração de uma versão do software adquirida anteriormente para uma nova versão do produto na mesma plataforma e com o incremento de novas funcionalidades (p.ex., Acess versão 1.0 ⇒ **Acess** versão 2.0). Este tipo de estratégia não pode exigir que o usuário

pague como se estivesse adquirindo o produto pela primeira vez, fornecendo descontos que chegam até 80 % do preço da lista. No mercado verifica-se ainda o *trade in* entre fornecedores, marcado pela atração de usuários de produtos concorrentes com ofertas "únicas" de preços, no caso da troca de produto concorrente pelo do fornecedor. Normalmente exige-se comprovação da aquisição legal do produto concorrente (como capas de manuais, cópias de notas fiscais, etc...)

d) "Crossgrade": migração de uma versão do software adquirida legalmente para uma versão do mesmo software em outra plataforma de hardware (p.e., versão 1.0 para HP Unix ⇔ versão 1.0 para IBM Aix). Os descontos prevalecem nesta situação, mas em menores proporções, chegando até 60% do preço da lista, pela incompatibilidade binária entre as máquinas, requerendo esforços de transporte de código e dados.

As alternativas de vendas não são exaustivas, mas são as mais comuns e podem ser combinadas de acordo com a necessidade do usuário. Evidentemente que as alternativas de migração (upgrade e crossgrade) são realizadas em combinação com o tipo de licença fornecida. Um upgrade, por exemplo, deve ser especificado se será liberado baseado num contrato de "Site License" ou numa licença por assento. "Upgrade" e "crossgrade" são ainda objetos de contrato de suporte técnico, i.e., estas modalidades de comercialização são frequentemente incluídas como "serviço" de suporte.

6.3 Definição de Preço

O sucesso de uma campanha de vendas depende do estabelecimento de uma política de preços adequada à expectativa de mercado e às necessidades de retorno de capital dos produtores. Assim

como a estratégia de vendas, a política de preços depende de alguns fatores que devem ser observados para que o produto possa ter preço que melhor atenda a este compromisso (mercado vs. produtores).

Não existe uma fórmula exata para elaborar o preço do software, mas a observação dos fatores considerados em seguida parece ser determinante.

a) Concorrência : preços de produtos similares no mercado servem como ponto de referência para a definição do preço do software. Analisando-se a funcionalidade (potencialidade) oferecida pelos concorrentes e os respectivos preços, o produtor pode decidir impor uma redução de preço em relação ao concorrente mesmo que ofereça mais funcionalidade ou então igualar e até elevar o preço, direcionando a sua campanha de marketing para as vantagens adicionais que o seu produto oferece.

b) Tamanho do mercado alvo : a estimativa do público alvo que virá a utilizar o produto influencia nas perspectivas de venda, podendo inclusive inviabilizar a comercialização do produto. Este trabalho deve ser realizado na concepção de produtos pelo pessoal de marketing, identificando as necessidades do público alvo e delineando metas a serem atingidas com a comercialização do produto em função da estratégia de marketing e de vendas a ser seguida. O delineamento do mercado alvo resulta em projeções relativas a plataformas de hardware específicas para se chegar a estimativas em torno dos segmentos de mercado que podem ser atingidos.

Em palestra proferida por Scott McNeally, um executivo da SUN Microsystems ®, foi discutida a questão da importância da escolha da plataforma de hardware que o software utilizará. Sugeriu-se que a plataforma de hardware deve vender no mínimo 100.000 unidades por ano

para interessar o produtor de software a desenvolver seus produtos para a plataforma. Esta sugestão toma como base que 100.000 é o mercado alvo mínimo para equilíbrio (*break even*) financeiro da operação mínima do produtor. Se não vejamos:

O preço típico aceitável do produto gira em torno de US\$ 200,00 por assento (um número razoável para software de médio porte na plataforma Unix). Realisticamente, consegue-se atingir no máximo 10% do mercado, o que resulta em 10.000 pacotes de software vendidos por ano. Supondo um desconto médio, chega-se a um faturamento anual de US\$ 175.000. Este valor equivaleria a um orçamento mínimo para manter o escritório, algum esforço de marketing e uma equipe reduzida de vendas e de suporte técnico.

Verifica-se que o tamanho do mercado oferecido pela plataforma-alvo é importante para identificar o potencial de comercialização do produto, o qual deve ser combinado com o preço para atingir as metas do produtor. Observe que um volume de vendas anual de 200.000 no caso acima, poderá significar uma redução de preço do software, digamos para US\$ 150,00 e ainda assim, melhorar a receita do produtor.

c) Oportunidade : Os preços dos produtos de software variam de acordo com o seu posicionamento frente ao mercado. O lançamento de produtos geralmente acontece com uma redução de preços como atrativo para se montar uma base inicial de usuários. Nichos específicos de mercado para um determinado software podem significar preços mais elevados pelo nível de demanda a ser atendido por oferta exclusiva.

d) Tempo : Existe uma tendência natural de redução de preços no decorrer da existência de um produto forçada pelo próprio mercado (face

ao surgimento de concorrentes, amortização de custos/investimentos, etc...).

e) Custos : O produtor deve avaliar o seu esforço de produção de software, contabilizando os custos de concepção, desenvolvimento, preparação, marketing, estruturas de vendas e suporte.

Todos estes fatores compõem a base para a determinação do preço do produto. O bom senso deve prevalecer ao se avaliar realmente quais são os objetivos do produtor com o produto, a situação do mercado e a reação esperada com o preço oferecido em função da política de preços do mercado. Em suma, o preço de um produto no mercado pode ser definido como uma função, F , :

Preço = F (concorrência, tamanho de mercado, oportunidade, tempo, custos)

6.4 Canais de Distribuição

A venda de software tem se tornado cada vez mais difícil devido à forte concorrência e à dispersão geográfica do mercado. Esta realidade tem levado produtores a capilarizar esforços de vendas distribuídos ao longo do seu mercado alvo para maximizar sua atuação.

Os resultados da descentralização de vendas normalmente exigem o envolvimento de terceiros, através de acordos entre produtores complementares (não concorrentes), através da participação de fabricantes de hardware ou por esforços de vendas regionais. No caso de Unix, diversos são os canais de distribuição para software de prateleira, dentre os quais se destacam os seguintes :

a) Vendas diretas : Vendas realizadas pelo produtor ao usuário utilizando sua própria estrutura sem o apoio de terceiros. As vendas diretas oferecem melhores margens de lucros ao produtor, mas ele tem que possuir a estrutura de vendas própria. As vendas podem acontecer porta-a-porta com RT's especializados ou através de estrutura de televendas (este último menos usual para software Unix devido à complexidade técnica). Os custos com suporte, marketing e empacotamento (reprodução, embalagem e transporte) podem ser significativos e devem ser assimilados pelo produtor.

b) Revendas : Estrutura de vendas através de terceiros, montada pelo produtor para capilarizar suas vendas em função de segmentos e/ou regiões de mercado. A revenda de software tem sua própria estrutura comercial e recebe descontos que variam em torno de 20 a 30 % sobre o preço de lista do produto. Opera como uma central de vendas do produtor com metas periódicas a serem cumpridas para justificar sua manutenção.

c) Distribuidores : caracterizados pelo repasse da responsabilidade de montar uma estrutura de revendas para uma outra parte, para que ela e/ou suas revendas comercializem o produto aos usuários. A contratação de um distribuidor varia muito em função do tipo de produto e da forma de atuação do distribuidor. O distribuidor pode adquirir o software do produtor em períodos e volumes pré-estabelecidos, assegurando assim uma receita para o produtor independente da aceitação do produto no mercado. Caso seja realizado desta forma, os descontos para o distribuidor ficam em torno de 50 % do preço de lista, pois caem substancialmente os custos do produtor e o distribuidor assume o risco.

O distribuidor pode também adquirir os produtos sob consignação, isto é, o distribuidor tem o direito de devolver aqueles produtos que não

foram comercializados depois de um determinado período. Alguns distribuidores fazem esta exigência quando estão distribuindo produtos de empresas ainda não consolidadas no mercado, e que conseqüentemente oferecem maior risco para o seu empreendimento. Não são verificados grandes diferenciais para este tipo de negociação, que gera incômodos para o produtor, o qual arca com o custo de empacotamento do seu produto, podendo vê-lo envelhecendo nas prateleiras do distribuidor.

Ao contrário de revendas, o distribuidor normalmente se responsabiliza pelo marketing (cooperado com o produtor ou não) e suporte técnico de primeira linha aos usuários.

d) VAR's ("Value Added Resellers") : Os VAR's agregam valores aos produtos para comercializá-los com maior potencial de mercado. O produtor permite que seu produto possa ser integrado a um outro produto em troca de "royalties" para cada cópia vendida. Com isso, o produtor praticamente se abstém dos investimentos em suporte, marketing e vendas, conseguindo uma boa margem de lucro, mesmo com margens mais estreitas que caracterizam esta negociação.

A agregação de valores também ocorre entre produtores, não sendo restrita apenas aos VAR's, quando desejam incorporar funcionalidades aos seus produtos que são especialidades de outros produtores. Os editores de texto são um bom exemplo de agregação de valores, recebendo características de correção ortográfica e dicionários de outros produtores para incrementar seu potencial rapidamente, sem o esforço de desenvolvimento.

e) Fabricantes de Hardware : pode-se ainda ter uma interação direta com os fabricantes de hardware, que adquirem os produtos de software a baixo custo para serem colocados em cada equipamento vendido como um

atrativo adicional para o usuário. A venda do software embutido (*bundled*) no hardware proporciona ao produtor uma vazão muito grande do seu produto, eliminando o trajeto de sedução do usuário, e aumentando sua clientela na mesma proporção do que a base usuária da plataforma do hardware.

Os fabricantes conseguem grande redução do preço de lista do software, pelas vantagens de disseminação do software na plataforma e pela eliminação dos custos do produtor com empacotamento do software e marketing. O produtor, por sua vez, ganha clientes que apesar de não lhe proporcionarem lucros altos num primeiro momento, poderão requerer evolução e suporte técnico do produto a posteriori.

Os fabricantes de hardware podem ainda comercializar o produto como simples revendas, anunciando o produto nos catálogos de soluções para suas plataformas.

A escolha pelo canal de distribuição correto depende muito do tipo de produto, do potencial de vendas do produtor, da sua localização geográfica, da dispersão do seu mercado alvo, dos custos envolvidos e das suas metas. Além do mais, o produtor deve ter em mente a relação existente entre os canais de distribuição, suas respectivas margens de lucro e custos com suporte, marketing e empacotamento, para orientar suas estratégias de venda. A figura 6.1 ilustra esta relação, exibindo uma região representativa da variação das margens de lucro em função dos diferentes canais de comercialização, e as possibilidades de atitudes e acordos de vendas discutidas.

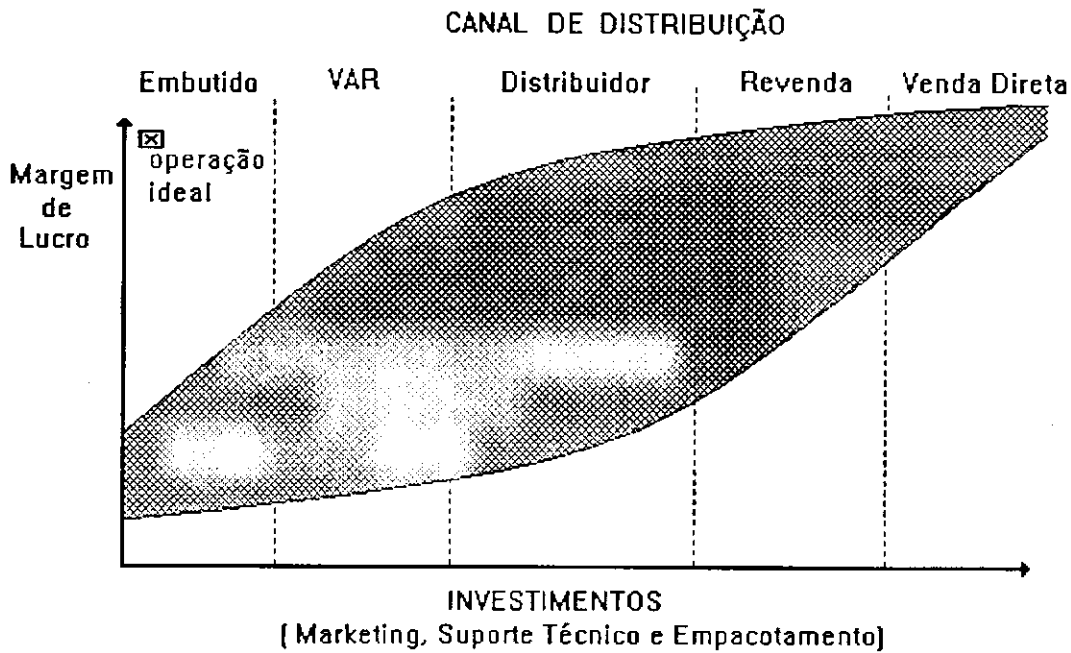


Figura 6.1 - Ilustração dos Custos x Margens de Lucro em Função do Canal de Comercialização

6.5 Homologação de Software

O segmento de mercado UNIX possui algumas particularidades, as quais obrigam os desenvolvedores de software a adaptarem sua comercialização à realidade mercadológica.

Nesta seção, são consideradas as experiências de produtores no fornecimento de cópias para homologação junto a usuários interessados. No seu desenrolar, são sugeridos alguns procedimentos que devem ser observados no processo de homologação de software.

Os usuários UNIX são majoritariamente corporativos, sendo altamente exigentes quanto a aquisição de produtos de software que porventura intervenham diretamente nos seus negócios. Por ser um mercado restrito, em volume de usuários, os produtores de software para

UNIX procuram respeitar as exigências de homologação das empresas usuárias¹.

Notadamente reconhecida como estratégia empresarial, a homologação marca o começo da relação do usuário com o produtor (ou fornecedor, se for o caso). Neste procedimento deve-se buscar a satisfação total do usuário, pois será decisiva para a aquisição do software e imagem da empresa no mercado. Entendendo-se neste momento como satisfação do usuário: a adequabilidade do produto as suas necessidades, assistência técnica, performance aceitável, apoio de marketing e vendas e qualidade intrínseca do produto.

O processo de homologação de um software se inicia pela solicitação de uma cópia do produto para ser testada e avaliada pela equipe técnica e/ou usuários finais do cliente potencial. Esta cópia se denomina cópia de homologação.

Abaixo são relacionados alguns pontos cuja observância colaboram para a qualificação do processo de homologação de um software UNIX :

- Assinatura de um contrato de homologação, especificando-se claramente as suas cláusulas (período de uso, número de usuários, suporte, limitações de distribuição e sanções por violação).
- Entrega do software completo, sem restrições de funcionalidade, preferencialmente a custo zero ;
- O software deve ser auto-documentado com ajudas *on-line*, e acompanhado apenas do material de instalação e operação básica.

¹Considerando-se o mercado UNIX no Brasil no ano de 1992, o segmento industrial responde por vinte e quatro por cento das instalações, seguido de perto de empresas de finanças privadas com um índice de quase vinte e dois por cento (fonte : Infocon).

Para clientes estratégicos a documentação completa pode ser eventualmente remetida ;

- O uso de tutoriais embutidos no software, com exemplos ilustrativos sobre sua utilização ;
- Utilização de algum esquema de limitação de acesso ao software por expiração do contrato de homologação (p.ex., servidores de licença);
- Definir meios para ampliar o prazo de homologação do software rapidamente (preferencialmente com chaves de ativação) ;
- Atuação contínua dos grupos de marketing e vendas sobre o processo. No fundo, deseja-se esclarecer ao usuário quanto às vantagens do software para o seu ambiente, tentando colocar o produto como a solução dos seus problemas. Algumas possíveis estratégias : avaliação periódica do interesse do usuário, possíveis demonstrações do produto, envio de notas técnicas para motivar usuários e divulgação da lista dos atuais clientes ;

Claramente pode ser notado que a homologação requer planejamento e controle, pois envolve atividades de controle de qualidade, marketing e investimentos. A empresa produtora de software deve levar em consideração todas as atividades, inclusive as de desenvolvimento de rotinas de bloqueio/desbloqueio de acesso ao software.

Antes de se pensar em distribuir um software para homologação, deve-se conceber um esquema apropriado para estabelecer os limites de utilização do software que o diferenciam de um software adquirido. Existem algumas opções de limitação de operação, que estabelecem um ponto de

decisão do usuário sobre a aquisição do produto. Dentre as existentes, as mais praticadas são :

- Limitação de uso por data ;
- Limitação de uso por volume de dados manipulados ;
- Restrição de funcionalidade do software ;
- Restrição do número de usuários ;

Genericamente falando, o usuário UNIX não aceita muito bem ter um software restrito funcionalmente, muito menos ser limitado pelo volume de dados que processa. Pelo que se verifica, existe uma tendência dos grandes produtores de software em limitar suas cópias de homologação por prazo e eventualmente pelo número de usuários.

Independentemente da limitação imposta ao usuário, este se sente incomodado em ter que conviver com esta situação temporária, o que deve ser bem dirigido para não implicar na perda do cliente. Visando permitir maior flexibilidade neste "incômodo" processo, são utilizados servidores de licença (descritos na seção 4.2.2.1) com limitações de acesso em função do prazo e do número de usuários pré-estabelecidos.

Ao solicitar uma cópia para homologação, o usuário descreve seu ambiente e negocia o prazo de testes e a quantidade de usuários que utilizarão o software. O usuário recebe o software acompanhado de uma chave de ativação gerada em função deste parâmetros, que habilita o software para funcionamento.

O mecanismo é bastante simples. Sempre que o software for acessado, são testadas a expiração da data e a quantidade de usuários que estão acessando o software naquele exato momento. Se o limite de

usuários tiver sido alcançado, o usuário deve esperar até que algum usuário pare de usar o software. No caso da data, o sistema fica inacessível² até que se contacte com o produtor ou distribuidor de software para liberá-lo.

Ao ser contactado, o distribuidor analisa a posição do cliente e deve tentar negociar a aquisição do produto. Em obtendo sucesso e efetivada a venda, basta informar uma chave de ativação definitiva e enviar o restante da documentação do software. No caso de prorrogação, o distribuidor deve estabelecer uma política de estreitamento de prazos e número de usuários para incentivar o usuário a tomar uma decisão. O distribuidor tem autonomia limitada pela empresa de software para prorrogação de prazos.

A relação com o usuário deve ser aproveitada ao máximo pela empresa de software, pois pode obter informações relevantes para atender melhor o mercado. Mesmo em caso de desistência, a empresa deve procurar saber o seu motivo e colher sugestões para atendê-los futuramente. Isto melhora a imagem da empresa frente aos clientes.

Toda a interação com o cliente, opiniões e críticas sobre o produto devem ser devidamente documentadas e de fácil acesso. Isto pressupõe o seu armazenamento automatizado, de forma que se possa recolher informações sobre necessidades emergentes no mercado, além de possuir dados sobre instalação e produtos de clientes, dados estatísticos sobre vendas, aceitação dos produtos em homologação no mercado, rendimento de distribuidores, dentre outros dados gerenciais envolvidos.

²Sugere-se que o software não seja bloqueado inadvertidamente. Avisos prévios periódicos devem ser exibidos ao usuário para não deixar o software inoperante.

Claramente pode ser notado que a homologação requer planejamento e controle, pois envolve atividades de controle de qualidade, marketing e investimentos, sendo uma atividade importante de pré-venda. A empresa produtora de software deve levar em consideração todas estas atividades, inclusive as de desenvolvimento de rotinas de bloqueio/desbloqueio de acesso envolvidas na homologação de um software.

Capítulo 7. Suporte Técnico

Neste capítulo são abordadas questões inerentes à atividade monofásica de suporte técnico, da fase de disponibilização do software para o mercado, conforme especifica o molde de produção de software comercial descrito no capítulo 3.

Basicamente, são detalhados os procedimentos necessários para o fornecimento de suporte técnico, sendo propostos alguns procedimentos para incrementar a qualidade do serviço de suporte oferecido ao usuário de software. Os resultados relatados são reforçados pela experiência prática de profissionais da área, que vislumbram o suporte como um serviço imprescindível para a sustentação e evolução dos produtos de software.

O capítulo ressalta o dinamismo do suporte durante vários momentos do processo de produção de software, destacando sua estreita relação com os grupos de marketing, vendas e manutenção do software.

Como será visto no capítulo, o grupo de suporte será necessário sempre que se colocar o usuário em contacto com o produto, seja em testes, aquisição ou homologação de produtos.

7.1 Introdução

Na literatura e no próprio mercado de informática, o conceito de suporte técnico não possui fronteiras bem definidas, sendo suas atividades variáveis em função do segmento de mercado e política da empresa. Na realidade, pode-se observar que alguns ambientes tratam o suporte técnico como uma equipe isolada, destinada apenas a administrar os recursos e garantir o funcionamento de uma máquina específica.

A diferenciação entre suporte técnico e manutenção deve ser claramente compreendida. Apesar de serem equipes que juntas, buscam discutir e encaminhar soluções para os usuários, a equipe de manutenção se volta claramente para questões de desenvolvimento que tratem da resolução de problemas (*bugs*), adaptação e evolução dos produtos de software (atividades estas discutidas em detalhes no capítulo seguinte).

O suporte técnico funciona em contacto direto com o usuário, se responsabilizando pela resolução dos problemas técnicos necessários para a operação normal do software (instalação, configuração, portabilidade, dúvidas em geral). O grupo de manutenção é ativado quando forem necessárias intervenções inerentes à fase de desenvolvimento do produto (i.e., que exijam alterações no seu código fonte).

Em [Evan87] considera-se que a existência de atividades de suporte a um produto de software, se justifica na busca contínua da satisfação plena do usuário, garantindo que suas dificuldades sejam resolvidas no menor espaço de tempo possível, possibilitando a utilização ótima dos

recursos oferecidos pelo software. Na norma ISO 9000-3 para software **[Iso9003]**, que direciona as normas ISO 9000 **[Iso9000]** para a área de produção de software, verifica-se que existe uma preocupação em estabelecer o domínio das atividades de suporte. De um modo geral, esta norma possui apenas diretrizes globais a serem observadas, não pretendendo ser uma fonte efetiva para quem quiser suportar tecnicamente um software no mercado. Ainda segundo esta referência, o produtor de software deve conceber uma estrutura flexível o bastante para atender a ocorrência inesperada de problemas, sugerindo que estes sejam detectados, analisados e corrigidos a contento do usuário. Se necessário for, o produtor deve buscar soluções temporárias, de forma a não descontinuar a operação do software (i.e., soluções que evitem a causa de mal funcionamento, chamadas de *workarounds*).

Vale ressaltar que a discussão a seguir, tem a preocupação de seguir as diretrizes gerais de procedimentos e documentação citadas na norma ISO 9003 para software, que tendem a ser um padrão de aceitação em qualidade de software, apesar da sua generalidade.

7.2 Uma visão de suporte

A satisfação plena do usuário deve ser resultado de um processo mútuo de busca por melhorias entre usuários e produtores. Intuitivamente, para se satisfazer um conjunto de pessoas que já utilizam ou que possam vir a utilizar um software, deve se levantar as necessidades reais e potenciais deste público alvo. Essa função, dentre outras mais, deve ser

incorporada em parte¹ ao suporte, que parece agora oferecer um perfil muito mais ativo junto aos usuários.

O suporte passa a ser encarado como uma atividade estratégica dentro da empresa produtora de software. O seu funcionamento tende a possuir uma gama de atividades que não ficam restritas à solução de aspectos técnicos. Pelo contrário, o suporte pode ser aproveitado como elo de ligação com o usuário, podendo conseguir informações importantes para a avaliação real do produto, conforme ilustrado na figura 7.1. Estas informações (opiniões, sugestões, levantamento de erros, necessidades funcionais, problemas técnicos, dentre outras) servem como base para implantação de melhorias em todas as etapas do ciclo de produção.

O mercado UNIX, caracterizado por usuários corporativos, exige um atendimento quase que personalizado a seus clientes, onde qualquer variação no quadro de clientes é significativo. Por isto, os produtores que conceberem suporte sob este prisma, efetivamente estarão buscando a conquista do seu espaço neste difícil e promissor mercado de software.

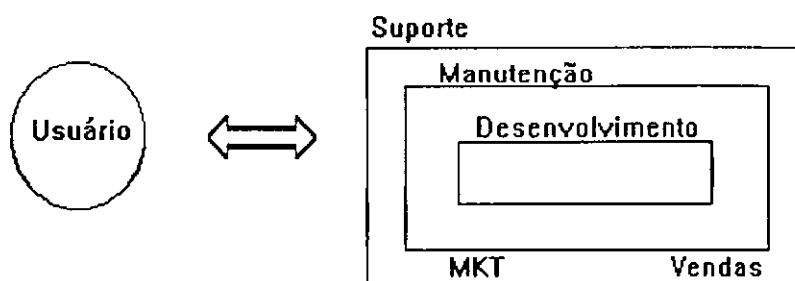


Figura 7.1 - O Suporte como Estratégia de Integração

¹No caso, o suporte funciona como receptor de informações, servindo de apoio ao marketing, que realiza a consulta seja na concepção de um novo produto, seja na evolução de algum já existente.

7.3 Ambiente Proposto

Atentando para a íntima relação existente entre satisfação do usuário a nível de suporte recebido e a credibilidade da empresa no mercado, esta seção sugere um ambiente básico de suporte UNIX, que pretende melhorar a qualidade dos serviços oferecidos e do processo de produção de software, baseados na realidade da sua comunidade usuária.

Sugerem-se critérios para alocação de pessoal especializado, tipos de serviços a serem oferecidos nas etapas de pré-venda e pós-venda, atendimento e coleta de informações do usuário, metodologia para análise e tratamento de problemas, relatórios de acompanhamento e documentação necessária. A discussão finaliza com algumas metas de qualidade e produtividade para o suporte.

7.4 Perfil do Profissional de Suporte

Antes de se pensar em qualquer mecanismo ou estrutura de suporte em produtos UNIX, o produtor de software deve se preocupar em montar uma equipe que possua notadamente o perfil para exercer um suporte dinâmico. Talvez seja esta a tarefa mais difícil de ser realizada em todo o processo, pois visualizando suporte sob este novo prisma, a equipe escolhida responderá sobre assuntos de naturezas distintas. O grupo de suporte amplia sua atuação para questões do tipo apoio a marketing, vendas, informações sobre evolução de produtos, controle de qualidade, satisfação do usuário, dentre outras. Por exemplo, um profissional deste gabarito pode ser questionado sobre problemas de portabilidade de produtos para uma nova versão do UNIX, como num momento seguinte pode estar atendendo a um questionamento sobre benefícios (custos,

funcionalidade, suporte oferecido, etc) de um *upgrade* para a versão recém-lançada pela empresa.

A definição de critérios para a escolha de um profissional notadamente versátil, pode variar de empresa para empresa. O profissional de suporte deve ter um perfil que atenda basicamente aos requisitos abaixo, sendo alguns destes conseguidos obviamente depois de uma orientação da empresa :

- Conhecimento dos fundamentos e funcionamento interno do sistema operacional UNIX;
- Conhecimento aprofundado sobre a família de programas da empresa (obviamente que exige-se pleno domínio sobre os que de fato serão suportados) ;
- Facilidade de comunicação e habilidade no trato com o público em geral ;
- Adoção de padrões de atendimento junto ao usuário : presteza, paciência, motivação do usuário, compreensão das suas necessidades e atitudes que tornem o contacto o mais agradável possível ;
- Noções sobre as políticas de vendas, distribuição e marketing da empresa ;
- Boa redação para elaboração de relatórios, sugestões e notas técnicas ;
- Capacidade de assimilar a evolução do UNIX, pelo menos nas suas principais versões, conhecendo suas novas características ;

- Conhecimento dos principais fornecedores de periféricos, permitindo a configuração do software para estes dispositivos.

7.5 Organização das Atividades de Suporte

As atividades de suporte de software nem sempre ficam reduzidas ao apoio ao usuário adquirente do produto da empresa. Com a concorrência acirrada do mercado de software, produtores e distribuidores se valem de estratégias de marketing para o convencimento de potenciais clientes (chamados de *prospects*) a comprarem o(s) produto(s), o que requer participação efetiva do grupo de suporte. As atividades exercidas pelo suporte nesta situação são conhecidas como suporte pré-venda.

No segmento Unix que exige normalmente maior atenção do atendimento técnico na homologação do produto, o suporte pré-venda é particularmente importante. Não menos importantes, as atividades de suporte ao usuário do(s) produto (s) após aquisição são coletivamente de responsabilidade do suporte pós-venda. As seções seguintes abordam as atividades do suporte pré e pós-venda respectivamente.

7.5.1 Suporte Pré-Venda

No Brasil, por exemplo, existe uma cultura criada localmente, de que qualquer software, independente da sua amplitude e preço, deve ser colocado num processo de *homologação* no usuário (seção 6.5). Os usuários exigem tal comportamento e quem quiser ter participação no mercado nacional deve atender as suas condições.

Como consequência imediata, o software precisa ser suportado, normalmente sem ônus para o *prospect*, de uma forma tal que atenda amplamente suas expectativas de aquisição do software. Afinal de contas,

o *prospect* está analisando o produto sobre todos os aspectos, e a qualidade do suporte oferecido também é questionada, pois pode ser o início de um longo relacionamento.

O suporte pré-venda por vezes, tem que promover *palestras, demonstrações técnicas e assessoria* ao *prospect*, colocando os benefícios que podem ser provenientes da aquisição do software. Ações como estas, são estratégias de marketing que cativam o mercado, que até então não possuía uma visão prática quanto à utilização do software no seu atual contexto.

A última questão que merece ser ressaltada diz respeito a avaliação dos resultados efetivos do suporte pré-venda. Cada empresa tem sua política particular de investimentos nesta linha e deve decidir onde, quando e principalmente para quem deve oferecer tais serviços. Apesar dos serviços de pré-venda impactarem em despesas imediatas para o produtor, o suporte deve ser monitorado como centro de lucros. Os serviços de pós-venda podem gerar receitas que talvez suplantem as despesas acumuladas no pré-venda.

É importante notar que apesar da organização do suporte se dar em duas partes funcionais de pré e pós-venda, a equipe de suporte pode ser única, responsabilizada pelas atividades de ambas as divisões.

7.5.2 Suporte Pós-Venda

O *suporte Pós-venda* se caracteriza pelas atividades de apoio ao usuário que adquiriu legalmente o software para uso pessoal ou na sua empresa. Dentro da perspectiva de se conceber um ambiente de suporte com uma visão mais ampla, deve-se atentar para os serviços básicos que o

usuário deseja receber, para então se planejar qualquer interferência perante o usuário.

Um levantamento neste sentido foi realizado [Evan87] para se registrar o que realmente o usuário espera do suporte do produto, e sobre que condições. Os usuários declararam que os serviços na tabela 7.1, por ordem de importância, devem ser fornecidos pelo produtor; e, de forma gratuita no caso dos serviços que forem declarados "padrão". Segundo o levantamento, os usuários consideram razoável a cobrança dos serviços "extras" :

SERVIÇO	TIPO
1. Atendimento por telefone no horário comercial	padrão
2. Documentação	padrão
3. Correções de erros	padrão
4. Procedimentos de Instalação	padrão
5. Garantia de funcionamento e possíveis danos	padrão
6. Atendimento por telefone 24 horas	extra
7. Discussão sobre o ambiente antes da instalação	padrão
8. Treinamento no usuário	extra
9. Suporte nas instalações do usuário	extra
10. Treinamento nas instalações do vendedor	extra
11. Notas técnicas / carta-resposta	padrão
12. Consultoria de software	extra
13. Grupo de Usuários	extra

Tabela 7.1 - Serviços Oferecidos pelo Suporte Pós-Venda

Na seção seguinte será enfatizada a questão do atendimento ao usuário, não apenas por ser considerada de maior importância pelo próprio usuário, mas pelo reconhecimento do seu valor estratégico para a implantação de melhorias, análise de produtos no mercado e detecção de novas oportunidades de negócio.

7.6 Atendimento ao usuário

Ao se considerar isoladamente o suporte técnico de um produtor de software, este pode ser classificado como prestador de serviços a comunidade usuária dos produtos da empresa. Como tal, a qualidade dos seus serviços é analisada pelos seus clientes em função do nível de atendimento prestado a sua clientela. O que genericamente pode ser expresso por :

$$\text{Qualidade do suporte} = f(\text{nível de atendimento do usuário}),$$

onde "atendimento" reflete todas as atividades que possam ser executadas a partir de uma solicitação do cliente, desde uma simples chamada até uma consultoria nas suas instalações.

Pode parecer um pouco precipitado expressar qualidade do suporte desta forma, mas é assim que os usuários a consideram. A partir deste princípio, deve-se ampliar o conceito de atendimento ao usuário e achar maneiras de mensurar e melhorar o nível de qualidade oferecido.

Na maioria dos casos, o contacto realizado entre o usuário e o suporte técnico, recai numa chamada telefônica onde o usuário busca respostas rápidas para solucionar seus problemas com o produto adquirido. Ciente deste contacto com o usuário, a empresa deve estar

preparada para atendê-lo eficientemente, em tempo hábil, procurando tornar este contacto um benefício para ambas as partes.

A seção seguinte propõe um mecanismo de suporte automatizado, descrevendo sucintamente as informações e procedimentos envolvidos na organização do atendimento aos usuários.

7.7 Metodologia para Automação do Suporte

Para que se possa otimizar o atendimento ao usuário, o suporte necessita passar por um processo de organização interna, onde é necessário levantar os procedimentos envolvidos e informações necessárias para se melhorar o nível de qualidade do suporte oferecido.

No intuito de organizar o trabalho do grupo de suporte, independentemente de como seja realizado, o atendimento à solicitação de clientes passa necessariamente pelas seguintes etapas :

1. Identificação do usuário e do produto ;
2. Autenticação ;
3. Solicitação do usuário ;
4. Identificação do interesse ;
5. Resolução ou encaminhamento.

Considerando-se estas etapas, verifica-se que algumas (1 e 2) são meramente formais, ficando restritas a procedimentos repetitivos de verificação da legitimidade do serviço. Por outro lado, outras subsequentes (4 e 5) requerem capacitação profissional e prática para serem resolvidas a contento, mas que também possuem caráter repetitivo a medida que os usuários possuem as mesmas dificuldades.

Analisando a natureza das etapas envolvidas, pode ser verificado junto a produtores de software a necessidade de uma *metodologia para automação do suporte técnico*. Esta idéia, já implantada diferenciadamente em alguns grandes produtores, como Lotus e Microsoft, dinamiza as etapas de atendimento, tornando o atendimento a clientes mais rápido e confortável para ambas as partes.

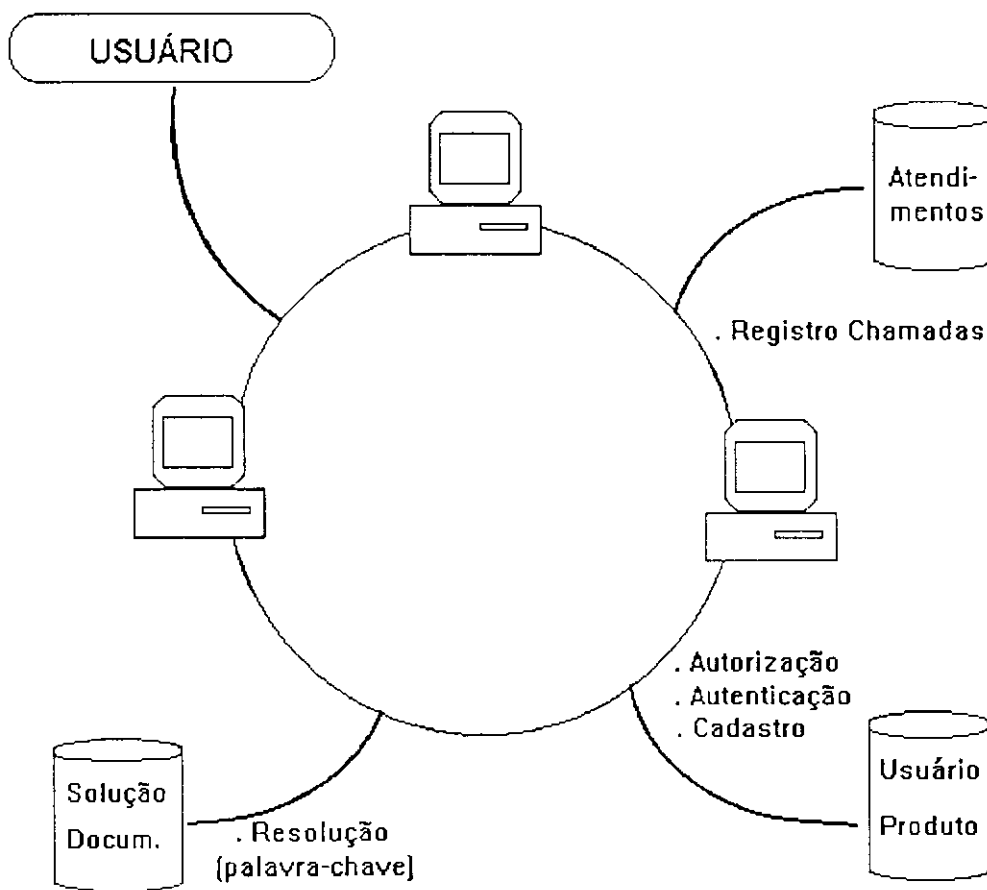


Figura 7.2 - Metodologia para Automação do Suporte

A metodologia, ilustrada na figura 7.2, se apoia na utilização de alguma ferramenta automatizada que suporte a criação e manipulação de

um banco de dados sobre produtos e usuários, capaz de alcançar os seguintes objetivos:

- Identificar rapidamente o usuário, obtendo uma descrição do seu ambiente de trabalho em termos de versão do produto, configuração de hardware e versão do sistema operacional.
- Verificar se o usuário tem direito aos serviços de suporte (autenticação), sugerindo se for o caso a sua legalização com o estabelecimento ou renovação de contrato.
- Registrar o atendimento do usuário, formando uma base para análise do tipo, quantidade e qualidade de serviços prestados.
- Auxiliar efetivamente na busca de soluções, baseando-se em problemas anteriormente resolvidos armazenados num banco de soluções.
- Fornecer rapidamente documentos e notas técnicas aos usuários.
- Registrar informações sobre a satisfação do usuário com o produto, desde correções de erros até solicitações de incremento de funcionalidade.

7.7.1 Descrição do Banco de Dados

Em seguida, têm-se uma *descrição lógica* do banco de dados baseada na abordagem relacional [Date84] para sistemas de bancos de dados. De acordo com esta abordagem, são descritas os diversos arquivos ou bancos de dados específicos (ou relações segundo o modelo relacional), com os seus campos especificados (atributos), e suas chaves de identificação única dos registros (chaves primárias) devidamente

sublinhadas. Vale ressaltar que não houve uma excessiva preocupação sobre aspectos de implementação nem de normalização das relações. A intenção neste momento é de levantar o conteúdo de um banco de dados para o suporte e benefícios advindos da sua adoção.

O banco de dados sugerido abaixo foi discutido junto com produtores locais de software Unix de prateleira que suportam tecnicamente clientes externos. A partir das observações realizadas e do atual nível de automação, foram propostas melhorias que possibilitem atender aos objetivos esperados com o uso da metodologia (descritos na seção anterior), possibilitando a geração de relatórios mais eficientes para monitoria e avaliação do suporte (seção 7.7.3).

→ Banco de Produtos

Número de série, empresa, tipo de uso (comercial, homologação, demonstração, cortesia), versão, versão interna, S.O., versão S.O., fabricante, mídias geradas, data da aquisição, nota fiscal, número do contrato de suporte

→ Banco de Usuários

Nome da empresa, endereço, telefone, fax, cgc, contacto 1, contacto 2

→ Banco de Problemas

Produto, versão, palavra-chave, identificação da solução

→ Banco de Soluções

Identificação da solução, solução, documento

→ Banco de Documentos

Tipo de Documento, conteúdo

→ Banco de Registros de Atendimento

Código de atendimento, data, hora, número de série do produto, tempo de atendimento, suporte, interesse do usuário (resolução de problemas, dúvidas, reclamações, compras, *upgrade*), status (pendente, resolvido), encaminhamento, satisfação com o produto, opiniões do usuário.

7.7.2 Procedimentos

A compreensão da metodologia se resume ao acompanhamento dos passos descritos logo a seguir, observando-se sempre a descrição do banco de dados e a figura 7.2.

Passo 1. A ferramenta automatizada deve estar sempre ativada e pronta para receber ligações. Ao atender qualquer chamada, deve se abrir um registro, ativando um temporizador, mesmo que a ligação não seja pertinente². O suporte deve gerar um código de atendimento para cada solicitação em função do produto. O código é composto, por exemplo, pelas iniciais do produto e um sequencial único.

Passo 2. Identificação do usuário . O usuário pode se identificar basicamente de duas maneiras perante o suporte :

- *Através do número de série do produto*, o qual deve ser checado junto ao banco de produtos se o usuário tem direito ao suporte (autenticação) . Caso o usuário ainda não tenha registrado o software, e conste no banco de produtos como número de série gerado pela empresa, o suporte pode fazê-lo pelo telefone para

²No levantamento do espaço amostral serão consideradas todas as ligações telefônicas destinadas ao suporte, afim de que o estudo probabilístico sobre estes dados possa ser realizado com maior precisão.

facilitar a vida do usuário (inclusão no banco de produtos e de usuário, caso seja o primeiro software adquirido pelo usuário).

- Quando uma solicitação já foi encaminhada anteriormente e o usuário deseja saber do seu andamento, basta informar o *código da solicitação*, que foi gerada pelo suporte e repassada para o usuário no momento da solicitação. De posse do código da solicitação do usuário, a ferramenta acessa o banco de registros de atendimento e verifica a situação daquela solicitação, fornecendo ao usuário o encaminhamento que foi dado.

P_sso_3. Dando sequência a um atendimento convencional de suporte, depois de estar de posse de todos os dados do software, e do ambiente do usuário na tela, o suporte tenta identificar *o que* realmente o usuário deseja.

O usuário pode requisitar o suporte pelos mais variados motivos, o que dificulta a sua classificação. Numa primeira instância, deve-se analisar o que realmente é de competência do suporte e que pode ser resolvido o mais rapidamente possível.

As solicitações de usuários podem ser divididas em alguns grandes grupos, o que facilita o seu encaminhamento, análise e classificação de dados para posteriores levantamentos sobre as atividades da empresa :

- Manutenção ;
- Vendas ;
- Problemas de suporte ;
- Solicitação de serviços adicionais de suporte pré-venda (seção 7.5.1) e pós-venda (ver tabela 7.1) ;

- Outras (sugestões, reclamações de atendimento, enganos) ;

No caso de solicitações que necessitem de interferência do grupo de *manutenção*, ou seja, que requeiram alteração no escopo do código fonte, deve-se registrar e colocá-las como pendentes, sendo encaminhadas a manutenção para análise . Estas solicitações são devidamente estudadas e atendidas por ordem de prioridade, recebendo um tratamento específico conforme será descrito no capítulo 8, seção 8.3.2.

Cada interesse do usuário merece um atendimento diferenciado do grupo de suporte, devendo, por exemplo, ter a flexibilidade em assimilar questões da área de *vendas* da empresa. Os fabricantes de software devem estar atentos para a possibilidade de ampliar seus negócios com o apoio do suporte, porque a solução de um simples problema do usuário pode ser a oportunidade para se gerar negócios. De sorte que o banco de atendimentos deve registrar qualquer interesse nesta linha, estreitando ao máximo suas relações com o pessoal de vendas e marketing.

Ao se manter contacto com os profissionais de suporte de produtos Unix, observam-se os seus registros de atendimento a usuários, chegando-se a uma classificação dos *problemas de suporte* típicos para este ambiente. A observação destes problemas verificados servem como referência para se estudar o comportamento dos produtos no mercado :

- Configuração de Terminal
- Teclado
- Impressão

- Sistema Operacional
- Engano do cliente
- Desempenho
- Hardware
- Descaso de documentação

Passo 4. Soluções. Pelo que se pode observar no suporte UNIX, os problemas se concentram em torno de configuração e da interface do software com o sistema operacional. Depois que o produto é disponibilizado ao usuário, as solicitações dos usuários começam a se tornar repetitivas, fazendo com que o atendimento às solicitações se torne uma tarefa demasiadamente desgastante e improdutiva.

A organização dos conhecimentos sobre os produtos, e a vivência do grupo de suporte podem evidentemente otimizar o atendimento do suporte, melhorando a sua qualidade em termos de tempo de resposta e eficiência.

Como pôde ser observado na figura 7.2, a metodologia sugere que se utilize um banco de soluções, funcionando como um banco de conhecimentos sobre suporte técnico de produtos. Este recurso provê o suporte de respostas rápidas a questões que já foram resolvidas anteriormente, onde cada nova solução alcançada seria incluída como um novo registro no banco de soluções com palavras-chave associadas.

O esquema para obtenção de soluções se baseia numa pesquisa seletiva com palavras-chave no banco de dados. Em primeiro lugar, o suporte classifica o problema, selecionando quais palavras-chave servem como entrada para o problema em questão. Após definidas a(s) palavra(s)-

chave para pesquisa, o banco de problemas é acessado e unindo-a(s) com o produto em questão se obtém uma identificação ou um conjunto de identificações de soluções a serem acessadas no banco de soluções.

A questão da dificuldade e demanda de tempo na elaboração de documentos não deve ser desprezada, pois requer do suporte uma interação contínua com seus usuários. Pensando neste fato, o banco de documentos armazena alguns documentos padrões que o suporte pode emitir para seus usuários. No caso de necessidade, são gerados alguns documentos, basicamente contendo soluções, para serem enviados para o usuário assim que estes problemas críticos sejam questionados.

Passo 5. Satisfação do usuário. O suporte deve ter em mente que o seu objetivo maior é satisfazer as necessidades de seus usuários da melhor forma possível, e por se tratar de uma atividade de serviço, fica difícil saber até que ponto esta meta foi alcançada. Por isso, deve-se tentar captar do usuário o seu nível de satisfação com o produto.

A princípio, a medição da satisfação do usuário durante o atendimento não pode ser realizado a contento pelas limitações de tempo e incômodo para o usuário, a não ser que sejam solicitadas apenas opiniões subjetivas sobre o produto. De qualquer forma, deve-se abrir um espaço no banco de dados para registrar o sentimento do usuário com relação ao produto, que pode ser complementado com uma simples atribuição pessoal do seu nível de satisfação.

Com relação ao nível de satisfação, solicita-se apenas que o usuário forneça uma opinião subjetiva que reflita o seu sentimento sobre a utilização do produto no seu ambiente de trabalho. O usuário pode fornecer uma resposta em função das seguintes opções, também utilizadas para

mensurar qualidade subjetivamente em outros trabalhos [Roch92] :
Deficiente, regular, médio, bom, excelente .

Ao se obter este retorno mínimo do usuário, a chamada pode ser encerrada e incluída no banco de atendimento. Esta avaliação preliminar sobre a satisfação do usuário pode ser ampliada com o envio de questionários mais elaboradas sobre a qualidade do produto, incluindo também o nível de atendimento prestado ao cliente.

A seguir, são descritas algumas maneiras de se aproveitar os dados advindos da utilização desta estratégia automatizada de suporte.

7.7.3 Análise dos Dados

Pelo que se tem percebido no mercado, não existem dados sobre o atendimento ao público, e quando existem não são bem aproveitados. A questão é puramente cultural. O empresariado brasileiro não possui o hábito de coletar e utilizar dados. Pelo que se pode observar, não existe nenhuma publicação nacional na área de suporte de software que forneça dados estatísticos sobre suporte de produto, necessidade de manutenção ou nível de satisfação do usuário.

A organização de um banco de dados que contenha informações sobre o andamento das atividades do suporte pode ser decisivo para a adoção de estratégias da empresa. A metodologia proposta a seguir tenciona fornecer informações que norteiem a empresa sobre a realidade vivenciada pelos seus usuários, seus anseios, sua interação com o suporte e a avaliação da qualidade do atendimento fornecido.

O banco de dados proposto está projetado para disponibilizar informações que podem variar de um simples acesso a um registro sobre usuário ou uma chamada, até relatórios e gráficos de análise sobre o

desempenho do produto no mercado. Destacamos três das possíveis alternativas que podem ser obtidas a partir do esquema proposto :

1. Utilizando *gráficos de Pareto* [Falc90] para evidenciar os principais problemas registrados sobre os produtos.

Considerada como uma das principais ferramentas utilizada na filosofia de controle de qualidade total, o gráfico de Pareto objetiva ressaltar quais são os problemas prioritários que interferem no processo analisado. Neste contexto, esta análise pode ser feita a partir do banco de registros de atendimento, onde constam os registros dos problemas inerentes a cada produto. A aplicação da análise de Pareto acontece da seguinte maneira:

- ◆ Levantamento e contabilização da ocorrência dos problemas no banco de registro de atendimentos (verifique tipos de problemas no passo 3).
- ◆ Ordenação descendente dos problemas em função da ocorrência de chamadas.
- ◆ Agregação de valores, cálculo do percentual acumulado e emissão do gráfico.

Para se ter uma idéia da aplicação de gráficos de Pareto, considera-se o exemplo na tabela 7.2 e figura 7.3, sobre possíveis problemas encontrados no produto fictício Connect , versão 2.0, durante o quarto trimestre de 1993.

Problema	Chamadas	Acumulado	%Acumulado
Descrição de Terminal	21	21	45,6
Configuração de Impressora	12	33	71,7
Sistema Operacional	6	39	84,7
Hardware	4	43	93,4
Erro do Cliente	3	46	100,0

Tabela 7.2 - Classificação de Problemas para Análise de Pareto

Como pode se observar na figura 7.3, a aplicação da análise de Pareto, evidencia que apenas alguns problemas são responsáveis pela maioria das ocorrências. Como benefício imediato, o suporte pode identificar quais são os fatores críticos no suporte do produto. No exemplo, verifica-se que os itens descrição de terminal e configuração de impressora são responsáveis por quase setenta e dois por cento (71,7%) das chamadas, o que merece um esforço especial para reduzir o índice de chamadas e conseqüentemente a qualidade do produto.

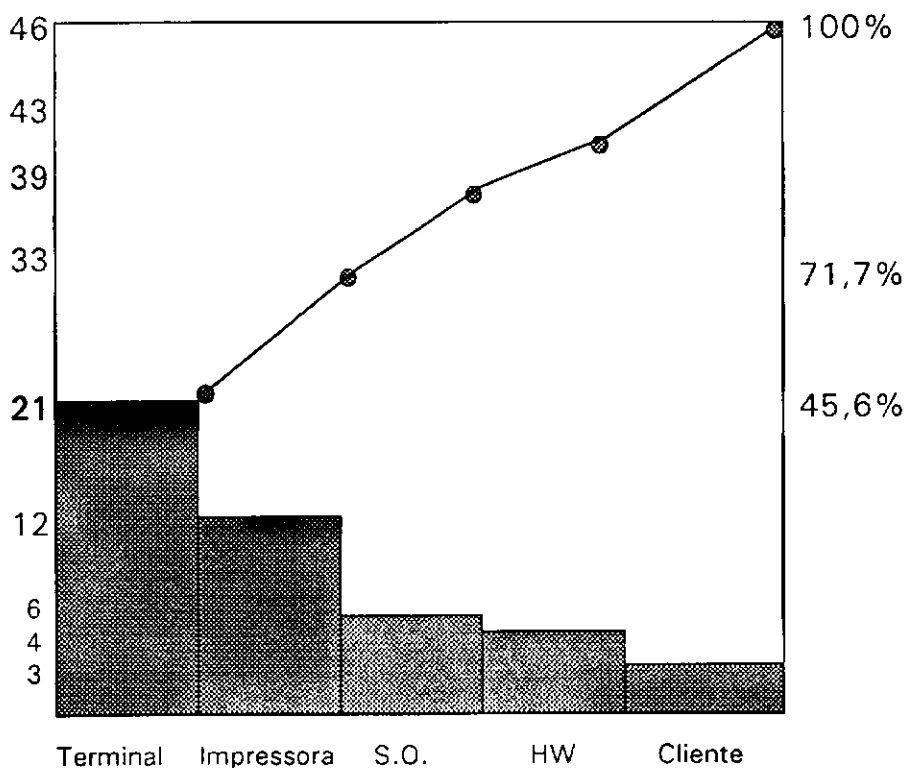


Figura 7.3 - Gráfico de Pareto para Identificação dos Problemas Críticos

2. Análise da eficiência de atendimento do grupo e de cada profissional individualmente.

A impressão inicial que se pode ter é que ações no sentido de avaliação da produtividade de um profissional podem gerar problemas internos na empresa. Apesar deste sentimento, experiências mostram que quando utilizadas para o benefício do grupo, informações de produtividade estimulam os profissionais no sentido de procurarem atingir metas até então desconhecidas.

A composição do banco de dados proposto permite que sejam obtidas algumas medidas que refletem a eficiência do grupo de suporte. Algumas delas são medidas simples que podem ser obtidas facilmente do banco de atendimentos:

- Quantidade de chamadas não resolvidas de imediato (pendentes) ;
- Tempo médio de resolução de chamadas (pendentes ou não) ;
- Tempo de atendimento individual e média do grupo.

Estas medidas servem para analisar tanto o rendimento do suporte quanto a demanda por atendimento. Em se tratando de avaliar especificamente a qualidade do suporte, a utilização de um índice que indique a eficiência de atendimento (IEA) pode ser levantada pela simples relação :

$$IEA = \frac{\text{número de chamadas resolvidas}}{\text{número total de chamadas}}$$

Esta relação, facilmente operacionalizada e entendida por todos, pode ser aplicada individualmente para cada elemento do suporte ou para o grupo como um todo. Funcionando como medida inicial de eficiência do

suporte, a melhoria do IEA pode ser entendido como elemento motivador, o que contribuirá decisivamente para a melhoria da qualidade do atendimento.

3. Análise de novos produtos no mercado

Muito se faz para colocar um produto no mercado. Produtores de software se esforçam para conceber um produto corretamente, desenvolvê-lo dentro de padrões de qualidade, disponibilizá-lo no momento exato conforme exigências impostas pelo mercado e finalmente ter condições de fazê-lo evoluir. Ao chegar no extremo desta cadeia de ações, é fundamental saber a direção que deve ser dada ao produto, o que fica difícil na ausência de informações sobre seu aceite pelo mercado.

Algumas das informações obtidas pelo suporte podem servir para o propósito de análise do software no mercado. Abaixo são relacionados relatórios acompanhados de breve descrição que podem ser de grande utilidade para traçar estratégias sobre o destino do produto. Sugere-se que seja dada atenção especial ao estudo do comportamento do software durante o seu primeiro ano de vida no mercado, ou de uma nova versão, pois neste período se concentram as principais descobertas sobre problemas e necessidades do produto, tendendo a se estabilizar posteriormente [Jones89].

- Quantidade de chamadas / mês

Medida simples, obtida no banco de atendimentos, que registra a variação de procura do usuário pelo suporte da empresa. Quer seja para evidenciar problemas, para comprar, para pedir serviços ou para qualquer outro fim, devem ser gerados relatórios separados por interesse para balizar alguma análise comparativa e ações corretivas.

- Volume de chamadas para manutenção

Fornece um relato, a partir da análise do banco de atendimentos, da demanda dos usuários por alterações no escopo do software que irão requerer esforço de desenvolvimento. Pode-se priorizar atendimento de solicitações a partir do volume de solicitações. (vide 8.3.2)

- Volume de vendas x volumes gerados x homologados

Pode-se obter uma posição real sobre a saída do produto para o mercado em termos financeiros. A partir de uma varredura no banco de produtos, pode se obter o percentual de vendas, o percentual gerado e o percentual colocado para homologação, que podem servir para analisar políticas de vendas e marketing da empresa, contribuindo para novas projeções de vendas que possam ser realizadas.

- Satisfação do usuário. O índice de satisfação do usuário deve ser sempre relatado por produto para que possa ser monitorado individualmente.

7.8 Alguns Objetivos do Suporte

A discussão sobre o suporte técnico deve ser valorizada em ambientes de produção de software comercial, pela papel ativo que o suporte deve exercer junto aos seus usuários e pelos benefícios que podem ser conseguidos com a prestação de serviços de qualidade aos usuários ou *prospects*. Assim sendo, este capítulo se deteve em parte a apresentar um ambiente apto a suportar agilmente seus usuários e algumas vantagens em otimizar o atendimento telefônico, um dos principais serviços do suporte.

Como o suporte funciona como uma organização embutida na empresa produtora de software, com profissionais e responsabilidades definidas, é necessário que sejam estabelecidos alguns objetivos a serem perseguidos. Os objetivos do suporte devem ser concebidos como elementos motivadores para que o grupo alocado e a gerência procurem oferecer melhorias contínuas nos serviços prestados. Dentre os objetivos podemos citar :

- Eliminar os problemas críticos detectados pela análise de Pareto ;
- Reduzir o tempo médio de atendimento ;
- Aumentar nível de satisfação do usuário ;
- Otimizar ao máximo o índice de eficiência de atendimento (IEA) ;
- Reduzir o volume de chamadas pendentes e o tempo de solução (estabelecer prioridades em função de criticidade, complexidade, status usuário e volume de chamadas) ; e,
- Tornar o suporte um centro de lucros, equilibrando as despesas do suporte pré-venda com as receitas do suporte pós-venda.

Capítulo 8. Manutenção

Este capítulo aborda a atividade de manutenção, delineando os principais conceitos envolvidos, a estrutura básica e procedimentos para se fornecer e avaliar manutenção de software.

São discutidos também, aspectos de gerenciamento de configuração, cuja existência se justifica pela constante necessidade de modificações no software e em seus componentes, que se intensificam durante a manutenção.

O capítulo finaliza tecendo considerações sobre portabilidade de software, uma questão relevante que trata da necessidade de se migrar software para novos ambientes computacionais, uma constante na manutenção de software UNIX.

8.1 Introdução

Efetivamente considerada pela prática e pela maioria dos autores que discutem o tema, a manutenção de software se caracteriza como a atividade mais crítica no ciclo de vida do software, concentrando a maior parcela de custos e esforço do corpo técnico durante a existência do software.

Pode-se considerar que o esforço de manutenção varia basicamente em função de três grandes fatores : ambientais, internos e externos (figura 8.1). Os fatores *ambientais* se constituem das características do ambiente de desenvolvimento de software. Os objetivos da organização, metodologia, métodos e ferramentas de trabalho, cultura de testes, linguagem de programação, adoção de práticas de controle de qualidade, experiência e motivação do corpo técnico são exemplos de fatores ambientais.

Outra classe de fatores, considerados *internos* ao software, incidem no nível de manutenção do produto de software. Nada mais são do que características inerentes ao próprio software ou a algum dos seus componentes, que dependendo do seu grau de existência no software interfere positiva ou negativamente na manutenção do software. Alguns fatores internos: complexidade, modularidade, reutilização, portabilidade e documentação.

Todo software tem requisitos de mudanças determinados por fatores *externos* à produção, assim considerados pois refletem as necessidades impostas pelo mercado depois da sua disponibilização. Este processo envolve além de requisitos de usuários, para atender melhor suas necessidades, decisões estratégicas do produtor para evolução e adaptação do software a novas realidades, aumentando a



Figura 8.1 - Fatores que Influenciam a Manutenção de Software

vida útil do seu software no mercado. Neste caso, podem ser considerados fatores externos: a adaptação a novas tecnologias (plataformas de hardware, sistemas operacionais, novos periféricos, etc), incremento de funcionalidade, análise da tendência dos concorrentes no mercado para manter seu produto atualizado.

A diversidade de fatores que incidem na manutenção de software reflete nos estudos já realizados sobre custos de software. Alguns autores apontam a manutenção como responsável por cerca de 60% dos custos totais do software [Pres87], enquanto que outros chegaram experimentalmente a um índice variando de 70 a 75% para sistemas de grande porte [Pari90].

Procurando definir conceitualmente manutenção de software, Longstreet[Long90] fornece uma abordagem genérica que caracteriza bem esta atividade :

Manutenção de software é a execução das atividades necessárias para manter um sistema de software em operação depois de aceito e colocado em produção. Se caracteriza pelo conjunto de atividades que resultam na mudança do produto originalmente aceito. Consistindo estas mudanças de correção, inserção, deleção, extensão e melhoria do produto original.

No contexto deste trabalho, manutenção será considerada em função das atividades necessárias a uma software house para alterar o software, a partir do momento em que este foi liberado para o mercado de usuários UNIX.

8.2 Mudanças e suas consequências

Ao se discutirem os elevados custos da manutenção de software e seu impacto nas organizações de desenvolvimento de software, impreterivelmente não pode deixar de constar na pauta a *flexibilidade e constância de mudanças* .

Diferente de produtos manufaturados, o software é um elemento passível de mudanças significativas mesmo depois de comercializado. A alteração de programas do software fica viável com a simples utilização de um editor de texto que acesse o programa fonte. Esta característica por vezes prejudica a produção de software, que se ampara na possibilidade de mudanças futuras, produzindo software de baixa qualidade .

Na realidade, a possibilidade de modificação de software durante ou depois da sua produção, deve ser aproveitada com disciplina. O erro clamoroso no processo é que o software não pode ser simplesmente concebido como um conjunto de programas ou rotinas facilmente modificáveis. Deve se conscientizar de que existe uma integração e interdependência entre os diversos itens do software (documentação, estrutura de dados, projetos, bibliotecas, dentre outros) e analisar principalmente o *impacto que estas mudanças* representam para o software.

Alguns problemas são tidos como clássicos na atividade de manutenção, tais como: documentação inconsistente, programas altamente pessoais, incompreensão de itens de software desenvolvidos por terceiros, programas desestruturados (*spaghetti*), programas rígidos que não acomodam mudanças facilmente (não utilizam modularização ou classes de objetos, por exemplo), e propagação de erros em programas que sofreram manutenção.

Problemas como estes poderiam ser pelo menos amenizados caso o ambiente de desenvolvimento utilizasse estratégia de *projeto defensivo*, onde se busca antecipar mudanças no software e com o apoio da engenharia de software e de técnicas e métodos de gestão de qualidade teríamos um maior grau de manutenibilidade¹ do software.

Sendo bastante objetivo, além dos altos custos já citados, o alto nível de manutenção pode:

- gerar *usuários insatisfeitos* com o produto e podem eventualmente decidir abandonar o produto ;

¹Manutenibilidade de software é conceituada, segundo [Iso 9126], como uma das subcaracterísticas de qualidade de software que se refere ao esforço necessário para fazer modificações no software.

- denegrir aos poucos a *imagem da empresa* no mercado, a qual pode ter demorado anos para ser criada;
- mobilizar parte ou até mesmo totalidade da equipe técnica que poderia estar sendo utilizada em outras atividades mais produtivas, como por exemplo, desenvolvendo novos produtos (*backlog*).

Os próximos itens exploram a estrutura e procedimentos adotados para manutenção de software UNIX, baseados em depoimentos práticos sobre a rotina de grupos de manutenção e com algumas sugestões da literatura sobre o assunto.

8.3 Estruturando a Manutenção em UNIX

Por ser a atividade do ciclo de vida do software de maior duração, a manutenção de software necessita planejamento e recursos próprios para sua subsistência. Além do mais, precisa ser concebida como qualquer outra atividade do ciclo, com tarefas bem definidas que possam ser executadas ordenadamente.

Observa-se na realidade de várias organizações, que a manutenção é tida como ônus, concebida unicamente como uma indesejada forma de aumentar a carga de trabalho e despesas da organização. Pode ser até que isto seja inevitável. Entretanto, a manutenção é essencial para a evolução do produto e garantia de sua continuidade no mercado, o que pode gerar benefícios para a empresa num momento posterior.

A figura 8.2 ilustra uma sequência de procedimentos que caracteriza todo o processo de manutenção de software em ambientes UNIX. No decorrer do texto serão explicados os procedimentos, com as principais atividades sublinhadas.

8.3.1 Recebimento de Solicitações

O processo ilustrado na figura 8.2 se inicia com a chegada de solicitações de manutenção para o gerente do produto em questão. As solicitações de modificação do software, causado por algum dos fatores citados anteriormente (ambientais, internos ou externos), podem ser requisitadas ou sugeridas pelos usuários, pelo suporte, pela equipe de desenvolvimento ou até mesmo estrategicamente pela empresa.

As solicitações chegam aleatoriamente ao gerente do produto, podendo significar erro no produto, solicitações de melhorias internas (performance, segurança), necessidade de se migrar para um novo ambiente (transporte) ou ainda solicitação de incremento de funcionalidade.

Em seguida, o gerente do produto juntamente com o corpo técnico ou a gerência, se for o caso, analisa tecnicamente a possibilidade de mudanças e seus efeitos colaterais no software e no ambiente de desenvolvimento. Neste processo, são considerados principalmente os custos, o prazo necessário para as alterações, quantidade de recursos humanos, alteração de documentação, projeto de dados e outras funções do sistema.

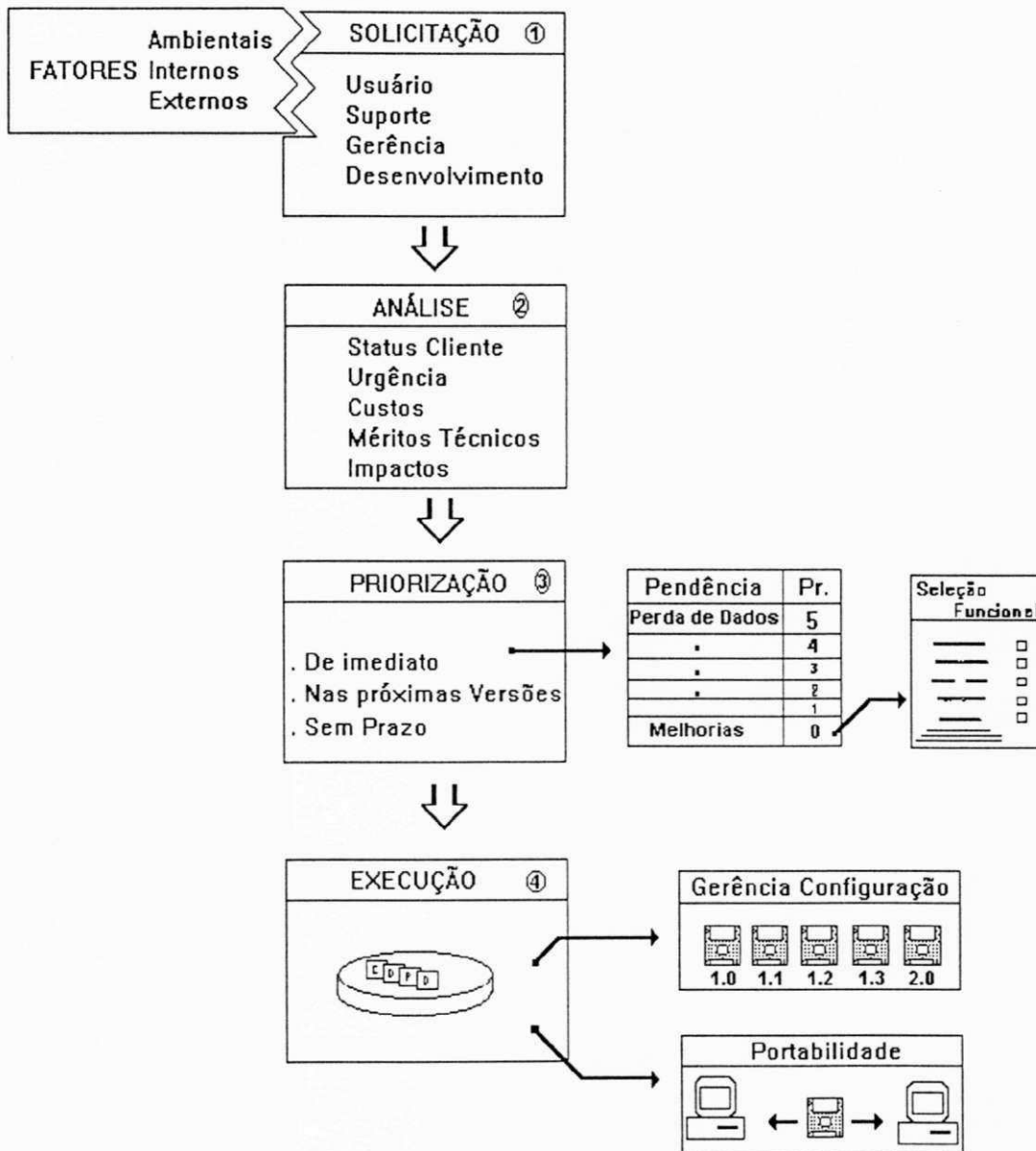


Figura 8.2 - Organização da Estrutura de Manutenção de Software

O término da análise culmina com a produção de um documento de pedido de manutenção contendo a especificação completa da solicitação recebida, gerando assim o primeiro indício de manutenção planejada. Este processo deve ser apoiado por ferramentas automatizadas, de forma a agilizar o cadastro e acesso as pendências do produto.

8.3.2 Priorização de Solicitações

O gerente do produto, de posse das pendências de manutenção devidamente especificadas, deve procurar estabelecer critérios de atendimento de acordo com a urgência do pedido, disponibilidade de recursos e com a estratégia para a evolução do produto. O primeiro passo para organizar a demanda é definir prioridades para as pendências e que tipo de encaminhamento deve ser dado para cada prioridade.

Existem alguns trabalhos publicados sobre priorização das solicitações em função da sua criticidade [Jone89], [Evan87], sendo este último particularmente interessante pelas considerações sobre as possibilidades de solicitação, descritas na tabela 6.1 :

Descrição	Prioridade
Perda de dados	5
Queda do sistema	4
Erros dificilmente contornáveis ²	3
Erros facilmente contornáveis	2
Problema de documentação	1
Solicitação de melhorias	0

Tabela 8.1 - Priorização de Solicitações

² Refere-se aos tipos de erros que impactam na reprogramação completa do código envolvido, ou de diversas funções ou de módulos do software.

A classificação apresentada na tabela 8.1, se aproxima bastante da realidade da rotina adotada por uma empresa de software básico.

Basicamente, o gerente do produto deve estabelecer uma fila de atendimento para as pendências do seu produto, ordenadas em ordem decrescente de prioridade, ou seja, a pendência com maior prioridade é atendida primeiro. O esquema apresentado prioriza as pendências com maior urgência que necessitam de pronto atendimento, sendo as demais atendidas em função da disponibilidade de recursos e dos interesses da empresa e dos seus usuários.

Como pode ser verificado, este esquema prioriza o atendimento de problemas (erros, documentação, etc...) detectados durante o uso do software, colocando intencionalmente as propostas de melhorias com prioridade zero na tabela. Isto significa que deve-se preocupar primeiro em manter um produto confiável no mercado, com o menor número de erros possível, para depois aplicar melhorias.

Entretanto, não só de reparos ou ajustes vive o software. Mesmo tendo que atender aos problemas detectados durante o uso por questões de prioridade, o produtor do software deve procurar sincronizar esforços para poder evoluir e atualizar seu produto, em prol da continuidade de sua aceitação no mercado.

8.3.3 Seleção Funcional

A realidade mostra que não se pode, nem se deve, embutir de imediato no produto, todas as melhorias funcionais (prioridade 0) solicitadas pelo mercado e pelos projetistas do software. O mercado de software exige dinamismo para a atualização de produtos, ficando a

cargo dos produtores elaborar uma receita para conjugar esta diversidade de fatores envolvidos, supra citados.

Algumas empresas, entretanto, têm procurado respostas sobre o que fazer com seus produtos, através do envolvimento daqueles que realmente utilizam ou distribuem o software no mercado. Isto porque, os usuários conhecem melhor do que ninguém o nível de adequação do produto as suas necessidades, e os distribuidores, por outro lado, possuem o sentimento de mercado e aceitação do produto.

O esquema encontrado pelos grandes produtores, dentre eles a Lotus e a Microsoft, para consultar o mercado sobre a evolução funcional dos seus produtos, foi o sistema de votação. Basicamente, o produtor solicita que seus usuários e distribuidores decidam dentre um conjunto de novas funcionalidades, aquelas que acreditam contribuir para a melhoria do produto, seja para o uso (usuário) ou para a revenda no mercado (distribuidor).

Ao analisar os sistemas de votação adotados, verifica-se o *envio de formulários de votação* para o mercado (usuários e distribuidores), seguidos da *coleta de dados, análise* e posterior *divulgação* das funcionalidades que serão incluídas nas próximas versões e releases do software.

O formulário de votação deve ser claro e que estimule o preenchimento. Tarefa não muito fácil, que pode ser amenizada ao se esclarecer ao usuário que ele estará contribuindo para o futuro do produto, e portanto, melhorando as suas condições de trabalho. A figura 8.3 apresenta um modelo que pode ser utilizado para votação de usuários.

Formulário de Votação - LightText Versão 9.9

A Exemplix software gostaria de contar com a sua rápida e valiosa colaboração no sentido de responder este formulário de votação sobre o LightText. Nosso objetivo é colher suas preferências e necessidades a serem incluídas na versão 9.9 , ou em futuras versões do LightText.

Agradecemos antecipadamente sua participação, com a certeza de que estaremos trabalhando juntos para a melhoria do produto.

Para votar, você dispõe de cinco pontos que devem ser distribuídos entre as funcionalidades de acordo com suas preferências. As opções que acumularem maior número de pontos serão prioritariamente incluídas na próxima versão do LightText:

1. Interface mais amigável (p.e., uso de mouse e janelas)	<input type="checkbox"/>
2. Corretor Ortográfico	<input type="checkbox"/>
3. Pesquisa por palavra-chave	<input type="checkbox"/>
4. Esquema de proteção com senhas de acesso	<input type="checkbox"/>
5. Mala-Direta	<input type="checkbox"/>
6. Tabela com recursos de planilha eletrônica	<input type="checkbox"/>
7. Importação de arquivos de outros editores	<input type="checkbox"/>
8. Controle de versões	<input type="checkbox"/>
9. Editor de desenhos	<input type="checkbox"/>
10. Gerador de gráficos	<input type="checkbox"/>

Comentários (anexar, se necessário) :

Empresa : _____

Nome : _____

Função : _____

Contato : _____

(Telefone, Fax, Endereço, E-Mail)

Figura 8.3 - Formulário de Votação para Evolução de Software

Existem vários critérios de votação que podem ser adotados pelo produtor. No caso do modelo sugerido na figura 8.3, além de selecionar as funcionalidades mais desejáveis para a nova versão do produto, o usuário prioriza as mais importantes atribuindo-lhes um maior número de pontos, dentre os cinco pontos disponíveis. Por exemplo : o usuário pode preencher o formulário acima com quatro votos para a funcionalidade 1 (interface amigável) e apenas um voto para a funcionalidade 4 (mala-direta), colocando assim pesos diferentes para as suas necessidades, dentre os cinco possíveis pontos (votos).

A evolução funcional necessita ser balanceada em virtude da análise de algumas variáveis de decisão, que aliadas ao resultado da apuração obtida com o sistema de votação, fornecem uma receita sobre a evolução do produto. Podem ser citadas as seguintes variáveis de decisão, como sendo fundamentais para determinar a evolução de um software:

- disponibilidade de recursos (mão-de-obra, tecnologia e capital) ;
- tendências do mercado (p.ex., interfaces amigáveis, novas plataformas) ;
- movimento dos concorrentes ;
- estratégia da empresa para o produto ;
- solicitações e peso dos clientes (teste beta, votação, suporte, etc...).

A priorização de pendências, entendendo-se como pendência os ajustes, as correções e funcionalidades selecionadas, possibilita que o gerente do produto planeje a evolução do produto ao longo do tempo de acordo com os problemas e as necessidades surgidas. Então, é

fundamental que se defina explicitamente qual o período de correção das pendências, podendo ser dados basicamente os seguintes encaminhamentos :

1. Pendência a ser atendida imediatamente ;
2. Pendência a ser atendida nas próximas release ou versão ;
3. Pendência sem prazo de atendimento.

8.3.4 Ativando o Processo de Mudanças

Após serem definidas as modificações (inovações, ajustes e correções) no produto, com suas respectivas documentação, prioridade e prazo de atendimento, resta então ativar o grupo de manutenção para atender as pendências segundo o planejado.

Alguns procedimentos de marketing também devem ser realizados para comunicar o mercado sobre o destino do produto, gerando uma expectativa de mercado, a ser concretizada posteriormente. Os distribuidores, principalmente, cientes das novas características que serão embutidas na próxima versão, iniciam o trabalho de divulgação do "novo" produto, identificando oportunidades de negócios junto a seus clientes. Os usuários que participaram da votação de funcionalidade também devem ser informados sobre o resultado alcançado.

O atendimento às solicitações de manutenção reinicia o ciclo do software. Geralmente, o "novo" ciclo do software assume menores proporções, o que varia de acordo com a complexidade da modificação solicitada e seu impacto no software. Não existem termos de

comparação entre o esforço necessário para se reparar um erro de programa, e o acréscimo de novas funcionalidades ao produto.

Independente da dimensão da modificação tratada, o gerente de produto deve estimular para que a manutenção seja conduzida como um projeto completo. Deve-se analisar as modificações necessárias em todas as atividades, desde a atualização de especificações e projetos, passando por recodificação, testes de aceitação e integração até a condução dos procedimentos de empacotamento e disponibilização no mercado.

8.3.5 Liberação

O ciclo de manutenção de software, explorado neste ítem sob o prisma organizacional, culmina na liberação do software para distribuição no mercado. Modificações oferecem perigos para o produtor de software, visto que podem alterar a concepção original do produto e incorrer em novos erros derivados de análise descuidada dos impactos causados. Por isto, este processo deve ser marcado por rigoroso controle de qualidade do produto. Revisões no produto são recomendadas em todas as etapas, acompanhadas de testes, se possível com a aprovação de usuários.

8.3.6 Documentação

Todo o processo de manutenção precisa ser devidamente documentado, seja internamente, atualizando os componentes do software, ou externamente, notificando os seus usuários. A partir do momento em que o software passa por diversas modificações e novas versões são colocadas no mercado, é essencial que o usuário seja

devidamente notificado sobre as inovações do software ou sobre a nova forma de operação.

A documentação, entretanto, requer dedicação de recursos para sua atualização e fica praticamente inviável se refazer e distribuir manuais de usuários completamente a cada modificação no produto, pelo menos para as pequenas alterações (*release*³). Uma prática do mercado tem sido o envio de documentação adicional, até mesmo na mídia em que vão as alterações de código, para os usuários que estão migrando para uma nova release do produto.

Logo abaixo verificamos uma reprodução do arquivo texto que acompanha a atualização do produto LTDhs da Infocon ®. Segundo informações da empresa, este arquivo parcialmente reproduzido (figura 8.4), foi gravado logo abaixo do diretório de instalação com o formato ASCII por permitir a leitura sem distorções no terminal de vídeo.

A possibilidade de confecção de um novo manual do usuário e da documentação que o acompanha está vinculada ao nível de compreensão e aceitação que se está tendo com a documentação remetida ao usuário. Antes que a documentação comece a se tornar confusa demais para seu bom entendimento, está na hora de elaborar nova documentação.

Geralmente, o lançamento de uma nova versão do software é acompanhado de uma nova documentação, pois envolve a adição de novos recursos e soluções para os usuários. A nova documentação se justifica não somente pelo acompanhamento dos aspectos técnicos e

³O termo "release" caracteriza o lançamento no mercado de modificações no software, que não alteram substancialmente as características do produto. Geralmente estas mudanças estão ligadas a ajustes e correções do software, ou até mesmo ao incremento de pequenas funcionalidades.

melhorias na sua forma, mas também pela aquisição de uma nova imagem, geralmente dada ao produto pelo seu pessoal de marketing.

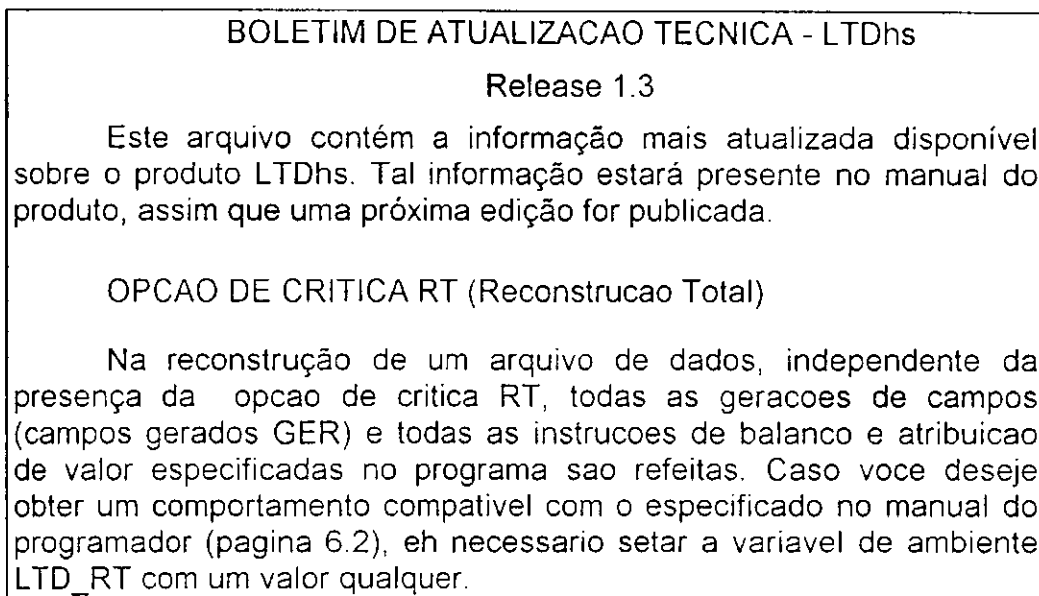


Figura 8.4 - Exemplo de Arquivo de Documentação para Nova Release

O sucesso na atualização da documentação, tarefa não muito atraente para o corpo técnico, depende muito da motivação do grupo e da cultura da organização, o que pode ser reforçado com o uso de ferramentas e de metodologias, principalmente nas atividades críticas de especificação e projeto.

8.4 Gerência de Configuração de Software (GCS)

Um dos passos importantes na evolução do software é identificar as versões dos seus componentes, controlando a devida disseminação para os usuários. Isto é feito através do gerenciamento de configuração de software (CGS).

A atividade de gerenciamento de configuração do software não se restringe a manutenção de software, sendo necessária em todas as

fases do processo. Na verdade, a gerência de configuração é uma atividade de acompanhamento, independentemente de que fase esteja, sendo puramente uma atividade de controle de qualidade durante todo o ciclo de vida [Press87].

Este tópico foi abordado propositalmente neste estágio do trabalho, pois a gerência de configuração, apesar de ser uma atividade de acompanhamento na produção de software, é necessária principalmente quando o produto passa pelo processo de manutenção, conforme descrito nas seções anteriores.

A implementação do controle de gerência de configuração numa organização, entretanto, deve ser amparada por uma estrutura formal, mesmo que pequena dentro da empresa, para organizar a discussão sobre as possíveis modificações nas versões oficiais do software, ou em qualquer um de seus componentes.

A figura do *gerente de configuração*, o qual pode até ser um dos componentes do grupo de controle de qualidade ou mesmo o gerente do produto, deve ser concebida como autoridade de interface entre as pessoas que interagem com o software, respondendo pela eficiência das mudanças na configuração dos produtos.

Os principais conceitos e ferramentas de GCS em UNIX, são abordados mais extensivamente nas seções seguintes.

8.4.1 Funções da GCS

Para se delinearem as atividades e mecanismos que estão envolvidos no gerenciamento de configuração de software, nada melhor que seguir a orientação das normas internacionais para gestão de qualidade de software. Basicamente, segundo estas diretrizes, o

software deve dispor de mecanismos que permitam a *identificação*, o *controle* e o *rastreamento* das versões oficiais do software disponibilizado ao usuário, preservando-se as versões anteriores **[Iso9003]**.

Vale ressaltar que o gerenciamento de configuração refere-se à execução de mecanismos que gerenciem os componentes que constituem o software e suas respectivas versões, assim denominados de *configuração de software*. São os seguintes os componentes do software que devem ser incluídos na gerência de configuração:

- Programas ;
- Bibliotecas de programas (compartilhadas ou não) ;
- Ferramentas utilizadas para desenvolvimento e operação do software ;
- Documentação em geral (p.e., manual do usuário, especificação) ;
- Interfaces com outros componentes de software e hardware.

As atividades de *identificação* e *rastreamento* na gerência de configuração, dizem respeito a necessidade de se identificarem unicamente e localizar os componentes citados acima que precisam ter suas versões controladas. Então, cada um destes componentes necessita ter as seguintes informações **[Pres87]** para diferenciá-lo internamente no ambiente de produção :

- Nome do Produto ;
- Número da Versão e Release ;
- Tipo do Componente (p.e., documento, código fonte) ;

- Objetivo (p.e., especificação de requisitos) ;
- Data de liberação do componente ;
- Código associado para identificação única (p.e., nome de programa, número de documento ou relatório técnico) ;
- Tipo de máquina UNIX, se for o caso .

A partir das solicitações atendidas pelo grupo de manutenção, devem ser estabelecidos procedimentos de *controle* para identificar, documentar, revisar e autorizar as modificações nos componentes de software [Iso9003]. Para se atingir estes objetivos, produtores UNIX contam com ferramentas de GCS embutidas no ambiente de desenvolvimento, as quais serão discutidas na seção 8.4.3.

8.4.2 Versionamento

Antes de se avaliar exatamente os mecanismos de gerência de configuração do software, é interessante ter uma visão sobre a atribuição de versões ao software e seus componentes.

Cada componente de software possui suas próprias versões e *releases*, dependendo da quantidade e profundidade das mudanças realizadas. O software por sua vez, também possui um número de versão e release associado em função de cada lançamento no mercado. Juntando as peças, verifica-se que cada versão de software pode ser composta de diferentes versões de componentes, segundo ilustrado na figura 8.5.

Segundo o exemplo, verificamos que a versão 1.2 do software A se compõe do programa P na sua versão 1.2 e da biblioteca B versão

2.0, além dos outros componentes não ilustrados na figura, tais como projeto, interface e especificação.

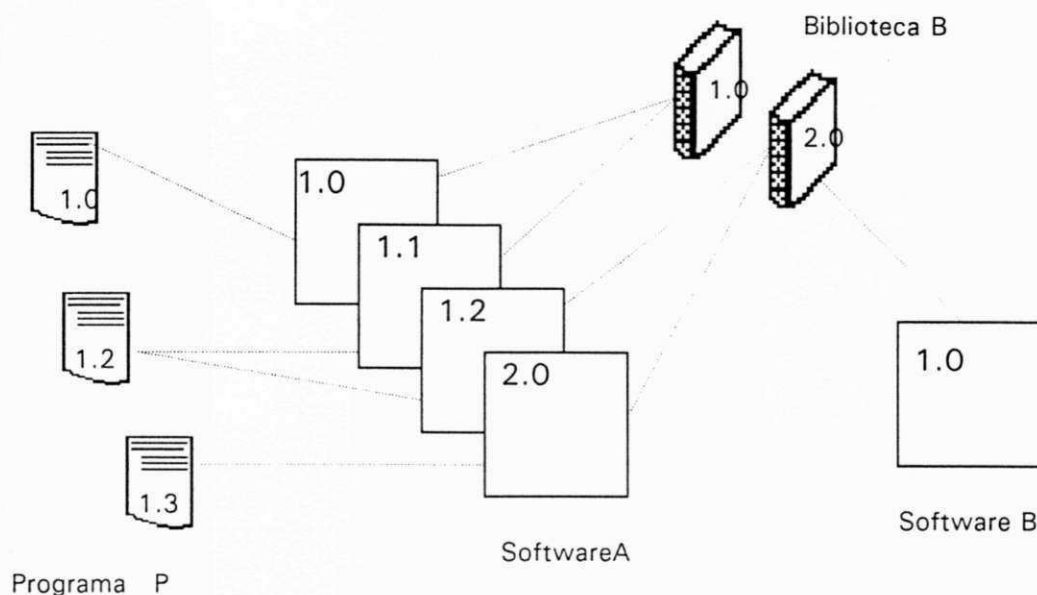


Figura 8.5 - Versões do Software e seus Componentes

Ainda na figura 8.5 pode-se perceber o problema de compartilhamento da biblioteca 2.0, que atende simultaneamente ao software A e ao software B. Por isso, cada versão do software deve saber exatamente quais são as versões correspondentes dos seus componentes e controlar as modificações nas bibliotecas compartilhadas de forma a garantir a compatibilidade entre o software e suas bibliotecas associadas.

Algumas ferramentas de GCS do tipo "Source Code Control System" (SccS), "Revision Control System" RCS ou "Software Configuration Management for Heterogeneous LAN-Bases Development" (PVCS) podem ser utilizados para auxiliar o controle do compartilhamento das bibliotecas. Esta questão será ampliada na seção

8.4.3, onde serão sugeridas a utilização de ferramentas de controle juntamente com procedimentos para compartilhar bibliotecas entre vários software.

8.4.2.1 Identificando Versões e Releases

A atribuição de versões e releases ao software varia de empresa para empresa. Entretanto pode-se chegar ao seguinte esquema de numeração ao se observar o mercado de uma maneira geral :

Versão. release. release interna (opcional)
--

Exemplo : Software SCO Unix 2.1 significa versão 2 release 1

Esta pequena regra a ser seguida, fornece um significado para o produtor e para o usuário que já compreende este linguajar. Geralmente, quando o software é lançado no mercado, o fabricante atribue uma versão inicial (p.e., SCO UNIX 1.0) que serve como base para modificações. A partir deste instante, quaisquer modificações realizadas na última versão lançada implicam na mudança de numeração.

No caso de pequenos ajustes e mudanças no software, o procedimento convencional é simplesmente lançar uma nova release no mercado, incrementando o valor da última release (p.e., SCO UNIX 1.1). Quando se tratam de modificações substanciais em termos de funcionalidade, inovações no produto ou até mesmo de estratégias de marketing para "reaquecer" um produto, caracteriza-se o surgimento de uma nova versão. Neste caso, o procedimento normal é acrescentar uma unidade a versão anterior e zerar a release (p.e., SCO UNIX 2.0).

As releases internas citadas no método de numeração, são utilizadas exclusivamente para liberar para o mercado o software após seus erros serem consertados. Vale salientar que alguns fabricantes utilizam esta versão para liberar alguns pacotes de mudanças ou ajustes para seus usuários, não seguindo necessariamente a padronização aqui mencionada.

8.4.3 Ferramentas de GCS

As ferramentas para gerenciar configuração de software surgiram como resposta à demanda de administração de vários produtos e suas versões simultaneamente num mesmo ambiente de produção de software. As ferramentas possuem propósitos gerais de controlar concorrência de acesso a códigos de programa e bibliotecas, criar, armazenar e recuperar versões de um programa e finalmente preservar a versão original do software, gerando versões anteriores automaticamente [Ghez91].

Atualmente existem várias ferramentas de GCS disponíveis no mercado, mas uma dentre elas tem sido largamente utilizada, sendo fornecida junto com o sistema operacional UNIX, como parte do ambiente de desenvolvimento. Esta ferramenta, chamada Sistema de Controle de Código Fonte (SccS), apresenta um funcionamento simples (figura 8.6) basicamente controlando versões de arquivos textos (os quais no UNIX podem ser utilizados para conter código fonte, arquivos de dados, manuais, bibliotecas, massas de testes, dentre outras possibilidades).

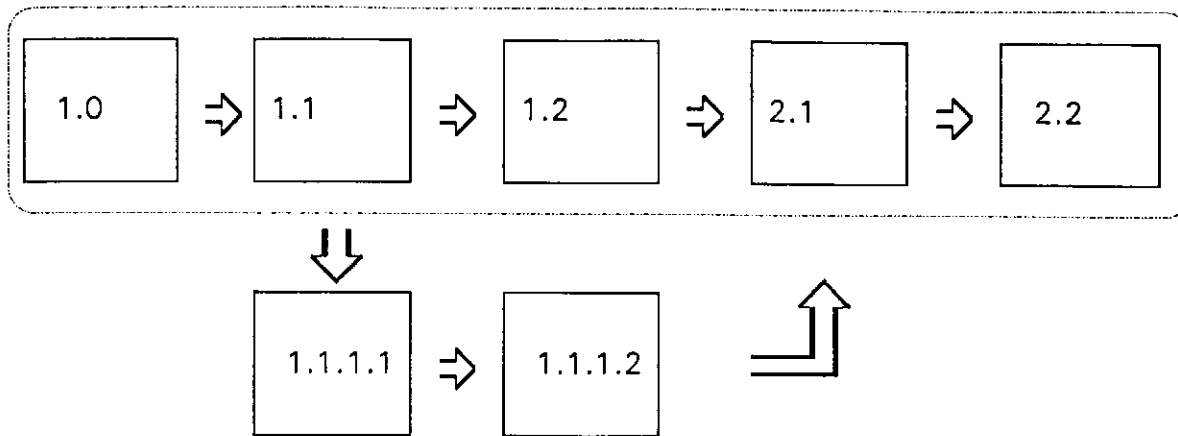


Figura 8.6 - Versionamento Utilizando SccS

A utilização do SccS é feita de acordo com o esquema de numeração de versões referenciado no item anterior, podendo seguir em duas direções. A primeira delas, ilustrada na figura pelo retângulo tracejado, permite uma evolução sequencial entre as versões, marcada pelo simples incremento de releases (de 1.1 para 1.2) para pequenas mudanças, e versões (de 1.2 para 2.1) para mudanças mais significativas.

A outra direção diz respeito à criação de ramificações (versão 1.1.1.1 na figura), abrindo a possibilidade de se trabalhar simultaneamente em mais de uma versão do software. Esta opção se adequa, por exemplo, ao desenvolvimento de testes paralelos ao produto e ao estudo de diferentes possibilidades de implementação e de funcionalidade a serem incluídas no software. As versões em paralelo podem ser unificadas pela equipe de manutenção num momento posterior.

O SccS oferece vantagens de otimização quanto ao armazenamento físico de versões de software. A ferramenta armazena a primeira versão do software em disco, e para cada nova versão, são armazenadas apenas as modificações necessárias para se gerar esta

última versão a partir da(s) anteriore(s). Estas modificações (chamadas de deltas) podem ser criadas, modificadas e fechadas para cada arquivo texto tratado.

Dentro do contexto de manutenção discutido para o ambiente UNIX, a utilização da ferramenta SccS pode ser conduzida de acordo com os seguintes procedimentos, baseados na estrutura ilustrada na figura 8.7 :

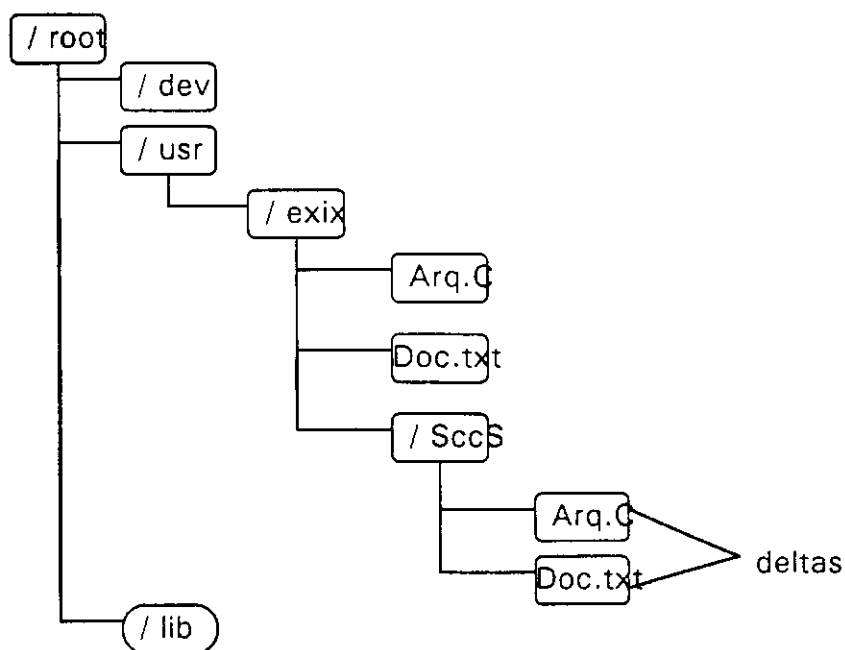


Figura 8.7 - Estrutura de Diretórios para Utilização do SccS

1. Para criar a primeira versão dos arquivos textos (documentos, programas, massa de testes, etc..) relativos a uma aplicação, deve-se a partir do diretório da aplicação (**exix**) ativar o comando *admin* do Unix. Automaticamente, o SccS cria um diretório abaixo do diretório corrente chamado de **Sccs**, e associa aos arquivos textos, o delta 1.0 ;
2. No ato de criação de qualquer delta, explicitar com comentários, as modificações a serem feitas, data de abertura,

prazo de trabalho, responsável e versão correspondente do software no mercado;

3. Após terem sido realizados os testes, a integração e a documentação, deve-se fechar o delta de trabalho antes de liberar o software para o usuário. Isto é feito no diretório da aplicação (**exix**) utilizando-se o comando *delta* ;
4. Quando necessário, podem ser feitas alterações em algum delta fechado, através do comando *get* do Unix.

Mesmo com o SccS, o produtor deve ter cuidados no tratamento de versões do software, principalmente no tocante aos componentes compartilhados por vários softwares, como é o caso das bibliotecas. O SccS fornece versionamento de arquivos textos, ficando a cargo do gerente de produto, atribuir que componentes fazem parte da versão do software. O tratamento convencional é iniciar um novo delta para todos os códigos e componentes do software a cada nova modificação.

Infelizmente isto não pode se verificar para as bibliotecas, pois possuem evolução diferenciada em função de servirem vários produtos. Neste caso, é aconselhável que as bibliotecas estejam fechadas no SccS quando forem vinculadas a um produto. Sendo necessário também, que se coloquem na descrição de cada versão do software no mercado, as versões das bibliotecas no SccS que o software utiliza.

Como veremos a seguir, o SccS desempenha papel de controle importante também no transporte de software, uma atividade de manutenção de produto com características diferentes das solicitações convencionais de manutenção em UNIX.

8.5 Portabilidade

Delineada pela ISO [Iso 9126] como uma das características de qualidade desejáveis para software, a portabilidade indica a facilidade de se mover produtos de software de um ambiente computacional para outro, sem a necessidade de reescrevê-los ou de realizar grandes esforços para adaptá-los à nova realidade.

O produtor de software para a plataforma Unix em particular, convive com a necessidade de transportar software entre máquinas com diferentes versões do Unix para aumentar seu mercado potencial. A tarefa de transporte requer cuidados devido às especificidades das diversas implementações do sistema operacional Unix (por exemplo, Bekerley, AT & T, System V, SCO, dentre outras).

O transporte representa para o produtor o início de um ciclo de atividades que requer sistematização e controle da qualidade dos resultados. Este processo, ilustrado na figura 8.8, visa uniformizar o código para todos os ambientes em que opera, simplificando futuras atividades de manutenção.

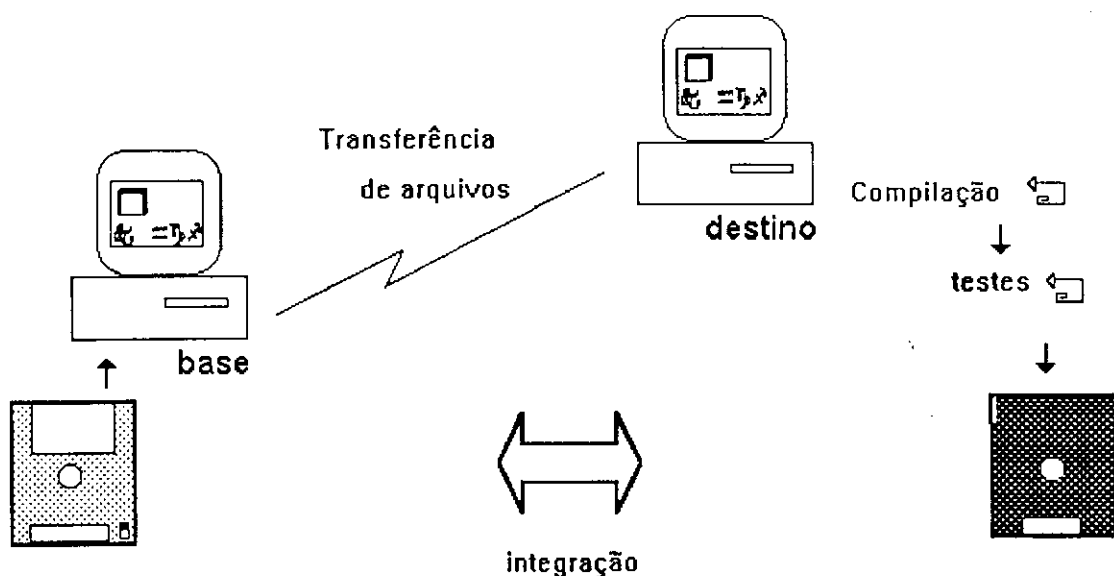


Figura 8.8 - Transporte de Software

O processo de transporte consiste em colocar uma versão do produto em outros ambientes Unix, aproveitando o código desenvolvido inicialmente numa determinada máquina, chamada de máquina base. Os produtores devem ajustar o seu software, mantendo apenas um código fonte que funcione em todas as máquinas atendidas pelo produto, minimizando assim os custos com manutenção.

A nível de mercado, o porte de um software normalmente representa o lançamento de uma nova versão para um ambiente ainda não explorado pelo software. As decisões de porte podem ser decorrentes da concorrência, da evolução da massa de usuários de um determinado fabricante, do atendimento a usuários de peso ou de tendências de mercado.

8.5.1 O transporte

Para um produtor que intenciona ter seus produtos funcionando em diversos ambientes computacionais, o passo inicial deve ser a definição de uma máquina base onde reside o software original. Esta máquina é o ponto de partida de qualquer manutenção que venha a ser realizada no produto. É na máquina base onde serão realizadas todas as modificações e testes necessários, para depois serem repassados aos outros ambientes.

Definida a máquina base do produtor, o processo de transporte se inicia com a seleção dos componentes que serão transportados para a máquina destino, dentre eles estão o código fonte, as bibliotecas, descrições de terminais e procedimentos de instalação. Ao serem selecionados os componentes de transporte, deve-se garantir que estes não serão modificados na máquina base durante o transporte. Para

tanto, o gerente de configuração deve fechar a versão (delta) do produto com o gerenciador de configuração (SccS). Desta forma, o projetista responsável pelo transporte assegura uma cópia na máquina base compatível com a versão na máquina destino.

Definidas as versões dos produtos a serem transportadas, deve-se então transferir uma cópia do código fonte da máquina base para a máquina destino. Esta função pode ser realizada através de algum mecanismo de comunicação compatível às duas máquinas, do tipo disquete ou fita magnética, mas pode-se utilizar preferencialmente algum serviço de transferência direta de arquivos (p.ex., o utilitário uucp entre máquinas Unix).

O trabalho de transporte começa a se particularizar quando da necessidade da compilação do código instalado na máquina destino. Nem todas as máquinas Unix possuem os mesmos recursos e diretivas de compilação. Por isso, o trabalho de compilação corriqueiramente exige ajustes que devem ser devidamente documentados de forma a facilitar futuros transportes entre as máquinas Unix em questão. A Infocon Tecnologia, por exemplo, com a experiência adquirida em transporte durante anos, montou um esquema de geração automática de arquivos do tipo make, com as particularidades de compilação de cada máquina envolvida no transporte, montando basicamente um roteiro de compilação ao qual o código fonte será submetido.

Uma vez compilado, exaurindo-se os erros de compilação, as atenções se voltam para verificação do comportamento do código fonte no novo ambiente. Se justifica neste momento, a realização do teste alfa. Questões de desempenho no novo ambiente, teste de novos recursos funcionais incorporados, robustez do software, e

funcionalidade de uma maneira geral, devem ser investigadas. Os resultados do teste alfa devem ser documentados para registrar a resposta do software no novo ambiente de trabalho.

No caso de serem necessários ajustes no código na máquina destino, estes não podem impactar diretamente o código na máquina base ou em qualquer outra máquina envolvida.

Resolvidos os problemas de compilação e ajustes no código fonte, garantindo-se que o software funciona corretamente na máquina destino, o projetista deve buscar a portabilidade total do software, marcando assim a atividade de integração do código transportado com aquele original, na máquina base. Para tanto, o projetista deve concentrar esforços no sentido de construir um único código fonte que seja executável tanto na máquina base quanto na máquina destino. O comando *diff* do Unix auxilia nesta tarefa, já que aponta para as diferenças entre arquivos, sendo bastante útil para garantir que os arquivos que contenham os códigos fontes não estejam diferindo.

Uma vez integrado o código fonte, o produtor pode, por exemplo, preparar um arquivo make ou do tipo batch executável ("script"), com as particularidades de compilação de cada máquina. Obtem-se assim um código fonte padrão para todas as máquinas. Sem dúvida, isto facilita o trabalho de reprodução de cópias para distribuição, bem como o próprio trabalho de manutenção do software.

No apêndice D têm-se um exemplo de um código fonte padrão, que foi integrado para atender a diferentes ambientes. A prática de programação mostrada no apêndice, coloca o controle dos trechos de código a serem compilados e ligado-editados, embutido no próprio

programa, em função de variáveis inicializadas na compilação (pré-processor).

Após serem realizados os ajustes, o projetista juntamente com o gerente de configuração, deve ter a preocupação de alterar a versão interna do produto na máquina base, caso tenham sido realizadas alterações nos seus arquivos. Neste caso, uma nova versão no SccS deve ser aberta para o processo de integração, ficando assegurada a portabilidade por completo.

O último procedimento necessário para terminar o processo de transporte de software diz respeito a geração da Cópia Mestre para ser usada na reprodução do produto para o mercado. Devem ser isolados os códigos fontes, bibliotecas, ferramentas, configuração do software para o ambiente, procedimentos de instalação necessários para geração do software.

O transporte pode ser considerado como concluído então, após terem sido atendidas as seguintes condições :

- ✓ Relatório e documentação sobre transporte
- ✓ Integração concluída
- ✓ Portabilidade concluída com unificação do código
- ✓ Versão do SccS fechada
- ✓ Cópia Mestre pronta para liberação

Capítulo 9.

Conclusões e Trabalhos Futuros

9.1 Sumário

O processo de produção de software, segundo Humphrey [Hump89], representa o conjunto de atividades, métodos e práticas utilizados na produção e evolução do software.

Com o aumento da complexidade do processo de produção e a necessidade de ordenação das suas atividades, métodos e práticas, surgiram os modelos de processos. A utilização de modelos de processos de produção de software foi reforçada pelo reconhecimento de que o software poderia ter sua produção controlada e planejada como para outros produtos manufaturados, possibilitando a previsão de investimentos e prazos.

Este trabalho levantou a existência dos principais modelos de processos sugeridos no passado (modelos clássicos): Cascata[**Royc70**], Evolutivos[**Broo75**], Espiral[**Boeh88**] e Transformacional[**Agre86**]; e as recentes propostas de modelos realizadas pela indústria e grupos de pesquisa (novos modelos) : Cosmos[**Raym91**] e o CDPM[**Aoya93**].

Ao se examinar a utilidade destes modelos, conclui-se que : eles servem principalmente como instrumentos didáticos para explicar o processo de desenvolvimento; adotam fluxos de trabalhos genéricos [Ghez91]; e, que apresentam restrições para serem aplicados à realidade de software no mercado por :

- ◆ não cobrirem todo o ciclo de produção do software comercial (excluem por exemplo, aspectos críticos para o sucesso como empacotamento, marketing e suporte) ;
- ◆ reforçarem aspectos de desenvolvimento, desfocando visão de atividades com maior fatia de custos (tais como suporte e comercialização) ;
- ◆ dificilmente assimilarem mudanças estratégicas inesperadas no processo ;
- ◆ não condizerem com a realidade vivenciada no mercado, seja pelo excessivo formalismo, pela inadequação do modelo aos ambientes ou pela sua generalidade ; e, por
- ◆ serem, na grande maioria, modelos orientados à tarefas, refletindo a familiaridade com outras engenharias que trabalham essencialmente com produtos manufaturados.

Complacente com o atual estágio de especialização da produção de software em função da segmentação de mercado, de sua complexidade inerente e da diversidade de alternativas de execução, verifica-se que não é plausível tentar elaborar um modelo único de produção de software universal e representativo para os diversos segmentos de mercado. Pesquisas realizadas pelo Instituto de Engenharia de Software (SEI) da Universidade Carnegie Mellon confirmam que a vasta maioria das organizações norte-americanas produtoras de software não possuem ou não utilizam modelos de processos de produção de software [Kras92]. Esta realidade parece ser similar no Brasil, onde no acompanhamento de empresas emergentes vinculadas ao projeto SoftEX 2000, percebe-se a ausência de modelos de processos que balizassem o planejamento dos produtores de software.

Afim de amenizar esta carência, este trabalho caracterizou um cenário de produção de software comercial, propondo um "molde" mais adequado à realidade do processo de produção vivenciado na prática. O molde pode ser instanciado para os diversos segmentos de mercado, afim de possibilitar o desenvolvimento de modelos, manuais e guias práticos que auxiliem efetivamente no processo de produção de software dos possíveis segmentos.

Apesar de sua aplicação incipiente, justificada pela complexidade do processo de produção e necessidade de ajustes, o molde integra atividades críticas ao processo de produção de software comercial, tais como: marketing, empacotamento, testes alfa e beta, controle de qualidade e suporte pré e pós-venda. A visão geral e estruturada do molde é oferecida pelos seus três componentes: fases, atividades monofásicas e atividades polifásicas.

As fases de concepção, desenvolvimento, preparação e disponibilização são consideradas marcos no processo de produção de software comercial. Cada uma destas fases se compõe de procedimentos, denominados de *atividades monofásicas*, os quais visam executar qualitativamente os propósitos de cada fase. As atividades polifásicas de investimento, documentação, marketing e qualidade complementam o molde, à medida em que estão presentes durante todo o ciclo de produção e disponibilização de software, fornecendo a sustentação necessária para o sucesso do produto.

Uma vez levantado o molde realista para o processo de produção de software comercial e seus componentes, este foi instanciado para a produção de software Unix de prateleira, a fim de levantar procedimentos componentes de um modelo ou guia prático para este segmento de mercado, que tipicamente não são considerados relevantes na ótica acadêmica, orientada a "programação" pura.

Na instanciação do molde para a produção de software Unix de prateleira, foram selecionadas as fases de preparação e disponibilização para serem detalhadas. Esta escolha proposital se deu pela excessiva concentração dos modelos existentes na fase de desenvolvimento, desconsiderando estas fases de extrema importância para o planejamento e viabilização do produto no mercado.

9.2 Conclusões

A realização deste trabalho ofereceu a oportunidade de atentarmos para determinados aspectos do processo de produção de software até o momento desconsiderados ou sem o devido destaque no ciclo de vida do software pela maioria dos modelos existentes.

O molde proposto visa contribuir para que a engenharia de software se torne mais realista e abrangente. Mais realista na medida em que retrata a realidade de produção de software, considerando as práticas de mercado e se constituindo num instrumento efetivo para o planejamento e controle do processo de produção de software; mais abrangente, por estender a abordagem tradicional centrada apenas no desenvolvimento de produtos, destacando a importância das fases de preparação e disponibilização do software para o mercado, assim como a influência das atividades polifásicas no processo.

Quanto às atividades extra-desenvolvimento que fomentaram a extensão dos modelos tradicionais e a instanciação do molde proposto, pode-se constatar a carga que estas exercem sobre o processo de produção de software. Dentre os aspectos críticos identificados que são integrados ao processo de produção, pudemos destacar alguns pela necessidade evidente de planejamento e controle contínuo:

- O papel do marketing na concepção e evolução de produtos. Durante a Conferência Internacional de Marketing para Software (GEOCOM 93), realizada em Boston, no período de 10 a 12 de novembro, Tom Stitt, presidente de uma empresa húngara com 7 funcionários e com uma base instalada de 100.000 usuários, destacou a sensibilidade do mercado norte americano ao marketing. Segundo Stitt, apenas 40% do

sucesso do produto está vinculado a sua qualidade e potencial, ficando o restante vinculado ao sucesso de marketing.

- Aspectos técnicos de empacotamento. Pelo que percebemos existe uma distância considerável entre o software desenvolvido e aquele que realmente é disponibilizado ao mercado. Pode-se constatar a complexidade de empacotamento do software ao se relatar a elaboração de procedimentos para instalação, proteção do software contra reprodução ilegal e ajustes provenientes dos testes de aceitação (testes alfa e beta). Estas atividades monofásicas e procedimentos resultam em ajustes no código original e geração de rotinas para garantir a utilização perfeita do software no ambiente do usuário.

- Importância e ônus do suporte técnico. O trabalho destacou a importância do suporte como atividade de sustentação e evolução do produto no mercado. Realmente, a confiabilidade do suporte oferecido pelo produtor tem influenciado diretamente nas vendas de software para o mercado, cujos usuários procuram soluções confiáveis e com garantias de assistência contínua. Esta necessidade tem onerado os custos de manutenção, que além de oferecer serviços de pré-venda para atrair novos usuários (no caso de software Unix de prateleira), precisa atender eficientemente seus usuários cada vez mais exigentes.

Embora algumas destas atividades críticas (i.e, empacotamento, marketing e comercialização) possam ser executadas por terceiros, devido à falta de perfil ou vocação, estas devem ser consideradas no planejamento e controle do processo de produção de software comercial. A terceirização também necessita de acompanhamento para verificação do nível de qualidade do serviço oferecido, pois o mercado enxerga a empresa e não o terceiro, prestador dos serviços. Entretanto, a opção por terceirizar

atividades pode agregar custos ao produto e via de regra, pode reduzir as margens de lucro do produtor ou até mesmo inviabilizar a sua comercialização a um preço atraente.

Os levantamentos realizados com o produtor local para instanciação do molde para software Unix de prateleira demonstram que existe uma extrema necessidade de compreensão e ordenação do negócio. A produção de software é uma atividade multidisciplinar, que requer um relacionamento estreito com outras áreas de estudo, tais como: controle de qualidade, administração, economia, direito, marketing, vendas, comércio exterior, merchandising. Por isto e pelo nível de especialização em que atingiu a produção de software, tanto o processo de produção quanto os modelos que os representam são complexos. Entretanto, os resultados conseguidos foram animadores para este produtor, que pode visualizar melhor a complexidade do seu processo de produção com a utilização de um molde aplicado ao seu segmento de mercado.

A visualização e compreensão do ciclo completo de produção de software comercial pode contribuir efetivamente para empresas produtoras de software. Neste sentido, acredita-se que este trabalho seja uma contribuição efetiva para o programa SoftEX 2000, à medida em que baliza as empresas emergentes sobre a complexidade de se produzir software para o exigente mercado internacional.

Com relação ao molde, podemos afirmar que sua consolidação depende do interesse dos diversos segmentos de mercado em aplicá-lo, para poder ser calibrado com as diversas alternativas de produção existentes. Uma das contribuições deste trabalho foi fornecer um "esqueleto" para a construção de modelos e guias práticos específicos que orientem efetivamente os participantes do processo de produção e

disponibilização de software para o mercado. Neste sentido, o trabalho fornece uma base para a produção de guias para a produção de software Unix de prateleira ao ilustrar a adequação do molde nas suas fases de preparação e disponibilização de software.

9.3 Trabalhos Futuros

A partir da concepção de um molde realista para a produção de software voltado para o mercado, propomos que esforços sejam concentrados na construção e validação de modelos apropriados para perfis específicos de produtores de software (empresas de terceirização, de mercado livre, de desenvolvimento interno), tipos de produtos (prateleira, sob-encomenda, embutido) e segmentos de mercado (Unix, Windows, OS/2).

Os esforços devem ser realizados a partir de estreita cooperação de grupos acadêmicos interessados em Engenharia de Software e a indústria, como proposto recentemente no Workshopp Protem-CC, realizado em Itaipava, RJ [Mart93]. Acreditamos que haja interesse em tal cooperação por parte da indústria. Por exemplo, a ASSESPRO - Associação de Empresas Brasileiras de Software e Processamento de Dados (com forte ligação ao programa SoftEx 2000), deseja implantar um Plano de Qualidade em 1994 junto as suas 800 empresas filiadas. O molde genérico apresentado pode servir de base a modelos de referência no detalhamento deste plano para os vários segmentos de atuação das empresas.

Nossa recomendação de execução e distribuição de trabalhos futuros nesta linha de pesquisa é feita em função da vocação, "expertise" e experiência, como ilustra a figura 9.1. A partir de experiências relatadas pela indústria, os grupos acadêmicos envolvidos no projeto formalizam e organizam os diversos modelos (possivelmente com o engajamento de

alunos de mestrado), bem como especificam e desenvolvem ferramentas de automação de alguma(s) atividade(s). Os modelos e ferramentas construídos são validados em campo pela indústria. Uma vez validados, os modelos poderão ser adotados pela academia na atualização do ensino e/ou calibrados pela indústria para uso interno.

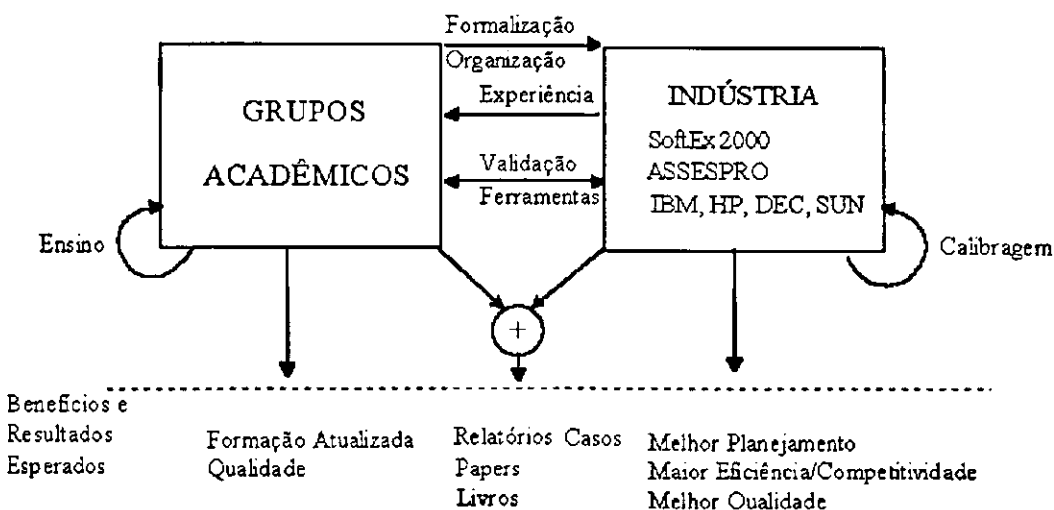


Figura 9.1 - Cooperação Academia-Indústria

Baseando-se na descrição do molde proposto, o processo de produção de software incorpora algumas atividades (monofásicas e polifásicas) ainda não devidamente exploradas pela engenharia de software e que merecem destaque pela sua importância no processo (empacotamento, suporte, testes alfa e beta, manutenção etc). Esta noção abre a oportunidade para o detalhamento de rotinas e procedimentos, bem como a construção de ferramentas que automatizem as diversas atividades previstas, sejam estas mono ou polifásicas.

Com relação ao detalhamento de atividades monofásicas, pode-se obter por exemplo, uma especificação mais detalhada de um sistema de automatização do suporte técnico integrado com as outras atividades do

molde. Talvez o suporte seja uma das mais importantes atividades no ciclo de vida do software, pela necessidade de sustentação técnica (pré e pós-venda) e evolução de produtos, e atualmente é carente de ferramental e bibliografias específicas. A especificação e implementação de um trabalho deste porte seria uma contribuição importante da academia para o setor produtivo.

A integração das atividades polifásicas ao processo de produção deve ser mais detalhada pelo valor estratégico que estas possuem. Existem trabalhos sobre custos, documentação, qualidade e marketing direcionados ao processo de produção de software, os quais poderiam ser mais interessantes caso estendessem a sua abordagem sobre todo o processo de produção e disponibilização de software. Por exemplo, um modelo de custos realista evidentemente teria que incluir aspectos de preparação e disponibilização, não ficando restrito à estimativas sobre o desenvolvimento de produtos.

Apêndice A

Arquivos Para Instalação de um Software

Os arquivos listados em seguida são utilizadas pela Infocon Tecnologia para descrever os requisitos de instalação do software InfoWord. Estes arquivos são solicitados pelo instalador genérico (Ic) para configurar o software na máquina Unix do usuário.

O arquivo 1 é um arquivo texto interpretado pelo Ic para obter basicamente a estrutura de diretórios, as permissões de acesso aos arquivos e a localização dos códigos executáveis e não-executáveis do software.

O arquivo 2 identifica as particularidades exigidas para complementar a instalação do InfoWord, através de comandos do shell do Unix padrão. A partir deste arquivo, o Ic obtém as variáveis de ambiente para o Unix, personaliza o produto através da sua chave de ativação e possibilita a instalação manual do produto.

Arquivo 1 - Especificação da Instalação do InfoWord

```
#
#   Processador de Textos InfoWord - Copyright ( C ) Infocon Tecnologia
#   Este modulo contem informacao de propriedade da
#   Infocon Tecnologia Ltda a qual deve ser tratada como
#   confidencial.
#
#
#set = "Processador de Textos InfoWord"
#prd = iw
#typ = disk
#rel = 4.0
#sel = ""
#rcmd = "tar - xF"
#rdev = xxx
#dir = /usr/lib/iw
uid   root      0
gid   root      0
iw    d755     root/root      1          ./ marks 01
iw    d755     root/root      1          ./ bin   01
iw    f444     root/root      1          ./marks/abicomp.mks 01
iw    f444     root/root      1          ./ marks/ibm_us.mks 01
iw    f444     root/root      1          ./ marks/iso8859.mks 01
iw    x511     root/root      1          ./ bin/iw   01
iw    x511     root/root      1          ./ bin/configiw 01
iw    x511     root/root      1          ./ im.ovl   01
iw    f644     root/root      1          ./ iw.msg   01
iw    f644     root/root      1          ./ iw.hlp   01
iw    f644     root/root      1          ./ iw.men   01
iw    f444     root/root      1          ./ release.doc 01
iw    f700     root/root      1          / tmp/init.iw 01
iw    x500     root/root      1          ./ bin/person 01
#!iw  1792     Processador de Textos InfoWord
```

Arquivo 2 - Particularidades da Instalação do InfoWord

```
#      Processador de Textos InfoWord - Copyright ( C ) Infocon Tecnologia
#      Este modulo contem informacao de proppriedade da
#      Infocon Tecnologia Ltda a qual deve ser tratada como
#      confidencial.
#
#      Script para inicialização da instalação

# Definição de códigos de retorno
: $ {OK= 0} $ {FAIL = 1} $ {STOP = 10} $ {HALT = 11}

manual = nao
[ "$1" = "-m" ] && manual = sim

# Acerta o conteudo de /etc/profile
PROFILE = /etc/profile
IWPATH = `pwd`
[ -f $PROFILE ] ||> $PROFILE

grep "IWPATH" $PROFILE >/dev/null 2>&1 || {
    echo "IWPATH = $IWPATH" >> /etc/profile
    echo "export IWPATH" >> /etc/profile}

# Personaliza o produto
PERSON = $IWPATH/bin/person
DIRCMD = $IWPATH/bin
OBJPERSON = "$DIRCMD/configiw $DIRCMD/iw $IWPATH/im.ovl"
echo "
*****
      Personalizacao do INFOWORD
***** "

RET=$FAIL
[-x $PERSON] && $PERSON $OBJPERSON && RET = $OK
[ $RET = $FAIL ] && {
    echo " *****
      ATENCAO:
      *****

      O InfoWord nao foi corretamente personalizado.
      Voce terah que reiniciar a instalacao!

      Pressione <Enter> para continuar .... \c"
    read key
    exit $RET}

echo " O InfoWord foi instalado no diretório $IWPATH.
      Para poder utilizar o Infoword, adicione a variavel de ambiente PATH do seu sistema o
      percurso $ IWPATH/bin. "

# Instalacao manual
[ "$manual" = "sim" ] && {
    rm -rf /tmp/_lbl /tmp/perms}
exit $OK
```

Apêndice B

Formulário Infocon de Licenciamento de Software

INFOCON

FICHA DE REGISTRO DE SOFTWARE

Pedimos o favor de preencher esta ficha e devolvê-la à **Central de TeleAtendimento - CTA** da INFOCON. Isto ativará, em seu benefício, nossos Serviços de Garantia e Assistência ao Usuário e mantê-lo-á informado sobre novos lançamentos e serviços.

EMPRESA

ENDEREÇO

BAIRRO

CIDADE

U.F.

CEP

FONE # 1

RAMAL FONE # 2

RAMAL

FAX

CONTATO # 1

CONTATO # 2

EQUIPAMENTO

S.O.

VERSÃO (S.O.)

FABRICANTE

MÍDIAS

<input type="checkbox"/> Disquete 5 1/4"	<input type="checkbox"/> Disquete 3 1/2"	<input type="checkbox"/> Fita Streamer	<input type="checkbox"/> Fita DAT
<input type="checkbox"/> HD <input type="checkbox"/> DD	<input type="checkbox"/> HD <input type="checkbox"/> DD	<input type="checkbox"/> Densidade	<input type="checkbox"/> Densidade

DE QUEM ADQUIRIU O PRODUTO ?

DATA DE COMPRA

Nº DA NOTA FISCAL

MARQUE COM UM "X" SE DESEJA RECEBER INFORMAÇÕES SOBRE:

<input type="checkbox"/> Outros Produtos e Serviços	<input type="checkbox"/> Cursos Ministrados pela INFOCON
---	--

IDENTIFICAÇÃO DO SOFTWARE

Apêndice C

Formulário Lotus de Licenciamento de Software

LOTUS 1-2-3 para UNIX

FORMULÁRIO PARA OBTENÇÃO DE CHAVES DE ATIVAÇÃO

Para que você instale e opere corretamente o 1-2-3 para UNIX em seu equipamento, você deve obter "Chaves de Ativação" para o produto em conformidade com a configuração necessária (ex.: multiusuário, licença de rede, etc). Este é um procedimento estabelecido pela Lotus Development Corp. e deveria ser seguido por você, em contato direto com o suporte técnico da Lotus Dublin, Irlanda.

A INFOCON, atendendo solicitação da Lotus e para facilitar-lhe o uso do 1-2-3 para Unix, se propõe a contatar o suporte da Lotus em Dublin a seu favor para obter as chaves necessárias. Desde já, pedimos sua compreensão para atrasos adicionais que nossa intermediação possa causar.

A solicitação de chaves de ativação ao suporte técnico da Lotus Dublin requer o preenchimento do formulário anexo. Uma vez preenchido, você deve encaminhá-lo para a Central de TeleAtendimento (CTA) da INFOCON através do número de fax abaixo:

*CTA - INFOCON
Av. Brig. Faria Lima, 1390 - Conj. 32
Jardim Paulistano
01452 - 001 - São Paulo - SP.
Fone: (011) 814 - 8677
Fax : (011) 816 - 2957*

Em caso de dúvidas no preenchimento, favor consultar o capítulo 4 do *Installation and Configuration Guide*

Nome, cargo:

Empresa:

Endereço Completo:

Fone: Fax:

Versão do Produto: Máquina:

Identificação da Máquina Servidora com a licença:
(OBS: Esta identificação, "host id", é o resultado do comando "lmid" localizado no diretório "bin" do 1-2-3)

Nome da Máquina Servidora com a licença:
(OBS.: Este nome, "host name", é o resultado do comando "uname -n").

Número de Série da "Media Edition":
(OBS.: Verifique o envelope "Lotus 1-2-3 Getting Started" incluído no produto).

Número de Série de cada "Node Edition" a instalar:
(OBS.: Verifique o envelope "Lotus 1-2-3 Getting Started" incluído no produto).

Data: ____/____/____

Apêndice D

Exemplo de Portabilidade de Código

Para examinar a portabilidade de código para diversos ambientes computacionais, pode-se tomar como exemplo a rotina de alocação de memória desenvolvida pela Infocon Tecnologia para os seus produtos. A seguir são exibidos parcialmente a rotina de alocação de memória genérica (rotina 1) e o arquivo de seleção de ambiente (rotina 2).

Ao se examinar a rotina de alocação de memória, pode-se constatar que esta utiliza apenas procedimentos e variáveis genéricas (do tipo TamSeg, MemaLinha) que isolam o código das particularidades dos diferentes fornecedores de Hardware e versões do Unix. Na verdade, esta rotina implementa um algoritmo de alocação de memória que atende a maioria das máquinas Unix no mercado, sem explicitar os detalhes de programação de cada uma destas máquinas.

A portabilidade deste código é conseguida através da inclusão da rotina 2 em tempo de compilação. Como pode ser verificado, a rotina 2 define os detalhes de implementação de cada fornecedor. De acordo com o ambiente em que será instalado o software (HP Unix, AIX, SCOUnix, etc.), o compilador seleciona através do comando `ifdef`, a parte do código da rotina 2 que será incluído na rotina de alocação de memória.

Rotina 1 - Rotina Genérica de Alocação de Memória

```
# include "rotina2.h"
/*****
**
** Nome : ani_inicmemoria (apenas para UNIX)
** Objetivo : Pre-alocar espaço de memória para tornar a análise de código intermediário de um
**            programa mais eficiente (evitando overhead de alocação)
**
** Valor de retorno : nenhum
** Algoritmo : - Calcula valor final de espaço a alocar (desejado + overhead)
**            - Trava código intermediário
**            - Aloca primeiro segmento e tabelas necessárias
*****/
ani_inicmemoria (nome, pgm, lnbytes)
char   *nome;          /* nome do programa */
FILE   *pgm;           /* arquivo aberto correspondente a "nome"*/
register long lnbytes; /* número de bytes a pre-alocar */
{
int     nbytes; /* número de bytes a local no primeiro segmento */
lnbytes = MEMALINHA (lnbytes);

#ifdef SHMEMORY

    _trav_fd = fileno (pgm);
    if ( trava ( _trava_fd ) < 0 ) {
        fatal (M_TRAVINTER, nome);
    }

#endif /* SHMEMORY */

    mem_falta = lnbytes + /* código intermediário */
                sizeof ( GLOBESTAT ) + /* estrutura global estática */
                TAMCOMUNIC; /* buffer de comunicação entre proc. */

#ifdef SHMEMORY

    mem_falta += ITMAXSEG * sizeof( SEGMENTO); /* tabela de segmentos */

#endif /* SHMEMORY */

    nbytes = (mem_falta <= TAMSEG) ? mem_falta : TAMSEG;
    mem_falta -= nbytes;

    ap_atual = get_segmento ( &nbytes, nome );
    ap_final = ap_atual + nbytes;

    _buf_comunic = ani_alocmemoria (1, TAMCOMUNI);
    globestat = ( GLOBESTAT * ) ani_alocmemoria (1, sizeof (GLOBESTAT));
}
```

Rotina 2 - Seleção de ambiente

```
# ifdef XENIX386

# define TAMSEG          1000000
# define ITMAXSEG       10
# define MIN_SHMENDER   (( char *) 0x64000000)
# define MAX_SHMENDER   (( char *) 0xFFFFFFFF)
# define SHW_W          0200
# define SHM_R          0400

# endif /* XENIX 386 */

# ifdef HPUX

# define TAMSEG          1000000
# define ITMAXSEG       10
# define MIN_SHMENDER   (( char *) 0x0)
# define MAX_SHMENDER   (( char *) 0x0)

# endif /* HPUX */

# ifdef SCOUNIX

# define TAMSEG          1000000
# define ITMAXSEG       10
# define MIN_SHMENDER   (( char *) 0x64000000)
# define MAX_SHMENDER   (( char *) 0xFFFFFFFF)
# define SHW_W          0200
# define SHM_R          0400

# endif /* SCOUNIX */

# ifdef AIX

# define TAMSEG          1000000
# define ITMAXSEG       10
# define MIN_SHMENDER   (( char *) 0x0)
# define MAX_SHMENDER   (( char *) 0x0)

# endif /* AIX 386 */
```

Bibliografia

- [Agre86] Agresti, W.W. *New Paradigms For Software Development*. IEEE Computer Society Press, 1986.
- [Agre91] Agresti, William. "Low-Tech Tips for High-Quality Software". in IEEE Software, pp. 86-89, November. 1991.
- [Aoya93] Aoyama, M. "Concurrent Development Process Model". in IEEE Software, pp. 46-55, July. 1993.
- [Basi91] Basili, V.R. & Musa, John. "The Future Engineering of Software : A management Perspective " in IEEE Computer, Los Alamitos, pp. 90-96, September. 1991.
- [Blac77] Black, R.K.D et alli. "BCS Software Production Data". Boeing Computer Services, Inc., Final Tech. Rep., RADC-TR-77-116, NTIS AD-A039852, March. 1977
- [Boeh81] Boehm, Barry, *Software Engineering Economics*, USA, Englewood Cliffs, Prentice Hall, 1981.
- [Boeh84] Boehm, Barry, " *Software Engineering Economics* ". in IEEE Trans. Software Eng. , vol. SE-10, pp. 4-21, January. 1984.
- [Boeh88] Boehm, Barry, " *Understanding and Controlling Software Cost*". in IEEE Trans. Software Eng. , pp. 1462 - 1477, October. 1988 .

- [Broo75] Brooks, F. *The Mythical Man-Month*, USA, Addison-Wesley, 1975.
- [Chmu90] Chmura, L.J. et alli. "Evaluating Software Design Processes by analyzing Change Data Over Time". in IEEE Trans. on Soft. Engineering, Vol. 16, No.7, pp. 729-740, July. 1990.
- [Conn91] Connie, Smith. "Improving Service While Controlling Costs ". in IEEE Software, pp. 95-96, November. 1991.
- [Cros79] Crosby, Philip. *Quality is Free*, USA, McGraw Hill, 1979.
- [Date84] Date, C.J. *Introdução a Sistemas de Bancos de Dados*, Rio de Janeiro, Campus, 1984.
- [Dema89] DeMarco, Tom. *Controle de Projetos de Software*, Rio de Janeiro, Campus, 1989.
- [Evan87] Evans, Tom. *The Software Support Handbook*, California, Business Knowledge, 1987.
- [Ells88] Ellsworth, Leon. "Information Systems (IS) Cost of Quality - Revisited". in Quality Data Processing, pp. 36-37, October. 1988.
- [Falc90] Falconi, Vicente. *Gerência da Qualidade Total*. Belo Horizonte, Fundação Christiano Ottoni, 1990.
- [Ghez91] Ghezzi, C et alli. *Fundamentals of Software Engineering*, New Jersey, Prentice Hall, 1991.
- [Genu91] Genuchten, M.V. "Why is Software Late ? An Empirical Study of Reasons for Delay in Software Development ". in IEEE Trans. on Soft. Engineering, Vol 17, No. 6, pp. 582-590, June. 1991.
- [Grum91] Gruman, Galen. "How to Assure Quality : Debate Shows Divisions". in IEEE Software, pp. 99-103, March. 1991.

- [Hage89] Hager, James A., " *Software Cost Reduction Methods in Practice* ". in IEEE Trans Software Eng., vol. 15, no. 12, December. 1989.
- [Hump88] Humphrey, W.S. " *Characterizing The Software Process : A Maturity Framework* " in IEEE Software, pp.73-79, March. 1988.
- [Hump89] Humphrey, W.S. *Managing the Software Process*, USA, Addison-Wesley, 1989.
- [Iso9000] ISO 9000 (E). " *Quality Management and Quality Assurance Standards - Guidelines for Selection and Use* ". ISO, Genève, Switzerland, 1991.
- [Iso9003] ISO 9000-3 (E). " *Quality Management and Quality Assurance Standards-Part 3 : Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software* ". ISO, Genève, Switzerland, 1991.
- [Iso9126] ISO/IEC 9126 : 1991 (E). " *Information Technology-Software Product Evaluation-Quality Characteristics and Guidelines for their Use* ". ISO/IEC, Genève, Switzerland, 1991.
- [Jens83] Jensem, R.W. " *An Improved Macrolevel Software Development Resource Estimation Model* " in Proc. 5th ISPA Conf., pp. 88-92, April. 1983.
- [Jone89] Jones, Caper. " *Measuring Software Defect Removal* " . in Quality Data Processing, pp. 24-28, January. 1989.
- [Jura79] Juran, J.M. *Quality Control Handbook*, USA, McGraw Hill, 1979.
- [Kras92] Krasner, H, et alli. " *Lessons Learned from a Software Process Modeling System* " in Communications of the ACM, pp. 91-100, September. 1992.

- [Lin90] Lin, Chi Y. et alli. " *Computer-Aided Software Development Process Design* " . in IEEE Transactions on Software Engineering, Vol. 15, No. 9, pp. 1025-1037, September. 1990.
- [Land90] Landsbaum, Jerome. " *Measuring the Cost of Imperfect Quality* " in Quality Data Processing, pp. 43-45, January. 1990.
- [Long90] Longstreet, David. *Software Maintenance and Computers*, California, IEEE Computer Society Press Tutorial ,1990.
- [Mart93] Martins, L. M. e Moura, A.. " *A Modelagem do Processo de Produção, Disponibilização e Evolução de Software* ". Anais do Workshop Protem-CC, Itaipava, RJ, Dezembro. 1993.
- [Madr89] Madron, Beverley. " *Cost of Quality : Measurement for Management* " . in Quality Data Processing, pp. 14-16, October. 1989.
- [Mist91] Mistrik, I. *Modern Software Engineering : Foundations and Current Perspectives*. Notas de Aula, Departamento de Sistemas e Computação, UFPB, 1991.
- [Muri88] Murine, Gerald E. " *Integrating Software Quality Metrics with Software QA* ". in Quality Progress, pp. 38-43, November. 1988.
- [Nise84] Nisen, William G. et alli. *Marketing your Software*, New York, Addison-Wesley, 1984.
- [Pari90] Parikh, Girish. *Reengenharia de software - técnicas de manutenção de programas e sistemas*, Rio de Janeiro, Livros Técnicos e Científicos Editora Ltda, 1990.

- [Parn87] Parnas,D.L. and Weiss,D.M. "*Active design reviews: principles and practices*". in *Journal of Systems and Software*, 7(4): pp. 259-65, December. 1987
- [Pere91] Pereira, Saulo. *Estudo de Validação de Métricas Aplicadas as Linguagens C e PASCAL*. Dissertação de Mestrado , Universidade Federal da Paraíba, Junho. 1991.
- [Poor88] Poore, J.H. "*Derivation of Local Software Quality Metrics*" . in *Software- Practice and Experience*,Vol. 18(11), pp. 1017-1027, November. 1988.
- [Pres87] Pressman, R.H. *Software Engineering : A Practioner's Approach*, USA, McGraw-Hill, 1987.
- [Putn91] Putnam, L.H. "*Trends in Measurement, Estimation and Control*". in *IEEE Software*, pp. 105-107, March. 1991
- [Raym91] Yeh, Raymond T. "*A Commonsense Management Model*". in *IEEE Software*, November 1991,pp. 23-33.
- [Reme87] Remer,D et alli. *Legal Care for Your Software - A Step-by-Step Guide for Computer Software Writers and Publishers*. USA, Nolo Press, 1987.
- [Remu83] Remus, Horst. "*Organizational Means to Increase Software Quality*", California, IBM, TR 03.232, 1983.
- [Roch92] Rocha, A.R. *Controle de Qualidade de Software*. Transparências de Curso, Departamento de Sistemas e Computação, UFPB, Maio. 1992.
- [Royc70] Royce,W.W. " *Managing the Development of Large Software Systems : Concepts and Techniques* " , *Proceedings Wescon*, August. 1970.
- [Sbes91] Anais do V Simpósio Brasileiro de Engenharia de Software, Ouro Preto, 1991.

- [Stew88] Stewart, Nick. "Software Error Costs" . in *Quality Progress*, 48-49, Nov. 1988.
- [Wals77] Walston, C. E. and Felix, C.P., " A Method of Programming Measurement and Estimation". in *IBM Systems Journal*, vol. 16, no. 1, pp. 54 - 73, 1977.