UNIVERSIDADE FEDERAL DE PERNAMBUCO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

O QFD COMO FOCO DA SATISFAÇÃO DOS CLIENTES: A QUALIDADE NO DESENVOLVIMENTO DE PRODUTOS E SERVIÇOS PARA A CRIAÇÃO DE SOFTWARE

DISSERTAÇÃO SUBMETIDA À UFPE
PARA OBTENÇÃO DE GRAU DE MESTRE
MODALIDADE MESTRADO PROFISSIONALIZANTE
POR

JOAQUIM CAMERINO MORAES DE SOUZA

Orientador: Profa. Dra. Ana Paula Cabral

RECIFE, SETEMBRO/2008

S729q **Souza, Joaquim Camerino Moraes de**

O QFD como foco da satisfação dos clientes: a qualidade no desenvolvimento de produtos e serviços para a criação de software / Joaquim Camerino Moraes de Souza. – Recife: O Autor, 2008.

x, 76 f.; il., figs., tabs.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia de Produção, 2008.

Inclui Referências Bibliográficas.

1. Engenharia de Produção. 2. Engenharia de Software. 3. QFD. 4. Desenvolvimento de Software. 5. Requisito Qualidade. I. Título.

658.5 CDD (22.ed.) **UFPE/BCTG/2008-232**

UNIVERSIDADE FEDERAL DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

PARECER DA COMISSÃO EXAMINADORA

DE DEFESA DE DISSERTAÇÃO DE

MESTRADO PROFISSIONAL DE

JOAQUIM CAMERINO MORAES DE SOUZA

"O QFD Como Foco da Satisfação dos Clientes: A Qualidade no Desenvolvimento de Produtos e Serviços para a Criação de Software"

ÁREA DE CONCENTRAÇÃO: GERÊNCIA DA PRODUÇÃO

A comissão examinadora, composta pelos professores abaixo, sob a presidência do(a) primeiro(a), considera o candidato JOAQUIM CAMERINO MORAES DE SOUZA **APROVADO.**

Recife, 26 de setembro de 2008.

Profa. ANA PAULA CABRAL SEIXAS COSTA, Doutor (UFPE)

Profa. LUCIANA HAZIN ALENCAR, Doutor (UFPE)

Prof. ANTONIO DE VASCONCELLOS CARNEIRO CAMPELLO, Doutor (UFPE)

É com muita gratidão que dedico à minha família, principalmente aos meus queridos filhos Caio e Pedro, pelo apoio, amor e compreensão durante o desenvolvimento deste trabalho. Por compreenderem minha ausência e estarem sempre presentes apesar das dificuldades desse período.

AGRADECIMENTOS

Quero agradecer, em primeiro lugar, a Deus, por me fortalecer e me dá coragem durante toda minha vida.

Aos meus pais, Pereira e Norma, que me presentearam com a vida e me ensinaram a vivê-la com dignidade. Pela dedicação e esforço que fizeram para garantir minha educação, dando-me oportunidade de chegar até aqui, esta conquista também é de vocês, meus grandes mestres.

Não poderia deixar de agradecer à minha esposa e companheira Kátia o apoio incondicional que me deu durante toda a caminhada, estando sempre ao meu lado, em todos os momentos, incentivando-me a superar as dificuldades.

Agradeço aos meus sogros, Sérgio e Leliana, não só pela amizade de pais que me acolheram em sua família, mas também pela formação da minha esposa e agora dos nossos filhos.

Um agradecimento especial à minha Orientadora Profª Doutora Ana Paula Cabral pelo muito que aprendi, com a dedicação, paciência e responsabilidade no ato de ensinar e de me amparar na hora mais difícil para finalização deste trabalho.

RESUMO

O presente estudo discute a aplicação da metodologia QFD (*Quality Function Deployment*) como foco da satisfação dos clientes, buscando enfatizar os parâmetros da qualidade no desenvolvimento de produtos e serviços para a criação de *software*. O QFD, apontado como meio de organizar o desenvolvimento de produtos a nível operacional, especificamente para o desenvolvimento de *software*, se depara de um lado, com as oportunidades de atendimento do mercado local e, de outro, com as limitações de conhecimentos técnicos, de recursos financeiros e até mesmo com a inexistência de um processo formalizado para a condução das suas atividades. O estudo tem como base a adaptação do QFD das Quatro Fases. Assim, a partir dessa adaptação, simular a aplicação, do modelo adaptado, ao processo de desenvolvimento de produtos, em uma empresa de *software*. Visando assim demonstrar que este será capaz de proporcionar à Organização conhecimento técnico no planejamento e execução das diversas etapas do processo de desenvolvimento do *software*.

Palavras-Chave: Engenharia de *Software*. QFD; Desenvolvimento de *Software;* Requisito Qualidade.

ABSTRACT

This study discusses the application of QFD (Quality Function Deployment) methodology as a focus on customer satisfaction, emphasizing the quality parameters on products and services development in software creation. The QFD, indicated as a path to organize products progress at operational level, specifically on software development. This faced on one hand, with local market necessities, and on the other hand, with limitations of expertise, economical resources and with a lack of formalized process to lead their activities. The study is an adaptation of QFD on Four Stages. In this manner, starting from the adapted model, simulate the application to develop products, in a software company. This work demonstrates that QFD is able to provide for the Organization expertise in planning and implementing during the process of software development.

Keywords: Software Engineering. QFD; Software Development, Quality Requirement

SUMÁRIO

LISTA DE FIGURAS

GLOSSÁRIO

BATCH – Lote;

BROWSER – Navegador, software que permite carregar e exibir páginas na internet;

CAD – Computer-Aided Design, desenhos auxiliados por computador;

CAM – Computer-Aided Manufacturing, processos fabris auxiliados por computador;

CD-ROM – Compact Disc Read Only Memory, CD somente leitura;

CNC – Computer Numerical Control, controle numéricos por computador;

GQT – Gestão da Qualidade Total;

GUI – Graphical User Interface, interface gráfica do usuário;

HARDWARE – Itens físicos, como computadores, impressoras e monitores;

INTELIGÊNCIA ARTIFICIAL – Capacidade de uma máquina imitar o comportamento humano

INTERNET – Rede mundial de comunicação de dados;

INTRANET – Rede particular pertencente a uma organização, apenas para uso interno.

LAN – Local Area Network, rede local de computadores;

LAYOUT – Formato

MAN – Metropolitan Area Network, rede formada entre duas LAN's;

ON-LINE – Em linha;

PDV – Terminal de Ponto-de-Venda;

PLC – Programmable Logic Controllers, controladores eletrônicos lógicos e programáveis;

QFD – Quality Function Deployment, desdobramento da função qualidade;

REAL-TIME – Tempo real;

SISTEMAS ESPECIALISTAS – Software aplicativos construídos com base no conhecimento de especialistas humanos;

SGBD – Sistema gerenciador de banco de dados;

SOFTWARE – Programa armazenado num computador e que permite desenvolver atividades diversas no computador;

STAKEHOLDERS – Partes interessadas, representam todas as pessoas envolvidas com o projeto;

TIME-SHARING – Compartilhado;

UCP – Unidade Central de Processamento;

UML – Unified Modeling Language, linguagem de modelagem unificada;

WAN – Wide Area Network, rede de longa distância;

1. INTRODUÇÃO

O novo ambiente de competitividade, promovido pela globalização, crescimento e evolução da economia, impõe as empresas, em qualquer parte do mundo, a buscarem um contínuo aperfeiçoamento de seus produtos, processos e a eliminação dos desperdícios.

As ineficiências não podem mais ser repassadas ao cliente, pois, agora ele possui opções de oferta proporcionada pela queda da barreira alfandegária e a conseqüente abertura de mercados. Assim a implantação de novos e eficientes sistemas de gestão, tanto no segmento industrial como no de serviços, quer para pequenas, médias ou grandes organizações, é uma necessidade imposta pelo mercado.

É neste ponto que surge a Gestão da Qualidade Total - GQT que consiste, hoje, numa estratégia adulta de desenvolvimento empresarial, que está a ser, ativamente, seguida em todos os setores da economia global, por empresas que procuram tornarem-se lucrativas, através da satisfação do cliente.

Podendo pensar na Qualidade Total como uma “melhoria acelerada da performance” tanto em termos de percepção do cliente, acerca da Qualidade do Serviço, como da rentabilidade do investimento e performance global da empresa. Uma boa implementação da Gestão da Qualidade Total inclui, fomentar o envolvimento e iniciativa dos funcionários, trabalho de equipe e planejamento para a qualidade, associado com técnicas e ferramentas de resolução de problemas, indicadores de desempenho, e análise sistemática quer dos produtos quer dos processos. Além disso, tem como objetivo desenvolver um ambiente de trabalho, que encoraja e desafia as pessoas para aprender, cooperar e explorar todo o seu potencial, em busca da satisfação dos consumidores externos e internos. Mas para que isto ocorra de maneira satisfatória, é necessário o desenvolvimento de *softwares* que estejam de acordo com o perfil do cliente. Este pensamento é embasado pela linha na qual o cliente é o principal avaliador, desdobrado em interno ou externo, conforme explica Souza (2003), através das abordagens de Thurston (1985) e CUNHA (1999).

De acordo com Vianna (2002) esses *softwares* são aplicativos ou programas de computador permeados por conjuntos de comandos, instruções ou ordens elaboradas pelo cliente e/ou usuário para o computador cumprir, visando resolver problemas e desenvolver atividades ou tarefas específicas. Também podem ser entendidos como um código que, aplicado à determinada máquina, possibilita a ela entender determinadas instruções e executá-

las de forma a que o seu objetivo seja cumprido. Como tal conceito costuma ser usado para diversos programas, como calculadoras, relógios inteligentes, telefones celulares e computadores, será empregado, neste trabalho, apenas no sentido de computadores.

Assim, ainda de acordo com Vianna (2002), o programa de computador é um código que é transformado em algo inteligível para a máquina. Seu princípio se baseia em funções escritas em uma linguagem de programação (cada vez mais próxima à humana), e que depois é convertida para a linguagem do computador, transformando-se em um arquivo executável, utilizado pelo usuário para cumprir seu objetivo.

Nota-se, assim, que a parte mais importante do programa é o código que lhe dá origem. Ele é denominado código-fonte, e é o objeto de proteção de direitos autorais do *software*. Tendo acesso a ele, é fácil fazer sua análise, identificando e consertando falhas, além de realizar seu aprimoramento ao incorporar novas funções.

Deve-se notar também que o programa em formato de arquivo executável não permite que se tenha acesso ao código-fonte. Isso se dá porque o código é convertido em linguagem de máquina, indecifrável para quem não tenha um programa que faça o caminho inverso, saindo do executável para se chegar ao código-fonte.

Esses aplicativos geralmente estão destinados ao negócio da empresa, visando atender às atividades das funções empresariais (produção e/ou serviços, comercial, materiais, financeira, recursos humanos e jurídico-legal).

Por outro lado é inquestionável que a Voz do Cliente (GARCEZ, s.d, [Online]) é a essência da definição dos requisitos, no entanto, dependendo do tipo de tecnologia a que se refere o cliente, há muitas alternativas para a sua voz. E é nessa busca da satisfação dos clientes que surge o desdobramento da função da Qualidade, o QFD (*Quality Function Deployment*) que pode ser considerado uma ferramenta apropriada para auxiliar na tomada de decisões para desenvolver produtos novos ou aperfeiçoar produtos existentes, sendo também aplicável a serviços.

Durante o desenvolvimento do *software*, as necessidades do cliente são convertidas em requisitos internos da empresa, chamados de requisitos de projeto. Tais requisitos costumam ser características globais do *software* a ser desenvolvido, geralmente mensuráveis, que irão satisfazer às necessidades do cliente se apropriadamente executadas.

A metodologia da ferramenta QFD está estruturada de forma a levar em consideração pontos de vista dos clientes, da organização, das áreas de produção e de setores de

desenvolvimento segundo as necessidades tecnológicas. A técnica apresenta resultados através de gráficos facilmente compreensíveis, além de várias matrizes que podem ser reutilizadas no futuro.

1.1. Problema

Como as características do *software* são bastante particulares, normalmente acaba-se encontrando diversos problemas associados ao *software*, e que podemos observar na realidade das mais diversas empresas, seja ela pequena, média ou grande.

Inicialmente, uma das principais queixas existentes é a de que custos e prazos não são respeitados, verifica-se que as estimativas feitas não correspondem à realidade, geralmente atribuídas ao fator tempo para a coleta de dados sobre o *software* que o cliente deseja.

Também é notório no mercado que a identificação dos requisitos é ineficiente, resultando em produtos que não atendem à necessidade dos clientes finais. Como falta utilização de técnicas adequadas para a coleta dos requisitos, os *softwares* produzidos não são confiáveis e não têm a robustez necessária.

A interação entre os analistas de sistemas e os clientes é difícil. Falta foco no problema que realmente se quer resolver. Quando está se lidando com o desenvolvimento de *software*, deve-se ter em mente que, normalmente, não se desenvolve um *software* para um problema da área da computação, mas tem que se desenvolver um sistema que atenda a outra área de conhecimento. Ou seja, é de extrema importância a boa comunicação entre analista de sistemas e cliente (PRESSMAN, 2006).

Agregado a estes fatores, existe um alto custo de manutenção envolvido, pois a tarefa de manutenção toma muito do orçamento destinado ao *software*, pois não se pensa na qualidade e facilidade da manutenção quando se está construindo o *software*.

E além de todos estes problemas, os gerentes e coordenadores de projetos se encontram, em muitos casos, despreparados para controlar adequadamente o desenvolvimento de um *software*. Desta forma, a produtividade das equipes é baixa frente à quantidade crescente de demandas de *software*. Desta forma, a produtividade das equipes é baixa frente à quantidade crescente de demandas de *software*, sendo assim, como a metodologia QFD pode ser considerada uma ferramenta apropriada para auxiliar na tomada de decisões para desenvolver produtos novos ou aperfeiçoar produtos existentes, sendo também aplicável a serviços diante das problemáticas apresentadas no desenvolvimento de um *software*?

1.2. Justificativa

As organizações estão cada vez mais utilizando *softwares* como ferramentas de suporte ao seu processo de negócio, objetivando alcançar diferenciais competitivos como qualidade, produtividade e eficiência. Nesse contexto o QFD surge como uma metodologia aderente com a finalidade de auxiliar as organizações na compreensão das expectativas de seus clientes, desdobrando-as em todo o processo de produção do produto, visando garantir que o produto final tenha a qualidade esperada pelos consumidores, fornecendo para a organização o esperado diferencial competitivo.

A relevância social é poder valorizar o papel do cliente através da metodologia QFD, minimizando custos e maximizando a produtividade, resultados obtidos a partir da melhoria da qualidade no processo de desenvolvimento do *software.*

Para demonstrar que a utilização da metodologia QFD contribui com a Engenharia de *Software* que procura fornecer métodos e ferramentas, com o objetivo de garantir a qualidade durante todo o ciclo de vida do *software*, e que pretende introduzir no desenvolvimento de *software* as sistemáticas já utilizadas em outras áreas da Engenharia.

Além da relevância à medida que a experiência na Qualidade Total se foi desenvolvendo, emergiu uma firme percepção em relação aos fundamentos necessários para obtenção desta melhoria na performance e na aplicação do QFD, desenvolvida nesta pesquisa como uma técnica de avaliação do grau de satisfação dos clientes, visando detectar problemas ocorridos e propor melhorias na eficácia da produção de *software*.

Outro ponto relevante deste estudo é demonstrar que o *software* deve ter o seu projeto concebido a partir das necessidades, exigências e expectativas de seus consumidores. Para tanto, o *software* pode ser constituído de um de alguns ou de inúmeros componentes; pode ser elaborado através de único processo, de vários processos ou de processos alternativos, mas sempre observando a qualidade exigida pelo cliente.

A qualidade do s*oftware* como qualquer outro produto ou serviço, pode ser avaliada através da satisfação do cliente.

1.3. Objetivos

1.3.1. Objetivo Geral

Propor um modelo do QFD adaptado para o desenvolvimento de *software*.

1.3.2. Objetivos Específicos

- Verificar na Engenharia de *Software* as dificuldades de satisfazer as necessidades dos clientes (usuários de *software*) fornecendo *software* de qualidade;
- Verificar como a metodologia QFD pode contribuir para atender as necessidades dos clientes de *software* e melhorar a qualidade de *software*;
- Fazer um estudo sobre a aplicação do QFD dentro do contexto de desenvolvimento de *software*, analisando os impactos positivos gerados por esta ferramenta para a satisfação dos clientes.

1.4. Metodologia

Esta Dissertação de Mestrado tem como propósito analisar o QFD - Desdobramento da Função Qualidade como foco da satisfação dos clientes e da qualidade no desenvolvimento de produtos e serviços para a criação de *software*. Especificamente verificando a importância de sua aplicação em uma empresa de desenvolvimento de *software*. A fim de demonstrar como é aplicada a metodologia de QFD, dentro do contexto de produção de *software*, analisando os impactos (ganhos) gerados em cada fase do desenvolvimento, com ênfase a voz do cliente.

Para Barros & Lehfeld (2000, p.64), "em pesquisas, seja qual for a sua tipologia, o levantamento e seleção de uma bibliografia concernente é um pré-requisito indispensável para a construção e demonstração das características de um objeto de estudo". A busca do conhecimento por meio da bibliografia pertinente permitirá o enriquecimento do embasamento teórico do pesquisador.

A pesquisa foi realizada através de levantamento bibliográfico por meio de livros, revistas, jornais, artigos eletrônicos, cujo delineamento será um estudo a fim de validar a utilização do QFD no desenvolvimento de *software* para a obtenção de vantagem competitiva.

A presente pesquisa também teve como método para responder à problemática proposta, o modelo de estudo de caso, a partir das proporções de Yin (2001), de natureza qualitativa, e caráter exploratório, por meio de uma simulação de caso utilizando a metodologia do QFD aplicada ao desenvolvimento de *software*, a ser mais bem detalhadas no capítulo específico.

Mesmo sendo uma simulação, permite demonstrar como se faz para determinar o valor relativo das características de qualidade que devem ser avaliadas e adquiridas para satisfazer o cliente.

A pesquisa qualitativa é apropriada para a avaliação formativa, quando se trata de melhorar a efetividade de um programa ou plano, ou mesmo quando se trata de selecionar as metas de um programa e construir uma intervenção (ROESCH, 1999).

1.5. Organização do estudo

Este estudo é organizado em quatro capítulos. O presente capítulo, a introdução, onde são apresentadas as preliminares da importância da metodologia do QFD para satisfação dos clientes. Neste mesmo capítulo também são apresentados o objetivo do trabalho, a problemática em questão e o método de pesquisa.

O segundo capítulo está estruturado com a revisão da literatura no qual se inicia com a discussão e apresentação conceitos e aplicações do que vem a ser a Engenharia de *Software* e a sua problemática, que se firma no princípio de definir o que deve ser construído, sendo este um dos pontos de maior dificuldade ao se iniciar a construção de um *software*, sendo assim, pertinente abordar as técnicas de requisitos para o desenvolvimento de um *software*. Finalizando-o com uma contextualização detalhada do que vem de fato ser a metodologia QFD, que é o desdobramento da função qualidade, analisando o seu sentido restrito e o seu sentido amplo de diferentes abordagens.

No terceiro capítulo, no qual se descreve o modelo proposto, através de uma simulação de adaptação da metodologia do QFD das Quatro Fases, adaptado para o processo de concepção e desenvolvimento de um *software*. Conforme dito anteriormente, mesmo sendo uma simulação permite demonstrar como se faz para determinar o valor relativo das características válidas e adquiridas para satisfazer o cliente.

Por fim, no quarto capítulo é apresentada a conclusão do trabalho como também sugestões para pesquisas futuras.

2. BASE CONCEITUAL E REVISÃO DA LITERATURA

O presente capítulo busca promover uma base conceitual do que foi encontrado ao longo da pesquisa dentro da literatura com assuntos pertinentes ao tema, trazendo o conceito e aplicação de desenvolvimento de *software*, através da Engenharia de *Software*. Objetivando demonstrar que a voz do cliente forma a qualidade exigida para o desenvolvimento do *software*, e contribui para uma qualidade projetada. O Desdobramento da Função Qualidade (QFD) é analisado até o nível dos componentes, com destaque a qualidade, a tecnologia, o custo e a confiabilidade do *software*.

2.1. Conceito de software

Segundo Pressman (2006), sob o ponto de vista da Engenharia de *Software*, um *software* é um conjunto composto por instruções de computador, estruturas de dados e documentos.

As instruções (ou código) produzem os resultados esperados, de acordo com os requisitos definidos. Já as estruturas de dados são as responsáveis por permitir a manipulação e armazenamento das informações. E por fim, os documentos irão conter uma descrição das funcionalidades previstas no software e da forma como elas foram implementadas. Fazem parte da documentação, ainda segundo Pressman (2006) um plano de projeto, uma especificação dos requisitos necessários, o projeto do sistema e a especificação de testes. Estes diferentes documentos são artefatos produzidos no processo de desenvolvimento do *software*.

2.1.1. O surgimento da Engenharia de *Software*

Os primeiros *softwares* surgiram na década de 50 (BOEHM, 2006). Nesta época, todo o foco dos pesquisadores estava concentrado no *hardware*. O *software* era desenvolvido sem nenhuma técnica de engenharia e a sua distribuição era bastante limitada. O *hardware* estava disponibilizado apenas em grandes centros de pesquisa, e por este motivo, o *software* era pouco conhecido.

De acordo com Boehm (2006) em meados dos anos 60, a situação começou a se modificar. Com o surgimento dos microprocessadores, o *hardware* deixou de representar um problema e, conseqüentemente, deixou de ser o foco dos pesquisadores, que passaram a

investir mais em *software*. Assim, as organizações começaram a desenvolver grandes sistemas, dando origem ao conceito de produto de *software*, que passou a ser comercializado (PRESSMAN, 2006).

Manteve-se a lógica de se desenvolver os sistemas "sem nenhuma metodologia", ou seja, desenvolver sem nenhum planejamento. As equipes de trabalho não tinham nenhuma orientação sobre como desenvolver e nenhum processo de desenvolvimento para seguir. Não havia uma documentação adequada do que estava sendo feito.

Com uma demanda maior e um maior número de usuários do sistema, as manutenções também cresceram. E como fazer para dar manutenção a um sistema que não tem um projeto? Analogamente, como saber se é possível derrubar a parede de uma casa, sem conhecer onde estão as vigas, as estruturas fundamentais? Pensando assim, podemos deduzir que o processo de manutenção destes sistemas se tornou impraticável. Os programadores gastavam horas tentando descobrir como realizar a manutenção, e não se conhecia com exatidão quais seriam os "efeitos colaterais" provocados (PRESSMAN, 2006).

Por todos estes motivos, os custos dos produtos se tornavam muito altos. Os recursos destinados ao projeto nunca eram suficientes. As soluções propostas não agradavam aos clientes. Um novo desafio estava surgindo para as empresas (e é um desafio presente ainda hoje) que não era mais o *hardware*, e sim reduzir o custo e melhorar a qualidade dos *softwares* produzidos (PRESSMAN, 2006).

Em virtude do caos que havia se instaurado nas empresas, alguns questionamentos surgiram: por que o *software* era desenvolvido sem planejamento? Por que não aplicar uma abordagem prática, organizada e ordenada no desenvolvimento dos projetos? Na verdade, quando o *software* começou a ser desenvolvido de acordo com Pressman (2006), ele era tido como um tipo de "arte", mas com o tempo, observou-se que era perfeitamente possível aplicar ao *software* uma abordagem de engenharia, dando origem à Engenharia de *Software*.

É, portanto, notório constatar que de fato existem dificuldades no processo de desenvolvimento de *software*. Porém, deve-se entender que o *software* é um produto abstrato e que apresenta características diferentes de um produto comum, como um televisor. Isto acaba tornando o seu processo de desenvolvimento mais complexo. Além disto, ele apresenta outras características que o diferenciam de um produto manufaturado comum (PRESSMAN, 2006):

1. O *software* é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico, sendo que o desenvolvimento de *software* é fundamentalmente diferente do desenvolvimento de *hardware* (PRESSMAN, 2006);

2. O *software* não se desgasta, mas se deteriora devido às diversas manutenções que pode sofrer. Teoricamente, o *software* inicia com um alto índice de falhas, mas as falhas são corrigidas e ele atinge a estabilidade, não sofrendo com desgaste, como ocorre com um produto comum, que se desgasta a medida que o tempo passa. Entretanto, na prática, manutenções são introduzidas no projeto, e juntamente com estas manutenções são introduzidos novos erros. Desta forma, o *software* acaba se deteriorando frente a sucessivas mudanças (PRESSMAN, 2006).

3. Embora já se observe que a indústria está se preocupando com a montagem de *softwares* baseada em componentes (reusabilidade), a maioria ainda é feita sob medida, em vez de ser montada a partir de componentes existentes.

Segundo Pressman (2006) os gerentes responsáveis pelo desenvolvimento do *software* concentram seus esforços nas soluções de problemas de "primeiro plano": estimativas de prazos e custos imprecisos; a relação inversamente proporcional entre produtividade as pessoas da área de *software* frente a demanda de serviços e a qualidade de *software* cada vez menos adequada. No entanto estes problemas podem ser corrigidos.

Uma das formas de solução é o detalhamento da informação, função, desempenho, interfaces, restrições do projeto e critérios de validação é primordial. Para isso o cliente e o desenvolvedor precisam se comunicar cuidadosamente.

2.1.2. Problemas com o desenvolvimento de software

Tendo em vista que as características do *software* são bastante particulares, acabamos encontrando diversos problemas associados ao desenvolvimento de *software*, e que podemos observar na realidade das mais diversas empresas, tanto maiores, quanto menores.

Para começar, uma das principais queixas existentes é a de que custos e prazos não são respeitados, sendo que as estimativas feitas são bem diferentes da realidade. Existem diversos motivos para que isso ocorra, principalmente, o fato de não termos tempo suficiente para coletar os dados sobre o *software* que o cliente deseja. Ou até mesmo, não termos um processo de estimativa de tempo adequado (PAULA FILHO, 2003).

Também observamos que a identificação dos requisitos é ineficiente, resultando em produtos que não atendem à necessidade dos clientes finais. Como falta utilização de técnicas

adequadas para a coleta dos requisitos, os *softwares* produzidos não são confiáveis e não têm a robustez necessária.

A interação entre os analistas de sistemas e os clientes é difícil. Falta foco no problema que realmente se quer resolver. Quando está lidando com o desenvolvimento de *software*, deve-se ter em mente que, normalmente, não desenvolve um *software* para um problema da área da computação, mas sim, tem que desenvolver um sistema que atenda a uma outra área de conhecimento. Ou seja, é de extrema importância a boa comunicação entre analista de sistemas e cliente (PRESSMAN, 2006).

Existe um alto custo de manutenção envolvido, pois a tarefa de manutenção toma muito do orçamento destinado ao *software*. Outro ponto relevante é que não se pensa na qualidade e facilidade da manutenção quando se está construindo o *software*.

E além de todos estes problemas, os gerentes e coordenadores de projetos se encontram, em muitos casos, despreparados para controlar adequadamente o desenvolvimento de um *software*. Desta forma, a produtividade das equipes é baixa frente à quantidade crescente de demandas de *software*. È em meio desta problemática que surgiu a metodologia do QFD para proporcionar a qualidade devida no desenvolvimento do *software*, onde busca antes de tudo, ouvir as necessidades dos clientes e facilitar o seu funcionamento (CHENG, 1995).

2.1.3. Engenharia de *Software*

A Engenharia de *Software* pretende introduzir no desenvolvimento de *software* as sistemáticas já utilizadas em outras áreas da Engenharia. Pretende levar os custos a níveis aceitáveis, gerenciar o processo de desenvolvimento, permitir o trabalho em grupo, com uma maior produtividade, além de permitir desenvolver *softwares* de qualidade.

Uma definição formal de Engenharia de *Software* é “a criação e a utilização de sólidos princípios de engenharia a fim de obter *softwares* econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais” (BAUER, 1969 *apud* PRESSMAN, 2006).

Para alcançar estes objetivos, a Engenharia de *Software* usa de planejamento, processos sistemáticos, gerenciamento e controle. Pode-se dizer que a Engenharia de *Software* é composta pelos seguintes elementos: Métodos; Ferramentas e Processos.

Os métodos indicam “como fazer”. Desta forma, existem diferentes métodos para as diferentes etapas do desenvolvimento de um *software*. Ou seja, existem métodos que podem

ser aplicados à análise de requisitos de *software* e projeto, à codificação, aos testes e à manutenção.

2.1.3.1. Engenharia de *Software* e seus recursos

Para Sommerville (2003) a Engenharia de *Software* é uma disciplina nova da engenharia, com finalidade de desenvolver *software* com melhor relação custo-benefício.

Os *softwares* e seus respectivos recursos, parte integrante da Tecnologia da Informação, também são subsistemas especiais do Sistema de Informação global das empresas (PAULA FILHO, 2003).

Existem diversos tipos de *softwares*, tais como os *softwares* de base ou operacionais, de rede, aplicativos, utilitários e de automação. Eles dirigem, organizam e controlam os recursos de *hardware*, fornecendo instruções, comandos, ou seja, programas (NORTON, 1996; STAIR& REYNOLDS, 2002; LAUDON & LAUDON, 1996).

A seguir serão relacionados alguns conceitos básicos, porém necessários à gestão da Tecnologia da Informação no que diz respeito a *software* e seus recursos (PAULA FILHO, 2003); (MEDEIROS, 2004).

a) Sistema operacional e redes

O sistema operacional pode ser visto como o administrador geral do computador, incluindo *hardware*, *softwar*e e respectivos dispositivos. Ele determina quais recursos computacionais serão utilizados para realizações de tarefas, solução de problemas, freqüência e prioridade de atividades, a partir de alocação e monitoramento dos recursos computacionais disponíveis. Também pode ser chamado de *software* de base ou conjunto de programas destinados a dar apoio ao sistema global dos computadores atuando com a Unidade Central de Processamento – UCP (PAULA FILHO, 2003).

Os **sistemas operacionais** podem ser divididos em proprietários (desenvolvido pela empresa que comercializa com *hardware* específico) ou genéricos ou portáveis. Os mais populares são o *MS-DOS, Windows (95, 98, 2000, NT, XP, 2003) da Microsoft*®. O *Unix* foi inicialmente desenvolvido para minicomputadores, mas atualmente também funciona em microcomputadores e possui diversos concorrentes. A *IBM* possui o *OS/2 Warp*® (MEDEIROS, 2004).

Com relação às redes, são *softwares* que permitem que os computadores se conectem entre si, através de dispositivos e recursos de telecomunicações. Hoje em dia, a maioria das

empresas tem seus computadores trabalhando em rede, pois isso ajuda na economia de equipamentos, como por exemplo, impressoras, *CD-Rom*, unidade de disquete entre outros dispositivos podem ser acessados de um outro computador através de uma rede local, com isso se economiza tempo na hora de precisar abrir um arquivo que esta em outro computador e dinheiro pois evita a necessidade de se comprar esses dispositivos para todos os micros de uma empresa (PAULA FILHO, 2003).

Além da economia em equipamentos e tempo, através de uma rede, pode se compartilhar também o acesso à *internet* para outros computadores, sendo assim, apenas um fica conectado a *internet* e compartilha esta conexão com os outros computadores da empresa.

Existem três tipos de rede de computadores, segundo Medeiros (2004):

A LAN (*Local Area Network*) é a rede interna ou local de uma empresa, escritório ou casa. Geralmente utilizadas para se acessar os arquivos e dispositivos de outros computadores. Hoje em dia também muito utilizada para jogar jogos.

A MAN (*Metropolitan Area Network*) pode se definir como uma LAN que, ao invés de se comunicar apenas com os micros que estão localizados em um mesmo local, também se comunica com outros micros que estão em uma outra rede local. Por exemplo, uma empresa que tem sede em varias localidades, alguns setores de uma sede precisam se comunicar com o mesmo setor de outra sede, e esta comunicação é através de uma MAN.

A WAN (*World Area Network*) é a mais conhecida de todas, seria a *internet* que interliga computadores do mundo inteiro, fazendo com que pessoas se comuniquem, e compartilhe arquivos, independente de onde estejam (MEDEIROS, 2004).

b) *Software* aplicativos e linguagens de programação

O *software aplicativo* ou programas de computador são conjuntos de comandos, instruções ou ordens elaboradas pelo cliente e/ou usuário para o computador cumprir, visando resolver problemas e desenvolver atividades ou tarefas específicas (RUMBAUGH & BLAHA, 2006).

Tais programas comumente se destinam ao negócio da empresa, e tem como objetivo suportar as atividades dos processos daquela organização.

Todos os programas de *software* são escritos em esquema de códigos chamados **linguagem de programação** que fornecem instruções ao computador para que este possa executar uma atividade de processamento e atingir um objetivo. Como exemplo, pode-se relatar a linguagem *Assembly* (de baixo nível), as linguagens tradicionais *RPG, Basic,*

Fortran, Cobol, Pascal®, C®, as linguagens visuais, *Visual C®, Visual Basic®, SQL Windows, Delphi®*, as linguagens com recursos de inteligência artificial e, ainda, as linguagens de programação orientadas a objetos: *C++®, Smalltalk®, POP 11®, Prolog®* , etc. (RUMBAUGH & BLAHA, 2006).

A linguagem de programação deve ser convertida para a linguagem de máquina para ser executada pela UCP (Unidade Central de Processamento) e isto é feito pelo *software* chamado compilador, que funciona como um tradutor de linguagem, convertendo o *código fonte* em *código objeto* ou *executável.* Os programas fonte e objeto são formas de escrita e execução de programas, ou seja, a fonte é onde e como o programador elaborou passível de alterações e geralmente escrito em inglês (RUMBAUGH & BLAHA, 2006). O programa objeto é o mesmo programa só que compilado, isto é, transformado em linguagem de máquina, executável em computadores. Os *pacotes* normalmente são programas-padrão executáveis desenvolvidos por *software-houses* para determinada aplicação.

É muito importante que os gestores observem essa questão, pois, se a empresa for proprietária do código fonte dos sistemas de aplicativos, também deverá ter a linguagem de programação específica, com seus utilitários, interpretador e compilador.

c) *Software* de automação de escritório ou Office

Também chamados de *software* aplicativos ou utilitários por alguns autores. Essa nova visão de gestão, porém, será distinguido este grupo de *software* e será chamado de *software* de automação de escritório ou *Office.*

Os **editores de texto** são fáceis de usar e trazem grandes benefícios quanto à melhoria da produtividade e a eficiência na elaboração de documentos, de simples cartas até livros ou relatórios técnicos, permitindo elaborar, armazenar, recuperar e editar informações em forma de textos (PAULA FILHO, 2003).

As **planilhas eletrônicas** contribuem na elaboração de relatórios que envolvam cálculos, devido a facilidade de executar diversas fórmulas matemáticas de maneira rápida e efetiva. Outra facilidade é a elaboração de macros (conjunto de várias fórmulas) e de gráficos, com diversas alternativas visuais, possibilitando também o manuseio dos valores e seus respectivos resultados dinâmicos (PAULA FILHO, 2003).

Os ***softwares* de apresentação** trazem ricos recursos de demonstração de dados e informações. Eles permitem utilizar recursos de multimídia (sons, imagens, vídeos), de gráficos, textos predefinidos e outros recursos de apresentação (PAULA FILHO, 2003).

Segundo o autor, esse tipo de *software* permite conjugar os diversos recursos disponíveis, facilitando uma apresentação, fazendo com que quem esteja observando acompanhe a evolução de um tema, fase a fase, por exemplo. Também são destinados à apresentação de resultados empresariais e/ou produtos que necessitem uma boa forma de aparências ou *marketing*.

O **banco de dados** é uma ferramenta que possibilita armazenar e organizar, classificar, recuperar e manipular dados, possibilitando uma grande diversidade de aplicações. Essa ferramenta, para ser efetiva, exige do cliente e/ou usuário maior atenção, análise e planejamento, ou seja, treinamento e aprofundamento específico (PRESSMAN, 2006). Os bancos de dados presentes num pacote *Office* podem ter limitação quanto ao tamanho e quantidade de dados a serem manipulados, bem como, geralmente, não possuem as prerrogativas de relação automática de registros, tais como os Sistemas Gerenciadores de Bancos de Dados (SGDB). São, porém, fortemente recomendados para pequenas aplicações destinadas a automação de escritórios. Esse *software* quando bem utilizado, contribui significativamente nos aspectos organizacionais da empresa, integrando informações e organizando dados (PRESSMAN, 2006).

Ainda neste grupo de *software* de automação de escritórios ou *Office*, podem aparecer os **pacotes integrados** que são recomendados para a resolução de problemas que exigem a junção de *software* de automação de escritórios, mesclando seus recursos e produtos. Esses pacotes também podem oferecer recursos de correio eletrônico (PRESSMAN, 2006).

Hoje em dia existem vários pacotes *Office* para serem utilizados, sendo os mais conhecidos o pacote *Microsoft® Office*, da Empresa de *softwares Microsoft® Corporation*, o *StarOffice* da *SUN Microsystems*, e o *OpenOffice*, que é um pacote *Office* de código aberto que tem como código inicial o mesmo do *StarOffice* da *SUN*, que liberou o código para que desenvolvedores continuassem o projeto do *OpenOffice* e hoje a *SUN* investe apenas no *StarOffice*. E Coincidência ou não, o *OpenOffice* que é de código aberto tem uma aceitação muito maior que o *StarOffice* (PRESSMAN, 2006).

d) Softwares utilitários

Os *softwares* utilitários têm como função principal a complementação dos *softwares* de automação de escritórios e dos aplicativos (PRESSMAN, 2006)..

Os *softwares* de cópia, também chamados de *backup*, têm como função básica de salvar os dados e/ou informações em dispositivos extras. Além de salvar os dados, devem ser

capazes de recuperar os mesmos, ou seja, retornar da cópia para a base de dados principal (PRESSMAN, 2006).

Os *antivírus*, também chamados de "vacinas", têm como função básica proteger as bases de dados da empresa contra vírus. Os vírus são programas de computadores que têm por objetivo causar danos ou apagar ou roubar os dados ou informações armazenadas nos computadores. Além de proteger os dados, o *antivírus* deve possibilitar a exclusão de eventuais vírus nos sistemas. Hoje em dia, os *antivírus* utilizados são o *Norton Antivírus*, da empresa de *softwares Symantec Corporation*, o *AVG* da *Grisoft*, o *Panda Antivírus* da *Panda Software*, e o *McAfee Antvírus* da *McAfee* (PRESSMAN, 2006).

Os **compactadores** são recursos de software que têm como objetivo principal a compactação de dados armazenados nos dispositivos dos computadores, a fim de reduzir os espaços utilizados por estes dados. As reduções de espaço tanto são úteis para armazenamento principal ou nos dispositivos de cópia de dados. Existem hoje vários programas para a compactação de arquivos, sendo os mais conhecidos e utilizados o *WinZIP* e *WinRAR* (PAULA FILHO, 2003; PRESSMAN, 2006).

Os **desfragmentadores** são recursos de *software* que têm como objetivo principal a reorganização de fragmentos de dados que estão armazenados nos dispositivos dos computadores. À medida que os clientes e/ou usuários vão criando e apagando arquivos, são criados diversos pequenos espaços fragmentados aleatórios, que devem ser desfragmentados, ou seja, reorganizados (PRESSMAN, 2006). A desfragmentação dos arquivos de disco é bastante aconselhável quando seu micro está com um desempenho abaixo do esperado. Existem programas como o *Speed Disk* que vem no pacote do *Norton System Works* que faz essa desfragmentação dos arquivos do disco, mais também existe o próprio desfragmentdador do *Windows* que também realiza a mesma tarefa com eficiência (PAULA FILHO, 2003).

Os ***softwares*** vinculados aos recursos da *internet* permitem principalmente os processos de troca e uso de informações, por meio de recursos de telecomunicação. Os *softwares* de comunicação na *internet* disponíveis no mercado permitem acessar a rede mundial de computadores; a *Intranet* permite a comunicação interna na empresa e a *Extranet* permite o acesso aos dados da empresa de fora da mesma e via *internet*. Atualmente, os recursos da *internet* são de muito valor para as empresas.

Os ***browsers*** são *softwares* com ferramentas para acesso a *World Wide Web* (WWW), ou seja, a rede mundial de computadores. Eles oferecem uma interface gráfica, com recursos

de multimídia e textos de informações. O *browser* padrão para usuários do *Windows* é o *Internet Explorer* da *Microsoft*®, porém existem hoje outros muito utilizados, como o *Netscape*, o *Firefox*, dentre outros (PRESSMAN, 2006).

Os comunicadores instantâneos são, hoje em dia, os programas mais utilizados pelos usuários da *internet*, pois permite a conversa entre duas ou mais pessoas em tempo real, independente de onde é que elas estejam. Tais aplicativos também permitem a troca de arquivos, conversas por voz, videoconferências, etc.. Os mais utilizados hoje em dia são o *MSN Messenger* da *Microsoft*® e o *ICQ da Mirabilis*, sendo que o primeiro vem dominando o mercado a algum tempo (PRESSMAN, 2006).

e) *Softwares* de automação

Os ***softwares* de automação** tratam as automações industriais, comerciais e de serviços. A automação industrial é a interface com diversas tecnologias, a exemplo de coletores e controladores eletrônicos lógicos e programáveis (PLC), controles numéricos por computador (CNC), pneumáticos, sensores, desenhos auxiliados por computador (CAD), processos fabris auxiliados por computadores (CAM), dentre outros (KOTONYA & SOMMERVILLE, 1997).

A automação comercial é a interface com diversas tecnologias, tais como caixas registradoras, PDV's (terminais de ponto-de-venda), impressoras de cheques, código de barras, leitoras ópticas, balanças eletrônicas etc. (KOTONYA & SOMMERVILLE,1997).

A automação de serviços é a interface com diversas tecnologias, tais como sistemas de controle de documentação e imagens, sistemas de atendimento, contratos, agências bancárias, sistemas de desenhos, arquitetura e engenharia etc., ou seja, para aperfeiçoar serviços específicos de determinadas empresas.

f) Outros softwares e demais recursos

Além desses *softwares* relatados aqui, ainda existem outros que podem auxiliar os gestores das empresas em suas atividades.

A computação gráfica ou editoração eletrônica permite produzir cartazes, folhetos, jornais, revistas, imagens, gráficos, *layouts* complexos etc. (PRESSMAN, 2006).

A multimídia ou hipermídia apresenta em um mesmo momento diferentes recursos da informação, tais como imagens, som, textos etc.

A realidade virtual possibilita as situações e as simulações apresentadas em computador, permitindo que os clientes e/ou usuários participem ativamente do processo em questão, vivenciando situações imaginárias em tempo real.

Outros *softwares* que também podem ser relatados seriam os sistemas tutoriais, *softwares* educativos, que tem como função principal o treinamento de um tema qualquer. E os *sistemas especialistas* com recursos de *inteligência artificial*, que tratam de domínios específicos.

A *interface gráfica* com o cliente e/ou usuário, *Graphical User Interface* – GUI, utiliza figuras ou ícones na tela e menus para enviar os comandos do computador para os clientes e/ou usuários (KOTONYA & SOMMERVILLE, 1997).

Os tipos de processamento relatam sobre a maneira como os dados são processados em determinado sistema, ou seja, *batch, on-line, real-time e time-sharing.* O processamento *batch* é feito em lote, tal como para lançamento em contabilidade ou folha de pagamento. O processamento *on-line* é elaborado em linha ou instantâneo, tal como consulta a saldo bancário em terminal (PRESSMAN, 2006). O processamento *real-time* existe quando requer um tempo real para sua execução, tal como, extrato bancário em terminal. E finalmente o processamento *time-sharing* é o processamento compartilhado por vários clientes e/ou usuários, tal como, uma rede de computadores executando várias atividades ao mesmo tempo.

2.1.4. Visão geral do software e seus recursos

A visão geral do *software* e seus recursos também têm conotação de gestão, ou seja, quais os elementos fundamentais e necessários para funcionamento dos Sistemas de Informação e processos da empresa. Reiterando que mesmo não tendo o conhecimento técnico depurado, o gestor deve conhecer esses conceitos elementares (PRESSMAN, 2006).

Para a gestão da Tecnologia da Informação quanto ao *software* e seus recursos, a visão geral pode ser assim representada na figura 2.1.

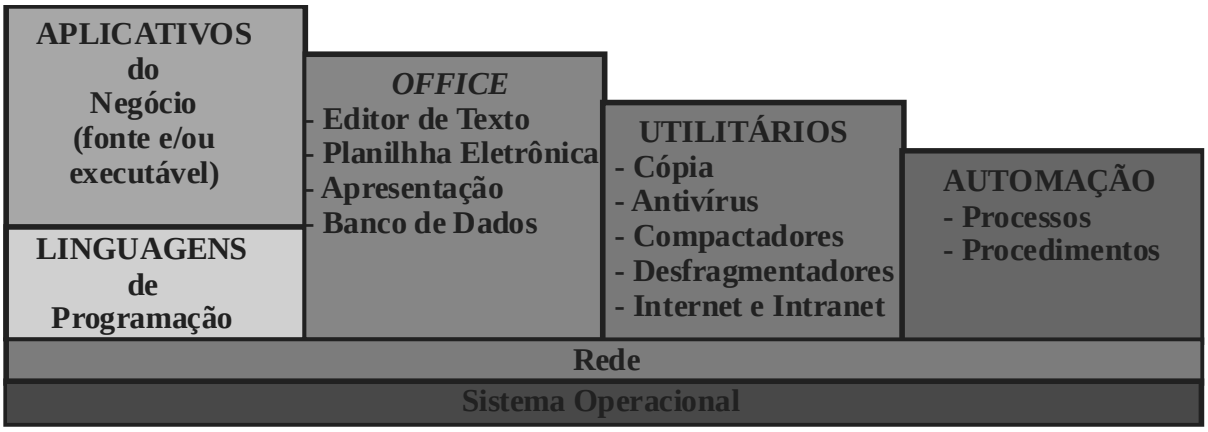


Figura 2.1 – Visão geral do software e seus recursos
Fonte: PRESSMAN, 2006.

2.2. Engenharia de requisitos de software

2.2.1. A problemática da engenharia de requisitos

De acordo com Sommerville (2003) a engenharia de requisitos é o processo de descobrir, analisar, documentar e verificar as funções e restrições do dustema.

Para Sommerville & Sawyer (1997) o documento de requisitos é o principal artefato trabalhado na engenharia de requisitos. Este se configura como uma declaração formal dos requisitos para os *stakeholders*, aplicável a clientes, usuários finais ou a equipe de desenvolvimento do software.

Sommerville & Sawyer (1997) orienta que cada organização deve desenvolver um processo de engenharia de requisitos adequado a sua realidade.

O requisito para Sommerville (2003) é um processo que determina os serviços que o cliente demanda de um sistema, assim como as restrições com qual opera e é desenvolvido.

Os requisitos são a descrição dos serviços do sistema e restrições que são geradas durante o processo de engenharia de requisitos.

Identificar um requisito pode variar de uma descrição abstrata de alto-nível de um serviço ou uma restrição do sistema, até um detalhamento específico funcional matemático.

Sommerville (2003) explica que possuem uma função dupla, de um lado pode ser a base para uma oferta de contrato – portanto, carece estar aberta a interpretação, de outro modo influencia o próprio contrato - então deve ser definido em detalhes.

Os tipos de requisitos baseados em Sommerville (2003) podem ser de utilizador ou de sistema. O primeiro com base a linguagem natural, mais diagramas dos serviços que o sistema fornece e suas restrições operacionais. Observadas para o cliente. O requisito do sistema é um documento estruturado, acrescido de descrição detalhada das funções do sistema, serviços e restrições operacionais. Neste momento define-se o que irá ser implantado e consensuado entre o cliente e o contratado através de um contrato.

Como requisito do utilizador o *software* deve oferecer meios de conceder e aderir a ficheiros externos criados por outras ferramentas (SOMMERVILLE, 2003).

Definir o que deve ser construído, este é um dos pontos de maior dificuldade ao se iniciar a construção de um *software*. O conjunto inicial de requisitos deve ser representativo daquilo que o usuário quer e precisa, pois de nada vale um *software* cheio de funcionalidades e opções, se o público-alvo do sistema, na verdade, precisa de um sistema “enxuto” e simples.

Você pode se perguntar: quais são as conseqüências de não se definir os requisitos adequadamente? Conforme aborda Sommerville (2003) Um dos grandes problemas é que requisitos mal definidos, ou que não atendam às expectativas dos clientes, exigem reparos durante o projeto de *software*, ou na sua construção, ou na sua manutenção. Quanto maior a demora para corrigir um requisito, maiores os custos para corrigi-lo. Este custo não envolve só o gasto maior de tempo e recursos. Pode envolver a perda de credibilidade da empresa, a perda do próprio cliente e uma quantidade muito maior de preocupação para os analistas e programadores que precisam resolver o problema.

Mas o que são exatamente estes requisitos, que podem representar tantos problemas para o *softwa*re? O requisito pode ser definido como uma função ou característica que o *software* deve ter para que possa alcançar seus objetivos. Ele pode também ser uma propriedade geral do sistema, ou, até mesmo, uma restrição para o sistema (SOMMERVILLE, 2003).

Summerville (2003) explica que é na fase de concepção que os requisitos são definidos, dando uma visão geral daquilo que o sistema é, ou deve ser. Entre os documentos produzidos nesta fase, temos o documento de visão e o documento de requisitos, que contém os requisitos funcionais e não funcionais.

A obtenção destes documentos envolve problemas diversos, iniciando com a comunicação entre usuários e analistas, onde pode ocorrer má interpretação do problema. O cliente, normalmente, possui apenas uma vaga idéia do que realmente quer e os requisitos que ele propõe podem ser perfeitos hoje, mas já não o são amanhã (PRESSMAN, 2006)

Para auxiliar neste processo de definição de requisitos, temos a Engenharia de Requisitos, que é uma sub-área da Engenharia de *Software*. Ela fornece métodos, técnicas e ferramentas que dão suporte adequado às tarefas de produção e gerência dos requisitos do sistema (PRESSMAN, 2006). Para Pressman (2006) tanto o cliente quanto o analista assumem um papel ativo na engenharia de requisitos. Ao primeiro, cabe a tarefa de informar como é o negócio que se quer implementar, e estas regras nem sempre são repassadas da forma mais clara. O segundo deve ser capaz de transformar este conhecimento repassado pelo cliente em um sistema. Para poder transformar o conhecimento em um bom sistema, o analista deve questionar cada detalhe do negócio, para garantir que toda informação seja repassada e interpretada de forma adequada.

Vários autores têm proposto metodologias diversas para a engenharia de requisitos. O processo proposto por Kotonya & Sommerville (1997). Para estes autores, a engenharia de requisitos tem quatro etapas: elicitação, análise e negociação, documentação e validação, conforme se visualiza na figura 2.2.

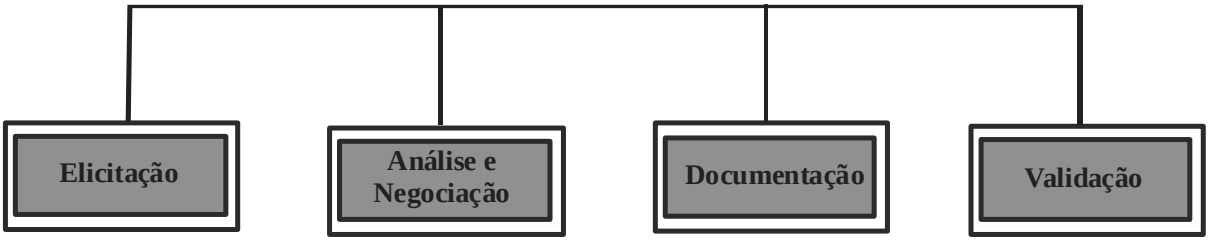


Figura 2.2 - Etapas da Engenharia de Requisitos
Fonte: Adaptado de Kotonya & Sommerville (1997)

A **elicitação** é a etapa onde se entende o problema e se obtém os requisitos juntamente com os usuários do sistema. Aqui se faz a coleta e identificação das fontes de dados. O conhecimento pode ser obtido por diversas técnicas (KOTONYA & SOMMERVILLE, 1997).

Em seguida, é feita a **análise e negociação** dos requisitos obtidos pela elicitacão. Nesta fase, os requisitos são analisados e os usuários envolvidos negociam para decidir quais requisitos realmente irão fazer parte do sistema. A elicitação pode produzir requisitos conflitantes (KOTONYA & SOMMERVILLE, 1997). Um exemplo de caso que pode dar origem a requisitos conflitantes: diferentes usuários com visões distintas do negócio. Cada um deles pode julgar que a sua visão é mais relevante, e por conseqüência, fazer propostas diferentes para a mesma situação. Neste caso, a negociação é indispensável para se chegar a um consenso sobre os requisitos necessários.

Na **documentação**, os requisitos são documentados em um nível adequado de detalhes. A documentação de requisitos é feita, normalmente, usando a linguagem natural, e deve ser escrita de tal forma que todos os usuários entendam a documentação (KOTONYA & SOMMERVILLE, 1997). Normalmente, o profissional irá desenvolver sistemas para áreas diversas. Neste ponto, um documento que auxilia bastante a comunicação entre os usuários e os analistas e desenvolvedores é o glossário, que é um dos artefatos previstos no Processo Unificado. O glossário traz os conceitos dos principais termos envolvidos no processo, usando a linguagem do cliente.

E por fim, a **validação** envolve uma checagem cuidadosa, para verificar se os requisitos estão completos e consistentes. Este processo detecta problemas no documento de requisitos,

antes que este documento comece a ser utilizado para as próximas etapas (KOTONYA & SOMMERVILLE, 1997).

2.2.2. Técnicas de requisitos

Normalmente, a elicitação de requisitos envolve conversas com os clientes, estudo de documentos, observação do comportamento e de outros sistemas de *software* já usados na empresa (SOMMERVILLE, 2003).

Um fato importante a se observar ao coletar requisitos consiste no fato de que muitas características são consideradas triviais para os usuários, pois já são automáticas ligadas aos seus processos diários. Estes "detalhes" não costumam ser mencionados nas entrevistas porque, de acordo com o usuário, "é óbvio". Muitas vezes, ao terminarmos um sistema, o usuário nos questiona sobre uma determinada funcionalidade que não está contemplada no sistema. Ao respondermos que tal funcionalidade não foi desenvolvida, ele nos responde: "mas era óbvio que esta funcionalidade era necessária". Por exemplo, em um sistema de contabilidade, é óbvio que são necessários relatórios analíticos e sintéticos. Mas esta realidade só é óbvia para os profissionais da área. Então, o analista deve perguntar tudo e deixar bastante claro que o que está sendo documentado é o que será produzido. Algumas vezes, tão importante quanto deixar claro aquilo que se espera do sistema, é deixar claro aquilo que o sistema não irá contemplar (SOMMERVILLE, 2003).

Existem diversas categorias de técnicas para a elicitação de requisitos, técnicas tradicionais convivem com novas técnicas, a fim de melhorar a identificação dos requisitos e reduzir problemas oriundos de uma má elicitação (BATISTA & CARVALHO, 2004).

Uma técnica bastante popular para a elicitação de requisitos é a entrevista. Na entrevista, é feito um contato direto com os atores, questionando-os sobre as necessidades do projeto. A entrevista pode ser livre (categoria levantamento não estruturado), ou estruturada (categoria levantamento estruturado). Na entrevista estruturada, é utilizado um questionário previamente elaborado (KOTONYA & SOMMERVILLE, 1997).

Segundo Kontonya & Sommerville (1997) as reuniões (categoria levantamento não estruturado) são uma extensão da entrevista, onde se encontram vários usuários que irão expor as diversas visões do sistema. Um problema comum da reunião é a dificuldade de se manter o foco dos usuários naquilo que se pretende do sistema, sem dispersão.

Na observação, um analista responsável se insere no ambiente do usuário final, observando as suas atividades. Todo comportamento observado é anotado. A observação pode ser feita com ou sem intervenção da pessoa que observa.

Ao final do processo de elicitação, independente da técnica que seja aplicada, espera-se ter um documento de requisitos que seja claro, bem escrito e que tenha requisitos que não tragam ambigüidade. Além disso, os requisitos devem ser escritos da maneira mais simples possível, evitando frases complexas, que na verdade indicam mais de um requisito. Espera-se que este documento contenha informações relevantes, que seja completo e sem inconsistências (KOTONYA & SOMMERVILLE, 1997).

Sommerville (2003) classifica os requisitos em funcionais, não-funcionais e requisitos do domínio. Os requisitos funcionais são as declarações de serviço que o sistema deve fornecer, como este se comporta a entradas particulares e como se comporta a situações específicas. Os requisitos não-funcionais são as restrições os serviços ou funções determinadas pelo sistema, a exemplo das restrições temporais, restrições no processo de desenvolvimento, padrões, etc. Por fim, Sommerville (2003) explica que os requisitos do domínio apresentam-se com o domínio de aplicação do sistema e que refletem características do domínio.

De acordo com Wazlawick (2004) Os requisitos funcionais devem traduzir aquilo que se espera do sistema, estando diretamente vinculados à funcionalidade do *software*. Devem descrever funções que o sistema precisa fornecer e como deve se comportar em determinadas situações, correspondendo a uma listagem de tudo que o sistema deve fazer.

Abaixo, seguem alguns exemplos de requisitos funcionais, segundo Wazlawick (2004):

- O sistema deve emitir um relatório mensal de clientes inadimplentes;
- O sistema deve prover um formulário para entrada de dados de um paciente;
- O sistema deve emitir um comprovante de pagamento para o cliente.

Os requisitos não funcionais são restrições colocadas sobre como o sistema deve realizar seus requisitos funcionais. Eles acabam por restringir as nossas opções para criar uma solução para o problema (WAZLAWICK, 2004).

Os requisitos não funcionais podem ser classificados segundo diferentes categorias. Ainda segundo Kotonya & Sommerville (1997), algumas categorias para classificação de requisitos não-funcionais são:

Requisitos de Produto: são aqueles que definem como o produto deve se comportar. Alguns exemplos de requisitos de produto são os requisitos de eficiência e de confiabilidade.

Requisitos de Processo: são conseqüências das políticas e normas estabelecidas pela organização ou pelo desenvolver. Entre estes requisitos, temos os requisitos de padrão, de implementação e de entrega.

Requisitos Externos: são aqueles que provêm de fatores que são externos ao sistema e ao seu processo de desenvolvimento. Por exemplo, temos os requisitos legais, que garantem que o sistema está de acordo com a lei vigente.

Após o levantamento de requisitos, é necessário gerenciá-los. Ou seja, por melhor que seja a qualidade dos requisitos, eles inevitavelmente vão se modificar ao longo do projeto (KOTONYA & SOMMERVILLE, 1997).

Uma técnica útil para gerenciar os requisitos é o uso de matrizes de rastreabilidade. Por meio destas matrizes, é possível organizar os requisitos, informando como eles estão interligados. Assim, mudanças em um determinado requisito podem ser facilmente rastreadas, verificando-se outros requisitos que possam sofrer conseqüências da alteração realizada (KOTONYA & SOMMERVILLE, 1997).

2.2.3. UML - *Unified Modeling Language*

A Linguagem de Modelagem Unificada (UML - *Unified Modeling Language*) surgiu em 1997 e deste então tem se tornado padrão para desenvolvimento de *software*. Descrito como linguagem e não um método ou processo de Engenharia de *Software*, conforme define Pender (2004). A UML é constituída por uma notação específica e usa regras relacionadas à gramática pra construção de modelos de *softwares.*

Pender (2004) explica que a UML é compreendida como um conjunto de documentos, servindo de padrão para diversos segmentos da Engenharia de *Software*.

A diferenciação da UML, que é uma linguagem de modelagem, é que esta não depende de sua sintaxe e sim de sua semântica. O seu êxito fica a cargo de seu criador, já que um programa computador depende de instruções corretas, para a sua interpretação.

Através da UML são disponibilizados para os desenvolvedores vocabulários gráficos, através de diagramas, que representam uma parte ou aspecto de um sistema. Um sistema modelado em UML é composto de vários diagramas. Estes podem possuir formas variadas, podendo ser demarcados por caixas ou quadrados, ou então contidos em pacotes (PENDER, 2004).

Ainda de acordo com Pender (2004) algumas características dos diagramas são:

a) Não há geometria ou geografia para os elementos que possam influenciar um diagrama UML. Para a maioria dos casos, o tamanho de um símbolo ou localização não está vinculado ao seu conteúdo semântico, exceto para os diagramas que possuam uma dimensão de tempo;
b) Todos os diagramas são de duas dimensões, e este limite é imposto pelas ferramentas e tecnologias atuais, havendo, contudo, exceções, como a utilização de cubos;
c) O texto pode ser utilizado para diversos propósitos em um diagrama. Exemplos disso podem ser: a definição de regra e a identificação de atributos.

2.2.3.1. Casos de Uso

Os casos de uso são classes que determinam unidades de funcionalidade ou comportamento disponíveis em um sistema. Representam os requisitos externos de um sistema e a funcionalidade dada por ele. O conjunto de casos de uso pode estar descrito por um retângulo que abarca o sistema como um todo. Os casos de uso são utilizados para exemplificar unidades de trabalho compreendidas em um sistema e que fornece serviços a usuários externos. O caso de uso é detalhado pelo diagrama de casos de uso, cujo objetivo é a de representar as interações externas entre o sistema e seus atores (PENDER, 2004).

2.2.3.2. Diagrama de Casos de Uso

O diagrama de casos de uso é embasado pela visão do usuário sobre um cenário de uso de um sistema que esta sendo modelado. O caso de uso é útil por ser simples e de fácil aprendizado.

O objetivo dos casos de uso, na Engenharia de *Software*, é capturar, documentar e validar requisitos, como também descrever as funcionalidades do *software*, conforme explica Cockburn (2001).

De acordo com Cockburn (2001) o diagrama de casos de uso identifica e transcreve o comportamento do sistema ou subsistema, substituindo a tradicional especificação funcional de um sistema. Em seus resultados responde a pergunta: qual o propósito do sistema para cada usuário? Percebe-se a funcionalidade do sistema e a perspectiva do usuário desse sistema.

O diagrama é constituído de atores, casos de uso e seus relacionamentos, e percorre as etapas: modelagem de atores; representação da comunicação do modelo entre os usuários

(elementos externos) e o sistema que está sendo modelado; modelagem de casos de uso (COCKBURN, 2001).

2.2.3.3. Diagramas de Classes

Os diagramas de classes pretendem apresentar a estrutura estática de um sistema (WASLAWICK, 2004). Segundo o autor os diagramas de classes pretendem fornecer uma visão de todas as "estruturas" que serão manipuladas ou gerenciadas pelo sistema. O elemento fundamental deste tipo de diagrama são as classes.

O diagrama de classes pode representar diferentes aspectos do sistema. Em um processo de *software*, você pode ter o diagrama de classes de domínio, que trata da aplicação em um nível mais alto, independente de tecnologias que possam ser utilizadas. Neste nível, são consideradas apenas as classes referentes ao negócio sob estudo, ou seja, ao domínio do problema, visando auxiliar na compreensão dos elementos envolvidos e na forma como estes elementos se relacionam (WASLAWICK, 2004).

Todos os detalhes referentes à solução do projeto ficam para uma fase posterior, que é a fase de projeto, onde devem ser vistos os detalhes de interface e formas de armazenamento (banco de dados, segurança de acesso, comunicação, entre outros) (WASLAWICK, 2004).

Todos estes detalhes referentes à implementação são representados no diagrama de classes de projeto, que se torna uma extensão do diagrama de classes de domínio. Ou seja, os dois diagramas não tratam de coisas diferentes. As classes definidas no nível de negócio devem ser mantidas e deve ser feita a adição de detalhes específicos, que vão depender das tecnologias escolhidas para desenvolvimento do sistema.

A classe representa conceitos que não podem ser descritos simplesmente por um tipo de alfanumérico, ou seja, representa um conceito mais complexo (WAZLAWICK, 2004).

Sendo assim, as classes representam entidades do mundo real que estão envolvidas no negócio, analisado e que resultará no *software*.

A visibilidade indica se o atributo está disponível para ser utilizado por outras classes. Um atributo pode ser público (representado pelo símbolo +) ou privado (-). Uma terceira opção é que o atributo seja protegido (#). Neste caso, o atributo pode ser acessado somente por classes derivadas da classe a qual o atributo pertence.

Depois da visibilidade, vem o nome do atributo, que deve identificar o atributo exclusivamente dentro da classe e pode vir seguido da multiplicidade, usado para o caso de atributos múltiplos, como vetores. A UML sugere que a primeira palavra do nome do atributo

deve iniciar por letra minúscula e as demais, por letras maiúsculas. A seguir, vem o tipo, indicando o tipo de dado que aquele atributo pode comportar (char, real, inteiro, entre outros) (MEDEIROS, 2004). A cada atributo pode ser atribuído um valor inicial assim que ele é criado, sendo que esta opção irá depender da linguagem de programação usada, que deve fornecer suporte para tal função.

Alguns exemplos de declaração de atributos para a classe "Professor" seriam:

- dataNascimento: data: Indica que o atributo data de nascimento é do tipo data e deve estar protegido de acesso externo.

+ nome: char: Indica que o nome é do tipo caráter e pode ser acessado por métodos de outras classes (WAZLAWICK, 2004).

A visibilidade segue os mesmos critérios dos atributos. O nome do método também deve identificá-lo de forma única dentro da classe, e a primeira palavra do nome deve iniciar com letra minúscula e as demais, com letra maiúscula. Os parâmetros irão indicar as variáveis de recebimento e retorno do método e o valor de retorno indica se o método retorna algum valor ao término de sua execução. Alguns exemplos de métodos estão definidos a seguir:

+ calcularValor(val1:int, val2:int): int: Função que retorna um valor inteiro, e que pode ser acessada externamente.

+ armazenarDados(nome:char, salario:float) : Método que armazena os dados enviados como parâmetro, e que pode ser acessada externamente (WAZLAWICK, 2004).

Normalmente, não se consegue definir todas as classes envolvidas em uma primeira tentativa. Fazer a identificação das classes e dos seus atributos é um trabalho que, normalmente, passa por sucessivos refinamentos. Quando um requisito modifica, ou mesmo um caso de uso é reestruturado, as classes devem ser revisitadas de forma que sempre reflitam as mudanças que foram especificadas.

2.3. QFD – desdobramento da função da qualidade

A gestão da qualidade se disseminou na década de 80 com a revolução provocada pelos produtos japoneses, que ganharam o mercado americano com preços mais acessíveis e qualidade superior. Diante desse cenário as organizações começaram a tomar consciência da necessidade de desenvolver a indústria americana, no sentido de tornar seus produtos competitivos mundialmente, desta forma a qualidade passou a ser vista como ponto estratégico fundamental para o crescimento e sobrevivência de várias indústrias.

Segundo Lins (1993) a adoção pela qualidade pressupõe a utilização sistemática de instrumentos que auxiliem na compreensão de problemas e solução de erros com base em dados fundamentados no cotidiano das organizações.

Os gestores empresariais brasileiros, frente ao momento de turbulência e de fortes transformações sócio-econômicas, têm buscado novas formas para a sobrevivência de suas organizações, assumindo um comportamento versátil e com grande poder de adaptação aos novos tempos.

2.3.1. Conceitos de QFD

As empresas buscam metodologias que suportem e integrem melhor o trabalho de pesquisa de mercado e o processo pró-ativo de desenvolvimento de produtos e serviços. O QFD representa uma metodologia desenvolvida pelos japoneses, criado principalmente pelos professores Mizuno (1994) e Akao (1990). Este método tanto pode ser aplicado a produto da empresa, quanto a produto intermediário entre cliente e fornecedor interno, na busca do ganho competitivo no mercado global, através de produtos e serviços superiores e que hoje é utilizada pela grande maioria das empresas de classe mundial.

Segundo Eureka & Rayan (1992) o QFD não deve ser usado para qualquer componente de qualquer produto, nem necessariamente para cada produto. O procedimento feito desta forma eliminaria a principal qualidade do QFD – colocar no mercado um produto de alta qualidade no menor espaço de tempo possível.

À medida que a experiência na Qualidade Total se foi desenvolvendo, emergiu uma firme percepção em relação aos fundamentos necessários para obtenção desta melhoria na performance como também na aplicação do QFD. Para Eureka & Rayan (1992) o sucesso do QFD se deve a criação de um plano disciplinado a ser seguido pelas pessoas envolvidas, proporcionando uma troca eficiente de informações. Estes fundamentos poderão ser descritos nos seguintes termos.

Centralização no Cliente – Definida como a capacidade para oferecer produtos e serviços que vão ao encontro das expectativas dos clientes, tendo como base o conhecimento e antecipação das suas necessidades. Apenas quando todos, na empresa, compreenderem o que o cliente deseja, e como poderão, enquanto indivíduos, contribuir para a satisfação do cliente, poderá a organização atingir o pretendido nível global em performance da Qualidade Total. Nas organizações de Qualidade Total, é reconhecido que o cliente só será satisfeito, em última análise, se todos os aspectos do produto, ou serviço, estiverem corretos. A idéia dos

fornecedores e clientes internos é utilizada para enfatizar o fato de que, para satisfazer o cliente final, todos dentro da empresa, têm um cliente que necessitam de satisfazer. Em última análise, o consumidor é sempre, o juiz da Qualidade (EUREKA & RAYAN, 1992).

Gestão dos Processos - Os produtos e serviços exigidos pelos clientes são produzidos e entregues, por elementos vindos das diferentes partes da empresa. Estas várias atividades individuais podem ser encaradas coletivamente, como um processo que poderá ser concebido no sentido da obtenção da satisfação total do cliente. As empresas de Qualidade Total, reconhecem que este processo empresarial básico necessita de ser concebido, orientado e gerido para ir ao encontro das expectativas do cliente, de modo lucrativo. Todas as barreiras colocadas à performance são, sistematicamente, identificadas, analisadas e derrubadas. Estas barreiras poderão apresentar-se em termos de divisões funcionais arbitrárias entre pessoas, burocracia ou, simplesmente, métodos de trabalho mal definidos (EUREKA & RAYAN, 1992).

Melhoria Contínua - A idéia de que a perfeição no contexto empresarial é impossível, e de que o único objetivo com significado é a luta contínua para melhorar algo que já foi atingido. O objetivo é fazer as coisas bem logo à primeira e, depois, começar a melhorá-las. Na empresa, estão empenhados em conseguir, diariamente, pequenos progressos de funcionamento. Isto significa que, com o tempo, muitos milhares de idéias para a melhoria, todas motivadas pelo objetivo da satisfação do cliente, serão incorporados no processo empresarial.

Envolvimento e Mobilização de Todos - As organizações de Qualidade Total reconhecem que as pessoas que estão, atualmente, desempenhando determinada função, se encontram em posição privilegiada para identificar o que está acontecendo, e como atingir, de forma mais eficientemente, as melhorias desejadas na performance da organização. Esta percepção leva, diretamente, à conclusão que toda a equipe deve ter a confiança, conhecimento e técnica, necessários para traduzir este seu potencial em ações. A mobilização implica a concessão de autoridade com base nos conhecimentos, técnica e formação, à pessoa que se encontra em melhor posição para proporcionar qualidade ao cliente (EUREKA & RAYAN, 1992).

Objetivo de Qualidade e Orientação do Progresso - A obtenção de Qualidade Total é um compromisso, a longo prazo, para qualquer empresa. O papel dos objetivos é atribuir significado, constituindo-se um desafio em termos de progresso, vital para o processo de

definição de uma nítida percepção das prioridades relativas à ação e para o reconhecimento do sucesso. O estabelecimento e a comunicação dos objetivos de melhoria de qualidade, ligados ao plano empresarial, proporcionam um foco de motivação poderoso que está aberto a todos, estabelece um mecanismo para a autogestão e uma plataforma para o reconhecimento do sucesso individual e em equipe (EUREKA & RAYAN, 1992).

O poder destes cinco fundamentos, enquanto potencialmente válidos, em si, para a organização, é desbloqueado através de uma liderança empenhada, que constitui a dinâmica transformadora destas idéias de senso comum, numa estratégia empresarial que aposta a nível mundial. A liderança, ou a falta dela é muitas vezes um dos maiores obstáculos à transformação para a qualidade. Só o total empenho e envolvimento de toda a organização, começando pelo topo da gestão até o menor nível hierárquico, pode garantir o sucesso da implementação da qualidade da empresa. Geralmente a atitude dos funcionários é determinada, pelo que eles pensam do nível de expectativa projetada pela gestão do trabalho deles. É então essencial que a gestão defina clara e precisa os objetivos em relação à qualidade, surgindo os seguintes princípios fundamentais da GQT:

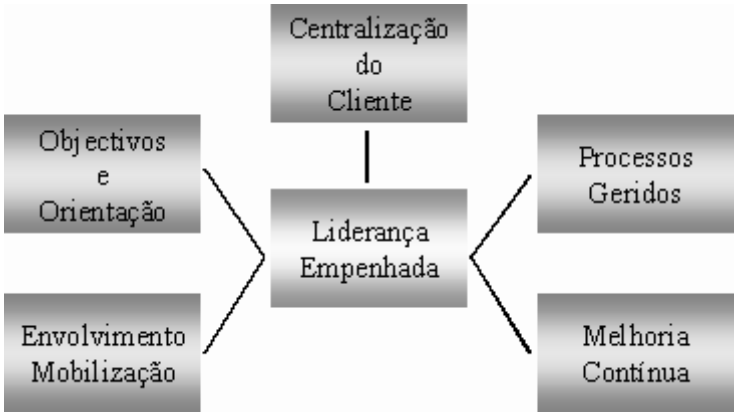


Figura 2.3 – Princípios Fundamentais de GQT
Fonte: Eureka & Rayan, 1992.

Feignbaum (1994) explica que deve se atentar para quatro princípios de gerenciamento da qualidade. O primeiro é que a qualidade não possui um nível constante, pois a competitividade e os consumidores sempre exigirão mais. O Segundo é a participação da liderança em conhecimento sobre qualidade, habilidade e atitudes positivas de cada membro da organização, ou seja, um gerenciamento satisfatório em vista ao envolvimento da equipe. Define no terceiro conceito que a qualidade é importante par se obter êxito em inovação. E por fim, ao contrário do que se imagina, a qualidade e custo se complementam e não conflitam com os objetivos comerciais.

De acordo com Eureka & Ryan (1992) o QFD, é um sistema para refletir os requisitos do Cliente em requisitos apropriados para a Organização, em todas as etapas da empresa, desde a Pesquisa e Desenvolvimento do Produto, Serviço, até a Engenharia, Fabricação, *Marketing*, Vendas e Distribuição

Segundo Cheng (1995) para que um produto atenda às necessidades dos clientes, ele precisa conter as características de qualidade projetada. Para tanto, o produto pode ser constituído de um, de alguns ou de muitos componentes; pode ser constituído de um único material de diversos materiais ou, ainda, de materiais alternativos; pode ser elaborado através de um único processo, de vários processos ou de processos alternativos. Cabe ainda projetar o produto de forma adequada para obter-se a qualidade exigida pelo cliente.

Cheng (1995) define que a aplicação do método QFD teve origem a partir do uso do diagrama de causa-e-efeito para definição de pontos de controle, e posteriormente, tabelas de garantia de qualidade, particularmente na produção.

Recentemente o QFD é utilizado como ferramenta inicial no ciclo de vida do desenvolvimento de produto e planejamento de produto dentro da empresa.

Desta forma pode-se apresentar o QFD como a melhor metodologia para traduzir a voz do cliente (subjetiva*)* em requisitos mensuráveis e (objetivos) que permearão e orientarão todas as fases do processo de desenvolvimento de produtos e serviços avalizando a satisfação do cliente.

Considera-se, desta forma, que o emprego do QFD - Desdobramento da Função Qualidade -, que representa como uma vicissitude viável na procura da qualidade para todos os empresários que almejam alcançar êxito por meio da satisfação total de seus clientes (GUAZZI, 1999).

A estrutura da QFD fez-se conhecida em todo mundo por sua sigla, que, em inglês, denomina *Quality Function Deployment, no* Brasil conhece-se como **Desdobramento da Função Qualidade**, mantendo-se, não obstante, o uso da sigla **QFD**, que é utilizada neste estudo.

O QFD, segundo Guazzi (1999), engloba a quebra dos conceitos tradicionais, transcendendo os padrões existentes, criando um terreno produtivo para o conceito de "*learning organizations*" (organizações que aprendem), sendo que cessa o raciocínio tradicional de resposta direta aos problemas que o ambiente traz, para um raciocínio superior

que procura a geração de idéias inovadoras aptas de pôr a organização à frente de seu mercado.

De acordo com Guazzi (1999), os principais conceitos formadores de sua base são:

a) Indagar aos clientes o que eles querem, na forma que eles usam para se expressar (voz do cliente), isto é, compreender como os clientes definem e percebem os produtos na perspectiva deles;

b) Usar toda a experiência e conhecimento da equipe multifuncional para identificar peculiaridades mensuráveis que irão de encontro às necessidades e desejos do cliente;

c) Priorizar e concentrar esforços nas peculiaridades mensuráveis, para que a voz do cliente seja conservada por todas as fases do desenvolvimento;

d) Admitir à área de *marketing*, "gerenciar" as expectativas do cliente e as ações dos concorrentes de tal forma que o produto ou serviço possa ser lucrativo por todo o seu ciclo de vida através da inovação constante.

O QFD é ainda empregado para desenvolver produtos para clientes externos à organização ou para produto mediador entre fornecedores e clientes internos (EUREKA & RAYAN, 1992).

Alcançadas através de apreciação de mercado, tendo-se as reais necessidades do cliente (aquelas explicitamente apregoadas em termos leigos) devem ser conhecidas antes que se possa dar início ao QFD. Sem o adequado entendimento da voz do cliente, o QFD pode tornar-se em ponto inexistente, buscar-se obter e entender a voz do cliente nem sempre é tarefa simples, pois, uma vez que se consiga, no entanto, fixa-se o panorama para uma aplicação bem-sucedida do QFD (EUREKA & RAYAN, 1992).

Deve-se buscar conhecer o que se quer saber de verdade é:

- Quem é o cliente?
- O que ele necessita?
- Por que ele necessita deste serviço?
- Como ele usa este serviço?
- Quando ele usa?
- Onde ele usa?

Segundo Guazzi (1999), o QFD - Desdobramento da Função Qualidade representa um sistema que traduz as necessidades dos clientes em requisitos técnicos adequados,

consentindo a introdução dos desejos nos produtos e serviços, sendo uma ferramenta que busca pôr os clientes numa posição de destaque.

Tal sistema é justaposto em cada estágio do ciclo de desenvolvimento do produto ou serviço, desde a pesquisa e o desenvolvimento até a engenharia, a produção, o *marketing*, as vendas e a distribuição.

Desta forma, profissionais em Qualidade, *Marketing*, Engenharia de Processo e Produto, obtém uma ferramenta ágil e comprovadamente eficaz que permite identificar e quantificar os vários requisitos que satisfarão os clientes, com a vantagem de reduzir os custos e aumentar a qualidade e a produtividade.

Segundo Eureka & Rayan (1992) o QFD representa uma sistemática de afiançar que o desenvolvimento de atributos, características e especificações do produto, assim como a seleção e o desenvolvimento de equipamentos, métodos e controles do processo sejam dirigidos para as demandas do cliente ou do mercado.

Para Eureka & Ryan (1992, p.3), o QFD: “É um caminho sistemático para avalizar que o desenvolvimento das características e especificações do produto, bem como o desenvolvimento de metodologias, processos e controles, sejam norteados pelas necessidades do consumidor”.

Ainda de acordo com Eureka e Rayan (1992), a base do QFD consiste em tornar o próprio processo um catalisador, que suscita esforço da equipe e cooperação; desse modo o QFD torna-se um mecanismo de comunicação entre as diversas áreas que trabalham no projeto.

Basicamente o QFD representa em diminuição de problemas no início da produção, menos mudanças no projeto, e encurta os ciclos de desenvolvimento do produto. Com isso, é conseqüente o aumento da produtividade e a redução de custos. A aplicação do QFD também ocasiona benefícios a longo prazo, tais como: satisfação dos clientes, custos de garantia baixos e ganhos de maiores fatias de mercado, dentre outros.

O QFD surgiu por meio da aplicação e desenvolvimento dos contemporâneos conceitos da gerência da qualidade no Japão. O progresso do controle da qualidade (qualidade reativa) para a garantia da qualidade (qualidade ativa), buscando incorporar ao produto ou serviço a qualidade que os clientes desejam em todas as fases de desenvolvimento.

Segundo Simões (2005) desde a sua criação, para ser aplicado no planejamento logístico da construção naval, o QFD firmou-se na indústria automobilística japonesa na

década de 70, empregado com extremo sucesso pela *Toyota*, e ingressou nos Estados Unidos, de maneira modesta no início dos anos 80, tendo o seu real desenvolvimento naquele país por volta dos anos 1985 e 1986, graças ao esforço dos três gigantes da indústria automobilística, *Chrysler*, *Ford* e *General Motors*, em *Detroit*, *Michigan*, onde a técnica japonesa foi ocidentalizada ao conhecimento tecnológico-gerencial daquelas grandes empresas do setor automotivo. Graças à extrema habilidade dos norte-americanos em difundir informações, o QFD espalhou-se por diversos segmentos: bancário, *software*, aeroespacial, eletro-eletrônico, *fast-food*, educação, médico, alimentos, etc.

Nos Estados Unidos, o QFD é adotado pelas maiores corporações industriais e de serviços. O QFD chegou ao Brasil há cerca de 3 a 5 anos, através de algumas iniciativas na indústria automobilística e na universidade. Hoje ele já está presente em aplicações nos setores de serviços de telecomunicações, siderurgia, cimento, automotivo, recursos humanos, elétrico-eletrônico, Educação, dentre outros.

Para que se faculte ter um perfeito entendimento do que vem a ser a metodologia do QFD, alguns conceitos chaves e métodos utilizados acabaram por sustentar e formar a metodologia. Tais conceitos distintos e, em determinadas ocasiões, complementares, foram desenvolvidos por Mizuno (1978), Feigenbaum (1994), Juran & Gryna (1991) e Akao (1990).

O QFD, segundo Akao (1990), representa uma metodologia que busca a conversão dos requisitos do consumidor em características de qualidade do produto ou serviço, por meio do desenvolvimento da qualidade de projeto, levando sistematicamente as relações entre os requisitos do consumidor e as características do produto ou serviço.

Segundo Guazzi (1999) a estrutura da garantia da qualidade do produto ou serviço é concretizada com atenção em quatro ênfases (qualidade, tecnologia, custos e confiabilidade), compreende-se o desdobramento para cada sistema e cada parte (figura 2.4). A qualidade total de um produto ou serviço será constituída por meio da rede de inter-relações.

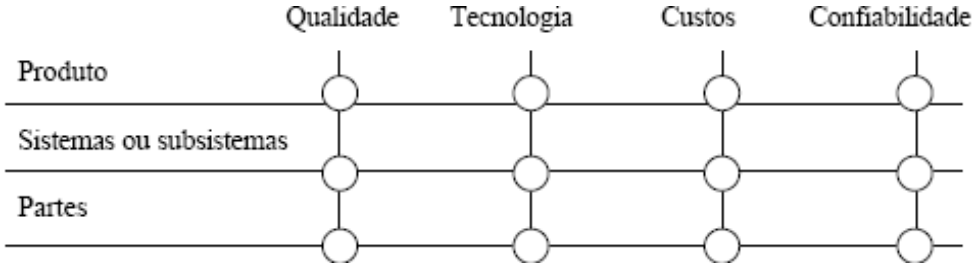


Figura 2.4 – A visão de Akao sobre a metodologia do QFD
Fonte: Guazzi (1999)

Eureka & Rayan (1992) apresenta o QFD, juntamente com outras ferramentas da qualidade, um sistema para a tradução dos requisitos do cliente em requisitos técnicos de fácil entendimento na empresa em cada uma das cinco fases do desenvolvimento de produtos (figura 2.5).

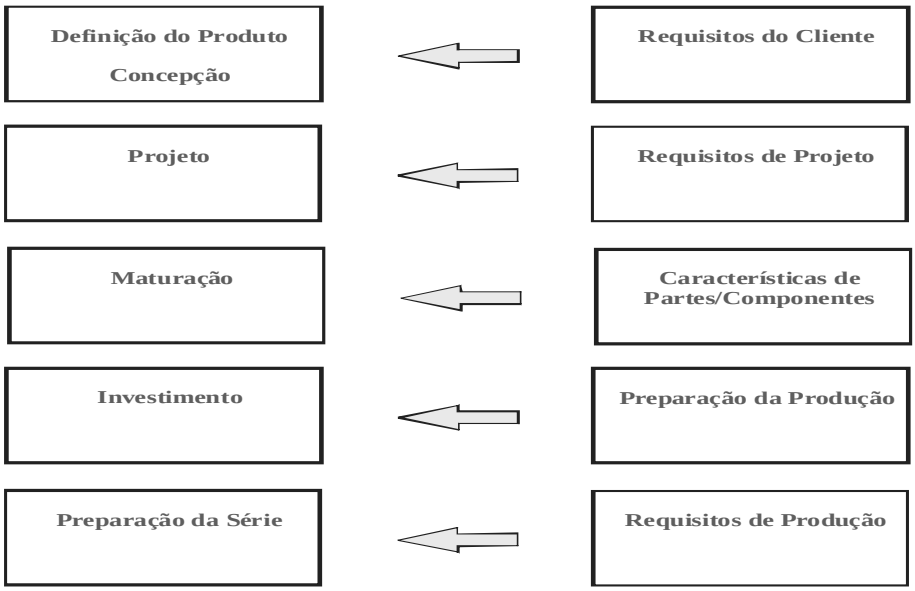


Figura 2.5 – Fases do desenvolvimento do produto, segundo Eureka
Fonte: Guazzi (1999)

De acordo com o *American Supplier Institute* - ASI (1989), não ocorre estrutura para o QFD, sendo de forma de um sistema para a tradução dos requisitos do cliente em requisitos adequados para a empresa em cada fase e finda-se que o ASI vê o QFD como um meio à orientação para o cliente (figura 2.6).

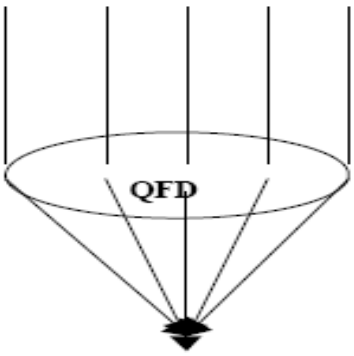


Figura 2.6 – O QFD como foco da satisfação dos clientes
Fonte: Kienitz (1995)

Para Hauser & Clausing (1988) o QFD representa um sistema coesivo de procedimentos de planejamento e comunicação, que possibilita a coordenação e submissão de

todos os talentos da empresa com o objetivo de desenvolver, produzir e vender produtos, que o cliente goste de adquirir hoje e no futuro. Para isso, o autor destaca o trabalho de equipe.

De acordo com Guazzi (1999, p.113):

> O QFD é um processo organizacional, que requer a conversão sistemática, passo a passo, e de maneira minuciosa dos requisitos do consumidor em características de qualidade do produto ou serviço, em tecnologia, em custos e na confiabilidade, com o envolvimento e integração de toda Função Qualidade através da utilização de rotinas técnicas e administrativas formalizadas, buscando a redução do ciclo de desenvolvimento de produtos e serviços, a qualidade do conceito de produto ou serviço e a garantia da qualidade do produto ou serviço em todas as fases do desenvolvimento.
> Com referência ao conhecimento articulado na literatura, utilizar-se-á a seguinte definição de QFD: O QFD é uma metodologia organizacional que atua em todas as fases do processo e consegue converter de forma sistemática os requisitos do cliente em características de qualidade (qualidade, custo, atendimento, moral e segurança), buscando a total satisfação dos clientes.
> O QFD admite que cada um na empresa saiba qual é a sua tarefa para satisfazer aos anseios e às exigências dos clientes, pois o seu enfoque é centrado totalmente na questão da satisfação total dos clientes.

Segundo Yukimura (apud Guazzi, 1999, p. 114), o QFD tem trazido diversas melhoras no projeto e desenvolvimento de novos produtos, tais como:

- Redução nas alterações de engenharia de 30% a 50%;
- Ciclo de projeto tem sido encurtado de 30% a 50%;
- Custos de início de operação têm redução de 20% a 60%;
- Redução de mais de 50% nas reclamações dentro da garantia do produto;
- Planejamento da garantia da qualidade mais estável;
- Favorecimento da comunicação entre os diferentes departamentos que atuam no desenvolvimento do produto, principalmente marketing e engenharia;
- Facilidade em traduzir os requisitos do consumidor;
- Facilidade na identificação das características que mais contribuem nos atributos da qualidade;
- Favorecimento do processo de balanceamento criterioso (trade-off) do projeto que afeta a função do produto para todos os consumidores;
- Melhor percepção de quais são as características e funções que receberão mais atenção;
- Melhor identificação das propriedades e das características de venda do produto.

Alcançando tais resultados, a empresa estará hábil a competir e manter-se no mundo globalizado, sendo que o QFD gerará grande produtividade, qualidade e lucratividade.

Não obstante, os melhores resultados incidindo da utilização do QFD são os de caráter organizacional, ou seja: enfoque da empresa voltada para o cliente, quebra de barreiras interdepartamentais e o incentivo ao trabalho em equipe.

2.3.2. Qualidade: função e sistema

O progresso da metodologia do QFD procede de conceitos tais como: Qualidade, Função Qualidade, Sistemas da Qualidade, Desdobramento da Qualidade e Desdobramento da Função Qualidade (sentido restrito). Esses conceitos progrediram e, juntos, formaram o conceito atual do QFD (GUAZZI, 1999).

O conceito de qualidade tem significados diferentes, sendo que a norma NBR ISO 8402 (1993) é muito ampla ao definir e elucidar os distintos aspectos da expressão e em seu interior o vocábulo é definida como sendo a "totalidade de características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas."

Juran & Gryna (1991) apresentou o conceito da Função Qualidade como representando toda área funcional da cadeia de valor (área de desenvolvimento, área de planejamento, ou do ciclo do produto, que compartilhar da formação da qualidade do produto. A Função Qualidade é o agente que impulsiona a ação de garantia da qualidade dentro do Sistema da Qualidade.

Para Juran (1997) a qualidade possui três seqüências universais na gerência da qualidade:

a) planejamento da qualidade;

b) controle da qualidade;

c) Melhoramento da qualidade.

O planejamento requer o estabelecimento de metas de qualidade, a identificação dos clientes, elencar as características do produto que atendam as necessidades dos clientes, e elaborar processos capazes de produzir tais características no produto. Estabelecer controles do processo; transferir os planos para as forças operacionais (JURAN, 1997).

No controle de qualidade é preciso avaliar o desempenho real, comparar o desempenho real com as metas de qualidade e agir sobre a diferença (JURAN, 1997).

Para atingir o melhoramento da qualidade é preciso provar a necessidade, estabelecer a infra-estrutura, identificar os projetos de melhoramento, estabelecer as equipes dos projetos, promover as equipes com recursos, treinamento e motivação para diagnosticar as causas e estimular a busca de soluções; estabelecer controles para manter os ganhos (JURAN, 1997).

Umeda (1995) descreve a qualidade sob três dimensões diferentes:

a) Aspecto de utilidade, através de sua função, performance e confiabilidade
b) Aspecto humano, através da segurança, operabilidade e sensibilidade; e
c) Aspecto social, através da preservação ambiental e da ética.

Whiteley (1992) já destacava a necessidade da empresa focar-se na qualidade conforme a necessidade do cliente, pois os indicadores de abandono da empresa pelos clientes estavam relacionados à qualidade do serviço mais do que a qualidade do produto. No entanto, com necessidades cada vez mais distintas e a competitividade, a qualidade do produto também passou a ser ressaltada, principalmente em se tratando da Engenharia de *Software*.

Godoy (1994) entende a característica da qualidade como a transformação dos desejos dos clientes em parâmetros mensuráveis. Em seu livro é distinguido o gerenciamento pelas diretrizes e o gerenciamento pela rotina. O primeiro é definido como "o desdobramento de cada fator crítico e sucesso nos itens indicadores de direção e respectivas metas" (GODOY, 1994, p.46). Seu principal objetivo é que todos os funcionários atinjam a "Visão da Empresa". Ao contrário do gerenciamento pelas diretrizes, o gerenciamento da rotina é o conjunto de ações que objetiva estabilizar o processo, definindo as responsabilidades sobre os resultados, a autoridade sobre meios e os itens de controle. Seu ciclo compreende a normalização, a operação normalizada do processo e ações corretivas nos resultados insatisfatórios as metas estabelecidas.

Miguel (2001) destaca duas dimensões fundamentais da QFD: o Desdobramento da Qualidade e o Desdobramento da Função. Apesar da literatura enfatizar mais o primeiro em seu sentido amplo, existe ainda o QFD em sentido restrito.

A Função Qualidade faz parte de um conjunto maior cognominado Sistema da Qualidade, que engloba todos os meios, atividades e responsabilidades pela realização da gestão da qualidade do produto.

2.3.2.1. Sistema da qualidade

O conceito de Sistema da Qualidade é apresentado por Feigenbaum (1994) como sendo a área de controle da qualidade adicionada à coleção de procedimentos imprescindíveis para fazer chegar às mãos do cliente, produtos dotados do padrão de qualidade desejado.

A Expressão qualidade tem significados diferentes. Inclusive a norma NBR ISO 8402 (1993) é muito ampla ao definir e elucidar os distintos aspectos da expressão e em seu interior

o vocábulo é definida como sendo a "totalidade de características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas."

2.3.2.2. Desdobramento da função qualidade: sentido restrito e sentido amplo

Cheng et al. (1995, p.24) define o QFD como,

> Uma forma de comunicar sistematicamente informação relacionada com a qualidade e de explicitar ordenadamente trabalho relacionado com a obtenção da qualidade; tem como objetivo alcançar o enfoque da garantia da qualidade durante o desenvolvimento de produto e é submetido em Desdobramento a Qualidade (QD) e Desdobramento da Função Qualidade no sentido restrito.

Esta definição abarca em si dois novos conceitos o de QD e de QDF Restrito. O primeiro recorre à lógica causa e efeito e o segundo é o desdobramento da função do trabalho, ou seja, de procedimentos gerenciais e técnicos. Os procedimentos do QDF restrito geram o Padrão Gerencial do Desenvolvimento de Produtos (PGDP) e o Plano de Atividades de Desenvolvimento (PADP), definidos pelas áreas funcionais da empresa. Seu desdobramento de forma ordenada exige o uso de uma ferramenta adequada. O que difere um método do outro é o processo seqüencial de desenvolvimento. Segundo Cheng et al (1995) existe uma dificuldade de compreender como cada função pode contribuir em cada etapa de desenvolvimento, principalmente da identificação das exigências do cliente até a inserção e formatação destas exigências no produto.

Para Ohfuji (1997) o Desdobramento da Qualidade tem essa finalidade, como ferramenta de desenvolvimento de novos produtos, como meio norteador entre o projeto e a fabricação, com garantia da qualidade.

Para entender melhor o QFD é preciso compreender os obstáculos surgidos entre as áreas funcionais que dificultam a obtenção dos resultados almejados e os seus benefícios. Para o planejamento da qualidade as conseqüências da ingerência do planejamento da qualidade e os benefícios do uso do QFD são apresentados na a figura 2.7.

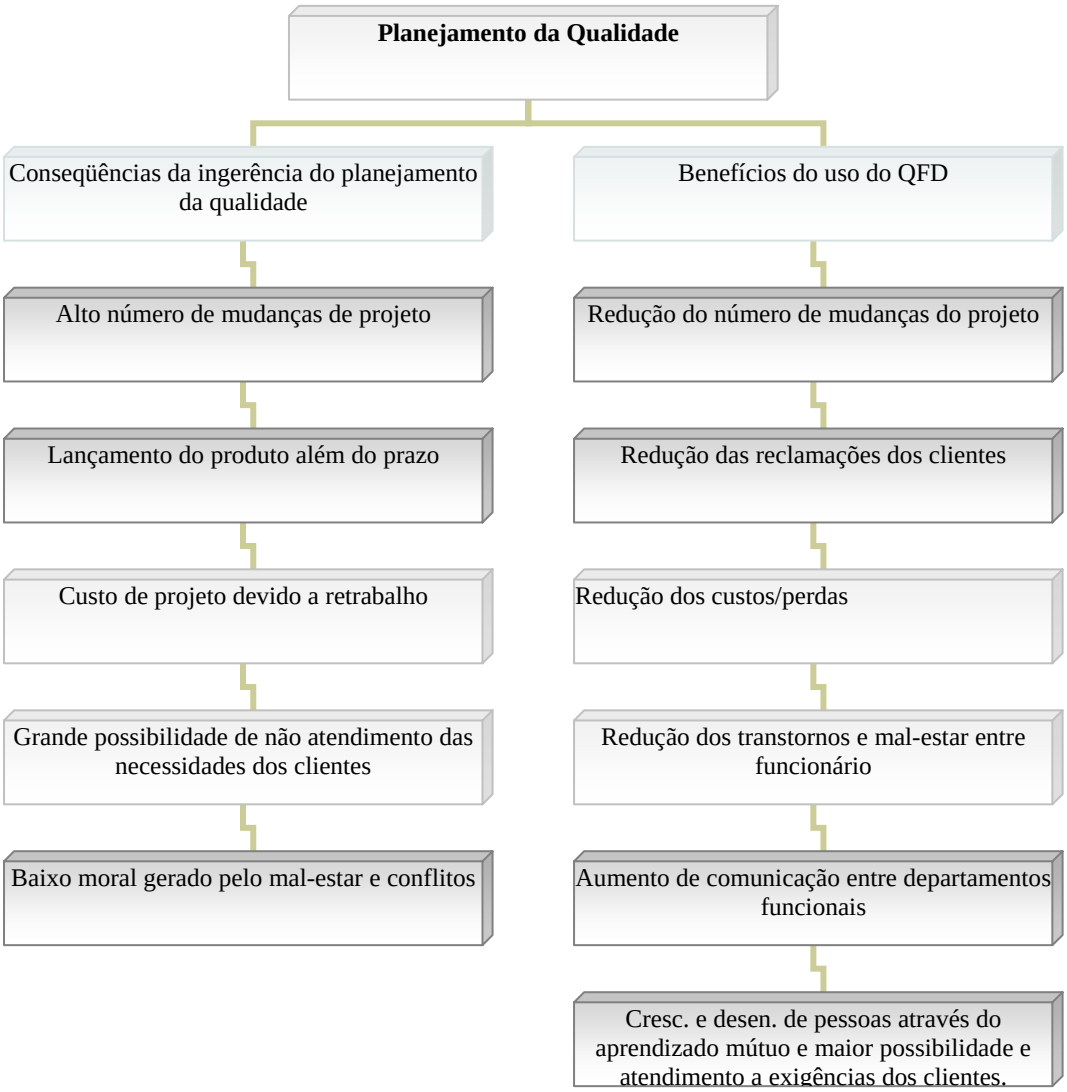


Figura 2.7 – Planejamento da Qualidade
Fonte: adaptado de Cheng et al, 1995, p.27.

A qualidade de um produto final, na concepção de Akao (1990), é função da qualidade de cada uma de suas partes, ou seja, a qualidade do produto é afiançada com a garantia da qualidade de cada parte (figura. 2.8).

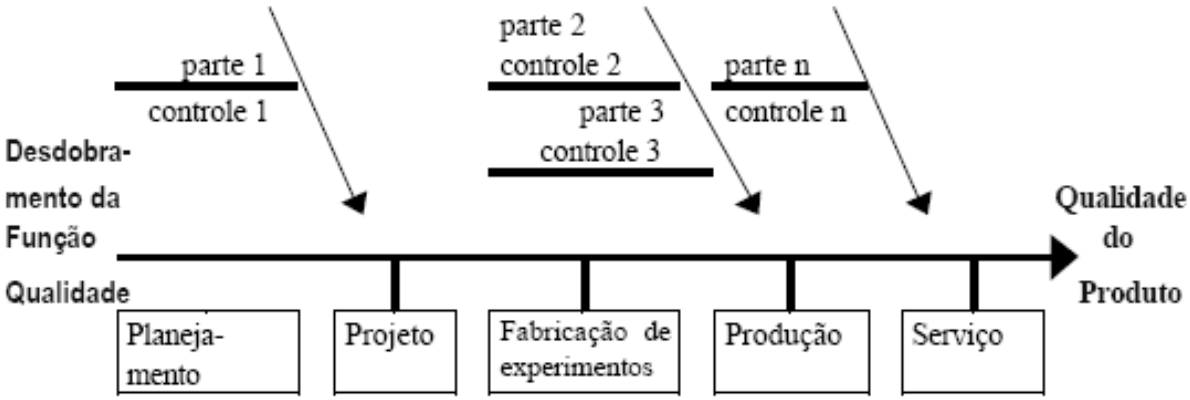


Figura 2.8 – Desdobramento da qualidade do produto
Fonte: Guazzi (1999)

Para Juran & Gryna (1991) o Desdobramento da Qualidade do Produto envolve as atividades particularizadas (controles da qualidade), executadas pelas várias áreas, que garantem a qualidade do produto.

Entende-se que o processo de garantia da qualidade do produto vem a ser o Desdobramento da Qualidade.

Akao (1994), finda a definição de Mizuno, delineando o conjunto formado pela Função Qualidade, definida por Juran & Gryna (1991), lado inferior da figura 2.9, e pelo Sistema da Qualidade como sendo, na prática, o Desdobramento da Função Qualidade (acepção restrita).

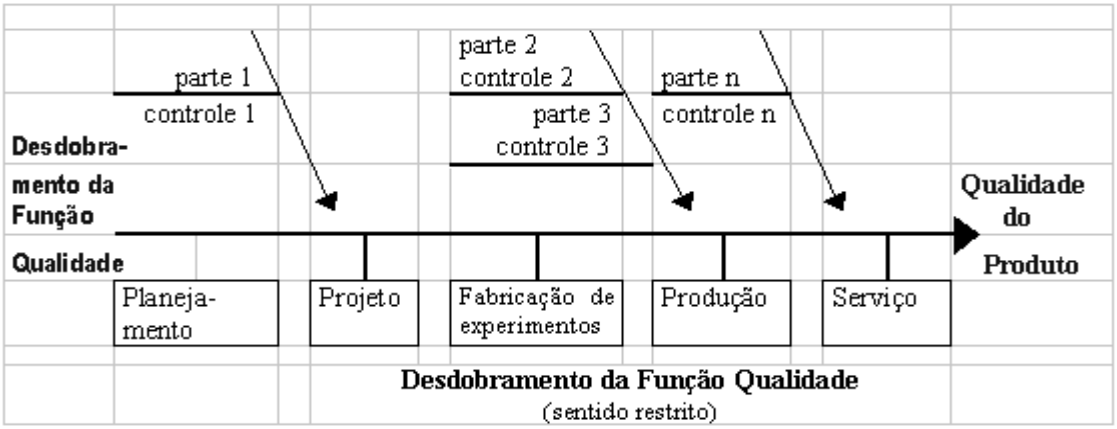


Figura 2.9 – Desdobramento da Função Qualidade no sentido restrito
Fonte: Guazzi (1999)

Ainda de acordo com Akao (1990), Desdobramento da Função Qualidade (acepção ampla) é a composição do Desdobramento da Qualidade do Produto e do Desdobramento da Função Qualidade (acepção restrita).

Akao (1990) enfoca que acontecem utilizações errôneas dos termos Desdobramento da Qualidade e Desdobramento da Função Qualidade (acepção restrita) como sinônimos. Conforme a descrição feita, os termos se referem a objetos diferentes.

A distinção dos conceitos-chave oferecida anteriormente e que contribuíram para o desenvolvimento do QFD, é relevante para o entendimento da metodologia e de sua definição formal.

2.3.3. Origem do QFD

Segundo Guazzi (1999) na conceituação japonesa, o nome da metodologia é escrito por seis caracteres da escrita *Kanji*, que o *American Supplier Institute* - ASI (1989) traduz como uma combinação importante e diversificada de sentidos e nuanças, que elucidam de forma prática e fácil os diversos aspectos e conteúdos do QFD (figura. 2.10)

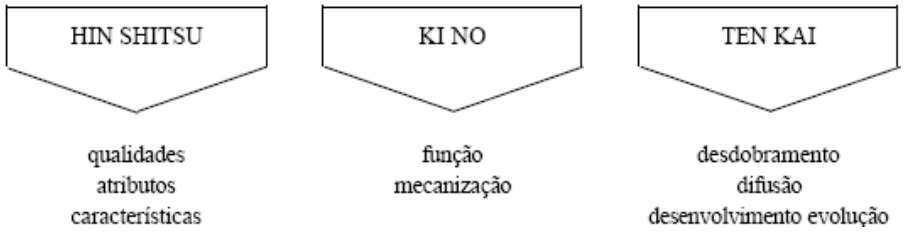


Figura 2.10 – QFD segundo a linguagem japonesa
Fonte: Guazzi (1999)

Segundo Guazzi (1999, p.118):

> Cada par de caracteres encena um conjunto de palavras ocidentais e, desta forma, o primeiro par significa: qualidade, características ou atributos. São palavras relacionadas à identificação e caracterização do produto.
>
> O segundo par de caracteres significa: função ou mecanização, e estão catalogados à forma de garantia da qualidade, ou seja, às atividades exercidas pelas diversas áreas ou funções de linha ou seus especialistas, com o objetivo de garantia da qualidade e o último par de caracteres significa: desdobramento, difusão, desenvolvimento ou evolução.
>
> Assim, este último par está catalogado à maneira de transmissão da informação pelas diversas áreas funcionais ou atividades executadas e dá sentido de movimento e dinamismo à metodologia.
>
> É relevante observar que a expressão **hin shitsu** é sinônimo de qualidade, no sentido de condição intrínseca, característica ou atributo, e não *qualidade* no sentido amplo, que pode ser aplicada tanto no singular como no plural.

O QFD apareceu na década de 60, no Japão, segundo Guazzi (1999.) no momento que a Mitsubishi buscou ao apoio do governo japonês para que fosse admissível desenvolver uma logística que permitisse a construção de navios-tanque e de superpetroleiros.

Uma questão que dizia respeito à visão estratégica de industrialização do Japão, segundo Guazzi (1999, p.118), o governo solicitou aos professores universitários das suas melhores faculdades para que criassem um sistema que assegurasse que cada etapa do processo de construção estivesse efetivamente ligada a uma particular exigência do cliente.

Segundo Guazi (1999, p.118-119):

> A partir daí que, através do comando dos professores Shigeru Mizuno e Yoji Akao nasceu o QFD, que por vários anos ficou versado como a *voz do cliente*.
> Nos Estados Unidos, o QFD só iniciou a ser aplicado no começo da década de 80, quando uma delegação japonesa comandada pelo professor e consultor Kaoru Ishikawa a levou para a Ford.
> No encadeamento, foi o educador Yoji Akao, quem conduziu um seminário em Chicago para os empresários, sobretudo do setor automobilístico. Em 1983, a revista Quality Progress divulgou um artigo atrelando o QFD à Gestão da Qualidade Total (TQC). Os americanos notaram que os fabricantes japoneses de automóveis estavam conseguindo captar a voz dos seus clientes, fazendo com que as suas preferências chegassem ligeiramente até aos processos de engenharia e manufatura.

O QFD foi empregado na Ford americana com o auxílio do professor Don Clausing e, a partir desta data, diversos artigos foram publicados, analisando a filosofia e os mecanismos do desdobramento e, em 1987, o segundo livro de Akao é editado, sendo traduzido para a língua inglesa. Ainda no ano de 1987, o consultor Robert King edita um livro de título "*Better designs in half the time - Implementing QFD in America"* e, a partir de então, a metodologia teve ampla publicação e principiou a ser empregada em uma enorme parte das organizações dos Estados Unidos (KIENITZ, 1995).

2.3.3.1. Estudo da abordagem do QFD das Quatro Fases.

De acordo com Guazzi (1999, p.119):

> As quatro fases ou quatro matrizes foram desenvolvidas primeiramente pelo engenheiro de confiabilidade Macabe. É a explanação mais conhecida tem como seguidores Sullivan (1986), Hauser e Clausing (1988) e o ASI (1989). A explanação das quatro fases se tornou conhecida nos EUA a partir do artigo "The house of quality", publicado na Harvard Business Review, em junho de 1988, de autoria de Hauser e Clausing (1988). Eles delineam o QFD como sendo executado em quatro fases, que se constituem de quatro matrizes encadeadas.

As quatro fases são (figura 2.11):

- Planejamento do Produto;
- Desdobramento das Partes;
- Planejamento do Processo;
- Planejamento da Produção.

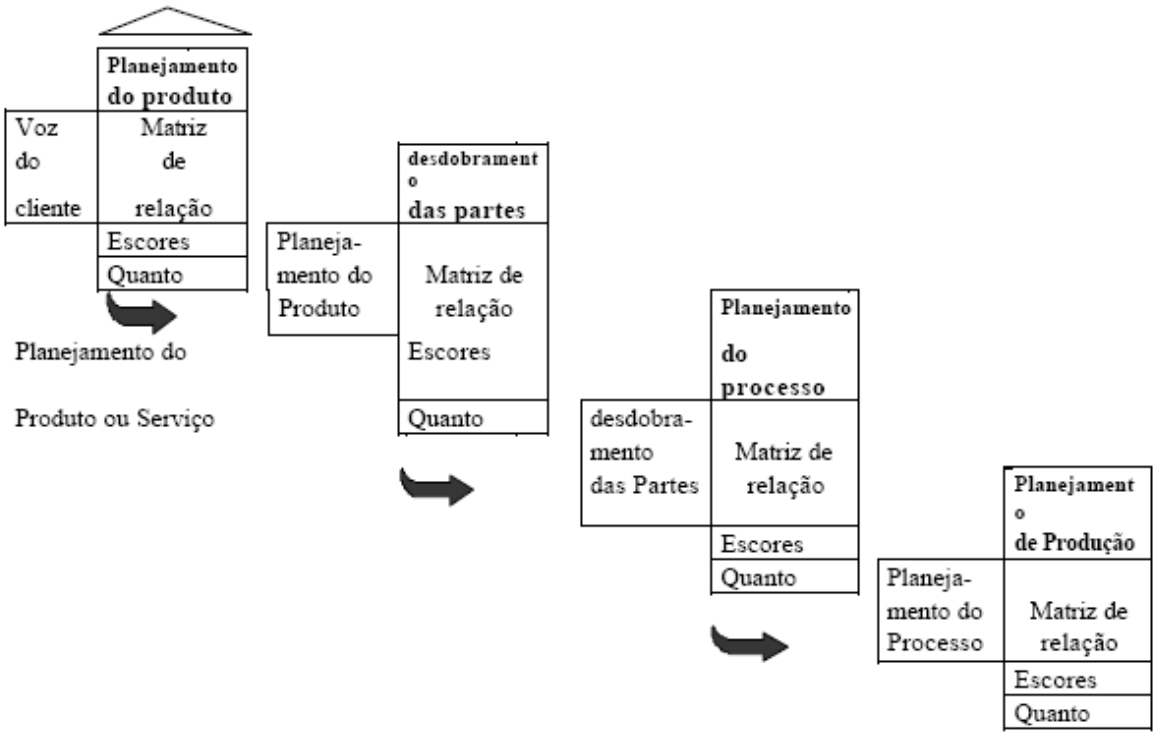
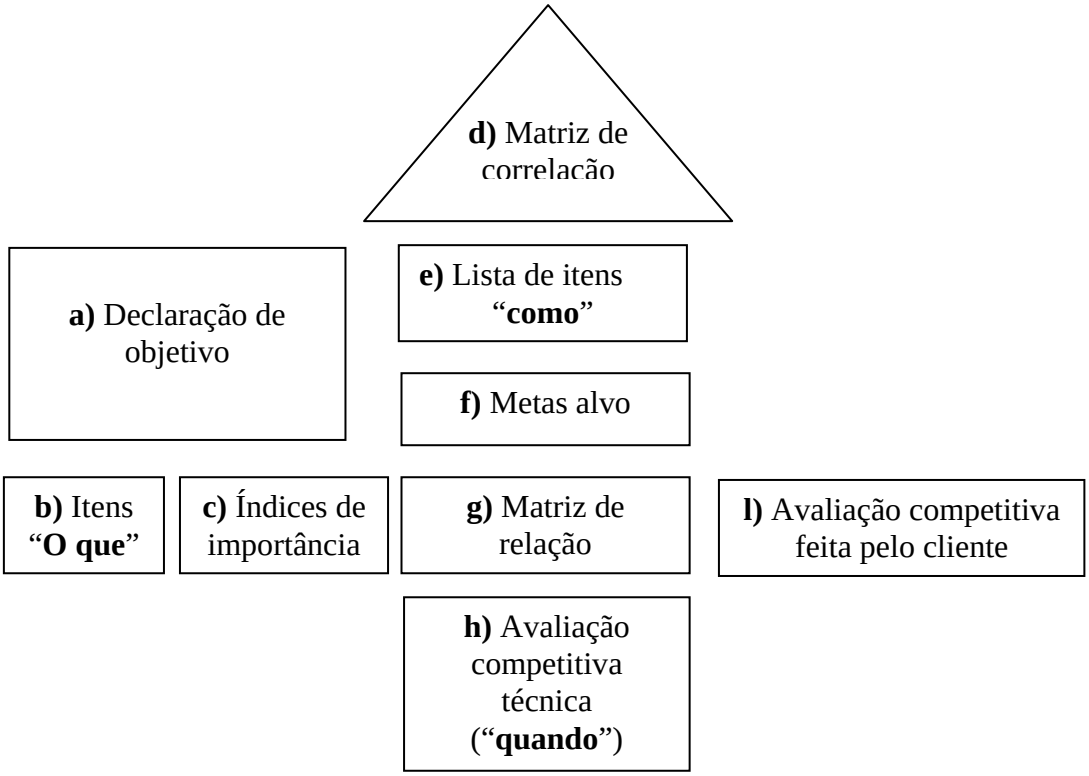


Figura 2.11 – Conexão das matrizes do QFD das quatro fases
Fonte: Guazzi (1999).

Segundo Guazzi (1999) as duas primeiras fases dizem respeito ao planejamento e projeto do produto ou serviço. As fases três e quatro dizem respeito ao planejamento do processo e das atividades de controle de qualidade. A Casa da Qualidade é a essência da primeira fase do QFD de quatro fases.

As matrizes do QFD podem ser projetadas de acordo com a figura 2.12



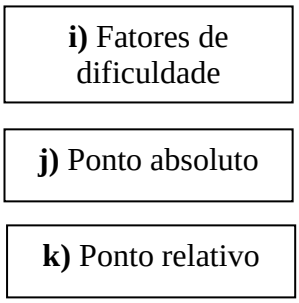


Figura 2.12 – Esquema do QFD
Fonte: Guazzi (1999).

a) Declaração de objetivo é a definição da meta, do objetivo, do problema que se quer deliberar ou para o qual se vai direcionar os esforços da equipe (GUAZZI, 1999).

O passo inicial para se desenvolver a metodologia QFD consiste em deliberar a declaração de objetivo, que pode aparecer na forma de uma pergunta que a empresa está tentando responder.

De forma criativa, segundo Guazzi (1999), os grupos envolvidos com o QFD necessitam atentar-se à relevância deste trabalho inicial, utilizando-se do tempo que for necessário para desenvolver uma declaração de objetivo real e correta.

De acordo com Guazi (1999) a metodologia QFD enfoca também a criação de estratégias a longo prazo para as empresas, podendo contribuir nas etapas do planejamento estratégico, além de permitir a identificação do tipo de pessoas que serão necessárias para poder atender às suas necessidades que surgirão no futuro.

b) Lista de itens “o que” (ou grupo de foco do cliente) é a descrição explícita e precisa das peculiaridades de um produto, de um processo ou de um serviço. O que o cliente deseja do serviço, ou seja, que características o serviço necessita ter para que seja bem aceito pela empresa? (GUAZZI, 1999).

Para Guazzi (1999) a necessidade de prestar um atendimento diferenciado ao cliente é uma realidade que aumenta a cada dia. Para tanto, é necessário entender suas expectativas, e isso só é alcançado se perguntar a ele. O grupo de foco do cliente é um artifício empregado para capturar a voz do cliente e ele representa uma amostra de clientes que usaria um determinado produto ou serviço e as qualidades que pareceram relevantes para o QFD tornar-se-ão os itens “o que" da matriz QFD.

A equipe de indivíduos que participarão das reuniões do grupo de foco do cliente deve ser composta por representantes dos departamentos envolvidos com a declaração de objetivo. É relevante que a equipe de QFD represente todas as funções existentes na empresa, compondo assim uma equipe multidisciplinar (GUAZZI, 1999).

c) Índices de relevância são pesos (ou valores) atribuídos a cada um dos "o que" o cliente quer.

Já na primeira sessão do QFD, escuta-se a voz do cliente. Seus requisitos, atributos ou solicitações são classificados como qualidades desejadas pelo cliente e estes constituem os itens chamados "o que", isto é, as características particulares do produto ou do serviço que o cliente quer. É bastante relevante capturar todos os “o que", devendo-se estar seguro que cada um deles representa uma reivindicação simples. Todos os itens "o que" são importantes para o cliente, não obstante, o QFD permite identificar quais são os mais relevantes, utilizando um método sistemático de ponderação (GUAZZI, 1999).

Os japoneses de acordo com Guazzi (1999) empregaram uma escala com símbolos representando os valores 1, 3 e 9, para identificar os pesos dos índices de satisfação do cliente, conforme demonstrado na figura 2.13.

		Contribuição
△	= 1	fraca
○	= 3	média ou moderada
◉	= 9	forte

Figura 2.13 – Peso dos índices de satisfação do cliente
Fonte: Eureka & Rayan (1992).

Para os índices de relevância da necessidade do cliente, habitua-se usar uma escala de 0 a 5, de forma a refletir a importância relativa desse item para o cliente, onde cada um desses valores será, posteriormente, multiplicado pelos pesos atribuídos para cada símbolo da matriz (△, ○ e ◉) representando a relação entre o requisito do produto ou serviço e a necessidade do cliente.

d) Matriz de correlação triangular é uma matriz onde se demonstra a analogia ou a dependência que existe entre os vários "como" entre si.

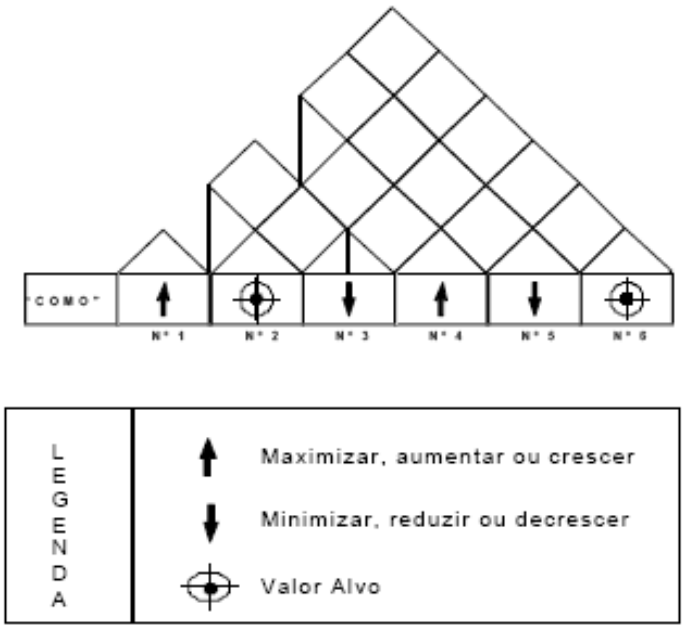


Figura 2.14 – Matriz de conexão triangular
Fonte: Guazzi (1999).

De acordo com Guazzi (1999) a matriz de conexão ou o telhado da Casa da Qualidade tem a forma de um triângulo, conforme figura 2.15. Tem caselas que representam as intersecções dos itens "como", que podem existir correlação positiva, negativa ou não serem correlacionadas. Por meio da matriz de correlação é admissível determinar quais itens "como" apóiam um ao outro e quais são conflitantes entre si.

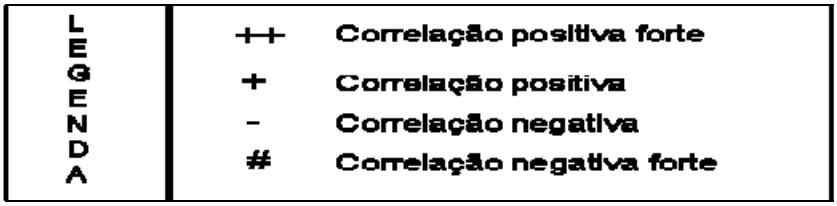


Figura 2.15 – Telhado da Casa da Qualidade
Fonte: Guazzi (1999).

e) Como atender o que o cliente quer? (lista de itens “como”) - Deve-se alterar as necessidades dos clientes, geralmente abstratas, em requisitos comensuráveis de projeto (GUAZZI, 1999).

Deste ponto em diante, segundo Guazzi (1999), a equipe do QFD facultará concentrar seus esforços para a solução dos problemas diagnosticados. É por meio da lista dos itens "como" que a empresa poderá atender às solicitações da lista dos itens "o que".

Os itens "como" incidirão de processos, métodos, facilidades, departamentos, funções da organização, etc.

3. MODELO PROPOSTO

Este capítulo foi dividido em duas etapas, na primeira é apresentado o modelo proposto para adaptar o QFD ao processo de desenvolvimento de *software*, na etapa seguinte é realizada uma simulação da aplicação do modelo proposto.

O modelo proposto é fundamentado no QFD das Quatro fases, onde importantes adequações foram realizadas visando tornar o modelo funcional e facilitar seu emprego, como também ajustá-lo para considerar algumas particularidades do processo de elaboração e desenvolvimento de *software*. As quatro fases são, conforme Sullivan (1986 *apud* FERNANDES, REBELATO, 2006) (figura 3.1):

1) **Planejamento do produto** - que transforma ou traduz "voz do cliente" em requisitos do produto;
2) **Desdobramento das partes** – que transforma as características do produto em requisitos dos componentes;
3) **Planejamento do processo** – que transforma as características dos componentes em requisitos do processo;
4) **Planejamento da produção** – que transforma as características do processo em requisitos da produção.

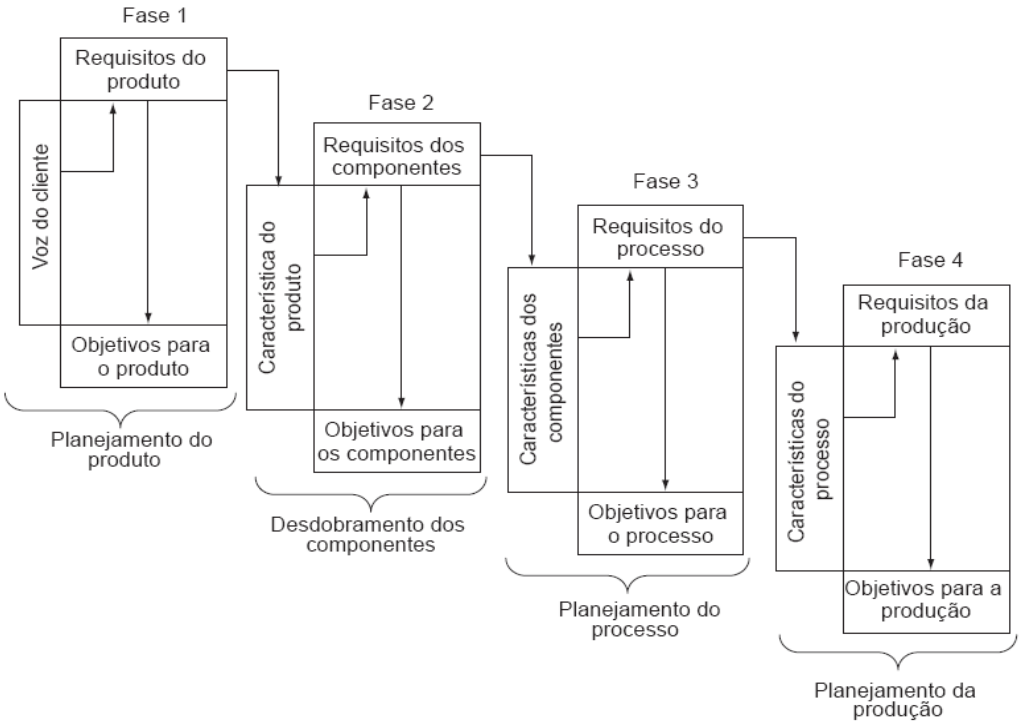


Figura 3.1 – Abordagem "das quatro fases" para desenvolvimento de QFD.
Fonte: Fernandes & Rebelato, 2006, p.247.

3.1. Apresentação do Modelo Proposto

3.1.1. Fase 1 – Planejamento do Produto ou "Casa da Qualidade"

3.1.1.1. Levantamento dos requisitos do cliente

- Identificar o Cliente

  O primeiro passo é a identificação dos usuários que participarão do processo de levantamento de requisitos. Para o sucesso desta atividade, e consequentemente do projeto, é de fundamental importância que esta equipe seja multifuncional, com representantes de todas as áreas envolvidas e ou afetadas com a utilização do *software*. Deve-se envolver todos que de alguma forma poderão colaborar com o projeto, abrangendo o cliente e a equipe de desenvolvimento.

- Identificar os requisitos do cliente ("o que")

  Após a identificação dos clientes, são coletadas as suas expectativas, exigências e necessidades, ou seja, entender o que o cliente necessita e espera do produto que será desenvolvido, neste caso o software.

- Analisar e Negociar os Requisitos

  Análise dos requisitos buscando identificar inconsistências e ou ambigüidades.

- Documentar os requisitos

  Nesta etapa os requisitos são documentados, em um nível adequado de detalhamento, de maneira que todos os envolvidos tenham entendimento do seu conteúdo, tendo em vista que este documento poderá ser consultado em todas as fases subseqüentes. O documento de requisitos deve traduzir de maneira clara, simples e objetiva os desejos do cliente.

- Validar os Requisitos

  Após documentados, os requisitos são validados, visando garantir que a especificação reproduz o que o cliente espera do sistema, incluindo as funcionalidades e restrições determinadas por ele.

3.1.1.2. Requisitos de Projeto

Baseado nos requisitos do cliente inicia-se o desenvolvimento da matriz de Planejamento do Produto ou "Casa da Qualidade".

- Índices de Relevância

Gradua os requisitos do cliente, em uma escala com valores de 1 a 5. É atribuída pelo cliente, uma nota para cada item “o que”, possibilitando classificar o grau de importância relativa daquele requisito para o cliente.

- Identificar Requisitos de Projeto (“como”)

  De forma análoga a fase de Identificação dos Requisitos, a equipe desta etapa deve ser multidisciplinar, com representatividade de todas as áreas envolvidas com o processo de desenvolvimento do software.

Agora para cada item “o que” é definida pelo menos uma forma para atender o que o requisito do cliente, que se tornarão a lista de itens “como”.

- Matriz de Relação

  Esta matriz estabelece a relação entre os requisitos do cliente (linha) e os requisitos de projeto (coluna), ou seja, na percepção do cliente como cada item “como” contribui com o atendimento do item “o que”.

- Peso Absoluto

  Agora é possível realizar o cálculo do peso absoluto, em função do Índice de Relevância e da Matriz de Relacionamentos, determinando-se assim quais os requisitos de projeto que possuem mais relação com o alcance da qualidade especificado pelo cliente, permitindo priorizar a função de trabalho com as necessidades a serem desenvolvidas.

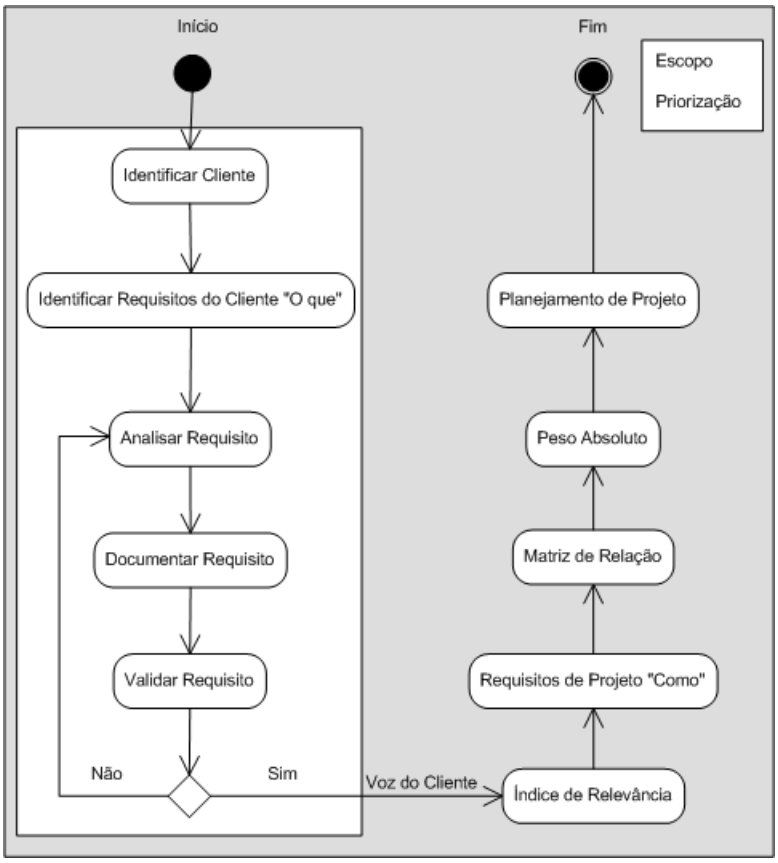


Figura 3.2 – Atividades modelo proposto para fase 1
Fonte: o autor

3.1.2. Fase 2 – Desdobramento das Partes ou Projeto do Produto

Os itens "como" que foram detalhados, priorizados e quantificados na fase anterior se tornam os itens "o que" desta fase.

- Desdobramento das Partes

  Os módulos do software serão desdobrados em partes menores, tais como funcionalidades, janelas, transações e formas de armazenamento dos dados, detalhando como os mesmo serão compostos. As janelas representam a interface com os usuários, e podem ser de entrada (exemplo tela de cadastro de usuários) ou saída (exemplo relatório ou tela de consulta). As transações são responsáveis por manipular (inserir, alterar e excluir) e consultar os dados armazenados.

- Documentar Requisito de Sistema – Casos de Uso

  No modelo proposto os requisitos de sistema são modelados e documentados por meio de modelo de casos de uso. Os modelos de casos de uso descrevem as funcionalidades ou serviços que devem ser suportados pelo software, representadas por uma sequência de mensagens trocadas entre o sistema e os atores. Atores representam as entidades (pessoas, outro software, hardware, etc.) que interagem com

o sistema durante sua utilização. O modelo de caso de uso é composto pela especificação do caso de uso e diagrama de casos de uso. Um caso de uso representa um recurso ou comportamento de um sistema, ou seja, um resultado que o software produz para a um ator. Um diagrama de caso de uso descreve graficamente o caso de uso, identificando também as relações entre eles.

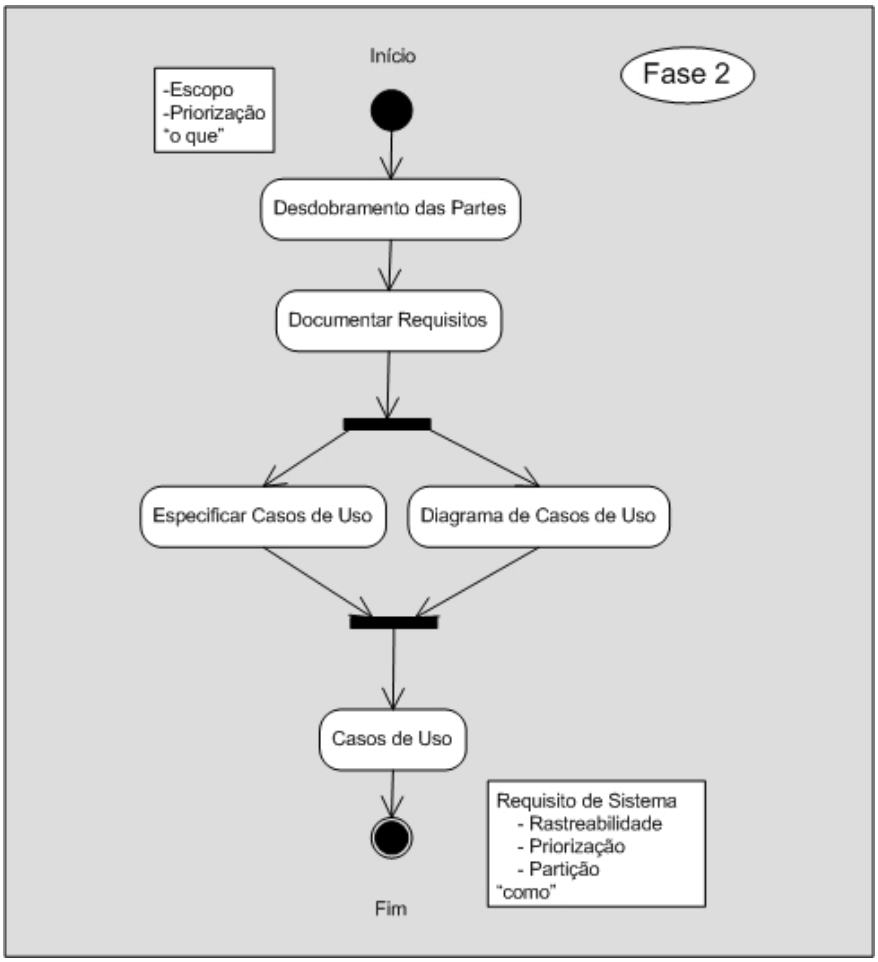


Figura 3.3 – Atividades modelo proposto para fase 2
Fonte: o autor

3.1.3. Fase 3 – Planejamento do Processo

No modelo proposto para o processo de desenvolvimento do *software*, nesta fase são definidas as técnicas, ferramentas e recursos que melhor atenderão os requisitos do produto, que foram especificados pelo cliente na primeira fase.

- Determinar a arquitetura (duas ou três camadas) adequada para o sistema;
- Definir a linguagem de programação para o desenvolvimento do *software*, que deve ser aderente a arquitetura escolhida, ao ambiente (sistema operacional e *hardware*) do cliente e ao modelo de dados, devem ser considerados também os custos de aquisição e treinamento;

- Estabelecer o meio e a forma de armazenamento de dados, considerando a linguagem e arquitetura selecionadas.
- Recursos inerentes a execução do processo.
    - Analista de sistemas;
    - Programador;
    - Microcomputador;
    - Impressora.

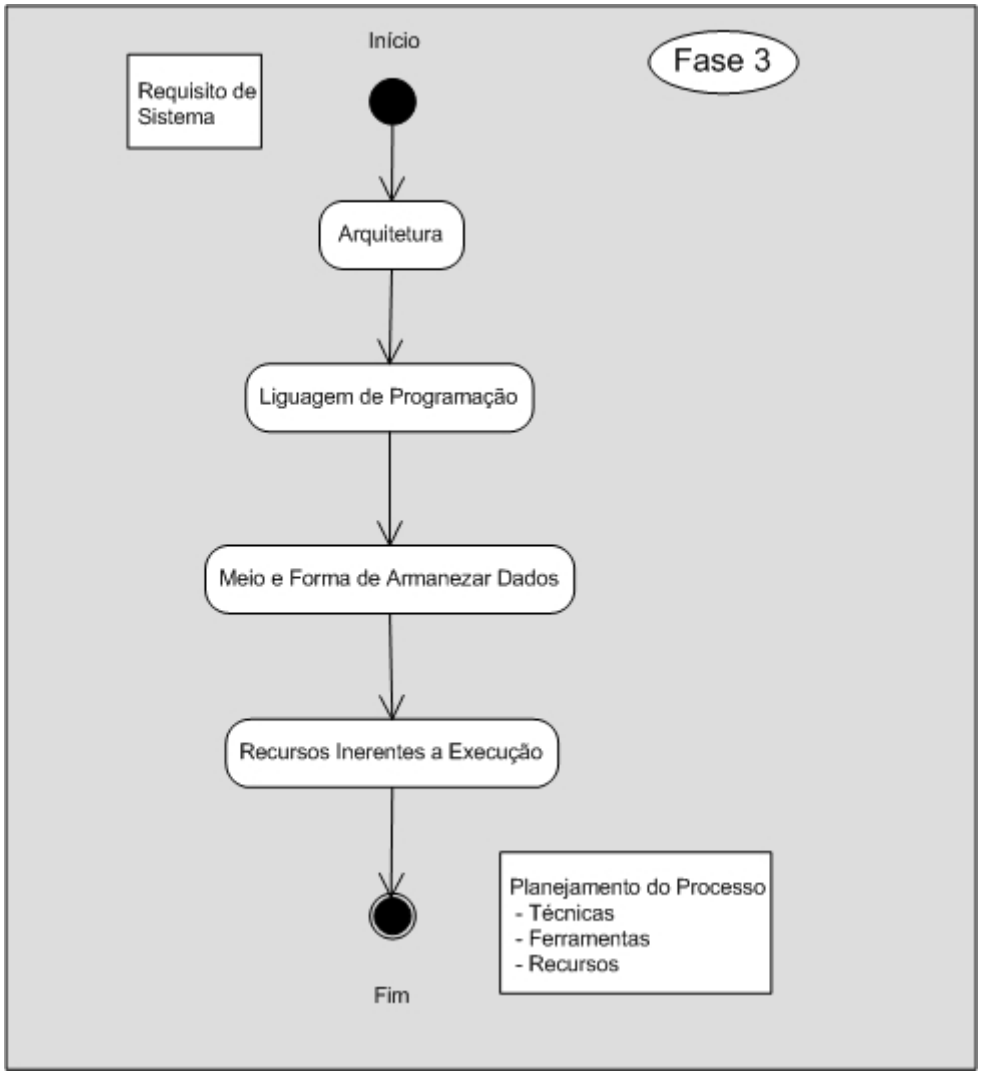


Figura 3.4 – Atividades modelo proposto para fase 3.
Fonte: o autor

3.1.4. Fase 4 – Planej. do Controle do Processo ou "Planejamento da Produção"

Na fase quatro são desenvolvidas as exigências de produção, como também o controle e acompanhamento da mesma, objetivando assegurar que o *software* será desenvolvido em

conformidade com as exigências do cliente. No modelo esta fase recebe como *input* o escopo do *software* e o Planejamento do Processo (técnicas, ferramentas e recursos).

- Definição das atividades:
  - A partir do detalhamento do escopo são definidas e descritas todas as atividades que serão realizadas durante o desenvolvimento do *software*.
- Seqüenciamento das atividades:
  - A partir da lista de atividades, são identificadas e documentadas as relações e dependências entre as atividades, tendo como base o conhecimento dos especialistas da equipe. Durante esta etapa também são definidos os Marcos (*Milestones*), que servirão como pontos de checagem para o monitoramento do cronograma.
- Estimar duração das atividades:
  - Consiste em estimar o esforço (humano) necessário para realização de cada atividade, que neste caso será realizado através da avaliação dos especialistas membros do time.
- Delimitar os recursos:
  - A partir da lista de atividades e do Planejamento do Processo (Fase 3), serão definidos quais os recursos (recursos humanos, materiais e equipamentos) e quais suas respectivas quantidades, são necessários para realização das atividades do projeto. Para os recursos humanos são considerados o perfil e conhecimento necessários para execução da atividade.
    - Analista;
    - Programador;
    - Administrador de banco de dados.
    - Linguagem de programação;
    - Banco de dados;
    - Computadores;
    - Impressoras.

- Estimar custo:
  - Com base nos recursos necessários e duração das atividades é possível estimar o custo de desenvolvimento do *software*.

- Determinar cronograma:
    - Agora serão definidas as datas de início e término das atividades, considerando o seqüenciamento, duração e os recursos necessários para realização de cada atividade. A partir do cronograma é possível planejar os entregáveis, ou seja, as datas das entregas dos módulos do *software*.

Após realizar o Planejamento da Produção, é iniciado o desenvolvimento do *software*, sendo também iniciadas as fases de monitoramento e controle, visando garantir a conversão das especificações (documentos de requisitos, planejamento do processo, etc.) no produto final (*software*):

- Monitoramento e controle dos prazos;
    - Asseverar que o *software* seja entregue dentro do prazo previsto.

- Monitoramento e controle dos custos;
    - Garantir o desenvolvimento do *software* dentro do orçamento previsto.

- Monitoramento e controle do escopo;
    - Assegurar que o *software* contemple tudo que foi especificado, e tão somente o que foi especificado.

- Controle da qualidade;
    - No modelo proposto a qualidade terá relação direta com prazo, custo e escopo, ou seja, a qualidade será alcançada se o *software* for entregue na data acertada, contenha os requisitos que foram especificados e seja desenvolvido dentro do custo previsto.

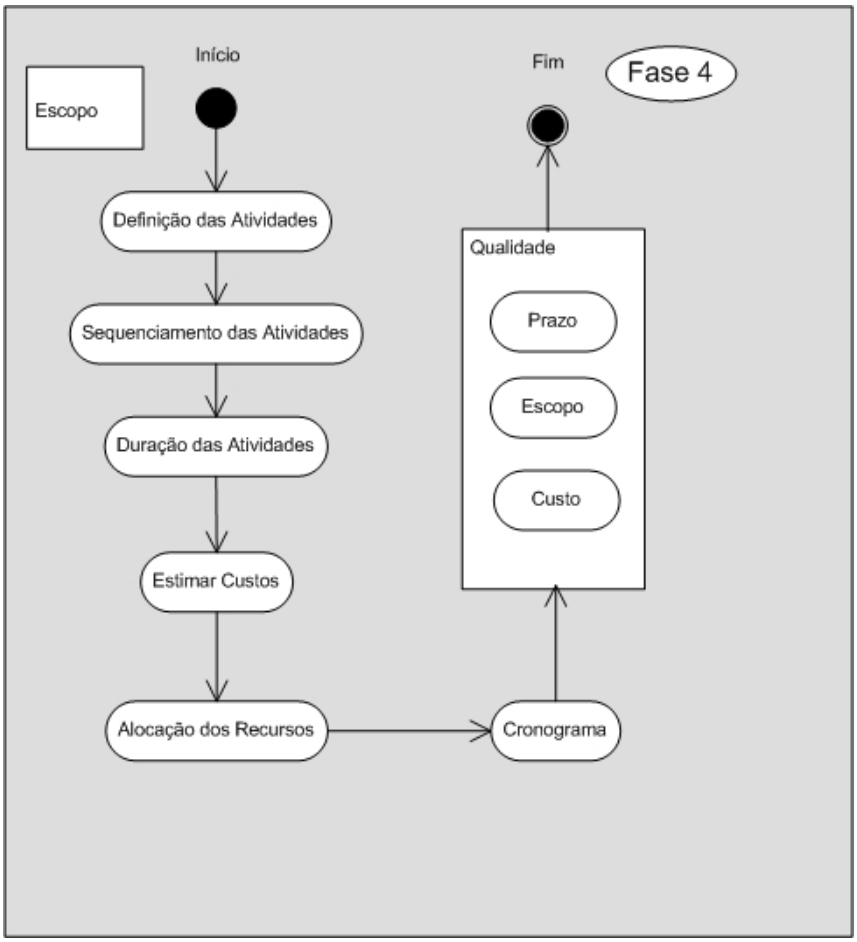


Figura 3.5 – Atividades modelo proposto para fase 4.
Fonte: o autor

O objetivo de utilizar esse modelo conceitual não foi apenas desenvolver o *software* de acordo com as exigências dos clientes, mas também criar condições para garantir a qualidade, desde a etapa de levantamento das necessidades e expectativas dos usuários, até a fase de desenvolvimento do produto final, visando assim garantir que o *software* produzido tem a qualidade exigida pelo cliente.

3.2. Simulação da Aplicação do Modelo Proposto

Para simular a aplicação do modelo proposto optou-se por utilizar o desenvolvimento de um sistema de gestão hospitalar, aqui denominado de Hosptools. O sistema será elaborado e desenvolvido pela equipe da PE-Softools, com exclusividade, para o Grupo Joaquim Camerino (GJC), rede hospitalar constituída por três hospitais e sete centros clínicos, distribuídos nos estados da Paraíba, Pernambuco e Rio Grande do Norte.

O Sistema Integrado de Gestão Hospitalar - Hosptools será composto dos seguintes módulos:

Receptools..... Recepção e admissão de pacientes;

Ambtools.....Agendamento e atendimento ambulatorial;

Diagtools..... Agendamento e atendimento de dignóstico;

Labtools.....Controle e atendimento laboratorial;

Estools..... Cadastro de produto, controle de estoque e compras;

Fatools.....Controle do consumo e faturamento;

Blotools.....Controle do bloco cirúrgico;

Relatools.....Geração de relatórios.

Tendo em vista que o Hosptools será um sistema construído em estrutura modular, para delimitar a aplicação do modelo proposto, será simulado apenas o processo de desenvolvimento de algumas funcionalidades do Receptools. Este módulo do sistema será utilizado para suportar o processo de recepção, admissão e encaminhamento dos pacientes aos demais setores envolvidos com o fluxo do atendimento hospitalar.

3.2.1. Simulação da fase 1 – Planejamento do Produto ou “Casa da Qualidade”

Esta fase foi iniciada com uma série de visitas ao Hospital Joaquim Camerino, já que esta será a unidade “piloto” para desenvolvimento e implantação do Sistema Integrado de Gestão Hospitalar - Hosptools no GJC. O objetivo desta etapa foi conhecer e documentar como o processo de recepção e admissão dos pacientes é realizado atualmente, e também identificar e selecionar os usuários que participarão do levantamento de requisitos.

A partir dos dados coletados, foi possível realizar o mapeamento e definir os profissionais que participarão do levantamento de requisitos, conforme abaixo:

- GJC
    - Administradores do Hospital;
    - Recepcionistas;
    - Enfermeiros;
    - Auxiliares de enfermagem;
    - Auxiliares administrativo;
    - Técnicos de informática;
    - Administradores de banco de dados;
    - Analistas de suporte.
- PE-Softools
    - Gerente de projetos;
    - Analistas de regras de negócios;
    - Analistas de sistema;
    - Programadores;
    - Arquitetos de sistema;
    - Administradores de bancos de dados.

Após definida a equipe do projeto, foi realizado o processo de levantamento de requisitos do cliente. A figura 3.6 demonstra os principais requisitos identificados durante esta etapa.

REQUISITOS DO CLIENTE - "O QUE"		
IDENTIFICAÇÃO	DESCRIÇÃO	FUNCIONAL
RC01	PERMITIR O CONTROLE (INCLUIR/EXCLUIR/ALTERAR) DO CADASTRO DE PACIENTES	SIM
RC02	PERMITIR O CONTROLE (INCLUIR/EXCLUIR/ALTERAR) DO CADASTRO DE PROFISSIONAIS	SIM
RC03	PERMITIR O CONTROLE (INCLUIR/EXCLUIR/ALTERAR) DO CADASTRO DE FORNECEDORES	SIM
RC04	PERMITIR O CONTROLE (INCLUIR/EXCLUIR/ALTERAR) DO CADASTRO DE PRESTADORES DE SERVIÇO	SIM
RC05	FÁCIL NAVEGAÇÃO	NÃO
RC06	TRATAMENTO DE ERROS	SIM
RC07	PERMITIR O CONTROLE (INCLUIR/EXCLUIR/ALTERAR) DO CADASTRO DOS CONVÊNIOS	SIM
RC08	CONTROLAR INTERNAÇÃO DOS PACIENTES	SIM
RC09	TEMPO BAIXO DE RESPOSTA	NÃO
RC10	DISPONIBILIZAR NA WEB GRADE DE HORÁRIOS POR ESPECIALIDADE	SIM
RC11	PERMITIR AGENDAMENTO DE CONSULTAS E CIRURGIAS	SIM
RC12	CONTROLAR DISPONIBILIDADE DOS LEITOS	SIM
RC13	PERMITIR O CONTROLE (INCLUIR/EXCLUIR/ALTERAR) DE EQUIPES MÉDICAS	SIM
RC14	CONFIDENCIALIDADE DOS DADOS DOS PACIENTES	NÃO
RC15	O SISTEMA DEVE UTILIZAR O SGBD ORACLE 10g	NÃO
RC16	O SISTEMA SERÁ UTILIZADO NO SISTEMA OPERACIONAL *MICROSOFT WINDOWS XP*	NÃO
RC17	SEGURANÇA DOS DADOS ARMAZENADOS	NÃO

Figura 3.6 – Requisitos do Cliente
Fonte: o autor

Baseando-se nos requisitos do cliente (“o que”) a equipe da PE-Softools especificou os requisitos de projeto (“como”), ou seja, como cada requisito do cliente será atendido pelo *software*, como é descrito na figura 3.7.

REQUISITOS DE PROJETO - "COMO"		
REQ. CLIENTE	REQ. PROJETO	DESCRIÇÃO
RC01	RP01	FUNCIONALIDADE PARA CADASTRAR PACIENTES
RC02	RP02	FUNCIONALIDADE PARA CADASTRAR PROFISSIONAIS
RC03	RP03	FUNCIONALIDADE PARA CADASTRAR FORNECEDORES
RC03	RP04	INTEGRAÇÃO COM SISTEMA CONTÁBIL - CAD. FORNECEDORES
RC04	RP05	FUNCIONALIDADE PARA CADASTRAR PRESTADORES DE SERVIÇOS
RC04	RP06	INTEGRAÇÃO COM SISTEMA CONTÁBIL - CAD. PREST.DE SERVIÇOS
RC05	RP07	DISPONIBILIZAR TECLAS DE ATALHO EM DIVERSOS PONTOS DO SISTEMA
RC06	RP08	ROTINA PARA VERIFICAR CONSISTÊNCIA DA INFORMAÇÃO INSERIDA PELO USUÁRIO
RC07	RP09	FUNCIONALIDADE PARA CADASTRAR CONVÊNIOS
RC08	RP10	ROTINA PARA CONTROLAR INTERNAÇÃO DOS PACIENTES
RC09	RP11	TEMPO DE RESPOSTA ABAIXO DE 5 SEGUNDOS
RC10	RP12	MÓDULO WEB COM GRADE DOS HORÁRIOS POR ESPECIALIDADE MÉDICA
RC11	RP13	FUNCIONALIDADE PARA AGENDAR CONSULTAS E CIRURGIAS
RC12	RP14	FUNCIOLALIDADE PARA CONTROLE DE LEITOS
RC13	RP15	FUNCIONALIDADE PARA CADASTRAR EQUIPES MÉDICAS
RC14	RP16	ACESSOS AOS MÓDULOS E SUAS FUNCIONALIDADES PROTEGIDAS POR SENHAS INDIVIDUAIS
RC14	RP17	ROTINA DE AUDITORIA PARA MONITORAR ACESSOS DOS USUÁRIOS
RC15	RP18	DESENVOLVER CAMADA DE ACESSO A DADOS COMPATÍVEL AO ORACLE
RC16	RP19	DESENVOLVER O SISTEMA COMPATÍVEL COM O AMBIENTE *WINDOWS XP*
RC17	RP20	ROTINA DE BACKUP DOS DADOS E PLANO DE RECUPERAÇÃO DE FALHAS

Figura 3.7 – Requisitos de Projeto.
Fonte: o autor

Agora o cliente define qual o grau de importância de cada requisito (“o que”), valorizando de 1 a 5, e estabelece a matriz de relação, que traduz a sua percepção de como cada item “como” contribui com o atendimento do item “o que”. Realiza-se agora cálculo do peso absoluto, determinando-se assim quais os requisitos de projeto que possuem mais relação com o alcance da qualidade especificado pelo cliente, permitindo priorizar a função de trabalho com as necessidades a serem desenvolvidas, conforme é observado na figura 3.8.

		REQUISITOS DE PROJETO - "COMO"																			
REQUISITOS DO CLIENTE - "O QUE"	**GRAU DE IMPORTÂNCIA**	RP01	RP02	RP03	RP04	RP05	RP06	RP07	RP08	RP09	RP10	RP11	RP12	RP13	RP14	RP15	RP16	RP17	RP18	RP19	RP20
RC01	5	9							9		3	9		3			9	9	3		
RC02	5		9						9		1	3	3	3		9	9	9	3		
RC03	5			1	9				3			3					3	3	3		
RC04	5					1	9		3			1					3	3	3		
RC05	3							9													
RC06	4								9			1									
RC07	5								9	9		1					9	9	3		
RC08	4										9	9					3	1	3		
RC09	4											9							9		
RC10	2								1			1	9								
RC11	4								3			3		9			3	1	3		
RC12	5								3			3			9		3	1	3		
RC13	4								9			1				9	3	3	3		
RC14	5																9	9			
RC15	5																		9	3	9
RC16	5																			9	
RC17	5																		9	1	9
Peso Absoluto		45	45	5	45	5	45	27	266	45	56	194	33	66	45	81	261	235	252	65	90

Figura 3.8 – Planejamento do produto.
Fonte: o autor

Com base no peso absoluto, podemos identificar que os requisitos de projeto RP08 (Rotina para verificar consistência das informações inseridas pelo usuário) e RP16 (Acesso aos módulos e suas funcionalidades protegidos por senhas individuais) são os que estão mais relacionados com a qualidade, segundo a percepção do cliente. Observa-se também que o requisito RP04 (Integração com sistema contábil – Cad. Fornecedores) deve ser priorizado, em detrimento do RP03 (Funcionalidade para Cadastrar Fornecedores), o mesmo acontecendo com o RP06 (Integração com sistema contábil – Cad. Prestadores de Serviços) e RP05 (Funcionalidade para Cadastrar Prestadores de Serviços).

3.2.2. Simulação da fase 2 – Desdobramento das Partes ou Projeto do Produto

A simulação desta fase será realizada no módulo de controle de internação dos pacientes, esta função destina-se a registrar a entrada do paciente no hospital. No sistema Hosptools, os dados pessoais básicos dos pacientes (nome, filiação, endereço, etc.) são cadastrados sob a denominação de “prontuário”, ficando as informações disponíveis para todos os hospitais do GJC. A cada comparecimento do paciente aos hospitais, as informações inerentes à internação (data, médico, convênio, etc.) são registradas sob a denominação de “registros de entrada”. Desta maneira, quando o paciente é internado pela primeira vez, existe a necessidade de incluir um prontuário para o mesmo. Nas internações subseqüentes, após

localizar o prontuário do paciente no sistema, será necessário apenas acrescentar as demais informações do registro de entrada.

O controle de internação dos pacientes será desdobrado em:

- Prontuário do paciente
  - Código:
  - Nome:
  - Data de Nascimento
  - Filiação (mãe)
  - Filiação (pai)
  - Estado Civil
  - Cônjuge: Nome do cônjuge
  - Identidade
  - CPF
  - Tipo sangue
  - Sexo
  - Naturalidade
  - Nacionalidade
  - Endereço
  - Bairro
  - Cidade
  - UF
  - CEP
  - Telefone
  - E-mail
- Entrada
  - Registro de entrada: sequêncial identificador único da internação, gerado automaticamente pelo sistema;
  - Unidade: unidade em que o paciente está sendo atendido;
  - Prontuário: código do paciente;
  - Convênio: código do convênio do paciente;
  - Plano: código do plano do convênio;
  - Matrícula: matrícula (carteirinha) do paciente no convênio;

- Guia: número da guia fornecida pelo convênio;
- Validade: validade da guia fornecida pelo convênio;
- Senha: senha da guia fornecida pelo convênio.

- Atendimento
  - Código do médico
  - Admissão (data e hora)
  - Previsão de alta (data e hora)
  - Alta (data e hora da alta)

Após o desdobramento, os requisitos de sistema são modelados e documentados utilizando o modelo de casos de uso.

- Diagrama de Caso de Uso – Realizar Entrada (figura 3.9):

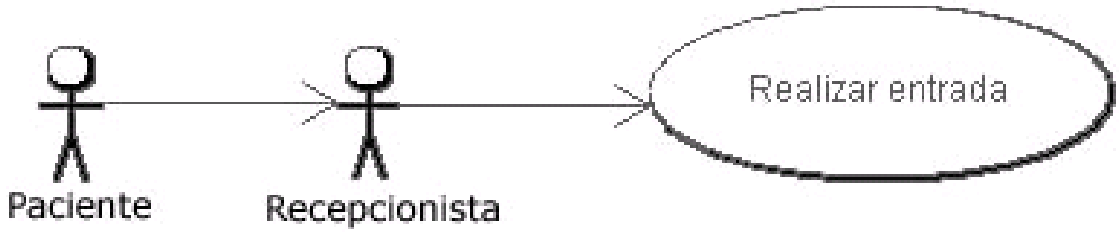


Figura 3.9 – Caso de uso realizar entrada.
Fonte: o autor

- Descrição do Diagrama de Caso de Uso – Realizar Entrada
  - **Descrição**
    Este caso de uso é responsável por realizar a entrada do paciente no hospital. Com os dados pessoais básicos dos pacientes (nome, filiação, endereço,...) já estão cadastrados, prontuário médico, nas internações subseqüentes será necessário apenas acrescentar as demais informações do registro de entrada (unidade em que o paciente está sendo atendido, convênio, plano do convênio, número da guia fornecida pelo convênio, ...).

  - **Pré condições**
    - A recepcionista tem que estar cadastrada no sistema hospitalar.
    - A recepcionista deve estar logada ao sistema.
    - O prontuário médico do paciente tem que estar cadastrado no sistema.

  - **Pós condições**
    - O registro de entrada do paciente deve estar registrado no sistema.

- **Fluxo de eventos principal**
  - 1. A recepcionista deve localizar o prontuário médico do paciente.
  - 2. A recepcionista deve preencher as informações necessárias para a entrada do paciente.
  - 3. O sistema apresenta uma mensagem solicitando a confirmação dos dados digitados pela recepcionista.
  - 4. Uma mensagem indicando o status da operação é apresentada à recepcionista.

- **Fluxos secundários**
  - No passo 1, se não conseguir localizar o prontuário médico, a recepcionista deverá realizar primeiramente a inclusão do prontuário.
  - No passo 2, caso não haja conformidade nas informações do sistema com as fornecidas pelo paciente, a recepcionista deverá atualizar as informações.
  - A qualquer momento a recepcionista pode cancelar a operação.

- Diagrama de Classes (de análise) – Realizar Entrada (figura 3.10):

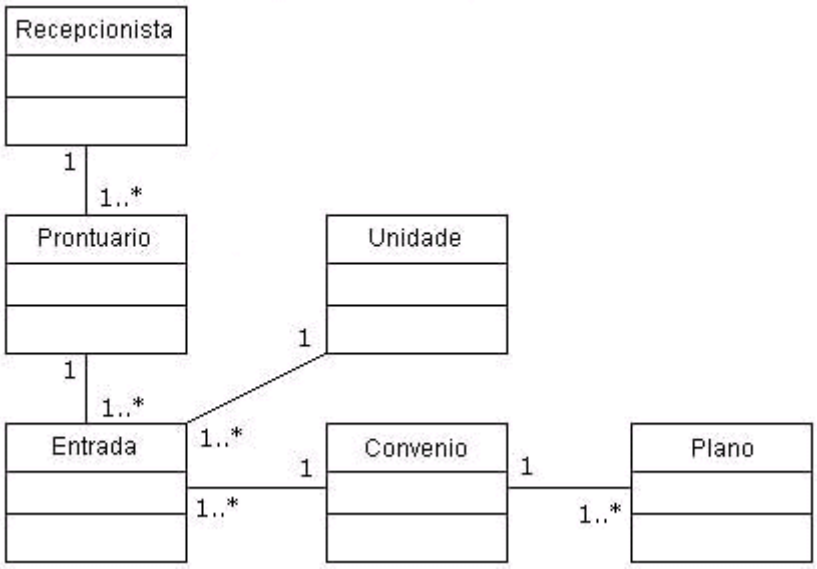


Figura 3.10 – Diagrama de Classes (de análise) realizar entrada.
Fonte: o autor

- Diagrama de Classes (de projeto) – Realizar Entrada (figura 3.11):

Figura 3.11 – Diagrama de Classes (de projeto) realizar entrada.
Fonte: o autor

3.2.3. Simulação da fase 3 – Planejamento do Processo

Como será necessário disponibilizar um módulo (RC10 – RP12) com interface *web*, optou-se por utilizar a arquitetura baseada em três camadas.

Para o desenvolvimento do *software* foi escolhida a linguagem de programação C# (.net) da Microsoft, tal escolha é justificada levando-se em consideração os seguintes aspectos:

- Aderente a arquitetura selecionada;
- Compatível com o SGBD exigido (RC15) pelo cliente.
    - O GJC possui licenças do SGBD Oracle 10g e também administradores de bancos de dados (DBA's) capacitados no uso da ferramenta;
        - Sem custo de aquisição e treinamento;
- Compatível com o sistema operacional exigido (RC16) pelo cliente;
    - Todo o parque computacional instalado no GJC utiliza a plataforma *Microsoft Windows XP*;
        - Sem custo de aquisição e treinamento;
- Domínio e *expertise* no uso da linguagem de programação pela Equipe da PE-Softools.

Abaixo são descritos os principais recursos que serão utilizados durante o desenvolvimento do *software*:

- Profissionais
    - Gerente de projeto;
    - Analista de sistema;
    - Programador;
    - Administrador de bancos de dados (DBA);
    - Arquiteto de sistema;
    - Analista de teste;
- *Hardware*
    - Computador;
    - Impressora,
- *Software*
    - Linguagem de programação;
    - Sistema gerenciador de banco de dados (SGBD);
    - Editor de texto;
    - Planilha eletrônica;
    - Sistema operacional.

3.2.4. Simulação da fase 4 – Planejamento da Produção

A coordenação da equipe como também todo o monitoramento e controle desta fase, ficou sob a responsabilidade do gerente de projeto da PE-Softools, que utilizou como ferramenta de apoio o *Microsoft Office Project*.

Objetivando facilitar o controle e acompanhamento das atividades, o gerente de projeto optou por dividir o processo de desenvolvimento do *software* em três etapas, onde as atividades foram alocadas a cada uma das fases, conforme descrito abaixo:

- Análise dos requisitos (figura 3.12).

Figura 3.12 – Atividades da fase análise.
Fonte: o autor

- Desenvolvimento dos requisitos (figura 3.13).

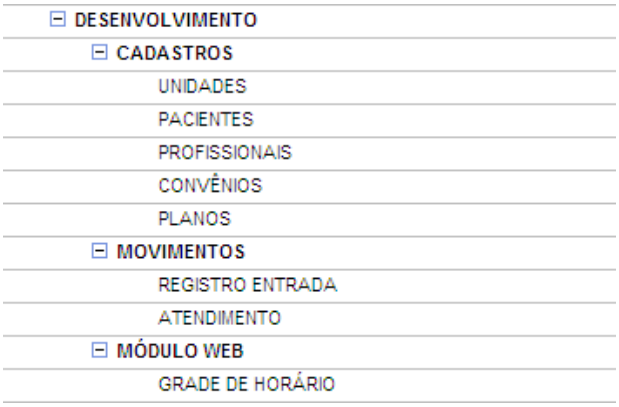


Figura 3.13 – Atividades da fase desenvolvimento.
Fonte: o autor

- Homologação dos requisitos (figura 3.14).

⊟ HOMOLOGAÇÃO
⊟ CADASTROS
UNIDADES
PACIENTES
PROFISSIONAIS
CONVÊNIOS
PLANOS
⊟ MOVIMENTOS
REGISTRO ENTRADA
ATENDIMENTO
⊟ MÓDULO WEB
GRADE DE HORÁRIO

Figura 3.14 – Atividades da fase homologação.
Fonte: o autor

Após fixadas as fases e atividades, a equipe definiu em qual sequência deve se realizar cada uma delas. Optou-se por iniciar a fase de desenvolvimento apenas após o término da fase de análise, como também só iniciar a fase de homologação após finalizar todas as atividades da fase de desenvolvimento, buscando com isso deixar "folgas" para possíveis manobras caso existam atrasos nas atividades. O seqüenciamento das fases é demonstrado na figura 3.15.

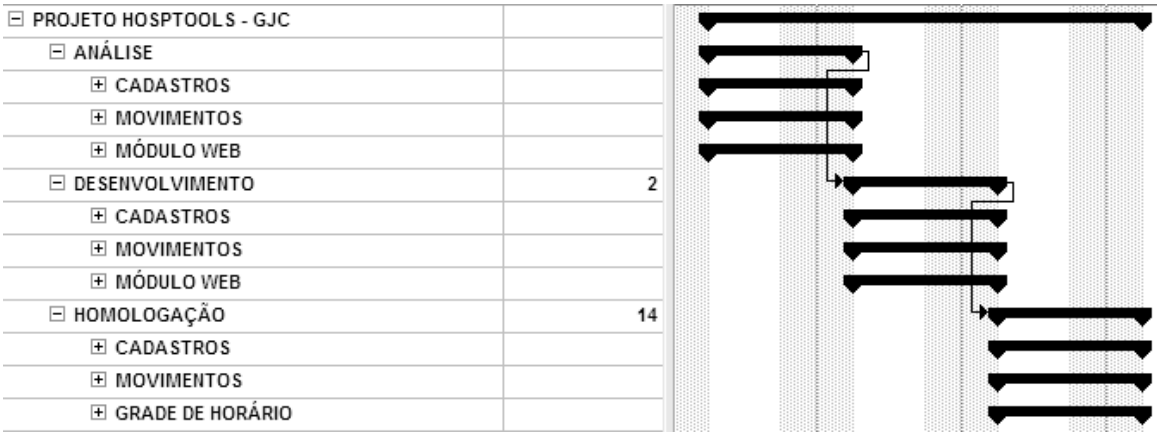


Figura 3.15 – Sequenciamento das fases.
Fonte: o autor

Logo em seguida a equipe definiu o seqüenciamento das atividades de cada fase, como é demonstrado nos gráficos de barras (figuras 3.16, 3.17 e 3.18).

- Seqüenciamento das atividades da fase de análise (figura 3.16);

ANÁLISE	
CADASTROS	
UNIDADES	
PACIENTES	
PROFISSIONAIS	
CONVÊNIOS	
PLANOS	7
MOVIMENTOS	
REGISTRO ENTRADA	3
ATENDIMENTO	10
MÓDULO WEB	
GRADE DE HORÁRIO	3

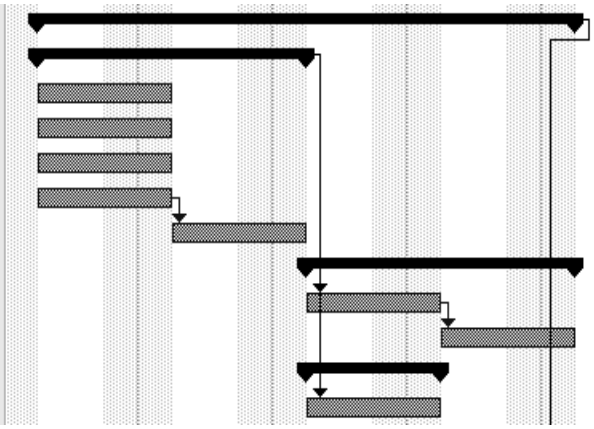


Figura 3.16 – Sequenciamento das atividades da fase de análise.
Fonte: o autor

- Seqüenciamento das atividades da fase desenvolvimento (figura 3.17);

DESENVOLVIMENTO	2
CADASTROS	
UNIDADES	
PACIENTES	
PROFISSIONAIS	
CONVÊNIOS	
PLANOS	19
MOVIMENTOS	
REGISTRO ENTRADA	15
ATENDIMENTO	22
MÓDULO WEB	
GRADE DE HORÁRIO	15

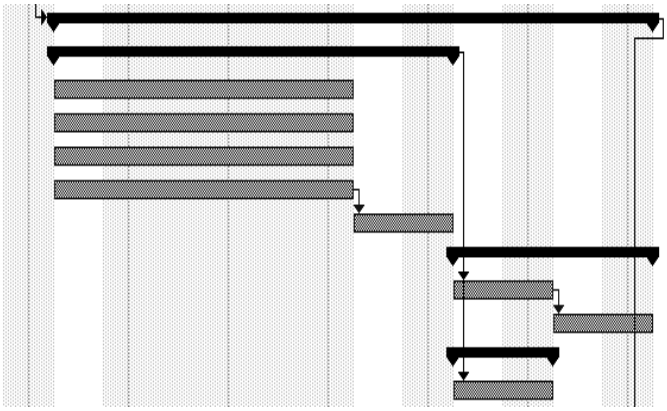


Figura 3.17 – Sequenciamento das atividades da fase de desenvolvimento.
Fonte: o autor

- Seqüenciamento das atividades da fase de homologação (figura 3.18);

GRADE DE HORÁRIO	15
HOMOLOGAÇÃO	14
CADASTROS	
UNIDADES	
PACIENTES	
PROFISSIONAIS	
CONVÊNIOS	
PLANOS	31
MOVIMENTOS	
REGISTRO ENTRADA	27
ATENDIMENTO	34
MÓDULO WEB	
GRADE DE HORÁRIO	27

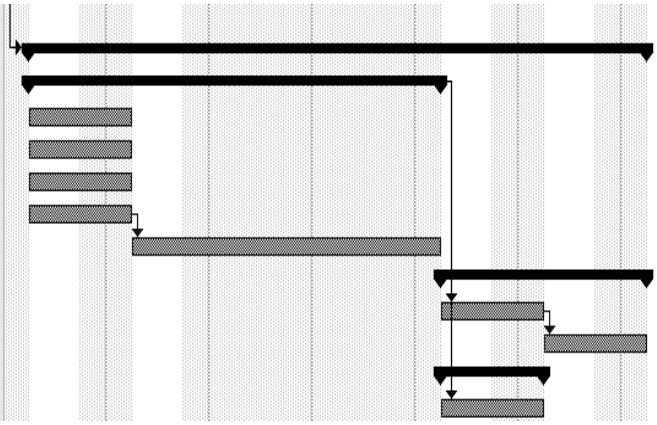


Figura 3.18 – Sequenciamento das atividades da fase de homologação.
Fonte: o autor

Nesta etapa a equipe estimou o esforço para realização de cada atividade, como pode ser observado na figura 3.19.

ANÁLISE	**15 dias**
CADASTROS	**8 dias**
UNIDADES	16 hrs
PACIENTES	32 hrs
PROFISSIONAIS	32 hrs
CONVÊNIOS	40 hrs
PLANOS	24 hrs
MOVIMENTOS	**7 dias**
REGISTRO ENTRADA	32 hrs
ATENDIMENTO	24 hrs
MÓDULO WEB	**5 dias**
GRADE DE HORÁRIO	40 hrs
DESENVOLVIMENTO	**11 dias**
CADASTROS	**6 dias**
UNIDADES	16 hrs
PACIENTES	24 hrs
PROFISSIONAIS	24 hrs
CONVÊNIOS	32 hrs
PLANOS	16 hrs
MOVIMENTOS	**5 dias**
REGISTRO ENTRADA	24 hrs
ATENDIMENTO	16 hrs
MÓDULO WEB	**5 dias**
GRADE DE HORÁRIO	40 hrs
HOMOLOGAÇÃO	**10 dias**
CADASTROS	**5 dias**
UNIDADES	16 hrs
PACIENTES	16 hrs
PROFISSIONAIS	16 hrs
CONVÊNIOS	24 hrs
PLANOS	16 hrs
MOVIMENTOS	**5 dias**
REGISTRO ENTRADA	24 hrs
ATENDIMENTO	16 hrs
MÓDULO WEB	**3 dias**
GRADE DE HORÁRIO	24 hrs

Figura 3.19 – Estimativa de duração das atividades
Fonte: o autor

Após a estimativa de duração das atividades, foram realizadas as estimativas de custos das fases, como também o custo total do projeto. As estimativas de custos são demonstradas na figura 3.20.

ANÁLISE				
RECUSRSO	ALOCAÇÃO	VALOR/HORA	HORAS	CUSTO (R$)
GERENTE DE PROJETO	30%	80,00	240	5.760,00
ANALISTA DE SISTEMA	100%	75,00	240	18.000,00
ARQUITETO DE SISTEMA	40%	75,00	240	7.200,00
DBA	20%	70,00	240	3.360,00
PROGRAMADOR	0%	60,00	240	0,00
ANALISTA DE TESTE	0%	45,00	240	0,00
			TOTAL (R$)	34.320,00

DESENVOLVIMENTO				
RECUSRSO	ALOCAÇÃO	VALOR/HORA	HORAS	CUSTO (R$)
GERENTE DE PROJETO	30%	80,00	184	4.416,00
ANALISTA DE SISTEMA	30%	75,00	184	4.140,00
ARQUITETO DE SISTEMA	0%	75,00	184	0,00
DBA	20%	70,00	184	2.576,00
PROGRAMADOR	100%	60,00	184	11.040,00
ANALISTA DE TESTE	0%	45,00	184	0,00
			TOTAL (R$)	22.172,00

HOMOLOGAÇÃO				
RECUSRSO	ALOCAÇÃO	VALOR/HORA	HORAS	CUSTO (R$)
GERENTE DE PROJETO	30%	80,00	88	2.112,00
ANALISTA DE SISTEMA	0%	75,00	88	0,00
ARQUITETO DE SISTEMA	0%	75,00	88	0,00
DBA	0%	70,00	88	0,00
PROGRAMADOR	10%	60,00	88	528,00
ANALISTA DE TESTE	100%	45,00	88	3.960,00
			TOTAL (R$)	6.600,00

PROJETO	
FASE	CUSTO (R$)
ANÁLISE	34.320,00
DESENVOLVIMENTO	22.172,00
HOMOLOGAÇÃO	6.600,00
TOTAL (R$)	63.092,00

Figura 3.20 – Estimativa de custo.
Fonte: o autor

Logo após realizada a estimativa de custos, os recursos que serão empregados no desenvolvimento do *software* são atribuídos a cada atividade, como é demonstrado na figura 3.21.

⊟ PROJETO HOSPTOOLS - GJC	32 dias	Gerente de Projeto[30%]
⊟ ANÁLISE	15 dias	DBA[20%];Arquiteto [40%]
⊟ CADASTROS	8 dias	
UNIDADES	16 hrs	Analista 1
PACIENTES	32 hrs	Analista 2
PROFISSIONAIS	32 hrs	Analista 3
CONVÊNIOS	40 hrs	Analista 4
PLANOS	24 hrs	Analista 4
⊟ MOVIMENTOS	7 dias	
REGISTRO ENTRADA	32 hrs	Analista 1
ATENDIMENTO	24 hrs	Analista 2
⊟ MÓDULO WEB	5 dias	
GRADE DE HORÁRIO	40 hrs	Analista 4
⊟ DESENVOLVIMENTO	11 dias	Analista 1[30%];DBA[20%]
⊟ CADASTROS	6 dias	
UNIDADES	16 hrs	Programador 1
PACIENTES	24 hrs	Programador 2
PROFISSIONAIS	24 hrs	Programador 3
CONVÊNIOS	32 hrs	Programador 4
PLANOS	16 hrs	Programador 4
⊟ MOVIMENTOS	5 dias	
REGISTRO ENTRADA	24 hrs	Programador 4
ATENDIMENTO	16 hrs	Programador 4
⊟ MÓDULO WEB	3,94 dias	
GRADE DE HORÁRIO	32 hrs	Programador 1
⊟ HOMOLOGAÇÃO	6 dias	Programador 1[10%]
⊟ CADASTROS	3 dias	
UNIDADES	8 hrs	Analista de Teste 1
PACIENTES	8 hrs	Analista de Teste 2
PROFISSIONAIS	8 hrs	Analista de Teste 1
CONVÊNIOS	16 hrs	Analista de Teste 1
PLANOS	8 hrs	Analista de Teste 2
⊟ MOVIMENTOS	3 dias	
REGISTRO ENTRADA	16 hrs	Analista de Teste 1
ATENDIMENTO	8 hrs	Analista de Teste 2
⊟ MÓDULO WEB	2 dias	
GRADE DE HORÁRIO	16 hrs	Analista de Teste 3

Figura 3.21 – Alocação dos recursos.
Fonte: o autor

Agora é possível determinar qual a data de início e término de cada atividade, tendo em vista que os recursos, as durações e as dependências já foram estabelecidas, ou seja, é possível determinar o cronograma (figura 3.22) para o desenvolvimento do *software*.

PROJETO HOSPTOOLS - GJC	**32 dias**	**Seg 03/11/08**	**Ter 16/12/08**	**Gerente de Projeto[30%]**
ANÁLISE	**15 dias**	**Seg 03/11/08**	**Seg 24/11/08**	**DBA[20%];Arquiteto [40%]**
CADASTROS	**8 dias**	**Seg 03/11/08**	**Qui 13/11/08**	
UNIDADES	16 hrs	Seg 03/11/08	Qua 05/11/08	Analista 1
PACIENTES	32 hrs	Seg 03/11/08	Sex 07/11/08	Analista 2
PROFISSIONAIS	32 hrs	Seg 03/11/08	Sex 07/11/08	Analista 3
CONVÊNIOS	40 hrs	Seg 03/11/08	Seg 10/11/08	Analista 4
PLANOS	24 hrs	Seg 10/11/08	Qui 13/11/08	Analista 4
MOVIMENTOS	**7 dias**	**Qui 13/11/08**	**Seg 24/11/08**	
REGISTRO ENTRADA	32 hrs	Qui 13/11/08	Qua 19/11/08	Analista 1
ATENDIMENTO	24 hrs	Qua 19/11/08	Seg 24/11/08	Analista 2
MÓDULO WEB	**5 dias**	**Qui 13/11/08**	**Qui 20/11/08**	
GRADE DE HORÁRIO	40 hrs	Qui 13/11/08	Qui 20/11/08	Analista 4
DESENVOLVIMENTO	**11 dias**	**Seg 24/11/08**	**Seg 08/12/08**	**Analista 1[30%];DBA[20%]**
CADASTROS	**6 dias**	**Seg 24/11/08**	**Seg 01/12/08**	
UNIDADES	16 hrs	Seg 24/11/08	Qua 26/11/08	Programador 1
PACIENTES	24 hrs	Seg 24/11/08	Qui 27/11/08	Programador 2
PROFISSIONAIS	24 hrs	Seg 24/11/08	Qui 27/11/08	Programador 3
CONVÊNIOS	32 hrs	Seg 24/11/08	Qui 27/11/08	Programador 4
PLANOS	16 hrs	Sex 28/11/08	Seg 01/12/08	Programador 4
MOVIMENTOS	**5 dias**	**Ter 02/12/08**	**Seg 08/12/08**	
REGISTRO ENTRADA	24 hrs	Ter 02/12/08	Qui 04/12/08	Programador 4
ATENDIMENTO	16 hrs	Sex 05/12/08	Seg 08/12/08	Programador 4
MÓDULO WEB	**3,94 dias**	**Ter 02/12/08**	**Sex 05/12/08**	
GRADE DE HORÁRIO	32 hrs	Ter 02/12/08	Sex 05/12/08	Programador 1
HOMOLOGAÇAO	**6 dias**	**Ter 09/12/08**	**Ter 16/12/08**	**Programador 1[10%]**
CADASTROS	**3 dias**	**Ter 09/12/08**	**Qui 11/12/08**	
UNIDADES	8 hrs	Ter 09/12/08	Ter 09/12/08	Analista de Teste 1
PACIENTES	8 hrs	Ter 09/12/08	Ter 09/12/08	Analista de Teste 2
PROFISSIONAIS	8 hrs	Ter 09/12/08	Ter 09/12/08	Analista de Teste 1
CONVÊNIOS	16 hrs	Ter 09/12/08	Qua 10/12/08	Analista de Teste 1
PLANOS	8 hrs	Sex 12/12/08	Qui 11/12/08	Analista de Teste 2
MOVIMENTOS	**3 dias**	**Ter 16/12/08**	**Ter 16/12/08**	
REGISTRO ENTRADA	16 hrs	Ter 16/12/08	Seg 15/12/08	Analista de Teste 1
ATENDIMENTO	8 hrs	Qua 31/12/08	Ter 16/12/08	Analista de Teste 2
MÓDULO WEB	**2 dias**	**Ter 16/12/08**	**Seg 15/12/08**	
GRADE DE HORÁRIO	16 hrs	Ter 16/12/08	Seg 15/12/08	Analista de Teste 3

Figura 3.22 – Cronograma do desenvolvimento do software.
Fonte: o autor

A partir deste momento o gerente de projeto dispõe dos insumos necessários para monitorar e controlar a produção do *software*, buscando assim garantir que o sistema será produzido conforme planejado, contém o que foi especificado pelo cliente, será entregue no prazo acordado e dentro do custo orçado.

4. CONCLUSÕES

4.1. Considerações finais

Após o que foi apresentado no decorrer deste estudo, é possível avaliar que o QFD é uma metodologia específica e eficiente para ouvir o que dizem os clientes, possibilitando descobrir exatamente o que eles querem, permitindo também determinar a melhor maneira de satisfazer tais necessidades. Através do QFD os desejos e necessidades dos clientes são captados, avaliando também as características do produto que será desenvolvido ou aperfeiçoado, suportando o projeto do produto durante todo o ciclo, desde a fase de identificação da necessidade, passando pela especificação e documentação dos requisitos até o controle e monitoramento da produção. Isto significa traduzir as necessidades dos clientes em requisitos técnicos apropriados, suportando as diversas fases do projeto de desenvolvimento ou aperfeiçoamento do *software*.

Por outro lado, um projeto de *software*, que inicialmente poderia ser considerado simples, quando mal planejado, pode produzir uma diversidade de problemas. É neste sentido que se percebe a importância da Engenharia de *Software*, que irá permitir que o processo de desenvolvimento do *software* seja disciplinado como também bem executado.

Conforme visto no modelo proposto, o QFD suportará o projeto durante todas as fases do desenvolvimento do *software*, permitindo que o sistema seja elaborado e construído visando atender a qualidade projetada para o produto final.

A qualidade exigida pelo cliente deve, portanto, ser desdobrada ao longo do detalhamento do projeto. Para tal, a comunicação entre as diversas áreas envolvidas nos processos que serão suportados pelo *software*, que neste caso específico é o processo de gestão hospitalar, como também a participação e o engajamento de representantes das áreas usuárias é de fundamental importância.

O emprego do modelo conceitual proposto proporcionou, antes de tudo, uma melhor visibilidade do escopo do *software* a ser desenvolvido, além de demonstrar ser um meio prático e eficaz de compreender o que deveria ser conduzido em cada etapa do processo e como realizá-lo

Sendo assim, a construção do modelo conceitual apresentado no Capítulo 3 (Modelo Proposto), refere-se ao detalhamento de um protótipo para suportar o desenvolvimento de

um produto, considerando todo o processo de produção. Através do QFD e da análise das suas relações com a qualidade exigida pelo cliente, e com os mecanismos ou processos necessários para alcançá-las, pode-se realmente conhecer e priorizar os principais requisitos. A identificação destes requisitos é a ponte entre a voz do cliente e a Engenharia de *Software*.

Portanto, diante do que foi apresentado nesta dissertação, chega-se a conclusão que o uso da metodologia QFD, cria um plano de ação que busca garantir que os produtos desenvolvidos ou aperfeiçoados tenham as características exigidas pelos clientes.

4.2. Recomendações para estudos futuros

Para uma perspectiva futura, avaliamos o QFD como um método de desenvolvimento capaz de elaborar produtos mais atrativos, motivados no planejamento estratégico do produto, observando fatores que podem trazer diferenciais competitivos para a organização, identificando o que os clientes querem hoje (necessidades) e o que esperam para o amanhã (desejo), como também comparando suas características funcionais com as dos produtos de mesma finalidade existentes no mercado. Para Engenharia de *Software*, é esperado que o QFD seja considerado um método adequado, voltado para o aperfeiçoamento dos processos de produção, garantindo a qualidade durante todo o ciclo de desenvolvimento do *software*.

Abaixo são apresentadas algumas sugestões para trabalhos futuros:

- Realizar a aplicação do estudo em outras organizações, e também em outros setores, objetivando o aprimoramento dos métodos utilizados, como também o seu desenvolvimento e consolidação;
- Elaborar pesquisa demonstrando os benefícios que as organizações podem obter a partir da adoção do modelo simulado de adaptação do QFD.

REFERÊNCIAS

AKAO, Y. Manual de aplicação do desdobramento da função qualidade. Belo Horizonte: Fundação Christiano Ottoni, 1994.

_____.History of Quality Function Deployment in Japan. In: The best on quality. Hanser Publishers, 1990.

BARROS, A. J. P.; LEHFELD, N. A. S. Fundamentos de metodologia: um guia para a iniciação científica. 2. ed. São Paulo: Makron Books, 2000.

BATISTA, E. A.; CAVARLHO, A. M. B.R. Uma Taxonomia Facetada para Técnicas de Elicitação de Requisitos Artigos publicados no WER. Instituto de Computação – UNICAMP 2004. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER03/edinelson_batista.pdf>. Acesso em 28 out. 2008.

BOEHM, B. A View of 20th and 21st Century Software Engineering. ICSE'06, 2006.

CHENG, L. C. Ampliando a vantagem competitiva através do QFD: Informe da Qualidade Total. Fundação Christiano Ottoni, Belo Horizonte, ano 04, n. 14, p. 2, maio/setembro, 1995.

COCKBURN, A. Writing effective: use cases. Addison - Wesley, USA, 2001.

EUREKA, W. E.; RYAN, N. E. QFD: Perspectivas gerenciais no desdobramento da função qualidade. Qualitymark. Rio de Janeiro, 1992.

FEIGENBAUM, A. V. Controle da Qualidade Total. São Paulo: Makron Books, 1994.

FERNANDES, J. M. R.; REBELATO, M. G.. Proposta de um método para integração entre QFD e FMEA. Gestão da Produção, São Carlos, v. 13, n. 2, 2006.

GARCEZ, A. S. A voz do cliente. Disponível em: <http:// www.performance.eti.br/AVOZ do CLIENTE. DOC.>. Acesso em: 01 out. 2008.

GODOY, J. M de. Casos reais de implantação de TQC: gerenciamento pelas diretrizes, programa 5 S e gerenciamento da rotina. Fundação Christiano Ottoni, São Paulo, v. 01, 1994.

GUAZZI, D. M. Utilização do QFD como ferramenta de melhoria contínua do grau de satisfação de clientes internos: uma aplicação em cooperativas agropecuárias. Florianópolis, 1999. 226 p. (Doutorado – Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina).

HAUSER, J.; CLAUSING, D. The house of quality. Harvard Business Review. USA, 1988.

JURAN, J. M.; GRYNA, M. F. Controle da qualidade. São Paulo: Makron Books, 1991.

JURAN, J. M. A qualidade desde o projeto: novos passos para o planejamento da qualidade em produtos e serviços. São Paulo: Pioneira, 1997.

KIENITZ, H. O. Proposta de implementação da metodologia do Quality Function Deployment na Mercedes-Benz do Brasil S.A. São Carlos, 1995. 335 p. (Mestrado – Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de São Carlos).

KOTONYA, G.; SOMMERVILLE, I. Requirements engineering: processes and techniques. John Wiley & Soons. England, 1997.

LAUDON, K. C.; LAUDON, J. P Management information System. A contemporary perspective. MacMillan. New York, 1996.

LINS, B. F. E. Ferramentas básicas da qualidade. Ci.Inf., Brasília, v. 22, n. 2, p.153 -161, mai./ago., 1993.

MEDEIROS, E. Desenvolvendo software com UML 2.0: definitivo. São Paulo: Pearson Makron Books, 2004.

MIGUEL, P. A. C. Qualidade: enfoques e ferramentas. São Paulo: Artliber Editora, 2001.

MIZUNO, S. Qualidade total na escola: fundamentos & implantação. Belo Horizonte: Pitágoras TEC, 1994.

MIZUNO, S.; AKAO, Y. Quality Function Deployment. JUSE. USA, 1994.

NORTON, P. Introdução à informática. São Paulo: Makron Books, 1996.

OHFJI, T. Método de desdobramento da qualidade. Belo Horizonte: Fundação Christiano Ottoni, Escola de Engenharia da UFMG, 256p., 1997

PAULA FILHO, W. de P. Engenharia de software: fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2003.

PENDER, T. UML: A Bíblia. Rio de Janeiro: Elsevier, 2004.

PRESSMAN, R. Engenharia de software. São Paulo: McGraw-Hill, 2006.

ROESCH, S. M. A. Projetos de estágios e de pesquisa em administração. São Paulo: Atlas, 1999.

RUMBAUGH, J.; BLAHA, M. Modelagem e projetos baseados em objetos. São Paulo: Campus, 2006.

SIMÕES, C. A. A metodologia QFD. Revista Controle de Qualidade. São Paulo, n. 37, p. 34-38, jun., 1995.

SOUZA, D. L. S. Ferramentas de Gestão de Tecnologia: Um Diagnóstico de Utilização nas Pequenas e Médias Empresas Industriais da Região de Curitiba. Paraná, 2003. 120p. (Mestrado - Programa de Pós-Graduação em Tecnologia do Centro Federal de Educação Tecnológica do Paraná).

STAIR, R. M.; REYNOLDS, G.W. Princípios de Sistemas de Informação. Rio de Janeiro: LTC, 20002.

SOMMERVILLE, I. Engenharia de Software. 6.ed. São Paulo: Addison Wesley, 2003.

SOMMERVILLE, I.; SAWYER, P. Requirements Engineering – a good practice guide. John Wiley & Sons Ltd. New York, 1997.

VIANNA, C. S. M. Software e privacidade: uma defesa do código-fonte aberto na preservação do direito constitucional à vida privada. Jus Navigandi, Teresina, ano 6, n. 57, jul. 2002. Disponível em: <http://jus2.uol.com.br/doutrina/texto.asp?id=2931>. Acesso em: 02 nov. 2008.

WAZLAWICK, R. S. Análise e projeto de sistemas de informação orientados a objetos. Rio de Janeiro: Elsevier, 2004.

WHITELEY, R. C. A empresa totalmente voltada para o cliente: do planejamento à ação. Rio de Janeiro: Campos, 1992.

YIN, R. Estudo de caso: planejamento e métodos. 2.ed. Porto Alegre: Bookman, 2001.