Universidade Estadual de Campinas
Instituto de Matemática, Estatística e Computação Científica
Mestrado em Qualidade

Elementos Intrínsecos do Software e sua Influência na Qualidade do Processo de Desenvolvimento

Izilda Gomes Garcez Capovilla

Orientador: Prof. Dr. Mario Jino

Campinas – SP Brasil
1999

Elementos Intrínsecos do Software e sua Influência na Qualidade do Processo de Desenvolvimento

Este exemplar corresponde à redação final do trabalho final de Mestrado Profissional devidamente corrigido e defendido por IZILDA GOMES GARCEZ CAPOVILLA e aprovado pela banca examinadora.

Campinas, 26 de novembro de 1999.

Mario Jino

Prof. Dr. Mario Jino
Orientador

Banca Examinadora:

1) Prof. Dr Mario Jino
2) Prof. Dr. Ademir J. Petenate
3) Profa. Dra. Ana Cervigni Guerra

Trabalho final de Mestrado Profissional apresentado ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do título de Mestre em Qualidade.

Dissertação de Mestrado defendida em 26 de novembro de 1999 e aprovada pela Banca Examinadora composta pelos Profs. Drs.

Prof (a). Dr (a). MARIO JINO

Prof (a). Dr (a). ADEMIR JOSÉ PETENATE

Prof (a). Dr (a). ANA CERVIGNI GUERRA

Ao meu marido, por seu apoio e compreensão.

Agradecimentos

Ao Professor Mario Jino pela orientação precisa, pela amizade e pelo exemplo de profissionalismo.

Ao Professor Ademir José Petenate pelas sugestões e pela confiança, me aceitando como aluna regular do Mestrado em Qualidade.

À Professora Ana Cervigni Guerra pelas inúmeras contribuições feitas ao trabalho e, principalmente, pelo incentivo em todos os momentos.

Aos amigos da Fundação CTI pela troca de idéias que fomentaram o início deste trabalho.

Aos amigos do Mestrado em Qualidade pela convivência enriquecedora e pelas excelentes oportunidades de crescimento pessoal.

Aos meus queridos Celso, Letícia e Thaís pela enorme paciência.

Resumo

A indústria mundial de software vem experimentando um crescimento que nenhum outro setor produtivo jamais viveu. A influência estratégica dos sistemas de software em áreas tão diversas como medicina, telefonia, transportes, comunicação, suporte à produção e à gestão empresarial, entre outros, reafirma a dependência das organizações em geral sobre a qualidade do software.

Apesar de todos os esforços por parte tanto da comunidade de pesquisa quanto dos desenvolvedores, ainda enfrentamos freqüentemente problemas básicos como elevada imprecisão nas estimativas de prazos e custos, baixa produtividade e qualidade inadequada do produto final.

Considerando esses aspectos, este trabalho visa apresentar elementos classificados como intrinsecamente naturais do software e analisar os desvios manifestados nos processos de desenvolvimento com relação a esses elementos. Também são discutidas algumas das principais iniciativas na área de qualidade de software – modelo CMM, projeto SPICE e normas ISO, e são apresentadas considerações sobre aplicabilidade.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

É fato notório que a qualidade, nos últimos anos, deixou de ser encarada unicamente como um fator de vantagem competitiva, passando a ser o ponto central do sucesso, e até mesmo da sobrevivência, de muitos setores produtivos. A busca pela qualidade é hoje tema dominante em todos os campos tecnológicos, e em diversas áreas o controle da qualidade é uma realidade estabelecida há décadas.

A indústria mundial de software vem experimentando um crescimento que nenhum outro setor produtivo jamais viveu: na década de 70 foram faturados US$ 1 bilhão, em 80 US$ 10 bilhões e em 90 estima-se US$ 100 bilhões. Dados como esses projetam a indústria de software como um dos melhores negócios do futuro.

É inegável a influência da qualidade de software na indústria em geral, haja vista não só a importância estratégica dos sistemas de apoio a gestão industrial, como também a participação cada vez maior de componentes de software nos produtos oferecidos pelas empresas. Cresce a cada dia a parcela de software relacionada direta ou indiretamente a produtos tão variados como aviões, telefonia e comunicações, equipamentos para controle e automação da produção, equipamentos médicos, alimentos e serviços. Parece um contraponto que, apesar de toda essa importância, a produção de software ainda transcorra em níveis baixos de controle, chegando, freqüentemente, a resultar em elementos de qualidade precária.

Segundo a última pesquisa realizada pelo Ministério de Ciência e Tecnologia [MCT97], as empresas nacionais de software caracterizam-se por serem empresas de pequeno porte – 68% na faixa de micro e pequenas empresas, com comercialização bruta proveniente de software de até R$ 720 mil. O sucesso na implantação de sistemas de melhoria de qualidade é crucial na sobrevivência dessas organizações, e o oposto muitas vezes pode determinar o seu fim.

Num cenário onde se juntam empresas com poucos recursos, com processos de produção mal controlados e um mercado cada vez maior e mais exigente, proliferam métodos para melhoria de qualidade, freqüentemente vistos como "tábuas de salvação" pelas empresas. Porém, o retorno após meses de investimento é muitas vezes frustrante. Não raramente, encontramos organizações marcadas por um rastro de grandes fracassos.

1.1 Panorama Tecnológico

Um programa de garantia de qualidade de software abrange tanto *o controle do processo de desenvolvimento* quanto a *avaliação do produto final de software.*

Duas séries de normas se destacam nesse contexto, ambas publicadas pela ISO (International Organization for Standardization): série ISO 9000 - aspectos relacionados ao processo, e série ISO/IEC 9126 - aspectos relacionados à avaliação de qualidade do produto de software [ABNT96].

A natureza peculiar do processo de desenvolvimento de software motivou a ISO em 1991 a aprovar uma norma internacional da série ISO 9000, denominada ISO 9000-3, que estabelece diretrizes para facilitar a aplicação da Norma ISO 9001 em organizações envolvidas com o desenvolvimento, fornecimento e manutenção de software.

Em relação aos modelos relacionados ao processo de desenvolvimento de software, um dos mais conhecidos é o Capability Maturity Model (CMM) - ou Modelo de Maturidade SEI, como é conhecido no Brasil, desenvolvido pelo Software Engineering Institute (SEI) da Universidade Carnegie Mellon, a partir das propostas de Philip Crosby [SEI99]. Nesse modelo, baseado na definição de 5 níveis de maturidade que correspondem a estágios de evolução dos processos da organização, são identificadas possíveis áreas para melhoria no processo de desenvolvimento de software adotado pelas empresas.

A ISO/IEC aprovou um programa de trabalho para o desenvolvimento de uma Norma Internacional para avaliação de processo de software, estabelecendo em 1993 o projeto SPICE com o objetivo de apoiar o trabalho de normatização,

através da elaboração de uma proposta de norma, desenvolvimento de testes de campo que possibilitem a coleta de dados de utilização dos padrões e disponibilização ao mercado das informações sobre o andamento dos trabalhos de padronização [SPICE99]. Atualmente o projeto SPICE encontra-se no estágio de análise dos dados levantados em campo.

A Secretaria de Política de Informática e Automação do Ministério da Ciência e Tecnologia publicou no ano de 1997 os resultados da pesquisa *Qualidade no Setor de Software Brasileiro* [MCT97], realizada entre empresas associadas à ABEP – Associação das Empresas Estaduais de Processamento de Dados, ABES – Associação Brasileira das Empresas de Software, ABINEE – Associação Brasileira da Indústria Elétrica e Eletrônica, ASSESPRO – Associação das Empresas Brasileiras de Software e Serviços de Informática e SUCESU – Sociedade dos Usuários de Informática e Telecomunicações.

A pesquisa , cujos dados estão apresentados no Anexo A, objetivou diagnosticar a qualidade do setor, fornecendo subsídios para o Subprograma Setorial da Qualidade e Produtividade em Software - SSQP/SW, do Programa Brasileiro de Qualidade e Produtividade (PBQP). A estratégia adotada pelo SSQP/SW visa, além de compor um diagnóstico do setor em relação à qualidade e produtividade, analisar tendências nacionais e internacionais e propor ações para a solução dos problemas que influenciam a obtenção de níveis mais elevados de qualidade.

É importante salientar que a disponibilização de serviços tecnológicos de apoio à melhoria de qualidade em software, como infra-estrutura de normatização, certificação, ensaios e informação tecnológica, é indispensável no avanço da qualidade. Atualmente, apenas alguns raros serviços relativos a essa infra-estrutura são oferecidos às empresas, e o que existe é muito pouco conhecido e utilizado.

1.2 Objetivo do Trabalho de Pesquisa e Metodologia Adotada

Existem várias maneiras de se organizar e definir o assunto Qualidade. Juran [JURA-91a] apresenta-os na forma de uma trilogia de processos inter-

relacionados: planejamento, controle e aperfeiçoamento. Ainda segundo Juran, na maioria das empresas existe uma situação geral na qual:

1. numerosos processos operacionais são deficientes;
2. a abordagem do planejamento da qualidade também é deficiente, e essa deficiência é a causa do surgimento de muitos dos processos operacionais deficientes.

A essência de qualquer processo de planejamento é, em primeiro lugar, identificar ***o que*** deve ser feito, e em seguida ***como*** deve ser feito. O processo de identificar o que deve ser feito implica, basicamente, em:

a) determinar as necessidades a serem atendidas;
b) identificar as características do produto que atendam às necessidades levantadas.

Sem sombra de dúvida, uma das maiores dificuldades no desenvolvimento de sistemas de software é *garantir que o sistema faça o que se espera que ele faça.* Apesar de todos os esforços de pesquisa e da disponibilização de ferramentas e métodos de produção, ainda hoje enfrentamos freqüentemente problemas básicos como elevada imprecisão nas estimativas de prazos e custos, baixa produtividade e qualidade inadequada do produto final. Por que, afinal, não dispomos na indústria de software de métodos capazes de suportar o processo produtivo com o nível de qualidade necessário e planejado?

Considerando esses aspectos, este trabalho visa apresentar elementos classificados como intrinsecamente naturais do software e analisar os desvios manifestados nos processos de desenvolvimento com relação a esses elementos. Também são discutidas algumas das principais iniciativas na área de qualidade de software – modelo CMM, projeto SPICE e normas ISO, e são apresentadas considerações sobre aplicabilidade.

O material apresentado está organizado como segue:

a) Capítulo 1: introdução, objetivo do trabalho de pesquisa, metodologia adotada e uma breve apresentação do panorama tecnológico enfocando qualidade de software;

b) Capítulo 2: visão geral do planejamento e melhoria da qualidade, apresentação de algumas definições gerais relacionadas ao tema e aspectos do planejamento da qualidade nas pequenas empresas;
c) Capítulo 3: caracterização do produto de software, apresentando fatores de natureza intrínseca do software e discutindo como esses fatores influenciam o processo de produção;
d) Capítulo 4: discussão de aspectos do processo de produção de software, considerações específicas sobre qualidade em software e apresentação de fatores de risco no processo de desenvolvimento;
e) Capítulo 5: análise da evolução da Engenharia de Software e da maturidade da tecnologia disponível para produção de software e discussão de aspectos comportamentais de clientes, gerentes e pessoal técnico;
f) Capítulo 6: apresentação das principais iniciativas na área de qualidade em software (normas ISO 9000, modelo CMM, projeto SPICE e normas ISO 9126) e considerações sobre aplicabilidade;
g) Capítulo 7: conclusões finais e temas para pesquisa futura;
h) Anexo A: principais dados da *pesquisa Qualidade no setor de Software Brasileiro* usados no trabalho;
i) Anexo B: glossário de termos utilizados.

Observação: Os conceitos e significados relacionados às palavras marcadas com o símbolo * em sua primeira ocorrência no texto estão apresentados no glossário do Anexo B.

Capítulo 2

Planejamento e Melhoria da Qualidade - Uma Visão Geral

O significado do que é *Qualidade* é controverso, pois sua interpretação envolve muitas opiniões diferentes. Cada um dos grandes autores da área apresenta seus próprios conceitos e orientações. Seguir uma ou outra orientação pode significar conduzir as coisas de forma diferente, porém o ponto de chegada, apesar das divergências no *"como fazer"*, é sempre o mesmo: *estabelecer um processo de melhoria contínua por toda a organização.* Só assim as organizações serão capazes de atingir e sustentar uma maior vantagem competitiva e permanecer no mercado. Deming apresenta a relação entre qualidade e participação no mercado como uma reação em cadeia iniciada a partir das ações de melhoria nos processos e sistemas da organização. Processos melhores levam a um aumento da produtividade, à diminuição dos desperdícios e à melhoria dos produtos. Os clientes passam a obter produtos melhores, o que impulsiona o crescimento da participação no mercado e a obtenção do retorno do investimento [DEM90].

A seguir estão discutidas algumas definições relacionadas à qualidade, conforme apresentadas por Juran [JURA-91a], que auxiliam no entendimento do significado de *Qualidade*.

2.1 Definições

A palavra ***Qualidade*** tem múltiplas interpretações, porém a idéia geral de *qualidade* relaciona-se a dois significados principais:

- as características do produto que vão ao encontro das necessidades dos clientes proporcionando a satisfação com o produto ;
- a ausência de falhas.

Produto é o resultado de qualquer processo. É composto por bens, software e serviços. *Bens* são componentes caracterizados pela tangibilidade. *Software* tem

mais de um significado neste caso: procedimentos lógicos para executar tarefas (incluindo-se aqui os programas de computador) ou informações organizadas em geral, tais como: relatórios, planos instruções e roteiros. *Serviços* são trabalhos executados para atender a terceiros.

Características do Produto são propriedades que os produtos possuem e que visam atender a certas necessidades do cliente.

Cliente é toda pessoa que sofre o impacto do produto. Podem ser externos ou internos. Clientes externos são aqueles que sofrem o impacto do produto mas não fazem parte da organização que produz o produto. Incluem-se nesta classe os compradores do produto, órgãos de governo, o público, por exemplo. Clientes internos são os receptores de produtos gerados por processos internos à organização (por exemplo, documentos de um departamento para outro).

Necessidades do Cliente relaciona-se às necessidades que devem ser atendidas por determinadas características do produto, e aplica-se tanto a clientes externos quanto internos. Para clientes externos, o atendimento das necessidades determina o grau de satisfação com o produto, e portanto a facilidade de venda. No caso de clientes internos, o atendimento das necessidades determina, entre outros, o grau de produtividade, qualidade e moral dos empregados.

Satisfação com o Produto é o fator decisivo para a comercialização do produto. A variação no atendimento das necessidades dos clientes leva a diferentes graus de satisfação com o produto e conseqüentemente às diferenças na participação de mercado.

Deficiência ou Falha do Produto são desvios que causam transtornos ao cliente. Podem aparecer de diversas formas, entre elas: atrasos na entrega, falhas em serviços, erros e retrabalho. Cada desvio é o resultado de alguma falha em um processo ou produto.

Garantia da Qualidade é todo o conjunto de ações sistemáticas ou planejadas necessárias para conferir um nível de confiança adequado aos produtos e

serviços, para que venham a atender às necessidades especificadas em termos de qualidade.

2.2 A Trilogia da Qualidade

Sob um ponto de vista metodológico, a administração da qualidade pode ser abordada como um conjunto de processos inter-relacionados de planejamento da qualidade, controle da qualidade e aperfeiçoamento da qualidade.

Planejamento da Qualidade é a atividade de desenvolvimento de produtos que atendam às necessidades do cliente, e envolve uma série de etapas:

- Determinação de quem são os clientes;
- Determinação das necessidades dos clientes;
- Estabelecimento das características do produto que atendam às necessidades dos clientes;
- Desenvolvimento de processos capazes de produzir as características do produto estabelecidas;
- Transferência dos resultados do planejamento aos grupos operativos.

Controle da Qualidade corresponde ao processo usado pelos grupos operativos para atender os objetivos do processo e do produto, e baseia-se no *ciclo do produto* que consiste nas etapas:

- Avaliação do desempenho operacional real;
- Comparação do desempenho real com os objetivos;
- Atuação com base na diferença.

Aperfeiçoamento da Qualidade é o processo que objetiva atingir níveis de desempenho sem precedentes, através da análise dos dados de acompanhamento e atitudes de correção e melhoria dos outros dois processos que compõem a administração da qualidade (planejamento e controle).

As atividades necessárias para se atingir qualidade também podem ser classificadas em termos de três disciplinas básicas:

- Negócios ou empreendimentos, relacionados aos riscos associados aos negócios em geral. Incluem-se aqui a análise da situação econômica, escolha de mercado, investimentos e financiamentos;
- Administração, através da qual é obtida a mobilização para atingir os objetivos;
- Tecnologia, relacionada à utilização das forças e materiais da natureza em benefício da sociedade. Incluem-se aqui os projetos, especificações, instrumentos, entre outros.

Juran afirma que "*muitos dos problemas com a qualidade estão relacionados ao fato de administradores de alto nível viverem no mundo dos negócios e da administração, ao passo que os especialistas em qualidade vivem no mundo da tecnologia*". De certa forma, essa atitude pode ser um reflexo histórico da evolução no trato com a qualidade. Até meados da década de 40, o controle da qualidade era realizado pelos operadores, capatazes e inspetores, indivíduos cujas tarefas concentravam-se, praticamente 100%, no uso das ferramentas e métodos de produção, ou seja, no domínio da tecnologia disponível. Somente a partir da década seguinte o controle estatístico começou a se evidenciar como um meio eficaz para o conhecimento e controle dos processos produtivos; a partir daí evoluíram os processos de garantia da qualidade e gestão pela qualidade conhecidos atualmente.

2.3 Planejando a Qualidade

Num sentido mais amplo, o planejamento é o processo para o estabelecimento dos objetivos da qualidade e para o desenvolvimento dos meios para realizar esses objetivos. Os objetivos podem ser planejados visando a ruptura, ou seja, um aperfeiçoamento de desempenho sem precedentes na organização, ou para a manutenção do controle nos níveis atuais. Tais objetivos são utilizados em toda a hierarquia da organização, porém existem diferenças em sua visualização de um nível da hierarquia para outro. No nível operário, por exemplo, os objetivos tendem a tomar a forma de características de conformidade ou parâmetros específicos de processos. Já nos níveis acima a visão é ampliada. Essas diferenças na interação com os objetivos pedem arranjos adequados nos planos da qualidade de cada setor.

No topo da organização os objetivos são amplos e pedem planos elaborados sobre bases amplas, que serão, por sua vez, detalhados e transcritos em objetivos parciais, cada um com planos específicos. Independentemente do nível de hierarquia, o planejamento da qualidade objetiva atender as necessidades de qualidade dos clientes, e para atender essas necessidades deve-se estabelecer uma série de atividades, ilustradas na Figura 2.1, que apresenta um roteiro resumido para o planejamento geral da qualidade. Esse roteiro é universal e pode ser utilizado por qualquer tipo de empresa em todos os níveis hierárquicos.

2.4 O Planejamento da Qualidade e as Pequenas Empresas

As pequenas empresas são mais informais e menos sistematizadas do que as grandes organizações, o que resulta numa abordagem diferenciada em relação ao planejamento da qualidade. Uma situação semelhante também pode ocorrer em grandes companhias que possuam setores relativamente autônomos; esses setores tendem a adotar as mesmas práticas encontradas nas pequenas empresas.

O comportamento dessas empresas é moldado em função de certas características inerentes: nicho de mercado limitado, ações gerenciais impactando poucos clientes, poucos níveis hierárquicos e poucos departamentos, o que leva a uma maior proximidade entre os gerentes e as ações operacionais. O planejamento é executado, em larga escala, pelo pessoal operacional.

A informalidade das pequenas empresas tem a vantagem de evitar os custos relacionados à preparação e manutenção de procedimentos escritos, pois a maioria das informações encontram-se na memória do pessoal. Porém, esse ambiente informal pode trazer dificuldades às empresas no caso de perda desse pessoal experiente. A falta de uma atividade específica de planejamento da qualidade também pode expor as empresas ao risco de tornarem-se isoladas por não manterem atualizados seus métodos de planejamento.

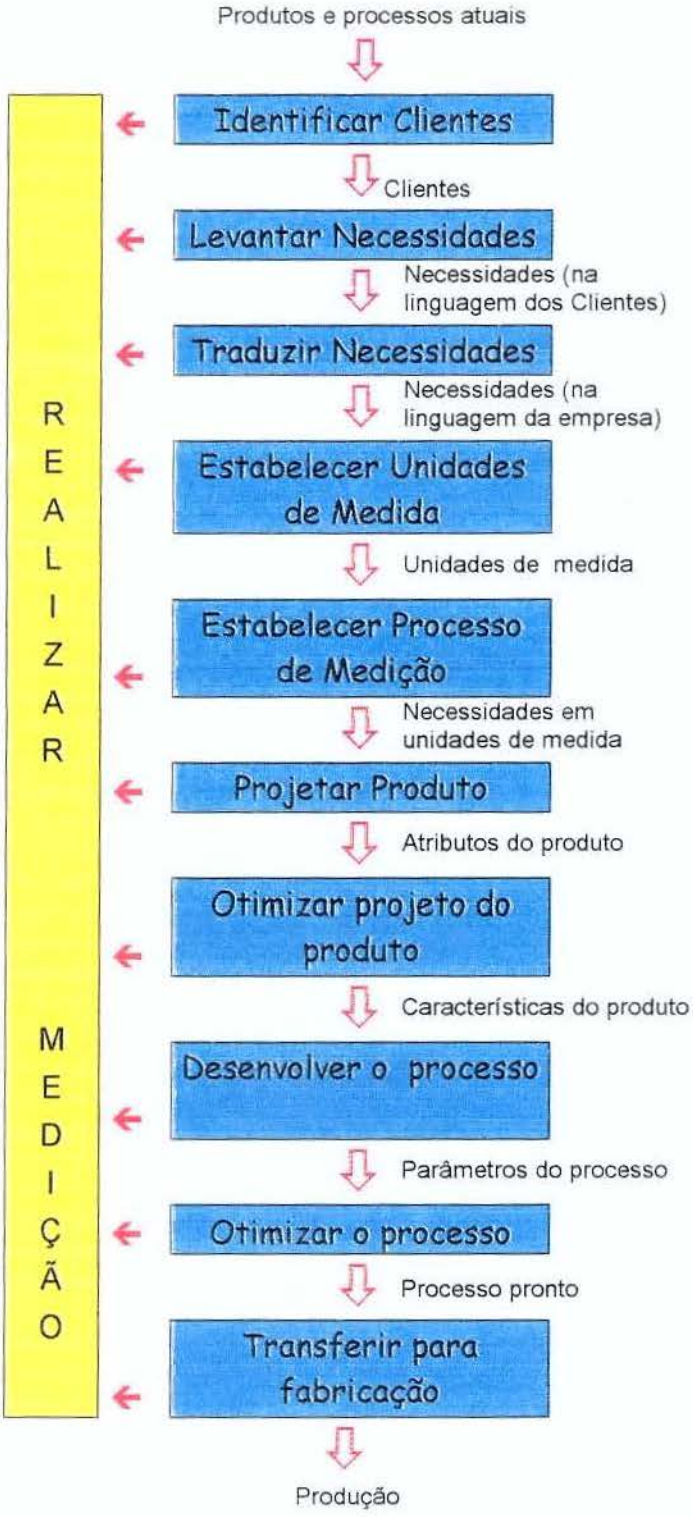


Figura 2.1 Roteiro para o Planejamento da Qualidade

(Fonte: Juran, J.M. *Controle da Qualidade - Conceitos Políticas e Filosofia da Qualidade*, pg 180)

Capítulo 3

Caracterização do Produto de Software

Definir "produto de software" remete-nos à idéia de descrever características pertencentes a uma dentre duas possíveis categorias:

- características particulares, dependentes do objetivo da implementação da solução a ser desenvolvida e dos recursos utilizados;
- características gerais, independentes da implementação ou recursos.

As características discutidas a seguir pertencem à segunda categoria. São aquelas inerentes à essência do software, ou seja, estão presentes em todos os sistemas, independentemente do tipo de produto, do modelo de projeto ou das técnicas de desenvolvimento adotadas [PRE95] [BRO87] [JURA-91b]. A discussão dessas características permite-nos identificar fatores específicos do software como produto e verificar como esses fatores influenciam e modificam o modelo geral de produção.

3.1 Complexidade

Para software, a relação "tamanho X complexidade" está entre as maiores encontradas em produtos da construção humana. Isso deve-se, principalmente, a fatores como o número e a complexidade de condições especiais tratadas, o grau de encadeamento entre os módulos componentes do sistema, tipos de estrutura de dados ou linguagens utilizadas, entre outros. Esses atributos são determinados em parte pelo problema a ser solucionado pelo software, e em parte pelas decisões de projeto tomadas [JURA-91b].

O número de elementos diferentes num *sistema de software** é bastante elevado, pois nesses sistemas não existem duas partes iguais, ao menos nos níveis anteriores à *codificação**. Se elas existirem, normalmente serão transformadas em uma única *subrotina**, o que torna novamente verdadeira a afirmação anterior. Cada uma dessas partes interage com diversas outras dentro e fora do sistema, o que torna o software, sob esse aspecto, profundamente diferente de produtos

tradicionais, como computadores, edifícios ou automóveis, que possuem inúmeras partes repetidas com níveis baixos de interdependência.

Notamos que, nesse quadro, o aumento da complexidade não é uma função linear do tamanho do sistema , e sim proporcionalmente muito maior. Dados estatísticos apontam, por exemplo, que a complexidade cresce na razão do quadrado do número de linhas de um programa. Isso se deve ao fato de o aumento no tamanho dos sistemas não corresponder somente à inclusão de mais módulos repetidos ou em tamanhos maiores, mas a inclusão de *novos* módulos, o que aumenta o número de elementos diferentes e o volume de interações entre eles.

Da complexidade decorrem muitos dos problemas relacionados à comunicação entre elementos da equipe e, consequentemente, o desvio nos custos, atrasos de cronograma e falhas no produto. Dificuldades na visualização e entendimento dos diversos estados possíveis de um sistema podem levar a falhas na segurança, já que estados "invisíveis" podem estar presentes sem terem sido projetados. É por esse motivo que muitas vezes sistemas de software fazem o que não era suposto que fizessem, causando operações imprevistas e danos inesperados. São inúmeros os casos conhecidos atribuídos a falhas de software [COLL94]. À medida que a complexidade aumenta, o problema de garantir a *confiabilidade** torna-se mais difícil [JURA-91b].

3.2 Conformidade e Modificabilidade

Uma parte significativa do projeto de sistemas de software relaciona-se ao tratamento das diversas interfaces que o sistema deve manter com as demais entidades do meio no qual será utilizado: equipamentos, demais sistemas de software já existentes, regras institucionais, comportamentos pessoais dos usuários e aspectos culturais da organização.

Em muitos casos, o software é o elemento escolhido para sofrer as adaptações por ser o último componente a ser incorporado ao processo, ou seja, freqüentemente será integrado em um conjunto já em utilização, com o qual os usuários e as organizações já estão familiarizados. Atualmente, na maioria dos casos o software expressa a *funcionalidade** do produto no qual está inserido

(quando não é, ele mesmo, o produto) por compor também a interface entre o produto e as demais entidades envolvidas, o que causa, freqüentemente, a necessidade de sua adaptação até mesmo a hábitos profissionais.

Em outros casos, o software é o escolhido para a adaptação por ser percebido como o componente mais maleável e adaptável do sistema. Na maioria dos casos isso é verdade, principalmente em relação aos custos da adaptação, pois os recursos necessários à adaptação do software, basicamente mão-de-obra intelectual, são geralmente inferiores aos custos de adaptação dos demais elementos (por exemplo, reconfiguração de placas de hardware).

Outra característica interessante surge quando analisamos o volume verificado de modificações realizadas ao longo da vida útil dos sistemas de software funcionalmente bem sucedidos. Contrariamente ao que se poderia esperar de um produto que atendeu adequadamente as necessidades especificadas, os registros históricos mostram que o volume de modificações é muito elevado, devendo-se exatamente aos fatores principais já citados:

- pleno atendimento das necessidades do usuário;
- longa vida útil do sistema.

O primeiro fator é relacionado a percepção da utilidade do sistema: quando o produto de software agrada ao *usuário**, este tende a estender sua utilização para situações no limite, e até mesmo além, do domínio especificado para a aplicação. A pressão para as modificações ocorre no sentido de estender o conjunto de funções originais, de forma a incorporar os novos usos inventados pelos usuários.

O segundo fator refere-se aos sistemas que, por atenderem adequadamente às necessidades, sobrevivem além do tempo normal de vida da tecnologia para a qual foram projetados, ou seja, o meio ambiente computacional e os *sistemas de suporte** . Os usuários relutam em substituir um bom sistema, do qual geralmente dependem várias atividades, temendo perder o nível de atendimento. Porém, com o passar do tempo, são lançados novos periféricos (impressoras, unidades de disco, etc), os sistemas de suporte passam a oferecer novas facilidades ("upgrades") e os sistemas de software devem então ser modificados para atender a evolução tecnológica.

Em resumo, produtos de software estão inseridos numa "matriz cultural" de aplicações, usuários, normas, hábitos profissionais e pessoais, tecnologia, etc, em constante evolução, forçando o elemento de software a mudar continuamente.

3.3 Intangibilidade

Apesar da simplificação na estrutura dos sistemas, conseguidas principalmente devido à adoção de algumas metodologias, o software ainda continua sendo inerentemente invisível e intangível. O que é visível são as conseqüências da execução do software sobre o equipamento ligado a ele, como a exibição de uma tela, um sinal sonoro ou a gravação de um dado no disco. A percepção do sistema é variável e subjetiva, dependendo da forma como o sistema é executado, do equipamento utilizado e da percepção do usuário.

Os *modelos lógicos** utilizados para visualização do sistema não capturam completamente a funcionalidade do produto. Freqüentemente são necessários vários esquemas lógicos, cada um representando um aspecto específico de projeto como o *fluxo de controle**, o *fluxo de dados**, relações de dependências, *temporização**. A visualização segmentada, distribuída em vários esquemas diferentes, dificulta a visão do sistema como um todo e, consequentemente, a execução de diversas etapas envolvidas no processo de desenvolvimento.

A intangibilidade dificulta a comunicação entre os elementos da equipe de projeto, mas, pior do que isso, prejudica a comunicação entre o cliente e o desenvolvedor, podendo provocar distorções nas fases de levantamento e validação de requisitos.

3.4 Não Desgaste

Se observarmos o índice de falhas verificadas ao longo do *ciclo de vida** de diversos produtos da indústria, veremos que ele apresenta uma variação semelhante ao perfil de curva apresentado na Figura 3.1. Um exemplo típico é o hardware computacional. Para alguns desses produtos o índice de falhas é relativamente alto no início (atribuídas a defeitos de projeto e fabricação), mas

após os defeitos serem corrigidos o número de falhas cai, permanecendo estável durante certo período. Com o passar do tempo, porém, o produto começa a sentir os efeitos acumulativos do uso (como poeira, exposição a condições ambientais desfavoráveis, etc), passando a apresentar sinais de *desgaste.* Para esses produtos, o *tempo médio entre falhas* (MTBF – Mean Time Between Failures) aplica-se diretamente.

Em software os componentes lógicos são duráveis. A falha[1] de software resulta de erros de projeto ou implementação, e os defeitos[2] permanecem no sistema até serem especificamente percebidos devido à ocorrência de um erro[3] quando uma determinada entrada acontece. Portanto, o MTBF, por definição, *não se aplica diretamente* [JURA-91b].

O software não é sensível a problemas ambientais e nem sofre nenhum tipo de defeito devido a efeito acumulativo. Os defeitos de projeto e "fabricação" também provocam um grande número de falhas logo no início, mas o que ocorre a seguir é bastante diferente. A Figura 3.2 mostra o perfil de curva modificado para software.

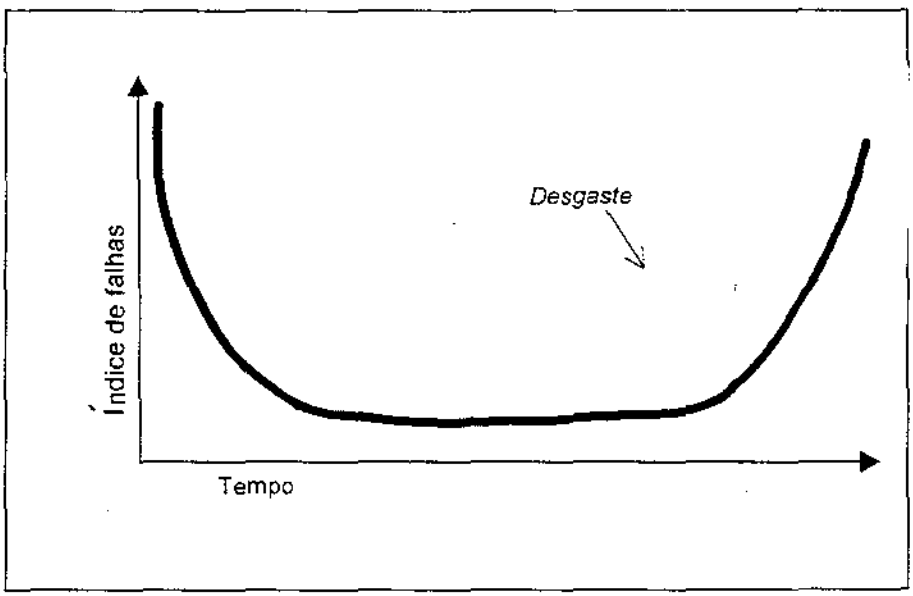


Figura 3.1 Curva de falhas padrão

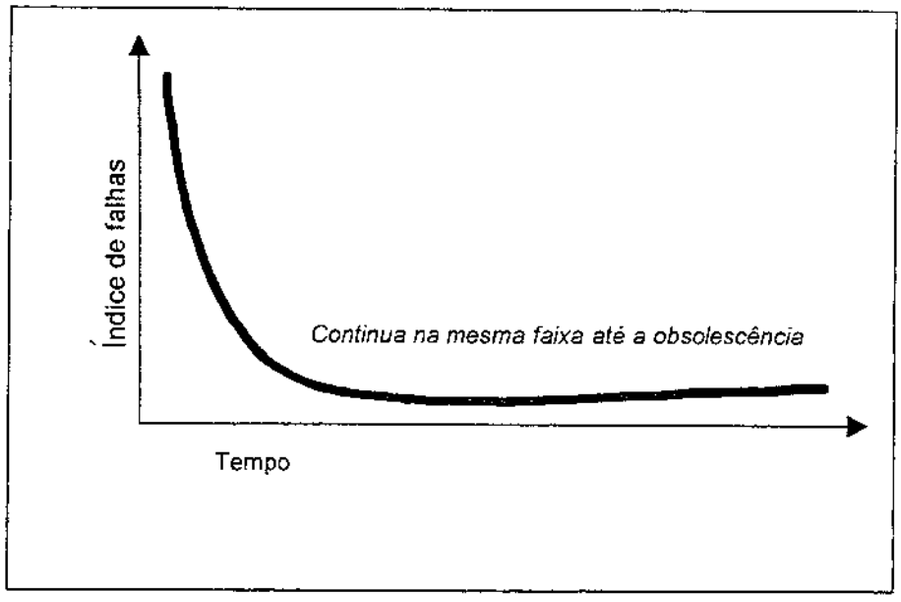


Figura 3.2 Curva idealizada de falhas de software

Muitos sistemas de software passam por uma série de modificações ao longo de sua vida útil. Elas fazem parte da manutenção do sistema, conforme visto na Seção anterior "Conformidade e Modificabilidade". Quando essas alterações são realizadas, observa-se um novo período de falhas, pois novos defeitos são introduzidos no sistema, fazendo com que a curva do índice de falhas apresente picos. O efeito observado sobre o índice indica que aos poucos o nível mínimo de falhas começa a elevar-se. O software não se recupera totalmente após a correção. O que verificamos, então, é um processo semelhante à *deterioração* [PRE95]. Esta *deterioração* é proporcional ao volume de modificações efetuadas, e é bastante diferente do *desgaste* (Figura 3.3).

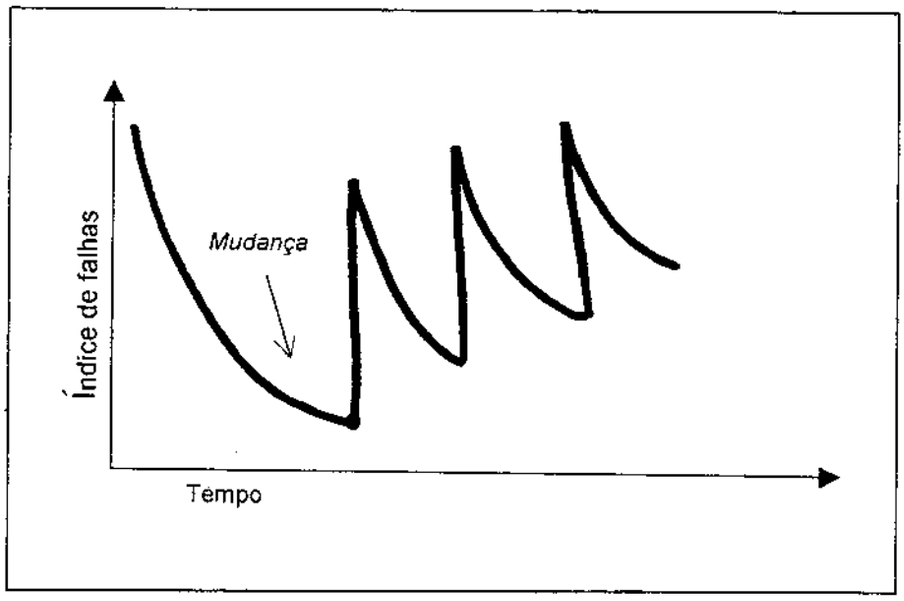


Figura 3.3 Curva real de falhas de software

A variação ascendente do índice de falhas acontece devido à forma como os defeito são corrigidos. Na maioria dos produtos, quando um componente apresenta um defeito ele é substituído por outro, normalmente uma "peça de reposição", com função idêntica à anterior. No software não existem peças de reposição. Toda falha indica um erro cometido nas fases anteriores, e a correção de um defeito remete a alterações de projeto.

Segundo Juran, esta diferença em durabilidade deve ser claramente compreendida, principalmente em relação aos conceitos de verificação e validação de software. Em [JURA-91b] encontramos a seguinte afirmação:

> *"O não reconhecimento deste fato pode levar a procedimentos de garantia de qualidade improdutivos e, portanto, demasiadamente dispendiosos. Pior: pode fazer com que as questões verdadeiramente importantes não sejam tratadas."*

3.5 Produção sob Medida

A maioria dos sistemas de software é produzida sob encomenda e não montada a partir de outros componentes de software já existentes. Cada necessidade implica um novo produto, e cada novo produto um novo projeto pois, com poucas

exceções, os projetistas não contam com um estoque de componentes prontos para utilização.

Apesar das técnicas de *reuso** enfocarem esse assunto existe ainda uma série de dificuldades a serem vencidas para a disponibilização de código reusável [GAR95]. Uma das mais importantes é o fato de que componentes de software reusáveis devem ser construídos e documentados para fins de reuso, isto é, o processo produtivo padrão adotado para desenvolvimento de sistemas não é o mesmo para construção de módulos reusáveis, o que desencoraja as empresas a efetuar o conseqüente investimento adicional sobre a produção.

3.6 Síntese

Foram apresentadas seis características intrínsecas de produtos de software: complexidade, conformidade, modificabilidade, intangibilidade, produção sob medida e não desgaste. As cinco primeiras características invocam o caráter lógico, abstrato e complexo do software, e determinam grande parte das dificuldades enfrentadas no seu desenvolvimento. O *não desgaste* diferencia o software de quase a totalidade dos produtos modernos. Apenas uma pequena parcela de produtos, como música e cinema por exemplo, aproximam-se do software sob esse aspecto.

Entender claramente essas características e focar esforços no sentido de planejar e controlar seus efeitos sobre a produção pode significar uma grande contribuição para o sucesso de projetos.

Capítulo 4

Aspectos do Processo de Produção de Software

Todo processo de desenvolvimento de software pode ser descrito como uma seqüência de três fases genéricas: *definição, desenvolvimento e manutenção.* Essas três fases são identificadas em qualquer projeto, independentemente da área de aplicação, tamanho ou complexidade do sistema ou *modelo de ciclo de vida utilizado** [DAV88].

A fase de definição focaliza *o que* deve ser desenvolvido, especificando as exigências fundamentais do sistema. As atividades de análise de requisitos, análise de sistema e planejamento do projeto são algumas dentre as previstas para essa fase.

A fase de desenvolvimento focaliza *como* o software será construído, detalhando a estrutura do sistema e todos os componentes (dados, linguagens, procedimentos, por exemplo). As atividades principais dessa fase são o projeto do sistema, implementação ou codificação e testes.

A última fase, manutenção, corresponde à etapa seguinte à entrega do sistema e concentra-se nas mudanças que ocorrerão devido à correção de erros, descobertos após a entrega do produto, às adaptações exigidas pela evolução do ambiente no qual o software está inserido e às solicitações de alterações ou ampliações exigidas pelo cliente. Pertencem a essa fase as atividades de correção, adaptação e melhoramento funcional.

A produção do sistema de software corresponde, portanto, às fases de definição e desenvolvimento descritas acima. A distribuição ideal do esforço para a produção está ilustrada na Figura 4.1. Os números apresentados referem-se a sistemas desenvolvidos dentro de esquemas de controle e planejamento bem elaborados, e enfatizam a importância das atividades de análise e projeto, realizadas no início da produção, que detém de 40% a 50% do esforço total. A atividade de testes é a

segunda mais importante, responsável por 30% a 40% do esforço [1]. A fase de codificação / implementação, que corresponde àquela na qual se obtém efetivamente como produto os programas de computador (código fonte), correspondem a somente 20% do esforço total [PRE95].

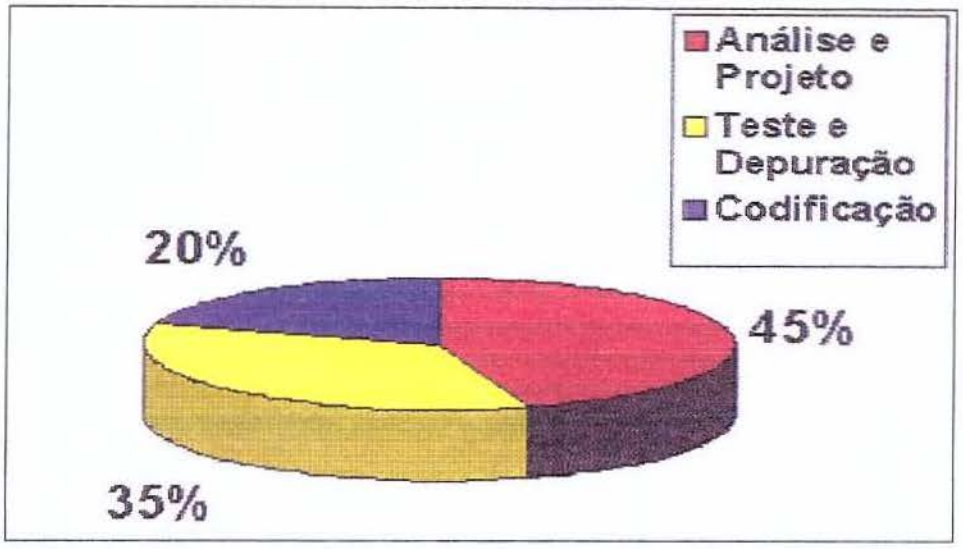


Figura 4.1 Distribuição do esforço na produção de sistemas de software

A etapa de testes é a última, dentro do processo de garantia de qualidade, na qual a qualidade pode ser avaliada e melhorada.

Cox [COX90] apresenta o software como uma forma híbrida, algo num domínio entre o concreto e o abstrato, entre o tangível e o intangível (Figura 4.2). Esses dois domínios coexistem na definição de software, influenciam-na em proporções diferentes e são altamente dependentes tanto do entendimento subjetivo sobre o que é o trabalho de desenvolver software quanto da área de conhecimento e atuação das pessoas envolvidas.

Em alguns grupos, o desenvolvimento de software é freqüentemente visto como uma atividade mental, abstrata, solitária e criativa. Cox ilustra essa visão citando uma afirmação feita por um dos programadores da Apple Computer, na época um

[1] Diversas outras atividades visando a inspeção e descoberta de defeitos podem e devem ser inseridas em fases anteriores ao teste, porém a maior parte do esforço ainda recairá na fase final de testes.

dos melhores, que compara a programação à criação de um romance: *"Reusar o código (código fonte dos programas) de outras pessoas pode provar que não nos importamos com o que produzimos. Eu não reusaria código, assim como Ernest Hemingway não reusaria parágrafos de outros autores."*
Pessoas com essa visão de "*software intangível*" tendem a favorecer a concentração de força e a valorização do produtor, que é a parte do processo que detém todos os "*dons*" e "*habilidades*" necessários para desenvolver o produto.

Uma visão, em parte, oposta a essa é a que enfoca a tangibilidade do software. Sob esse ponto de vista, a característica de intangibilidade é também contemplada, mas trabalhada de forma diferente da visão anterior. Os esforços ocorrem no sentido de transformar o software, fazendo-o tão acessível quanto possível aos não especialistas, ou seja, aqueles que não possuem o "*dom da arte*". Neste caso, o principal favorecido é o cliente. Foram nesses círculos de pensamento que surgiram as técnicas de manipulação de interfaces com o usuário, as técnicas de reuso e os populares "browsers" (navegadores da Internet), algumas entre tantas facilidades que tornaram os programas mais tangíveis, menos abstratos e mais próximos tanto dos usuários quanto da capacidade profissional da média dos programadores.

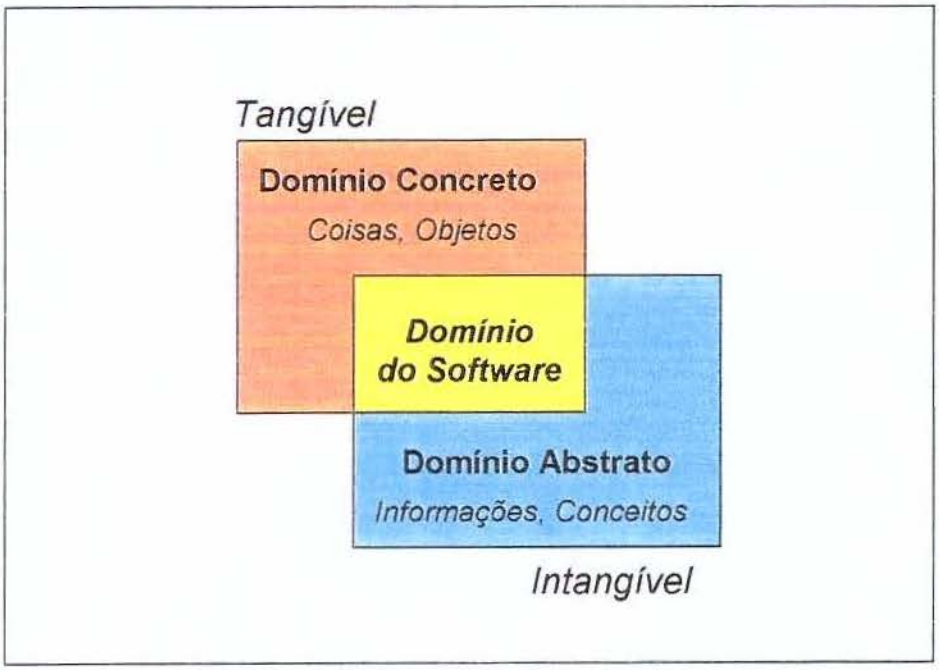


Figura 4.2 Domínio do Software

Wesselius e Ververs [WES90] complementam a discussão do tangível e intangível identificando 3 componentes da qualidade para o software (Figura 4.3):
a) *Componentes objetivamente avaliáveis*,
b) *Componentes subjetivamente avaliáveis*, e
c) *Componentes não avaliáveis ou imprevisíveis*.

Os *componentes objetivamente avaliáveis* são aqueles indicados na especificação do produto, principalmente durante a fase de especificação dos requisitos iniciais, e são baseados em duas fontes:

- o uso *conhecido* para o qual o produto foi especificado;
- preferências pessoais do cliente, que puderam ser fornecidas objetivamente.

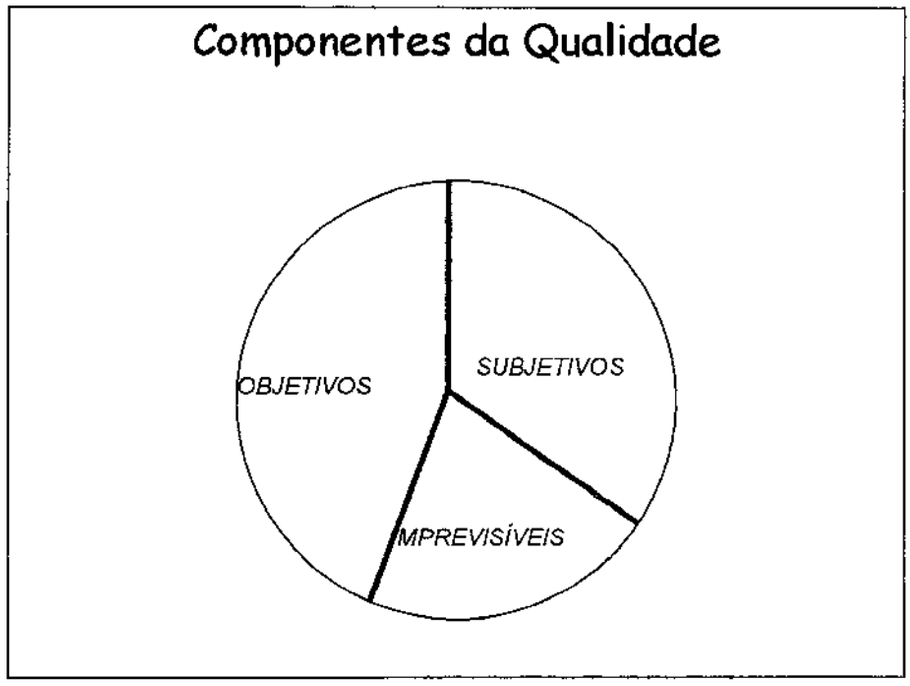


Figura 4.3 Componentes da Qualidade em Software

Os *componentes subjetivamente avaliáveis* são aqueles relacionados à variação no julgamento do que é qualidade para cada cliente, individualmente. Referem-se ao grau de correção e completitude dos requisitos objetivamente especificados, tentando responder à questão de *quanto*, realmente, o produto cobre as

expectativas do cliente (ou seja, o grau de adequação às necessidades especificadas).

Os *componentes não avaliáveis ou imprevisíveis* são aqueles relacionados às características indeterminadas do produto. Um exemplo típico é o nível de adaptação do sistema às plataformas de hardware e software ainda não disponíveis. Como vimos no Capítulo 2, este é um fato freqüentemente verificado, pois é comum o produto ter uma vida útil maior que a do ambiente computacional para o qual ele foi originalmente projetado. Esse conjunto de componentes referem-se à capacidade do produto de adaptar-se a necessidades emergentes, ainda desconhecidas.

Apenas a primeira categoria de componentes pode oferecer uma boa base para um projeto objetivo e passível de verificação. As duas outras formam um conjunto para o qual a avaliação de conformidade é praticamente impossível. Brooks apresenta muito claramente o cenário da especificação do projeto de software:

> *"A parte mais difícil da construção de sistemas de software é decidir o que precisamente desenvolver. Nenhuma outra parte do trabalho envolve tanta dificuldade quanto estabelecer o detalhamento técnico dos requisitos, incluindo todas as interfaces com as pessoas, com as máquinas e com os demais sistemas de software. Nenhuma outra parte ameaça tão intensamente o resultado do projeto se for feita incorretamente, e nenhuma outra é tão difícil de reparar depois."*

Weinberg [WEIN93] afirma que a conformidade com os requisitos não é suficiente para garantir a qualidade, referindo-se ao trabalho de Crosby [CRO79]. Obviamente, esse conceito é aplicável *somente se os requisitos estiverem corretos.* Como no caso de software, na grande maioria dos casos, isso não acontece, essa abordagem aplicar-se-ia somente à categoria de componentes objetivamente avaliáveis.

Notamos que, para haver conformidade com os requisitos, os componentes objetivamente avaliáveis devem ser preponderantes. Quanto maior a porcentagem de componentes dessa categoria na especificação do produto, mais objetiva será a nossa avaliação e acompanhamento e maiores serão as chances de alcançar graus mais elevados de qualidade.

Muito do que necessitamos conhecer a respeito de gerência de qualidade em software concentra-se na determinação de métodos adequados para o levantamento de requisitos. Esse é, sem dúvida, um dos pontos centrais da eficiência dos produtos.

4.1 Algumas Considerações sobre Qualidade em Software

Além dos problemas relacionados aos requisitos do produto, a gerência da qualidade também enfrenta dificuldades em estabelecer a relação entre as características internas do processo e do produto e os níveis de qualidade atingidos. Algumas pesquisas indicam que características e propriedades internas do produto, inseridas durante a produção, determinam *atributos externos de qualidade**; porém, é necessário obter-se um modelo que identifique claramente a relação entre propriedades internas e atributos externos, e qual o efeito dessas propriedades sobre a qualidade. Isso significaria que a qualidade final do produto poderia ser avaliada e, portanto, prevista durante a fase de produção, a partir de medições das propriedades internas. Essa possibilidade agrada tanto desenvolvedores quanto clientes, pois ambos interessam-se em conhecer tão cedo quanto possível a qualidade do produto final. Porém, até o momento não foram identificadas métricas baseadas em atributos internos capazes de fazer uma boa previsão da qualidade final [KIT96].

No contexto da qualidade a medição é um aspecto de extrema importância. A comunidade desenvolvedora de software aos poucos vai reconhecendo que o sucesso na definição, implementação e evolução tanto de seus produtos quanto de seus processos está diretamente relacionado à implantação de programas de medição adequados na organização. As medições devem ser estabelecidas em consonância com as metas e características da organização e de seus projetos [IESE99]. Um dos fatores que tem impulsionado o crescimento na utilização de medições em software é a grande aceitação do modelo CMM (ver Capítulo 6). O modelo vem sendo adotado por um grande número de organizações, espalhadas por todo o mundo e, segundo ele, para alcançar níveis mais elevados de capacidade as empresas devem incorporar métricas em seus processos de

desenvolvimento de software. Existem métricas específicas para produtos e processos de software, usadas no contexto descrito a seguir.

A qualidade do software pode ser vista como a composição de dois ramos, cada um com características particulares (Figura 4.4). *Qualidade de processo* refere-se aos modelos e técnicas de produção adotados no desenvolvimento do produto e está diretamente relacionada a características como produtividade, custo e controle de possíveis propriedades internas determinantes da qualidade final. Algumas dessas propriedades são relacionadas ao grau de *portabilidade**, confiabilidade e *eficiência** do produto, por exemplo.

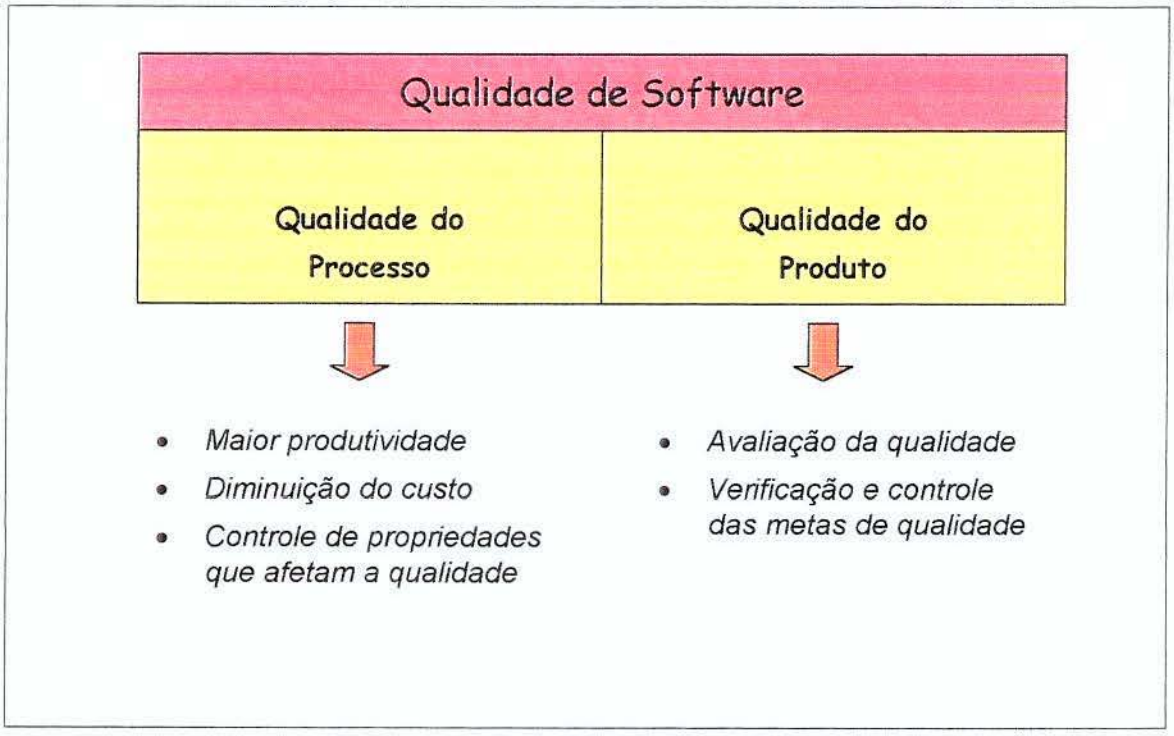


Figura 4.4 Composição da Qualidade em Software

Qualidade de produto relaciona-se aos atributos de qualidade externos e internos e seu controle contribui para a adequação e melhoria do processo produtivo, além de possibilitar a verificação da qualidade externa (ou qualidade em uso) do produto final.

Portanto, a qualidade de software é alcançada atuando-se em conjunto sobre a qualidade do processo e a qualidade do produto. Cada uma dessas áreas possui objetivos e métodos próprios. Apesar de serem de grande importância, esses métodos não serão inseridos neste trabalho por não enfocarem o objetivo principal da pesquisa.

4.2 Desenvolvimento por Projeto de Engenharia

O desenvolvimento de software é um projeto de engenharia e não uma operação de manufatura. Embora existam algumas semelhanças com atividades de um processo de manufatura, o software não é manufaturado, e sim *contém operações de manufatura* em uma pequena parcela do processo. Poderíamos identificar como operações de manufatura incluídas no processo de desenvolvimento a duplicação do sistema (ou seja, a criação de cópias do produto) e a impressão de manuais de documentação. Weinberg [WEIN93] classifica as atividades de produção em:

- *projeto e criação*, relativo as fases de levantamento de requisitos, projeto básico ou *projeto de alto nível** e preparação da documentação;

- *pseudo-manufatura*, que corresponderia a operações com algumas propriedades de projeto e criação e algumas propriedades de manufatura, relativas às fases de projeto, codificação e *testes**;

- *manufatura*, relativa às fases de impressão de manuais e duplicação do produto (preparação dos discos, CD-ROM, ou outro meio de distribuição do produto).

Todas essas atividades dependem de pessoas, mas a relação entre a pessoa e seu trabalho é profundamente diferente entre atividades das classes "manufatura" e "projeto / criação" [PER94].

Outra diferença importante está na concentração de custos, que no software está nas fases de projeto, codificação e testes, ou seja, nas fases de engenharia e não de produção como na manufatura [PRE95].

4.3 Fatores de Risco no Processo de Desenvolvimento

Como vimos nos Capítulos anteriores, diversos fatores influenciam a produção de software. A seguir faremos uma revisão desses fatores, apresentando a relação entre as características dos produtos de software e o processo produtivo.

Tabela 4.1 Relação entre características dos produtos de software e o processo produtivo

Características	**Impacto no processo de produção**
Complexidade	• inexistência de partes(componentes) repetidas; • grande volume de interações entre partes do produto e entre partes do produto e outros elementos no mesmo ambiente; • dificuldades na visualização e entendimento do produto tanto para desenvolvedores quanto para clientes
Conformidade e Modificabilidade	• interfaces variadas e numerosas entre o sistema e diversos elementos do ambiente; • grande volume de modificações adaptativas;

Intangibilidade	• dificuldades no entendimento do produto; • subjetividade na percepção do produto e no julgamento da qualidade devida ao alto grau de abstração; • dificuldades nas medições;
Não desgaste	• vida útil longa; • observada a "deterioração" devida à manutenção e modificações • MTBF não aplicável;
Produção sob medida	• projetos específicos e altamente dependentes da aplicação; • quase inexistência de componentes reusáveis prontos para utilização; • baixo índice de repetição de procedimentos de produção (baixa repetibilidade);
Imaturidade dos métodos de suporte à produção	• dificuldades na comunicação das boas experiências conhecidas; • dificuldades na repetição, análise, controle e adaptação de práticas anteriores de projeto.

Apesar de estar relacionada juntamente com as demais características identificadas como intrínsecas ao software, a imaturidade dos métodos de suporte é uma característica com tendência transitória. Os problemas relacionados à falta de métodos adequados de suporte à produção deverão, aos poucos, desaparecer, na medida em que o hiato entre a ciência e a prática for sendo superado.

A análise dos diferentes impactos apresentados na Tabela anterior levam ao levantamento de possíveis riscos, indicados a seguir. Além dos riscos potenciais,

foram incluídas algumas observações, visando o esclarecimento da relação entre o risco indicado e fatores relacionados anteriormente na Tabela.

- Imprecisão no levantamento de requisitos, provocando desvios no produto final que poderá apresentar-se incompleto ou com funções inadequadas.
- Dificuldades na modelagem da solução, o que poderia provocar alocação inadequada de recursos. A modelagem da solução é em parte determinada pela solução a ser implementada pelo sistema e em parte pelas decisões de projeto tomadas para alcançar a solução final.
- Possibilidade de existência de estados ou funções desconhecidas, causadores de operação não projetada e efeitos colaterais imprevistos na execução.
- Dificuldades em diversos níveis de comunicação, gerando desvios no acompanhamento gerencial
- Alta sensibilidade às modificações do ambiente no qual o sistema se insere, devido ao número elevado de interfaces mantidas, o que implica na geração constante de custos relacionados a manutenção adaptativa enquanto o sistema existir.
- Queda do nível da qualidade oferecida (qualidade final do produto), devido ao efeito "degeneração" provocado pelo volume de modificações acumulados em longa vida útil dos sistemas.
- Dificuldades no aproveitamento de práticas de sucesso, devido a falta de organização e disponibilização de técnicas e experiências já conhecidas.

Dentre as várias análises realizadas em projetos de software fracassados, muitas indicam que os problemas poderiam ter sido evitados, ou no mínimo bastante reduzidos, se os elementos de alto risco do projeto tivessem sido identificados e resolvidos no início [BOE91] [MAC95].

4.4 Síntese

A produção de software reside num espaço entre os domínios concreto e abstrato, entre o tangível das coisas e objetos e o intangível das informações e conceitos. Esses domínios participam e influenciam tanto a definição dos requisitos do produto de software, quanto o julgamento de sua qualidade. Como reflexo dessa influência, parte dos componentes da qualidade são objetivos e passíveis de

avaliação e quantificação; parte são subjetivos e imprevisíveis e, portanto, não diretamente avaliáveis.

A obtenção de qualidade em software é alcançada atuando-se tanto sobre o processo de desenvolvimento quanto sobre os produtos intermediários e final. *Qualidade de processo* está relacionada a características como produtividade e custo, e *qualidade de produto* contribui para verificação da adequação do processo produtivo e de atributos de qualidade externos como, por exemplo, qualidade em uso.

A produção de software não é puramente um processo de manufatura; as atividades de produção podem ser melhor classificadas em projeto e criação, pseudo-manufatura e manufatura. Essa diferenciação implica a adoção de métodos gerenciais específicos, adequados ao esquema da produção de software. A incorporação no desenvolvimento de software de métodos ou técnicas originalmente destinados ao ambiente manufatureiro deve ser analisada profunda e adequadamente.

É tema comum em todos os trabalhos citados a confirmação de que não é possível, pelo menos até o momento, alcançar um nível de controle da qualidade completamente objetivo na área de software. A subjetividade tem participação significativa no julgamento final da qualidade, principalmente sob o ponto de vista do cliente.

A identificação e resolução dos elementos de alto risco dos projetos pode evitar grande parte dos problemas enfrentados.

Capítulo 5

A Evolução da Engenharia de Software

O sistema de produção de software é, ainda hoje, mal conhecido, e portanto, mal controlado. Essa afirmação é comprovada em diversos estudos [DeMA89] [PRE95] [WES90] [BRO87]. Os resultados dessa falta de controle são orçamentos estourados, cronogramas atrasados, requisitos deixando de ser atendidos e perda de clientes. Alguns autores defendem a hipótese de que o desenvolvimento de produtos de software é "diferente do resto da indústria", sugerindo a idéia de que o que serve bem para outros setores da indústria não se adequa à produção de software. Para outros, essa diferença é decorrente da imaturidade natural de uma indústria muito recente, que ainda deve evoluir para atingir os níveis hoje encontrados nos demais setores da produção. As duas abordagens parecem apontar para a realidade do dia-a-dia do trabalho de desenvolvimento de software. A seguir analisaremos alguns pontos relacionados a esses aspectos.

O termo *Engenharia de Software* surgiu em 1968 como título de um worshop que discutiu problemas relacionados a produção de software, no âmbito do Comitê de Ciências da OTAN - Organização do Tratado do Atlântico Norte, e ganhou popularidade na década seguinte. Atualmente, *Engenharia de Software* (ES) refere-se ao conjunto de processos gerenciais, ferramentas e atividades de projeto para desenvolvimento de programas de computador, ou seja, o conjunto de práticas relacionadas ao desenvolvimento de software. Porém, conforme discutiremos a seguir, o resultado prático da aplicação da ES difere sensivelmente dos resultados apresentados por outras áreas da engenharia.

O uso mais comum do termo *engenharia* refere-se à aplicação disciplinada do conhecimento científico na resolução de conflitos entre condicionantes e requisitos de um problema de importância prática e imediata. Sob esse enfoque, a engenharia fornece respostas a questões comuns a um determinado domínio tecnológico, através da organização, codificação e formatação do conhecimento científico, tornando-o diretamente útil e aplicável. Através da aplicação das práticas estabelecidas, profissionais medianos podem criar sistemas sofisticados, com boas características funcionais e alta confiabilidade. Exemplos clássicos da

organização dessas práticas são os manuais encontrados para diversas aplicações em Engenharia Química e Engenharia Civil.

Segundo Mary Shaw [SHA90], na área de software o conhecimento sobre técnicas que funcionam não é efetivamente compartilhado, nem existe um volume razoável de conhecimento sobre o processo de desenvolvimento, organizado de forma a facilitar a pronta utilização. Há um vazio a ser vencido entre as teorias desenvolvidas pela Ciência da Computação e a sua aplicação prática; essa lacuna é relacionada ao tipo de projeto e proporcional à complexidade e ao grau de inovação do produto a ser desenvolvido.

De uma forma geral, todo projeto pode ser enquadrado como rotineiro ou inovador. Projetos rotineiros envolvem soluções já conhecidas,e a repetição de práticas anteriormente utilizadas geralmente é suficiente. Já os projetos inovadores envolvem novos problemas e a pesquisa de novas soluções. Para auxiliar o desenvolvimento de projetos rotineiros, várias áreas de engenharia recorrem à utilização de documentação contendo as técnicas mais conhecidas, já verificadas e validadas, diretamente aplicáveis. No software, a documentação disponível é basicamente para auxílio na fase de projeto ou voltada ao usuário final; as notações não são adequadas ao registro em documentos como os manuais da engenharia citados no exemplo anterior. Com isso, a comunicação das experiências conhecidas é deficiente devido à falta de informação adequada.

Talvez devido a isso, projetos de software são usualmente tratados como inovadores, com certeza mais freqüentemente do que o necessário se tivéssemos recolhidas, organizadas e disponibilizadas as técnicas e experiências já conhecidas. Apesar de algumas técnicas de reuso - que capturam e organizam o conhecimento existente na forma de código (fonte ou executável), enfocarem esse aspecto, existem dificuldades na disponibilização de código reusável. Como já foi dito, componentes de software reusáveis devem ser construídos e documentados para fins de reuso, o que modifica o processo produtivo padrão adotado, desencorajando as empresas a efetuar o conseqüente investimento adicional sobre a produção. Tais modificações variam desde a adoção de características para aumento de portabilidade dos componentes reusáveis até a preparação e fornecimento de informações detalhadas sobre os possíveis ambientes permitidos para sua utilização. O reuso pode ser uma das soluções mais promissoras para a

construção de sistemas de software , que a cada dia tornam-se mais complexos, extensos e exigindo mais qualidade, pois, para tais sistemas, a técnica de "construir do zero" já oferece sérias limitações, principalmente em relação à taxa de produtividade [GAR95].

Para que o reuso torne-se uma prática no desenvolvimento, além do aumento da disponibilidade de componentes reusáveis é necessário que dois elementos sejam bem trabalhados, o que implicaria uma mudança nos hábitos e na cultura das equipes (ver Seção "Aspectos de Comportamento", neste Capítulo):

a) os programadores devem conhecer os módulos disponibilizados para reuso, isto é, deve haver documentação e treinamento adequados;
b) os programadores devem ser encorajados a adotar o reuso.

5.1 Do Artesanato à Profissionalização

Historicamente a engenharia desenvolveu-se a partir de práticas *ad hoc*, que envolviam técnicas mínimas para suporte à produção de uso pessoal ou, no máximo, à produção rotineira em pequena escala. Nesta fase artesanal, o talento pessoal, o uso extravagante de materiais e a baixa transmissão de informações entre os artesãos são características típicas. Num determinado momento, os produtos passam a ser mais amplamente aceitos e a demanda excede a oferta. Nesse ponto surge a prática comercial e as técnicas da fase artesanal não oferecem mais o suporte necessário à produção. Os problemas se avolumam e estimulam o desenvolvimento de uma ciência voltada ao suporte da produção (um exemplo bastante atual desse estágio é a área de Mecatrônica, uma nova ciência que integra engenharia eletrônica, engenharia mecânica e informática). Com o desenvolvimento, a ciência ganha maturidade, tornando-se um fator importante no esquema produtivo. É nesse ponto que surge a prática da engenharia como a conhecemos atualmente - a maturidade da ciência propicia a formação de profissionais educados em uma base científica sólida, capazes de aplicar a teoria na análise dos problemas e síntese das soluções. A Figura 5.1 ilustra o desenvolvimento descrito.

Figura 5.1 Evolução das práticas de produção e engenharia

5.2 A Maturidade da Tecnologia de Software

Nossa habilidade em construir sistemas de software certamente evoluiu bastante desde os primeiros programas; porém, o que verificamos é que esta evolução aconteceu de formas diferentes nos campos da ciência e da prática.

Até a década de 70 a programação era essencialmente feita *ad hoc*, como na produção artesanal descrita. Sistemas complexos eram criados, mas sua construção envolvia aspectos altamente empíricos ou era resultado do virtuosismo de alguns profissionais. Algumas pesquisas em *algoritmos** e *estruturas de dados** auxiliaram o desenvolvimento de sistemas; entretanto, a programação continuava sendo voltada à construção de programas pequenos, pois programas pequenos era tudo o que se podia manipular de forma previsível. Com o avanço das aplicações do software, em meados da década de 70 várias pesquisas enfocaram a construção de sistemas complexos, cujas especificações passavam a conter não apenas as necessidades de funcionalidade dos

programas, mas também características de desempenho e confiabilidade. Por essa época também surgiram as sementes da base científica de suporte da área. As técnicas desenvolvidas buscaram principalmente o suporte ao trabalho em equipe, possibilitando o estabelecimento da prática comercial no setor de software.

Analisando a evolução das práticas gerais de engenharia e produção e localizando-as no contexto de produção de software, verificamos que as técnicas de desenvolvimento de software encontram-se em parte na fase de artesanato e em parte na fase de comércio. A ciência começa a oferecer resultados e em alguns casos começam a surgir práticas profissionais de engenharia [WAS96].

5.3 O Meio-Ambiente de Software

Muitos autores referem-se ao conjunto de problemas encontrados no desenvolvimento de software como "a crise do software". Se analisarmos o significado de "crise" veremos que ele define algo pontual e decisivo no curso da evolução das coisas, uma "manifestação violenta e repentina de ruptura de equilíbrio"[1], o que absolutamente não se verifica na área de software, onde temos observado essa "crise" há pelo menos 30 anos, ou seja, desde que a produção de software assumiu um caráter de negócio. O que temos observado em relação a esses problemas tem na verdade características *crônicas*, algo "persistente, entranhado e de longa duração"[2].

Seja qual for a definição adotada, o que devemos ter em mente é que a grande maioria dos problemas enfrentados não são pontuais e nem específicos deste ou daquele sistema.

[1,2] Fonte: Dicionário Aurélio da Língua Portuguesa

5.3.1 Os Problemas

Os problemas que atingem o desenvolvimento de software podem ser caracterizados sob diversas perspectivas. A seguir, estão algumas constatações que, de uma forma geral, refletem a situação enfrentada [PRE95]:

- as estimativas de prazo e custos são freqüentemente imprecisas;
- a produtividade das pessoas da área é baixa;
- a qualidade do produto final é inadequada.

Essa situação é manifestada devido a algumas dificuldades bastante conhecidas na área:

⇨ não planejamos e nem dedicamos tempo para a coleta de dados sobre o processo de desenvolvimento; sem histórico, as estimativas são feitas precariamente e dos resultados, previsivelmente ruins, não podemos extrair indicadores para avaliação da produtividade ou da eficácia de novos métodos e ferramentas adotados.

⇨ os projetos são desenvolvidos sem um levantamento objetivo das necessidades e exigências dos clientes (como já apresentado no início deste Capítulo), que freqüentemente demonstram insatisfação em relação aos resultados finais;

⇨ a qualidade é freqüentemente "suspeita", pois falta a compreensão dos conceitos de confiabilidade e garantia da qualidade e a inclusão no processo de desenvolvimento dos itens destinados ao seu acompanhamento;

⇨ a manutenção dos sistemas existentes é custosa e difícil, pois raramente a *capacidade de manutenção do software* é enfatizada como um critério importante de qualidade.

5.3.2 Aspectos de Comportamento

As "estórias" a respeito de software demonstram ainda hoje uma forte influência do passado e do modo como eram produzidos os sistemas no início, em

detrimento dos avanços apresentados por outros setores e da profunda mudança cultural ocorrida. Pressman [PRE95] afirma que muitos dos problemas podem remeter-se a um conjunto de asserções surgidas nos primórdios do software, e que até hoje influenciam fortemente a cultura da área, propagando muita desinformação e confusão. Essas asserções seriam a base de "modelos implícitos" utilizados na formação do pensamento, e conseqüentemente do comportamento, do pessoal envolvido [WEIN93]. A seguir estão relacionadas algumas dessas asserções citadas por Pressman.

I) Gerência

- *Manuais de padrões e procedimentos para construção são suficientes para orientar o pessoal em tudo o que precisam saber.*
  Apesar desses manuais poderem existir, muitas vezes eles são desconhecidos pelos profissionais de desenvolvimento, pois não houve treinamento para sua utilização. Além disso, podem apresentar-se incompletos ou desatualizados.

- *A disponibilização de equipamentos de última geração melhora a qualidade.*
  A experiência mostra que é preciso muito mais do que belos equipamentos para se obter alta qualidade. Mais importante do que eles são as ferramentas e métodos de engenharia de software (que, apesar disso, são muito pouco conhecidos e utilizados).

- *Atrasos nos prazos podem ser compensados com o aumento da equipe.*
  Este é um dos maiores e mais comum dos erros. Deve-se lembrar que a produção de software não é um processo mecânico como a manufatura [HAM96]. Diversos estudos constatam que acrescentar pessoas em um projeto atrasado torna-o ainda mais atrasado. A entrada de novos elementos provoca descontinuidade no desenvolvimento, na medida em que há a necessidade desses elementos serem instruídos sobre detalhes do andamento do projeto, que só os que já estavam trabalhando conhecem. Ou seja, ocorre um desvio da atenção do grupo. A inclusão de elementos na equipe deve ser feita de forma planejada e coordenada com o desenvolvimento do projeto. Além disso, as grandes variações de desempenho entre profissionais, bem conhecidas no setor, também são um fator complicador. Pode-se esperar que uma equipe apresente diferenças de até quatro para um, mesmo que não tenha profissionais excelentes ou muito fracos trabalhando [DeMA89]. Essa regra se

aplica em particular à variação do índice de introdução de erros; funcionários abaixo da média são responsáveis por duas vezes mais erros do que os acima da média. Se lembrarmos que as atividades de identificação e remoção de erros são responsáveis por 35% a 55% do custo e esforço de desenvolvimento, muitas vezes a retirada de um elemento fraco (com índice elevado de inclusão de erros) pode ser mais produtivo do que acrescentar um competente.

II) Clientes

- *Uma declaração geral do que se deseja com o sistema é suficiente para iniciá-lo, os detalhes podem ser completados depois.*
  Como já citado anteriormente, uma definição precária dos requisitos do sistema é a causa principal dos fracassos. É necessário não só um levantamento o mais completo possível dos requisitos conhecidos, mas também a indicação clara das restrições de projeto, critérios de aceitação final, etc.

- *As modificações dos requisitos do projeto podem ser facilmente acomodadas pois o software é flexível o suficiente para isso.*
  Devemos levar em consideração que os requisitos inicialmente levantados podem modificar-se ao longo do desenvolvimento, porém é um erro pensar que essas modificações poderão ser incorporadas ao sistema facilmente. O impacto da mudança varia de acordo com o momento em que ela é introduzida. Quanto mais adiantado o projeto, maior o custo das modificações.

III) Profissionais de Desenvolvimento

- *Nosso trabalho termina com o programa escrito e funcionando.*
  Dados históricos demonstram que quanto mais cedo se começa a escrever o programa, mais ele demora para ficar pronto. Isto porque a pressa em escrever o código normalmente implica na redução do tempo gasto com o planejamento e com fases de projeto anteriores. Sabe-se que atualmente entre 50 e 70% de todo o esforço de desenvolvimento é gasto após o sistema ser entregue (isso obviamente devido ao baixo nível de planejamento e controle existente sobre o sistema de produção).

- *Só é possível avaliar a qualidade depois dos programas estarem funcionando.* Grande parte dos erros apresentados podem ser detectados em fases iniciais de projeto, com o uso de técnicas de verificação, validação e testes, desde que tenham sido planejadas e inseridas no processo de produção.

- *O que deve ser entregue ao cliente em um projeto bem sucedido é o sistema funcionando.*
  O conjunto de programas funcionando é apenas um dos componentes do produto final. Além desses devem ter sido preparados a documentação de usuário e de projeto (que auxilia na manutenção futura do sistema), a especificação dos testes efetuados, a especificação dos serviços de suporte técnico para implantação e manutenção oferecidos pelo produtor, etc. Isso é o mínimo necessário.

5.3.3 Equipamentos e Ferramentas

Em [RAM96] Ramamoorthy e Tsai relacionam dois fatores que influenciam fortemente o processo de desenvolvimento de software: o avanço dos sistemas de software por diversos domínios e a evolução tecnológica. O avanço das aplicações do software (como os citados na introdução deste trabalho) forçou os sistemas a tornarem-se aplicações complexas, envolvendo muitas vezes características de *tempo real** e *multimídia**. Essas novas aplicações foram determinantes no espantoso desenvolvimento que atualmente podemos verificar em outras áreas, entre elas o hardware, as comunicações e, mais especificamente na área da computação, os sistemas de suporte (bancos de dados, sistemas operacionais, por exemplo); estes, por sua vez, levaram a um avanço ainda maior dos sistemas de software. A Figura 5.2 ilustra essa idéia de inter-relação.

O desenvolvimento tecnológico propiciou a construção de computadores mais compactos, mais rápidos, mais baratos e com mais recursos integrados - som e imagem, por exemplo, o que expandiu o domínio de sua utilização e, consequentemente, o domínio das aplicações do software.

Figura 5.2 Fatores de influência no avanço das aplicações de software

Estes fatores influenciam fortemente o desenvolvimento das aplicações de software, pois existe não só a necessidade contínua dos sistemas incorporarem as novas facilidades disponibilizadas pelo avanço tecnológico como também a do pessoal de desenvolvimento de adaptar-se às transformações nas ferramentas utilizadas no processo (linguagens de programação, sistemas de suporte, equipamentos computacionais, etc) que, da mesma forma como os outros sistemas de software, modificam-se continuamente.

5.4 Síntese

A habilidade em desenvolver software certamente cresceu nos últimos 40 anos; porém, o progresso observado na área do desenvolvimento da ciência não teve seu equivalente na área das práticas de desenvolvimento. Os resultados científicos devem ser maduros o suficiente para realmente modelarem os problemas encontrados na produção. Devem, além disso, ser organizados e formatados de forma a tornarem-se utilizáveis pelo pessoal de desenvolvimento.

Os problemas enfrentados não devem ser caracterizados como "crise" e nem são específicos de determinadas classes de sistemas. São, na verdade, aspectos crônicos presentes na maioria dos problemas e que vêm acompanhando o processo produtivo apesar das evoluções conseguidas.

Dificuldades bastante conhecidas na área poderiam estar relacionadas ao modo como clientes, gerência e técnicos pensam e, conseqüentemente, se comportam com relação ao software.

O avanço tecnológico, a expansão no domínio das aplicações e a evolução dos sistemas de suporte são fatores que se inter-relacionam e também influenciam fortemente o desenvolvimento de software.

Capítulo 6

Iniciativas na Área de Qualidade em Software

6.1 A Família de Normas ISO 9000

A Série ISO 9000 despertou muito interesse desde que foi introduzida em 1987. Esse interesse foi motivado principalmente pelo foco na qualidade adotado nas Normas, mas deveu-se também à imposição do certificado ISO 9000 que alguns países da Europa adotaram para seus fornecedores. O conjunto de normas que formam a família ISO 9000 foi desenvolvido com o objetivo de orientar as organizações industriais, comerciais ou governamentais a estabelecerem sistemas da qualidade eficazes e eficientes, que resultem na melhoria contínua dos produtos e no aumento da satisfação dos seus clientes e colaboradores. As Normas enfocam aspectos relativos à gestão da qualidade e elementos dos sistemas da qualidade, incluindo diretrizes, guias e modelos para garantia da qualidade.

A certificação ISO 9000 é atualmente um dos meios mais utilizados para o reconhecimento público dos sistemas de qualidade adotados nas organizações em geral e nas empresas de software em particular. O certificado ISO 9000 indica que há evidências de que a organização seja capaz de oferecer produtos ou serviços de qualidade, mas não é, em absoluto, capaz de avaliar diretamente a qualidade de qualquer um dos produtos ou serviços dessa organização.

Quando uma empresa busca a certificação, a entidade certificadora escolhida determina qual das Normas da série será aplicada, baseando-se no tipo das atividades de negócio para as quais pretende-se obter o certificado. As normas ISO 9001 – *Sistemas da Qualidade – Modelo para garantia da qualidade em projetos, desenvolvimento, produção, instalação e serviços associados*, ISO 9000-3 – *Normas de Gestão da Qualidade e Garantia da Qualidade – Parte 3: Diretrizes para a aplicação da ISO 9001 ao desenvolvimento, fornecimento e manutenção de software*, e ISO 9004-2– *Gestão da qualidade e elementos do sistema da qualidade – Parte 2: Diretrizes para serviços*, formam um conjunto básico que especifica os principais requisitos para um sistema de qualidade de software. As

empresas envolvidas no desenvolvimento ou manutenção de software são avaliadas em relação à norma ISO 9001, usando as diretrizes especificadas pela norma ISO 9000-3. As diretrizes apresentadas na norma ISO 9004-2 também poderão ser utilizadas, no caso da avaliação de serviços relacionados ao suporte, manutenção e aperfeiçoamento de produtos de software. Se os serviços oferecidos pela empresa não envolverem o projeto de software, ou este for considerado trivial, a norma ISO 9002 em conjunto com a ISO 9004-2 poderão ser usadas. Esta situação aplica-se às empresas de consultoria, treinamento e serviços de informática.

Existem duas razões principais para que uma empresa de software busque a certificação ISO 9000:

- ganho de vantagem competitiva;
- estabelecimento de metas objetivas para implantação de programas de qualidade.

O mercado de software é globalizado, com grande competição em vários países. Cresce a cada dia o número de multinacionais e organizações do setor público que requerem o certificado ISO 9000 de seus fornecedores de software. Essa tendência é particularmente importante na Europa, onde algumas comunidades passaram a adotar as normas internacionais ISO 9000 como base para uma política européia de avaliação de conformidade, principalmente a partir do estabelecimento do Mercado Comum Europeu, em 1994. Estima-se que 5000 empresas européias obterão o certificado ISO 9001 para seus sistemas de qualidade em software até o final de 1999 [SEN94]. Num futuro próximo, não ser certificado significará uma desvantagem.

Mesmo equipes de desenvolvimento internas, que produzem software para consumo próprio em algumas organizações e que, portanto, não competem diretamente com o mercado, estão sendo levadas a justificar seus custos contra serviços terceirizados. A certificação, neste caso, pode ajudá-las a competir, num mesmo nível, com os serviços de terceiros, além de possibilitar o reconhecimento da competência de seus sistemas de qualidade internamente na organização.

Quando o objetivo principal da empresa é estabelecer marcos para implantação de programas de qualidade, a certificação ISO 9000 pode favorecer a

especificação de metas claras e objetivas, que facilitam a avaliação e medição do progresso das atividades do programa. O custo da certificação dependerá de fatores como o tamanho da empresa e o número de departamentos envolvidos. Deve-se também considerar os benefícios indiretos advindos do processo de certificação: obter o certificado promove o reconhecimento dos empregados e eleva o moral da organização.

6.2 A Certificação ISO 9000 Dentro e Fora da Indústria de Software

O crescimento na demanda por certificados ISO 9000 fora da indústria de software foi explosivo a partir de 1987, quando a Norma foi introduzida, e tem dobrado a cada ano em alguns países. A Europa é a líder em número de empresas certificadas, sendo a Inglaterra o principal expoente. O interesse pela certificação fora da Europa é menor, mas tem crescido bastante a partir do reconhecimento de que, para muitas indústrias, o certificado é quase obrigatório para o sucesso no mercado europeu.

A indústria de software despertou para a certificação ISO 9000 bem mais tarde, mas há sinais de que a demanda pela certificação crescerá tão rápido quanto cresceu para as demais indústrias. O impulso desse crescimento tem sido determinado por dois fatores principais (conforme visto anteriormente): de um lado as instituições governamentais e empresas multinacionais, que requerem o certificado para qualificar seus fornecedores e, do outro, os próprios fornecedores buscando obter vantagem competitiva.

As pesquisas do MCT [MCT95][MCT97] confirmam essa tendência também no Brasil: em 1995, 1,8% das empresas pesquisadas possuíam certificados ISO 9001 e 0,2% ISO 9002; em 1997 esses números passaram para 6,1% e 1,8%, respectivamente, contabilizados até julho de 1997.

6.3 O Modelo CMM

O CMM – Capability Maturity Model, é atualmente o modelo mais conhecido no Brasil para avaliação da qualidade de software: 29% das empresas o conhecem e 5% o usam, segundo a pesquisa MCT [MCT97]. Isso deve-se em parte ao fato de que, apesar da possível necessidade de acompanhamento de consultoria externa especializada para sua aplicação, o documento do CMM em si é de domínio público [SEI99] [PAU95].

O CMM é usado tanto como um modelo para classificação do sistema de qualidade de uma equipe ou empresa produtora de software quanto para melhorar o processo de desenvolvimento, auxiliando as organizações a planejar, desenvolver e implementar mudanças. Trata-se de um conjunto de métodos e ações que vêm sendo aperfeiçoadas pelo Software Engineering Institute (SEI) da Universidade Carnegie Mellon, a partir de propostas de Philip Crosby, e foi originalmente desenvolvido para o Departamento de Defesa dos EUA com o objetivo de quantificar a capacidade de seus fornecedores de produzir, consistente e previsivelmente, software de alta qualidade.

As empresas ou equipes de desenvolvimento podem usar o modelo para identificar o estado atual de seus processos e, a partir daí, melhorá-los seguindo práticas sugeridas de acordo com seu nível atual de maturidade. Os clientes ou terceiros, também podem utilizar o modelo CMM para estimar o potencial de riscos associados a contratos com possíveis fornecedores, a partir da avaliação de seus níveis de maturidade.

O modelo dos níveis de maturidade foi definido como uma escala ascendente de estágios de evolução dos processos da organização. Cada estágio define um nível de capacidade[1] específico para os processos de software da organização: Nível 1- "inicial" ou "ad hoc", Nível 2 – "repetível", Nível 3 – "definido", Nível 4 – "gerenciado" e Nível 5 - "otimizado" (Figura 6.2).

---

[1] capacidade de processo: a capacidade de processo de software descreve a faixa de resultados esperados que podem ser alcançados pelo processo.

O CMM descreve elementos-chaves para um processo de software capaz e efetivo, cobrindo práticas que envolvem planejamento, engenharia e gerência em desenvolvimento e manutenção de software. O modelo define 5 níveis crescentes de maturidade

Com exceção do Nível 1, caracterizado tipicamente pela ausência de um ambiente estável para desenvolvimento ou manutenção de software, os demais níveis são compostos por uma série de *áreas-chave de processo* (key process areas ou KPAs), que identificam os pontos onde a organização deve focar seus esforços para elevar a capacidade de seus processos de software. Cada *área-chave* é posteriormente decomposta em *características* e *práticas-chave* que descrevem as atividades e infra-estrutura necessárias para a efetiva implementação e institucionalização da *área-chave* (Figura 6.1).

Os 5 níveis de maturidade

Segundo o modelo CMM, à medida que uma organização estabelece e melhora o processo que rege o desenvolvimento e a manutenção de seus produtos de software, esta organização ascende em maturidade. A cada um dos níveis definidos no modelo corresponde uma estrutura típica de processo de software presente na organização, conforme representado na Figura 6.2.

Nível 1 – Inicial

Uma organização de Nível 1 tipicamente não apresenta um ambiente estável para desenvolvimento e manutenção de software. Nestas organizações, a capacidade do processo é indeterminada e o sucesso é altamente dependente da habilidade da equipe (ou de elementos da equipe, em particular). A atuação de uma organização de Nível 1 é marcada pela "reação ao que acontece", pois o planejamento, quando existe, não é efetivo.

Nível 2 – Repetível

Neste Nível os custos e cronogramas são controlados e estabelecidos com base em experiências de projetos anteriores. Não há estudo de riscos e os problemas só são identificados quando já aconteceram. A capacidade do processo pode ser

classificada como "disciplinada", pois o nível de planejamento e acompanhamento estabelecido permitem que atitudes bem sucedidas possam ser repetidas, porém não há regras ou padrões gerais na organização para o processo de desenvolvimento como um todo (não há integração).

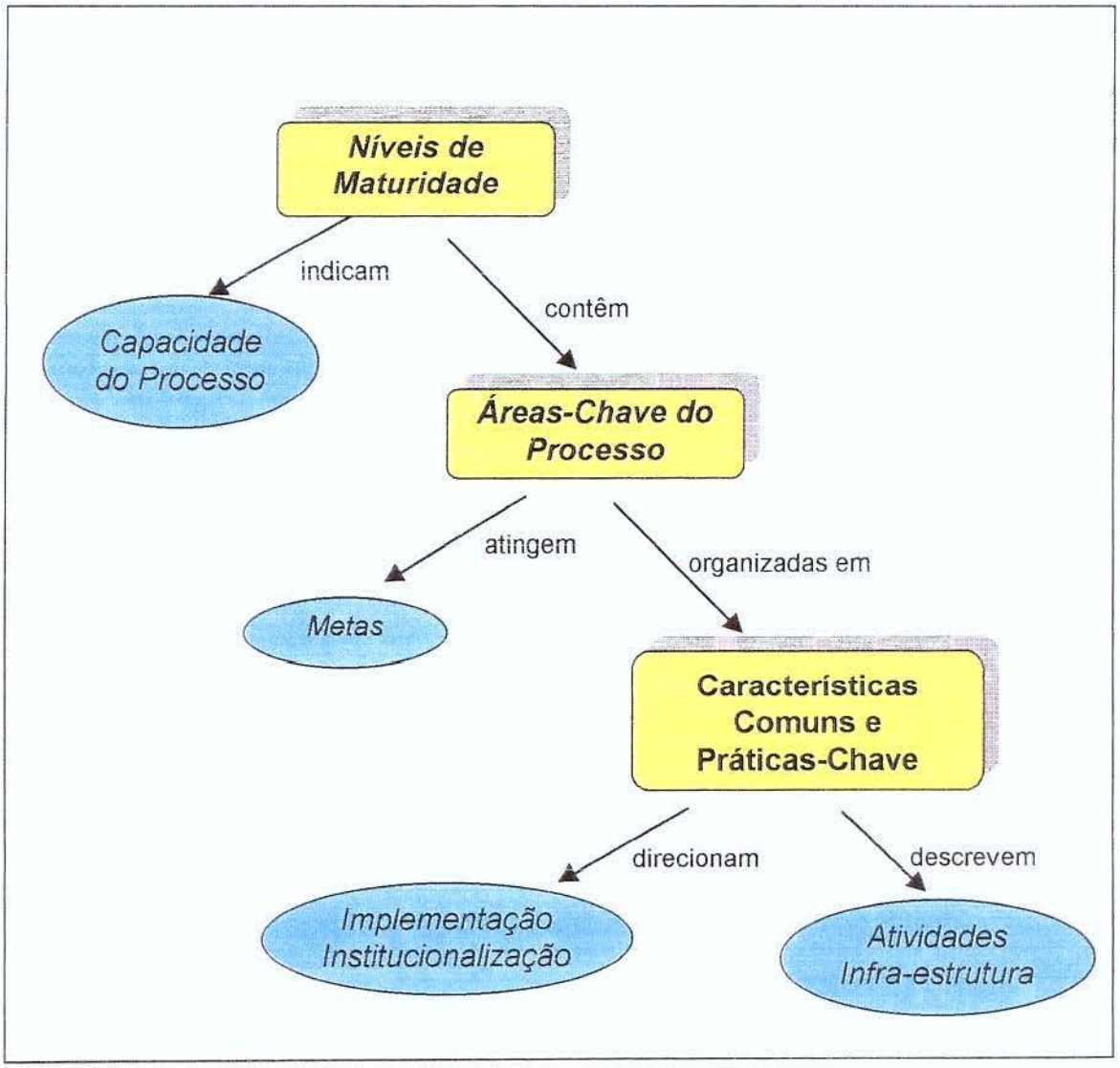


Figura 6.1 A estrutura do modelo CMM

Nível 3 – Definido

Organizações de Nível 3 possuem um processo padrão para desenvolvimento e manutenção de software bem estabelecido e integrado, que pode ser adaptado

para as peculiaridades de cada projeto em particular. Tanto as atividades de gerência quanto as de engenharia estão estabelecidas. A capacidade do processo é definida como padronizada e consistente.

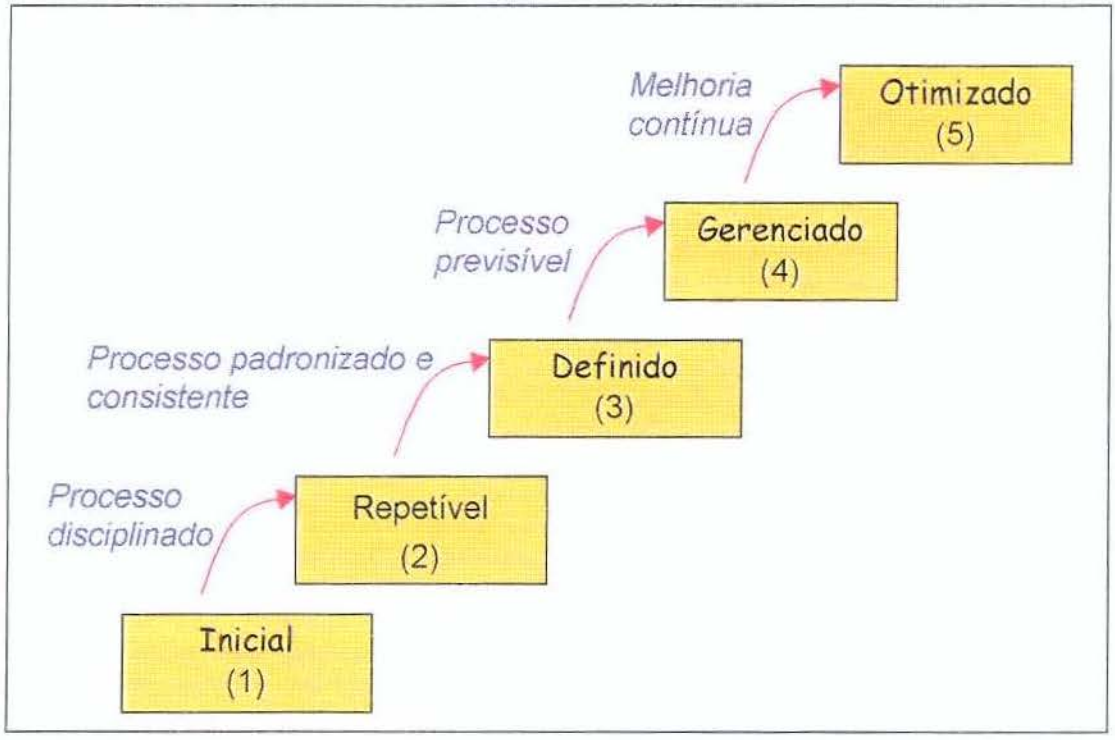


Figura 6.2 Os 5 níveis de maturidade do processo de software segundo o modelo CMM

Nível 4 – Gerenciado

No Nível Gerenciado a organização é capaz de definir metas quantitativas e qualitativas para produtos e processos. A produtividade e qualidade são itens de medição incluídos no programa da organização. Os processos de software são acompanhados por medições consistentes e bem definidas, que estabelecem as bases para a avaliação tanto dos produtos quanto dos processos. A capacidade do processo neste Nível pode ser definida como previsível, pois o processo é medido e opera dentro de limites, o que permite à organização prever tendências tanto nos produtos quanto nos processos.

Nível 5 – Otimizado

Neste Nível a organização como um todo está focada no processo de melhoria contínua, e atua de forma pró-ativa. As ações são tomadas para prevenir a ocorrência de defeitos. Os dados de acompanhamento são usados para a realização de análises de custo-benefício de novas tecnologias e para propor mudanças no processo de software da organização.

A escala dos 5 níveis de maturidade representa, caracteriza e organiza a estrutura principal do modelo CMM, a partir da qual todas as demais ferramentas e metodologias se aplicam. A Tabela 6.1 a seguir apresenta as *áreas-chave de processo* para cada um dos níveis de maturidade:

Tabela 6.1 – Áreas-chave por Nível segundo o Modelo CMM

Nível	**Foco principal**	**Áreas-Chave de Processo**
Inicial	Competência individual Heroísmo	
Repetível	Gerência de projetos	Gerência de requisitos; Planejamento e controle de projetos de software; Gerência de subcontratos de software; Garantia da qualidade de software; Gerência de configuração de software;
Definido	Processos de engenharia Suporte organizacional	Foco no processo organizacional; Definição do processo organizacional; Programa de treinamento; Gerência integrada de software; Engenharia do produto de software; Coordenação da interação entre grupos; Revisões do tipo *peer-review* *
Gerenciado	Qualidade do processo e do produto	Gerência quantitativa de processos; Gerência da qualidade de software;
Otimizado	Melhoria contínua de processos	Prevenção de defeitos; Gerência de evolução tecnológica; Gerência de evolução de processos;

6.4 O Projeto SPICE

O projeto SPICE – Software Process Improvement and Capability Determination, estabelecido pela ISO/IEC em 1993, é uma iniciativa internacional de apoio ao desenvolvimento de uma norma internacional para avaliação do processo de software, e foi baseado nas características dos melhores modelos de avaliação de processos existentes, como o CMM, Trillium, Software Technology Diagnostic (STD) e Bootstrap, além das normas ISO 9001 e ISO 9000-3 [SPICE99][HAA94][BAS94]. O projeto possui 3 objetivos principais:

- desenvolver uma proposta de norma de avaliação do processo de software;
- conduzir ensaios dessa Norma na área industrial;
- promover a transferência da tecnologia relativa à avaliação do processo de software para toda a indústria de software.

O primeiro objetivo foi alcançado em 1995, com a liberação da Versão 1 do projeto de norma. Este documento, dando prosseguimento ao processo para desenvolvimento de normas internacionais, deverá transformar-se na norma ISO 15504.

A condução dos ensaios é o foco principal do projeto atualmente. Foram criados 5 Centros Técnicos Internacionais, responsáveis pela coordenação das atividades do projeto, que envolve entidades normalizadoras, desenvolvedores de software e entidades de pesquisa e ensino em mais de 20 países, incluindo o Brasil.

Os percentuais observados pela pesquisa MCT em relação ao SPICE são inferiores ao modelo CMM: 18% das empresas conhecem o projeto e 1% usam. Assim como o modelo CMM, o projeto SPICE considera que a avaliação do processo pode direcionar efetivamente os programas de melhoria de qualidade de software, conforme ilustrado na Figura 6.3.

A proposta principal do SPICE é consolidar as diversas iniciativas na área de avaliação de processo, propondo um padrão internacional, cujos benefícios principais são:

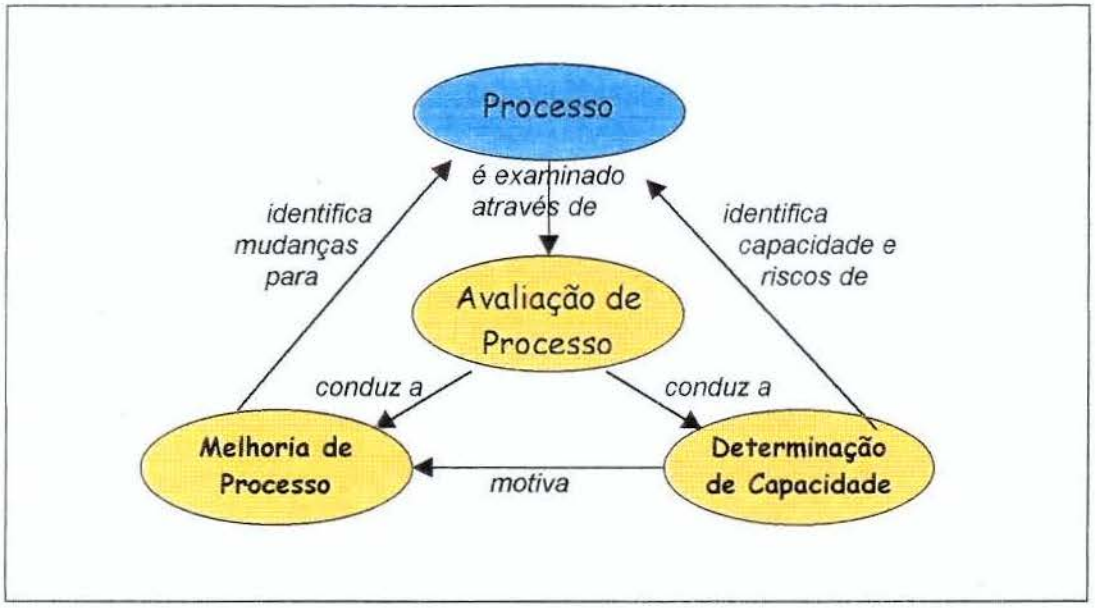


Figura 6.3 Integração entre Avaliação de Processo, Melhoria de Processo e Determinação de Capacidade segundo o projeto SPICE

a) para a indústria de software:

- os fornecedores de software poderão ser submetidos a um esquema padronizado de avaliação (atualmente, vários esquemas diferentes são usados);
- as organizações de desenvolvimento de software terão uma ferramenta para iniciar e sustentar um processo de melhoria contínua;
- a gerência terá meios de assegurar o alinhamento do desenvolvimento de software com os negócios da empresa, oferecendo o suporte adequado às suas necessidades.

b) para os compradores de software:

- os compradores poderão determinar a capacidade de fornecedores de software e avaliar os riscos envolvidos em sua seleção.

O conjunto de documentos SPICE oferece uma estrutura básica para a avaliação do processo de software, que poderá ser utilizada por organizações envolvidas

com planejamento, gerência, controle e melhoria dos processos de aquisição, fornecimento, desenvolvimento, operação, manutenção e suporte de software.

6.5 O Conjunto de Normas ISO 9126

As normas do conjunto ISO 9126 foram desenvolvidas para auxiliar a avaliação da qualidade de software enfocando a avaliação do produto, dentro da iniciativa européia do projeto Esprit SCOPE – Software Certification Programme in Europe. A ISO 9126 – *Avaliação do produto de software - Características de qualidade e diretrizes para o seu uso*, apresenta seis características que definem qualidade em software, aplicáveis na especificação dos requisitos e na avaliação da qualidade dos produtos de software ao longo do ciclo de desenvolvimento e produção. São elas [ABNT96]:

- *Funcionalidade*: atributos que evidenciam as funções que satisfazem as necessidades do produto;

- *Confiabilidade*: atributos que evidenciam a capacidade do produto de manter seu nível de desempenho sob condições estabelecidas;

- *Usabilidade*: atributos que evidenciam o esforço necessário para se utilizar o produto por um conjunto de usuários;

- *Eficiência*: atributos que evidenciam o relacionamento entre o nível de desempenho do produto e a quantidade de recursos usados;

- *Manutenibilidade*: atributos que evidenciam o esforço necessário para se fazer modificações no produto;

- *Portabilidade*: atributos que evidenciam a capacidade do produto de ser transferido de um ambiente para outro.

As características e o modelo do processo de avaliação apresentados permitem que tanto a definição quanto a avaliação de qualidade de produtos de software sejam executadas sob uma base de conceitos única.

A importância de cada característica de qualidade varia dependendo da classe de software e da visão de qualidade presente. Para sistemas de missão crítica, por exemplo, a *confiabilidade* pode ser a mais importante, sistemas de tempo real podem indicar a *eficiência* como a mais importante, assim como a *usabilidade* pode ser a mais importante para sistemas com marcada interação com o usuário. As visões de qualidade abordadas pela Norma são:

- Visão do usuário, interessados principalmente no uso do software e em seu desempenho;

- Visão da equipe de desenvolvimento, interessada em satisfazer os requisitos especificados e nas características de manutenção do produto.

- Visão do gerente, interessado na qualidade de uma forma geral e em critérios de otimização da qualidade dentro das limitações de custo, recursos e prazos.

Segundo a Norma, entende-se por avaliação o processo que envolve a medição, pontuação e julgamento de produtos. Atualmente existem poucas métricas de aceitação geral destinadas a medir as características descritas, o que implica que as organizações deverão estabelecer os seus próprios modelos de processos de avaliação e métodos para criação e validação de métricas. O que a Norma apresenta é apenas um modelo geral que auxilia o planejamento de processos de avaliação.

O modelo apresentado é composto de três estágios: definição de requisitos da qualidade, preparação da avaliação e procedimento de avaliação. No estágio de definição dos requisitos da qualidade o objetivo é especificar os requisitos que expressem as exigências a serem atendidas pelo produto. A preparação da avaliação envolve as etapas de seleção de métricas de qualidade, definição dos níveis de pontuação e definição dos critérios de julgamento, que devem ser estabelecidos para cada avaliação, especificamente. Durante a avaliação as métricas selecionadas serão aplicadas ao produto e o valor medido será pontuado para posterior julgamento. O resultado final é a decisão quanto à aceitação ou rejeição, ou liberação ou não liberação do produto de software.

Uma das iniciativas atuais que consideram as diretrizes apresentadas pela ISO 9126 é o projeto Squid, apoiado pelo Programa Europeu de Tecnologias de Informação Esprit. O Squid é um conjunto de métodos e ferramentas para o planejamento, controle e avaliação da qualidade dos produtos de software durante o desenvolvimento [BOE99]. O enfoque do projeto propõe a definição dos requisitos de qualidade de produtos a partir de metas para características de qualidade externas, que serão rastreadas durante as etapas de desenvolvimento a partir de características de qualidade internas (como tamanho, taxa de falhas, cobertura de testes, entre outros).

6.6 Considerações Sobre Aplicabilidade

6.6.1 A Certificação ISO 9000 em Software

Como visto anteriormente, as normas ISO 9001 e 9000-3 oferecem uma base que especifica os principais requisitos para um sistema de qualidade em software. Porém, a norma ISO 9001 tem ênfase no controle de qualidade na manufatura. Nesse meio, a aquisição de produtos ocorre normalmente em bases contratuais formais, com detalhamento abrangente e correto de requisitos. Conforme analisado no Capítulo 3, tais condições raramente se aplicam a produtos como software.

Produtos de software são inerentemente complexos e seu projeto, desenvolvimento, verificação, validação e manutenção são processos, no mínimo, desafiadores. Os elementos analisados indicam a necessidade de um forte enfoque na qualidade total, principalmente quanto ao atendimento e satisfação dos clientes e à melhoria contínua. O suporte oferecido pela Norma ISO 9001 para melhoria contínua é limitado nesse aspecto, restringindo-se basicamente ao controle das não-conformidades e recomendando ações de prevenção e correção [COA94] [PAU95]. Sob esse ponto de vista, modelos altamente relacionados à melhoria contínua como o Kaisen japonês, os requisitos do Prêmio Malcolm Baldridge de Qualidade e, particularmente em software, o modelo CMM e o SPICE, contrastam com a norma ISO 9001.

Outro aspecto importante é quanto à natureza da avaliação ou certificação da qualidade. A certificação ISO 9000 implica em um processo de auditoria externa, motivado principalmente pela necessidade do certificado para fechamento de contratos. Sob o ponto de vista da qualidade total, uma organização madura obterá um retorno muito mais proveitoso em processos de auto-avaliação do que numa auditoria externa, pois na auto-avaliação o propósito é a melhoria da organização e, consequentemente, do ambiente de trabalho como um todo. [PAU95] e [COA94] trazem observações importantes sobre o processo de auditoria ISO 9001 especificamente para software; em alguns casos foram encontradas diferenças significativas em relação à capacidade indicada pelo certificado ISO 9000 e a capacidade indicada pelo modelo CMM a uma mesma organização. À primeira vista, apenas organizações de Nível 3 ou 4 no modelo

CMM (ver "O modelo CMM") seriam capazes de obter o certificado ISO 9001; porém, há organizações de Nível 1 certificadas. Segundo o autor, uma das razões para essa discrepância seria a interpretação da Norma pelos auditores de formas diferentes, devido ao alto nível de abstração da ISO 9001 e à falta de profissionais auditores habilitados a aplicarem a Norma no ambiente particular de software. O programa TickIt, uma iniciativa do governo inglês, busca contribuir com esse aspecto, oferecendo um esquema mais adequado para a certificação ISO 9001 em software.

6.6.2 A Norma ISO 9001 no Contexto da Pequena Empresa

As normas da série ISO 9000 foram idealizadas para atender a qualquer tipo de organização; porém, algumas empresas, principalmente as pequenas, têm dificuldades em implementá-las [MART].

No Brasil, segundo a pesquisa MCT, há a predominância de micro e pequenas empresas no setor de software, com um percentual de 77% segundo classificação por número de funcionários (até 50 pessoas), e 68% segundo classificação por valor de comercialização (até R$ 720 mil / ano).

O mais importante é analisar a forma como a pequena empresa funciona. Nessas organizações o negócio é conduzido por um gerente, que é geralmente o proprietário ou tem relações estreitas com ele. Esse gerente sabe por contato pessoal como o trabalho está sendo realizado e conhece os procedimentos que cada um deve seguir. A necessidade de documentação é, portanto, significativamente reduzida. O gerente também é o responsável pelo controle da qualidade, analisando o esquema adotado através de pequenas auditorias internas e supervisionando diretamente as atividades da empresa. Neste ambiente, a garantia da qualidade depende muito mais da competência dos funcionários do que das instruções documentadas.

Algumas vantagens e desvantagens ao se implantar um programa de certificação estão resumidamente apresentadas na Tabela a seguir:

Tabela 6.2 Vantagens e desvantagens na implantação de programas de certificação nas pequenas empresas

Vantagens	Desvantagens
• *possibilidade de abertura de novos mercados;* • *redução de desperdícios;* • *aumento da vantagem competitiva;* • *alcance de credibilidade para a estrutura de gerência da qualidade da empresa;* • *fortalecimento do controle dos processos da empresa;* • *aumento da eficiência e, potencialmente, dos lucros;* • *redução de gastos com garantias;* • *reconhecimento potencial no Mercado Comum Europeu.*	• *aumento do tempo gasto com o desenvolvimento do sistema de qualidade;* • *custo para a obtenção e manutenção da certificação;* • *possibilidade de tornar os sistemas mais rígidos e burocráticos, com conseqüente perda de flexibilidade e velocidade de mudanças;* • *tempo requerido para a qualificação e obtenção do certificado;* • *dificuldades na implementação das Normas;* • *resistência por parte dos funcionários e problemas de relacionamento interno.*

A aplicabilidade das normas ISO 9000 precisa ser analisada *antes da determinação da necessidade do certificado de conformidade.* Devido à forma como as pequenas empresas operam, a exigência indiscriminada do certificado por parte dos compradores pode não trazer benefício algum. Ao contrário, pode fazer com que os próprios compradores percam bons fornecedores e prejudicar os negócios de muitas empresas. Existem outras opções, além da certificação, que podem ser úteis nestes casos, como os certificados de qualidade ou as inspeções diretas por parte dos compradores. Antes de se solicitar a certificação deve-se analisar cada opção, considerando, caso a caso, suas vantagens e desvantagens.

6.6.3 O Uso de Normas de Qualidade na Área de Software

Pode-se afirmar, sem sombra de dúvida, que o uso das normas e padrões atuais na área de qualidade de software é controverso, principalmente se for considerada a relação entre o uso dos padrões e a melhoria da qualidade nos produtos de software. A unanimidade só é alcançada em um ponto: o que temos hoje disponível não é suficiente para atender as necessidades de desenvolvimento e qualidade. Enquanto alguns acreditam que mesmo normas imperfeitas são melhores do que nada, outros contra-atacam afirmando que padrões só são efetivos se sua adoção melhorar a qualidade, dentro de uma relação custo-benefício adequada e que, até o momento, não há evidências disso para nenhum dos padrões disponíveis.

Schneidwind e Fenton apresentam uma boa discussão sobre o assunto [SCH96]. Segundo Schneidwind, o uso das diretrizes e práticas indicadas pelas normas leva à melhoria de qualidade do produto no sentido de que a utilização das normas requer que os desenvolvedores adotem processos rigorosos, disciplinados e repetíveis , o que implicaria num avanço natural da qualidade de seus processos. A principal vantagem na utilização dessas normas estaria na comunicação e registro das informações técnicas de forma consensual e padronizada, já que as normas internacionais são elaboradas considerando a visão de toda a comunidade e não de um indivíduo ou organização em especial. Se não fossem as normas essas informações estariam sujeitas a diferentes definições e interpretações, o que dificultaria bastante tanto os processos internos quanto os externos de comunicação. Além disso, a utilização das normas expõe o desenvolvedor aos métodos e práticas correntes da engenharia de software, influenciando seus processos de trabalho e educação, o que indiretamente poderia provocar o início de uma mudança cultural bastante positiva.

Os contrapontos apresentados por Fenton consideram principalmente a maturidade e aplicabilidade das normas atualmente disponíveis. Existem hoje mais de 250 normas dentro do escopo da engenharia de software, variando desde a apresentação de procedimentos gerais de qualidade até a definição de técnicas bastante específicas. Não há consenso em relação a um conjunto das melhores práticas, nem na seleção das técnicas a serem sempre aplicadas. Como conseqüência, os desenvolvedores estão expostos a diferentes modelos para um

mesmo objetivo. Segundo Fenton, esse fato sugere que, no domínio da engenharia de software, poucos pontos estão suficientemente maduros para serem normatizados.

Se analisarmos as normas gerais de engenharia, verificaremos que a confirmação da qualidade de produtos baseia-se na especificação de propriedades que o produto deve satisfazer, incluindo aqui testes extensivos. No caso de software, praticamente a totalidade dos padrões enfocam o processo de produção, e não o produto. Deve-se lembrar, entretanto, que até o momento não foram acumulados dados suficientes para comprovar que "*bons processos implicam bons produtos*", ou estabelecer a relação entre as práticas utilizadas e a qualidade liberada no produto final. Poucos estudos se dedicam a verificação científica dos métodos e práticas propostos [FEN94]. A ênfase dada ao processo pelos padrões atualmente disponíveis pode estar negligenciando pontos importantes da qualidade do produto final, simplesmente por ignorarem métodos específicos para produtos (ver Capítulo 2, "Algumas Considerações sobre Qualidade de Software"), ou por não terem recebido o embasamento essencial das evidências colhidas por experimentação.

Sabemos também que as normas tradicionais de outros setores produtivos são organizadas como um conjunto de requisitos obrigatórios, apresentados de forma suficientemente precisa para possibilitar a verificação de conformidade. No contexto da aplicação de padrões visando verificação ou certificação, seria impraticável a forma como muitas das normas de software foram elaboradas; "*linhas gerais*", "*diretrizes básicas*" e termos desse gênero, tão freqüentes em padrões da área, não são passíveis de verificação. Portanto, de uma forma geral, não é possível determinar se uma norma foi ou não realmente aplicada, o que frustra a maioria dos benefícios advindos da normatização.

6.6.4 Qualidade de Software e os Modelos de Melhoria de Processo

A adoção do modelo CMM é um fenômeno crescente mundialmente. Apesar de inicialmente o modelo ter sido desenvolvido para o Departamento de Defesa Americano, gradualmente a metodologia foi ganhando visibilidade em outras

áreas de desenvolvimento. Organizações comerciais começaram a adotar suas diretrizes para estruturar programas de melhoria e, a partir de 1993, o número de organizações comerciais avaliadas pelo SEI superou o total de avaliações dirigidas a contratos com o Departamento de Defesa e o Governo americano juntas. Entre as organizações comerciais que utilizaram com sucesso o modelo estão a Motorola, Schlumberger, Bull e Siemens que, em muitos casos, após o sucesso na implantação, passaram a recomendá-lo para suas várias unidades espalhadas pelo mundo, o que contribui, sem dúvida para a popularidade do CMM. A Figura 6.4 apresenta a distribuição das empresas avaliadas em relação aos níveis de maturidade do modelo, até 1998.

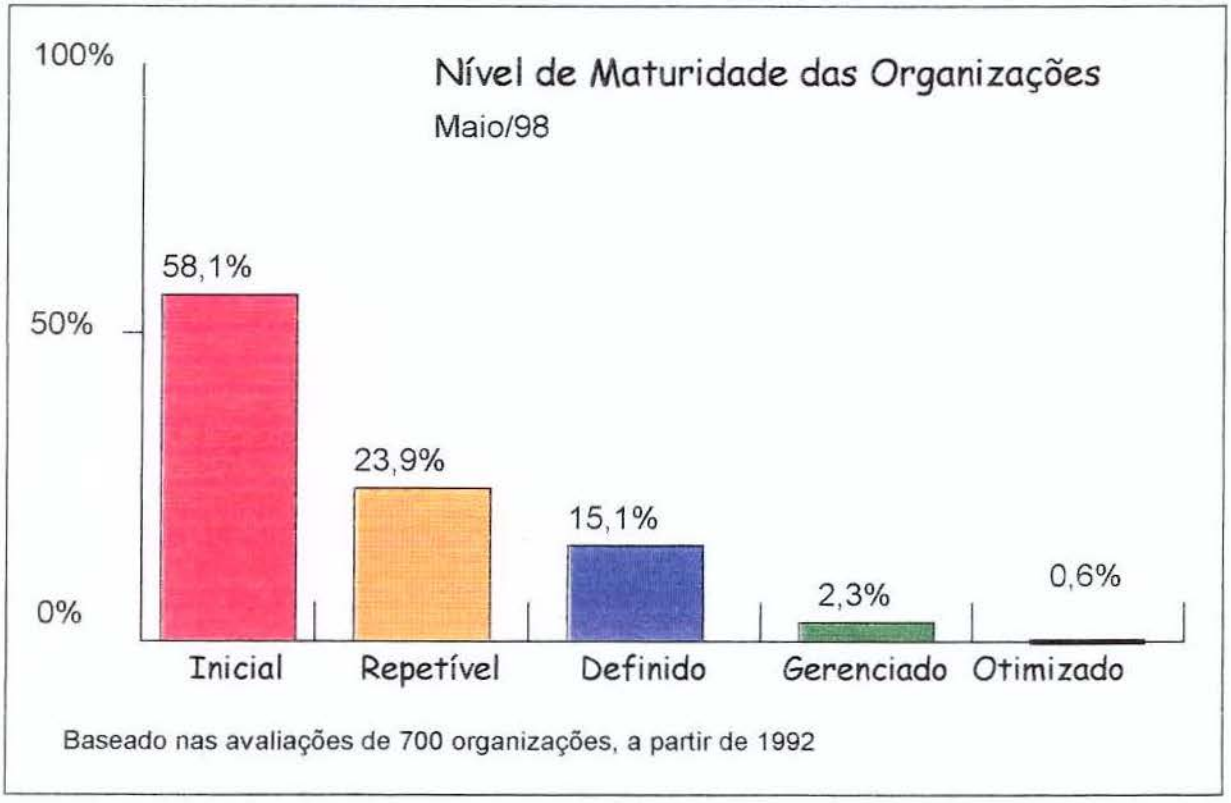


Figura 6.4 Nível de maturidade das organizações

(Fonte: Software Engineering Institute – Universidade Carnegie Mellon)

Metodologias relacionadas à melhoria de processos como o modelo CMM, SPICE e ISO 9001 identificam os processos imprescindíveis, em que ordem devem ser implementados e como verificar se estão realmente institucionalizados na

empresa, mas não especificam um determinado processo ou modelo a ser seguido. Isso implica que cada empresa deve implementar seus próprios métodos e procedimentos para produção e gerência, de acordo com sua realidade. Dois pontos são cruciais nesse contexto: assessoria adequada (na maioria das vezes há a necessidade de acompanhamento por especialistas externos) e o estabelecimento de um programa de medição.

Apesar de todos os modelos citados enfatizarem a necessidade de medição, os padrões atualmente disponíveis praticamente não oferecem suporte para especificação do que deve exatamente ser medido e como usar os resultados das medições na obtenção de software de alta qualidade.

A medição deve levar em consideração o contexto do software, pois um conjunto de métricas válidas para um projeto podem resultar em baixa qualidade ou custo de desenvolvimento elevado se aplicadas em outro projeto. Quando o objetivo é medir a qualidade, diversos fatores devem ser verificados: características do produto (como complexidade, por exemplo), nível de maturidade do processo adotado pela organização , ambiente de suporte ao desenvolvimento disponível (como metodologia de projeto usada e ferramentas de suporte) e a experiência e habilidade da equipe de desenvolvimento.

A amplitude das considerações necessárias para a medição da qualidade dificulta o registro e a utilização das informações, pois quanto mais características são consideradas, mais restrita será a possibilidade de comparações tanto entre projetos quanto entre produtos diferentes. É necessário armazenar as medições consistentemente de forma a permitir o aproveitamento de experiências. Preferivelmente, devem ser usadas métricas baseadas em características externas de qualidade (vide Capítulo 4). Um exemplo entre as metodologias de métricas é a GQM – Goal-Question-Metric, que possibilita a definição de um programa de medição a partir das metas de negócio estabelecidas, usadas como um guia para a escolha das métricas a serem utilizadas durante o desenvolvimento [IESE99].

Uma outra questão central na consideração dos programas de melhoria de qualidade é a relação entre a qualidade dos processos de software da organização e a qualidade de seus produtos. Com respeito a esse aspecto,

poucos dados foram encontrados. Em [HER97] são apresentados alguns resultados obtidos em um levantamento elaborado pelo Software Engineering Institute com o objetivo de avaliar a aplicação do modelo CMM. Os dados foram extraídos de estudos de caso de 48 organizações que conduziram programas de melhoria de processos baseados no CMM e que realizaram no mínimo duas avaliações de seus processos de software segundo o mesmo modelo. Entre os resultados apresentados apenas um está diretamente relacionado à qualidade final do produto de software: foi verificada a redução no número de defeitos pós-entrega, numa faixa variando entre 10% a 97% de redução/ano, com média 39%. Destacam-se também os seguintes dados de desempenho:

- ganho de produtividade/ano: 9% a 67% (média 35%);
- redução no tempo de desenvolvimento/ano: 15% a 23%;
- relação benefício/custo: 4:1 a 8:1 (média 5:1).

Alguns dados apresentados relacionam-se ao grau de dificuldade encontrado em aplicar os programas de melhoria e aos resultados obtidos em termos de ascensão na escala de maturidade do modelo CMM:

- 25% das organizações levaram 21 meses para subir do Nível 1 ao Nível 2; outros 25% levaram no mínimo 37 meses;
- 25% levaram no máximo 17 meses para subir do Nível 2 para o Nível 3, e 25% levaram 31 meses ou mais;
- 77% consideraram que o programa levou mais tempo do que o esperado;
- 68% consideraram que o programa custou mais do que o esperado;
- 70% das organizações consideraram no máximo moderado o nível de sucesso obtido na implantação dos itens recomendados pelo programa de melhoria;
- 66% afirmaram que apesar de entender *o que* deveria ser melhorado, tiveram dificuldades em determinar *como* implementar a melhoria.

Esses resultados indicam claramente que não há garantia de sucesso na implantação dos programas de melhoria. O tempo e custo geralmente excedem as expectativas e é necessária uma assessoria prática e clara para sua implantação [SAI95]. Porém, para as empresas que obtiverem sucesso, o retorno do investimento é bastante significativo.

Capítulo 7

Conclusão

Como proposta inicial este trabalho objetivava levantar os elementos do software relacionados aos desvios manifestados nos processos de desenvolvimento e discutir algumas das principais iniciativas na área de qualidade em software. A apresentação e análise das características intrínsecas de produtos de software – complexidade, conformidade, modificabilidade, intangibilidade, produção sob medida e não desgaste, e das peculiaridades da produção de software permite afirmar que as características citadas relacionam-se fortemente à grande parte das dificuldades verificadas.

A análise da aplicabilidade das iniciativas em qualidade de software selecionadas foi uma tarefa difícil, principalmente devido à quase inexistência, com exceção das referências citadas, de estudos enfocando os resultados da aplicação desses modelos. Além de raros, os dados encontrados são pouco conclusivos em relação à melhoria da qualidade do software como produto, uma questão essencial no contexto indicado.

7.1 Considerações Adicionais

Apesar das pesquisas focarem esforços na análise e levantamento de ferramentas e métodos mais adequados ao desenvolvimento de software, muito pouco do que se obteve é efetivamente utilizado pelas empresas. Devemos analisar com cuidado os motivos que sustentam essa situação. Numa análise rápida, poderíamos indicar a hipótese de que as empresas não utilizam as técnicas por essas não serem adequadas à realidade do trabalho de desenvolvimento. Certamente não podemos afastar essa hipótese, principalmente por constatarmos que a Engenharia de Software ainda não é madura o suficiente para suprir as necessidades da produção de software. Além disso, fala-se muito em melhoria de desempenho, mas pouco em melhoria do produto. São raros os dados que tentam esclarecer a relação entre programas de melhoria de processos

e a qualidade do produto final de software o que, de certa forma, desestimula as empresas a investirem em novos processos.

Outra hipótese seria a imaturidade das empresas, incapazes de perceber a necessidade de produzirem com maior qualidade. Essa hipótese também é plausível, haja vista tanto o cenários nacional quanto o internacional - 82% das empresas (na maioria norte-americanas) foram classificadas no nível 1 ou 2 do modelo CMM, quadro muito semelhante ao apontado pelos dados da pesquisa MCT. A causa parece ser múltipla e ir além desses fatores.

O que leva uma empresa a alterar seus processos é a possibilidade de lucro. Algumas empresas enxergam na melhoria a possibilidade de expandir sua participação no mercado. Outras mudam seus processos porque são ameaçadas pelos clientes de perder seus contratos caso não elevem seu nível de controle e qualidade. Seja qual for o motivo, o fato é que alguma manifestação de insatisfação prévia deve ocorrer para motivar a mudança.

Nas últimas décadas temos observado o rápido crescimento das aplicações de computador, e o software vem se tornando parte vital de produtos e serviços. Porém, apesar do avanço, o valor da qualidade do software para o mercado ainda não foi estabelecido. Podemos até mesmo questionar se nossos clientes têm a consciência de que poderiam ter à sua disposição produtos de melhor qualidade. De certa forma, a baixa qualidade do software é aceita pelo mercado como um fato normal.

Certa vez, num seminário sobre ES, um palestrante apresentou uma questão intrigante: *"Existem convocações (recalls) quando os carros têm defeitos, por que não fazem o mesmo para programas?"*. Quando é constatado um defeito de fabricação num automóvel, o carro é recolhido e consertado. Se descobrimos um defeito num sistema operacional, por exemplo, somos instruídos pelo próprio fabricante a comprar a nova versão do sistema ! A responsabilidade profissional dos desenvolvedores e a consciência da qualidade dos clientes ainda parecem estar em fase embrionária.

Uma organização pode permanecer em seu padrão atual de qualidade por longo período se os clientes não demandarem mudanças, não existirem concorrentes

para oferecer alternativas e os problemas não aumentarem em dificuldade. Parece que vivemos uma certa estagnação e, sem motivação externa, *poucas organizações melhoram suas práticas de desenvolvimento a não ser por pura motivação moral.* Essa estagnação seria completa, não fosse o fato de que os *problemas estão aumentando em dificuldade*, provocados pela elevação da complexidade das soluções a serem implementadas pelo software. Na verdade, os avanços na qualidade parecem ser motivados por fatores internos ao desenvolvimento e não pelo mercado.

7.2 Temas para trabalhos futuros

As informações apresentadas neste documento foram fruto do estudo de trabalhos de diversos autores reconhecidos no meio de software, porém não foi localizado nenhum estudo relacionado à confirmação das hipóteses formuladas, seja por estudo de casos ou experimentação.

Trabalhos futuros poderiam enfocar esses aspectos, não só verificando a pertinência das colocações apresentadas neste trabalho, mas também quantificando-as. A análise de registros históricos de projetos , assim como o acompanhamento de novos projetos em desenvolvimento, pode fornecer dados importantes sobre a ocorrência dos desvios e riscos analisados. Talvez a dificuldade maior seja a obtenção desses dados pois, pelo menos em nível nacional, as empresas não costumam registrar tais ocorrências, o que poderia explicar a não localização de trabalhos de pesquisa nessa área. Os dados coletados no projeto SPICE poderiam fornecer informações importantes; porém, até o encerramento deste trabalho esses dados tinham circulação restrita apenas às entidades participantes do ensaio e não haviam sido disponibilizados para utilização em pesquisa.

Estudos de caso sobre utilização do modelo CMM tanto para apoio a programas de melhoria de processos quanto para avaliação do nível de capacidade e maturidade de empresas também poderão fornecer informações valiosas para a verificação dos aspectos discutidos.

Referências Bibliográficas

[ABNT96] Associação Brasileira de Normas Técnicas, NRB 13596 "Avaliação do produto de software – Características de qualidade e diretrizes para o seu uso", Rio de Janeiro, ABNT, 1996.

[BAS94] V. Basili, S. Green, "Software Process Evolution at the SEL", IEEE Software, julho 1994, pp.58-66.

[BOE99] J. Bøegh, S. Depanfilis, B. Kitchenham, A. Pasquini, "A Method for Software Quality Planning, Control, and Evaluation", IEEE Software, maio/abril 1999, pp. 69-77.

[BOE91] B.W. Boehm, "Software Risk management: Principles and Practices", IEEE Software, janeiro 1991, pp.32-41.

[BRO87] F. P. Brooks, "No Silver Bullet", Computer, abril, 1987, pp10-19.

[COA94] F. Coallier, "How ISO 9001 Fits Into the Software World", IEEE Software, janeiro, 1994, pp. 98-100.

[COLL94] W. R. Collins, K.W. Miller, B.J.Spielman, P. Wherry, "How Good is Good Enough", Communications of the ACM, janeiro 1994, pp.81-91.

[COX90] B.J. Cox, "Planning the Software Industrial Revolution", IEEE Software, novembro 1990, pp.25-33.

[CRO79] P.B. Crosby, "Quality is Free", McGraw-Hill, 1979.

[DAS94] M. K. Daskalantonakis, "Achieving Higher SEI Levels", IEEE Software, julho 1994, pp.17-24.

[DAV88] A.M.Davis, E.H.Bersoff, E.R.Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models", IEEE Transactions on Software Enginnering, outubro 1988, pp.1453-1460.

[DeMA89] T. DeMarco, "Controle de Projetos de Software", Editora Campus - Série Yourdon Press, 1989.

[DEM90] W.E.Deming, "Qualidade: a Revolução da Administração", Editora Marques Saraiva , 1990

[FEN94] N. Fenton, S.L.Pfleeger, R.L.Glass, "Science and Substance: A Challenge to Software Engineers", IEEE Software, julho 1994, pp.86-95.

[GAR95] D. Garlan, R. Allen, J. Ockerbloom, "Architectural Mismatch: Why Reuse Is So Hard", IEEE Software, novembro 1995, pp.17-26.

[HAM96] T.K. Abdel-Hamid, "The Slippery Path to Productivity Improvement", IEEE Software, julho, 1996, pp. 43-52.

[HAA94] V. Haase, R. Messnarz, G. Koch, H.J. Kugler, P. Decrinis,"Bootstrap: Fine-Tuning Process Assessment", IEEE Software, julho 1994, pp-25-35.

[HER97] J. Herbsleb, D. Zubrow, D. Goldenson, W. Hayes, M. C. Paulk, "Software Quality and the Capability Maturity Model", Communications of the ACM, junho 1997, pp. 30-40.

[IESE99] Fraunhofer Institute for Experimental Software Engineering, site oficial do projeto GQM – Goal-Question-Metric:
http://www.iese.fhg.de/Services/Projects/Public-Projects/Perfect/GQM/GQM.html

[JURA-91a] J. M. Juran. et al. "Controle de Qualidade – Conceitos, Políticas e Filosofia da Qualidade", McGraw-Hill, 1991.

[JURA-91b] J. M. Juran. et al. "Controle de Qualidade – Ciclo dos Produtos: Do Projeto à Produção", McGraw-Hill, 1991

[KIT96] B. Kitchenham, S. L. Pfleeger, "Software Quality: The Elusive Target", IEEE Software, janeiro 1996, pp.12-21.

[MAC95] K. Mackey, "Why Bad Things Happen to Good Projects", IEEE Software, maio 1996, pp.27-32.

[MART] S. Martins, "Sistemas da qualidade em pequenas empresas", Controle da Qualidade, agosto 1995, pp 66-80

[MCT95] Ministério da Ciência e Tecnologia, Secretaria de Política de Informática e Automação, "Qualidade no Setor de Software Brasileiro - 1995", Brasília, 1995.

[MCT97] Ministério da Ciência e Tecnologia, Secretaria de Política de Informática e Automação, "Qualidade no Setor de Software Brasileiro - 1997", Brasília, 1997.

[PAU95] M.C.Paulk, "How ISO 9001 Compares with the CMM", IEEE Software, janeiro 1995, pp. 74-83.

[PER94] D.E. Perry, N.A.Staudenmayer, L.G.Votta, "People, Organizations and Process Improvemet", IEEE Software, julho 1994, pp.36-45.

[PRE95] R. Pressman, "Engenharia de Software", Makron Books - McGraw Hill, 3a. edição, 1995.

[RAM96] C.V. Ramamoorthy, W. Tsai, "Advances in Software Enginnering", Computer, outubro 1996, pp.47-58.

[SAI95] H. Saiedian, R. Kuzara, "SEI Capability Model's Impact on Contractors", Computer, janeiro 1995, pp.16-26.

[SCH96] N.F. Schneidewind, N. Fenton, "Do Standards Improve Quality ?", IEEE Software, julho 1996, pp.22-24.

[SEI99] "Capability Maturity Model for Software", Software Engineering Institute, site em http://www.sei.cmu.edu.

[SEN94] J. Senders, E. Curran, "Software Quality: A Framework for Success in Software Development", Addison-Wesley, 1994.

[SHA90] M. Shaw, "Prospects for an Engineering Discipline of Software", IEEE Software, novembro 1990, pp.15-24

[SPICE99] Software Process Improvement and Capability Determination, site oficial em http://www.sqi.gu.edu.au/spice

[WAS96] A. I. Wasserman, "Toward a Discipline of Software Engineering", IEEE Software, novembro 1996, pp. 23-31

[WEIN93] G.M. Weinberg, "Software com Qualidade - Pensando e Idealizando Sistemas", Makron Books do Brasil Editora Ltda, 1993.

[WES90] J. Wesselius, F. Ververs, "Some Elementary Questions on Software Quality Control", Software Emginnering Journal, novembro 1990, pp.319-330.

ANEXO A

Estão apresentados neste Anexo os dados referentes à pesquisa "Qualidade no Setor de Software Brasileiro", da Secretaria de Política de Informática e Automação do Ministério da Ciência e Tecnologia – SEPIN/MCT. Foi incluída apenas a parcela dos dados utilizados na elaboração do trabalho de pesquisa, referentes aos levantamentos realizados em 1997 e 1995. Para maiores detalhes veja [MCT97], [MCT95] ou consulte o site MCT/SEPIN em http://www.mct.gov.br/sepin .

"Qualidade no Setor de Software Brasileiro"
1997

Introdução

Nos meses de junho e julho de 1997, no âmbito do Subcomitê Setorial da Qualidade e Produtividade em Software do Programa Brasileiro da Qualidade e Produtividade - SSQP/SW-PBQP, foi realizada pesquisa direta junto a 589 empresas desenvolvedoras de software no País, sob a responsabilidade da Secretaria de Política de Informática e Automação do Ministério da Ciência e Tecnologia - DSI/SEPIN/MCT, através da Divisão de Sistemas de Informação sobre Informática da Coordenação-Geral de Software, Serviços e Aplicações da Informática - DSI/CGSA.

Trata-se da 3ª edição da pesquisa "Qualidade no Setor de Software Brasileiro", com periodicidade bienal proposta e mantida, que coletou dados para o ano-base de 1993 de 282 empresas em sua versão original, passando a 445 empresas em 1995.

Cabe lembrar que aspectos da qualidade foram abordados, originalmente, na pesquisa "Panorama do Setor de Informática - 1990" junto aos segmentos de hardware, software e serviços técnicos de informática.

Esta pesquisa não tem como objetivo avaliar a qualidade do produto de software brasileiro, mas sim avaliar a gestão da qualidade nas empresas, permitindo acompanhar a evolução do setor quanto a este aspecto, além de direcionar as ações do SSQP/SW-PBQP.

A elaboração de planos estratégicos, a inclusão de metas para a qualidade nesses planos, a coleta de indicadores da qualidade, a contabilidade de custos da qualidade, a implantação de programas da qualidade total e a certificação dos sistemas da qualidade são algumas das questões disponíveis para avaliar a gestão da qualidade nas empresas de software brasileiras.

O relacionamento das empresas com seus empregados é acompanhado a partir de aspectos da participação dos mesmos na solução de problemas, sua satisfação e oportunidades de aperfeiçoamento profissional através de treinamentos oferecidos, enquanto o relacionamento com o mercado é avaliado pela realização de pesquisas de expectativas e satisfação dos clientes, da existência de estruturas de atendimento e resolução de reclamações mantidas e do uso desses tipos de dados na revisão de projetos ou na especificação de novos produtos e serviços.

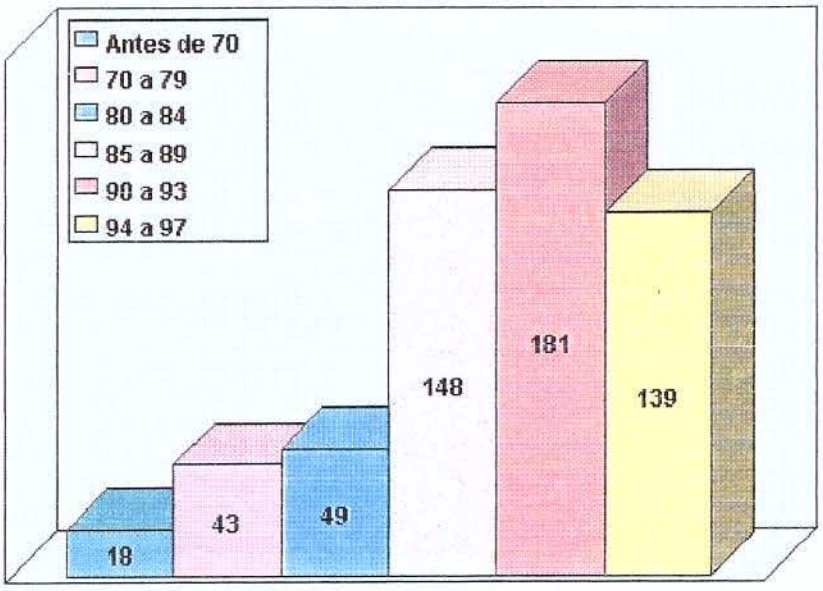
Os procedimentos específicos para qualidade em software são acompanhados por indicadores da adoção de métodos de engenharia de software para prevenção ou detecção de defeitos, da utilização de ferramentas automatizadas de desenvolvimento e do tipo de documentação adotada.

Adicionalmente, todo um conjunto de aspectos são levantados visando a caracterização das empresas e do software desenvolvido no Brasil.

Atividades das empresas no tratamento de software

Categorias	Nº de empresas	%
Desenvolvimento		
Software pacote	457	77,6
Software sob encomenda	419	71,1
Software embarcado	57	9,7
Software para uso próprio	219	37,2
Distribuição ou editoração	140	23,8
Software de terceiros		
Base	**589**	**100**

Distribuição das empresas, segundo ano de início das atividades de informática

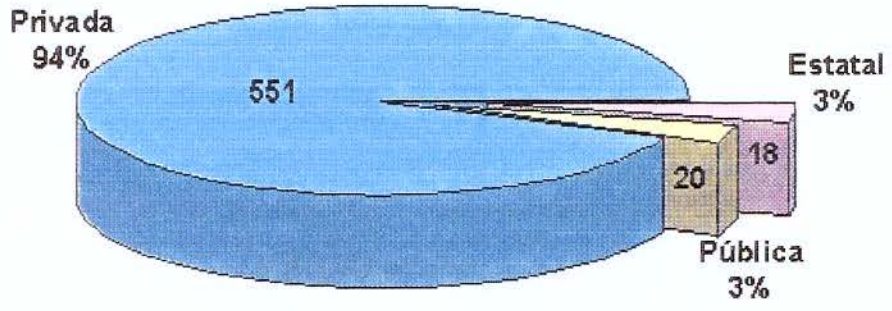


Nota: Na apuração do número de empresas com início de atividades em 1997, foram considerados apenas os meses de janeiro a julho, quando se encerrou o trabalho de campo da pesquisa; 11 empresas não responderam à pergunta.

Atividades de informática características das empresas

Categorias	Nº de empresas	%
Desenvolvimento de software pacote	434	73,7
Desenvolvimento de software sob encomenda	399	67,7
Consultoria e projetos em informática	335	56,9
Treinamento em informática	174	29,5
Distribuição e revenda de informática	158	26,8
Manutenção e assistência técnica	97	16,5
Serviços de processamento de dados	90	15,3
Serviços de automação comercial	89	15,1
Indústria de inf., telecom. ou automação	60	10,2
Serviços de automação industrial	58	9,8
Comerc. de dados ou bases de dados	49	8,3
Serviços de entrada de dados	48	8,1
Serviços de automação bancária	31	5,3
Exclusivamente uso próprio	12	2,0
Provedor Internet	9	1,5
Outras	35	5,9
Base	**589**	**100**

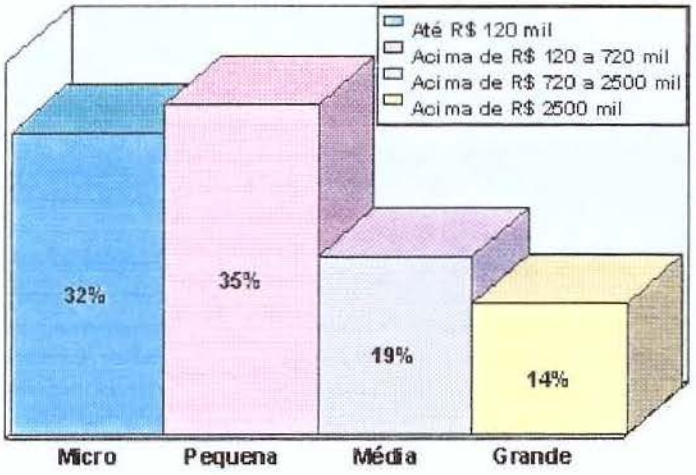
Tipos de empresas



Porte das empresas sengundo comercialização bruta anual proveniente de software

Portes	Faixas de Comercialização	Nº	%	%
Apropria receita especifica		431	74,2	100,0
Microempresas	Até R$ 120 mil	139	23,9	32,3
	Acima de R$ 120 mil a R$ 720 mil	152	26,2	35,3
Pequenas	Acima de R$ 720 mil a R$ 2,5 milhões	81	13,9	18,8
	Acima de R$ 2,5 milhões	59	10,2	13,7
Médias				
Grandes				
Não apropria receita específica		**78**	**13,4**	
Não comercializa software		**72**	**12,4**	
Base		**581**	**100**	**431**

Porte das empresas, segundo comercialização bruta anual de software - 1996



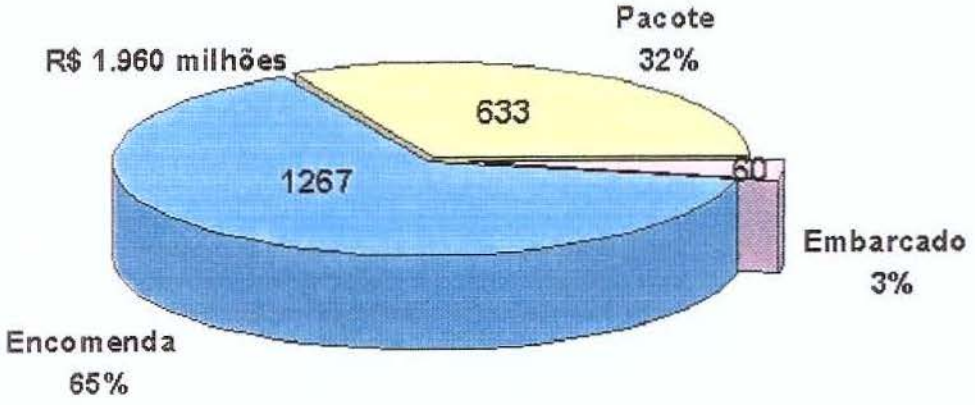
Nota: Questão respondida por 431 empresas que comercializaram software.

Comercialização bruta anual proveniente de software - 1996

em R$ milhões

Categorias	Total	Mercado interno	Mercado externo
Total de software	1.964,7	1.960,2	4,5
Software pacote	636,1	633,4	2,7
Software sob encomenda	1.268,8	1.267,2	1,6
Software embarcado	59,8	59,6	0,2
Base	**431**	**429**	**22**

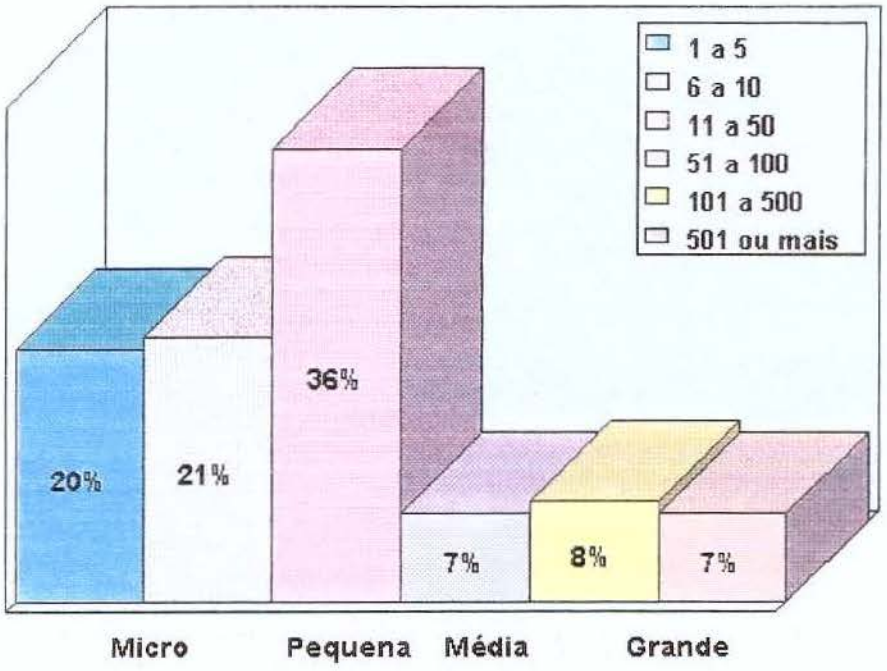
Distribuição da comercialização bruta anual proveniente de software no mercado interno – 1996



Porte das empresas, segundo número de pessoas - dez/1996

Categorias	Faixas	Nº de empresas	%
Microempresas	de 1 a 5 pessoas	120	20,4
	de 6 a 10	125	21,3
Pequenas	de 11 a 50	209	35,6
Médias	de 51 a 100	44	7,5
Grandes	mais de 100	89	15,2
Base		**587**	**100**

Porte das empresas, segundo numero de pessoas – dez/1996

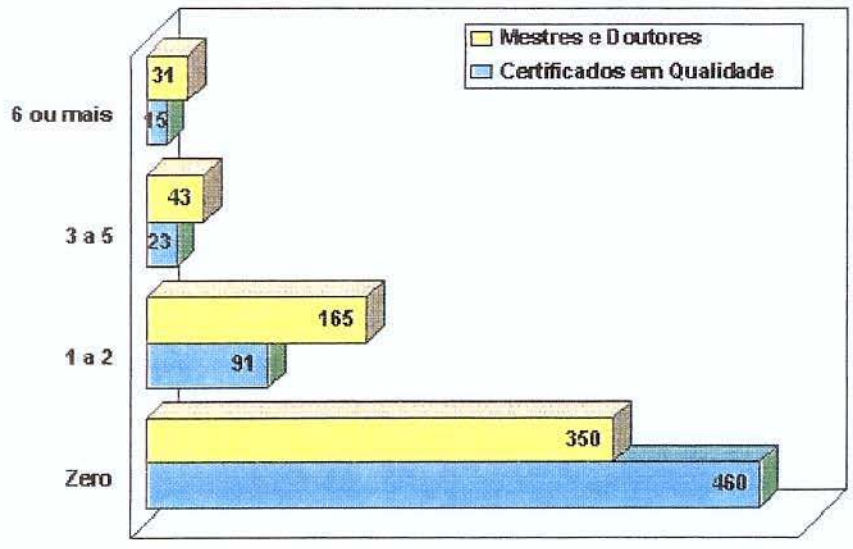


Produtividade de software por empregado – 1996

Porte, segundo número de pessoas	Categorias	Porte, segundo comerc. bruta anual de software			
		Micro	Pequenas	Médias	Grandes
		Até R$120 mil	Acima de R$120 a 720	Acima de R$720 a 2500	Mais de R$2500
Micro (de 1 a 10)	Nº empresas	117	63	3	1
	Produtividade	10,0	33,8	127,6	x
Pequenas (de 11 a 50)	Nº empresas	19	84	54	13
	Produtividade	3,9	19,2	47,4	190,5
Médias (de 51 a 100)	Nº empresas	2	2	12	16
	Produtividade	x	x	31,3	68,7
Grandes (mais de 100)	Nº empresas	1	3	12	29
	Produtividade	x	x	x	45,5

Nota: A produtividade de software por empregado refere-se ao valor bruto proveniente da comercialização de software nos mercados interno e externo sobre o número de pessoas nas empresas, medido em R$ mil.

Distribuição das empresas, segundo qualificação dos profissionais – dez/1996



Distribuição das empresas, segundo o número de profissionais certificados em qualidade - dez/1996

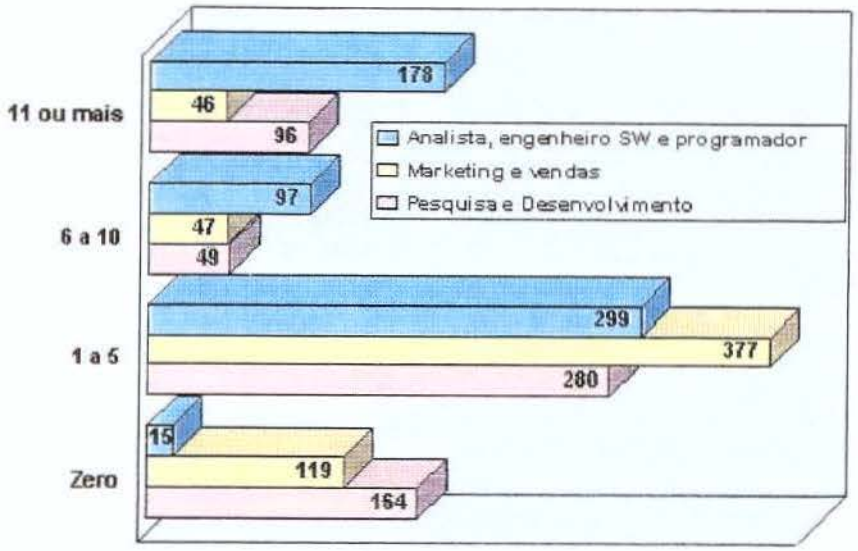
Faixas de Profissionais	Nº de empresas	%
Zero 1 a 2	460	78,1
3 a 5	91	15,4
6 a 10	23	3,9
11 a 20	11	1,9
21 a 50	3	0,5
51 ou mais	1	0,2
Base	**589**	**100**

Nota: Foram considerados como profissionais certificados em qualidade aqueles com certificação ASQ, Lead Assessor ou pós-graduação *lato sensu e stricto sensu* em gestão da qualidade.

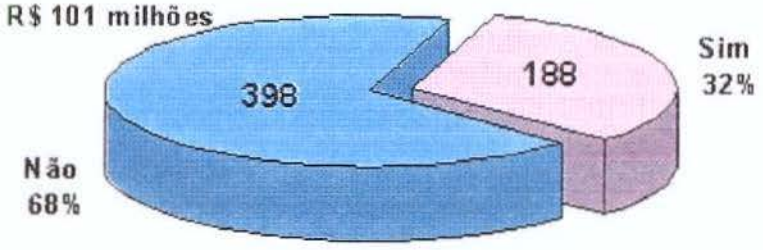
Distribuição das empresas, segundo o número de profissionais envolvidos em atividades de P&D - dez/1996

Faixas de Profissionais	Nº de empresas	%
Zero	164	27,8
1 a 2	155	26,3
3 a 5	125	21,2
6 a 10	49	8,3
11 a 20	46	7,8
21 a 50	24	4,1
51 ou mais	26	4,4
Base	**589**	**100**

Distribuição das empresas, segundo tipo de atividade desenvolvida pelos profissionais – dez/1996



Terceirização de serviços de análise e programação - 1996



Nota: 3 empresas não responderam à pergunta e 167 informaram valor desses serviços.

Elaboração de planos estratégicos, planos de negócios ou planos de metas

Categorias	Nº de empresas	%
Atualização sistemática	159	27,0
Revisão sem periodicidade fixa	183	31,1
Em implantação	150	25,5
Não elabora	97	16,5
Base	**589**	**100**

Inclusão de metas ou diretrizes para qualidade nos planos

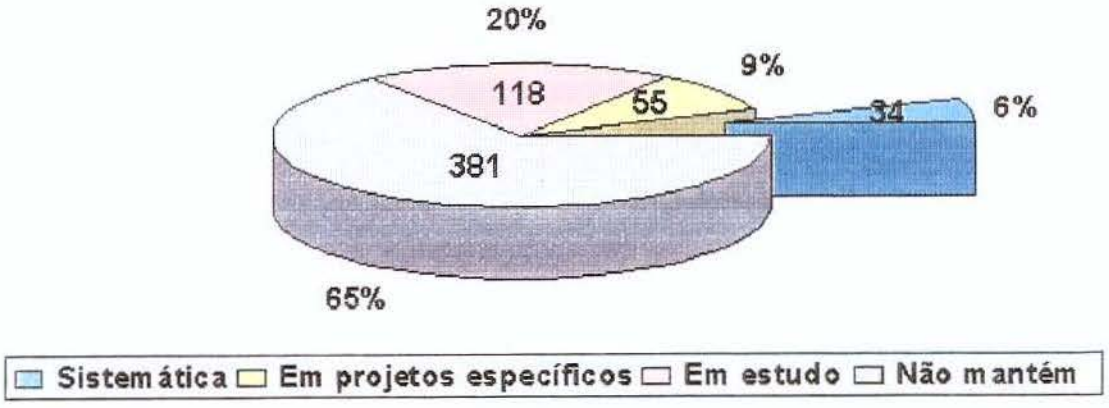
Categorias	Nº de empresas	%
Sistemática	199	40,4
Eventual	141	28,7
Pretende incluir	131	26,6
Não inclui	21	4,3
Base	**492**	**100**

Nota: Questão respondida apenas pelas empresas que elaboram ou estão implantando plano estratégico, plano de negócios ou plano de metas.

Coleta de Indicadores da qualidade de produtos e serviços

Categorias	Nº de empresas	%
Sistemática	174	29,5
Quando necessário	192	32,6
Em estudo ou implantação	141	23,9
Não coleta	82	13,9
Base	589	100

Contabilidade dos custos da qualidade

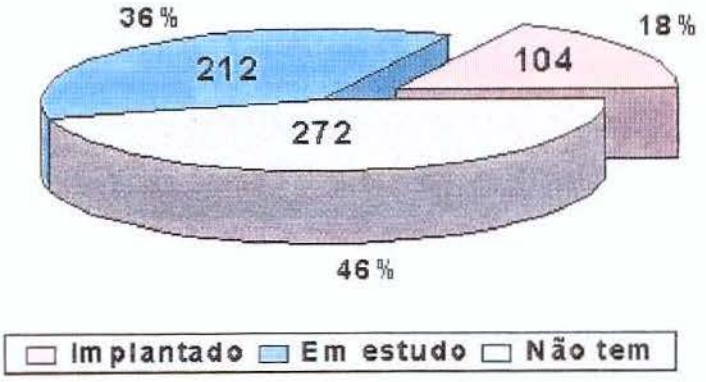


Certificação do sistema da qualidade

Empresas Certificadas	1997
Todos os setores	1.78
Setor de Informática	8
Pesquisa da Qualidade em Software	129
Certificação ISO 9001	45
	36
Certificação ISO 9002	11
	16
Software explicitado no escopo do certificado	

Nota: A fonte para empresas certificadas de todos os setores foi a ABNT/CB-25 (Associação Brasileira de Normas Técnicas / Comissão Brasilieira de qualidade).

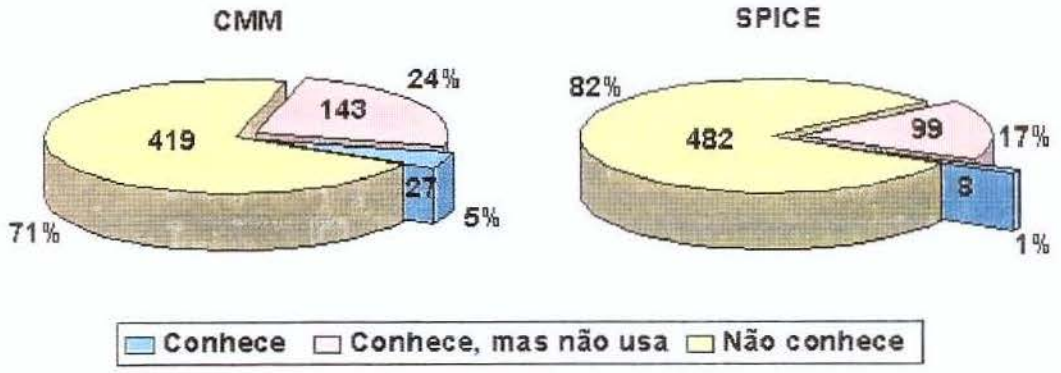
Programa da Qualidade Total ou similar



Conhecimento dos modelos CMM e SPICE

Categorias	CMM		SPICE	
	Nº	%	Nº	%
Conhece e usa	7	1,2	1	0,2
Conhece e começa a usar	20	3,4	7	1,2
Conhece, mas não usa	143	24,3	99	16,8
Não conhece	419	71,1	482	81,8
Base	**589**	**100**	**589**	**100**

Conhecimento dos Modelos



Principais métodos utilizados para prevenção de defeitos

Categorias	Nº de empresas	%
Controles de versão	327	55,5
Prototipação	259	44,0
Gerência de projetos	235	39,9
Métodos orientados a objetos	216	36,7
Métodos estruturados	210	35,7
Análise de requisitos	209	35,5
Projetos de interface com o usuário	207	35,1
Análise crítica conjunta	194	32,9
Base	**589**	**100**

Outros métodos utilizados para prevenção de defeitos

Categorias	Nº de empresas	%
Reuso	110	18,7
Engenharia da informação	104	17,7
Medições da qualidade (Métricas)	48	8,1
Joint Application Design - JAD	46	7,8
Gestão de configuração	40	6,8
Desenvolvimento em sala-limpa (clean room)	36	6,1
Gestão de mudança	32	5,4
Estimação da confiabilidade	32	5,4
Quality Function Deployment - QFD	11	1,9
Outros	19	3,2
Base	**589**	**100**

Métodos utilizados para detecção de defeitos

Categorias	Nº de empresas	%
Testes de sistema	392	66,6
Testes de campo	357	60,6
Testes funcionais	329	55,9
Testes de usabilidade	327	55,5
Testes de aceitação	278	47,2
Validação	250	42,4
Testes de unidade	137	23,3
Revisões estruturadas	113	19,2
Auditorias	102	17,3
Inspeções formais	100	17,0
Testes estruturais	97	16,5
Verificação independente	81	13,8
Outros	16	2,7
Base	**589**	**100**

Principais ferramentas utilizadas

Categorias	Nº de empresas	%
Dicionário de dados	222	37,7
Gerador de relatórios	218	37,0
Gerador de telas	207	35,1
Depurador interativo	167	28,4
Gerador de código-fonte	162	27,5
Gerenciador de projetos	143	24,3
CASE	119	20,2
CASE Lower	81	13,8
CASE Upper	79	13,4
Gerenciador de bibliotecas de módulos	116	19,7
Gerador de gráficos	115	19,5
Base	**589**	**100**

Outras ferramentas utilizadas

Categorias	Nº de empresas	%
Documentador	104	17,7
Prototipador	90	15,3
Distribuição de software	86	14,6
Gerador de entrada de dados	63	10,7
Gerenciador de configuração	57	9,7
Gerenciador de documentos	54	9,2
Driver de teste	52	8,8
Analisador de código	50	8,5
Gerador de dados de teste	50	8,5
Otimizador	38	6,5
Outras	26	4,4
Não utiliza ferramentas automatizadas	121	20,5
Base	**589**	**100**

Principais documentos adotados

Categorias	Nº de empresas	%
Manual do usuário	437	76,0
Help on-line	370	64,3
Contratos e acordos	361	62,8
Guia de instalação	318	55,3
Documentação de programas	314	54,6
Documentação no código	312	54,3
Especificação de sistema	294	51,1
Base	**575**	**100**

Outros documentos adotados

Categorias	Nº de empresas	%
Documentação comercial	249	43,3
Documentação de descrição do produto	242	42,1
Projeto de sistema	223	38,8
Manual de treinamento	210	36,5
Documentação de marketing	194	33,7
Documentação do processo de software	142	24,7
Resultado de revisões e testes	126	21,9
Plano de testes	125	21,7
Plano de controle da qualidade	39	6,8
Outras	12	2,1
Não adota documentação	17	3,0
Base	**575**	**100**

"Qualidade no Setor de Software Brasileiro"
1995

Introdução

A pesquisa Qualidade no Setor de Software Brasileiro foi realizada, em sua segunda edição, sob a coordenação da Divisão de Sistemas de Informação sobre Informática da Secretaria de Política de Informática e Automação do Ministério da Ciência e Tecnologia - DSI/SEPIN/MCT, no âmbito do Subprograma Setorial da Qualidade e Produtividade em Software, do Programa Brasileiro da Qualidade e Produtividade - SSQP/SW-PBQP.

Esta pesquisa provê dados para o Sistema Brasileiro de Informação sobre Software - SBIS, concebido pelo Programa Brasileiro de Software para Exportação - SOFTEX 2000 e integra o projeto Competitividade da Indústria Brasileira de Software, apresentado ao SSQP/SW-PBQP, que objetiva levantar informações sobre aspectos relevantes para a qualidade da produção de software no País.

Aspectos da qualidade para os segmentos de hardware, software e serviços técnicos de informática foram abordados, originalmente, na pesquisa Panorama do Setor de Informática em 1990.

Em 1993, pesquisa específica foi realizada junto aos produtores de software brasileiros e seus resultados utilizados na elaboração do Termo de Referência do Subprograma, publicado em 1994.

Em setembro de 1995, foram iniciados os trabalhos para definição da pesquisa Qualidade no Setor de Software Brasileiro - 1995, cujo conteúdo foi submetido a membros do SSQP/SW para críticas e sugestões.

A comparação entre resultados das duas versões da pesquisa permite avaliar a evolução do setor, identificando pontos fortes e áreas indicadas para melhorias nas atividades das empresas, decisivas para a obtenção de padrões internacionalmente aceitos de qualidade e produtividade, buscando assim direcionar as ações do SSQP/SW.

Participaram deste esforço o Programa SOFTEX 2000, por intermédio do Grupo de Pesquisa e Informações da Coordenação Nacional, e as entidades Associação Brasileira de Empresas Estaduais de Processamento de Dados - ABEP, Associação Brasileira da Indústria Elétrica e Eletrônica - ABINEE, Associação das Empresas Brasileiras de Software e Serviços de Informática - ASSESPRO e Sociedade dos Usuários de Informática e Telecomunicações - SUCESU através da distribuição e recebimento dos formulários de coleta de seus associados, que transcorreu nos meses de novembro e dezembro de 1995.

Foram processados os dados de 445 empresas com atividades características de desenvolvimento de software - pacotes para comercialização, encomendas para terceiros, softwares embarcados ou para uso próprio; distribuidoras e editoras de softwares de terceiros.

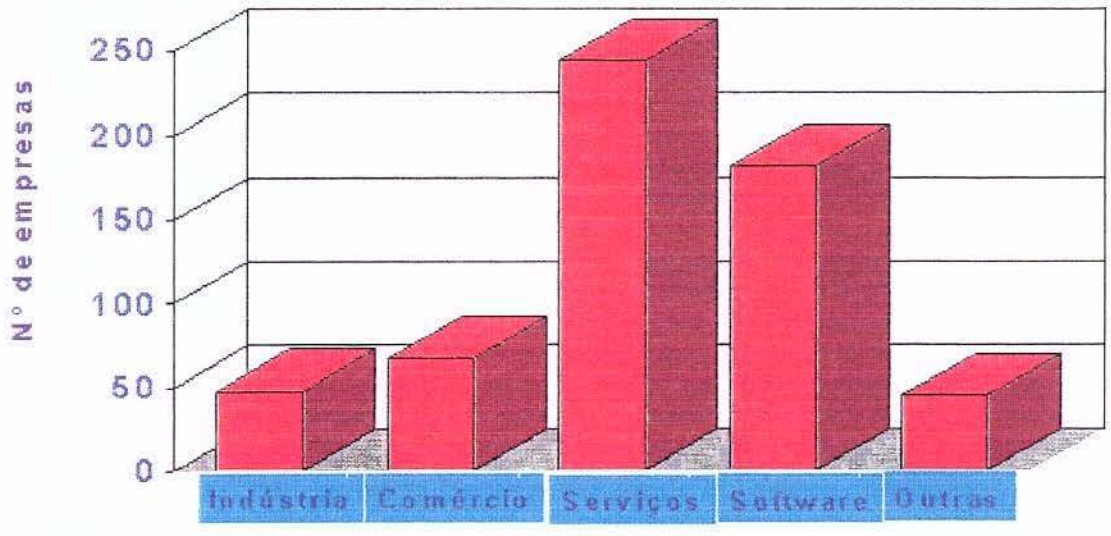
O Centro de Tecnologia de Software - TECSOFT, núcleo regional de Brasília do Programa SOFTEX, providenciou serviço de digitação, realizado através de profissional da Secretaria de Fazenda e Planejamento do Governo do Distrito Federal.

Para crítica, digitação, consistência e processamento dos dados levantados, a Divisão de Sistemas de Informação sobre Informática - DSI/SEPIN desenvolveu dois sistemas que permitem, ainda, a consulta e emissão de relatórios específicos.
Um dos sistemas contempla a parte cadastral de identificação das empresas e dos responsáveis pelo preenchimento. O outro, utilizando-se do Sistema de Acompanhamento de Pesquisas de Opinião - SELAP, cedido pela Divisão de Desempenho Mercadológico da Diretoria de Coordenação de Operações e Serviços da TELEBRÁS, trata das questões da pesquisa propriamente dita, que, no instrumento de coleta, foram organizadas nos blocos caracterização da empresa, caracterização dos softwares, gestão da qualidade, gestão de recursos humanos, atendimento a clientes e procedimentos para a qualidade em software.
A disponibilização dos dados cadastrais das empresas participantes e os resultados da pesquisa consolidados está sendo feita através de meio magnético, da distribuição de relatórios impressos e, via Internet, através da home-page da SEPIN.
Os dados de pesquisa individualizados não serão fornecidos, assegurando o compromisso de sigilo assumido quanto às informações prestadas pelas empresas.
Texto relativo ao Diagnóstico da Qualidade e Produtividade em Software, elaborado a partir de resultados desta pesquisa, já foi atualizado e alguns dos indicadores definidos pelo SSQP/SW-PBQP, com suas metas estabelecidas para os anos de 1996 e 2000, foram medidos através também desta pesquisa para publicação junto ao Termo de Referência desse Subprograma.

Atividades características das empresas
no tratamento de software

Categorias	Nº de empresas	%
Desenvolvimento para comercialização (pacote)	300	67,4
Desenvolvimento sob encomenda	268	60,2
Desenvolvimento para uso próprio	166	37,3
Distribuição ou edição de software	107	24,0
Software embarcado	29	6,5

Gráfico 01 - Atividades principais das empresas



Nota: Questão de múltipla escolha para cada empresa.

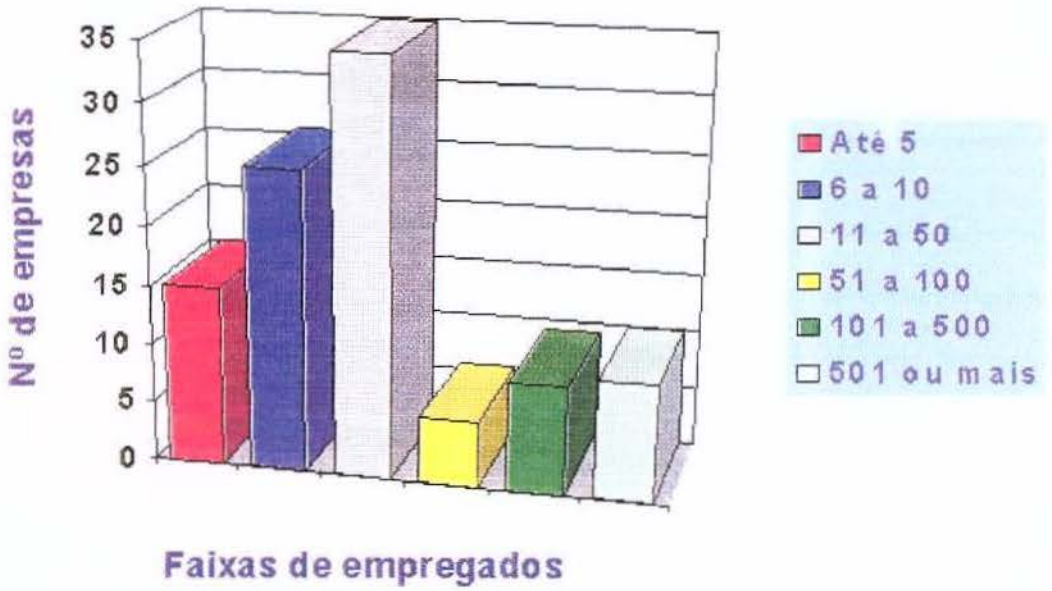
Atividades de informática características das empresas

Categorias	Nº de empresas	%
Desenvolvimento de software produto	315	70,8
Consultoria e projetos	244	54,8
Projetos de software sob encomenda	237	53,3
Treinamento em informática	142	31,9
Distribuição e revenda	129	29,0
Manutenção e assistência técnica	82	18,4
Serviços de automação comercial	80	18,0
Serviços de automação industrial	56	12,6
Processamento e entrada de dados	50	11,2
Indústria de inform., Telecom. ou automação	48	10,8
Serviços de automação bancária	28	6,3
Comerc. de dados/ bases de dados	28	6,3
Outras	28	6,3

Porte das empresas segundo número de pessoas

Categorias	Faixas	1995	
		Nº	%
Microempresas	até 10 pessoas	180	40,5
Pequeno porte	de 11 a 50	154	34,7
Médio porte	de 51 a 100	25	5,6
Grande porte	mais de 100	85	19,1

Gráfico 02 - Pessoas empregadas nas empresas



Número de profissionais certificados em qualidade

Faixas de Profissionais	Nº de empresas	%
Zero	322	75,8
1 a 2	72	16,9
3 a 5	19	4,5
6 a 10	4	0,9
11 a 20	3	0,7
21 ou mais	5	1,2

Gráfico 03 - Mestres, doutores e profissionais certificados em qualidade

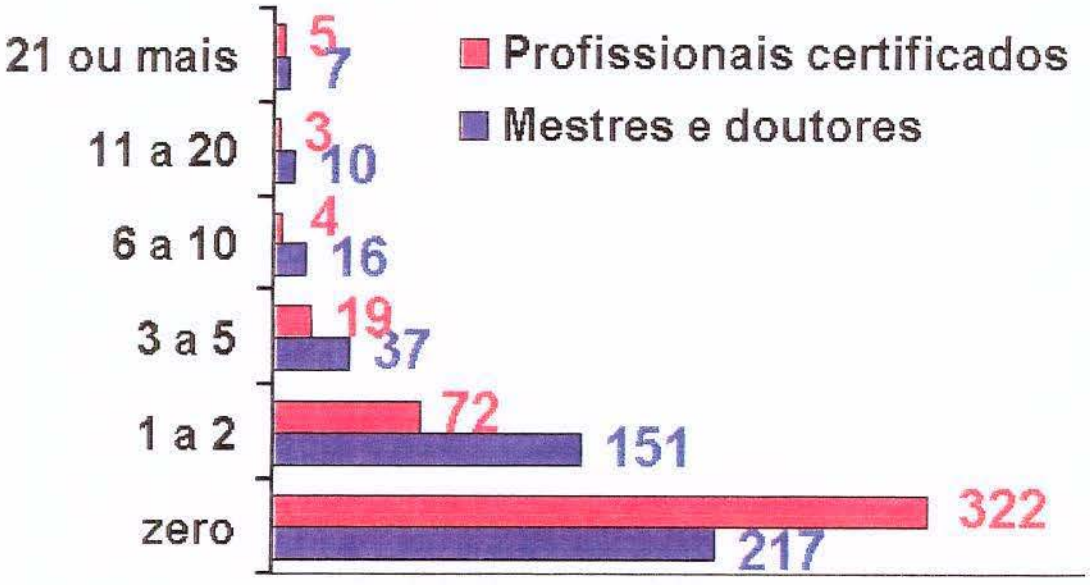
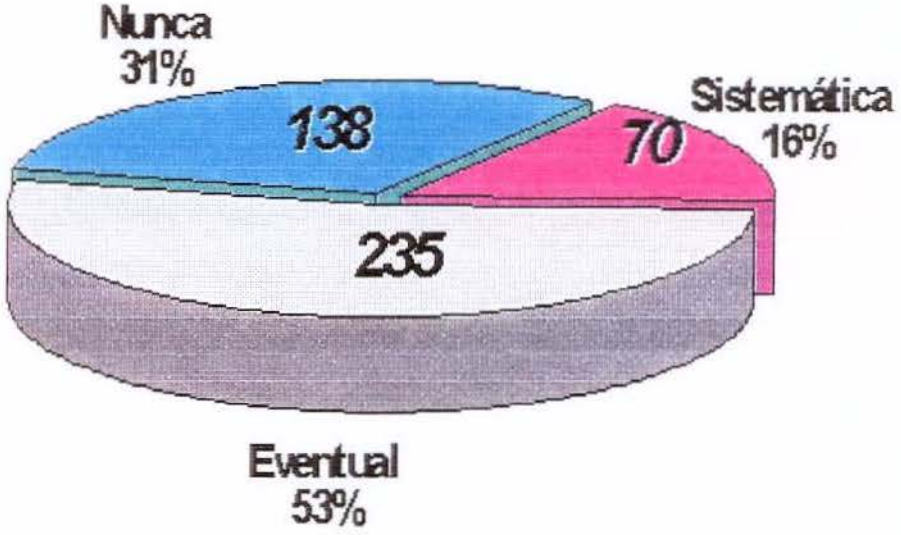


Gráfico 04 - Analistas de sistemas, engenheiros de software, programadores e profissionais de marketing e vendas

Gráfico 05 - Terceirização de serviços de análise e programação



Nota: Duas empresas não responderam à pergunta.

Comercialização bruta proveniente de software

Faixas de Comercialização	Nº de empresas	%
Não comercializa software	51	11,9
Até R$ 100 mil	156	36,4
Acima de R$ 100 mil a R$ 500 mil	110	25,6
Acima de R$ 500 mil a R$ 1 millhão	31	7,2
Acima de R$ 1 milhão a R$ 5 milhões	42	9,8
Acima de R$ 5 milhões	19	4,4
Não apropria receita específica	20	4,7

Elaboração de planos estratégicos ou planos de metas

Categorias	Nº de empresas	%
Elabora e atualiza sistematicamente	96	21,7
Elabora e revisa sem periodicidade fixa	151	34,2
Em implantação	109	24,7
Não elabora	86	19,5

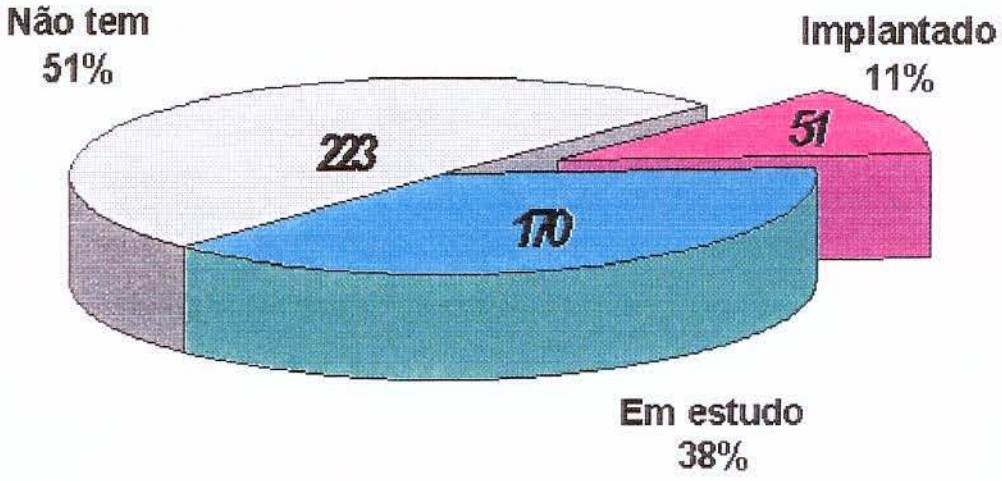
Inclusão de metas ou diretrizes para a qualidade nos planos

Categorias	Nº de empresas	%
Inclui sistematicamente	138	38,9
Inclui eventualmente	104	29,3
Pretende incluir	102	28,7
Não inclui	11	3,1

Coleta de indicadores da qualidade de produtos e serviços

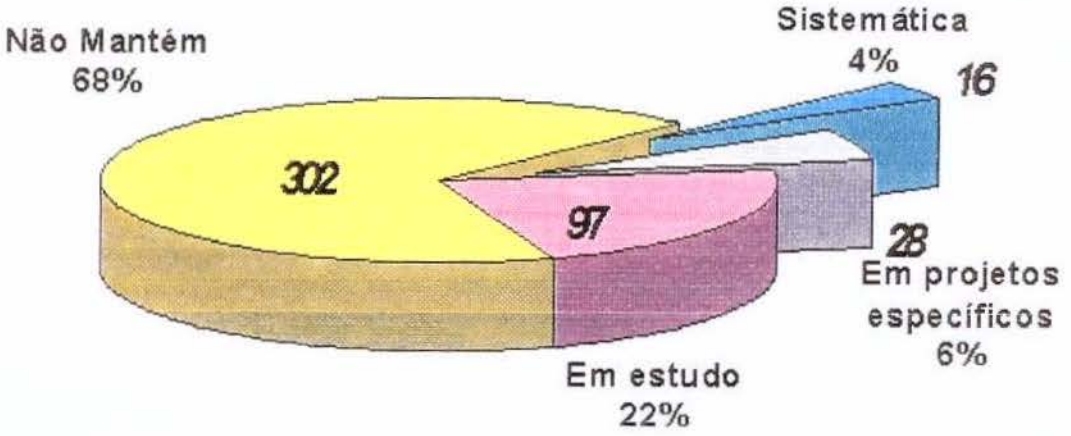
Categorias	Nº de empresas	%
Sistemática	111	25,1
Quando necessário	186	42,1
Em estudo	96	21,7
Não coleta	49	11,1

Gráfico 09 - Programa da qualidade total ou similar



Nota: Uma empresa não respondeu à pergunta.

Gráfico10 - Contabilidade de custos da qualidade e da não-qualidade



Nota: Duas empresas não responderam à pergunta.

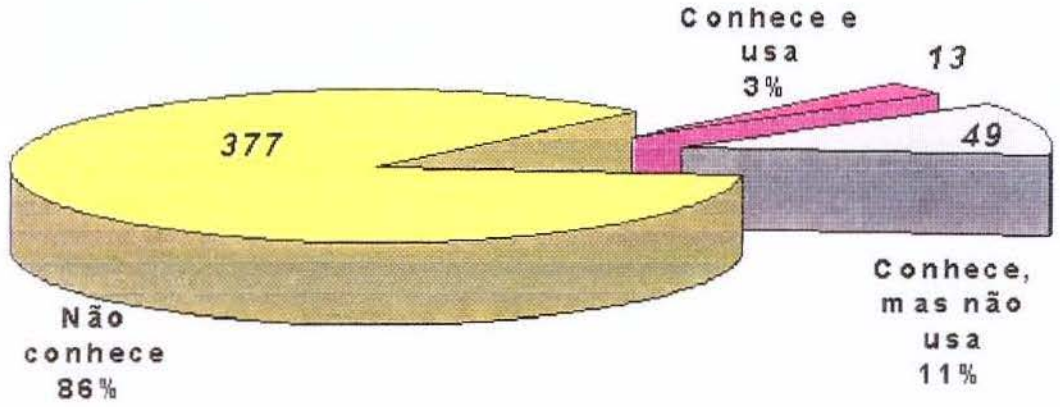
Certificação do sistema da qualidade

Categorias	Nº de empresas	%
ISO 9001	8	1,8
ISO 9002	1	0,2
Certificado por cliente	14	3,2

Número de profissionais certificados em qualidade

Faixas de Profissionais	Nº de empresas	%
Zero	322	75,8
1 a 2	72	16,9
3 a 5	19	4,5
6 a 10	4	0,9
11 a 20	3	0,7
21 ou mais	5	1,2

Gráfico 11 - Conhecimento do modelo CMM (Capability Maturity Model)



Nota: Seis empresas não responderam à pergunta.

Equipes dedicadas à garantia da qualidade

Categorias	Nº de empresas	%
Há equipe específica	66	14,9
Há equipes temporárias	50	11,3
Não há equipe exclusiva	327	73,8

Técnicas de engenharia de software adotadas

Especificação	Nº de empresas	%
Testes de sistema	277	62,2
Testes de campo	259	58,2
Metodologias de desenvolvimento	246	55,3
Controles de versão	239	53,7
Testes funcionais	217	48,8
Testes de aceitação	212	47,6
Análise de requisitos	211	47,4
Prototipação	207	46,5
Programação orientada a objeto	193	43,4
Reuso de código	166	37,3
Testes de unidade	105	23,6
Provas de correção	66	14,8
Planos e estimativas formais	54	12,1
Estimação de confiabilidade	45	10,1
Inspeção formal	45	10,1
JAD	44	9,9
Reengenharia	43	9,7

Ferramentas utilizadas

Especificação	Nº de empresas	%
Gerador de telas ou de entrada de dados	208	46,7
Gerador de relatórios	198	44,5
Dicionários de dados	171	38,4
Gerador de código-fonte	166	37,3
Depurador interativo	148	33,3
CASE	118	26,5
Gerador de gráficos	92	20,7
Gerenciador de bibliotecas de módulos	91	20,4
Documentador	83	18,6
Prototipador	74	16,6
Gerenciador de configuração	46	10,3
Analisador de código	44	9,9
Otimizador	39	8,8
Driver de teste	29	6,5
Gerador de massas de teste	26	5,8
Outras	31	7,0
Não utiliza ferramenta automatizada	48	10,8

Documentação adotada

Especificação	Nº de empresas	%
Manual de usuário	365	83,3
Contratos e acordos	294	67,1
Help on-line	273	62,3
Especificação de sistema	237	54,1
Documentação no código	207	47,3
Documentação comercial	204	46,6
Especificação de programas	199	45,4
Guia de instalação	195	44,5
Material para treinamento	192	43,8
Projeto de sistema	176	40,2
Plano de testes	98	22,4
Resultados de revisões e testes	96	21,9
Não adota documentação	11	2,5

ANEXO B

Glossário de Termos Utilizados

Algoritmos: uma seqüência finita de instruções ou operações computacionais básicas incluídas em um programa de computador, cuja execução, em tempo finito, resolve um determinado problema computacional específico.

Atributos externos de qualidade: conjunto de características externas de um produto (como por exemplo telas de interface, tempo de execução de determinada operação, embalagem, documentação, etc) de software através do qual sua qualidade é descrita e avaliada.

Atributos internos de qualidade: conjunto de características internas de um produto (como por exemplo rotinas para recuperação de erros, documentação do código, estruturação, etc) de software através do qual sua qualidade é descrita e avaliada.

Ciclo de vida: conjunto de todas as etapas que compõem a existência de um sistema de software, desde o projeto até a retirada de operação.

Codificação: etapa do desenvolvimento de software na qual as representações de projeto são convertidas numa linguagem de programação convencional, resultando em instruções que possam ser executadas pelo computador.

Confiabilidade: conjunto de atributos que evidenciam a capacidade do software de manter seu nível de desempenho sob condições estabelecidas durante um período de tempo estabelecido. (NBR ISO/IEC 9126)

Eficiência: conjunto de atributos que evidenciam o relacionamento entre o nível de desempenho do software e a quantidade de recursos usados, sob condições estabelecidas. (NBR ISO/IEC 9126)

Estruturas de dados: localizações de memória (no computador) onde os dados usados pelos programas são armazenados. As estruturas de dados são organizadas conforme os tipos de dados disponíveis na linguagem de programação, de forma a representar o mais fielmente possível as informações consideradas pelo programa.

Fluxo de controle: diagrama em forma de rede que descreve as seqüências de processamento das diversas unidades de um sistema de software.

Fluxo de dados: diagrama em forma de rede que descreve as seqüências de entrada e saída dos dados e informações das diversas unidades de um sistema de software.

Funcionalidade: conjunto de atributos que evidenciam a existência de um conjunto de funções e suas propriedades especificadas. (NBR ISO/IEC 9126)

Modelo de ciclo de vida: modelo de organização das diversas etapas e atividades de projeto de sistemas de software. São exemplos de modelo de ciclo de vida: Modelo Cascata, Modelo Espiral, etc.

Modelo lógico: diagrama ou documento que descreve as informações, programas e demais entidades lógicas de um sistema de software.

Multimídia: termo utilizado para caracterizar aplicações de software que utilizam recursos como som e imagem de forma integrada aos programas.

Peer-review: técnica de revisão de um produto, na qual um colega (peer) do projetista ou do programador revisa a atividade desenvolvida, buscando encontrar erros ou oferecer sugestões de melhoria.

Portabilidade: conjunto de atributos que evidenciam a capacidade do software de ser transferido de um ambiente para outro. (NBR ISO/IEC 9126)

Projeto de alto nível: conjunto de procedimentos, metodologia ou documentação específica relacionados à fase inicial de projeto de sistemas de software.

Projeto básico: o mesmo que Projeto de alto nível.

Reuso: utilização de um programa, de uma rotina ou de uma biblioteca de rotinas na construção de mais de um sistema de software. O reuso implica que o código tenha sido projetado e implementado de forma a possibilitar sua reutilização em programas ou sistemas diferentes, considerando aspectos de abrangência funcional e de independência de ambiente.

Sistema de software: conjunto de programas de computador integrados que implementam uma função especificada.

Sistema de suporte: conjunto de programas de computador integrados que implementam funções necessárias para que outros sistemas de software possam ser executados, como por exemplo os sistemas operacionais, sistemas gerenciadores de bancos de dados, etc.

Subrotina: uma parte funcionalmente independente de código computacional, que pode ter sua execução solicitada de forma simplificada em diversos pontos diferentes do programa, sem a necessidade de duplicação do código.

Tempo real: um software de tempo real é aquele que monitora, analisa e controla eventos do mundo real, provenientes do ambiente externo ao sistema de software, devendo atender a esses eventos em intervalos de tempo restrito.

Temporização: característica de sistemas de software relacionada ao tempo de execução de comandos, subrotinas ou programas.

Testes: etapa do desenvolvimento de sistemas de software na qual os documentos de projeto são revisados ou os programas executados com o objetivo de descobrir e corrigir erros. São exemplos de teste: testes funcionais, testes de programa, testes de integração, revisões de projeto, walkthrough, etc.

Usuário: pessoa que interage diretamente com o software em questão.